CHAPTER 1

INTRODUCTION

The evolution of digital circuit design

n

Compelling issues in digital circuit design

n

How to measure the quality of a design

n

Valuable references

1.1 A Historical Perspective

1.2 Issues in Digital Integrated Circuit Design

1.3 Quality Metrics of a Digital Design

1.4 Summary

1.5 To Probe Further

1.1 A Historical Perspective

The concept of digital data manipulation has made a dramatic impact on our society. One has long grown accustomed to the idea of digital computers. Evolving steadily from mainframe and minicomputers, personal and laptop computers have proliferated into daily life. More significant, however, is a continuous trend towards digital solutions in all other areas of electronics. Instrumentation was one of the first noncomputing domains where the potential benefits of digital data manipulation over analog processing were recognized. Other areas such as control were soon to follow. Only recently have we witnessed the conversion of telecommunications and consumer electronics towards the digital format. Increasingly, telephone data is transmitted and processed digitally over both wired and wireless networks. The compact disk has revolutionized the audio world, and digital video is following in its footsteps.

The idea of implementing computational engines using an encoded data format is by no means an idea of our times. In the early nineteenth century, Babbage envisioned large-scale mechanical computing devices, called *Difference Engines* [Swade93]. Although these engines use the decimal number system rather than the binary representation now common in modern electronics, the underlying concepts are very similar. The Analytical Engine, developed in 1834, was perceived as a general-purpose computing machine, with features strikingly close to modern computers. Besides executing the basic repertoire of operations (addition, subtraction, multiplication, and division) in arbitrary sequences, the machine operated in a two-cycle sequence, called “store” and “mill” (execute), similar to current computers. It even used pipelining to speed up the execution of the addition operation! Unfortunately, the complexity and the cost of the designs made the concept impractical. For instance, the design of Difference Engine I (part of which is shown in Figure 1.1) required 25,000 mechanical parts at a total cost of £17,470 (in 1834!).

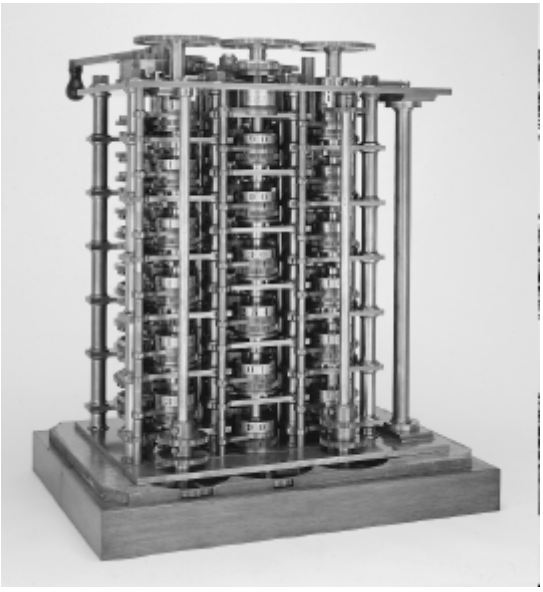


Figure 1.1 Working part of Babbage’s Difference Engine I (1832), the first known automatic calculator (from [Swade93], courtesy of the Science Museum of London).

The electrical solution turned out to be more cost effective. Early digital electronics systems were based on magnetically controlled switches (or relays). They were mainly used in the implementation of very simple logic networks. Examples of such are train safety systems, where they are still being used at present. The age of digital electronic computing only started in full with the introduction of the vacuum tube. While originally used almost exclusively for analog processing, it was realized early on that the vacuum tube was useful for digital computations as well. Soon complete computers were realized. The era of the vacuum tube based computer culminated in the design of machines such as the ENIAC (intended for computing artillery firing tables) and the UNIVAC I (the first successful commercial computer). To get an idea about *integration density*, the ENIAC was 80 feet long, 8.5 feet high and several feet wide and incorporated 18,000 vacuum tubes. It became rapidly clear, however, that this design technology had reached its limits. Reliability problems and excessive power consumption made the implementation of larger engines economically and practically infeasible.

All changed with the invention of the *transistor* at Bell Telephone Laboratories in 1947 [Bardeen48], followed by the introduction of the bipolar transistor by Schockley in 1949 [Schockley49][1]. It took till 1956 before this led to the first bipolar digital logic gate, introduced by Harris [Harris56], and even more time before this translated into a set of integrated-circuit commercial logic gates, called the Fairchild Micrologic family [Norman60]. The first truly successful IC logic family, *TTL (Transistor-Transistor Logic)* was pioneered in 1962 [Beeson62]. Other logic families were devised with higher performance in mind. Examples of these are the current switching circuits that produced the first subnanosecond digital gates and culminated in the *ECL (Emitter-Coupled Logic)* family [Masaki74]. TTL had the advantage, however, of offering a higher integration density and was the basis of the first integrated circuit revolution. In fact, the manufacturing of TTL components is what spear-headed the first large semiconductor companies such as Fairchild, National, and Texas Instruments. The family was so successful that it composed the largest fraction of the digital semiconductor market until the 1980s.

Ultimately, bipolar digital logic lost the battle for hegemony in the digital design world for exactly the reasons that haunted the vacuum tube approach: the large power consumption per gate puts an upper limit on the number of gates that can be reliably integrated on a single die, package, housing, or box. Although attempts were made to develop high integration density, low-power bipolar families (such as *I^2L—Integrated Injection Logic* [Hart72]), the torch was gradually passed to the MOS digital integrated circuit approach.

The basic principle behind the MOSFET transistor (originally called IGFET) was proposed in a patent by J. Lilienfeld (Canada) as early as 1925, and, independently, by O. Heil in England in 1935. Insufficient knowledge of the materials and gate stability problems, however, delayed the practical usability of the device for a long time. Once these were solved, MOS digital integrated circuits started to take off in full in the early 1970s. Remarkably, the first MOS logic gates introduced were of the CMOS variety [Wanlass63], and this trend continued till the late 1960s. The complexity of the manufacturing process delayed the full exploitation of these devices for two more decades. Instead,

[1] An intriguing overview of the evolution of digital integrated circuits can be found in [Murphy93]. (Most of the data in this overview has been extracted from this reference). It is accompanied by some of the historically ground-breaking publications in the domain of digital IC's.

the first practical MOS integrated circuits were implemented in PMOS-only logic and were used in applications such as calculators. The second age of the digital integrated circuit revolution was inaugurated with the introduction of the first microprocessors by Intel in 1972 (the 4004) [Faggin72] and 1974 (the 8080) [Shima74]. These processors were implemented in NMOS-only logic, which has the advantage of higher speed over the PMOS logic. Simultaneously, MOS technology enabled the realization of the first high-density semiconductor memories. For instance, the first 4Kbit MOS memory was introduced in 1970 [Hoff70].

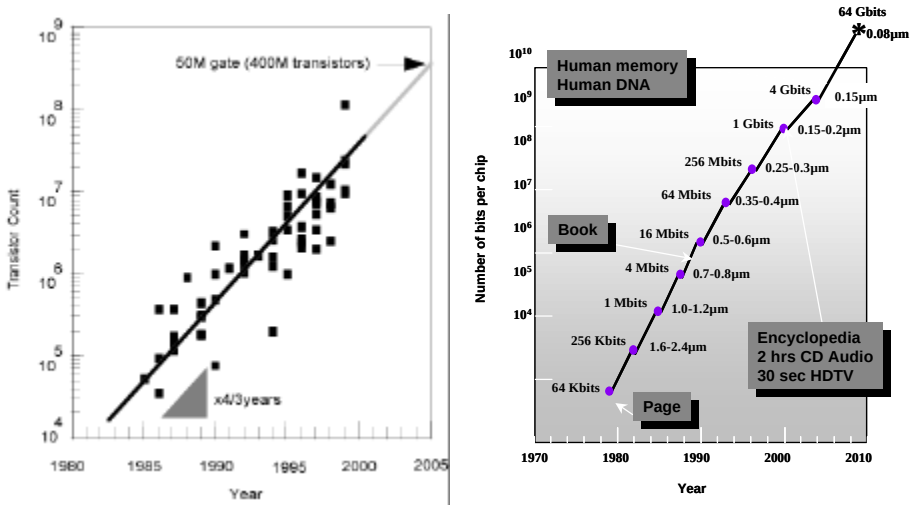
These events were at the start of a truly astounding evolution towards ever higher integration densities and speed performances, a revolution that is still in full swing right now. The road to the current levels of integration has not been without hindrances, however. In the late 1970s, NMOS-only logic started to suffer from the same plague that made high-density bipolar logic unattractive or infeasible: power consumption. This realization, combined with progress in manufacturing technology, finally tilted the balance towards the CMOS technology, and this is where we still are today. Interestingly enough, power consumption concerns are rapidly becoming dominant in CMOS design as well, and this time there does not seem to be a new technology around the corner to alleviate the problem.

Although the large majority of the current integrated circuits are implemented in the MOS technology, other technologies come into play when very high performance is at stake. An example of this is the BiCMOS technology that combines bipolar and MOS devices on the same die. BiCMOS is used in high-speed memories and gate arrays. When even higher performance is necessary, other technologies emerge besides the already mentioned bipolar silicon ECL family—Gallium-Arsenide, Silicon-Germanium and even superconducting technologies. These technologies only play a very small role in the overall digital integrated circuit design scene. With the ever increasing performance of CMOS, this role is bound to be further reduced with time. Hence the focus of this textbook on CMOS only.

1.2 Issues in Digital Integrated Circuit Design

Integration density and performance of integrated circuits have gone through an astounding revolution in the last couple of decades. In the 1960s, Gordon Moore, then with Fairchild Corporation and later cofounder of Intel, predicted that the number of transistors that can be integrated on a single die would grow exponentially with time. This prediction, later called *Moore's law*, has proven to be amazingly visionary [Moore65]. Its validity is best illustrated with the aid of a set of graphs. Figure 1.2 plots the integration density of both logic IC's and memory as a function of time. As can be observed, integration complexity doubles approximately every 1 to 2 years. As a result, memory density has increased by more than a thousandfold since 1970.

An intriguing case study is offered by the microprocessor. From its inception in the early seventies, the microprocessor has grown in performance and complexity at a steady and predictable pace. The transistor counts for a number of landmark designs are collected in Figure 1.3. The million-transistor/chip barrier was crossed in the late eighties. Clock frequencies double every three years and have reached into the GHz range. This is illus-



(a) Trends in logic IC complexity (b) Trends in memory complexity

Figure 1.2 Evolution of integration complexity of logic ICs and memories as a function of time.

trated in Figure 1.4, which plots the microprocessor trends in terms of performance at the beginning of the 21st century. An important observation is that, as of now, these trends have not shown any signs of a slow-down.

It should be no surprise to the reader that this revolution has had a profound impact on how digital circuits are designed. Early designs were truly hand-crafted. Every transistor was laid out and optimized individually and carefully fitted into its environment. This is adequately illustrated in Figure 1.5a, which shows the design of the Intel 4004 microprocessor. This approach is, obviously, not appropriate when more than a million devices have to be created and assembled. With the rapid evolution of the design technology, time-to-market is one of the crucial factors in the ultimate success of a component.

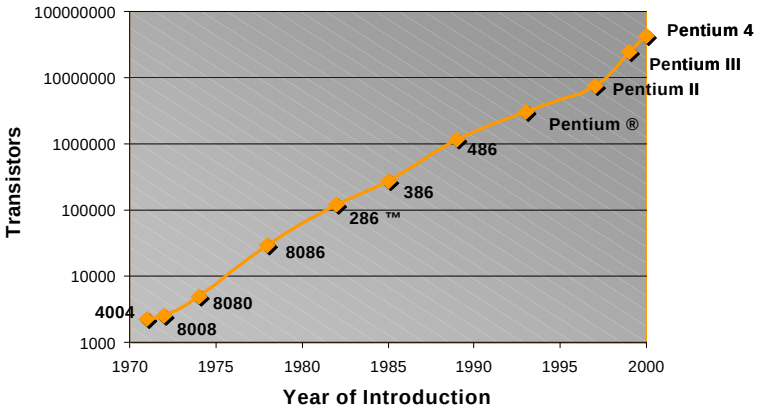


Figure 1.3 Historical evolution of microprocessor transistor count (from [Intel01]).

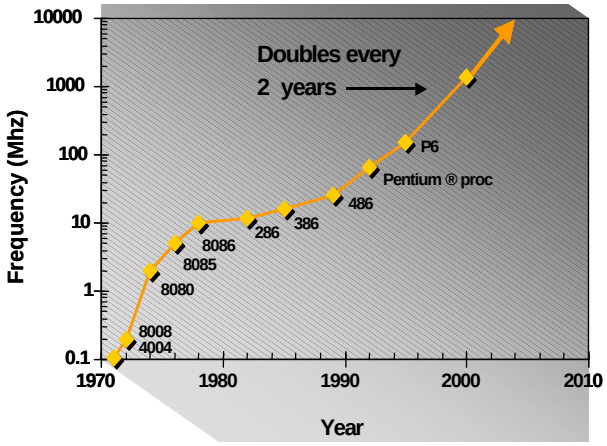
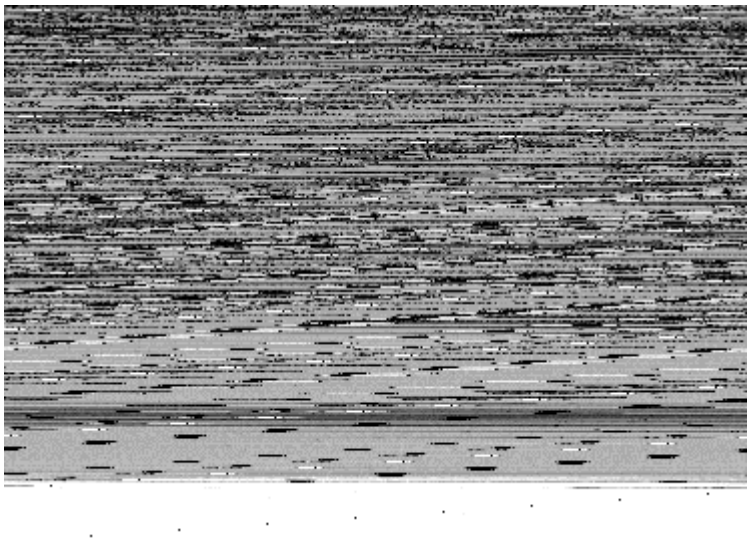


Figure 1.4 Microprocessor performance trends at the beginning of the 21st century.

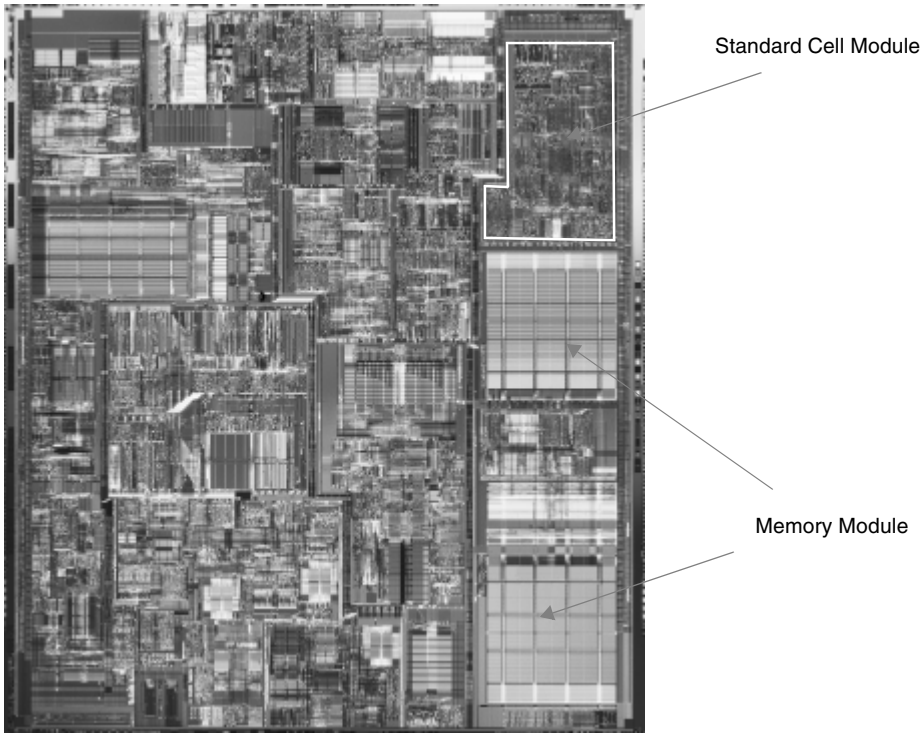
Designers have, therefore, increasingly adhered to rigid design methodologies and strategies that are more amenable to design automation. The impact of this approach is apparent from the layout of one of the later Intel microprocessors, the Pentium® 4, shown in Figure 1.5b. Instead of the individualized approach of the earlier designs, a circuit is constructed in a hierarchical way: a processor is a collection of modules, each of which consists of a number of cells on its own. Cells are reused as much as possible to reduce the design effort and to enhance the chances for a first-time-right implementation. The fact that this hierarchical approach is at all possible is the key ingredient for the success of digital circuit design and also explains why, for instance, very large scale analog design has never caught on.

The obvious next question is why such an approach is feasible in the digital world and not (or to a lesser degree) in analog designs. The crucial concept here, and the most important one in dealing with the complexity issue, is *abstraction*. At each design level, the internal details of a complex module can be abstracted away and replaced by a *black box view* or *model*. This model contains virtually all the information needed to deal with the block at the next level of hierarchy. For instance, once a designer has implemented a multiplier module, its performance can be defined very accurately and can be captured in a model. The performance of this multiplier is in general only marginally influenced by the way it is utilized in a larger system. For all purposes, it can hence be considered a black box with known characteristics. As there exists no compelling need for the system designer to look inside this box, design complexity is substantially reduced. The impact of this *divide and conquer* approach is dramatic. Instead of having to deal with a myriad of elements, the designer has to consider only a handful of components, each of which are characterized in performance and cost by a small number of parameters.

This is analogous to a software designer using a library of software routines such as input/output drivers. Someone writing a large program does not bother to look inside those library routines. The only thing he cares about is the intended result of calling one of those modules. Imagine what writing software programs would be like if one had to fetch every bit individually from the disk and ensure its correctness instead of relying on handy “file open” and “get string” operators.



(a) The 4004 microprocessor



(b) The Pentium ® 4 microprocessor

Figure 1.5 Comparing the design methodologies of the Intel 4004 (1971) and Pentium ® 4 (2000 microprocessors (reprinted with permission from Intel).

Typically used abstraction levels in digital circuit design are, in order of increasing abstraction, the device, circuit, gate, functional module (e.g., adder) and system levels (e.g., processor), as illustrated in Figure 1.6. A semiconductor device is an entity with a

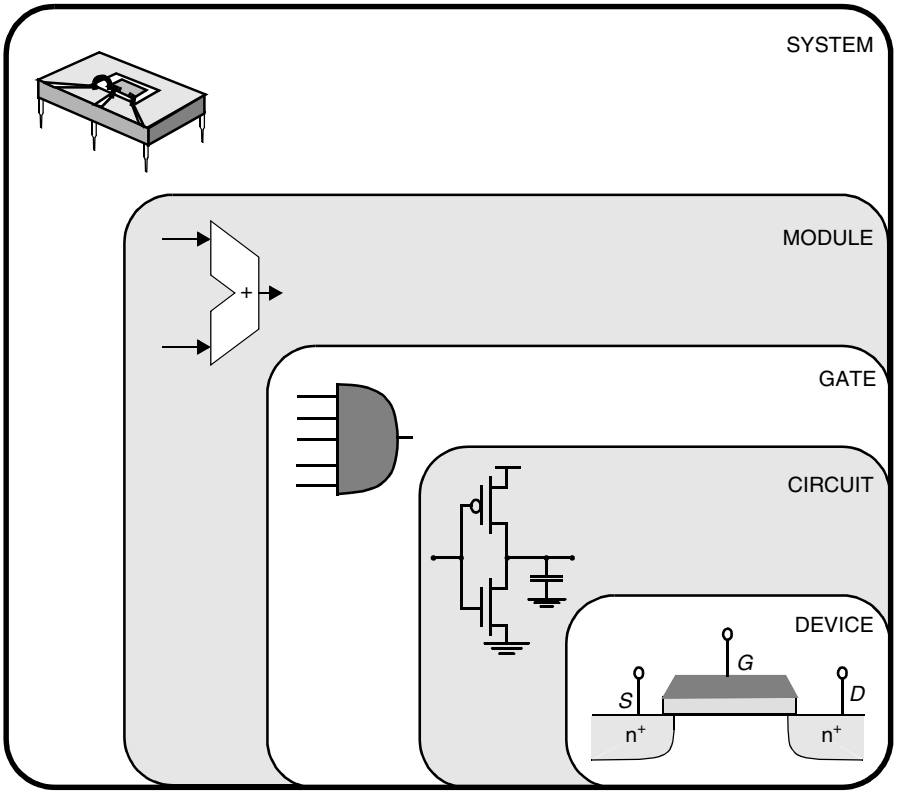


Figure 1.6 Design abstraction levels in digital circuits.

very complex behavior. No circuit designer will ever seriously consider the solid-state physics equations governing the behavior of the device when designing a digital gate. Instead he will use a simplified model that adequately describes the input-output behavior of the transistor. For instance, an AND gate is adequately described by its Boolean expression ($Z = A.B$), its bounding box, the position of the input and output terminals, and the delay between the inputs and the output.

This design philosophy has been the enabler for the emergence of elaborate *computer-aided design* (CAD) frameworks for digital integrated circuits; without it the current design complexity would not have been achievable. Design tools include simulation at the various complexity levels, design verification, layout generation, and design synthesis. An overview of these tools and design methodologies is given in Chapter 8 of this textbook.

Furthermore, to avoid the redesign and reverification of frequently used cells such as basic gates and arithmetic and memory modules, designers most often resort to *cell libraries.* These libraries contain not only the layouts, but also provide complete documentation and characterization of the behavior of the cells. The use of cell libraries is, for

instance, apparent in the layout of the Pentium ® 4 processor (Figure 1.5b). The integer and floating-point unit, just to name a few, contain large sections designed using the so-called *standard cell approach*. In this approach, logic gates are placed in rows of cells of equal height and interconnected using routing channels. The layout of such a block can be generated automatically given that a library of cells is available.

The preceding analysis demonstrates that design automation and modular design practices have effectively addressed some of the complexity issues incurred in contemporary digital design. This leads to the following pertinent question. If design automation solves all our design problems, why should we be concerned with digital circuit design at all? Will the next-generation digital designer ever have to worry about transistors or parasitics, or is the smallest design entity he will ever consider the gate and the module?

The truth is that the reality is more complex, and various reasons exist as to why an insight into digital circuits and their intricacies will still be an important asset for a long time to come.

- First of all, someone still has to *design and implement the module libraries*. Semiconductor technologies continue to advance from year to year. Until one has developed a fool-proof approach towards "porting" a cell from one technology to another, each change in technology—which happens approximately every two years—requires a redesign of the library.
- Creating an adequate *model* of a cell or module requires an in-depth understanding of its internal operation. For instance, to identify the dominant performance parameters of a given design, one has to recognize the critical timing path first.
- The library-based approach works fine when the design constraints (speed, cost or power) are not stringent. This is the case for a large number of *application-specific designs*, where the main goal is to provide a more integrated system solution, and performance requirements are easily within the capabilities of the technology. Unfortunately for a large number of other products such as microprocessors, success hinges on high performance, and designers therefore tend to push technology to its limits. At that point, the hierarchical approach tends to become somewhat less attractive. To resort to our previous analogy to software methodologies, a programmer tends to "customize" software routines when execution speed is crucial; compilers—or design tools—are not yet to the level of what human sweat or ingenuity can deliver.
- Even more important is the observation that the abstraction-based approach is only correct to a certain degree. The performance of, for instance, an adder can be substantially influenced by the way it is connected to its environment. The interconnection wires themselves contribute to delay as they introduce parasitic capacitances, resistances and even inductances. The impact of the *interconnect parasitics* is bound to increase in the years to come with the scaling of the technology.
- Scaling tends to emphasize some other deficiencies of the abstraction-based model. Some design entities tend to be *global or external* (to resort anew to the software analogy). Examples of global factors are the clock signals, used for synchronization in a digital design, and the supply lines. Increasing the size of a digital design has a

profound effect on these global signals. For instance, connecting more cells to a supply line can cause a voltage drop over the wire, which, in its turn, can slow down all the connected cells. Issues such as clock distribution, circuit synchronization, and supply-voltage distribution are becoming more and more critical. Coping with them requires a profound understanding of the intricacies of digital circuit design.

- Another impact of technology evolution is that *new design issues* and constraints tend to emerge over time. A typical example of this is the periodical reemergence of power dissipation as a constraining factor, as was already illustrated in the historical overview. Another example is the changing ratio between device and interconnect parasitics. To cope with these unforeseen factors, one must at least be able to model and analyze their impact, requiring once again a profound insight into circuit topology and behavior.
- Finally, when things can go wrong, they do. A fabricated circuit does not always exhibit the exact waveforms one might expect from advance simulations. Deviations can be caused by variations in the fabrication process parameters, or by the inductance of the package, or by a badly modeled clock signal. *Troubleshooting* a design requires circuit expertise.

For all the above reasons, it is my belief that an in-depth knowledge of digital circuit design techniques and approaches is an essential asset for a digital-system designer. Even though she might not have to deal with the details of the circuit on a daily basis, the understanding will help her to cope with unexpected circumstances and to determine the dominant effects when analyzing a design.

Example 1.1 Clocks Defy Hierarchy

To illustrate some of the issues raised above, let us examine the impact of deficiencies in one of the most important global signals in a design, the *clock*. The function of the clock signal in a digital design is to order the multitude of events happening in the circuit. This task can be compared to the function of a traffic light that determines which cars are allowed to move. It also makes sure that all operations are completed before the next one starts—a traffic light should be green long enough to allow a car or a pedestrian to cross the road. Under ideal circumstances, the clock signal is a periodic step waveform with transitions synchronized throughout the designed circuit (Figure 1.7a). In light of our analogy, changes in the traffic lights should be synchronized to maximize throughput while avoiding accidents. The importance of the *clock alignment* concept is illustrated with the example of two cascaded registers, both operating on the rising edge of the clock ϕ (Figure 1.7b). Under normal operating conditions, the input *In* gets sampled into the first register on the rising edge of ϕ and appears at the output exactly one clock period later. This is confirmed by the simulations shown in Figure 1.8c (signal *Out*).

Due to delays associated with routing the clock wires, it may happen that the clocks become misaligned with respect to each other. As a result, the registers are interpreting time indicated by the clock signal differently. Consider the case that the clock signal for the second register is delayed—or skewed—by a value δ. The rising edge of the delayed clock ϕ' will postpone the sampling of the input of the second register. If the time it takes to propagate the output of the first register to the input of the second is smaller than the clock delay, the latter will sample the wrong value. This causes the output to change prematurely, as clearly illustrated in the simulation, where the signal *Out*$'$ goes high at the first rising edge of ϕ' instead of

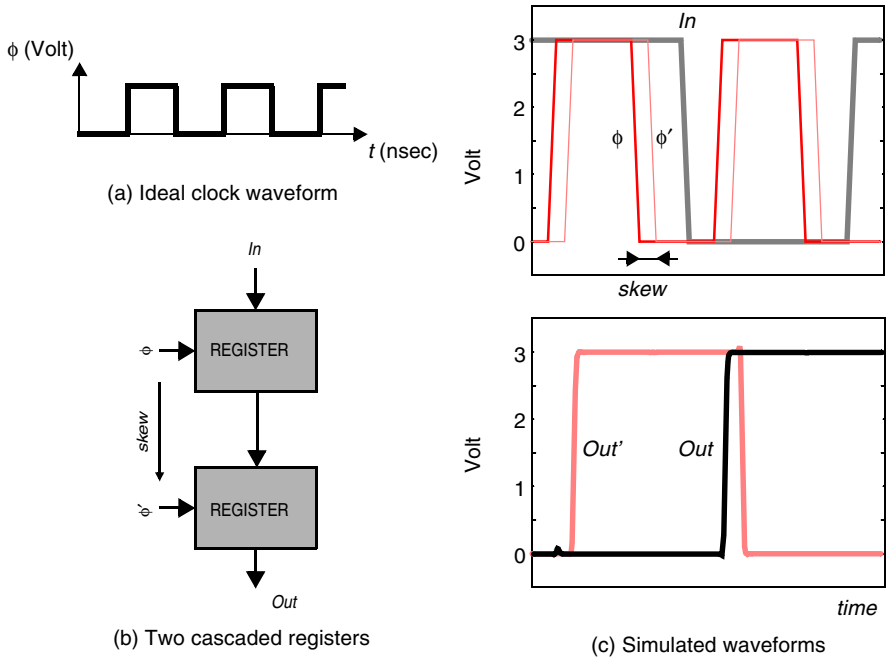


Figure 1.7 Impact of clock misalignment.

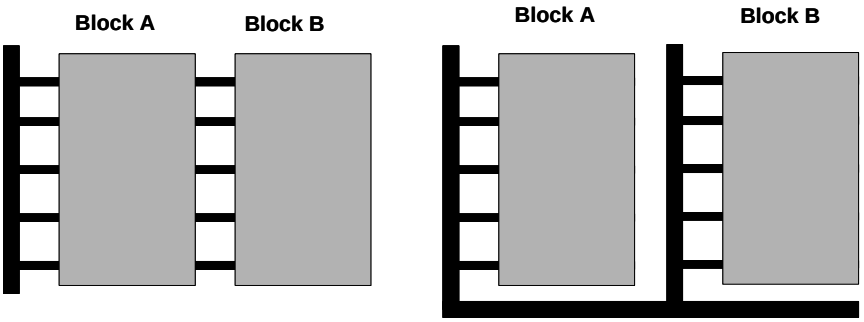
the second one. In terms of our traffic analogy, cars of a first traffic light hit the cars of the next light that have not left yet.

Clock misalignment, or *clock skew*, as it is normally called, is an important example of how global signals may influence the functioning of a hierarchically designed system. Clock skew is actually one of the most critical design problems facing the designers of large, high-performance systems.

Example 1.2 Power Distribution Networks Defy Hierarchy

While the clock signal is one example of a global signal that crosses the chip hierarchy boundaries, the power distribution network represents another. A digital system requires a stable DC voltage to be supplied to the individual gates. To ensure proper operation, this voltage should be stable within a few hundred millivolts. The power distribution system has to provide this stable voltage in the presence of very large current variations. The resistive nature of the on-chip wires and the inductance of the IC package pins make this a difficult proposition. For example, the average DC current to be supplied to a 100 W-1V microprocessor equals 100 A! The peak current can easily be twice as large, and current demand can readily change from almost zero to this peak value over a short time—in the range of 1 nsec or less. This leads to a current variation of 100 GA/sec, which is a truly astounding number.

Consider the problem of the resistance of power-distribution wires. A current of 1 A running through a wire with a resistance of 1 Ω causes a voltage drop of 1V. With supply voltages of modern digital circuits ranging between 1.2 and 2.5 V, such a drop is unaccept-



(a) Routing through the block (b) Routing around the block

Figure 1.8 Power distribution network design.

able. Making the wires wider reduces the resistance, and hence the voltage drop. While this sizing of the power network is relatively simple in a flat design approach, it is a lot more complex in a hierarchical design. For example, consider the two blocks below in Figure 1.8a [Saleh01]. If power distribution for Block A is examined in isolation, the additional loading due to the presence of Block B is not taken into account. If power is routed through Block A to Block B, a larger IR drop will occur in Block B since power is also being consumed by Block A before it reaches Block B.

Since the total IR drop is based on the resistance seen from the pin to the block, one could route around the block and feed power to each block separately, as shown in Figure 1.8b. Ideally, the main trunks should be large enough to handle all the current flowing through separate branches. Although routing power this way is easier to control and maintain, it also requires more area to implement. The large metal trunks of power have to be sized to handle all the current for each block. This requirement forces designers to set aside area for power busing that takes away from the available routing area.

As more and more blocks are added, the complex interactions between the blocks determine the actual voltage drops. For instance, it is not always easy to determine which way the current will flow when multiple parallel paths are available between the power source and the consuming gate. Also, currents into the different modules do rarely peak at the same time. All these considerations make the design of the power-distribution a challenging job. It requires a design methodology approach that supersedes the artificial boundaries imposed by hierarchical design.

The purpose of this textbook is to provide *a bridge between the abstract vision of digital design and the underlying digital circuit and its peculiarities*. While starting from a solid understanding of the operation of electronic devices and an in-depth analysis of the nucleus of digital design—the inverter—we will gradually channel this knowledge into the design of more complex entities, such as complex gates, datapaths, registers, controllers, and memories. The persistent quest for a designer when designing each of the mentioned modules is to identify the dominant design parameters, to locate the section of the design he should focus his optimizations on, and to determine the specific properties that make the module under investigation (e.g., a memory) different from any others.

The text also addresses other compelling (global) issues in modern digital circuit design such as *power dissipation, interconnect, timing, and synchronization*.

1.3 Quality Metrics of a Digital Design

This section defines a set of basic properties of a digital design. These properties help to quantify the quality of a design from different perspectives: cost, functionality, robustness, performance, and energy consumption. Which one of these metrics is most important depends upon the application. For instance, pure speed is a crucial property in a compute server. On the other hand, energy consumption is a dominant metric for hand-held mobile applications such as cell phones. The introduced properties are relevant at all levels of the design hierarchy, be it system, chip, module, and gate. To ensure consistency in the definitions throughout the design hierarchy stack, we propose a bottom-up approach: we start with defining the basic quality metrics of a simple inverter, and gradually expand these to the more complex functions such as gate, module, and chip.

1.3.1 Cost of an Integrated Circuit

The total cost of any product can be separated into two components: the recurring expenses or the *variable cost*, and the non-recurring expenses or the *fixed cost*.

Fixed Cost

The fixed cost is independent of the sales volume, the number of products sold. An important component of the fixed cost of an integrated circuit is the effort in time and manpower it takes to produce the design. This design cost is strongly influenced by the complexity of the design, the aggressiveness of the specifications, and the productivity of the designer. Advanced design methodologies that automate major parts of the design process can help to boost the latter. Bringing down the design cost in the presence of an ever-increasing IC complexity is one of the major challenges that is always facing the semiconductor industry.

Additionally, one has to account for the *indirect costs*, the company overhead that cannot be billed directly to one product. It includes amongst others the company's research and development (R&D), manufacturing equipment, marketing, sales, and building infrastructure.

Variable Cost

This accounts for the cost that is directly attributable to a manufactured product, and is hence proportional to the product volume. Variable costs include the costs of the parts used in the product, assembly costs, and testing costs. The total cost of an integrated circuit is now

$$\text{cost per IC} = \text{variable cost per IC} + \left(\frac{\text{fixed cost}}{\text{volume}}\right) \tag{1.1}$$

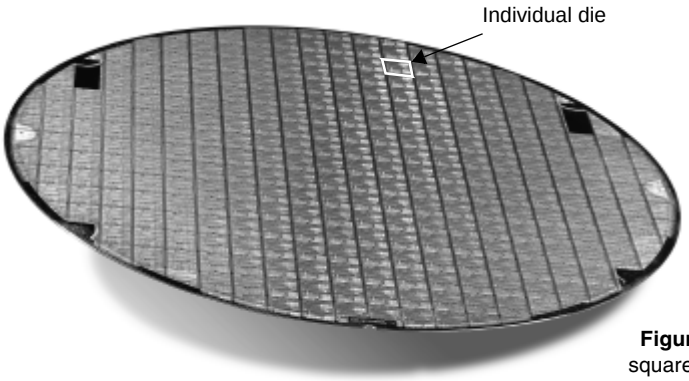


Figure 1.9 Finished wafer. Each square represents a die - in this case the AMD Duron™ microprocessor (Reprinted with permission from AMD).

The impact of the fixed cost is more pronounced for small-volume products. This also explains why it makes sense to have large design team working for a number of years on a hugely successful product such as a microprocessor.

While the cost of producing a single transistor has dropped exponentially over the past decades, the basic variable-cost equation has not changed:

$$\text{variable cost} = \frac{\text{cost of die + cost of die test + cost of packaging}}{\text{final test yield}} \tag{1.2}$$

As will be elaborated on in Chapter 2, the IC manufacturing process groups a number of identical circuits onto a single *wafer* (Figure 1.9). Upon completion of the fabrication, the wafer is chopped into *dies*, which are then individually packaged after being *tested*. We will focus on the cost of the dies in this discussion. The cost of packaging and test is the topic of later chapters.

The die cost depends upon the number of good die on a wafer, and the percentage of those that are functional. The latter factor is called the *die yield*.

$$\text{cost of die} = \frac{\text{cost of wafer}}{\text{dies per wafer} \times \text{die yield}} \tag{1.3}$$

The number of dies per wafer is, in essence, the area of the wafer divided by the die area.The actual situation is somewhat more complicated as wafers are round, and chips are square. Dies around the perimeter of the wafer are therefore lost. The size of the wafer has been steadily increasing over the years, yielding more dies per fabrication run. Eq. (1.3) also presents the first indication that the cost of a circuit is dependent upon the chip area—increasing the chip area simply means that less dies fit on a wafer.

The actual relation between cost and area is more complex, and depends upon the die yield. Both the substrate material and the manufacturing process introduce faults that can cause a chip to fail. Assuming that the defects are randomly distributed over the wafer, and that the yield is inversely proportional to the complexity of the fabrication process, we obtain the following expression of the die yield:

$$\text{die yield} = \left(1 + \frac{\text{defects per unit area} \times \text{die area}}{\alpha}\right)^{-\alpha} \tag{1.4}$$

α is a parameter that depends upon the complexity of the manufacturing process, and is roughly proportional to the number of masks. $\alpha = 3$ is a good estimate for today's complex CMOS processes. The defects per unit area is a measure of the material and process induced faults. A value between 0.5 and 1 defects/cm^2 is typical these days, but depends strongly upon the maturity of the process.

Example 1.3 Die Yield

Assume a wafer size of 12 inch, a die size of 2.5 cm^2, 1 defects/cm^2, and $\alpha = 3$. Determine the die yield of this CMOS process run.

The number of dies per wafer can be estimated with the following expression, which takes into account the lost dies around the perimeter of the wafer.

$$\text{dies per wafer} = \frac{\pi \times (\text{wafer diameter}/2)^2}{\text{die area}} - \frac{\pi \times \text{wafer diameter}}{\sqrt{2 \times \text{die area}}}$$

This means 252 (= 296 - 44) potentially operational dies for this particular example. The die yield can be computed with the aid of Eq. (1.4), and equals 16%! This means that on the average only 40 of the dies will be fully functional.

The bottom line is that the number of functional of dies per wafer, and hence the cost per die is a strong function of the die area. While the yield tends to be excellent for the smaller designs, it drops rapidly once a certain threshold is exceeded. Bearing in mind the equations derived above and the typical parameter values, we can conclude that die costs are proportional to the fourth power of the area:

$$\text{cost of die} = f(\text{die area})^4 \tag{1.5}$$

The area is a function that is directly controllable by the designer(s), and is the prime metric for cost. Small area is hence a desirable property for a digital gate. The smaller the gate, the higher the integration density and the smaller the die size. Smaller gates furthermore tend to be faster and consume less energy, as the total gate capacitance—which is one of the dominant performance parameters—often scales with the area.

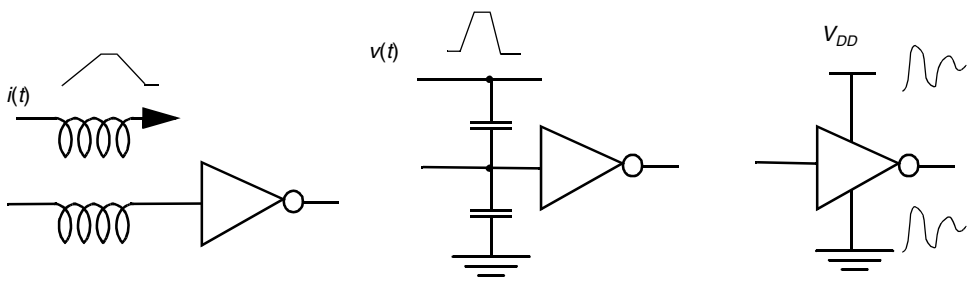
The *number of transistors* in a gate is indicative for the expected implementation area. Other parameters may have an impact, though. For instance, a complex interconnect pattern between the transistors can cause the wiring area to dominate. The *gate complexity*, as expressed by the number of transistors and the regularity of the interconnect structure, also has an impact on the design cost. Complex structures are harder to implement and tend to take more of the designers valuable time. Simplicity and regularity is a precious property in cost-sensitive designs.

1.3.2 Functionality and Robustness

A prime requirement for a digital circuit is, obviously, that it performs the function it is designed for. The measured behavior of a manufactured circuit normally deviates from the

expected response. One reason for this aberration are the variations in the manufacturing process. The dimensions, threshold voltages, and currents of an MOS transistor vary between runs or even on a single wafer or die. The electrical behavior of a circuit can be profoundly affected by those variations. The presence of disturbing noise sources on or off the chip is another source of deviations in circuit response. The word *noise* in the context of digital circuits means *"unwanted variations of voltages and currents at the logic nodes."* Noise signals can enter a circuit in many ways. Some examples of digital noise sources are depicted in Figure 1.10. For instance, two wires placed side by side in an integrated circuit form a coupling capacitor and a mutual inductance. Hence, a voltage or current change on one of the wires can influence the signals on the neighboring wire. Noise on the power and ground rails of a gate also influences the signal levels in the gate.

Most noise in a digital system is internally generated, and the noise value is proportional to the signal swing. Capacitive and inductive cross talk, and the internally-generated power supply noise are examples of such. Other noise sources such as input power supply noise are external to the system, and their value is not related to the signal levels. For these sources, the noise level is directly expressed in Volt or Ampere. Noise sources that are a function of the signal level are better expressed as a fraction or percentage of the signal level. Noise is a major concern in the engineering of digital circuits. How to cope with all these disturbances is one of the main challenges in the design of high-performance digital circuits and is a recurring topic in this book.



(a) Inductive coupling (b) Capacitive coupling (c) Power and ground noise

Figure 1.10 Noise sources in digital circuits.

The steady-state parameters (also called the *static behavior*) of a gate measure how robust the circuit is with respect to both variations in the manufacturing process and noise disturbances. The definition and derivation of these parameters requires a prior understanding of how digital signals are represented in the world of electronic circuits.

Digital circuits (DC) perform operations on *logical* (or *Boolean*) variables. A logical variable x can only assume two discrete values:

$$x \in \{0,1\}$$

As an example, the inversion (i.e., the function that an inverter performs) implements the following compositional relationship between two Boolean variables x and y:

$$y = \bar{x}: \{x = 0 \Rightarrow y = 1; x = 1 \Rightarrow y = 0\} \tag{1.6}$$

A logical variable is, however, a mathematical abstraction. In a physical implementation, such a variable is represented by an electrical quantity. This is most often a node voltage that is not discrete but can adopt a continuous range of values. This electrical voltage is turned into a discrete variable by associating *a nominal voltage level* with each logic state: $1 \Leftrightarrow V_{OH}$, $0 \Leftrightarrow V_{OL}$, where V_{OH} and V_{OL} represent the *high* and the *low* logic levels, respectively. Applying V_{OH} to the input of an inverter yields V_{OL} at the output and vice versa. The difference between the two is called the *logic or signal swing* V_{sw}.

$$\begin{aligned} V_{OH} &= \overline{(V_{OL})} \\ V_{OL} &= \overline{(V_{OH})} \end{aligned} \tag{1.7}$$

The Voltage-Transfer Characteristic

Assume now that a logical variable *in* serves as the input to an inverting gate that produces the variable *out*. The electrical function of a gate is best expressed by its *voltage-transfer characteristic* (VTC) (sometimes called the *DC transfer characteristic*), which plots the output voltage as a function of the input voltage $V_{out} = f(V_{in})$. An example of an inverter VTC is shown in Figure 1.11. The high and low nominal voltages, V_{OH} and V_{OL}, can readily be identified—$V_{OH} = f(V_{OL})$ and $V_{OL} = f(V_{OH})$. Another point of interest of the VTC is the *gate or switching threshold voltage* V_M (not to be confused with the threshold voltage of a transistor), that is defined as $V_M = f(V_M)$. V_M can also be found graphically at the intersection of the VTC curve and the line given by $V_{out} = V_{in}$. The gate threshold voltage presents the midpoint of the switching characteristics, which is obtained when the output of a gate is short-circuited to the input. This point will prove to be of particular interest when studying circuits with feedback (also called *sequential circuits*).

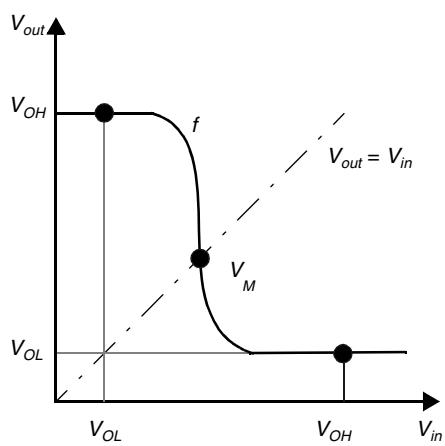


Figure 1.11 Inverter voltage-transfer characteristic.

Even if an ideal nominal value is applied at the input of a gate, the output signal often deviates from the expected nominal value. These deviations can be caused by noise or by the loading on the output of the gate (i.e., by the number of gates connected to the output signal). Figure 1.12a illustrates how a logic level is represented in reality by a range of acceptable voltages, separated by a region of uncertainty, rather than by nominal levels

alone. The regions of acceptable high and low voltages are delimited by the $\boldsymbol{V}_{IH}$ and $\boldsymbol{V}_{IL}$ voltage levels, respectively. These represent by definition the points where the gain (= dV_{out} / dV_{in}) of the VTC equals –1 as shown in Figure 1.12b. The region between V_{IH} and V_{IL} is called the *undefined region* (sometimes also referred to as *transition width*, or *TW*). Steady-state signals should avoid this region if proper circuit operation is to be ensured.

Noise Margins

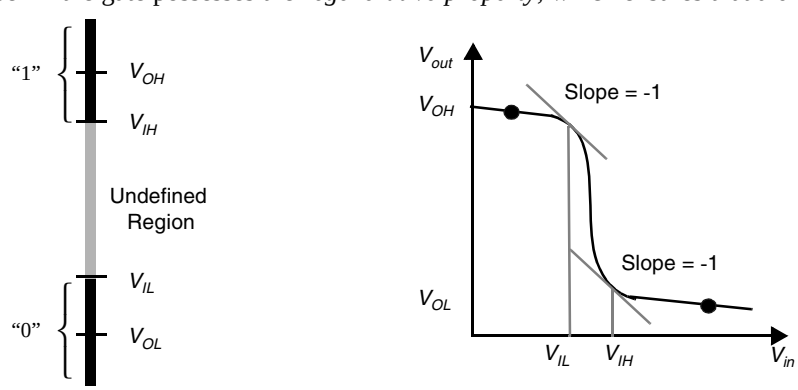
For a gate to be robust and insensitive to noise disturbances, it is essential that the "0" and "1" intervals be as large as possible. A measure of the sensitivity of a gate to noise is given by the noise margins NM_L (*noise margin low*) and NM_H (*noise margin high*), which quantize the size of the legal "0" and "1", respectively, and set a fixed maximum threshold on the noise value:

$$\begin{aligned} NM_L &= V_{IL} - V_{OL} \\ NM_H &= V_{OH} - V_{IH} \end{aligned} \tag{1.8}$$

The noise margins represent the levels of noise that can be sustained when gates are cascaded as illustrated in Figure 1.13. It is obvious that the margins should be larger than 0 for a digital circuit to be functional and by preference should be as large as possible.

Regenerative Property

A large noise margin is a desirable, but not sufficient requirement. Assume that a signal is disturbed by noise and differs from the nominal voltage levels. As long as the signal is within the noise margins, the following gate continues to function correctly, although its output voltage varies from the nominal one. This deviation is added to the noise injected at the output node and passed to the next gate. The effect of different noise sources may accumulate and eventually force a signal level into the undefined region. This, fortunately, does not happen if the gate possesses the *regenerative property*, which ensures that a dis-



(a) Relationship between voltage and logic levels (b) Definition of V_{IH} and V_{IL}

Figure 1.12 Mapping logic levels to the voltage domain.

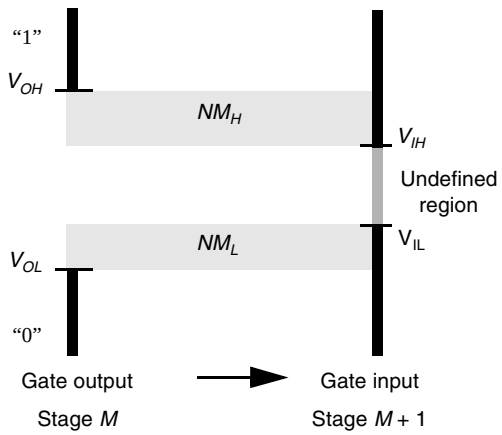
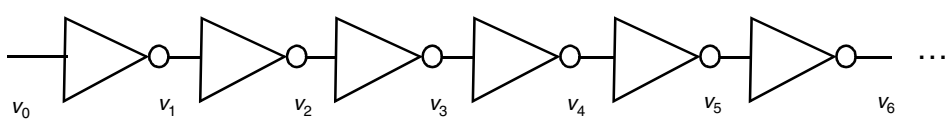


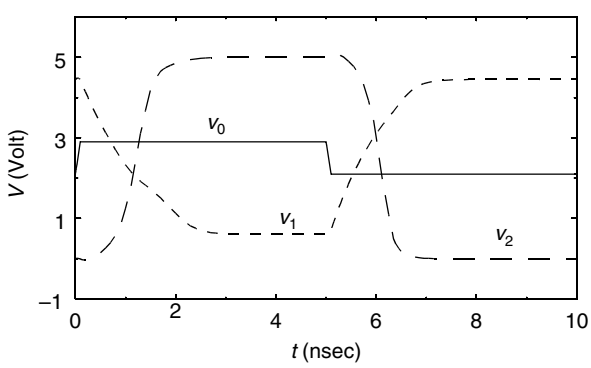
Figure 1.13 Cascaded inverter gates: definition of noise margins.

turbed signal gradually converges back to one of the nominal voltage levels after passing through a number of logical stages. This property can be understood as follows:

An input voltage v_{in} ($v_{in} \in$ "0") is applied to a chain of N inverters (Figure 1.14a). Assuming that the number of inverters in the chain is even, the output voltage v_{out} ($N \rightarrow \infty$) will equal V_{OL} if and only if the inverter possesses the regenerative property. Similarly, when an input voltage v_{in} ($v_{in} \in$ "1") is applied to the inverter chain, the output voltage will approach the nominal value V_{OH}.



(a) A chain of inverters



(b) Simulated response of chain of MOS inverters

Figure 1.14 The regenerative property.

Example 1.4 Regenerative property

The concept of regeneration is illustrated in Figure 1.14b, which plots the simulated transient response of a chain of CMOS inverters. The input signal to the chain is a step-waveform with

a degraded amplitude, which could be caused by noise. Instead of swinging from rail to rail, v_0 only extends between 2.1 and 2.9 V. From the simulation, it can be observed that this deviation rapidly disappears, while progressing through the chain; v_1, for instance, extends from 0.6 V to 4.45 V. Even further, v_2 already swings between the nominal V_{OL} and V_{OH}. The inverter used in this example clearly possesses the regenerative property.

The conditions under which a gate is regenerative can be intuitively derived by analyzing a simple case study. Figure 1.15(a) plots the VTC of an inverter $V_{out} = f(V_{in})$ as well as its inverse function *finv*(), which reverts the function of the *x*- and *y*-axis and is defined as follows:

$$in = f(out) \Rightarrow in = finv(out) \tag{1.9}$$

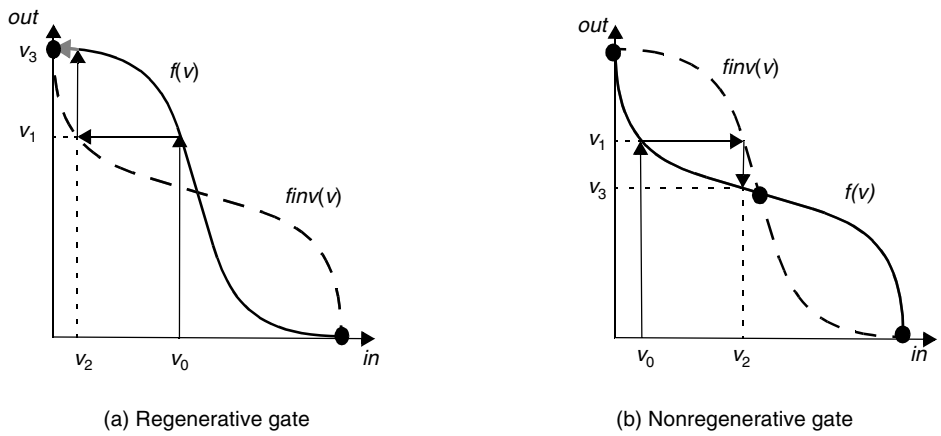


Figure 1.15 Conditions for regeneration.

Assume that a voltage v_0, deviating from the nominal voltages, is applied to the first inverter in the chain. The output voltage of this inverter equals $v_1 = f(v_0)$ and is applied to the next inverter. Graphically this corresponds to $v_1 = finv(v_2)$. The signal voltage gradually converges to the nominal signal after a number of inverter stages, as indicated by the arrows. In Figure 1.15(b) the signal does not converge to any of the nominal voltage levels but to an intermediate voltage level. Hence, the characteristic is nonregenerative. The difference between the two cases is due to the gain characteristics of the gates. To be regenerative, the VTC should have a transient region (or undefined region) with a gain *greater than* 1 in absolute value, bordered by the two legal zones, where the gain should be *smaller than* 1. Such a gate has two stable operating points. This clarifies the definition of the V_{IH} and the V_{IL} levels that form the boundaries between the legal and the transient zones.

Noise Immunity

While the noise margin is a meaningful means for measuring the robustness of a circuit against noise, it is not sufficient. It expresses the capability of a circuit to "overpower" a

noise source. *Noise immunity*, on the other hand, expresses the ability of the system to process and transmit information correctly in the presence of noise [Dally98]. Many digital circuits with low noise margins have very good noise immunity because they *reject a noise source* rather than overpower it. These circuits have the property that only a small fraction of a potentially-damaging noise source is coupled to the important circuit nodes. More precisely, the transfer function between noise source and signal node is far smaller than 1. Circuits that do not posses this property are *susceptible* to noise.

To study the noise immunity of a gate, we have to construct a noise budget that allocates the power budget to the various noise sources. As discussed earlier, the noise sources can be divided into sources that are

- proportional to the signal swing V_{sw}. The impact on the signal node is expressed as $g\ V_{sw}$.
- fixed. The impact on the signal node equals $f\ V_{Nf}$, with V_{nf} the amplitude of the noise source, and f the transfer function from noise to signal node.

We assume, for the sake of simplicity, that the noise margin equals half the signal swing (for both H and L). To operate correctly, the noise margin has to be larger than the sum of the coupled noise values.

$$V_{NM} = \frac{V_{sw}}{2} \geq \sum_i f_i V_{Nfi} + \sum_j g_j V_{sw} \tag{1.10}$$

Given a set of noise sources, we can derive the minimum signal swing necessary for the system to be operational,

$$V_{sw} \geq \frac{2\sum_i f_i V_{Nfi}}{1 - 2\sum_j g_j} \tag{1.11}$$

This makes it clear that the signal swing (and the noise margin) has to be large enough to overpower the impact of the fixed sources ($f\ V_{Nf}$). On the other hand, the sensitivity to internal sources depends primarily upon the noise suppressing capabilities of the gate, this is the proportionality or gain factors g_j. In the presence of large gain factors, increasing the signal swing does not do any good to suppress noise, as the noise increases proportionally. In later chapters, we will discuss some differential logic families that suppress most of the internal noise, and hence can get away with very small noise margins and signal swings.

Directivity

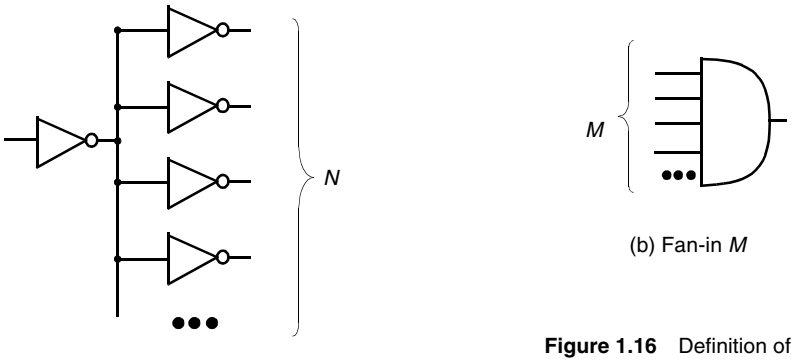
The directivity property requires a gate to be *unidirectional*, that is, changes in an output level should not appear at any unchanging input of the same circuit. If not, an output-signal transition reflects to the gate inputs as a noise signal, affecting the signal integrity.

In real gate implementations, full directivity can never be achieved. Some feedback of changes in output levels to the inputs cannot be avoided. Capacitive coupling between inputs and outputs is a typical example of such a feedback. It is important to minimize these changes so that they do not affect the logic levels of the input signals.

Fan-In and Fan-Out

The *fan-out* denotes *the number of load gates N that are connected to the output of the driving gate* (Figure 1.16). Increasing the fan-out of a gate can affect its logic output levels. From the world of analog amplifiers, we know that this effect is minimized by making the input resistance of the load gates as large as possible (minimizing the input currents) and by keeping the output resistance of the driving gate small (reducing the effects of load currents on the output voltage). When the fan-out is large, the added load can deteriorate the dynamic performance of the driving gate. For these reasons, many generic and library components define a *maximum fan-out* to guarantee that the static and dynamic performance of the element meet specification.

The *fan-in* of a gate is defined as the *number of inputs* to the gate (Figure 1.16b). Gates with large fan-in tend to be more complex, which often results in inferior static and dynamic properties.



(a) Fan-out *N*

(b) Fan-in *M*

Figure 1.16 Definition of fan-out and fan-in of a digital gate.

The Ideal Digital Gate

Based on the above observations, we can define the *ideal* digital gate from a static perspective. The ideal inverter model is important because it gives us a metric by which we can judge the quality of actual implementations.

Its VTC is shown in Figure 1.17 and has the following properties: infinite gain in the transition region, and gate threshold located in the middle of the logic swing, with high and low noise margins equal to half the swing. The input and output impedances of the ideal gate are infinity and zero, respectively (i.e., the gate has unlimited fan-out). While this ideal VTC is unfortunately impossible in real designs, some implementations, such as the static CMOS inverter, come close.

Example 1.5 Voltage-Transfer Characteristic

Figure 1.18 shows an example of a voltage-transfer characteristic of an actual, but outdated gate structure (as produced by SPICE in the DC analysis mode). The values of the dc-parameters are derived from inspection of the graph.

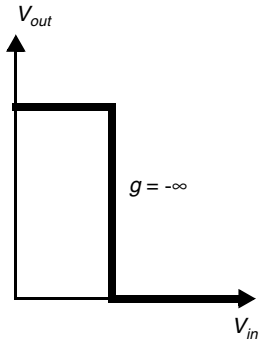


Figure 1.17 Ideal voltage-transfer characteristic.

$$V_{OH} = 3.5 \text{ V}; \quad V_{OL} = 0.45 \text{ V}$$

$$V_{IH} = 2.35 \text{ V}; \quad V_{IL} = 0.66 \text{ V}$$

$$V_M = 1.64 \text{ V}$$

$$NM_H = 1.15 \text{ V}; \quad NM_L = 0.21 \text{ V}$$

The observed transfer characteristic, obviously, is far from ideal: it is asymmetrical, has a very low value for NM_L, and the voltage swing of 3.05 V is substantially below the maximum obtainable value of 5 V (which is the value of the supply voltage for this design).

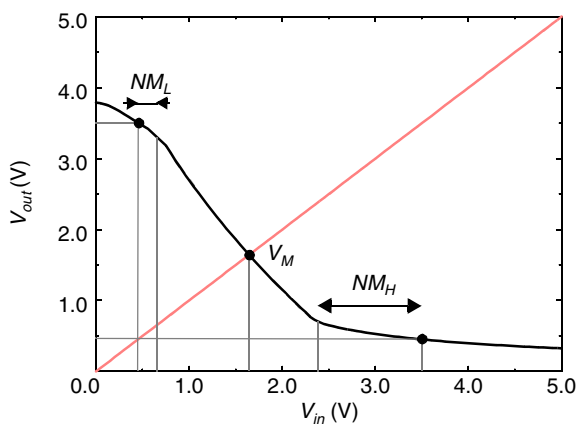


Figure 1.18 Voltage-transfer characteristic of an NMOS inverter of the 1970s.

1.3.3 Performance

From a system designers perspective, the performance of a digital circuit expresses the computational load that the circuit can manage. For instance, a microprocessor is often characterized by the number of instructions it can execute per second. This performance

metric depends both on the architecture of the processor—for instance, the number of instructions it can execute in parallel—, and the actual design of logic circuitry. While the former is crucially important, it is not the focus of this text book. We refer the reader to the many excellent books on this topic [for instance, Hennessy96]. When focusing on the pure design, performance is most often expressed by the duration of the clock period (*clock cycle time)*, or its rate (*clock frequency*). The minimum value of the clock period for a given technology and design is set by a number of factors such as the time it takes for the signals to propagate through the logic, the time it takes to get the data in and out of the registers, and the uncertainty of the clock arrival times. Each of these topics will be discussed in detail on the course of this text book. At the core of the whole performance analysis, however, lays the performance of an individual gate.

The *propagation delay* t_p of a gate defines how quickly it responds to a change at its input(s). It expresses *the delay experienced by a signal when passing through a gate*. It is measured between the 50% transition points of the input and output waveforms, as shown in Figure 1.19 for an inverting gate.[2] Because a gate displays different response times for rising or falling input waveforms, two definitions of the propagation delay are necessary. The t_{pLH} defines the response time of the gate for a *low to high* (or positive) output transition, while t_{pHL} refers to a *high to low* (or negative) transition. The propagation delay t_p is defined as the average of the two.

$$t_p = \frac{t_{pLH} + t_{pHL}}{2} \tag{1.12}$$

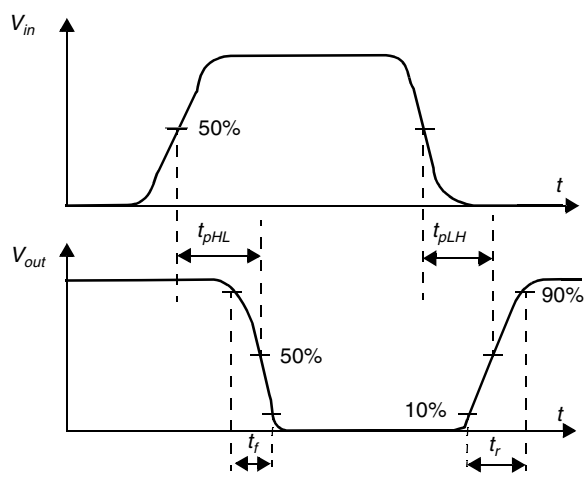


Figure 1.19 Definition of propagation delays and rise and fall times.

[2] The 50% definition is inspired the assumption that the switching threshold V_M is typically located in the middle of the logic swing.

CAUTION: : Observe that the propagation delay t_p, in contrast to t_{pLH} and t_{pHL}, is an artificial gate quality metric, and has no physical meaning per se. It is mostly used to compare different semiconductor technologies, or logic design styles.

The propagation delay is not only a function of the circuit technology and topology, but depends upon other factors as well. Most importantly, the delay is a function of the *slopes* of the input and output signals of the gate. To quantify these properties, we introduce the *rise and fall times* t_r and t_f, which are metrics that apply to individual signal waveforms rather than gates (Figure 1.19), and express how fast a signal transits between the different levels. The uncertainty over when a transition actually starts or ends is avoided by defining the rise and fall times between the 10% and 90% points of the waveforms, as shown in the Figure. The rise/fall time of a signal is largely determined by the strength of the driving gate, and the load presented by the node itself, which sums the contributions of the connecting gates (fan-out) and the wiring parasitics.

When comparing the performance of gates implemented in different technologies or circuit styles, it is important not to confuse the picture by including parameters such as load factors, fan-in and fan-out. A uniform way of measuring the t_p of a gate, so that technologies can be judged on an equal footing, is desirable. The de-facto standard circuit for delay measurement is the *ring oscillator*, which consists of an odd number of inverters connected in a circular chain (Figure 1.20). Due to the odd number of inversions, this circuit does not have a stable operating point and oscillates. The period T of the oscillation is determined by the propagation time of a signal transition through the complete chain, or $T = 2 \times t_p \times N$ with N the number of inverters in the chain. The factor 2 results from the observation that a full cycle requires both a low-to-high and a high-to-low transition. Note that this equation is only valid for $2Nt_p >> t_f + t_r$. If this condition is not met, the circuit might not oscillate—one "wave" of signals propagating through the ring will overlap with a successor and eventually dampen the oscillation. Typically, a ring oscillator needs a least five stages to be operational.

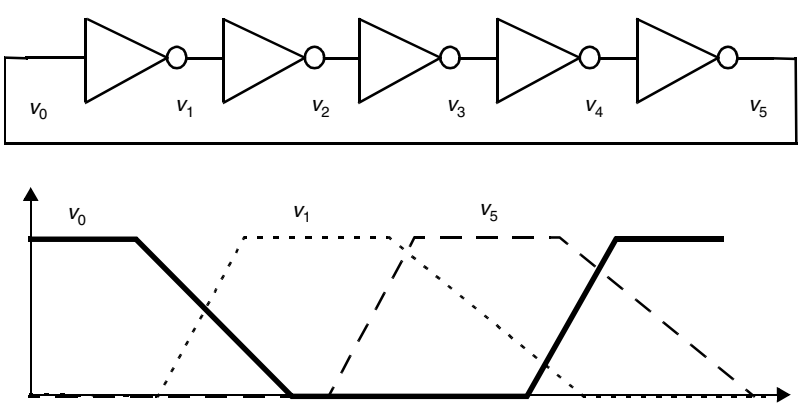


Figure 1.20 Ring oscillator circuit for propagation-delay measurement.

CAUTION: We must be extremely careful with results obtained from ring oscillator measurements. A t_p of 20 psec by no means implies that a circuit built with those gates will operate at 50 GHz. The oscillator results are primarily useful for quantifying the differences between various manufacturing technologies and gate topologies. The oscillator is an idealized circuit where each gate has a fan-in and fan-out of exactly one and parasitic loads are minimal. In more realistic digital circuits, fan-ins and fan-outs are higher, and interconnect delays are non-negligible. The gate functionality is also substantially more complex than a simple invert operation. As a result, the achievable clock frequency on average is 50 to a 100 times slower than the frequency predicted from ring oscillator measurements. This is an average observation; carefully optimized designs might approach the ideal frequency more closely.

Example 1.6 Propagation Delay of First-Order *RC* Network

Digital circuits are often modeled as first-order *RC* networks of the type shown in Figure 1.21. The propagation delay of such a network is thus of considerable interest.

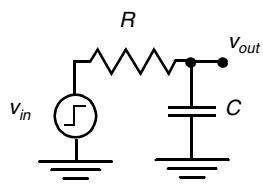


Figure 1.21 First-order *RC* network.

When applying a step input (with v_{in} going from 0 to V), the transient response of this circuit is known to be an exponential function, and is given by the following expression (where $\tau = RC$, the time constant of the network):

$$v_{out}(t) = (1 - e^{-t/\tau})\, V \tag{1.13}$$

The time to reach the 50% point is easily computed as $t = \ln(2)\tau = 0.69\tau$. Similarly, it takes $t = \ln(9)\tau = 2.2\tau$ to get to the 90% point. It is worth memorizing these numbers, as they are extensively used in the rest of the text.

1.3.4 Power and Energy Consumption

The power consumption of a design determines how much energy is consumed per operation, and much heat the circuit dissipates. These factors influence a great number of critical design decisions, such as the power-supply capacity, the battery lifetime, supply-line sizing, packaging and cooling requirements. Therefore, power dissipation is an important property of a design that affects feasibility, cost, and reliability. In the world of high-performance computing, power consumption limits, dictated by the chip package and the heat removal system, determine the number of circuits that can be integrated onto a single chip, and how fast they are allowed to switch.With the increasing popularity of mobile and distributed computation, energy limitations put a firm restriction on the number of computations that can be performed given a minimum time between battery recharges.

Depending upon the design problem at hand, different dissipation measures have to be considered. For instance, the peak power P_{peak} is important when studying supply-line sizing. When addressing cooling or battery requirements, one is predominantly interested in the average power dissipation P_{av}. Both measures are defined in equation Eq. (1.14):

$$P_{peak} = i_{peak} V_{supply} = max[p(t)]$$
$$P_{av} = \frac{1}{T}\int_0^T p(t)dt = \frac{V_{supply}}{T}\int_0^T i_{supply}(t)dt \tag{1.14}$$

where $p(t)$ is the instantaneous power, i_{supply} is the current being drawn from the supply voltage V_{supply} over the interval $t \in [0,T]$, and i_{peak} is the maximum value of i_{supply} over that interval.

The dissipation can further be decomposed into *static* and *dynamic* components. The latter occurs only during transients, when the gate is switching. It is attributed to the charging of capacitors and temporary current paths between the supply rails, and is, therefore, proportional to the switching frequency: *the higher the number of switching events, the higher the dynamic power consumption.* The static component on the other hand is present even when no switching occurs and is caused by static conductive paths between the supply rails or by leakage currents. It is always present, even when the circuit is in stand-by. Minimization of this consumption source is a worthwhile goal.

The propagation delay and the power consumption of a gate are related—the propagation delay is mostly determined by the speed at which a given amount of energy can be stored on the gate capacitors. The faster the energy transfer (or the higher the power consumption), the faster the gate. For a given technology and gate topology, the product of power consumption and propagation delay is generally a constant. This product is called the *power-delay product* (or PDP) and can be considered as a quality measure for a switching device. The PDP is simply the *energy* consumed by the gate *per switching event*. The ring oscillator is again the circuit of choice for measuring the PDP of a logic family.

An ideal gate is one that is fast, and consumes little energy. The *energy-delay* product (E-D) is a combined metric that brings those two elements together, and is often used as the ultimate quality metric. From the above, it should be clear that the E-D is equivalent to *power-delay*^{2}.

Example 1.7 Energy Dissipation of First-Order *RC* Network

Let us consider again the first-order *RC* network shown in Figure 1.21. When applying a step input (with V_{in} going from 0 to V), an amount of energy is provided by the signal source to the network. The total energy delivered by the source (from the start of the transition to the end) can be readily computed:

$$E_{in} = \int_0^\infty i_{in}(t)v_{in}(t)dt = V\int_0^\infty C\frac{dv_{out}}{dt}dt = (CV)\int_0^V dv_{out} = CV^2 \tag{1.15}$$

It is interesting to observe that the energy needed to charge a capacitor from 0 to V volt with a step input is a function of the size of the voltage step and the capacitance, but is inde-

pendent of the value of the resistor. We can also compute how much of the delivered energy gets stored on the capacitor at the end of the transition.

$$E_C = \int_0^\infty i_C(t)v_{out}(t)dt = \int_0^\infty C\frac{dv_{out}}{dt}v_{out}dt = C\int_0^V v_{out}dv_{out} = \frac{CV^2}{2} \tag{1.16}$$

This is exactly half of the energy delivered by the source. For those who wonder happened with the other half—a simple analysis shows that an equivalent amount gets dissipated as heat in the resistor during the transaction. We leave it to the reader to demonstrate that during the discharge phase (for a step from V to 0), the energy originally stored on the capacitor gets dissipated in the resistor as well, and turned into heat.

1.4 Summary

In this introductory chapter, we learned about the history and the trends in digital circuit design. We also introduced the important quality metrics, used to evaluate the quality of a design: cost, functionality, robustness, performance, and energy/power dissipation. At the end of the Chapter, you can find an extensive list of reference works that may help you to learn more about some of the topics introduced in the course of the text.

1.5 To Probe Further

The design of digital integrated circuits has been the topic of a multitude of textbooks and monographs. To help the reader find more information on some selected topics, an extensive list of reference works is listed below. The state-of-the-art developments in the area of digital design are generally reported in technical journals or conference proceedings, the most important of which are listed.

JOURNALS AND PROCEEDINGS

IEEE Journal of Solid-State Circuits
IEICE Transactions on Electronics (Japan)
Proceedings of The International Solid-State and Circuits Conference (ISSCC)
Proceedings of the VLSI Circuits Symposium
Proceedings of the Custom Integrated Circuits Conference (CICC)
European Solid-State Circuits Conference (ESSCIRC)

REFERENCE BOOKS

MOS

M. Annaratone, *Digital CMOS Circuit Design*, Kluwer, 1986.

T. Dillinger, *VLSI Engineering*, Prentice Hall, 1988.

E. Elmasry, ed., *Digital MOS Integrated Circuits*, IEEE Press, 1981.

E. Elmasry, ed., *Digital MOS Integrated Circuits II*, IEEE Press, 1992.

L. Glasser and D. Dopperpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, 1985.

A. Kang and Leblebici, *CMOS Digital Integrated Circuits*, 2nd Ed., McGraw-Hill, 1999.

C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

K. Martin, *Digital Integrated Circuit Design*, Oxford University Press, 2000.

D. Pucknell and K. Eshraghian, *Basic VLSI Design*, Prentice Hall, 1988.

M. Shoji, *CMOS Digital Circuit Technology*, Prentice Hall, 1988.

J. Uyemura, *Circuit Design for CMOS VLSI*, Kluwer, 1992.

H. Veendrick, *MOS IC's: From Basics to ASICS*, VCH, 1992.

Weste and Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1985, 1993.

High-Performance Design

K. Bernstein et al, *High Speed CMOS Design Styles*, Kluwer Academic, 1998.

A. Chandrakasan, F. Fox, and W. Bowhill, ed., *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2000.

M. Shoji, *High-Speed Digital Circuits*, Addison-Wesley, 1996.

Low-Power Design

A. Chandrakasan and R. Brodersen, ed., *Low-Power Digital CMOS Design*, IEEE Press, 1998.

J. Rabaey and M. Pedram, ed., *Low-Power Design Methodologies*, Kluwer Academic, 1996.

G. Yeap, *Practical Low-Power CMOS Design*, Kluwer Academic, 1998.

Memory Design

K. Itoh, *VLSI Memory Chip Design*, Springer, 2001.

B. Prince, *Semiconductor Memories*, Wiley, 1991.

B. Prince, *High Performance Memories*, Wiley, 1996.

D. Hodges, *Semiconductor Memories*, IEEE Press, 1972.

Interconnections and Packaging

H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, 1990.

W. Dally and J. Poulton, *Digital Systems Engineering*, Cambridge University Press, 1998.

E. Friedman, ed., *Clock Distribution Networks in VLSI Circuits and Systems*, IEEE Press, 1995.

J. Lau et al, ed., *Electronic Packaging: Design, Materials, Process, and Reliability*, McGraw-Hill, 1998.

Design Tools and Methodologies

V. Agrawal and S. Seth, *Test Generation for VLSI Chips*, IEEE Press, 1988.

D. Clein, *CMOS IC Layout*, Newnes, 2000.

G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

S. Rubin, *Computer Aids for VLSI Design*, Addison-Wesley, 1987.

J. Uyemura, *Physical Design of CMOS Integrated Circuits Using L-Edit*, PWS, 1995.

A. Vladimirescu, *The Spice Book*, John Wiley and Sons, 1993.

W. Wolf, *Modern VLSI Design*, Prentice Hall, 1998.

Bipolar and BiCMOS

A. Alvarez, *BiCMOS Technology and Its Applications*, Kluwer, 1989.

M. Elmasry, ed., *BiCMOS Integrated Circuit Design*, IEEE Press, 1994.

S. Embabi, A. Bellaouar, and M. Elmasry, *Digital BiCMOS Integrated Circuit Design*, Kluwer, 1993.

Lynn et al., eds., *Analysis and Design of Integrated Circuits*, McGraw-Hill, 1967.

General

J. Buchanan, *CMOS/TTL Digital Systems Design*, McGraw-Hill, 1990.

H. Haznedar, *Digital Micro-Electronics*, Benjamin/Cummings, 1991.

D. Hodges and H. Jackson, *Analysis and Design of Digital Integrated Circuits*, 2nd ed., McGraw-Hill, 1988.

M. Smith, *Application-Specific Integrated Circuits*, Addison-Wesley, 1997.

R. K. Watts, *Submicron Integrated Circuits*, Wiley, 1989.

REFERENCES

[Bardeen48] J. Bardeen and W. Brattain, “The Transistor, a Semiconductor Triode,” *Phys. Rev.*, vol. 74, p. 230, July 15, 1948.

[Beeson62] R. Beeson and H. Ruegg, “New Forms of All Transistor Logic,” *ISSCC Digest of Technical Papers*, pp. 10–11, Feb. 1962.

[Dally98] B. Dally, *Digital Systems Engineering*, Cambridge University Press, 1998.

[Faggin72] F. Faggin, M.E. Hoff, Jr, H. Feeney, S. Mazor, M. Shima, “The MCS-4 - An LSI Micro-Computer System,” 1972 IEEE Region Six Conference Record, San Diego, CA, April 19-21, 1972, pp.1-6.

[Harris56] J. Harris, “Direct-Coupled Transistor Logic Circuitry in Digital Computers,” *ISSCC Digest of Technical Papers*, p. 9, Feb. 1956.

[Hart72] C. Hart and M. Slob, “Integrated Injection Logic—A New Approach to LSI,” *ISSCC Digest of Technical Papers*, pp. 92–93, Feb. 1972.

[Hoff70] E. Hoff, “Silicon-Gate Dynamic MOS Crams 1,024 Bits on a Chip,” *Electronics*, pp. 68–73, August 3, 1970.

[Intel01] “Moore’s Law”, *http://www.intel.com/research/silicon/mooreslaw.htm*

[Masaki74] A. Masaki, Y. Harada and T. Chiba, “200-Gate ECL Master-Slice LSI,” *ISSCC Digest of Technical Papers*, pp. 62–63, Feb. 1974.

[Moore65] G. Moore, “Cramming more Components into Integrated Circuits,” Electronics, Vol. 38, Nr 8, April 1965.

[Murphy93] B. Murphy, "Perspectives on Logic and Microprocessors," *Commemorative Supplement to the Digest of Technical Papers, ISSCC Conf.*, pp. 49–51, San Francisco, 1993.

[Norman60] R. Norman, J. Last and I. Haas, "Solid-State Micrologic Elements," *ISSCC Digest of Technical Papers*, pp. 82–83, Feb. 1960.

[Hennessy96] J. Hennessy and D. Patterson, *Computer Architecture A Quantitative Approach*, Second Edition, Morgan Kaufmann Publishers, 1996

[Saleh01] R. Saleh, M. Benoit, and P, McCrorie, "Power Distribution Planning", Simplex Solutions, *http://www.simplex.com/wt/sec.php?page_name=wp_powerplan*

[Schockley49] W. Schockley, "The Theory of pn Junctions in Semiconductors and pn-Junction Transistors," *BSTJ*, vol. 28, p. 435, 1949.

[Shima74] M. Shima, F. Faggin and S. Mazor, "An N-Channel, 8-bit Single-Chip Microprocessor," *ISSCC Digest of Technical Papers*, pp. 56–57, Feb. 1974.

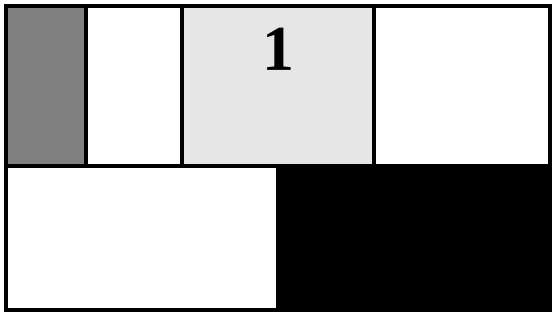
[Swade93] D. Swade, "Redeeming Charles Babbage's Mechanical Computer," *Scientific American*, pp. 86–91, February 1993.

[Wanlass63] F. Wanlass, and C. Sah, "Nanowatt logic Using Field-Effect Metal-Oxide Semiconductor Triodes," *ISSCC Digest of Technical Papers*, pp. 32–32, Feb. 1963.

1.6 Exercises

1. [E, None, 1.2] Based on the evolutionary trends described in the chapter, predict the integration complexity and the clock speed of a microprocessor in the year 2015. Determine also how much DRAM should be available on a single chip at that point in time, if Moore's law would still hold.
2. [D, None, 1.2] Visit the Intel on-line microprocessor museum (*http://www.intel.com/intel/intelis/museum/exhibit/hist_micro/index.htm*). While browsing through the microprocessor hall-of-fame, determine the rate of increase in transistor counts and clock frequencies in the 70's, 80's, and 90's. Also, create a plot of the number of transistors versus technology feature size. Spend some time browsing the site. It contains a large amount of very interesting information.
3. [D, None, 1.2] By scanning the literature, find the leading-edge devices at this point in time in the following domains: microprocessor, signal processor, SRAM, and DRAM. Determine for each of those, the number of integrated devices, the overall area and the maximum clock speed. Evaluate the match with the trends predicted in section 1.2.
4. [D, None, 1.2] Find in the library the latest November issue of the *Journal of Solid State Circuits.* For each of the papers, determine its application class (such as microprocessor, signal processor, DRAM, SRAM), the type of manufacturing technology used (MOS, bipolar, etc.), the minimum feature size, the number of devices on a single die, and the maximum clock speed. Tabulate the results along the various application classes.
5. [E, None, 1.2] Provide at least three examples for each of the abstraction levels described in Figure 1.6.

CHAPTER



INTRODUCTION

The evolution of digital circuit design

n

Compelling issues in digital circuit design

n

How to measure the quality of a design

n

Valuable references

1.1 A Historical Perspective

1.2 Issues in Digital Integrated Circuit Design

1.3 Quality Metrics of a Digital Design

1.4 Summary

1.5 To Probe Further

Exercises

1. [E, None, 1.2] Based on the evolutionary trends described in the chapter, predict the integration complexity and the clock speed of a microprocessor in the year 2015. Determine also how much DRAM should be available on a single chip at that point in time, if Moore's law would still hold.
2. [D, None, 1.2] Visit the Intel on-line microprocessor museum (*http://www.intel.com/intel/intelis/museum/exhibit/hist_micro/index.htm*). While browsing through the microprocessor hall-of-fame, determine the rate of increase in transistor counts and clock frequencies in the 70's, 80's, and 90's. Also, create a plot of the number of transistors versus technology feature size. Spend some time browsing the site. It contains a large amount of very interesting information.
3. [D, None, 1.2] By scanning the literature, find the leading-edge devices at this point in time in the following domains: microprocessor, signal processor, SRAM, and DRAM. Determine for each of those, the number of integrated devices, the overall area and the maximum clock speed. Evaluate the match with the trends predicted in section 1.2.
4. [D, None, 1.2] Find in the library the latest November issue of the *Journal of Solid State Circuits*. For each of the papers, determine its application class (such as microprocessor, signal processor, DRAM, SRAM), the type of manufacturing technology used (MOS, bipolar, etc.), the minimum feature size, the number of devices on a single die, and the maximum clock speed. Tabulate the results along the various application classes.
5. [E, None, 1.2] Provide at least three examples for each of the abstraction levels described in Figure 1.6.

More to come in the very near future!

CHAPTER 2

THE MANUFACTURING PROCESS

Overview of manufacturing process

n

Design rules

n

IC packaging

n

Future Trends in Integrated Circuit Technology

2.1 Introduction

2.2 Manufacturing CMOS Integrated Circuits

- 2.2.1 The Silicon Wafer
- 2.2.2 Photolithography
- 2.2.3 Some Recurring Process Steps
- 2.2.4 Simplified CMOS Process Flow

2.3 Design Rules — The Contract between Designer and Process Engineer

2.4 Packaging Integrated Circuits

- 2.4.1 Package Materials
- 2.4.2 Interconnect Levels
- 2.4.3 Thermal Considerations in Packaging

2.5 Perspective — Trends in Process Technology

- 2.5.1 Short-Term Developments
- 2.5.2 In the Longer Term

2.6 Summary

2.1 Introduction

Most digital designers will never be confronted with the details of the manufacturing process that lies at the core of the semiconductor revolution. Yet, some insight in the steps that lead to an operational silicon chip comes in quite handy in understanding the physical constraints that are imposed on a designer of an integrated circuit, as well as the impact of the fabrication process on issues such as cost.

In this chapter, we briefly describe the steps and techniques used in a modern integrated circuit manufacturing process. It is not our aim to present a detailed description of the fabrication technology, which easily deserves a complete course [Plummer00]. Rather we aim at presenting the general outline of the flow and the interaction between the various steps. We learn that a set of *optical masks* forms the central interface between the intrinsics of the manufacturing process and the design that the user wants to see transferred to the silicon fabric. The masks define the patterns that, when transcribed onto the different layers of the semiconductor material, form the elements of the electronic devices and the interconnecting wires. As such, these patterns have to adhere to some constraints in terms of minimum width and separation if the resulting circuit is to be fully functional. This collection of constraints is called the *design rule set*, and acts as the contract between the circuit designer and the process engineer. If the designer adheres to these rules, he gets a guarantee that his circuit will be manufacturable. An overview of the common design rules, encountered in modern CMOS processes, will be given. Finally, an overview is given of the *IC packaging* options. The package forms the interface between the circuit implemented on the silicon die and the outside world, and as such has a major impact on the performance, reliability, longevity, and cost of the integrated circuit.

2.2 Manufacturing CMOS Integrated Circuits

A simplified cross section of a typical CMOS inverter is shown in Figure 2.1. The CMOS process requires that both *n*-channel (NMOS) and *p*-channel (PMOS) transistors be built in the same silicon material. To accommodate both types of devices, special regions called *wells* must be created in which the semiconductor material is opposite to the type of the channel. A PMOS transistor has to be created in either an *n*-type substrate or an *n*-well, while an NMOS device resides in either a *p*-type substrate or a *p*-well. The cross section

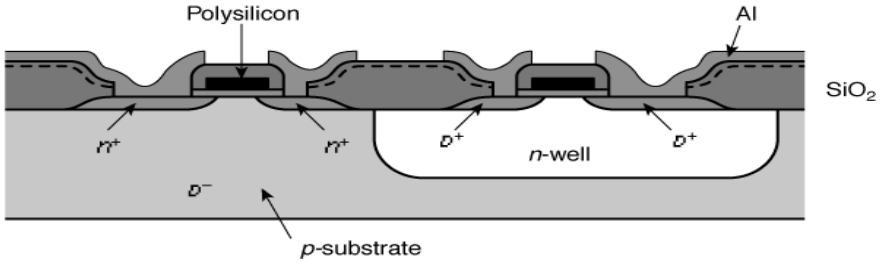


Figure 2.1 Cross section of an *n*-well CMOS process.

shown in Figure 2.1 features an *n*-well CMOS process, where the NMOS transistors are implemented in the *p*-doped substrate, and the PMOS devices are located in the *n*-well. Modern processes are increasingly using a *dual-well* approach that uses both *n*- and *p*-wells, grown on top on a epitaxial layer, as shown in Figure 2.2. We will restrict the remainder of this discussion to the latter process (without loss of generality).

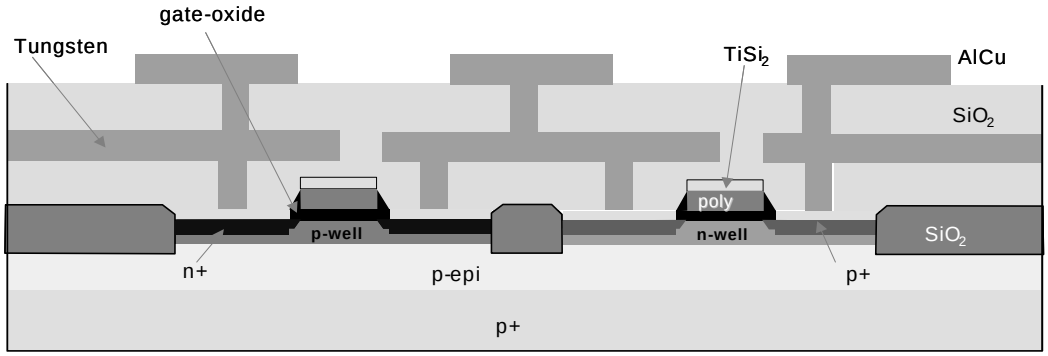


Figure 2.2 Cross section of modern dual-well CMOS process.

The CMOS process requires a large number of steps, each of which consists of a sequence of basic operations. A number of these steps and/or operations are executed very repetitively in the course of the manufacturing process. Rather than diving directly into a description of the overall process flow, we first discuss the starting material followed by a detailed perspective on some of the most-often recurring operations.

2.2.1 The Silicon Wafer

The base material for the manufacturing process comes in the form of a single-crystalline, lightly doped *wafer*. These wafers have typical diameters between 4 and 12 inches (10 and 30 cm, respectively) and a thickness of at most 1 mm, and are obtained by cutting a single-crystal ingot into thin slices (Figure 2.3). A starting wafer of the p^--type might be doped around the levels of 2×10^{21} impurities/m^3. Often, the surface of the wafer is doped more heavily, and a single crystal *epitaxial layer* of the opposite type is grown over the surface before the wafers are handed to the processing company. One important metric is the defect density of the base material. High defect densities lead to a larger fraction of non-functional circuits, and consequently an increase in cost of the final product.

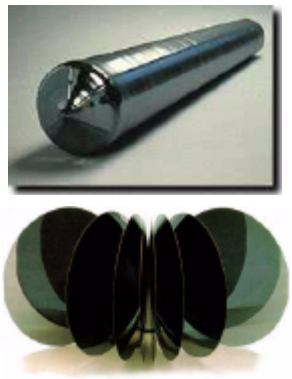


Figure 2.3 Single-crystal ingot and sliced wafers (from [Fullman99]).

2.2.2 Photolithography

In each processing step, a certain area on the chip is masked out using the appropriate optical mask so that a desired processing step can be selectively applied to the remaining regions. The processing step can be any of a wide range of tasks including oxidation, etching, metal and polysilicon deposition, and ion implantation. The technique to accomplish this selective masking, called *photolithography*, is applied throughout the manufacturing process. Figure 2.4 gives a graphical overview of the different operations involved in a typical photolitographic process. The following steps can be identified:

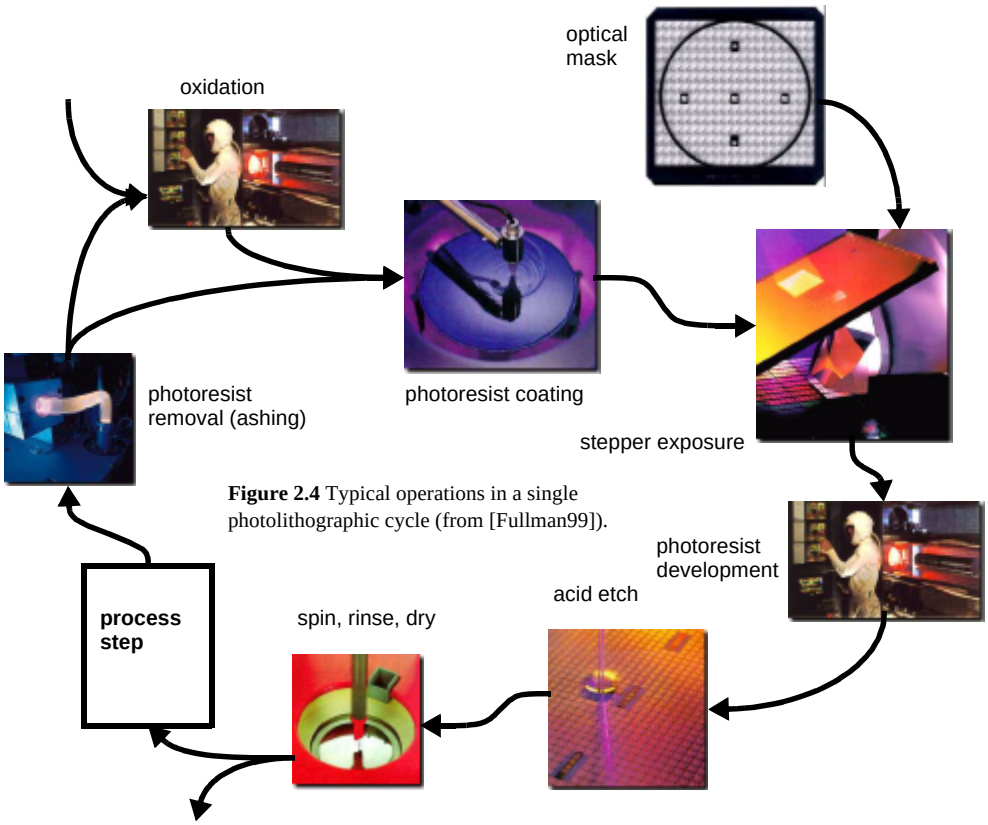


Figure 2.4 Typical operations in a single photolithographic cycle (from [Fullman99]).

1. *Oxidation layering* — this optional step deposits a thin layer of SiO_2 over the complete wafer by exposing it to a mixture of high-purity oxygen and hydrogen at approximately 1000°C. The oxide is used as an insulation layer and also forms transistor gates.

2. *Photoresist coating* — a light-sensitive polymer (similar to latex) is evenly applied while spinning the wafer to a thickness of approximately 1 µm. This material is originally soluble in an organic solvent, but has the property that the polymers cross-

link when exposed to light, making the affected regions insoluble. A photoresist of this type is called *negative*. A positive photoresist has the opposite properties; originally insoluble, but soluble after exposure. By using both positive and negative resists, a single mask can sometimes be used for two steps, making complementary regions available for processing. Since the cost of a mask is increasing quite rapidly with the scaling of technology, a reduction of the number of masks is surely of high priority.

3. *Stepper exposure* — a glass mask (or reticle), containing the patterns that we want to transfer to the silicon, is brought in close proximity to the wafer. The mask is opaque in the regions that we want to process, and transparent in the others (assuming a negative photoresist). The glass mask can be thought of as the negative of one layer of the microcircuit. The combination of mask and wafer is now exposed to ultra-violet light. Where the mask is transparent, the photoresist becomes insoluble.

4. *Photoresist development and bake* — the wafers are developed in either an acid or base solution to remove the non-exposed areas of photoresist. Once the exposed photoresist is removed, the wafer is "soft-baked" at a low temperature to harden the remaining photoresist.

5. *Acid Etching* — material is selectively removed from areas of the wafer that are not covered by photoresist. This is accomplished through the use of many different types of acid, base and caustic solutions as a function of the material that is to be removed. Much of the work with chemicals takes place at large wet benches where special solutions are prepared for specific tasks. Because of the dangerous nature of some of these solvents, safety and environmental impact is a primary concern.

6. *Spin, rinse, and dry* — a special tool (called SRD) cleans the wafer with deionized water and dries it with nitrogen. The microscopic scale of modern semiconductor devices means that even the smallest particle of dust or dirt can destroy the circuitry. To prevent this from happening, the processing steps are performed in ultra-clean rooms where the number of dust particles per cubic foot of air ranges between 1 and 10. Automatic wafer handling and robotics are used whenever possible. This explains why the cost of a state-of-the-art fabrication facility easily ranges in the multiple billions of dollars. Even then, the wafers must be constantly cleaned to avoid contamination, and to remove the left-over of the previous process steps.

7. *Various process steps* — the exposed area can now be subjected to a wide range of process steps, such as ion implantation, plasma etching, or metal deposition. These are the subjects of the subsequent section.

8. *Photoresist removal (or ashing)* — a high-temperature plasma is used to selectively remove the remaining photoresist without damaging device layers.

We illustrate the use of the photolitographic process for one specific example, the patterning of a layer of SiO_2, in Figure 2.5. The sequence of process steps shown in the Figure patterns exactly one layer of the semiconductor material, and may seem very complex. Yet, the reader has to bear in mind that same sequence patterns the layer of **the complete surface of the wafer**. It is hence a very parallel process, transferring hundreds of

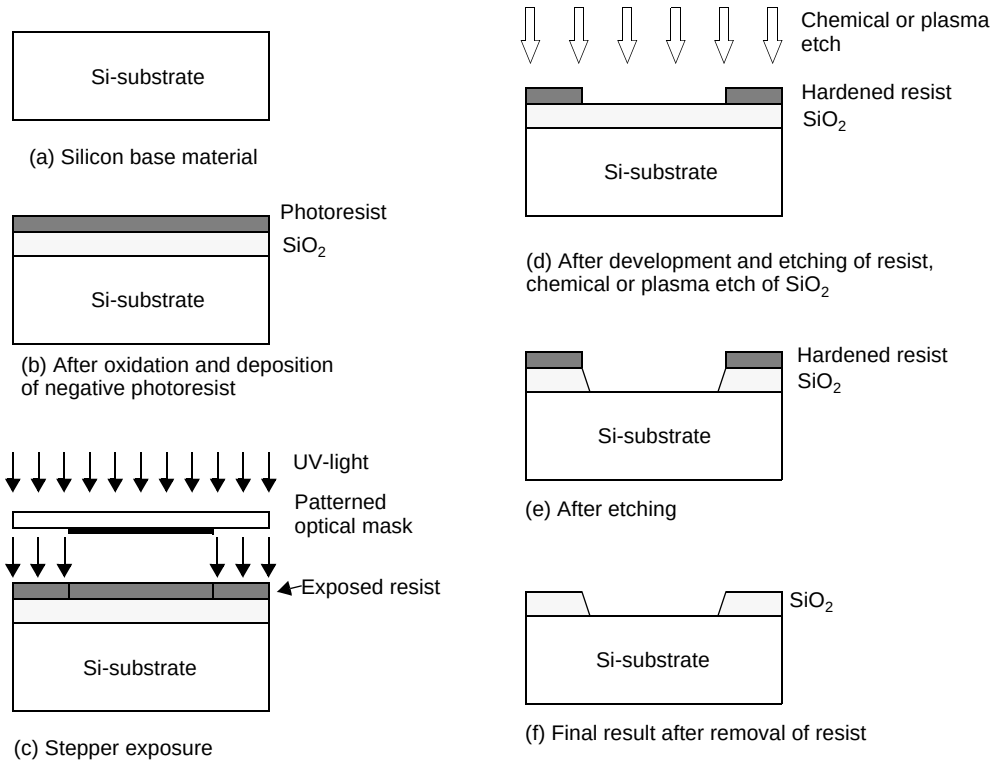


Figure 2.5 Process steps for patterning of SiO_2.

millions of patterns to the semiconductor surface simultaneously. The concurrent and scalable nature of the optolithographical process is what makes the cheap manufacturing of complex semiconductor circuits possible, and lies at the core of the economic success of the semiconductor industry.

The continued scaling of the minimum feature sizes in integrated circuits puts an enormous burden on the developer of semiconductor manufacturing equipment. This is especially true for the optolithographical process. The dimensions of the features to be transcribed surpass the wavelengths of the optical light sources, so that achieving the necessary resolution and accuracy becomes harder and harder. So far, engineering engineering has extended the lifetime of this process at least until the 100 nm (or 0.1 μm) process generation. Techniques such as optical-mask correction (OPC) pre-warp the drawn patterns to account for the diffraction phenomena, encountered when printing close to the limits of optical lithography. This adds substantially to the cost of mask making. In the foreseeable future, other solutions that offer a a finer resolution such as extreme-ultraviolet (EUV), X-ray or electron-beam may be needed. These techniques, while fully functional, are currently less attractive from an economic viewpoint.

2.2.3 Some Recurring Process Steps

Diffusion and Ion Implantation

Many steps of the integrated circuit manufacturing process require a chance in the dopant concentration of some parts of the material. The creation of the source and drain regions, well and substrate contacts, the doping of the polysilicon, and the adjustments of the device threshold are examples of such. There exist two approaches for introducing these dopants—diffusion and ion implantation. In both techniques, the area to be doped is exposed, while the rest of the wafer is coated with a layer of buffer material, typically SiO_2.

In *diffusion implantation*, the wafers are placed in a quartz tube embedded in a heated furnace. A gas containing the dopant is introduced in the tube. The high temperatures of the furnace, typically 900 to 1100 °C, cause the dopants to diffuse into the exposed surface both vertically and horizontally. The final dopant concentration is the greatest at the surface and decreases in a gaussian profile deeper in the material.

In *ion implantation*, dopants are introduced as ions into the material. The ion implantation system directs and sweeps a beam of purified ions over the semiconductor surface. The acceleration of the ions determines how deep they will penetrate the material, while the beam current and the exposure time determine the dosage. The ion implantation method allows for an independent control of depth and dosage. This is the reason that ion implantation has largely displaced diffusion in modern semiconductor manufacturing.

Ion implantation has some unfortunate side effects however, the most important one being lattice damage. Nuclear collisions during the high energy implantation cause the displacement of substrate atoms, leading to material defects. This problem is largely resolved by applying a subsequent *annealing* step, in which the wafer is heated to around 1000°C for 15 to 30 minutes, and then allowed to cool slowly. The heating step thermally vibrates the atoms, which allows the bonds to reform.

Deposition

Any CMOS process requires the repetitive deposition of layers of a material over the complete wafer, to either act as buffers for a processing step, or as insulating or conducting layers. We have already discussed the oxidation process, which allows a layer of SiO_2 to be grown. Other materials require different techniques. For instance, silicon nitride (Si_3N_4) is used as a sacrificial buffer material during the formation of the field oxide and the introduction of the stopper implants. This silicon nitride is deposited everywhere using a process called *chemical vapor deposition* or CVD, which uses a gas-phase reaction with energy supplied by heat at around 850°C.

Polysilicon, on the other hand, is deposited using a chemical deposition process, which flows silane gas over the heated wafer coated with SiO_2 at a temperature of approximately 650°C. The resulting reaction produces a non-crystalline or amorphous material called polysilicon. To increase to conductivity of the material, the deposition has to be followed by an implantation step.

The Aluminum interconnect layers are typically deployed using a process known as *sputtering*. The aluminum is evaporated in a vacuum, with the heat for the evaporation

delivered by electron-beam or ion-beam bombarding. Other metallic interconnect materials such as Copper require different deposition techniques.

Etching

Once a material has been deposited, etching is used to selectively form patterns such as wires and contact holes. The *wet etching* process was described earlier, and makes use of acid or basic solutions. For instance, hydrofluoric acid buffered with ammonium fluoride is typically used to etch SiO_2.

In recent years, *dry* or *plasma etching* has made a lot of inroad. A wafer is placed into the etch tool's processing chamber and given a negative electrical charge. The chamber is heated to 100°C and brought to a vacuum level of 7.5 Pa, then filled with a positively charged plasma (usually a mix of nitrogen, chlorine and boron trichloride). The opposing electrical charges cause the rapidly moving plasma molecules to align themselves in a vertical direction, forming a microscopic chemical and physical “sandblasting” action which removes the exposed material. Plasma etching has the advantage of offering a well-defined directionality to the etching action, creating patterns with sharp vertical contours.

Planarization

To reliably deposit a layer of material onto the semiconductor surface, it is essential that the surface is approximately flat. If no special steps were taken, this would definitely not be the case in modern CMOS processes, where multiple patterned metal interconnect layers are superimposed onto each other. Therefore, a *chemical-mechanical planarization* (CMP) step is included before the deposition of an extra metal layer on top of the insulating SiO_2 layer. This process uses a slurry compound—a liquid carrier with a suspended abrasive component such as aluminum oxide or silica—to microscopically plane a device layer and to reduce the step heights.

2.2.4 Simplified CMOS Process Flow

The gross outline of a potential CMOS process flow is given in Figure 2.6. The process starts with the definition of the *active regions*, this is the regions where transistors will be constructed. All other areas of the die will be covered with a thick layer of silicon dioxide (SiO_2), called the *field oxide*. This oxide acts as the insulator between neighboring devices, and is either grown (as in the process of Figure 2.1), or deposited in etched trenches (Figure 2.2) — hence the name *trench insulation*. Further insulation is provided by the addition of a reverse-biased *np*-diode, formed by adding an extra p^+ region, called the *channel-stop implant* (or *field implant*) underneath the field oxide. Next, lightly doped *p*- and *n*-wells are formed through ion implantation. To construct an NMOS transistor in a *p*-well, heavily doped *n*-type *source* and *drain* regions are implanted (or diffused) into the lightly doped *p*-type substrate. A thin layer of SiO_2, called the *gate oxide*, separates the region between the source and drain, and is itself covered by conductive polycrystalline silicon (or polysilicon, for short). The conductive material forms the *gate* of the transistor. PMOS transistors are constructed in an *n*-well in a similar fashion (just reverse *n*’s and

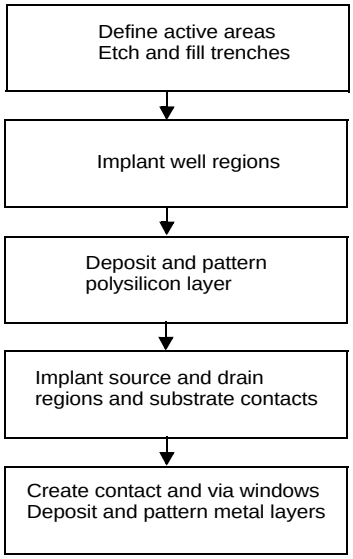
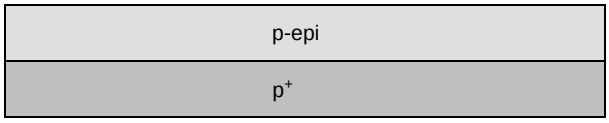


Figure 2.6 Simplified process sequence for the manufacturing of a n-dual-well CMOS circuit.

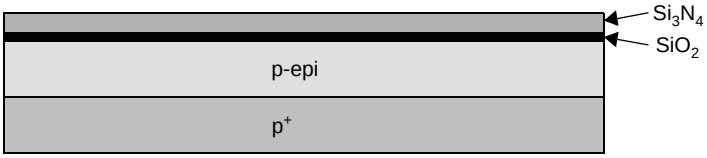
p's). Multiple insulated layers of metallic (most often Aluminum) wires are deposited on top of these devices to provide for the necessary interconnections between the transistors.

A more detailed breakdown of the flow into individual process steps and their impact on the semiconductor material is shown graphically in Figure 2.7. While most of the operations should be self-explanatory in light of the previous descriptions, some comments on individual operations are worthwhile. The process starts with a *p*-substrate surfaced with a lightly doped *p*-epitaxial layer (a). A thin layer of SiO_2 is deposited, which will serve as the gate oxide for the transistors, followed by a deposition of a thicker sacrificial silicon nitride layer (b). A plasma etching step using the complementary of the active area mask creates the trenches, used for insulating the devices (c). After providing the channel stop implant, the trenches are filled with SiO_2 followed by a number of steps to provide a flat surface (including inverse active pattern oxide etching, and chemical-mechanical planarization). At that point, the sacrificial nitride is removed (d). The *n*-well mask is used to expose only the *n*-well areas (the rest of the wafer is covered by a thick buffer material), after which an implant-annealing sequence is applied to adjust the well-doping. This is followed by a second implant step to adjust the threshold voltages of the PMOS transistors. This implant only impacts the doping in the area just below the gate oxide (e). Similar operations (using other dopants) are performed to create the p-wells, and to adjust the thresholds of the NMOS transistors (f). A thin layer of polysilicon is chemically deposited, and patterned with the aid of the polysilicon mask. Polysilicon is used both as gate electrode material for the transistors as well as an interconnect medium (g). Consecutive ion implantations are used to dope the source and drain regions of the PMOS (p^+) and NMOS (n^+) transistors, respectively (h), after which the thin gate oxide not covered by the polysilicon is etched away[1]. The same implants are also used to dope

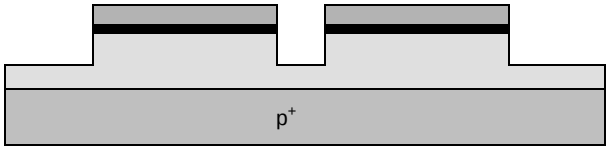
[1] Most modern processes also include extra implants for the creation of the lightly-doped drain regions (LDD), and the creation of gate spacers at this point. We have omitted these for the sake of simplicity.



(a) Base material: p+ substrate with p-epi layer

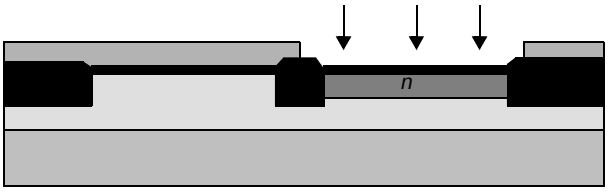


(b) After deposition of gate-oxide sacrificial nitride (acts as a buffer layer)

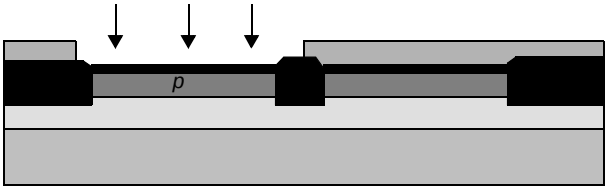


(c) After plasma etch of insulating trenches using the inverse of the active area mask

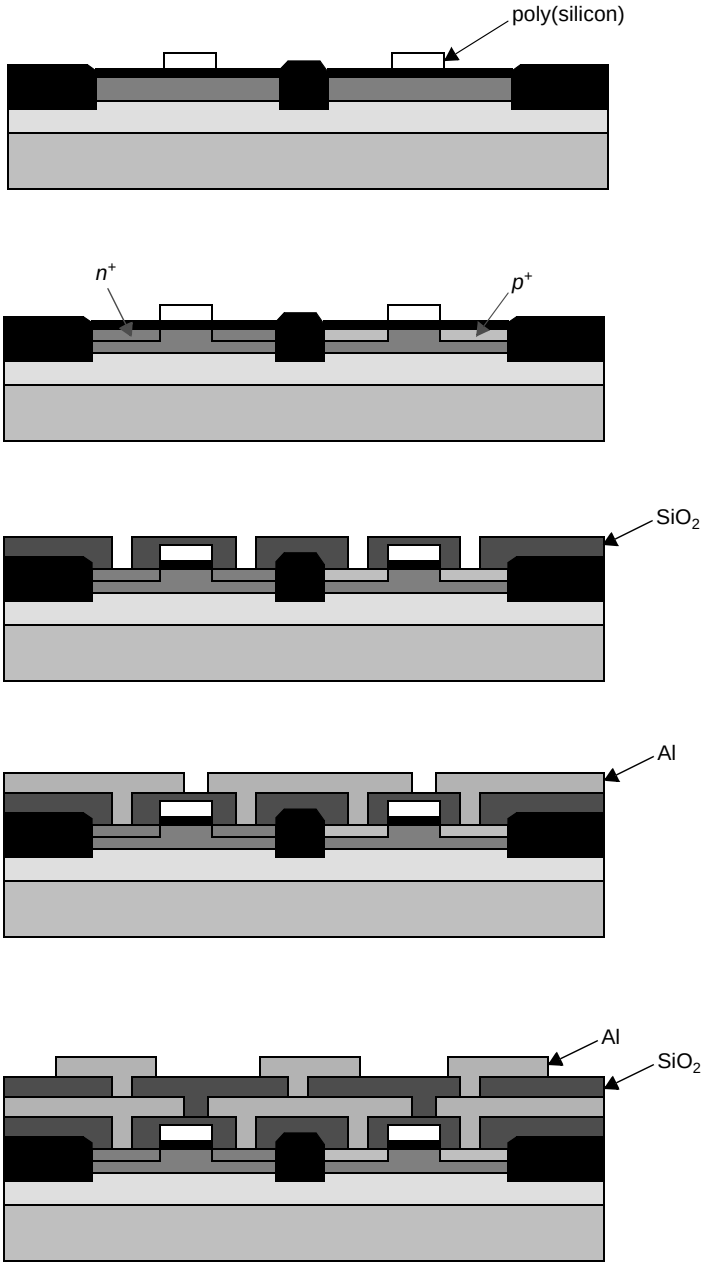
(d) After trench filling, CMP planarization, and removal of sacrificial nitride



(e) After *n*-well and V_{Tp} adjust implants



(f) After *p*-well and V_{Tn} adjust implants



(g) After polysilicon deposition and etch

(h) After n^+ source/drain and p^+ source/drain implants. These steps also dope the polysilicon.

(i) After deposition of SiO_2 insulator and contact hole etch.

(j) After deposition and patterning of first Al layer.

(k) After deposition of SiO_2 insulator, etching of via's, deposition and patterning of second layer of Al.

Figure 2.7 Process flow for the fabrication of an NMOS and a PMOS transistor in a dual-well CMOS process. Be aware that the drawings are stylized for understanding, and that the aspects ratios are not proportioned to reality.

the polysilicon on the surface, reducing its resistivity. Undoped polysilicon has a very high resistivity. Note that the polysilicon gate, which is patterned before the doping, actually defines the precise location of the channel region, and hence the location of the source and drain regions. This procedure allows for a very precise positioning of the two regions relative to the gate, and hence is called the *self-aligned process*. The process continues with the deposition of the metallic interconnect layers. These consists of a repetition of the following steps (i-k): deposition of the insulating material (most often SiO_2), etching of the contact or via holes, deposition of the metal (most often Aluminum and Copper, although Tungsten is often used for the lower layers), and patterning of the metal. Intermediate planarization steps ensure that the surface remains reasonable flat, even in the presence of multiple interconnect layers. After the last level of metal is deposited, a final passivation or *overglass* is deposited for protection. This layer would be CVD SiO_2, although often an additional layer of nitride is deposited as it is more impervious to moisture. The final processing step is to etch openings to the pads used for bonding.

A cross-section of the final artifact is shown in Figure 2.8. Observe how the transistors occupy only a small fraction of the total height of the structure. The interconnect layers take up the majority of the vertical dimension.

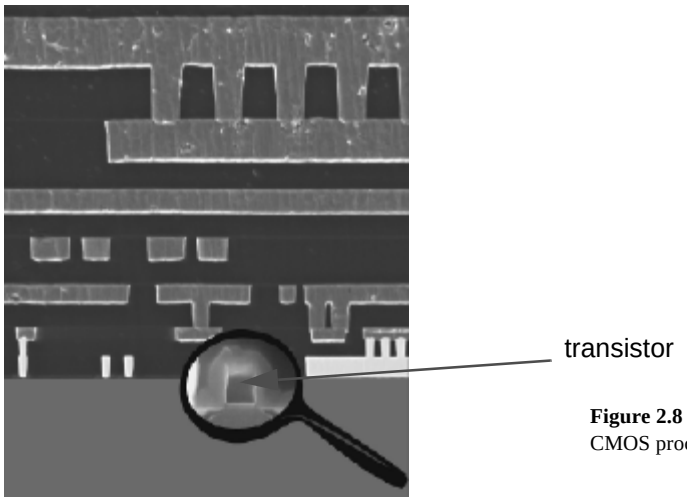


Figure 2.8 Cross-section of state-of-the-art CMOS process.

2.3 Design Rules — The Contract between Designer and Process Engineer

As processes become more complex, requiring the designer to understand the intricacies of the fabrication process and interpret the relations between the different masks is a sure road to trouble. The goal of defining a set of design rules is to allow for a ready translation of a circuit concept into an actual geometry in silicon. The design rules act as the interface or even the contract between the circuit designer and the process engineer.

Circuit designers in general want tighter, smaller designs, which lead to higher performance and higher circuit density. The process engineer, on the other hand, wants a reproducible and high-yield process. Design rules are, consequently, a compromise that attempts to satisfy both sides.

The design rules provide a set of guidelines for constructing the various masks needed in the patterning process. They consist of minimum-width and minimum-spacing constraints and requirements between objects on the same or on different layers.

The fundamental unity in the definition of a set of design rules is the *minimum line width*. It stands for the minimum mask dimension that can be safely transferred to the semiconductor material. In general, the minimum line width is set by the resolution of the patterning process, which is most commonly based on optical lithography. More advanced approaches use electron-beam. EUV or X-ray sources that offer a finer resolution, but are less attractive from an economical viewpoint today.

Even for the same minimum dimension, design rules tend to differ from company to company, and from process to process. This makes porting an existing design between different processes a time-consuming task. One approach to address this issue is to use advanced CAD techniques, which allow for migration between compatible processes. Another approach is to use *scalable design rules*. The latter approach, made popular by Mead and Conway [Mead80], defines all rules as a function of a single parameter, most often called λ. The rules are chosen so that a design is easily ported over a cross section of industrial processes. Scaling of the minimum dimension is accomplished by simply changing the value of λ. This results in a *linear scaling* of all dimensions. For a given process, λ is set to a specific value, and all design dimensions are consequently translated into absolute numbers. Typically, the minimum line width of a process is set to 2λ. For instance, for a 0.25 μm process (i.e., a process with a minimum line width of 0.25 μm), λ equals 0.125 μm.

This approach, while attractive, suffers from some disadvantages:

1. Linear scaling is only possible over a limited range of dimensions (for instance, between 0.25 μm and 0.18 μm). When scaling over larger ranges, the relations between the different layers tend to vary in a nonlinear way that cannot be adequately covered by the linear scaling rules.

2. Scalable design rules are conservative. As they represent a cross section over different technologies, they have to represent the worst-case rules for the whole set. This results in over-dimensioned and less-dense designs.

For these reasons, scalable design rules are normally avoided by industry.[2] As circuit density is a prime goal in industrial designs, most semiconductor companies tend to use *micron rules*, which express the design rules in absolute dimensions and can therefore exploit the features of a given process to a maximum degree. Scaling and porting designs between technologies under these rules is more demanding and has to be performed either manually or using advanced CAD tools.

For this textbook, we have selected a "vanilla" 0.25 μm CMOS process as our preferred implementation medium. The rest of this section is devoted to a short introduction and overview of the design rules of this process, which fall in the micron-rules class. A complete design-rule set consists of the following entities: a set of layers, relations

[2] While not entirely accurate, lambda rules are still useful to estimate the impact of a technology scale on the area of a design.

between objects on the same layer, and relations between objects on different layers. We discuss each of them in sequence.

Layer Representation

The layer concept translates the intractable set of masks currently used in CMOS into a simple set of conceptual layout levels that are easier to visualize by the circuit designer. From a designer's viewpoint, all CMOS designs are based on the following entities:

- S*ubstrates* and/or *wells*, being *p*-type (for NMOS devices) and *n*-type (for PMOS)
- *Diffusion regions* (n^+ and p^+) defining the areas where transistors can be formed. These regions are often called the *active areas*. Diffusions of an inverse type are needed to implement contacts to the wells or to the substrate. These are called *select regions*.
- One or more *polysilicon* layers, which are used to form the gate electrodes of the transistors (but serve as interconnect layers as well).
- A number of *metal interconnect* layers.
- *Contact and via* layers to provide interlayer connections.

A layout consists of a combination of polygons, each of which is attached to a certain layer. The functionality of the circuit is determined by the choice of the layers, as well as the interplay between objects on different layers. For instance, an MOS transistor is formed by the cross section of the diffusion layer and the polysilicon layer. An interconnection between two metal layers is formed by a cross section between the two metal layers and an additional contact layer. To visualize these relations, each layer is assigned a standard color (or stipple pattern for a black-and-white representation). The different layers used in our CMOS process are represented in Colorplate 1 (color insert).

Intralayer Constraints

A first set of rules defines the minimum dimensions of objects on each layer, as well as the minimum spacings between objects on the same layer. All distances are expressed in μm. These constraints are presented in a pictorial fashion in Colorplate 2.

Interlayer Constraints

Interlayer rules tend to be more complex. The fact that multiple layers are involved makes it harder to visualize their meaning or functionality. Understanding layout requires the capability of translating the two-dimensional picture of the layout drawing into the three-dimensional reality of the actual device. This takes some practice.

We present these rules in a set of separate groupings.

1. *Transistor Rules* (Colorplate 3). A transistor is formed by the overlap of the active and the polysilicon layers. From the intralayer design rules, it is already clear that the minimum length of a transistor equals 0.24 μm (the minimum width of polysilicon), while its width is at least 0.3 μm (the minimum width of diffusion). Extra rules

include the spacing between the active area and the well boundary, the gate overlap of the active area, and the active overlap of the gate.

2. *Contact and Via Rules* (Colorplates 2 and 4). A contact (which forms an interconnection between metal and active or polysilicon) or a via (which connects two metal layers) is formed by overlapping the two interconnecting layers and providing a contact hole, filled with metal, between the two. In our process, the minimum size of the contact hole is 0.3 μm, while the polysilicon and diffusion layers have to extend at least over 0.14 μm beyond the area of the contact hole. This sets the minimum area of a contact to 0.44 μm × 0.44 μm. This is larger than the dimensions of a minimum-size transistor! Excessive changes between interconnect layers in routing are thus to be avoided. The figure, furthermore, points out the minimum spacings between contact and via holes, as well as their relationship with the surrounding layers.

Well and Substrate Contacts (Colorplate 5). For robust digital circuit design, it is important for the well and substrate regions to be adequately connected to the supply voltages. Failing to do so results in a resistive path between the substrate contact of the transistors and the supply rails, and can lead to possibly devastating parasitic effects, such as latchup. It is therefore advisable to provide numerous substrate (well) contacts spread over the complete region. To establish an ohmic contact between a supply rail, implemented in metal1, and a *p*-type material, a p^+ diffusion region must be provided. This is enabled by the *select* layer, which reverses the type of diffusion. A number of rules regarding the use of the *select layer* are illustrated in Colorplate 5.

Consider an *n*-well process, which implements the PMOS transistors into an *n*-type well diffused in a *p*-type material. The nominal diffusion is p^+. To invert the polarity of the diffusion, an *n*-select layer is provided that helps to establish the n^+ diffusions for the well-contacts in the *n*-region as well as the n^+ source and drain regions for the NMOS transistors in the substrate.

Verifying the Layout

Ensuring that none of the design rules is violated is a fundamental requirement of the design process. Failing to do so will almost surely lead to a nonfunctional design. Doing so for a complex design that can contain millions of transistors is no sinecure, especially when taking into account the complexity of some design-rule sets. While design teams in the past used to spend numerous hours staring at room-size layout plots, most of this task is now done by computers. Computer-aided *Design-Rule Checking* (called *DRC*) is an integral part of the design cycle for virtually every chip produced today. A number of layout tools even perform *on-line DRC* and check the design in the background during the time of conception.

Example 2.1 Layout Example

An example of a complete layout containing an inverter is shown in Figure 2.9. To help the visualization process, a vertical cross section of the process along the design center is included as well as a circuit schematic.

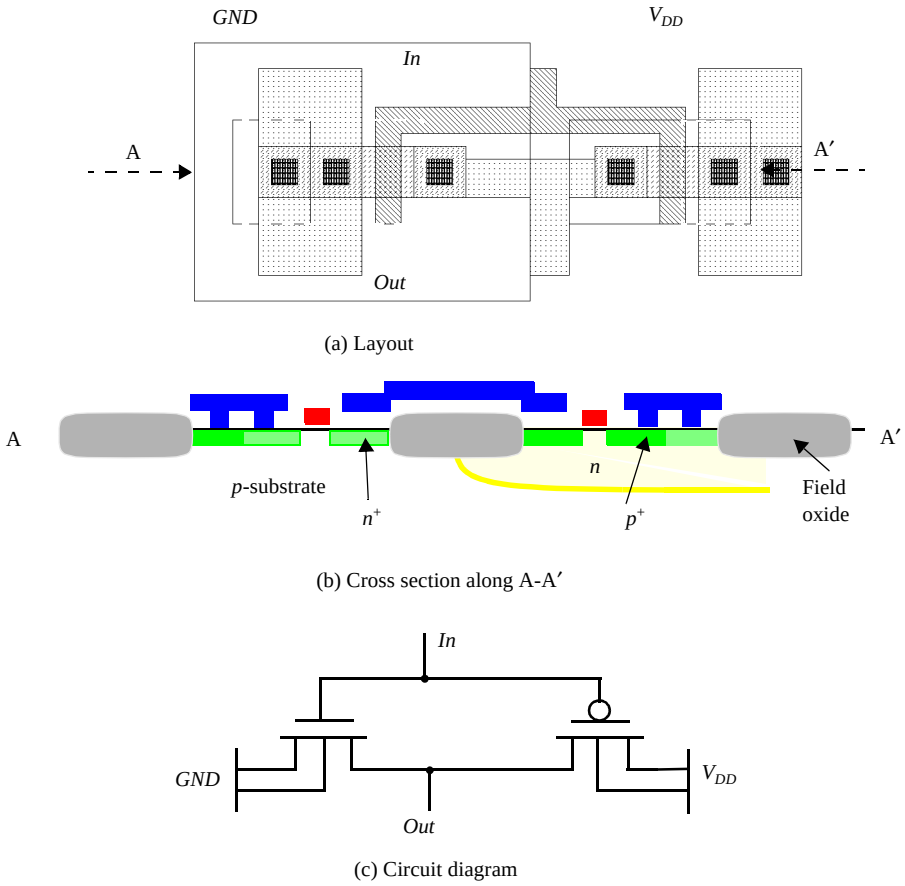


Figure 2.9 A detailed layout example, including vertical process cross section and circuit diagram.

It is left as an exercise for the reader to determine the sizes of both the NMOS and the PMOS transistors.

2.4 Packaging Integrated Circuits

The IC package plays a fundamental role in the operation and performance of a component. Besides providing a means of bringing signal and supply wires in and out of the silicon die, it also removes the heat generated by the circuit and provides mechanical support. Finally, its also protects the die against environmental conditions such as humidity.

The packaging technology furthermore has a major impact on the performance and power-dissipation of a microprocessor or signal processor. This influence is getting more pronounced as time progresses by the reduction in internal signal delays and on-chip capacitance as a result of technology scaling. Up to 50% of the delay of a high-performance computer is currently due to packaging delays, and this number is expected to rise.

The search for higher-performance packages with fewer inductive or capacitive parasitics has accelerated in recent years.

The increasing complexity of what can be integrated on a single die also translates into a need for ever more input-output pins, as the number of connections going off-chip tends to be roughly proportional to the complexity of the circuitry on the chip. This relationship was first observed by E. Rent of IBM (published in [Landman71]), who translated it into an empirical formula that is appropriately called *Rent's rule*. This formula relates the number of input/output pins to the complexity of the circuit, as measured by the number of gates.

$$P = K \times G^{\beta} \tag{2.1}$$

where K is the average number of I/Os per gate, G the number of gates, β the Rent exponent, and P the number of I/O pins to the chip. β varies between 0.1 and 0.7. Its value depends strongly upon the application area, architecture, and organization of the circuit, as demonstrated in Table 2.1. Clearly, microprocessors display a very different input/output behavior compared to memories.

Table 2.1 Rent's constant for various classes of systems ([Bakoglu90])

Application	β	K
Static memory	0.12	6
Microprocessor	0.45	0.82
Gate array	0.5	1.9
High-speed computer (chip)	0.63	1.4
High-speed computer (board)	0.25	82

The observed rate of pin-count increase for integrated circuits varies between 8% to 11% per year, and it has been projected that packages with more than 2000 pins will be required by the year 2010. For all these reasons, traditional dual-in-line, through-hole mounted packages have been replaced by other approaches such as surface-mount, ball-grid array, and multichip module techniques. It is useful for the circuit designer to be aware of the available options, and their pros and cons.

Due to its multi-functionality, a good package must comply with a large variety of requirements.

- **Electrical requirements**—Pins should exhibit low capacitance (both interwire and to the substrate), resistance, and inductance. A large characteristic impedance should be tuned to optimize transmission line behavior. Observe that intrinsic integrated-circuit impedances are high.
- **Mechanical and thermal properties**—The heat-removal rate should be as high as possible. Mechanical reliability requires a good matching between the thermal properties of the die and the chip carrier. Long-term reliability requires a strong connection from die to package as well as from package to board.

- **Low Cost**—Cost is always one of the more important properties. While ceramics have a superior performance over plastic packages, they are also substantially more expensive. Increasing the heat removal capacity of a package also tends to raise the package cost. The least expensive plastic packaging can dissipate up to 1 W. Somewhat more expensive, but still cheap, plastic packages can dissipate up to 2W. Higher dissipation requires more expensive ceramic packaging. Chips dissipating over 50 W require special heat sink attachments. Even more extreme techniques such as fans and blowers, liquid cooling hardware, or heat pipes, are needed for higher dissipation levels.
  Packing density is a major factor in reducing board cost. The increasing pin count either requires an increase in the package size or a reduction in the pitch between the pins. Both have a profound effect on the packaging economics.

Packages can be classified in many different ways —by their main material, the number of interconnection levels, and the means used to remove heat. In this short section, we can only glance briefly at each of those issues.

2.4.1 Package Materials

The most common materials used for the package body are ceramic and polymers (plastics). The latter have the advantage of being substantially cheaper, but suffer from inferior thermal properties. For instance, the ceramic Al_2O_3 (Alumina) conducts heat better than SiO_2 and the Polyimide plastic, by factors of 30 and 100 respectively. Furthermore, its thermal expansion coefficient is substantially closer to the typical interconnect metals. The disadvantage of alumina and other ceramics is their high dielectric constant, which results in large interconnect capacitances.

2.4.2 Interconnect Levels

The traditional packaging approach uses a two-level interconnection strategy. The die is first attached to an individual chip carrier or substrate. The package body contains an internal cavity where the chip is mounted. These cavities provide ample room for many connections to the chip leads (or pins). The leads compose the second interconnect level and connect the chip to the global interconnect medium, which is normally a PC board. Complex systems contain even more interconnect levels, since boards are connected together using backplanes or ribbon cables. The first two layers of the interconnect hierarchy are illustrated in the drawing of Figure 2.10. The following sections provide a brief overview of the interconnect techniques used at levels one and two of the interconnect hierarchy, followed by a short discussion of some more advanced packaging approaches.

Interconnect Level 1 —Die-to-Package-Substrate

For a long time, *wire bonding* was the technique of choice to provide an electrical connection between die and package. In this approach, the backside of the die is attached to the substrate using glue with a good thermal conductance. Next, the chip pads are individually connected to the lead frame with aluminum or gold wires. The wire-bonding machine use

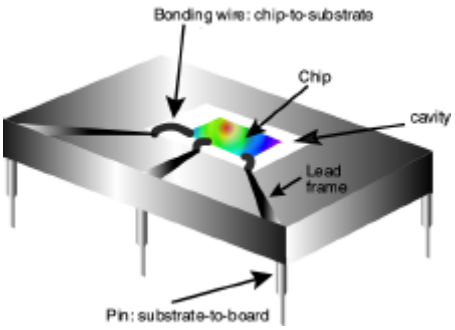


Figure 2.10 Interconnect hierarchy in traditional IC packaging.

for this purpose operates much like a sewing machine. An example of wire bonding is shown in Figure 2.11. Although the wire-bonding process is automated to a large degree, it has some major disadvantages.

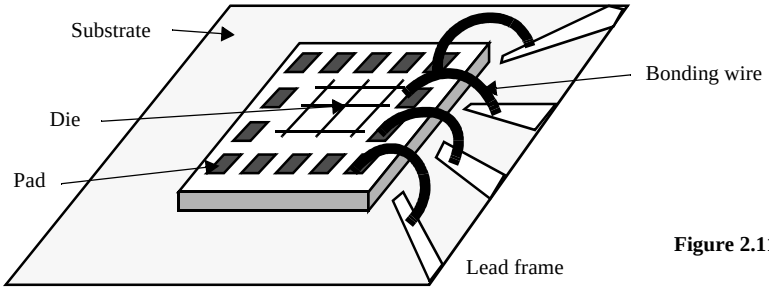


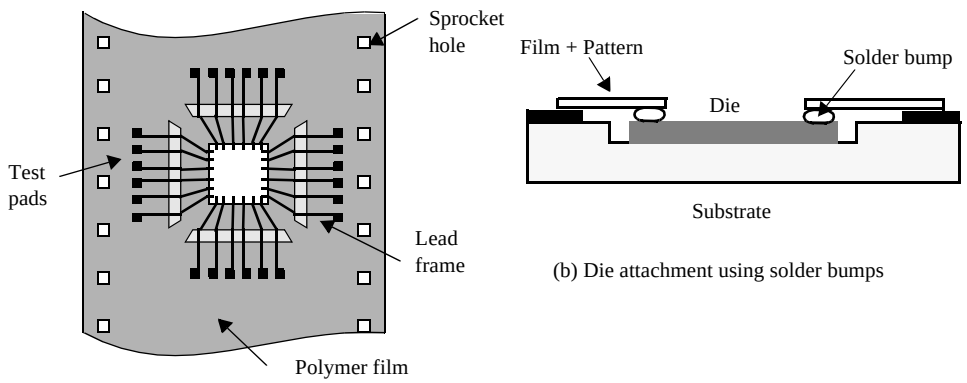
Figure 2.11 Wire bonding.

1. Wires must be attached serially, one after the other. This leads to longer manufacturing times with increasing pin counts.
2. Larger pin counts make it substantially more challenging to find bonding patterns that avoid shorts between the wires.

Bonding wires have inferior electrical properties, such as a high individual inductance (5 nH or more) and mutual inductance with neighboring signals. The inductance of a bonding wire is typically about 1 nH/mm, while the inductance per package pin ranges between 7 and 40 nH per pin depending on the type of package as well as the positioning of the pin on the package boundary [Steidel83].Typical values of the parasitic inductances and capacitances for a number of commonly used packages are summarized in Table 2.2.

3. The exact value of the parasitics is hard to predict because of the manufacturing approach and irregular outlay.

New attachment techniques are being explored as a result of these deficiencies. In one approach, called *Tape Automated Bonding* (or TAB), the die is attached to a metal lead frame that is printed on a polymer film (typically polyimide) (Figure 2.12a). The connection between chip pads and polymer film wires is made using solder bumps (Figure 2.12b). The tape can then be connected to the package body using a number of techniques. One possible approach is to use pressure connectors.



(a) Polymer tape with imprinted wiring pattern

(b) Die attachment using solder bumps

Figure 2.12 Tape-automated bonding (TAB).

Table 2.2 Typical capacitance and inductance values of package and bonding styles (from [Steidel83] and [Franzon93]).

Package Type	Capacitance (pF)	Inductance (nH)
68-pin plastic DIP	4	35
68-pin ceramic DIP	7	20
256-pin grid array	1–5	2–15
Wire bond	0.5–1	1–2
Solder bump	0.1–0.5	0.01–0.1

The advantage of the TAB process is that it is highly automated. The sprockets in the film are used for automatic transport. All connections are made simultaneously. The printed approach helps to reduce the wiring pitch, which results in higher lead counts. Elimination of the long bonding wires improves the electrical performance. For instance, for a two-conductor layer, 48 mm TAB Circuit, the following electrical parameters hold: $L \approx 0.3$–0.5 nH, $C \approx 0.2$–0.3 pF, and $R \approx 50$–$200\ \Omega$ [Doane93, p. 420].

Another approach is to flip the die upside-down and attach it directly to the substrate using solder bumps. This technique, called *flip-chip* mounting, has the advantage of a superior electrical performance (Figure 2.13). Instead of making all the I/O connections on the die boundary, pads can be placed at any position on the chip. This can help address the power- and clock-distribution problems, since the interconnect materials on the substrate (e.g., Cu or Au) are typically of a better quality than the Al on the chip.

Interconnect Level 2—Package Substrate to Board

When connecting the package to the PC board, *through-hole mounting* has been the packaging style of choice. A PC board is manufactured by stacking layers of copper and insu-

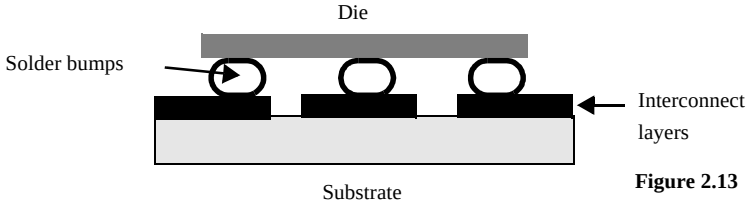
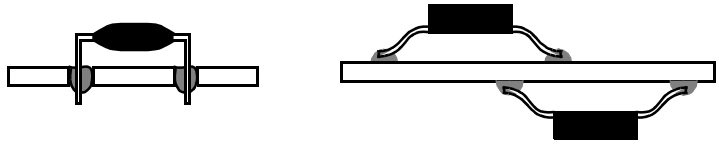


Figure 2.13 Flip-chip bonding.

lating epoxy glass. In the through-hole mounting approach, holes are drilled through the board and plated with copper. The package pins are inserted and electrical connection is made with solder (Figure 2.14a). The favored package in this class was the *dual-in-line* package or DIP (Figure 2.15a). The packaging density of the DIP degrades rapidly when the number of pins exceeds 64. This problem can be alleviated by using the *pin-grid-array* (PGA) package that has leads on the entire bottom surface instead of only on the periphery (Figure 2.15b). PGAs can extend to large pin counts (over 400 pins are possible).



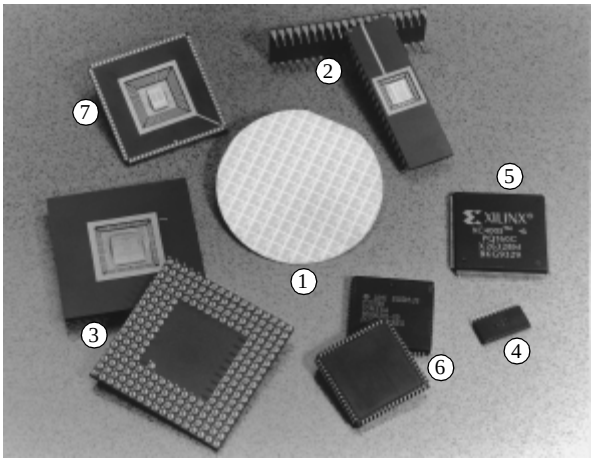
(a) Through-hole mounting (b) Surface mount

Figure 2.14 Board-mounting approaches.

The through-hole mounting approach offers a mechanically reliable and sturdy connection. However, this comes at the expense of packaging density. For mechanical reasons, a minimum pitch of 2.54 mm between the through-holes is required. Even under those circumstances, PGAs with large numbers of pins tend to substantially weaken the board. In addition, through-holes limit the board packing density by blocking lines that might otherwise have been routed below them, which results in longer interconnections. PGAs with large pin counts hence require extra routing layers to connect to the multitudes of pins. Finally, while the parasitic capacitance and inductance of the PGA are slightly lower than that of the DIP, their values are still substantial.

Many of the shortcomings of the through-hole mounting are solved by using the *surface-mount* technique. A chip is attached to the surface of the board with a solder connection without requiring any through-holes (Figure 2.14b). Packing density is increased for the following reasons: (1) through-holes are eliminated, which provides more wiring space; (2) the lead pitch is reduced; and (3) chips can be mounted on both sides of the board. In addition, the elimination of the through-holes improves the mechanical strength of the board. On the negative side, the on-the-surface connection makes the chip-board connection weaker. Not only is it cumbersome to mount a component on a board, but also more expensive equipment is needed, since a simple soldering iron will not do anymore. Finally, testing of the board is more complex, because the package pins are no longer accessible at the backside of the board. Signal probing becomes hard or even impossible.

A variety of surface-mount packages are currently in use with different pitch and pin-count parameters. Three of these packages are shown in Figure 2.15: the *small-outline package* with gull wings, the *plastic leaded package* (PLCC) with J-shaped leads, and the



1 Bare die
2 DIP
3 PGA
4 Small-outline IC
5 Quad flat pack
6 PLCC
7 Leadless carrier

Figure 2.15 An overview of commonly used package types.

leadless chip carrier. An overview of the most important parameters for a number of packages is given in Table 2.3.

Table 2.3 Parameters of various types of chip carriers.

Package type	Lead spacing (Typical)	Lead count (Maximum)
Dual-in-line	2.54 mm	64
Pin grid array	2.54 mm	> 300
Small-outline IC	1.27 mm	28
Leaded chip carrier (PLCC)	1.27 mm	124
Leadless chip carrier	0.75 mm	124

Even surface-mount packaging is unable to satisfy the quest for evermore higher pin-counts. This is worsened by the demand for power connections: today's high performance chips, operating at low supply voltages, require as many power and ground pins as signal I/Os! When more than 300 I/O connections are needed, solder balls replace pins as the preferred interconnect medium between package and board. An example of such a packaging approach, called ceramic *ball grid array* (BGA), is shown in Figure 2.16. Solder bumps are used to connect both the die to the package substrate, and the package to the board. The area array interconnect of the BGA provides constant input/output density regardless of the number of total package I/O pins. A minimum pitch between solder balls of as low as 0.8 mm can be obtained, and packages with multiple 1000's of I/O signals are feasible.

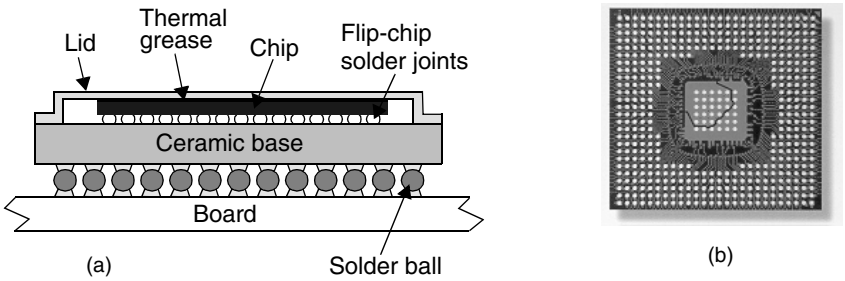


Figure 2.16 Ball grid array packaging; (a) cross-section, (b) photo of package bottom

Multi-Chip Modules—Die-to-Board

The deep hierarchy of interconnect levels in the package is becoming unacceptable in today's complex designs with their higher levels of integration, large signal counts, and increased performance requirements. The trend is toward reducing the number of levels. For the time being, attention is focused on the elimination of the first level in the packaging hierarchy. Eliminating one layer in the packaging hierarchy by mounting the die directly on the wiring backplanes—board or substrate—offers a substantial benefit when performance or density is a major issue. This packaging approach is called the multichip module technique (or MCM), and results in a substantial increase in packing density as well as improved performance.

A number of the previously mentioned die-mounting techniques can be adapted to mount dies directly on the substrate. This includes wire bonding, TAB, and flip-chip, although the latter two are preferable. The substrate itself can vary over a wide range of materials, depending upon the required mechanical, electrical, thermal, and economical requirements. Materials of choice are epoxy substrates (similar to PC boards), metal, ceramics, and silicon. Silicon has the advantage of presenting a perfect match in mechanical and thermal properties with respect to the die material.

The main advantages of the MCM approach are the increased packaging density and performance. An example of an MCM module implemented using a silicon substrate (commonly dubbed *silicon-on-silicon*) is shown in Figure 2.17. The module, which implements an avionics processor module and is fabricated by Rockwell International, contains 53 ICs and 40 discrete devices on a $2.2'' \times 2.2''$ substrate with aluminum polyimide interconnect. The interconnect wires are only an order of magnitude wider than what is typical for on-chip wires, since similar patterning approaches are used. The module itself has 180 I/O pins. Performance is improved by the elimination of the chip-carrier layer with its assorted parasitics, and through a reduction of the global wiring lengths on the die, a result of the increased packaging density. For instance, a solder bump has an assorted capacitance and inductance of only 0.1 pF and 0.01 nH respectively. The MCM technology can also reduce power consumption significantly, since large output drivers—and associated dissipation—become superfluous due to the reduced load capacitance of the output pads.

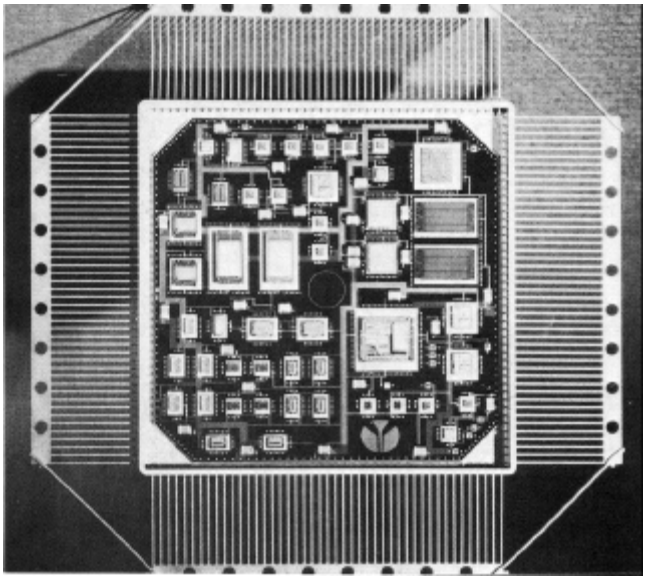


Figure 2.17 Avionics processor module. *Courtesy of Rockwell International*.

The dynamic power associated with the switching of the large load capacitances is simultaneously reduced.

While MCM technology offers some clear benefits, its main disadvantage is economic. This technology requires some advanced manufacturing steps that make the process expensive. The approach is only justifiable when either dense housing or extreme performance is essential. In the near future, this argument might become obsolete as MCM approaches proliferate.

2.4.3 Thermal Considerations in Packaging

As the power consumption of integrated circuits rises, it becomes increasingly important to efficiently remove the heat generated by the chips. A large number of failure mechanisms in ICs are accentuated by increased temperatures. Examples are leakage in reverse-biased diodes, electromigration, and hot-electron trapping. To prevent failure, the temperature of the die must be kept within certain ranges. The supported temperature range for commercial devices during operation equals 0° to 70°C. Military parts are more demanding and require a temperature range varying from –55° to 125°C.

The cooling effectiveness of a package depends upon the thermal conduction of the package material, which consists of the package substrate and body, the package composition, and the effectiveness of the heat transfer between package and cooling medium. Standard packaging approaches use still or circulating air as the cooling medium. The transfer efficiency can be improved by adding finned metal heat sinks to the package. More expensive packaging approaches, such as those used in mainframes or supercomput-

ers, force air, liquids, or inert gases through tiny ducts in the package to achieve even greater cooling efficiencies.

<We may want to briefly introduce the thermal equation here.>

As an example, a 40-pin DIP has a thermal resistance of 38 °C/W and 25 °C/W for natural and forced convection of air. This means that a DIP can dissipate 2 watts (3 watts) of power with natural (forced) air convection, and still keep the temperature difference between the die and the environment below 75 °C. For comparison, the thermal resistance of a ceramic PGA ranges from 15 ° to 30 °C/W.

Since packaging approaches with decreased thermal resistance are prohibitively expensive, keeping the power dissipation of an integrated circuit within bounds is an economic necessity. The increasing integration levels and circuit performance make this task nontrivial. An interesting relationship in this context has been derived by Nagata [Nagata92]. It provides a bound on the integration complexity and performance as a function of the thermal parameters

$$\frac{N_G}{t_p} \leq \frac{\Delta T}{\theta E} \tag{2.2}$$

where N_G is the number of gates on the chip, t_p the propagation delay, ΔT the maximum temperature difference between chip and environment, θ the thermal resistance between them, and E the switching energy of each gate.

Example 2.2 Thermal Bounds On Integration

For ΔT = 100 °C, θ = 2.5 °C/W and E = 0.1 pJ, this results in $N_G/t_p \leq 4 \times 10^5$ (gates/nsec). In other words, the maximum number of gates on a chip, when all gates are operating simultaneously, must be less than 400,000 if the switching speed of each gate is 1 nsec. This is equivalent to a power dissipation of 40 W.

Fortunately, not all gates are operating simultaneously in real systems. The maximum number of gates can be substantially larger, based on the activity in the circuit. For instance, it was experimentally derived that the ratio between the average switching period and the propagation delay ranges from 20 to 200 in mini- and large-scale computers [Masaki92].

Nevertheless, Eq. (2.2) demonstrates that heat dissipation and thermal concerns present an important limitation on circuit integration. Design approaches for low power that reduce either E or the activity factor are rapidly gaining importance.

2.5 Perspective — Trends in Process Technology

Modern CMOS processes pretty much track the flow described in the previous sections although a number of the steps might be reversed, a single well approach might be followed, a grown field oxide instead of the trench approach might be used, or extra steps such as LDD (Lightly Doped Drain) might be introduced. Also, it is quite common to cover the polysilicon interconnections as well as the drain and source regions with a *silicide* such as $TiSi_2$ to improve the conductivity (see Figure 2.2). This extra operation is

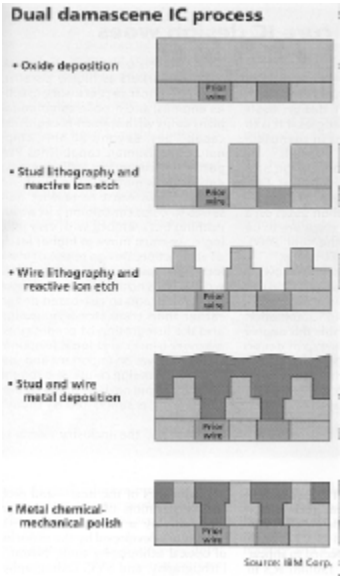
inserted between steps *i* and *j* of our process. Some important modifications or improvements to the technology are currently under way or are on the horizon, and deserve some attention. Beyond these, it is our belief that no dramatic changes, breaking away from the described CMOS technology, must be expected in the next decade.

2.5.1 Short-Term Developments

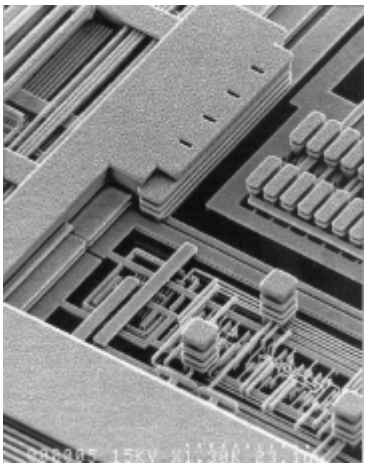
Copper and Low-k Dielectrics

A recurring theme in this text book will be the increasing impact of interconnect on the overall design performance. Process engineers are continuously evaluating alternative options for the traditional 'Aluminum conductor—SiO_2 insulator' combination that has been the norm for the last decades. In 1998, engineers at IBM introduced an approach that finally made the use of Copper as an interconnect material in a CMOS process viable and economical [IEEESpectrum98]. Copper has the advantage of have a resistivity that is substantially lower than Aluminum. Yet it has the disadvantage of easy diffusion into silicon, which degrades the characteristics of the devices. Coating the copper with a buffer material such as Titanium Nitride, preventing the diffusion, addresses this problem, but requires a special deposition process. The Dual Damascene process, introduced by IBM, (Figure 2.18) uses a metallization approach that fills trenches etched into the insulator, followed by a chemical-mechanical polishing step. This is in contrast with the traditional approach that first deposits a full metal layer, and removes the redundant material through etching.

In addition to the lower resistivity interconnections, insulator materials with a lower dielectric constant than SiO_2 —and hence lower capacitance— have also found their way into the production process starting with the 0.18 μm CMOS process generation.



(a)



(b)

Figure 2.18
The damascene process (from Geppert98]):
process steps (a), and microphotograph of interconnect after removal of insulator (b)

Silicon-on-Insulator

While having been around for quite a long time, there seems to be a good chance that Silicon-on-Insulator (SOI) CMOS might replace the traditional CMOS process, described in the previous sections (also known as the *bulk CMOS process*). The main difference lies in the start material: the transistors are constructed in a very thin layer of silicon, deposited on top of a thick layer of insulating SiO_2 (Figure 2.19). The primary advantages of the SOI process are reduced parasitics and better transistor on-off characteristics. It has, for instance, been demonstrated by researchers at IBM that the porting of a design from a bulk CMOS to an SOI process —leaving all other design and process parameters such as channel length and oxide thickness identical— yields a performance improvement of 22% [Allen99]. Preparing a high quality SOI substrate at an economical cost was long the main hindrance against a large-scale introduction of the process. This picture has changed at the end of the nineties, and SOI is steadily moving into the mainstream.

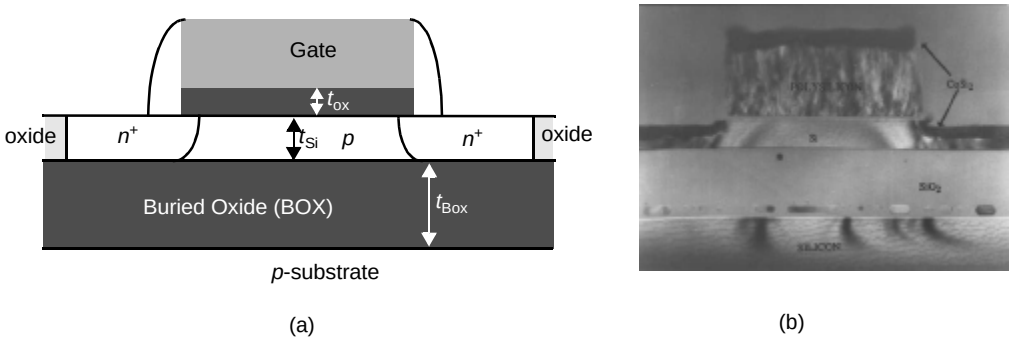


Figure 2.19 Silicon-on-insulator process— schematic diagram (a) and SEM cross-section (b).

2.5.2 In the Longer Term

Extending the life of CMOS technology beyond the next decade, and deeply below the 100 nm channel length region however will require re-engineering of both the process technology and the device structure. We already are witnessing the emergence of a wide range of new devices (such as organic transistors, molecular switches, and quantum devices). While projecting what approaches will dominate in that era equals resorting to crystal-ball gazing, one interesting development is worth mentioning.

Truly Three-Dimensional Integrated Circuits

Getting signals in and out of the computation elements in a timely fashion is one of the main challenges presented by the continued increase in integration density. One way to address this problem is to introduce extra active layers, and to sandwich them in-between the metal interconnect layers (Figure 2.20). This enables us to position high density memory on top of the logic processors implemented in the bulk CMOS, reducing the distance between computation and storage, and hence also the delay [Souri00]. In addition, devices with different voltage, performance, or substrate material requirements can be placed in

different layers. For instance, the top active layer can be reserved for the realization of optical transceivers, which may help to address the input/output requirements, or MEMS (Micro Electro-Mechanical Systems) devices providing sensoring functions or radio-frequency (RF) interfaces.

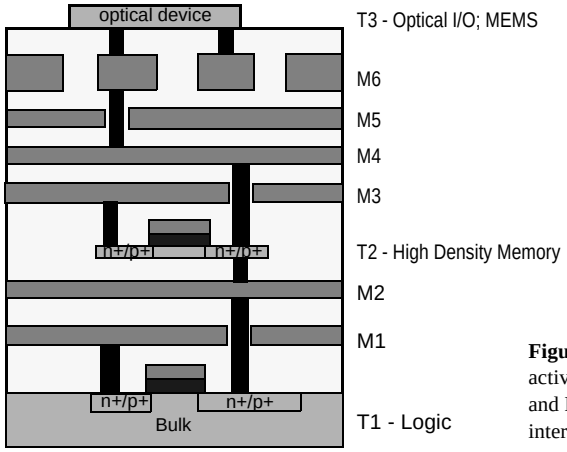


Figure 2.20 Example of true 3D integration. Extra active layers (T*), implementing high density memory and I/O, are sandwiched between the metal interconnect layers (M*).

While this approach may seem to be promising, a number of major challenges and hindrances have to be resolved to make it really viable. How to remove the dissipated heat is one of the compelling questions. Ensuring yield is another one. Yet, researchers are demonstrating major progress, and 3D integration might very well be on the horizon. Before the true solution arrives, we might have to rely on some intermediate approaches. One alternative, called 2.5D integration, is to bond two fully processed wafers, on which circuits are fabricated on the surface such that the chips completely overlap. Vias are etched to electrically connect both chips after metallization. The advantages of this technology lie in the similar electrical properties of devices on all active levels and the independence of processing temperature since all chips can be fabricated separately and later bonded. The major limitation of this technique is its lack of precision (best case alignment +/- 2 µm), which restricts the inter-chip communication to global metal lines.

One picture that strongly emerges from these futuristic devices is that the line between chip, substrate, package, and board is blurring, and that designers of these systems-on-a-die will have to consider all these aspects simultaneously.

2.6 Summary

This chapter has presented an a birds-eye view on issues regarding the manufacturing and packaging of CMOS integrated circuits.

- The manufacturing process of integrated circuits require a large number of steps, each of which consists of a sequence of basic operations. A number of these steps and/or operations, such as photolithograpical exposure and development, material deposition, and etching, are executed very repetitively in the course of the manufacturing process.

- The *optical masks* forms the central interface between the intrinsics of the manufacturing process and the design that the user wants to see transferred to the silicon fabric.
- The *design rules set* define the constraints n terms of minimum width and separation that the IC design has to adhere to if the resulting circuit is to be fully functional. This design rules acts as the contract between the circuit designer and the process engineer.
- The *package* forms the interface between the circuit implemented on the silicon die and the outside world, and as such has a major impact on the performance, reliability, longevity, and cost of the integrated circuit.

2.7 To Probe Further

Many textbooks on semiconductor manufacturing have been published over the last few decades. An excellent overview of the state-of-the-art in CMOS manufacturing can be found in the "Silicon VLSI Technology" book by J. Plummer, M. Deal, and P. Griffin [Plummer00]. A visual overview of the different steps in the manufacturing process can be found on the web at [Fullman99]. Other sources for information are the IEEE Transactions on Electron Devices, and the Technical Digest of the IEDM conference.

REFERENCES

[Allen99] D. Allen, et al., "A 0.2 μm 1.8 V SOI 550 MHz PowerPC Microprocessor with Copper Interconnects," *Proceedings IEEE ISSCC Conference*, vol. XLII, pp. 438-439, February 1999.

[Bakoglu90] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990.

[Doane93] D. Doane, ed., *Multichip Module Technologies and Alternatives*, Van Nostrand-Reinhold, 1993.

[Franzon93] P. Franzon, "Electrical Design of Digital Multichip Modules," in [Doane93], pp 525–568, 1993.

[Fullman99] Fullman Kinetics, "The Semiconductor Manufacturing Process", *http://www.fullman-kinetics.com/semiconductors/semiconductors.html*, 1999.

[Geppert98] L. Geppert, "Technology—1998 Analysis and Forecast", *IEEE Spectrum*, Vol. 35, No 1, pp. 23, January 1998.

[Landman71] B. Landman and R. Russo, "On a Pin versus Block Relationship for Partitions of Logic Graphs," *IEEE Trans. on Computers*, vol. C-20, pp. 1469–1479, December 1971.

[Masaki92] A. Masaki, "Deep-Submicron CMOS Warms Up to High-Speed Logic," *Circuits and Devices Magazine*, Nov. 1992.

[Mead80] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[Nagata92] M. Nagata, "Limitations, Innovations, and Challenges of Circuits and Devices into a Half Micrometer and Beyond," *IEEE Journal of Solid State Circuits*, vol. 27, no. 4, pp. 465–472, April 1992.

[Plummer00] J. Plummer, M. Deal, and P. Griffin, *Silicon VLSI Technology*, Prentice Hall, 2000.

[Steidel83] C. Steidel, "*Assembly Techniques and Packaging*," in [Sze83], pp. 551–598, 1983.

[Souri00] S. J. Souri, K. Banerjee, A. Mehrotra and K. C. Saraswat, "*Multiple Si Layer ICs: Motivation, Performance Analysis, and Design Implications*," Proceedings 37th Design Automation Conference, pp. 213-220, June 2000.

CHAPTER 3

THE DEVICES

Qualitative understanding of MOS devices

n

Simple component models for manual analysis

n

Detailed component models for SPICE

n

Impact of process variations

3.1 Introduction

3.2 The Diode

- 3.2.1 A First Glance at the Diode — The Depletion Region
- 3.2.2 Static Behavior
- 3.2.3 Dynamic, or Transient, Behavior
- 3.2.4 The Actual Diode—Secondary Effects
- 3.2.5 The SPICE Diode Model

3.3 The MOS(FET) Transistor

- 3.3.1 A First Glance at the Device
- 3.3.2 The MOS Transistor under Static Conditions
- 3.3.3 Dynamic Behavior
- 3.3.4 The Actual MOS Transistor—Some Secondary Effects
- 3.3.5 SPICE Models for the MOS Transistor

3.4 A Word on Process Variations

3.5 Perspective: Technology Scaling

3.1 Introduction

It is a well-known premise in engineering that the conception of a complex construction without a prior understanding of the underlying building blocks is a sure road to failure. This surely holds for digital circuit design as well. The basic building blocks in today's digital circuits are the silicon semiconductor devices, more specifically the MOS transistors and to a lesser degree the parasitic diodes, and the interconnect wires. The role of the semiconductor devices has been appreciated for a long time in the world of digital integrated circuits. On the other hand, interconnect wires have only recently started to play a dominant role as a result of the advanced scaling of the semiconductor technology.

Giving the reader the necessary *knowledge and understanding of these components* is the prime motivation for the next two chapters. It is not our intention to present an in-depth treatment of the physics of semiconductor devices and interconnect wires. We refer the reader to the many excellent textbooks on semiconductor devices for that purpose, some of which are referenced in the *To Probe Further* section at the end of the chapters. The goal is rather to describe the functional operation of the devices, to highlight the properties and parameters that are particularly important in the design of digital gates, and to introduce notational conventions.

Another important function of this chapter is the introduction of *models*. Taking all the physical aspects of each component into account when designing complex digital circuits leads to an unnecessary complexity that quickly becomes intractable. Such an approach is similar to considering the molecular structure of concrete when constructing a bridge. To deal with this issue, an abstraction of the component behavior called a *model* is typically employed. A range of models can be conceived for each component presenting a trade-off between accuracy and complexity. A simple first-order model is useful for manual analysis. It has limited accuracy but helps us to understand the operation of the circuit and its dominant parameters. When more accurate results are needed, complex, second- or higher-order models are employed in conjunction with computer-aided simulation. In this chapter, we present both first-order models for manual analysis as well as higher-order models for simulation for each component of interest.

Designers tend to take the component parameters offered in the models for granted. They should be aware, however, that these are only nominal values, and that the actual parameter values vary with operating temperature, over manufacturing runs, or even over a single wafer. To highlight this issue, a short discussion on *process variations* and their impact is included in the chapter.

3.2 The Diode

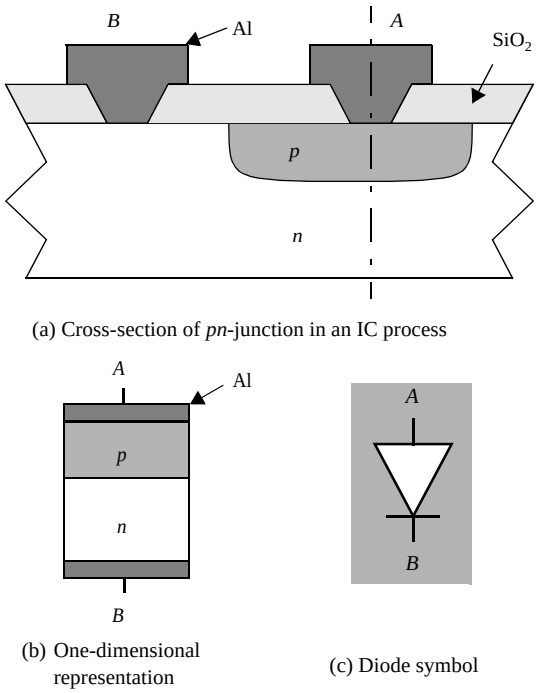
Although diodes rarely occur directly in the schematic diagrams of present-day digital gates, they are still omnipresent. Each MOS transistor implicitly contains a number of reverse-biased diodes that directly influence the behavior of the device. Especially, the voltage-dependent capacitances contributed by these parasitic elements play an important role in the switching behavior of the MOS digital gate. Diodes are also used to protect the input devices of an IC against static charges. Therefore, a brief review of the basic properties and device equations of the diode is appropriate. Rather than being comprehensive,

we choose to focus on those aspects that prove to be influential in the design of digital MOS circuits, this is the operation in reverse-biased mode.[1]

3.2.1 A First Glance at the Diode — The Depletion Region

The *pn*-junction diode is the simplest of the semiconductor devices. Figure 3.1a shows a cross-section of a typical *pn*-junction. It consists of two homogeneous regions of *p*- and *n*-type material, separated by a region of transition from one type of doping to another, which is assumed thin. Such a device is called a *step* or *abrupt junction*. The *p*-type material is doped with *acceptor* impurities (such as boron), which results in the presence of holes as the dominant or majority carriers. Similarly, the doping of silicon with *donor* impurities (such as phosphorus or arsenic) creates an *n*-type material, where electrons are the majority carriers. Aluminum contacts provide access to the *p*- and *n*-terminals of the device. The circuit symbol of the diode, as used in schematic diagrams, is introduced in Figure 3.1c.

To understand the behavior of the *pn*-junction diode, we often resort to a one-dimensional simplification of the device (Figure 3.1b). Bringing the *p*- and *n*-type materials together causes a large concentration gradient at the boundary. The electron concentration changes from a high value in the *n*-type material to a very small value in the *p*-type material. The reverse is true for the hole concentration. This gradient causes electrons to



(a) Cross-section of *pn*-junction in an IC process

(b) One-dimensional representation

(c) Diode symbol

Figure 3.1 Abrupt *pn*-junction diode and its schematic symbol.

[1] We refer the interested reader to the web-site of the textbook for a comprehensive description of the diode operation.

diffuse from *n* to *p* and holes to diffuse from *p* to *n*. When the holes leave the *p*-type material, they leave behind immobile acceptor ions, which are negatively charged. Consequently, the *p*-type material is negatively charged in the vicinity of the *pn*-boundary. Similarly, a positive charge builds up on the *n*-side of the boundary as the diffusing electrons leave behind the positively charged donor ions. The region at the junction, where the majority carriers have been removed, leaving the fixed acceptor and donor ions, is called the *depletion* or *space-charge region*. The charges create an electric field across the boundary, directed from the *n* to the *p*-region. This field counteracts the diffusion of holes and electrons, as it causes electrons to *drift* from *p* to *n* and holes to drift from *n* to *p*. Under equilibrium, the depletion charge sets up an electric field such that the drift currents are equal and opposite to the diffusion currents, resulting in a zero net flow.

The above analysis is summarized in Figure 3.2 that plots the current directions, the charge density, the electrical field, and the electrostatic field of the abrupt *pn*-junction under zero-bias conditions. In the device shown, the *p* material is more heavily doped than

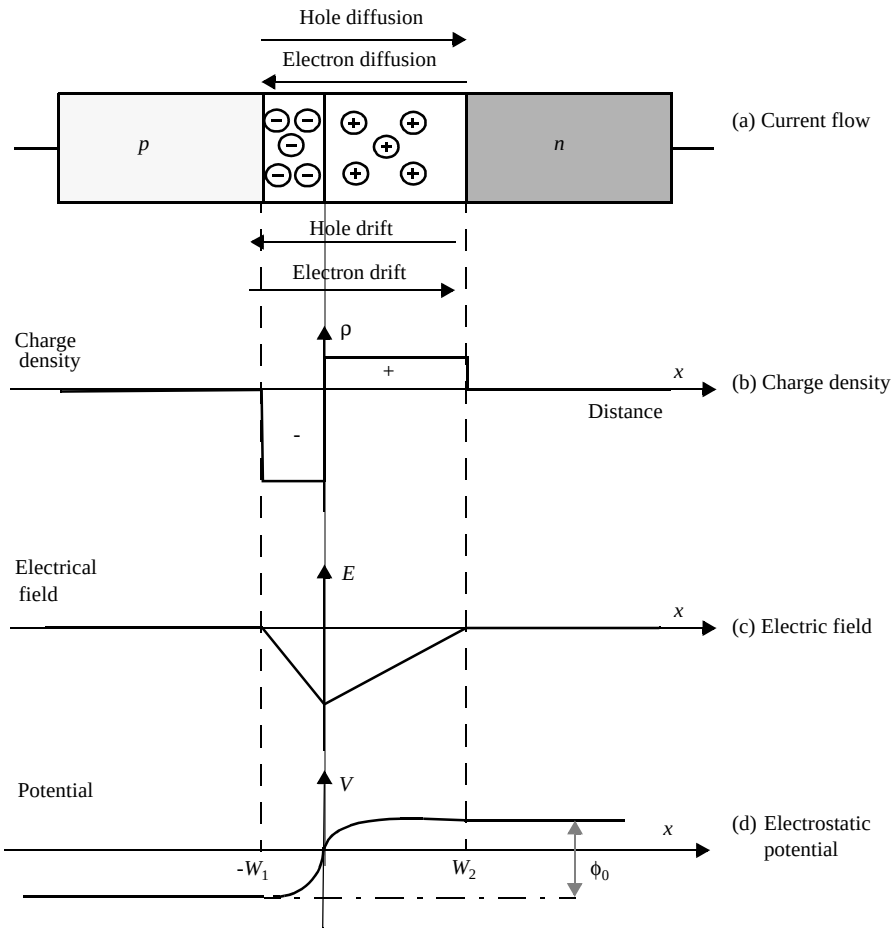


Figure 3.2 The abrupt *pn*-junction under equilibrium bias.

the n, or $N_A > N_D$, with N_A and N_D the acceptor and donor concentrations, respectively. Hence, the charge concentration in the depletion region is higher on the p-side of the junction. Figure 3.2 also shows that under zero bias, there exists a voltage ϕ_0 across the junction, called the *built-in potential*. This potential has the value

$$\phi_0 = \phi_T \ln\left[\frac{N_A N_D}{n_i^2}\right] \tag{3.1}$$

where ϕ_T is the *thermal voltage*

$$\phi_T = \frac{kT}{q} = 26\,\text{mV at 300 K} \tag{3.2}$$

The quantity n_i is the intrinsic carrier concentration in a pure sample of the semiconductor and equals approximately 1.5×10^{10} cm^{-3} at 300 K for silicon.

Example 3.1 Built-in Voltage of *pn*-junction

An abrupt junction has doping densities of $N_A = 10^{15}$ atoms/cm^3, and $N_D = 10^{16}$ atoms/cm^3. Calculate the built-in potential at 300 K.

From Eq. (3.1),

$$\phi_0 = 26 \ln\left[\frac{10^{15} \times 10^{16}}{2.25 \times 10^{20}}\right] \text{mV} = 638 \text{ mV}$$

3.2.2 Static Behavior

The Ideal Diode Equation

Assume now that a forward voltage V_D is applied to the junction or, in other words, that the potential of the p-region is raised with respect to the n-zone. The applied potential lowers the potential barrier. Consequently, the flow of mobile carriers across the junction increases as the diffusion current dominates the drift component. These carriers traverse

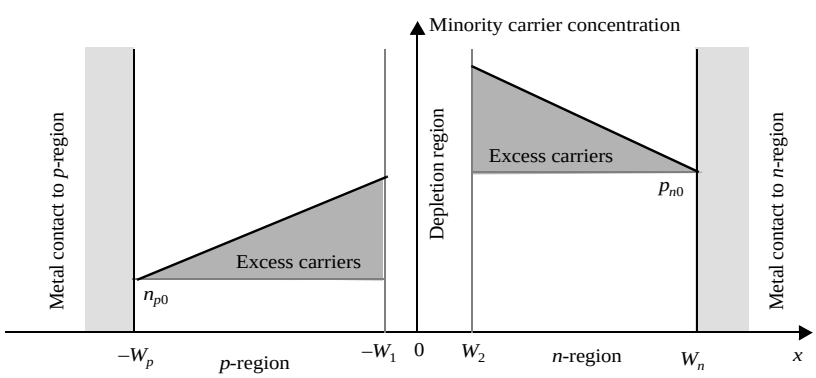


Figure 3.3 Minority carrier concentrations in the neutral region near an abrupt *pn*-junction under forward-bias conditions.

the depletion region and are injected into the neutral *n*- and *p*-regions, where they become minority carriers, as is illustrated in Figure 3.3. Under the assumption that no voltage gradient exists over the neutral regions, which is approximately the case for most modern devices, these minority carriers will diffuse through the region as a result of the concentration gradient until they get recombined with a majority carrier. The net result is a current flowing through the diode from the *p*-region to the *n*-region, and the diode is said to be in the *forward-bias* mode.

On the other hand, when a reverse voltage V_D is applied to the junction or when the potential of the *p*-region is lowered with respect to the *n*-region, the potential barrier is raised. This results in a reduction in the diffusion current, and the drift current becomes dominant. A current flows from the *n*-region to the *p*-region. Since the number of minority carriers in the neutral regions (electrons in the *p*-zone, holes in the *n*-region) is very small, this drift current component is virtually ignorable (Figure 3.4). It is fair to state that in the *reverse-bias* mode the diode operates as a nonconducting, or blocking, device. The diode thus acts as a one-way conductor.

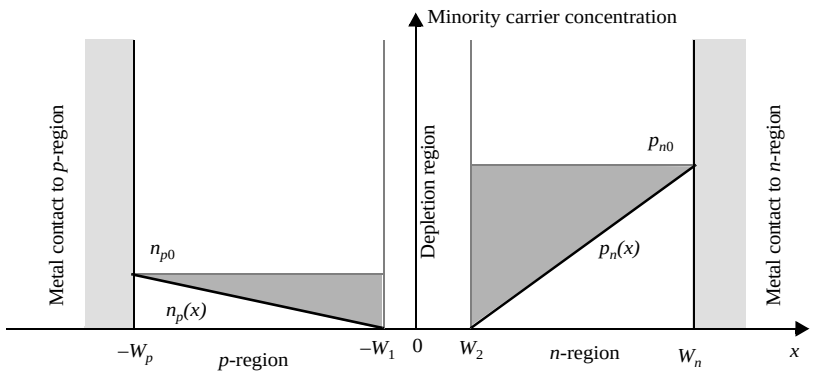


Figure 3.4 Minority carrier concentration in the neutral regions near the *pn*-junction under reverse-bias conditions.

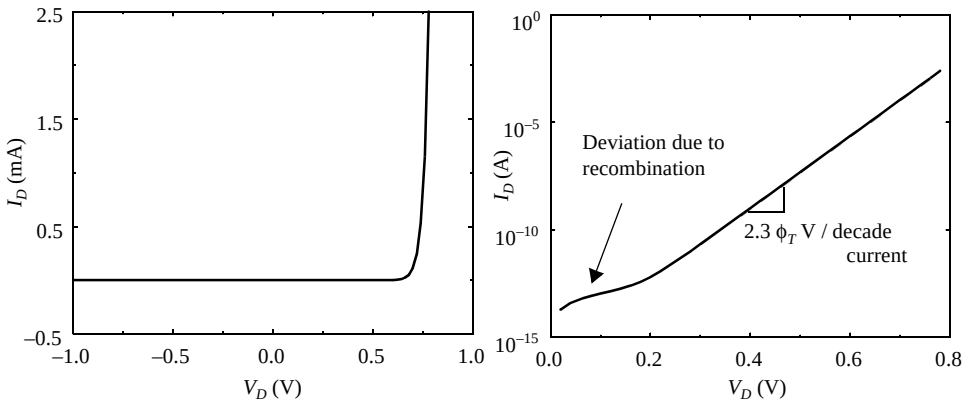
The most important property of the diode current is its *exponential dependence* upon the applied bias voltage. This is illustrated in Figure 3.5, which plots the diode current I_D as a function of the bias voltage V_D. The exponential behavior for positive-bias voltages is even more apparent in Figure 3.5b, where the current is plotted on a logarithmic scale. The current increases by a factor of 10 for every extra 60 mV (= 2.3 ϕ_T) of forward bias. At small voltage levels (V_D < 0.15 V), a deviation from the exponential dependence can be observed, which is due to the recombination of holes and electrons in the depletion region.

The behavior of the diode for both forward- and reverse bias conditions is best described by the well-known *ideal diode equation*, which relates the current through the diode I_D to the diode bias voltage V_D

$$I_D = I_S(e^{V_D/\phi_T} - 1) \tag{3.3}$$

Observe how Eq. (3.3) corresponds to the exponential behavior plotted in Figure 3.5. ϕ_T is the thermal voltage of Eq. (3.2) and is equal to 26 mV at room temperature.

I_S represents a constant value, called the *saturation current* of the diode. It is proportional to the area of the diode, and a function of the doping levels and widths of the neutral



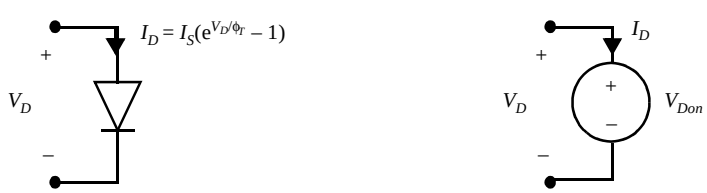
(a) On a linear scale (b) On a logarithmic scale (forward bias only)

Figure 3.5 Diode current as a function of the bias voltage V_D.

regions. Most often, I_S is determined empirically.It is worth mentioning that in actual devices, the reverse currents are substantially larger than the saturation current I_S. This is due to the thermal generation of hole and electron pairs in the depletion region. The electric field present sweeps these carriers out of the region, causing an additional current component. For typical silicon junctions, the saturation current is nominally in the range of 10^{-17} A/μm^2, while the actual reverse currents are approximately three orders of magnitude higher. Actual device measurements are, therefore, necessary to determine realistic values for the reverse diode leakage currents.

Models for Manual Analysis

The derived current-voltage equations can be summarized in a set of simple models that are useful in the manual analysis of diode circuits. A first model, shown in Figure 3.6a, is based on the ideal diode equation Eq. (3.3). While this model yields accurate results, it has the disadvantage of being strongly nonlinear. This prohibits a fast, first-order analysis of the dc-operation conditions of a network. An often-used, simplified model is derived by inspecting the diode current plot of Figure 3.5. For a "fully conducting" diode, the voltage drop over the diode V_D lies in a narrow range, approximately between 0.6 and 0.8 V. To a



(a) Ideal diode model (b) First-order diode model

Figure 3.6 Diode models.

first degree, it is reasonable to assume that a conducting diode has a fixed voltage drop V_{Don} over it. Although the value of V_{Don} depends upon I_S, a value of 0.7 V is typically assumed. This gives rise to the model of Figure 3.6b, where a conducting diode is replaced by a fixed voltage source.

Example 3.2 Analysis of Diode Network

Consider the simple network of Figure 3.7 and assume that $V_S = 3$ V, $R_S = 10$ kΩ and $I_S = 0.5 \times 10^{-16}$ A. The diode current and voltage are related by the following network equation

$$V_S - R_S I_D = V_D$$

Inserting the ideal diode equation and (painfully) solving the nonlinear equation using either numerical or iterative techniques yields the following solution: $I_D = 0.224$ mA, and $V_D = 0.757$ V. The simplified model with $V_{Don} = 0.7$ V produces similar results ($V_D = 0.7$ V, $I_D = 0.23$ A) with far less effort. It hence makes considerable sense to use this model when determining a first-order solution of a diode network.

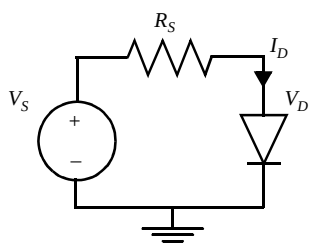


Figure 3.7 A simple diode circuit.

3.2.3 Dynamic, or Transient, Behavior

So far, we have mostly been concerned with the static, or steady-state, characteristics of the diode. Just as important in the design of digital circuits is the response of the device to changes in its bias conditions. The transient, or dynamic, response determines the maximum speed at which the device can be operated. Because the operation mode of the diode is a function of the amount of charge present in both the neutral and the space-charge regions, its dynamic behavior is strongly determined by how fast charge can be moved around.

While we could embark at this point onto an in-depth analysis of the switching behavior of the diode in the forward-biasing mode, it is our conviction that this would be besides the point and unnecessarily complicate the discussion. In fact, all diodes in an operational MOS digital integrated circuit are reverse-biased and are supposed to remain so under all circumstances. Only under exceptional conditions may forward-biasing occur. A signal over(under) shooting the supply rail is an example of such. Due to its detrimental impact on the overall circuit operation, this should be avoided under all circumstances.

Hence, we will devote our attention solely to what governs the dynamic response of the diode under reverse-biasing conditions, the depletion-region charge.

Depletion-Region Capacitance

In the ideal model, the depletion region is void of mobile carriers, and its charge is determined by the immobile donor and acceptor ions. The corresponding charge distribution under zero-bias conditions was plotted in Figure 3.2. This picture can be easily extended to incorporate the effects of biasing. At an intuitive level the following observations can be easily verified—under forward-bias conditions, the potential barrier is reduced, which means that less space charge is needed to produce the potential difference. This corresponds to a reduced depletion-region width. On the other hand, under reverse conditions, the potential barrier is increased corresponding to an increased space charge and a wider depletion region. These observations are confirmed by the well- known depletion-region expressions given below (a derivation of these expressions, which are valid for abrupt junctions, is either simple or can be found in any textbook on devices such as [Howe97]). One observation is crucial — due to the global charge neutrality requirement of the diode, the total acceptor and donor charges must be numerically equal.

1. Depletion-region charge (V_D is positive for forward bias).

$$Q_j = A_D\sqrt{\left(2\varepsilon_{si}q\frac{N_A N_D}{N_A + N_D}\right)(\phi_0 - V_D)} \tag{3.4}$$

2. Depletion-region width.

$$W_j = W_2 - W_1 = \sqrt{\left(\frac{2\varepsilon_{si}}{q}\frac{N_A + N_D}{N_A N_D}\right)(\phi_0 - V_D)} \tag{3.5}$$

3. Maximum electric field.

$$E_j = \sqrt{\left(\frac{2q}{\varepsilon_{si}}\frac{N_A N_D}{N_A + N_D}\right)(\phi_0 - V_D)} \tag{3.6}$$

In the preceding equations ε_{si} stands for the electrical permittivity of silicon and equals 11.7 times the permittivity of a vacuum, or 1.053×10^{-10} F/m. The ratio of the *n*- versus *p*-side of the depletion-region width is determined by the doping-level ratios: $W_2/(-W_1) = N_A/N_D$.

From an abstract point of view, it is possible to visualize the depletion region as a capacitance, albeit one with very special characteristics. Because the space-charge region contains few mobile carriers, it acts as an insulator with a dielectric constant ε_{si} of the semiconductor material. The *n*- and *p*-regions act as the capacitor plates. A small change in the voltage applied to the junction dV_D causes a change in the space charge dQ_j. Hence, a depletion-layer capacitance can be defined

$$\begin{aligned} C_j &= \frac{\mathrm{d}Q_j}{\mathrm{d}V_D} = A_D\sqrt{\left(\frac{\varepsilon_{si}q}{2}\frac{N_A N_D}{N_A + N_D}\right)(\phi_0 - V_D)^{-1}} \\ &= \frac{C_{j0}}{\sqrt{1 - V_D/\phi_0}} \end{aligned} \tag{3.7}$$

where C_{j0} is the capacitance under zero-bias conditions and is only a function of the physical parameters of the device.

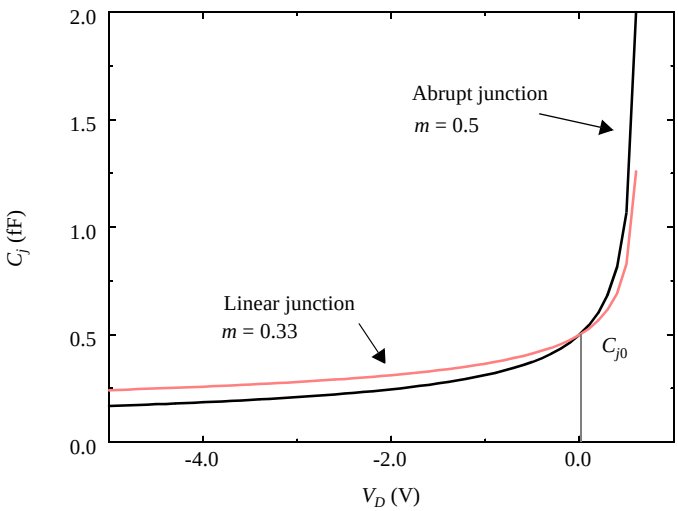


Figure 3.8 Junction capacitance (in fF/μm^2) as a function of the applied bias voltage.

$$C_{j0} = A_D \sqrt{\left(\frac{\varepsilon_{si} q}{2} \frac{N_A N_D}{N_A + N_D}\right) \phi_0^{-1}} \tag{3.8}$$

Notice that the same capacitance value is obtained when using the standard parallel-plate capacitor equation $C_j = \varepsilon_{si} A_D / W_j$ (with W_j given in Eq. (3.5)). Typically, the A_D factor is omitted, and C_j and C_{j0} are expressed as a capacitance/unit area.

The resulting junction capacitance is plotted in the function of the bias voltage in Figure 3.8 for a typical silicon diode found in MOS circuits. A strong *nonlinear dependence* can be observed. Note also that the capacitance decreases with an increasing reverse bias: a reverse bias of 5 V reduces the capacitance by more than a factor of two.

Example 3.3 Junction Capacitance

Consider the following silicon junction diode: $C_{j0} = 2 \times 10^{-3}$ F/m^2, $A_D = 0.5$ μm^2, and $\phi_0 = 0.64$ V. A reverse bias of –2.5 V results in a junction capacitance of 0.9×10^{-3} F/m^2 (0.9 fF/μm^2), or, for the total diode, a capacitance of 0.45 fF.

Equation (3.7) is only valid under the condition that the *pn*-junction is an *abrupt junction*, where the transition from *n* to *p* material is instantaneous. This is often not the case in actual integrated-circuit *pn*-junctions, where the transition from *n* to *p* material can be gradual. In those cases, a linear distribution of the impurities across the junction is a better approximation than the step function of the abrupt junction. An analysis of the *linearly-graded junction* shows that the junction capacitance equation of Eq. (3.7) still holds, but with a variation in order of the denominator. A more generic expression for the junction capacitance can be provided,

$$C_j = \frac{C_{j0}}{(1 - V_D/\phi_0)^m} \tag{3.9}$$

where m is called the *grading coefficient* and equals 1/2 for the abrupt junction and 1/3 for the linear or graded junction. Both cases are illustrated in Figure 3.8.

Large-Signal Depletion-Region Capacitance

Figure 3.8 raises awareness to the fact that the junction capacitance is a voltage-dependent parameter whose value varies widely between bias points. In digital circuits, operating voltages tend to move rapidly over a wide range. Under those circumstances, it is more attractive to replace the voltage-dependent, nonlinear capacitance C_j by an equivalent, linear capacitance C_{eq}. C_{eq} is defined such that, for a given voltage swing from voltages V_{high} to V_{low}, the same amount of charge is transferred as would be predicted by the nonlinear model

$$C_{eq} = \frac{\Delta Q_j}{\Delta V_D} = \frac{Q_j(V_{high}) - Q_j(V_{low})}{V_{high} - V_{low}} = K_{eq} C_{j0} \tag{3.10}$$

Combining Eq. (3.4) (extended to accommodate the grading coefficient m) and Eq. (3.10) yields the value of K_{eq}.

$$K_{eq} = \frac{-\phi_0^m}{(V_{high} - V_{low})(1 - m)} [(\phi_0 - V_{high})^{1-m} - (\phi_0 - V_{low})^{1-m}] \tag{3.11}$$

Example 3.4 Average Junction Capacitance

The diode of Example 3.3 is switched between 0 and –2.5 V. Compute the average junction capacitance (m = 0.5).

For the defined voltage range and for ϕ_0 = 0.64 V, K_{eq} evaluates to 0.622. The average capacitance hence equals 1.24 fF/μm^2.

3.2.4 The Actual Diode—Secondary Effects

In practice, the diode current is less than what is predicted by the ideal diode equation. Not all applied bias voltage appears directly across the junction, as there is always some voltage drop over the neutral regions. Fortunately, the resistivity of the neutral zones is generally small (between 1 and 100 Ω, depending upon the doping levels) and the voltage drop only becomes significant for large currents (>1 mA). This effect can be modeled by adding a resistor in series with the n- and p-region diode contacts.

In the discussion above, it was further assumed that under sufficient reverse bias, the reverse current reaches a constant value, which is essentially zero. When the reverse bias exceeds a certain level, called the *breakdown voltage*, the reverse current shows a dramatic increase as shown in Figure 3.9. In the diodes found in typical CMOS processes, this increase is caused by the *avalanche breakdown*. The increasing reverse bias heightens the magnitude of the electrical field across the junction. Consequently, carriers crossing the depletion region are accelerated to high velocity. At a critical field E_{crit}, the carriers

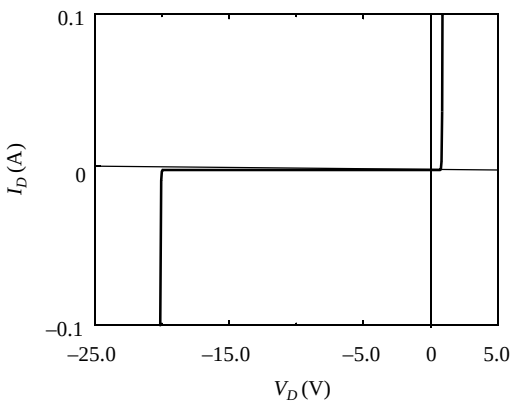


Figure 3.9 *I-V* characteristic of junction diode, showing breakdown under reverse-bias conditions (Breakdown voltage = 20 V).

reach a high -enough energy level that electron-hole pairs are created on collision with immobile silicon atoms. These carriers create, in turn, more carriers before leaving the depletion region. The value of E_{crit} is approximately 2×10^5 V/cm for impurity concentrations of the order of 10^{16} cm^{-3}. While avalanche breakdown in itself is not destructive and its effects disappear after the reverse bias is removed, maintaining a diode for a long time in avalanche conditions is not recommended as the high current levels and the associated heat dissipation might cause permanent damage to the structure. Observe that avalanche breakdown is not the only breakdown mechanism encountered in diodes. For highly doped diodes, another mechanism, called Zener breakdown, can occur. Discussion of this phenomenon is beyond the scope of this text.

Finally, it is worth mentioning that the diode current is affected by the operating *temperature* in a dual way:

1. The thermal voltage ϕ_T, which appears in the exponent of the current equation, is linearly dependent upon the temperature. An increase in ϕ_T causes the current to drop.
2. The saturation current I_S is also temperature-dependent, as the thermal equilibrium carrier concentrations increase with increasing temperature. Theoretically, the saturation current approximately doubles every 5 °C. Experimentally, the reverse current has been measured to double every 8 °C.

This dual dependence has a significant impact on the operation of a digital circuit. First of all, current levels (and hence power consumption) can increase substantially. For instance, for a forward bias of 0.7 V at 300 K, the current increases approximately 6%/°C, and doubles every 12 °C. Secondly, integrated circuits rely heavily on reverse-biased diodes as isolators. Increasing the temperature causes the leakage current to increase and decreases the isolation quality.

3.2.5 The SPICE Diode Model

In the preceding sections, we have presented a model for manual analysis of a diode circuit. For more complex circuits, or when a more accurate modeling of the diode that takes

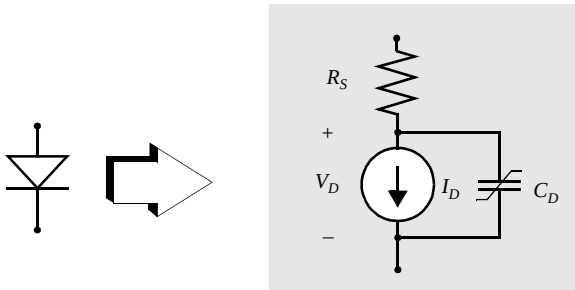


Figure 3.10 SPICE diode model.

into account second-order effects is required, manual circuit evaluation becomes intractable, and computer-aided simulation is necessary. While different circuit simulators have been developed over the last decades, the SPICE program, developed at the University of California at Berkeley, is definitely the most successful [Nagel75]. Simulating an integrated circuit containing active devices requires a mathematical model for those devices (which is called the *SPICE model* in the rest of the text). The accuracy of the simulation depends directly upon the quality of this model. For instance, one cannot expect to see the result of a second-order effect in the simulation if this effect is not present in the device model. Creating accurate and computation-efficient SPICE models has been a long process and is by no means finished. Every major semiconductor company has developed their own proprietary models, which it claims have either better accuracy or computational efficiency and robustness.

The standard SPICE model for a diode is simple, as shown in Figure 3.10. The steady-state characteristic of the diode is modeled by the nonlinear current source I_D, which is a modified version of the ideal diode equation

$$I_D = I_S(e^{V_D/n\phi_T} - 1) \tag{3.12}$$

The extra parameter n is called the *emission coefficient*. It equals 1 for most common diodes but can be somewhat higher than 1 for others. The resistor R_s models the series resistance contributed by the neutral regions on both sides of the junction. For higher current levels, this resistance causes the internal diode V_D to differ from the externally applied voltage, hence causing the current to be lower than what would be expected from the ideal diode equation.

The dynamic behavior of the diode is modeled by the nonlinear capacitance C_D, which combines the two different charge-storage effects in the diode: the space (or depletion-region) charge, and the excess minority carrier charge. Only the former was discussed in this chapter, as the latter is only an issue under forward-biasing conditions.

$$C_D = \frac{C_{j0}}{(1 - V_D/\phi_0)^m} + \frac{\tau_T I_S}{\phi_T} e^{V_D/n\phi_T} \tag{3.13}$$

A listing of the parameters used in the diode model is given in Table 3.1. Besides the parameter name, symbol, and SPICE name, the table contains also the default value used by SPICE in case the parameter is left undefined. Observe that this table is by no means complete. Other parameters are available to govern second-order effects such as break-

down, high-level injection, and noise. To be concise, we chose to limit the listing to the parameters of direct interest to this text. For a complete description of the device models (as well as the usage of SPICE), we refer to the numerous textbooks devoted to SPICE (e.g., [Banhzaf92], [Thorpe92]).

Table 3.1 First-order SPICE diode model parameters.

Parameter Name	Symbol	SPICE Name	Units	Default Value
Saturation current	I_S	IS	A	1.0 E−14
Emission coefficient	n	N	–	1
Series resistance	R_S	RS	Ω	0
Transit time	τ_T	TT	s	0
Zero-bias junction capacitance	C_{j0}	CJ0	F	0
Grading coefficient	m	M	–	0.5
Junction potential	ϕ_0	VJ	V	1

3.3 The MOS(FET) Transistor

The metal-oxide-semiconductor field-effect transistor (MOSFET or MOS, for short) is certainly the workhorse of contemporary digital design. Its major asset from a digital perspective is that the device performs very well as a switch, and introduces little parasitic effects. Other important advantages are its integration density combined with a relatively "simple" manufacturing process, which make it possible to produce large and complex circuits in an economical way.

Following the approach we took for the diode, we restrict ourselves in this section to a general overview of the transistor and its parameters. After a generic overview of the device, we present an analytical description of the transistor from a static (steady-state) and dynamic (transient) viewpoint. The discussion concludes with an enumeration of some second-order effects and the introduction of the SPICE MOS transistor models.

3.3.1 A First Glance at the Device

The MOSFET is a four terminal device. The voltage applied to the *gate* terminal determines if and how much current flows between the *source* and the *drain* ports. The *body* represents the fourth terminal of the transistor. Its function is secondary as it only serves to modulate the device characteristics and parameters.

At the most superficial level, the transistor can be considered to be a switch. When a voltage is applied to the gate that is larger than a given value called the *threshold voltage* V_T, a conducting channel is formed between drain and source. In the presence of a voltage difference between the latter two, current flows between them. The conductivity of the channel is modulated by the gate voltage—the larger the voltage difference between gate and source, the smaller the resistance of the conducting channel and the larger the current.

When the gate voltage is lower than the threshold, no such channel exists, and the switch is considered open.

Two types of MOSFET devices can be identified. The NMOS transistor consists of n^+ drain and source regions, embedded in a *p*-type substrate. The current is carried by electrons moving through an *n*-type channel between source and drain. This is in contrast with the *pn*-junction diode, where current is carried by both holes and electrons. MOS devices can also be made by using an *n*-type substrate and p^+ drain and source regions. In such a transistor, current is carried by holes moving through a *p*-type channel. The device is called a *p*-channel MOS, or PMOS transistor. In a complementary MOS technology (CMOS), both devices are present. The cross-section of a contemporary dual-well CMOS process was presented in Chapter 2, and is repeated here for convenience (Figure 3.11).

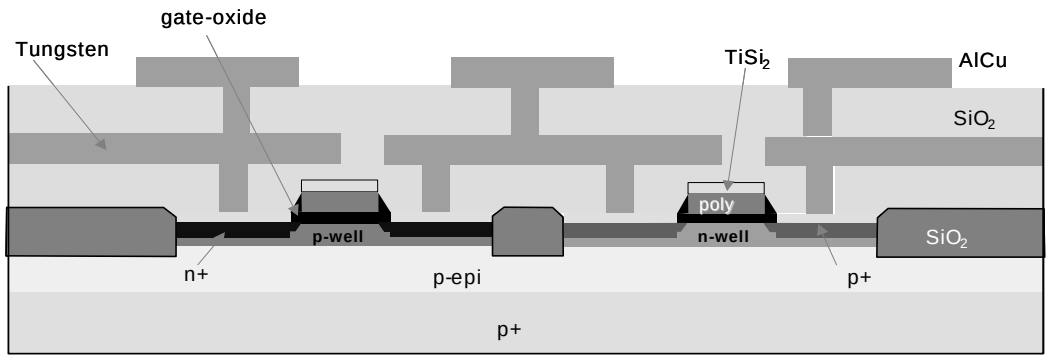


Figure 3.11 Cross-section of contemporary dual-well CMOS process.

Circuit symbols for the various MOS transistors are shown in Figure 3.12. As mentioned earlier, the transistor is a four-port device with gate, source, drain, and body terminals (Figures a and c). Since the body is generally connected to a dc supply that is identical for all devices of the same type (GND for NMOS, V_{dd} for PMOS), it is most often not shown on the schematics (Figures b and d). **If the fourth terminal is not shown, it is assumed that the body is connected to the appropriate supply.**

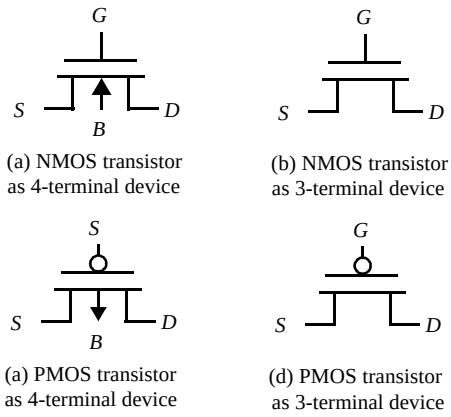


Figure 3.12 Circuit symbols for MOS transistors.

3.3.2 The MOS Transistor under Static Conditions

In the derivation of the static model of the MOS transistor, we concentrate on the NMOS device. All the arguments made are valid for PMOS devices as well as will be discussed at the end of the section.

The Threshold Voltage

Consider first the case where $V_{GS} = 0$ and drain, source, and bulk are connected to ground. The drain and source are connected by back-to-back *pn*-junctions (substrate-source and substrate-drain). Under the mentioned conditions, both junctions have a 0 V bias and can be considered off, which results in an extremely high resistance between drain and source.

Assume now that a positive voltage is applied to the gate (with respect to the source), as shown in Figure 3.13. The gate and substrate form the plates of a capacitor with the gate oxide as the dielectric. The positive gate voltage causes positive charge to accumulate on the gate electrode and negative charge on the substrate side. The latter manifests itself initially by repelling mobile holes. Hence, a depletion region is formed below the gate. This depletion region is similar to the one occurring in a *pn*-junction diode. Consequently, similar expressions hold for the width and the space charge per unit area. Compare these expressions to Eq. (3.4) and Eq. (3.5).

$$W_d = \sqrt{\frac{2\varepsilon_{si}\phi}{qN_A}} \tag{3.14}$$

and

$$Q_d = \sqrt{2qN_A\varepsilon_{si}\phi} \tag{3.15}$$

with N_A the substrate doping and ϕ the voltage across the depletion layer (i.e., the potential at the oxide-silicon boundary).

As the gate voltage increases, the potential at the silicon surface at some point reaches a critical value, where the semiconductor surface inverts to *n*-type material. This

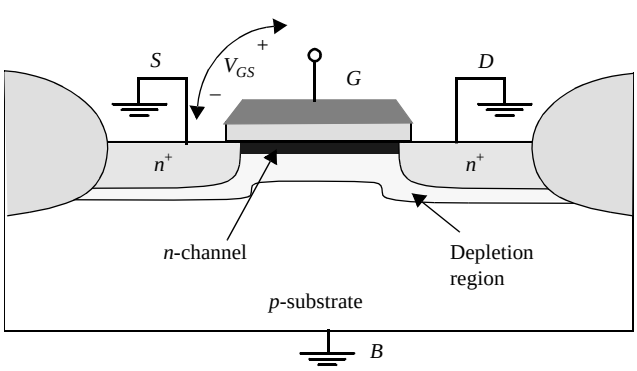


Figure 3.13 NMOS transistor for positive V_{GS}, showing depletion region and induced channel.

point marks the onset of a phenomenon known as *strong inversion* and occurs at a voltage equal to twice the *Fermi Potential* (Eq. (3.16)) ($\phi_F \approx -0.3$ V for typical *p*-type silicon substrates):

$$\phi_F = -\phi_T \ln\left(\frac{N_A}{n_i}\right) \tag{3.16}$$

Further increases in the gate voltage produce no further changes in the depletion-layer width, but result in additional electrons in the thin inversion layer directly under the oxide. These are drawn into the inversion layer from the heavily doped *n*+ source region. Hence, a continuous *n*-type channel is formed between the source and drain regions, the conductivity of which is modulated by the gate-source voltage.

In the presence of an inversion layer, the charge stored in the depletion region is fixed and equals

$$Q_{B0} = \sqrt{2qN_A\varepsilon_{si}|-2\phi_F|} \tag{3.17}$$

This picture changes somewhat in case a substrate bias voltage V_{SB} is applied (V_{SB} is normally positive for *n*-channel devices). This causes the surface potential required for strong inversion to increase and to become $|-2\phi_F + V_{SB}|$. The charge stored in the depletion region now is expressed by Eq. (3.18)

$$Q_B = \sqrt{2qN_A\varepsilon_{si}(|-2\phi_F + V_{SB}|)} \tag{3.18}$$

The value of V_{GS} where strong inversion occurs is called the *threshold voltage* V_T. V_T is a function of several components, most of which are material constants such as the difference in work-function between gate and substrate material, the oxide thickness, the Fermi voltage, the charge of impurities trapped at the surface between channel and gate oxide, and the dosage of ions implanted for threshold adjustment. From the above arguments, it has become clear that the source-bulk voltage V_{SB} has an impact on the threshold. as well. Rather than relying on a complex (and hardly accurate) analytical expression for the threshold, we rely on an empirical parameter called V_{T0}, which is the threshold voltage for $V_{SB} = 0$, and is mostly a function of the manufacturing process. The threshold voltage under different body-biasing conditions can then be determined in the following manner,

$$V_T = V_{T0} + \gamma\left(\sqrt{|-2\phi_F + V_{SB}|} - \sqrt{|-2\phi_F|}\right) \tag{3.19}$$

The parameter γ (gamma) is called the *body-effect coefficient*, and expresses the impact of changes in V_{SB}. Observe that the threshold voltage has a **positive** value for a typical **NMOS** device, while it is **negative** for a normal **PMOS** transistor.

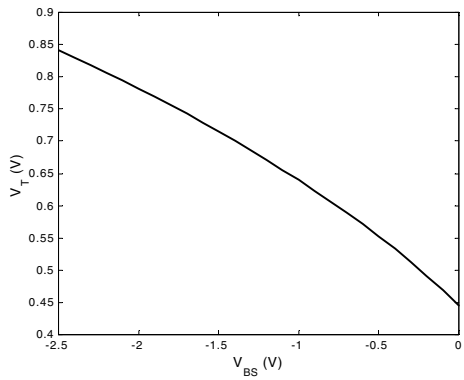


Figure 3.14 Effect of body-bias on threshold.

The effect of the well bias on the threshold voltage of an NMOS transistor is plotted in for typical values of $|-2\phi_F|$ = 0.6 V and γ = 0.4 $V^{0.5}$. A negative bias on the well or substrate causes the threshold to increase from 0.45 V to 0.85 V. Note also that V_{SB} always has to be larger than -0.6 V in an NMOS. If not, the source-body diode becomes forward biased, which deteriorates the transistor operation.

Example 3.5 Threshold Voltage of a PMOS Transistor

An PMOS transistor has a threshold voltage of -0.4 V, while the body-effect coefficient equals -0.4. Compute the threshold voltage for V_{SB} = -2.5 V. $2\phi_F$ = 0.6 V.

Using Eq. (3.19), we obtain V_T(-2.5 V) = -0.4 - 0.4 × ((2.5+0.6)$^{0.5}$ - 0.6$^{0.5}$) V = -0.79 V, which is twice the threshold under zero-bias conditions!

Resistive Operation

Assume now that $V_{GS} > V_T$ and that a small voltage, V_{DS}, is applied between drain and source. The voltage difference causes a current I_D to flow from drain to source (Figure 3.15). Using a simple analysis, a first-order expression of the current as a function of V_{GS} and V_{DS} can be obtained.

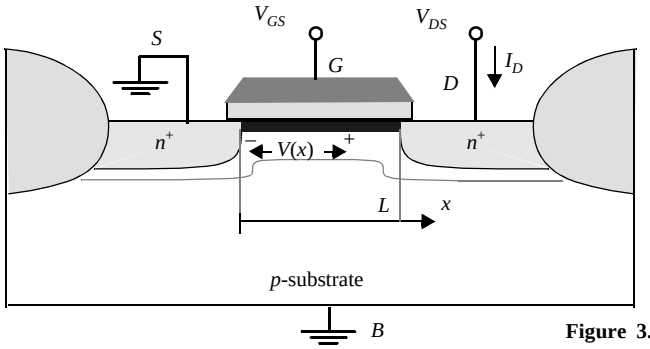


Figure 3.15 NMOS transistor with bias voltages.

At a point x along the channel, the voltage is $V(x)$, and the gate-to-channel voltage at that point equals $V_{GS} - V(x)$. Under the assumption that this voltage exceeds the threshold voltage all along the channel, the induced channel charge per unit area at point x can be computed.

$$Q_i(x) = -C_{ox}[V_{GS} - V(x) - V_T] \tag{3.20}$$

C_{ox} stands for the capacitance per unit area presented by the gate oxide, and equals

$$C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}} \tag{3.21}$$

with $\varepsilon_{ox} = 3.97 \times \varepsilon_o = 3.5 \times 10^{-11}$ F/m the oxide permittivity, and t_{ox} is the thickness of the oxide. The latter which is 10 nm (= 100 Å) or smaller for contemporary processes. For an oxide thickness of 5 nm, this translates into an oxide capacitance of 7 fF/μm^2.

The current is given as the product of the drift velocity of the carriers υ_n and the available charge. Due to charge conservation, it is a constant over the length of the channel. W is the width of the channel in a direction perpendicular to the current flow.

$$I_D = -\upsilon_n(x)Q_i(x)W \tag{3.22}$$

The electron velocity is related to the electric field through a parameter called the *mobility* μ_n (expressed in m^2/V·s). The mobility is a complex function of crystal structure, and local electrical field. In general, an empirical value is used.

$$\upsilon_n = -\mu_n\xi(x) = \mu_n\frac{dV}{dx} \tag{3.23}$$

Combining Eq. (3.20) – Eq. (3.23) yields

$$I_D dx = \mu_n C_{ox} W(V_{GS} - V - V_T)dV \tag{3.24}$$

Integrating the equation over the length of the channel L yields the voltage-current relation of the transistor.

$$I_D = k'_n \frac{W}{L}\left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right] = k_n\left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right] \tag{3.25}$$

k'_n is called the *process transconductance parameter* and equals

$$k'_n = \mu_n C_{ox} = \frac{\mu_n \varepsilon_{ox}}{t_{ox}} \tag{3.26}$$

The product of the process transconductance k'_n and the (W/L) ratio of an (NMOS) transistor is called the *gain factor* k_n of the device. For smaller values of V_{DS}, the quadratic factor in Eq. (3.25) can be ignored, and we observe a linear dependence between V_{DS} and I_D. The operation region where Eq. (3.25) holds is hence called the *resistive* or *linear* region. One of its main properties is that it displays a continuous conductive channel between source and drain regions.

NOTICE: The W and L parameters in Eq. (3.25) represent the *effective channel width and length* of the transistor. These values differ from the dimensions *drawn* on the layout due to effects such as lateral diffusion of the source and drain regions (L), and the encroachment of the isolating field oxide (W). In the remainder of the text, W and L will

always stand for the effective dimensions, while a *d* subscript will be used to indicate the drawn size. The following expressions related the two parameters, with ΔW and ΔL parameters of the manufacturing process:

$$\begin{aligned} W &= W_d - \Delta W \\ L &= L_d - \Delta L \end{aligned} \tag{3.27}$$

The Saturation Region

As the value of the drain-source voltage is further increased, the assumption that the channel voltage is larger than the threshold all along the channel ceases to hold. This happens when $V_{GS} - V(x) < V_T$. At that point, the induced charge is zero, and the conducting channel disappears or is *pinched off*. This is illustrated in Figure 3.16, which shows (in an

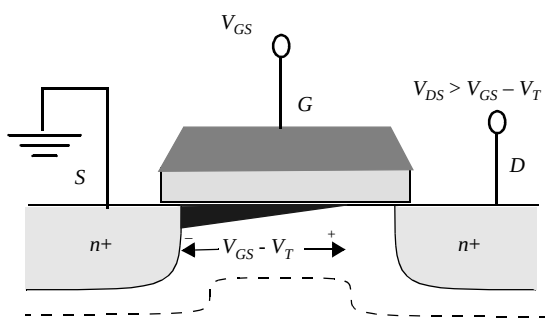


Figure 3.16 NMOS transistor under pinch-off conditions.

exaggerated fashion) how the channel thickness gradually is reduced from source to drain until pinch-off occurs. No channel exists in the vicinity of the drain region. Obviously, for this phenomenon to occur, it is essential that the pinch-off condition be met at the drain region, or

$$V_{GS} - V_{DS} \le V_T. \tag{3.28}$$

Under those circumstances, the transistor is in the *saturation* region, and Eq. (3.25) no longer holds. The voltage difference over the induced channel (from the pinch-off point to the source) remains fixed at $V_{GS} - V_T$, and consequently, the current remains constant (or saturates). Replacing V_{DS} by $V_{GS} - V_T$ in Eq. (3.25) yields the drain current for the saturation mode. It is worth observing that, to a first agree, the current is no longer a function of V_{DS}. Notice also the *squared dependency* of the drain current with respect to the control voltage V_{GS}.

$$I_D = \frac{k'_n}{2}\frac{W}{L}(V_{GS} - V_T)^2 \tag{3.29}$$

Channel-Length Modulation

The latter equation seems to suggest that the transistor in the saturation mode acts as a perfect current source — or that the current between drain and source terminal is a constant, independent of the applied voltage over the terminals. This not entirely correct. The effective length of the conductive channel is actually modulated by the applied V_{DS}: increasing V_{DS} causes the depletion region at the drain junction to grow, reducing the length of the effective channel. As can be observed from Eq. (3.29), the current increases when the length factor L is decreased. A more accurate description of the current of the MOS transistor is therefore given in Eq. (3.30).

$$I_D = I_D'(1 + \lambda V_{DS}) \tag{3.30}$$

with I_D' the current expressions derived earlier, and λ an empirical parameter, called the *channel-length modulation*. Analytical expressions for λ have proven to be complex and inaccurate. λ varies roughly with the inverse of the channel length. In shorter transistors, the drain-junction depletion region presents a larger fraction of the channel, and the channel-modulation effect is more pronounced. It is therefore advisable to resort to long-channel transistors if a high-impedance current source is needed.

Velocity Saturation

The behavior of transistors with very short channel lengths (called *short-channel devices*) deviates considerably from the resistive and saturated models, presented in the previous paragraphs. The main culprit for this deficiency is the *velocity saturation* effect. Eq. (3.23) states that the velocity of the carriers is proportional to the electrical field, independent of the value of that field. In other words, the carrier mobility is a constant. However, at high field strengths, the carriers fail to follow this linear model. In fact, when the electrical field along the channel reaches a critical value ξ_c, the velocity of the carriers tends to saturate due to scattering effects (collisions suffered by the carriers). This is illustrated in Figure 3.17.

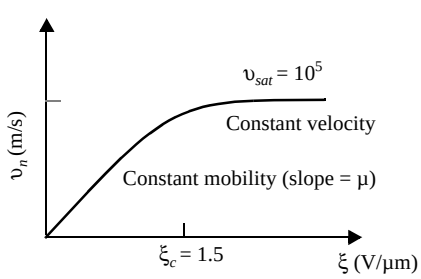


Figure 3.17 Velocity-saturation effect.

For *p*-type silicon, the critical field at which electron saturation occurs is around 1.5 $\times 10^6$ V/m (or 1.5 V/μm), and the saturation velocity υ_{sat} approximately equals 10^5 m/s. This means that in an NMOS device with a channel length of 1 μm, only a couple of volts between drain and source are needed to reach the saturation point. This condition is easily met in current short-channel devices. Holes in a *n*-type silicon saturate at the same velocity, although a higher electrical field is needed to achieve saturation. Velocity-saturation effects are hence less pronounced in PMOS transistors.

This effect has a profound impact on the operation of the transistor. We will illustrate this with a first-order derivation of the device characteristics under velocity-saturating conditions [Ko89]. The velocity as a function of the electrical field, plotted in Figure 3.17, can be roughly approximated by the following expression:

$$\begin{aligned} \upsilon &= \frac{\mu_n \xi}{1 + \xi/\xi_c} \quad \text{for} \quad \xi \le \xi_c \\ &= \upsilon_{sat} \qquad \text{for} \quad \xi \ge \xi_c \end{aligned} \tag{3.31}$$

The continuity requirement between the two regions dictates that $\xi_c = 2\upsilon_{sat}/\mu_n$. Re-evaluation of Eq. (3.20) and Eq. (3.22) in light of revised velocity formula leads to a modified expression of the drain current in the resistive region:

$$I_D = \kappa(V_{DS})\mu_n C_{ox}\frac{W}{L}\left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right] \tag{3.32}$$

with

$$\kappa(V) = \frac{1}{1 + (V/\xi_c L)} \tag{3.33}$$

κ is a measure of the degree of velocity saturation, since V_{DS}/L can be interpreted as the average field in the channel. In case of long-channel devices (large values of L) or small values of V_{DS}, κ approaches 1 and Eq. (3.32) simplifies to the traditional current equation for the resistive operation mode. For short-channel devices, κ is smaller than 1, which means that the delivered current is smaller than what would be normally expected.

When increasing the drain-source voltage, the electrical field in the channel will ultimately reach the critical value, and the carriers at the drain become velocity saturated. The saturation drain voltage V_{DSAT} can be calculated by equating the current at the drain to the current given by Eq. (3.32) for $V_{DS} = V_{DSAT}$. The former is derived from Eq. (3.22), assuming that the drift velocity is saturated and equals υ_{sat}.

$$\begin{aligned} I_{DSAT} &= \upsilon_{sat} C_{ox} W (V_{GT} - V_{DSAT}) \\ &= \kappa(V_{DSAT})\mu_n C_{ox}\frac{W}{L}\left[V_{GT}V_{DSAT} - \frac{V_{DSAT}^2}{2}\right] \end{aligned} \tag{3.34}$$

V_{GT} is a shorthand notation for V_{GS} - V_T. After some algebra, we obtain

$$V_{DSAT} = \kappa(V_{GT})V_{GT} \tag{3.35}$$

Further increasing the drain-source voltage does not yield more current (to a first degree) and the transistor current saturates at I_{DSAT}. This leads to some interesting observations:

- For a short-channel device and for large enough values of V_{GT}, $\kappa(V_{GT})$ is substantially smaller than 1, hence $V_{DSAT} < V_{GT}$. The device enters saturation before V_{DS} reaches V_{GS} - V_T. Short-channel devices therefore experience an extended saturation

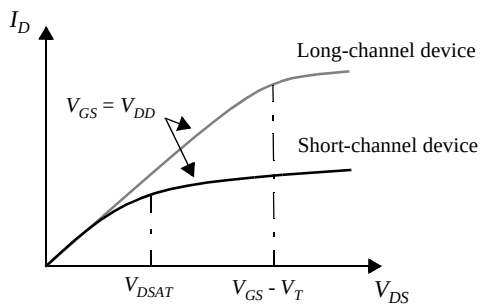


Figure 3.18 Short-channel devices display an extended saturation region due to velocity-saturation.

region, and tend to operate more often in saturation conditions than their long-channel counterparts, as is illustrated in Figure 3.18.

- The saturation current I_{DSAT} displays a *linear dependence* with respect to the gate-source voltage V_{GS}, which is in contrast with the squared dependence in the long-channel device. This reduces the amount of current a transistor can deliver for a given control voltage. On the other hand, reducing the operating voltage does not have such a significant effect in submicron devices as it would have in a long-channel transistor.

The equations above ignore that a larger portion of the channel becomes velocity-saturated with a further increase of V_{DS}. From a modeling perspective, it appears as though the effective channel is shortening with increasing V_{DS}, similar in effect to the channel-length modulation. The resulting increase in current is easily accommodated by introducing an extra $(1 + \lambda \times V_{DS})$ multiplier.

Thus far we have only considered the effects of the tangential field along the channel due to the V_{DS}, when considering velocity-saturation effects. However, there also exists a normal (vertical) field originating from the gate voltage that further inhibits channel carrier mobility. This effect, which is called *mobility degradation*, reduces the surface mobility with respect to the bulk mobility. Eq. (3.36) provides a simple estimation of the mobility reduction,

$$\mu_{n,\,eff} = \frac{\mu_{n0}}{1 + \eta(V_{GS} - V_T)} \tag{3.36}$$

with μ_{n0} the bulk mobility and η an empirical parameter. A typical approach is to use derive the actual value of μ for a given field strength from tables or empirical charts.

Readers interested in a more in-depth perspective on the short-channel effects in MOS transistors are referred to the excellent reference works on this topic, such as [Ko89].

Velocity Saturation — Revisited

Unfortunately, the drain-current equations Eq. (3.32) and Eq. (3.33) are complex expressions of V_{GS} and V_{DS}, which makes them rather unwieldy for a first-order manual analysis. A substantially simpler model can be obtained by making two assumptions:

1. The velocity saturates abruptly at ξ_c, and is approximated by the following expression:

$$\begin{aligned} \upsilon &= \mu_n \xi && \text{for} \quad \xi \le \xi_c \\ &= \upsilon_{sat} = \mu_n \xi_c && \text{for} \quad \xi \ge \xi_c \end{aligned} \tag{3.37}$$

2. The drain-source voltage V_{DSAT} at which the critical electrical field is reached and velocity saturation comes into play is *constant* and is approximated by Eq. (3.38). From Eq. (3.35), it can be observed that this assumption is reasonable for larger values of V_{GT} (>> $\xi_c L$).

$$V_{DSAT} = L\xi_c = \frac{L\upsilon_{sat}}{\mu_n} \tag{3.38}$$

Under these circumstances, the current equations for the resistive region remain unchanged from the long-channel model. Once V_{DSAT} is reached, the current abruptly saturates. The value for I_{DSAT} at that point can be derived by plugging the saturation voltage into the current equation for the resistive region (Eq. (3.25)).

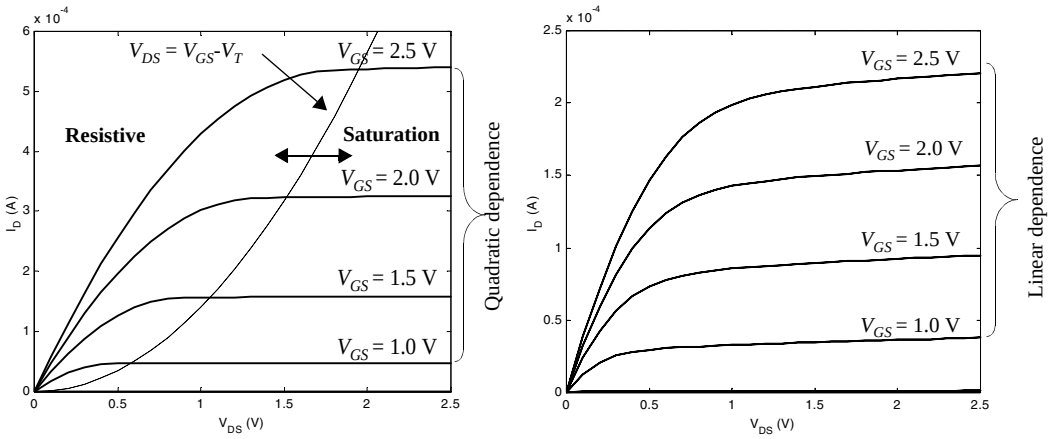
$$\begin{aligned} I_{DSAT} &= I_D(V_{DS} = V_{DSAT}) \\ &= \mu_n C_{ox} \frac{W}{L}\left((V_{GS} - V_T)V_{DSAT} - \frac{V_{DSAT}^2}{2}\right) \\ &= \upsilon_{sat} C_{ox} W\left(V_{GS} - V_T - \frac{V_{DSAT}}{2}\right) \end{aligned} \tag{3.39}$$

This model is truly first-order and empirical. The simplified velocity model causes substantial deviations in the transition zone between linear and velocity-saturated regions. Yet, by carefully choosing the model parameters, decent matching can be obtained with empirical data in the other operation regions, as will be shown in one of the following sections. Most importantly, the equations are coherent with the familiar long-channel equations, and provide the digital designer with a much needed tool for intuitive understanding and interpretation.

Drain Current versus Voltage Charts

The behavior for the MOS transistor in the different operation regions is best understood by analyzing its I_D-V_{DS} curves, which plot I_D versus V_{DS} with V_{GS} as a parameter. Figure 3.19 shows these charts for two NMOS transistors, implemented in the same technology and with the same W/L ratio. One would hence expect both devices to display identical *I-V* characteristics, The main difference however is that the first device has a long channel length (L_d = 10 μm), while the second transistor is a short channel device (L_d = 0.25 μm), and experiences velocity saturation.

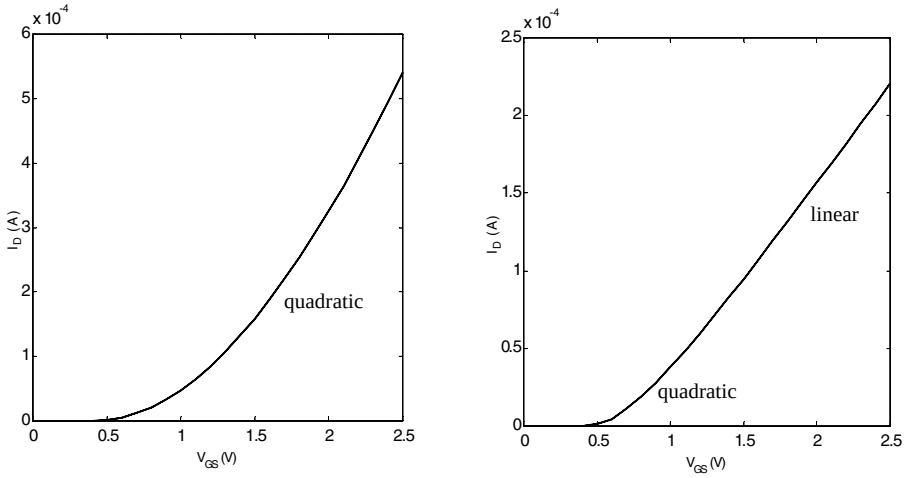
Consider first the long-channel device. In the resistive region, the transistor behaves like a voltage-controlled resistor, while in the saturation region, it acts as a voltage-controlled current source (when the channel-length modulation effect is ignored). The transi-



(a) Long-channel transistor (L_d = 10 μm) (b) Short-channel transistor (L_d = 0.25 μm)

Figure 3.19 *I-V* characteristics of long- and a short-channel NMOS transistors in a 0.25 μm CMOS technology. The (*W*/*L*) ration of both transistors is identical and equals 1.5

tion between both regions is delineated by the V_{DS} = V_{GS} - V_T curve. The squared dependence of I_D as a function of V_{GS} in the saturation region — typical for a long channel device — is clearly observable from the spacing between the different curves. The linear dependence of the saturation current with respect to VGS is apparent in the short-channel device of b. Notice also how velocity-saturation causes the device to saturate for substantially smaller values of V_{DS}. This results in a substantial drop in current drive for high voltage levels. For instance, at (V_{GS} = 2.5 V, V_{DS} = 2.5 V), the drain current of the short transistor is only 40% of the corresponding value of the longer device (220 μA versus 540 μA).



(a) Long-channel device (L_d = 10 μm) (b) Short-channel device (L_d = 0.25 μm)

Figure 3.20 NMOS transistor I_D-V_{GS} characteristic for long and short-channel devices (0.25 μm CMOS technology). *W*/*L* = 1.5 for both transistors and V_{DS} = 2.5 V.

The difference in dependence upon V_{GS} between long- and short-channel devices is even more pronounced in another set of simulated charts that plot I_D as a function of V_{GS} for a fixed value of V_{DS} ($\geq V_{GS}$ — hence ensuring saturation) (Figure 3.20). A quadratic versus linear dependence is apparent for larger values of V_{GS}.

All the derived equations hold for the PMOS transistor as well. The only difference is that **for PMOS devices, the polarities of all voltages and currents are reversed**. This is illustrated in Figure 3.21, which plots the I_D-V_{DS} characteristics of a minimum-size PMOS transistor in our generic 0.25 µm CMOS process. The curves are in the third quadrant as I_D, V_{DS}, and V_{GS} are all negative. Interesting to observe is also that the effects of velocity saturation are less pronounced than in the CMOS devices. This can be attributed to the higher value of the critical electrical field, resulting from the smaller mobility of holes versus electrons.

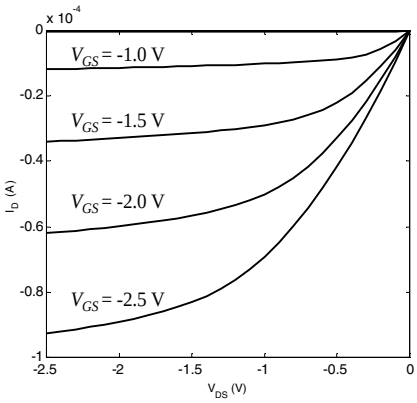


Figure 3.21 *I-V* characteristics of (W_d=0.375 µm, L_d=0.25 µm) PMOS transistor in 0.25 µm CMOS process. Due to the smaller mobility, the maximum current is only 42% of what is achieved by a similar NMOS transistor.

Subthreshold Conduction

A closer inspection of the I_D-V_{GS} curves of Figure 3.20 reveals that the current does not drop abruptly to 0 at $V_{GS} = V_T$. It becomes apparent that the MOS transistor is already partially conducting for voltages below the threshold voltage. This effect is called *subthreshold* or *weak-inversion* conduction. The onset of strong inversion means that ample carriers are available for conduction, but by no means implies that no current at all can flow for gate-source voltages below V_T, although the current levels are small under those conditions. The transition from the on- to the off-condition is thus not abrupt, but gradual.

To study this effect in somewhat more detail, we redraw the I_D versus V_{GS} curve of Figure 3.20b on a logarithmic scale as shown in Figure 3.22. This confirms that the current does not drop to zero immediately for $V_{GS} < V_T$, but actually decays in an exponential fashion, similar to the operation of a bipolar transistor.[2] In the absence of a conducting channel, the n^+ (source) - p (bulk) - n^+ (drain) terminals actually form a parasitic bipolar transistor. The current in this region can be approximated by the expression

[2] Discussion of the operation of bipolar transistors is out of the scope of this textbook. We refer to various textbooks on semiconductor devices, or to the additional information that is available on the web-site of this book.

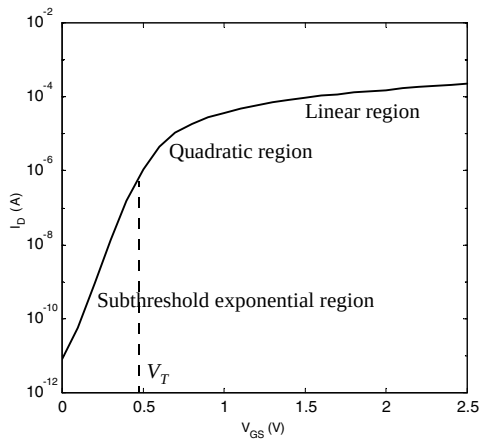


Figure 3.22 I_D current versus V_{GS} (on logarithmic scale), showing the exponential characteristic of the subthreshold region.

$$I_D = I_S e^{\frac{V_{GS}}{nkT/q}}\left(1 - e^{-\frac{V_{DS}}{kT/q}}\right) \tag{3.40}$$

where I_S and n are empirical parameters, with $n \geq 1$ and typically ranging around 1.5.

In most digital applications, the presence of subthreshold current is undesirable as it detracts from the ideal switch-like behavior that we like to assume for the MOS transistor. We would rather have the current drop as fast as possible once the gate-source voltage falls below V_T. The (inverse) rate of decline of the current with respect to V_{GS} below V_T hence is a quality measure of a device. It is often quantified by the *slope factor S*, which measures by how much V_{GS} has to be reduced for the drain current to drop by a factor of 10. From Eq. (3.40), we find

$$S = n\left(\frac{kT}{q}\right)\ln(10) \tag{3.41}$$

with S is expressed in mV/decade. For an ideal transistor with the sharpest possible roll-off, $n = 1$ and $(kT/q)\ln(10)$ evaluates to 60 mV/decade at room temperature, which means that the subthreshold current drops by a factor of 10 for a reduction in V_{GS} of 60 mV. Unfortunately, n is larger than 1 for actual devices and the current falls at a reduced rate (90 mV/decade for $n = 1.5$). The current roll-off is further affected in a negative sense by an increase in the operating temperature (most integrated circuits operate at temperatures considerably beyond room temperature). The value of n is determined by the intrinsic device topology and structure. Reducing its value hence requires a different process technology, such as silicon-on-insulator.

Subthreshold current has some important repercussions. In general, we want the current through the transistor to be as close as possible to zero at $V_{GS} = 0$. This is especially important in the so-called *dynamic circuits*, which rely on the storage of charge on a capacitor and whose operation can be severely degraded by subthreshold leakage. Achieving this in the presence of subthreshold current requires a firm lower bound on the value of the threshold voltage of the devices.

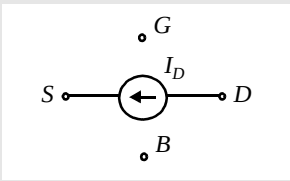
Example 3.6 Subthreshold Slope

For the example of Figure 3.22, a slope of 89.5 mV/decade is observed (between 0.2 and 0.4 V). This is equivalent to an *n*-factor of 1.49.

In Summary – Models for Manual Analysis

The preceding discussions made it clear that the deep-submicron transistor is a complex device. Its behavior is heavily non-linear and is influenced by a large number of second-order effects. Fortunately, accurate circuit-simulation models have been developed that make it possible to predict the behavior of a device with amazing precision over a large range of device sizes, shapes, and operation modes, as we will discuss later in this chapter. While excellent from an accuracy perspective, these models fail in providing a designer with an intuitive insight in the behavior of a circuit and its dominant design parameters. Such an understanding is necessary in the design analysis and optimization process. A designer who misses a clear vision on what drives and governs the circuit operation by necessity resorts on a lengthy trial by error optimization process, that most often leads to an inferior solution.

The obvious question is now how to abstract the behavior of our MOS transistor into a simple and tangible analytical model that does not lead to hopelessly complex equations, yet captures the essentials of the device. It turns out that the first-order expressions, derived earlier in the chapter, can be combined into a single expression that meets these goals. The model presents the transistor as a single current source (Figure 3.23), the value of which is given defined in the Figure. The reader can verify that, depending upon the operating condition, the model simplifies into either Eq. (3.25), Eq. (3.29), or Eq. (3.39) (corrected for channel-length modulation), depending upon operating conditions.



$$I_D = 0 \text{ for } V_{GT} \le 0$$

$$I_D = k'\frac{W}{L}\left(V_{GT}V_{min} - \frac{V_{min}^2}{2}\right)(1 + \lambda V_{DS}) \text{ for } V_{GT} \ge 0$$

with $V_{min} = \min(V_{GT}, V_{DS}, V_{DSAT})$,

$V_{GT} = V_{GS} - V_T$,

and $V_T = V_{T0} + \gamma(\sqrt{|-2\phi_F + V_{SB}|} - \sqrt{|-2\phi_F|})$

Figure 3.23 A unified MOS model for manual analysis.

Besides being a function of the voltages at the four terminals of the transistor, the model employs a set of five parameters: V_{TO}, γ, V_{DSAT}, k', and λ. In principle, it would be possible to determine these parameters from the process technology and from the device physics equations. The complexity of the device makes this a precarious task. A more rewarding approach is to choose the values such that a good matching with the actual device characteristics is obtained. More significantly, the model should match the best in

the regions that matter the most. In digital circuits, this in the region of high V_{GS} and V_{DS}. The performance of an MOS digital circuit is primarily determined by the maximum available current (i.e., the current obtained for V_{GS} = V_{DS} = supply voltage). A good matching in this region is therefore essential.

Example 3.7 Manual Analysis Model for 0.25 μm CMOS Process[3]

Based on the simulated I_D-V_{DS} and I_D-V_{GS} plots of a (W_d = 0.375 μm, L_d = 0.25 μm) transistor, implemented in our generic 0.25 micron CMOS process (Figure 3.19, Figure 3.20), we have derived a set of device parameters to match well in the (V_{DS} = 2.5 V, V_{GS} = 2.5 V) region — 2.5 V being the typical supply voltage for this process. The resulting characteristics are plotted in Figure 3.24 for the NMOS transistor, and compared to the simulated values. Overall, a

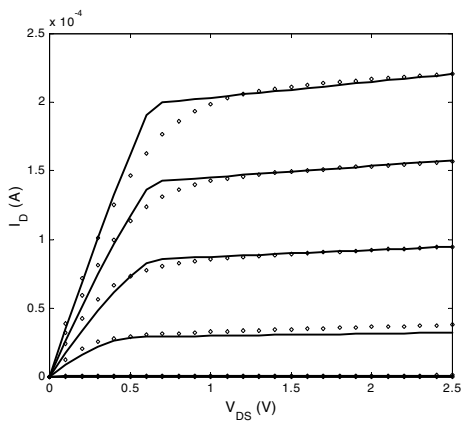


Figure 3.24 Correspondence between simple model (solid line) and SPICE simulation (dotted) for minimum-size NMOS transistor (W_d=0.375 μm, L_d=0.25 μm). Observe the discrepancy in the transition zone between resistive and velocity saturation.

good correspondence can be observed with the exception of the transition region between resistive and velocity-saturation. This discrepancy, which is due the simple velocity model of Eq. (3.37) as explained earlier, is acceptable as it occurs in the lower value-range of V_{DS}. It demonstrates that our model, while simple, manages to give a fair indication of the overall behavior of the device.

Design Data — Transistor Model for Manual Analysis

Table 3.2 tabulates the obtained parameter values for the minimum-sized NMOS and a similarly sized PMOS device in our generic 0.25 μm CMOS process. These values will be used as generic model-parameters in later chapters.

Table 3.2 Parameters for manual model of generic 0.25 μm CMOS process (minimum length device).

	V_{T0} (V)	γ ($V^{0.5}$)	V_{DSAT} (V)	k' (A/V^2)	λ (V^{-1})
NMOS	0.43	0.4	0.63	115×10^{-6}	0.06
PMOS	–0.4	-0.4	-1	-30×10^{-6}	-0.1

[3] A MATLAB implementation of the model is available on the web-site of the textbook.

A word of caution — The model presented here is derived from the characteristics of a single device with a minimum channel-length and width. Trying to extrapolate this behavior to devices with substantially different values of W and L will probably lead to sizable errors. Fortunately, digital circuits typically use only minimum-length devices as these lead to the smallest implementation area. Matching for these transistors will typically be acceptable. It is however advisable to use a different set of model parameters for devices with dramatically different size- and shape-factors.

The presented current-source model will prove to be very useful in the analysis of the basic properties and metrics of a simple digital gate, yet its non-linearity makes it intractable for anything that is somewhat more complex. We therefore introduce an even more simplified model that has the advantage of being linear and straightforward. It is based on the underlying assumption in most digital designs that the transistor is nothing more than a switch with an infinite off-resistance, and a finite on-resistance R_{on}.

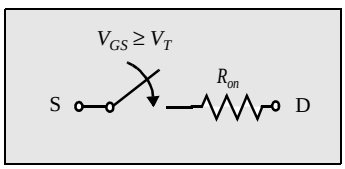


Figure 3.25 NMOS transistor modeled as a switch.

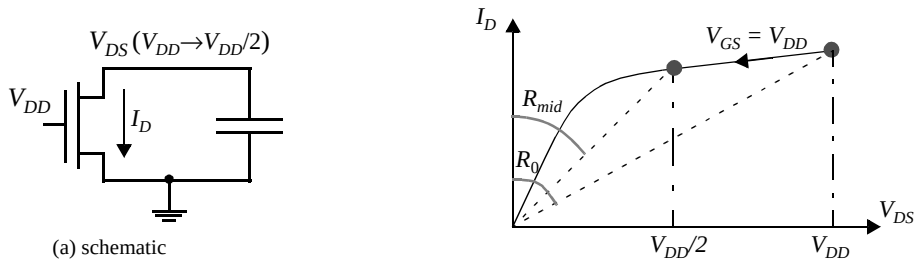
The main problem with this model is that R_{on} is still time-variant, non-linear and depending upon the operation point of the transistor. When studying digital circuits in the transient mode — which means while switching between different logic states — it is attractive to assume R_{on} as a constant and linear resistance R_{eq}, chosen so that the final result is similar to what would be obtained with the original transistor. A reasonable approach in that respect is to use the average value of the resistance over the operation region of interest, or even simpler, the average value of the resistances at the end-points of the transition. The latter assumption works well if the resistance does not experience any strong non-linearities over the range of the averaging interval.

$$R_{eq} = \text{average}_{t = t_1 \ldots t_2}(R_{on}(t)) = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2} R_{on}(t)dt = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2} \frac{V_{DS}(t)}{I_D(t)}dt \approx \frac{1}{2}(R_{on}(t_1) + R_{on}(t_2)) \tag{3.42}$$

Example 3.8 Equivalent resistance when (dis)charging a capacitor

One of the most common scenario's in contemporary digital circuits is the discharging of a capacitor from V_{DD} to GND through an NMOS transistor with its gate voltage set to V_{DD}, or vice-versa the charging of the capacitor to V_{DD} through a PMOS with its gate at GND. Of special interest is the point where the voltage on the capacitor reaches the mid-point ($V_{DD}/2$) — this is by virtue of the definition of the propagation delay as introduced in Chapter 2. Assuming that the supply voltage is substantially larger than the velocity-saturation voltage V_{DSAT} of the transistor, it is fair to state that the transistor stays in velocity saturation for the entire

duration of the transition. This scenario is plotted in for the case of an NMOS discharging a capacitor from V_{DD} to $V_{DD}/2$.



(a) schematic

(b) trajectory traversed on ID-VDS curve.

Figure 3.26 Discharging a capacitor through an NMOS transistor: Schematic (a) and *I-V* trajectory (b). The instantaneous resistance of the transistor equals (V_{DS}/I_D) and is visualized by the angle with respect to the *y*-axis.

With the aid of Eq. (3.42) and Eq. , we can derive the value of the equivalent resistance, which averages the resistance of the device over the interval.

$$R_{eq} = \frac{1}{-V_{DD}/2}\int_{V_{DD}}^{V_{DD}/2} \frac{V}{I_{DSAT}(1+\lambda V)}dV \approx \frac{3}{4}\frac{V_{DD}}{I_{DSAT}}\left(1-\frac{7}{9}\lambda V_{DD}\right)$$
$$\text{with} \quad I_{DSAT} = k'\frac{W}{L}\left((V_{DD}-V_T)V_{DSAT}-\frac{V_{DSAT}^2}{2}\right) \tag{3.43}$$

A similar result can be obtained by just averaging the values of the resistance at the end points (and simplifying the result using a Taylor expansion):

$$R_{eq} = \frac{1}{2}\left(\frac{V_{DD}}{I_{DSAT}(1+\lambda V_{DD})}+\frac{V_{DD}/2}{I_{DSAT}(1+\lambda V_{DD}/2)}\right) \approx \frac{3}{4}\frac{V_{DD}}{I_{DSAT}}\left(1-\frac{5}{6}\lambda V_{DD}\right) \tag{3.44}$$

A number of conclusions are worth drawing from the above expressions:

- The resistance is inversely proportional to the (W/L) ratio of the device. Doubling the transistor width halves the resistance.
- For $V_{DD} >> V_T + V_{DSAT}/2$, the resistance becomes virtually independent of the supply voltage. This is confirmed in halves, which plots the simulated equivalent resistance as a function of the supply voltage V_{DD}. Only a minor improvement in resistance, attributable to the channel-length modulation, can be observed when raising the supply voltage.
- Once the supply voltage approaches V_T, a dramatic increase in resistance can be observed.

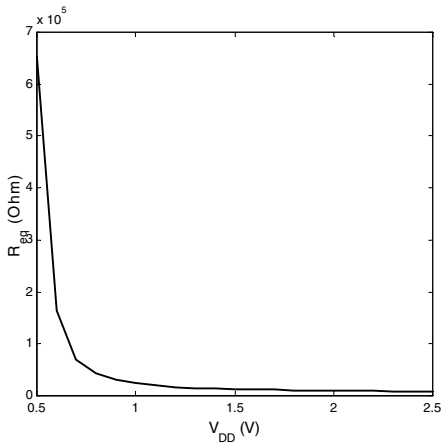


Figure 3.27 Simulated equivalent resistance of a minimum size NMOS transistor in 0.25 µm CMOS process as a function of V_{DD} ($V_{GS} = V_{DD}$, $V_{DS} = V_{DD} \rightarrow V_{DD}/2$).

Design Data — Equivalent Resistance Model

Table 3.3 enumerates the equivalent resistances obtained by simulation of our generic 0.25 µm CMOS process. These values will come in handy when analyzing the performance of CMOS gates in later chapters.

Table 3.3 Equivalent resistance R_{eq} ($W/L = 1$) of NMOS and PMOS transistors in 0.25 µm CMOS process (with $L = L_{min}$). For larger devices, divide R_{eq} by W/L.

V_{DD} (V)	1	1.5	2	2.5
NMOS (kΩ)	35	19	15	13
PMOS (kΩ)	115	55	38	31

3.3.3 Dynamic Behavior

The dynamic response of a MOSFET transistor is a sole function of the time it takes to (dis)charge the parasitic capacitances that are intrinsic to the device, and the extra capacitance introduced by the interconnecting lines (and are the subject of Chapter 4). A profound understanding of the nature and the behavior of these intrinsic capacitances is essential for the designer of high-quality digital integrated circuits. They originate from three sources: the basic MOS structure, the channel charge, and the depletion regions of the reverse-biased *pn*-junctions of drain and source. Aside from the MOS structure capacitances, all capacitors are nonlinear and vary with the applied voltage, which makes their analysis hard. We discuss each of the components in turn.

MOS Structure Capacitances

The gate of the MOS transistor is isolated from the conducting channel by the gate oxide that has a capacitance per unit area equal to $C_{ox} = \varepsilon_{ox} / t_{ox}$. We learned earlier that from a *I-V* perspective it is useful to have C_{ox} as large as possible, or to keep the oxide thickness very thin. The total value of this capacitance is called the *gate capacitance* C_g and can be decomposed into two elements, each with a different behavior. Obviously, one part of C_g contributes to the channel charge, and is discussed in a subsequent section. Another part is solely due to the topological structure of the transistor. This component is the subject of the remainder of this section.

Consider the transistor structure of Figure 3.28. Ideally, the source and drain diffusion should end right at the edge of the gate oxide. In reality, both source and drain tend to extend somewhat below the oxide by an amount x_d, called the *lateral diffusion*. Hence, the effective channel of the transistor L becomes shorter than the drawn length L_d (or the length the transistor was originally designed for) by a factor of $\Delta L = 2x_d$. It also gives rise to a parasitic capacitance between gate and source (drain) that is called the *overlap capacitance*. This capacitance is strictly linear and has a fixed value

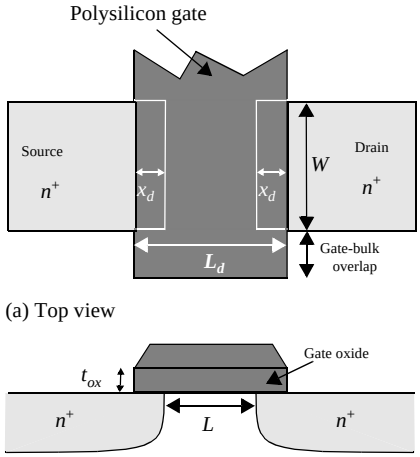


Figure 3.28 MOSFET overlap capacitance.

$$C_{GSO} = C_{GDO} = C_{ox}x_dW = C_oW \tag{3.45}$$

Since x_d is a technology-determined parameter, it is customary to combine it with the oxide capacitance to yield the overlap capacitance per unit transistor width C_o (more specifically, C_{gso} and C_{gdo}).

Channel Capacitance

Perhaps the most significant MOS parasitic circuit element, the gate-to-channel capacitance C_{GC} varies in both magnitude and in its division into three components C_{GCS}, C_{GCD}, and C_{GCB} (being the gate-to-source, gate-to-drain, and gate-to-body capacitances, respectively), depending upon the operation region and terminal voltages. This varying distribution is best explained with the simple diagrams of Figure 3.29. When the transistor is in

cut-off (a), no channel exists, and the total capacitance C_{GC} appears between gate and body. In the resistive region (b), an inversion layer is formed, which acts as a conductor between source and drain. Consequently, $C_{GCB} = 0$ as the body electrode is shielded from the gate by the channel. Symmetry dictates that the capacitance distributes evenly between source and drain. Finally, in the saturation mode (c), the channel is pinched off. The capacitance between gate and drain is approximately zero, and so is the gate-body capacitance. All the capacitance hence is between gate and source.

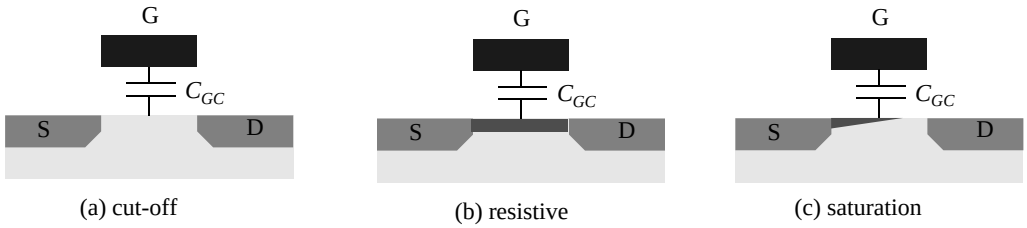
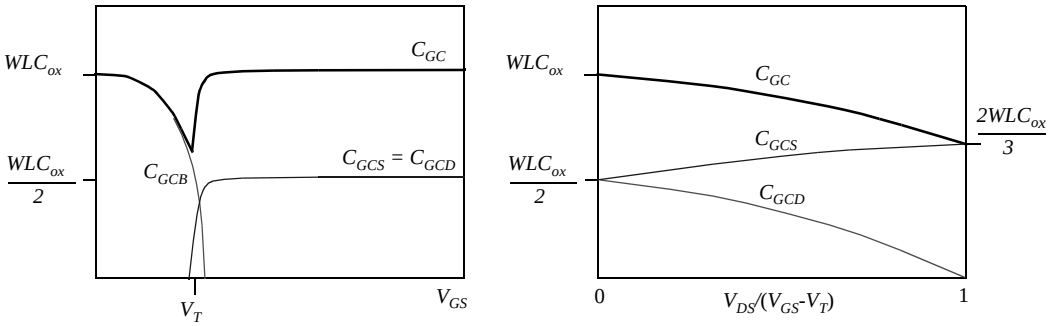


Figure 3.29 The gate-to-channel capacitance and how the operation region influences is distribution over the three other device terminals.

To actual value of the total gate-channel capacitance and its distribution over the three components is best understood with the aid of a number of charts. The first plot (Figure 3.30a) captures the evolution of the capacitance as a function of V_{GS} for $V_{DS} = 0$. For $V_{GS} = 0$, the transistor is off, no channel is present and the total capacitance, equal to WLC_{ox}, appears between gate and body. When increasing V_{GS}, a depletion region forms under the gate. This seemingly causes the thickness of the gate dielectric to increase, which means a reduction in capacitance. Once the transistor turns on ($V_{GS} = V_T$), a channel is formed and C_{GCB} drops to 0. With $V_{DS} = 0$, the device operates in the resistive mode and the capacitance divides equally between source and drain, or $C_{GCS} = C_{GCD} = WLC_{ox}/2$. The large fluctuation of the channel capacitance around $V_{GS}=V_T$ is worth remembering. A designer looking for a well-behaved linear capacitance should avoid operation in this region.



(a) C_{GC} as a function of V_{GS} (with V_{DS}=0) (b) C_{GC} as a function of the degree of saturation

Figure 3.30 Distribution of the gate-channel capacitance as a function of V_{GS} and V_{DS} (from [Dally98]).

Once the transistor is on, the distribution of its gate capacitance depends upon the degree of saturation, measured by the $V_{DS}/(V_{GS}-V_T)$ ratio. As illustrated in Figure 3.30b,

C_{GCD} gradually drops to 0 for increasing levels of saturation, while C_{GCS} increases to 2/3 $C_{ox}WL$. This also means that the total gate capacitance is getting smaller with an increased level of saturation.

From the above, it becomes clear that the gate-capacitance components are nonlinear and varying with the operating voltages. To make a first-order analysis possible, we will use a simplified model with a constant capacitance value in each region of operation in the remainder of the text. The assumed values are summarized in Table 3.4.

Table 3.4 Average distribution of channel capacitance of MOS transistor for different operation regions.

Operation Region	C_{GCB}	C_{GCS}	C_{GCD}	C_{GC}	C_G
Cutoff	$C_{ox}WL$	0	0	$C_{ox}WL$	$C_{ox}WL+2C_oW$
Resistive	0	$C_{ox}WL$ / 2	$C_{ox}WL$ / 2	$C_{ox}WL$	$C_{ox}WL+2C_oW$
Saturation	0	$(2/3)C_{ox}WL$	0	$(2/3)C_{ox}WL$	$(2/3)C_{ox}WL+2C_oW$

Example 3.9 Using a circuit simulator to extract capacitance

Determining the value of the parasitic capacitances of an MOS transistor for a given operation mode is a labor-intensive task, and requires the knowledge of a number of technology parameters that are often not explicitly available. Fortunately, once a SPICE model of the transistor is attained, a simple simulation can give you the data you are interested in. Assume we would like to know the value of the total gate capacitance of a transistor in a given technology as a function of V_{GS} (for $V_{DS} = 0$). A simulation of the circuit of Figure 3.31a will give us exactly this information. In fact, the following relation is valid:

$$I = C_G(V_{GS})\frac{\mathrm{d}V_{GS}}{\mathrm{d}t}$$

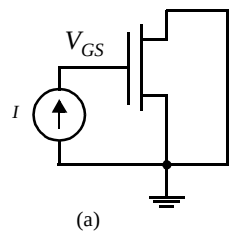
which can be rewritten to yield an expression for C_G.

$$C_G(V_{GS}) = I/\left(\frac{\mathrm{d}V_{GS}}{\mathrm{d}t}\right)$$

A transient simulation gives us V_{GS} as a function of time, which can be translated into the capacitance with the aid of some simple mathematical manipulations. This is demonstrated in Figure 3.31b, which plots the simulated gate capacitance of a minimum size 0.25 μm NMOS transistor as a function of V_{GS}. The graphs clearly shows the drop of the capacitance when V_{GS} approaches V_T and the discontinuity at V_T, predicted in Figure 3.30.

Junction Capacitances

A final capacitive component is contributed by the reverse-biased source-body and drain-body *pn*-junctions. The depletion-region capacitance is nonlinear and decreases when the reverse bias is raised as discussed earlier. To understand the components of the junction capacitance (often called the *diffusion capacitance*), we must look at the source (drain)



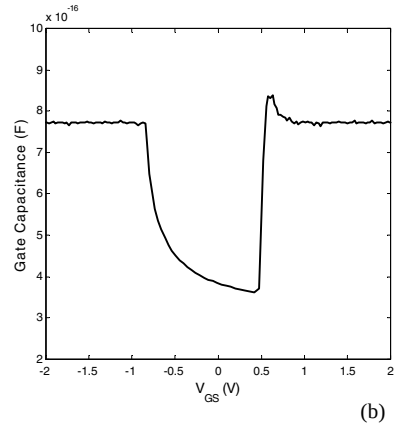


Figure 3.31 Simulating the gate capacitance of an MOS transistor; (a) circuit configuration used for the analysis, (b) resulting capacitance plot for minimum-size NMOS transistor in 0.25 μm technology.

region and its surroundings. The detailed picture, shown in Figure 3.32, shows that the junction consists of two components:

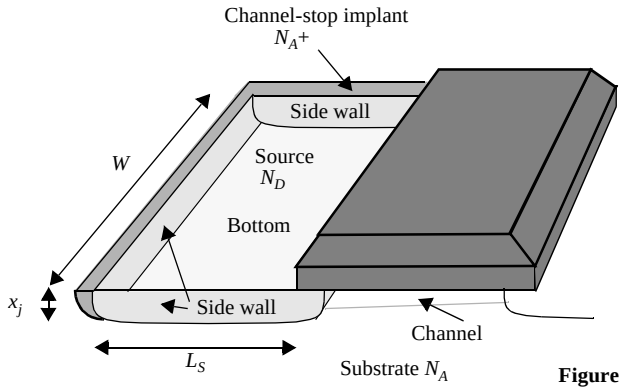


Figure 3.32 Detailed view of source junction.

- The *bottom-plate* junction, which is formed by the source region (with doping N_D) and the substrate with doping N_A. The total depletion region capacitance for this component equals $C_{bottom} = C_j W L_S$, with C_j the junction capacitance per unit area as given by Eq. (3.9). As the bottom-plate junction is typically of the abrupt type, the grading coefficient m approaches 0.5.

- The *side-wall* junction, formed by the source region with doping N_D and the p^+ channel-stop implant with doping level N_A^+. The doping level of the stopper is usually larger than that of the substrate, resulting in a larger capacitance per unit area. The side-wall junction is typically graded, and its grading coefficient varies from 0.33 to 0.5. Its capacitance value equals $C_{sw} = C'_{jsw} x_j (W + 2 \times L_s)$. Notice that no side-wall capacitance is counted for the fourth side of the source region, as this represents the conductive channel.[4]
  Since x_j, the junction depth, is a technology parameter, it is normally combined with C'_{jsw} into a capacitance per unit perimeter $C_{jsw} = C'_{jsw} x_j$. An expression for the total junction capacitance can then be derived,

$$\begin{aligned} C_{diff} &= C_{bottom} + C_{sw} = C_j \times AREA + C_{jsw} \times PERIMETER \\ &= C_j L_S W + C_{jsw}(2L_S + W) \end{aligned} \tag{3.46}$$

Since all these capacitances are small-signal capacitances, we normally linearize them and use average capacitances along the lines of Eq. (3.10).

Problem 3.1 Using a circuit simulator to determine the drain capacitance

Derive a simple circuit that would help you to derive the drain capacitance of an NMOS transistor in the different operation modes using circuit simulation (in the style of Figure 3.31).

Capacitive Device Model

All the above contributions can be combined in a single capacitive model for the MOS transistor, which is shown Figure 3.33. Its components are readily identified on the basis of the preceding discussions.

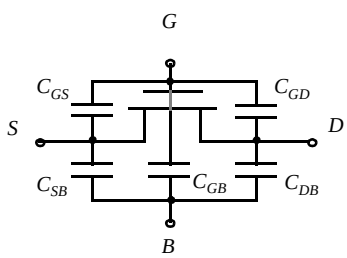


Figure 3.33 MOSFET capacitance model.

$$C_{GS} = C_{GCS} + C_{GSO};\ C_{GD} = C_{GCD} + C_{GDO};\ C_{GB} = C_{GCB}$$
$$C_{SB} = C_{Sdiff};\ C_{DB} = C_{Ddiff} \tag{3.47}$$

It is essential for the designers of high-performance and low-energy circuits to be very familiar with this model as well as to have an intuitive feeling of the relative values of its components.

Example 3.10 MOS Transistor Capacitances

Consider an NMOS transistor with the following parameters: t_{ox} = 6 nm, L = 0.24 μm, W = 0.36 μm, $L_D = L_S$ = 0.625 μm, $C_O = 3 \times 10^{-10}$ F/m, $C_{j0} = 2 \times 10^{-3}$ F/m^2, $C_{jsw0} = 2.75 \times 10^{-10}$ F/m. Determine the zero-bias value of all relevant capacitances.

The gate capacitance per unit area is easily derived as ($\varepsilon_{ox} / t_{ox}$) and equals 5.7 fF/μm^2. The gate-to-channel C_{GC} then equals WLC_{ox} = 0.49 fF. To find the total gate capacitance, we have to add the source and drain overlap capacitors, each of which equals WC_O = 0.105 fF. This leads to a total gate capacitance of 0.7 fF.

[4] To be entirely correct, we should take the diffusion capacitance of the source(drain)-to-channel junction into account. Due to the doping conditions and the small area, this component can virtually always be ignored in a first-order analysis. Detailed SPICE models most often include a factor C_{JSWG} to account for this junction.

The diffusion capacitance consists of the bottom and the side-wall capacitances. The former is equal to $C_{j0} L_D W = 0.45$ fF, while the side-wall capacitance under zero-bias conditions evaluates to $C_{jsw0} (2L_D + W) = 0.44$ fF. This results in a total drain(source)-to-bulk capacitance of 0.89 fF.

The diffusion capacitance seems to dominate the gate capacitance. This is a worst-case condition, however. When increasing the value of the reverse bias over the junction — as is the normal operation mode in MOS circuits —, the diffusion capacitance is substantially reduced. Also, clever design can help to reduce the value of L_D (L_S). In general, it can be stated that the contribution of diffusion capacitances is at most equal, and very often substantially smaller than the gate capacitance.

Design Data — MOS Transistor Capacitances

Table 3.5 summarizes the parameters needed to estimate the parasitic capacitances of the MOS transistors in our generic 0.25 μm CMOS process.

Table 3.5 Capacitance parameters of NMOS and PMOS transistors in 0.25 μm CMOS process.

	C_{ox} (fF/μm^2)	C_O (fF/μm)	C_j (fF/μm^2)	m_j	ϕ_b (V)	C_{jsw} (fF/μm)	m_{jsw}	ϕ_{bsw} (V)
NMOS	6	0.31	2	0.5	0.9	0.28	0.44	0.9
PMOS	6	0.27	1.9	0.48	0.9	0.22	0.32	0.9

Source-Drain Resistance

The performance of a CMOS circuit may further be affected by another set of parasitic elements, being the resistances in series with the drain and source regions, as shown in Figure 3.34a. This effect become more pronounced when transistors are scaled down, as this leads to shallower junctions and smaller contact openings become smaller. The resistance of the drain (source) region can be expressed as

$$R_{S,D} = \frac{L_{S,D}}{W} R_{\square} + R_C \tag{3.48}$$

with R_C the contact resistance, W the width of the transistor, and $L_{S,D}$ the length of the source or drain region (Figure 3.34b). $R_{\square}$ is the *sheet resistance* per square of the drain-source diffusion, and ranges from 20 to 100 Ω/❑. Observe that the resistance of a square of material is constant, independent of its size (see also Chapter 4).

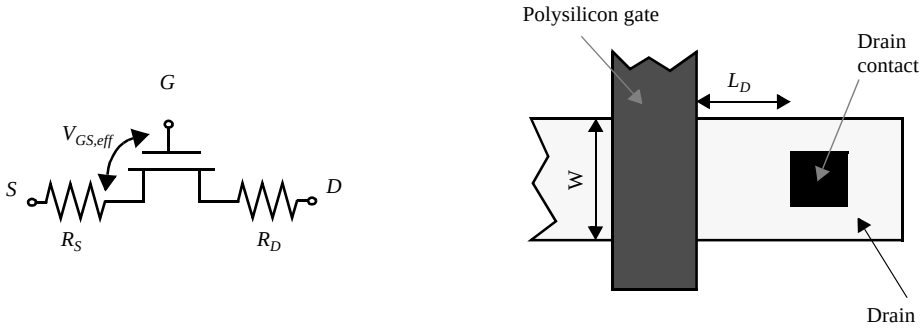
The series resistance causes a deterioration in the device performance, as it reduces the drain current for a given control voltage. Keeping its value as small as possible is thus an important design goal for both the device and the circuit engineer. One option, popular in most contemporary processes, is to cover the drain and source regions with a low-resistivity material such as titanium or tungsten. This process is called *silicidation* and effec-

tively reduces the sheet resistance to values in the range from 1 to 4 Ω/❑.[5] Making the transistor wider than needed is another possibility as should be obvious from Eq. (3.48). With a process that includes silicidation and proper attention to layout, parasitic resistance is not important. However, the reader should be aware that careless layout may lead to resistances that severely degrade the device performance.

3.3.4 The Actual MOS Transistor—Some Secondary Effects

The operation of a contemporary transistor may show some important deviations from the model we have presented so far. These divergences become especially pronounced once the dimensions of the transistor reach the deep sub-micron realm. At that point, the assumption that the operation of a transistor is adequately described by a one-dimensional model, where it is assumed that all current flows on the surface of the silicon and the electrical fields are oriented along that plane, is not longer valid. Two- or even three-dimensional models are more appropriate. An example of such was already given in Section 3.2.2 when we discussed the mobility degradation.

The understanding of some of these second-order effects and their impact on the device behavior is essential in the design of today's digital circuits and therefore merits some discussion. One word of warning, though. Trying to take all those effects into account in a manual, first-order analysis results in intractable and opaque circuit models. It is therefore advisable to analyze and design MOS circuits first using the ideal model. The impact of the non-idealities can be studied in a second round using computer-aided simulation tools with more precise transistor models.

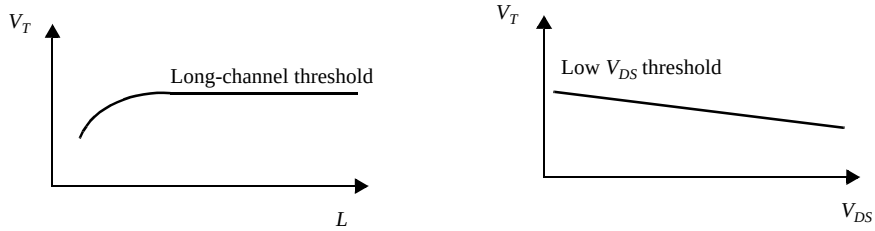


(a) Modeling the series resistance

(b) Parameters of the series resistance

Figure 3.34 Series drain and source resistance.

[5] Silicidation is also used to reduce the resistance of the polysilicon gate, as will be discussed in Chapter 4.



(a) Threshold as a function of the length (for low V_{DS})

(b) Drain-induced barrier lowering (for low L)

Figure 3.35 Threshold variations.

Threshold Variations

Eq. (3.19) states that the threshold voltage is only a function of the manufacturing technology and the applied body bias V_{SB}. The threshold can therefore be considered as a constant over all NMOS (PMOS) transistors in a design. As the device dimensions are reduced, this model becomes inaccurate, and the threshold potential becomes a function of L, W, and V_{DS}. Two-dimensional second-order effects that were ignorable for long-channel devices suddenly become significant.

In the traditional derivation of the V_{TO}, for instance, it is assumed that the channel depletion region is solely due to the applied gate voltage and that all depletion charge beneath the gate originates from the MOS field effects. This ignores the depletion regions of the source and reverse-biased drain junction, which become relatively more important with shrinking channel lengths. Since a part of the region below the gate is already depleted (by the source and drain fields), a smaller threshold voltage suffices to cause strong inversion. In other words, V_{T0} decreases with L for short-channel devices (Figure 3.35a). A similar effect can be obtained by raising the drain-source (bulk) voltage, as this increases the width of the drain-junction depletion region. Consequently, the threshold decreases with increasing $V_{DS.}$ This effect, called the *drain-induced barrier lowering*, or *DIBL*, causes the threshold potential to be a function of the operating voltages (Figure 3.35b). For high enough values of the drain voltage, the source and drain regions can even be shorted together, and normal transistor operation ceases to exist. The sharp increase in current that results from this effect, which is called *punch-through*, may cause permanent damage to the device and should be avoided. Punch-through hence sets an upper bound on the drain-source voltage of the transistor.

Since the majority of the transistors in a digital circuit are designed at the minimum channel length, the variation of the threshold voltage as a function of the length is almost uniform over the complete design, and is therefore not much of an issue except for the increased sub-threshold leakage currents. More troublesome is the DIBL, as this effect varies with the operating voltage. This is, for instance, a problem in dynamic memories, where the leakage current of a cell (being the subthreshold current of the access transistor) becomes a function of the voltage on the data-line, which depends upon the applied data patterns. From the cell perspective, DIBL manifests itself as a data-dependent noise source.

Worth mentioning is that the threshold of the MOS transistor is also subject to *narrow-channel* effects. The depletion region of the channel does not stop abruptly at the edges of the transistor, but extends somewhat under the isolating field-oxide. The gate voltage must support this extra depletion charge to establish a conducting channel. This effect is ignorable for wide transistors, but becomes significant for small values of W, where it results in an increase of the threshold voltage. For small geometry transistors, with small values of L and W, the effects of short- and narrow channels may tend to cancel each other out.

Hot-Carrier Effects

Besides varying over a design, threshold voltages in short-channel devices also have the tendency to *drift over time*. This is the result of the *hot-carrier* effect [Hu92]. Over the last decades, device dimensions have been scaled down continuously, while the power supply and the operating voltages were kept constant. The resulting increase in the electrical field strength causes an increasing velocity of the electrons, which can leave the silicon and tunnel into the gate oxide upon reaching a high-enough energy level. Electrons trapped in the oxide change the threshold voltage, typically increasing the thresholds of NMOS devices, while decreasing the V_T of PMOS transistors. For an electron to become hot, an electrical field of at least 10^4 V/cm is necessary. This condition is easily met in devices with channel lengths around or below 1 μm. The hot-electron phenomenon can lead to a long-term reliability problem, where a circuit might degrade or fail after being in use for a while. This is illustrated in Figure 3.36, which shows the degradation in the *I-V* characteristics of an NMOS transistor after it has been subjected to extensive operation. State-of-the-art MOSFET technologies therefore use specially-engineered drain and source regions to ensure that the peaks in the electrical fields are bounded, hence preventing carriers to reach the critical values necessary to become hot. The reduced supply voltage that is typical for deep sub-micron technologies can in part be attributed to the necessity to keep hot-carrier effects under control.

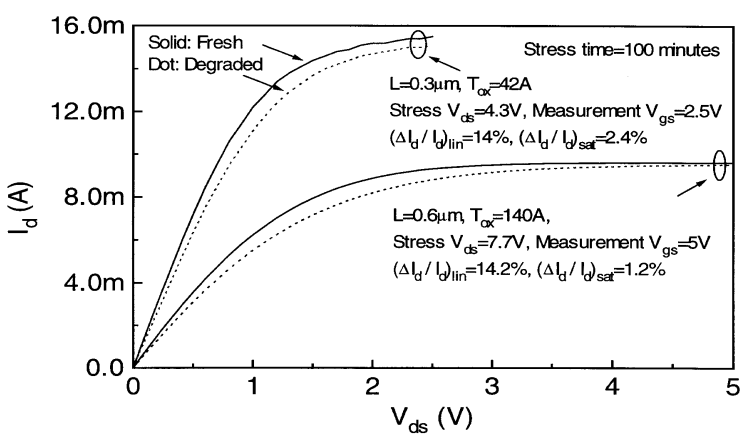
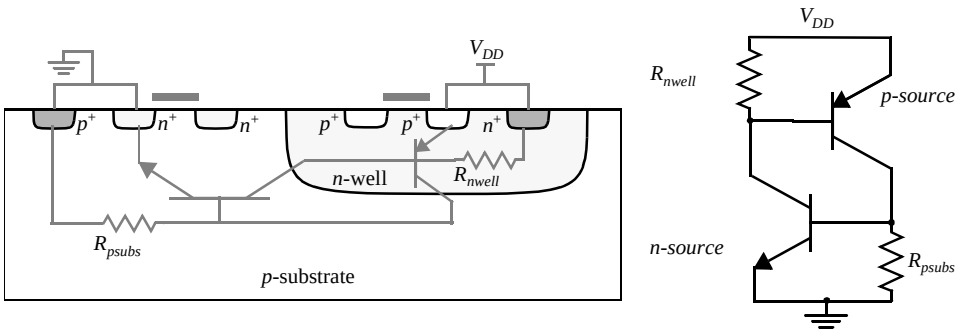


Figure 3.36 Hot-carrier effects cause the *I-V* characteristics of an NMOS transistor to degrade from extensive usage (from [McGaughy98]).

CMOS Latchup

The MOS technology contains a number of intrinsic bipolar transistors. These are especially troublesome in CMOS processes, where the combination of wells and substrates results in the formation of parasitic *n-p-n-p* structures. Triggering these thyristor-like devices leads to a shorting of the V_{DD} and V_{SS} lines, usually resulting in a destruction of the chip, or at best a system failure that can only be resolved by power-down.

Consider the *n*-well structure of Figure 3.37a. The *n-p-n-p* structure is formed by the source of the NMOS, the *p*-substrate, the *n*-well and the source of the PMOS. A circuit equivalent is shown in Figure 3.37b. When one of the two bipolar transistors gets forward biased (e.g., due to current flowing through the well, or substrate), it feeds the base of the other transistor. This positive feedback increases the current until the circuit fails or burns out.



(a) Origin of latchup (b) Equivalent circuit

Figure 3.37 CMOS latchup.

From the above analysis the message to the designer is clear—to avoid latchup, the resistances R_{nwell} and R_{psubs} should be minimized. This can be achieved by providing numerous well and substrate contacts, placed close to the source connections of the NMOS/PMOS devices. Devices carrying a lot of current (such as transistors in the I/O drivers) should be surrounded by *guard rings*. These circular well/substrate contacts, positioned around the transistor, reduce the resistance even further and reduce the gain of the parasitic bipolars. For an extensive discussion on how to avoid latchup, please refer to [Weste93]. The latchup effect was especially critical in early CMOS processes. In recent years, process innovations and improved design techniques have all but eliminated the risks for latchup.

3.3.5 SPICE Models for the MOS Transistor

The complexity of the behavior of the short-channel MOS transistor and its many parasitic effects has led to the development of a wealth of models for varying degrees of accuracy and computing efficiency. In general, more accuracy also means more complexity and, hence, an increased run time. In this section, we briefly discuss the characteristics of the

more popular MOSFET models, and describe how to instantiate a MOS transistor in a circuit description.

SPICE Models

SPICE has three built-in MOSFET models, selected by the LEVEL parameter in the model card. Unfortunately, all these models have been rendered obsolete by the progression to short-channel devices. They should only be used for first-order analysis, and we therefore limit ourselves to a short discussion of their main properties.

- The LEVEL 1 SPICE model implements the *Shichman-Hodges model*, which is based on the square law long-channel expressions, derived earlier in this chapter. It does not handle short-channel effects.
- The LEVEL 2 model is a geometry-based model, which uses detailed device physics to define its equations. It handles effects such as velocity saturation, mobility degradation, and drain-induced barrier lowering. Unfortunately, including all 3D-effects of an advanced submicron process in a pure physics-based model becomes complex and inaccurate.
- LEVEL 3 is a semi-empirical model. It relies on a mixture of analytical and empirical expressions, and uses measured device data to determine its main parameters. It works quite well for channel lengths down to 1 μm.

In response to the inadequacy of the built-in models, SPICE vendors and semi-conductor manufacturers have introduced a wide range of accurate, but proprietary models. A complete description of all those would take the remainder of this book, which is, obviously, not the goal. We refer the interested reader to the extensive literature on this topic [e.g. Vladimirescu93].

The BSIM3V3 SPICE Model

The confusing situation of having to use a different model for each manufacturer has fortunately been partially resolved by the adoption of the BSIM3v3 model as an industry-wide standard for the modeling of deep-submicron MOSFET transistors. The **B**erkeley **S**hort-Channel **I**GFET **M**odel (or BSIM in short) provides a model that is analytically simple and is based on a ‘small’ number of parameters, which are normally extracted from experimental data. Its popularity and accuracy make it the natural choice for all the simulations presented in this book.

A full-fledged BSIM3v3 model (denoted as LEVEL 49) contains over 200 parameters, the majority of which are related to the modeling of second-order effects. Fortunately, understanding the intricacies of all these parameters is not a requirement for the digital designer. We therefore only present an overview of the parameter categories (Table 3.6). The *Bin* category deserves some extra attention. Providing a single set of parameters that is acceptable over all possible device dimensions is deemed to be next to impossible. So, a set of models is provided, each of which is valid for a limited region delineated by

LMIN, LMAX, WMIN, and WMAX (called a bin). It is typically left to the user to select the correct bin for a particular transistor.

Table 3.6 BSIM3-V3 model parameter categories, and some important parameters.

Parameter Category	Description
Control	Selection of level and models for mobility, capacitance, and noise LEVEL, MOBMOD, CAPMOD
DC	Parameters for threshold and current calculations VTH0, K1, U0, VSAT, RSH,
AC & Capacitance	Parameters for capacitance computations CGS(D)O, CJ, MJ, CJSW, MJSW
dW and dL	Derivation of effective channel length and width
Process	Process parameters such as oxide thickness and doping concentrations TOX, XJ, GAMMA1, NCH, NSUB
Temperature	Nominal temperature and temperature coefficients for various device parameters TNOM
Bin	Bounds on device dimensions for which model is valid LMIN, LMAX, WMIN, WMAX
Flicker Noise	Noise model parameters

We refer the interested reader to the BSIM3v3 documentation provided on the website of the textbook (REFERENCE) for a complete description of the model parameters and equations. The LEVEL-49 models for our generic 0.25 μm CMOS process can be found at the same location.

Transistor Instantiation

The parameters that can be specified for an individual transistor are enumerated in Table 3.7. Not all these parameters have to be defined for each transistor. SPICE assumes default values (which are often zero!) for the missing factors.

WARNING: It is hard to expect accuracy from a simulator, when the circuit description provided by the designer does not contain the necessary details. For instance, you must accurately specify the area and the perimeter of the source and drain regions of the devices when performing a performance analysis. Lacking this information, which is used for the computation of the parasitic capacitances, your transient simulation will be next to useless. Similarly, it is often necessary to painstakingly define the value of the drain and source resistance. The NRS and NRD values multiply the sheet resistance specified in the transistor model for an accurate representation of the parasitic series source and drain resistance of each transistor.

Table 3.7 SPICE transistor parameters.

Parameter Name	Symbol	SPICE Name	Units	Default Value
Drawn Length	*L*	L	m	–
Effective Width	*W*	W	m	–
Source Area	*AREA*	AS	m^2	0
Drain Area	*AREA*	AD	m^2	0
Source Perimeter	*PERIM*	PS	m	0
Drain Perimeter	*PERIM*	PD	m	0
Squares of Source Diffusion		NRS	–	1
Squares of Drain Diffusion		NRD	–	1

Example 3.11 SPICE description of a CMOS inverter

An example of a SPICE description of a CMOS inverter, consisting of an NMOS and a PMOS transistor, is given below. Transistor M1 is an NMOS device of model-type (and bin) *nmos*.1 with its drain, gate, source, and body terminals connected to nodes *nvout*, *nvin*, 0, and 0, respectively. Its gate length is the minimum allowed in this technology (0.25 μm). The '+' character at the start of line 2 indicates that this line is a continuation of the previous one.

The PMOS device of type *pmos*.1, connected between nodes *nvout*, *nvin*, *nvdd*, and *nvdd* (D, G, S, and B, respectively), is three times wider, which reduces the series resistance, but increases the parasitic diffusion capacitances as the area and perimeter of the drain and source regions go up.

Finally, the *.lib* line refers to the file that contains the transistor models.

```
M1 nvout nvin 0 0 nmos.1 W=0.375U L=0.25U
+AD=0.24P PD=1.625U AS=0.24P PS=1.625U NRS=1 NRD=1
M2 nvout nvin nvdd nvdd pmos.1 W=1.125U L=0.25U
+AD=0.7P PD=2.375U AS=0.7P PS=2.375U NRS=0.33 NRD=0.33
.lib 'c:\Design\Models\cmos025.l'
```

3.4 A Word on Process Variations

The preceding discussions have assumed that a device is adequately modeled by a single set of parameters. In reality, the parameters of a transistor vary from wafer to wafer, or even between transistors on the same die, depending upon the position. This observed random distribution between supposedly identical devices is primarily the result of two factors:

1. Variations in the process parameters, such as impurity concentration densities, oxide thicknesses, and diffusion depths, caused by nonuniform conditions during the deposition and/or the diffusion of the impurities. These result in diverging values for sheet resistances, and transistor parameters such as the threshold voltage.

2. Variations in the dimensions of the devices, mainly resulting from the limited resolution of the photolithographic process. This causes deviations in the (W/L) ratios of MOS transistors and the widths of interconnect wires.

Observe that quite a number of these deviations are totally uncorrelated. For instance, variations in the length of an MOS transistor are unrelated to variations in the threshold voltage as both are set by different process steps. Below we examine the impact on some of the parameters that determine the transistor current.

- The *threshold voltage* V_T can vary for numerous reasons: changes in oxide thickness, substrate, poly and implant impurity levels, and the surface charge. Accurate control of the threshold voltage is an important goal for many reasons. Where in the past thresholds could vary by as much as 50%, state-of-the-art digital processes manage to control the thresholds to within 25-50 mV.
- k_n': The main cause for variations in the process transconductance is changes in oxide thickness. Variations can also occur in the mobility but to a lesser degree.
- Variations in W and L. These are mainly caused by the lithographic process. Observe that variations in W and L are totally uncorrelated since the first is determined in the field-oxide step, while the second is defined by the polysilicon definition and the source and drain diffusion processes.

The measurable impact of the process variations may be a substantial deviation of the circuit behavior from the nominal or expected response, and this could be in either positive or negative directions. This poses the designer for an important economic dilemma. Assume, for instance, that you are supposed to design a microprocessor running at a clock frequency of 500 MHz. It is economically important that the majority of the manufactured dies meet that performance requirement. One way to achieve that goal is to design the circuit assuming worst-case values for all possible device parameters. While safe, this approach is prohibitively conservative and results in severely overdesigned and hence uneconomical circuits.

To help the designer make a decision on how much margin to provide, the device manufacturer commonly provides fast and slow device models in addition to the nominal ones. These result in larger or smaller currents than expected, respectively.

Example 3.12 MOS Transistor Process Variations

To illustrate the possible impact of process variations on the performance of an MOS device, consider a minimum-size NMOS device in our generic 0.25 μm CMOS process. A later chapter will establish that the speed of the device is proportional to the drain current that can be delivered.

Assume initially that $V_{GS} = V_{DS} = 2.5$ V. From earlier simulations, we know that this produces a drain current of 220 μA. The nominal model is now replaced by the fast and slow models, that modify the length and width (±10%), threshold (±60 mV), and oxide thickness (±5%) parameters of the device. Simulations produce the following data:

Fast: $I_d = 265$ μA: +20%

Slow: $I_d = 182$ μA: -17%

Let us now proceed one step further. The supply voltage delivered to a circuit is by no means a constant either. For instance, the voltage delivered by a battery can drop off substantially

towards the end of its lifetime. In practice, a variation in 10% of the supply voltage may well be expected.

Fast + V_{dd} = 2.75 V: I_d = 302 μA: +37%

Slow + V_{dd} = 2.25 V: I_d = 155 μA: -30%

The current levels and the associated circuit performance can thus vary by almost 100% between the extreme cases. To guarantee that the fabricated circuits meet the performance requirements under all circumstances, it is necessary to make the transistor 42% (=220μA/155μA) wider then would be required in the nominal case. This translates into a severe area penalty.

Fortunately, these worst- (or best-) case conditions occur only very rarely in reality. The probability that all parameters assume their worst-case values simultaneously is very low, and most designs will display a performance centered around the nominal design. The art of the *design for manufacturability* is to center the nominal design so that the majority of the fabricated circuits (e.g., 98%) will meet the performance specifications, while keeping the area overhead minimal.

Specialized design tools to help meet this goal are available. For instance, the Monte Carlo analysis approach [Jensen91] simulates a circuit over a wide range of randomly chosen values for the device parameters. The result is a distribution plot of design parameters (such as the speed or the sensitivity to noise) that can help to determine if the nominal design is economically viable. Examples of such distribution plots, showing the impact of variations in the effective transistor channel length and the PMOS transistor thresholds on the speed of an adder cell, are shown in Figure 3.38. As can be observed, technology variations can have a substantial impact on the performance parameters of a design.

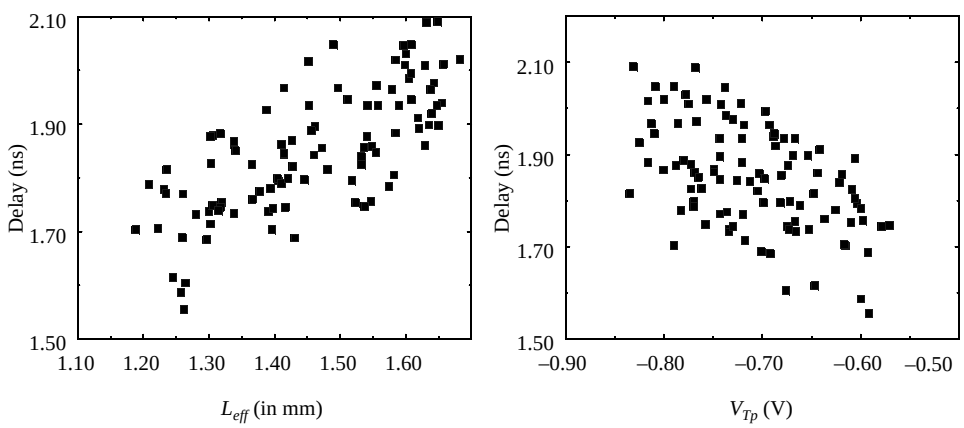


Figure 3.38 Distribution plots of speed of adder circuit as a function of varying device parameters, as obtained by a Monte Carlo analysis. The circuit is implemented in a 2 μm (nominal) CMOS technology (*courtesy of Eric Boskin, UCB, and ATMEL corp.*).

One important conclusion from the above discussion is that SPICE simulations should be treated with care. The device parameters presented in a model represent average values, measured over a batch of manufactured wafers. Actual implementations are bound to differ from the simulation results, and for reasons other than imperfections in the mod-

eling approach. Be furthermore aware that temperature variations on the die can present another source for parameter deviations. Optimizing an MOS circuit with SPICE to a resolution level of a picosecond or a microvolt is clearly a waste of effort.

3.5 Perspective: Technology Scaling

Over the last decades, we have observed a spectacular increase in integration density and computational complexity of digital integrated circuits. As already argued in the introduction, applications that were considered implausible yesterday are already forgotten today. Underlying this revolution are the advances in device manufacturing technology that allow for a steady reduction of the minimum feature size such as the minimum transistor channel length realizable on a chip. To illustrate this point, we have plotted in Figure 3.39 the evolution of the (average) minimum device dimensions starting from the 1960s and projecting into the 21st century. We observe a reduction rate of approximately 13% per year, halving every 5 years. Another interesting observation is that no real sign of a slowdown is in sight, and that the breathtaking pace will continue in the foreseeable future.

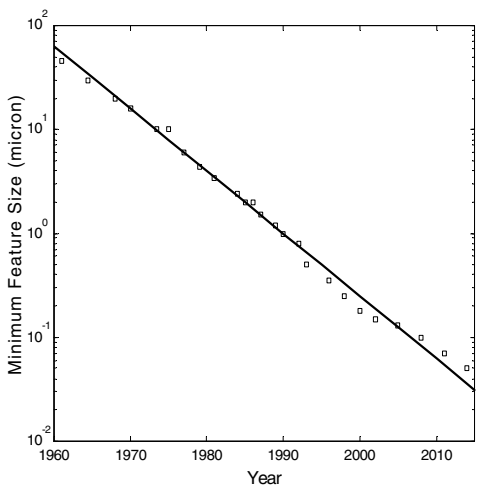


Figure 3.39 Evolution of (average) minimum channel length of MOS transistors over time. Dots represent observed or projected (2000 and beyond) values. The continuous line represents a scaling scenario that reduces the minimum feature with a factor 2 every 5 years.

A pertinent question is how this continued reduction in feature size influences the operating characteristics and properties of the MOS transistor, and indirectly the critical digital design metrics such as switching frequency and power dissipation. A first-order projection of this behavior is called a *scaling analysis*, and is the topic of this section. In addition to the minimum device dimension, we have to consider the supply voltage as a second independent variable in such a study. Different scaling scenarios result based on how these two independent variables are varied with respect to each other [Dennard74, Baccarani84].

Three different models are studied in Table 3.8. To make the results tractable, it is assumed that all device dimensions scale by the same factor S (with $S > 1$ for a reduction in size). This includes the width and length of the transistor, the oxide thickness, and the junction depths. Similarly, we assume that all voltages, including the supply voltage and

the threshold voltages, scale by a same ratio U. The relations governing the scaling behavior of the dependent variables are tabulated in column 2. Observe that this analysis only considers short-channel devices with a linear dependence between control voltage and saturation current (as expressed by Eq. (3.39)). We discuss each scenario in turn.

Table 3.8 Scaling scenarios for short-channel devices.

Parameter	Relation	Full Scaling	General Scaling	Fixed-Voltage Scaling
W, L, t_{ox}		$1/S$	$1/S$	$1/S$
V_{DD}, V_T		$1/S$	$1/U$	1
N_{SUB}	V/W_{depl}^2	S	S^2/U	S^2
Area/Device	WL	$1/S^2$	$1/S^2$	$1/S^2$
C_{ox}	$1/t_{ox}$	S	S	S
C_{gate}	$C_{ox}WL$	$1/S$	$1/S$	$1/S$
k_n, k_p	$C_{ox}W/L$	S	S	S
I_{sat}	$C_{ox}WV$	$1/S$	$1/U$	1
Current Density	$I_{sat}/Area$	S	S^2/U	S^2
Ron	V/I_{sat}	1	1	1
Intrinsic Delay	$R_{on}C_{gate}$	$1/S$	$1/S$	$1/S$
P	$I_{sat}V$	$1/S^2$	$1/U^2$	1
Power Density	$P/Area$	1	S^2/U^2	S^2

Full Scaling (Constant Electrical Field Scaling)

In this ideal model, voltages and dimensions are scaled by the same factor S. The goal is to keep the electrical field patterns in the scaled device identical to those in the original device. Keeping the electrical fields constant ensures the physical integrity of the device and avoids breakdown or other secondary effects. This scaling leads to greater device density (*Area*), higher performance (*Intrinsic Delay*), and reduced power consumption (P). The effects of full scaling on the device and circuit parameters are summarized in the third column of Table 3.8. We use the intrinsic time constant, which is the product of the gate capacitance and the on-resistance, as a measure for the performance. The analysis shows that the on-resistance remains constant due to the simultaneous scaling of voltage swing and current level. The performance improved is solely due to the reduced capacitance. The results clearly demonstrate the beneficial effects of scaling—the speed of the circuit increases in a linear fashion, while the power/gate scales down quadratically![6]

[6] Some assumptions were made when deriving this table:

1. It is assumed that the carrier mobilities are not affected by the scaling.
2. The substrate doping N_{sub} is scaled so that the maximum depletion-layer width is reduced by a factor S.
3. It is furthermore assumed that the delay of the device is mainly determined by the intrinsic capacitance (the gate capacitance) and that other device capacitances, such as the diffusion capacitances, scale appropriately. This assumption is approximately true for the full-scaling case, but not for fixed-voltage scaling, where C_{diff} scales as $1/\sqrt{S}$.

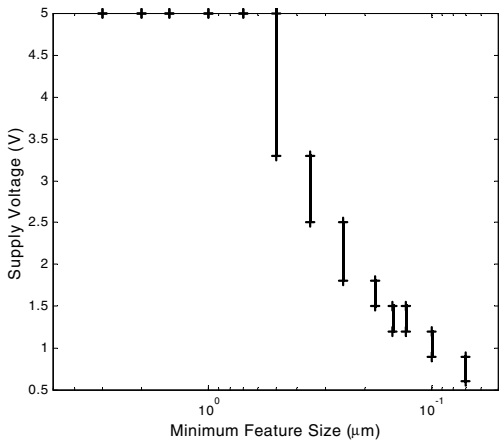


Figure 3.40 Evolution of min and max supply-voltage in digital integrated circuits as a function of feature size. All values for 0.15 micron and below are projected.

Fixed-Voltage Scaling

In reality, full scaling is not a feasible option. First of all, to keep new devices compatible with existing components, voltages cannot be scaled arbitrarily. Having to provide for multiple supply voltages adds considerably to the cost of a system. As a result, voltages have not been scaled down along with feature sizes, and designers adhere to well-defined standards for supply voltages and signal levels. As is illustrated in Figure 3.40, 5 V was the de facto standard for all digital components up to the early 1990s, and a *fixed-voltage scaling model* was followed.

Only with the introduction of the 0.5 μm CMOS technology did new standards such as 3.3 V and 2.5 V make an inroad. Today, a closer tracking between voltage and device dimension can be observed.The reason for this change in operation model can partially be explained with the aid of the fixed-voltage scaling model, summarized in the fifth column of Table 3.8. In a velocity-saturated device, keeping the voltage constant while scaling the device dimensions does not give a performance advantage over the full-scaling model, but instead comes with a major power penalty. The gain of an increased current is simply offset by the higher voltage level, and only hurts the power dissipation. This scenario is very different from the situation that existed when transistors were operating in the long-channel mode, and the current was a quadratic function of the voltage (as per Eq. (3.29)). Keeping the voltage constant under these circumstances gives a distinct performance advantage, as it causes a net reduction in on-resistance.

While the above argumentation offers ample reason to scale the supply voltages with the technology, other physical phenomena such as the hot-carrier effect and oxide breakdown also contributed to making the fixed-voltage scaling model unsustainable.

Problem 3.2 Scaling of Long-channel Devices

Demonstrate that for a long-channel transistor, full-voltage scaling results in a reduction of the intrinsic delay with a factor S^2, while increasing the power dissipation/device by S.

Reconstruct Table 3.8 assuming that the current is a quadratic function of the voltage (Eq. (3.29)).

WARNING: The picture painted in the previous section represents a first-order model. Increasing the supply voltage still offers somewhat of a performance benefit for short-channel transistors. This is apparent in Figure 3.27 and Table 3.3, which show a reduction of the equivalent on-resistance with increasing supply voltage — even for the high voltage range. Yet, this effect, which is mostly due to the channel-length modulation, is secondary and is far smaller than what would be obtained in case of long-channel devices.

The reader should keep this warning in the back of his mind throughout this scaling study. The goal is to discover first-order trends. This implies ignoring second-order effects such as mobility-degradation, series resistance, etc.

General Scaling

We observe in Figure 3.40 that the supply voltages, while moving downwards, are not scaling as fast as the technology. For instance, for the technology scaling from 0.5 μm to 0.1 μm, the maximum supply-voltage only reduces from 5 V to 1.5 V. The obvious question is why not to stick to the full-scaling model, when keeping the voltage higher does not yield any convincing benefits? This departure is motivated by the following argumentation:

- Some of the intrinsic device voltages such as the silicon bandgap and the built-in junction potential, are material parameters and cannot be scaled.
- The scaling potential of the transistor threshold voltage is limited. Making the threshold too low makes it difficult to turn off the device completely. This is aggravated by the large process variation of the value of the threshold, even on the same wafer.

Therefore, a more general scaling model is needed, where dimensions and voltages are scaled independently. This general scaling model is shown in the fourth column of Table 3.8. Here, device dimensions are scaled by a factor S, while voltages are reduced by a factor U. When the voltage is held constant, $U = 1$, and the scaling model reduces to the fixed-voltage model. Note that the general-scaling model offers a performance scenario identical to the full- and the fixed scaling, while its power dissipation lies between the two models (for $S > U > 1$).

Verifying the Model

To summarize this discussion on scaling, we have combined in Table 3.9 the characteristics of some of the most recent CMOS processes and projections on some future ones. Observe how the operating voltages are being continuously reduced with diminishing device dimensions. As predicted by the scaling model, the maximum drive current remains approximately constant. Maintaining this level of drive in the presence of a

reduced supply voltage requires an aggressive lowering of the threshold voltage, which translates in a rapid increase of the sub-threshold leakage current.

Table 3.9 MOSFET technology projection for high performance logic (from [SIA01]).

Year of Introduction	2001	2003	2005	2007	2010	2013	2016
Drawn channel length (nm)	90	65	45	35	25	18	13
Physical channel length (nm)	65	45	32	25	18	13	9
Gate oxide (nm)	2.3	2.0	1.9	1.4	1.2	1.0	0.9
V_{DD} (V)	1.2	1.0	0.9	0.7	0.6	0.5	0.4
NMOS I_{Dsat} (μA/μm)	900	900	900	900	1200	1500	1500
NMOS I_{leak} (μA/μm)	0.01	0.07	0.3	1	3	7	10

From the above, it is reasonable to conclude that both integration density and performance will continue to increase. The obvious question is for how long? Experimental 25 nm CMOS devices have proven to be operational in the laboratories and to display current characteristics that are surprisingly close to present-day transistors. These transistors, while working along similar concepts as the current MOS devices, look very different from the structures we are familiar with, and require some substantial *device engineering*. For instance, Figure 3.41 shows a potential transistor structure, the Berkeley FinFET dual-gated transistor, which has proven to be operational up to very small channel lengths.

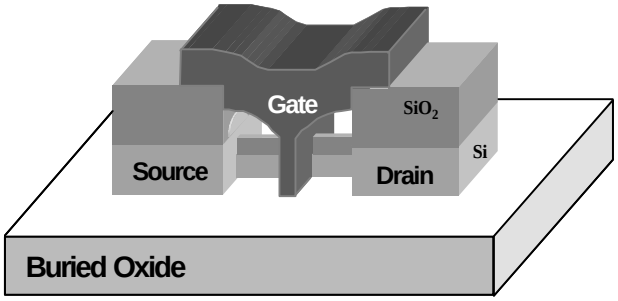


Figure 3.41 FinFET dual-gated transistor with 25 nm channel length [Hu99].

Another option is the *vertical transistor*. Even while the addition of many metal layers has turned the integrated circuit into a truly three-dimensional artifact, the transistor itself is still mostly laid out in a horizontal plane. This forces the device designer to jointly optimize packing density and performance parameters. By rotating the device so that the drain ends up on top, and the source at the bottom, these concerns are separated: packing density still is dominated by horizontal dimensions, while performance issues are mostly determined by vertical spacings (Figure 3.42). Operational devices of this type have been fabricated with channel lengths substantially below 0.1 μm. [Eaglesham00].

Integrated circuits integrating more then one billion transistors clocked at speeds of tens of GHz's hence seem to be well under way. Whether this will actually happen is an open question. Even though it might be technologically feasible, other parameters have an equal impact on the feasibility of such an undertaking. A first doubt is if such a part can be

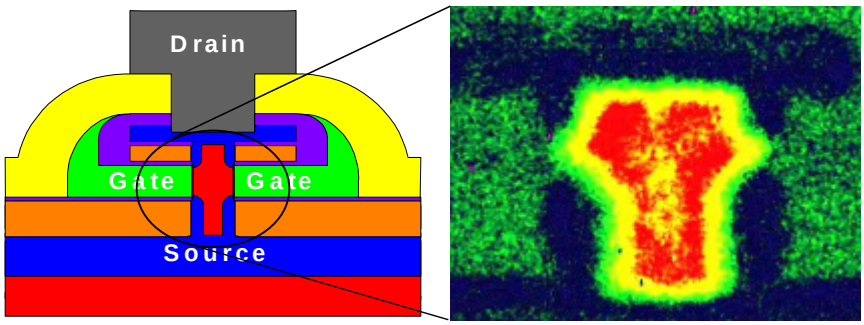


Figure 3.42 Vertical transistor with dual gates. The photo on the right shows an enlarged view of the channel area.

manufactured in an economical way. Current semiconductor plants cost over $5 billion, and this price is expected to rise substantially with smaller feature sizes. Design considerations also play a role. Power consumption of such a component might be prohibitive. The growing role of interconnect parasitics might put an upper bound on performance. Finally, system considerations might determine what level of integration is really desirable. All in all, it is obvious that the design of semiconductor circuits still faces an exciting future.

3.6 Summary

In this chapter, we have presented a a comprehensive overview of the operation of the MOSFET transistor, the semiconductor device at the core of virtually all contemporary digital integrated circuits. Besides an intuitive understanding of its behavior, we have presented a variety of modeling approaches ranging from simple models, useful for a first-order manual analysis of the circuit operation, to complex SPICE models. These models will be used extensively in later chapters, where we look at the fundamental building blocks of digital circuits. We started off with a short discussion of the semiconductor diode, one of the most dominant parasitic circuit elements in CMOS designs.

- The static behavior of the junction diode is well described by the ideal diode equation that states that the current is an *exponential function of the applied voltage bias*.
- In reverse-biased mode, the depletion-region space charge of the diode can be modeled as a non-linear voltage-dependent capacitance. This is particularly important as the omnipresent source-body and drain-body junctions of the MOS transistors all operate in this mode. A linearized large-scale model for the depletion capacitance was introduced for manual analysis.
- The MOS(FET) transistor is a voltage-controlled device, where the controlling gate terminal is insulated from the conducting channel by a SiO_2 capacitor. Based on the value of the gate-source voltage with respect to a threshold voltage V_T, three operation regions have been identified: *cut-off*, *linear*, and *saturation*. One of the most enticing properties of the MOS transistor, which makes it particularly amenable to digital design, is that it approximates a voltage-controlled switch: when the control voltage is low, the switch is nonconducting (open); for a high control voltage, a con-

ducting channel is formed, and the switch can be considered closed. This two-state operation matches the concepts of binary digital logic.

- The continuing reduction of the device dimensions to the submicron range has introduced some substantial deviations from the traditional long-channel MOS transistor model. The most important one is the *velocity saturation* effect, which changes the dependence of the transistor current with respect to the controlling voltage from *quadratic to linear*. Models for this effect as well as other second-order parasitics have been introduced. One particular effect that is gaining in importance is the *subthreshold conduction*, which causes devices to conduct current even when the control voltage drops below the threshold.
- The dynamic operation of the MOS transistor is dominated by the *device capacitors*. The main contributors are the gate capacitance and the capacitance formed by the depletion regions of the source and drain junctions. The minimization of these capacitances is the prime requirement in high-performance MOS design.
- SPICE models and their parameters have been introduced for all devices. It was observed that these models represent an average behavior and can vary over a single wafer or die.
- The MOS transistor is expected to dominate the digital integrated circuit scene for the next decade. Continued scaling will lead to device sizes of approximately 0.07 micron by the year 2010, and logic circuits integrating more than 1 billion transistors on a die.

3.7 To Probe Further

Semiconductor devices have been discussed in numerous books, reprint volumes, tutorials, and journal articles. The *IEEE Journal on Electron Devices* is the premier journal, where most of the state-of-the-art devices and their modeling are discussed. Another valuable resource are the proceedings of the International Electron Devices Meeting (IEDM). The books (such as [Streetman95] and [Pierret96]) and journal articles referenced below contain excellent discussions of the semiconductor devices of interest or refer to specific topics brought up in the course of this chapter.

REFERENCES

[Baccarani84] G. Baccarani, M. Wordeman, and R. Dennard, “Generalized Scaling Theory and Its Application to 1/4 Micrometer MOSFET Design,” *IEEE Trans. Electron Devices*, ED-31(4): p. 452, 1984.

[Banzhaf92] W. Bhanzhaf, *Computer Aided Analysis Using PSPICE*, 2nd ed., Prentice Hall, 1992.

[Dennard74] R. Dennard et al., “Design of Ion-Implanted MOSFETS with Very Small Physical Dimensions,” *IEEE Journal of Solid-State Circuits*, SC-9, pp. 256–258, 1974.

[Eaglesham99] D. Eaglesham, "*0.18 mm CMOS and Beyond*,", Proceedings 1999 Design Automation Conference, pp. 703-708, June 1999.

[Howe97] R. Howe and S. Sodini, *Microelectronics: An Integrated Approach*, Prentice Hall, 1997.

[Hu92] C. Hu, "IC Reliability Simulation," *IEEE Journal of Solid State Circuits*, vol. 27, no. 3, pp. 241–246, March 1992.

[Jensen91] G. Jensen et al., "Monte Carlo Simulation of Semiconductor Devices," *Computer Physics Communications*, 67, pp. 1–61, August 1991.

[Ko89] P. Ko, "Approaches to Scaling," in *VLSI Electronics: Microstructure Science*, vol. 18, chapter 1, pp. 1–37, Academic Press, 1989.

[Nagel75] L. Nagel, "SPICE2: a Computer Program to Simulate Semiconductor Circuits," Memo ERL-M520, Dept. Elect. and Computer Science, University of California at Berkeley, 1975.

[Pierret96] R. Pierret, *Semiconductor Device Fundamentals*, Addison-Wesley, 1996.

[SIA01] *International Technology Roadmap for Semiconductors*, http://www.sematech.org, 2001.

[Streetman95] B. Streetman, Solid State Electronic Devices, Prentice Hall, 1995.

[Thorpe92] T. Thorpe, *Computerized Circuit Analysis with SPICE*, John Wiley and Sons, 1992.

[Vladimirescu93] A. Vladimirescu, *The SPICE Book*, John Wiley and Sons, 1993.

[Weste93] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, 1993.

Chapter 3
PROBLEMS

For all problems, use the device parameters provided in Chapter 3 (Tables 3.2 and 3.5) and the inside back book cover, unless otherwise mentioned. Also assume T = 300 K by default.

1. [E,SPICE,3.2.2]

a. Consider the circuit of Figure 0.1. Using the simple model, with $V_{Don} = 0.7$ V, solve for I_D.

b. Find I_D and V_D using the ideal diode equation. Use $I_s = 10^{-14}$ A and $T = 300$ K.

c. Solve for V_{D1}, V_{D2}, and I_D using SPICE.

d. Repeat parts *b* and *c* using $I_S = 10^{-16}$ A, $T = 300$K, and $I_S = 10^{-14}$A, $T = 350$ K.

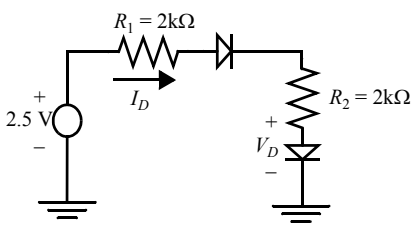


Figure 0.1 Resistor diode circuit.

2. [M, None, 3.2.3] For the circuit in Figure 0.2, $V_s = 3.3$ V. Assume $A_D = 12$ μm^2, $\phi_0 = 0.65$ V, and $m = 0.5$. $N_A = 2.5$ E16 and $N_D = 5$ E15.

a. Find I_D and V_D.

b. Is the diode forward- or reverse-biased?

c. Find the depletion region width, W_j, of the diode.

d. Use the parallel-plate model to find the junction capacitance, C_j.

e. Set $V_s = 1.5$ V. Again using the parallel-plate model, explain qualitatively why C_j increases.

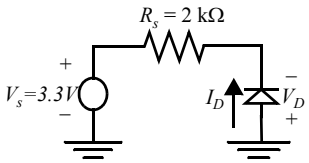


Figure 0.2 Series diode circuit

3. [E, None, 3.3.2] Figure 0.3 shows NMOS and PMOS devices with drains, source, and gate ports annotated. Determine the mode of operation (saturation, linear, or cutoff) and drain current I_D for each of the biasing configurations given below. Verify with SPICE. Use the following transistor data: NMOS: $k_n' = 115$μA/V^2, $V_{T0} = 0.43$ V, $\lambda = 0.06$ V^{-1}, PMOS: $k_p' = 30$ μA/V^2, $V_{T0} = -0.4$ V, $\lambda = -0.1$ V^{-1}. Assume $(W/L) = 1$.

a. NMOS: $V_{GS} = 2.5$ V, $V_{DS} = 2.5$ V. PMOS: $V_{GS} = -0.5$ V, $V_{DS} = -1.25$ V.

b. NMOS: $V_{GS} = 3.3$ V, $V_{DS} = 2.2$ V. PMOS: $V_{GS} = -2.5$ V, $V_{DS} = -1.8$ V.

c. NMOS: $V_{GS} = 0.6$ V, $V_{DS} = 0.1$ V. PMOS: $V_{GS} = -2.5$ V, $V_{DS} = -0.7$ V.

4. [E, SPICE, 3.3.2] Using SPICE plot the *I-V* characteristics for the following devices.

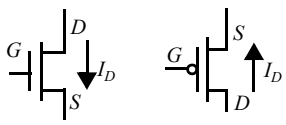


Figure 0.3 NMOS and PMOS devices.

a. NMOS $W = 1.2\mu m$, $L = 0.25\mu m$

b. NMOS $W = 4.8\mu m$, $L = 0.5\mu m$

c. PMOS $W = 1.2\ \mu m$, $L = 0.25\ \mu m$

d. PMOS $W = 4.8\ \mu m$, $L = 0.5\ \mu m$

5. [E, SPICE, 3.3.2] Indicate on the plots from problem 4.

a. the regions of operation.

b. the effects of channel length modulation.

c. Which of the devices are in velocity saturation? Explain how this can be observed on the *I-V* plots.

6. [M, None, 3.3.2] Given the data in Table 0.1 for a short channel NMOS transistor with $V_{DSAT} = 0.6\ V$ and $k' = 100\ \mu A/V^2$, calculate V_{T0}, γ, λ, $2|\phi_f|$, and W / L:

Table 0.1 Measured NMOS transistor data

	V_{GS}	V_{DS}	V_{BS}	I_D (μA)
1	2.5	1.8	0	1812
2	2	1.8	0	1297
3	2	2.5	0	1361
4	2	1.8	–1	1146
5	2	1.8	-2	1039

7. [E, None, 3.3.2] Given Table 0.2 ,the goal is to derive the important device parameters from these data points. As the measured transistor is processed in a deep-submciron technology, the '**unified model**' holds. From the material constants, we also could determine that the saturation voltage V_{DSAT} equals -1V. You may also assume that $-2\Phi_F = -0.6V$.

NOTE: The parameter values on Table 3.3 do NOT hold for this problem.

a. Is the measured transistor a PMOS or an NMOS device? Explain your answer.

b. Determine the value of V_{T0}.

c. Determine γ.

d. Determine λ.

e. Given the obtained answers, determine for each of the measurements the operation region of the transistor (choose from *cutoff, resistive, saturated, and velocity saturated*). Annotate your finding in the right-most column of the above.

Table 0.2 Measurements taken from the MOS device, at different terminal voltages.

Measurement number	VGS (V)	VDS (V)	VSB (V)	ID (μA)	Operation Region?
1	-2.5	-2.5	0	-84.375	
2	1	1	0	0.0	
3	-0.7	-0.8	0	-1.04	
4	-2.0	-2.5	0	-56.25	
5	-2.5	-2.5	-1	-72.0	
6	-2.5	-1.5	0	-80.625	
7	-2.5	-0.8	0	-66.56	

8. [M, None, 3.3.2] An NMOS device is plugged into the test configuration shown below in Figure 0.4. The input V_{in} =2V. The current source draws a constant current of 50 μA. *R* is a variable resistor that can assume values between 10kΩ and 30 kΩ. Transistor M1 experiences short channel effects and has following transistor parameters: $k' = 110*10^{-6}$ V/A^2, $V_T = 0.4$, and V_{DSAT} = 0.6V. The transistor has a W/L = 2.5μ/0.25μ. For simplicity body effect and channel length modulation can be neglected. i.e λ=0, γ=0. .

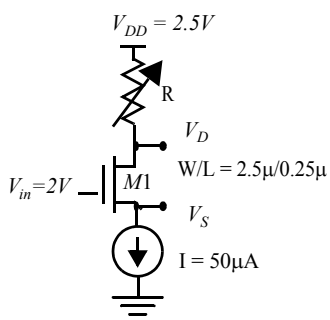


Figure 0.4 Test configuration for the NMOS device.

a. When *R* =10kΩ find the operation region, V_D and V_S.

b. When R= 30kΩ again determine the operation region V_D, V_S

c. For the case of R = 10kΩ, would V_S increase or decrease if $\lambda \neq 0$. Explain qualitatively

9. [M, None, 3.3.2] Consider the circuit configuration of Figure 0.5.

a. Write down the equations (and only those) which are needed to determine the voltage at node X. Do NOT plug in any values yet. Neglect short channel effects and assume that $\lambda_p = 0$.

b. Draw the (approximative) load lines for both MOS transistor and resistor. Mark some of the significant points.

c. Determine the required width of the transistor (for $L = 0.25\mu m$) such that X equals 1.5 V.

d. We have, so far, assumed that M_1 is a long-channel device. Redraw the load lines assuming that M_1 is velocity-saturated. Will the voltage at X rise or fall?

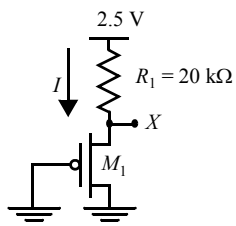


Figure 0.5 MOS circuit.

10. [M, None, 3.3.2] The circuit of Figure 0.6 is known as a *source-follower* configuration. It achieves a DC level shift between the input and output. The value of this shift is determined by the current I_0. Assume $\gamma = 0.4$, $2|\phi_f| = 0.6$ V, $V_{T0} = 0.43$ V, $k' = 115\ \mu A/V^2$, and $\lambda = 0$. The NMOS device has W/L = 5.4μ/1.2μ such that the short channel effects are not observed.

a. Derive an expression giving V_i as a function of V_o and $V_T(V_o)$. If we neglect body effect, what is the nominal value of the level shift performed by this circuit.

b. The NMOS transistor experiences a shift in V_T due to the body effect. Find V_T as a function of V_o for V_o ranging from 0 to 1.5V with 0.25 V intervals. Plot V_T vs. V_o.

c. Plot V_o vs. V_i as V_o varies from 0 to 1.5 V with 0.25 V intervals. Plot two curves: one neglecting the body effect and one accounting for it. How does the body effect influence the operation of the level converter? At V_o (body effect) = 1.5 V, find V_o (ideal) and, thus, determine the maximum error introduced by body effect.

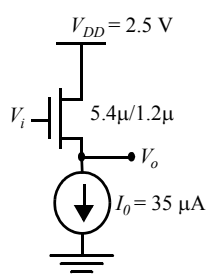


Figure 0.6 Source-follower level converter.

11. [M, SPICE, 3.3.2] Problem 11 uses the MOS circuit of Figure 0.7.

a. Plot V_{out} vs. V_{in} with V_{in} varying from 0 to 2.5 volts (use steps of 0.5V). $V_{DD} = 2.5$ V.

b. Repeat *a* using SPICE.

c. Repeat *a* and *b* using a MOS transistor with $(W / L) = 4 / 1$. Is the discrepancy between manual and computer analysis larger or smaller. Explain why.

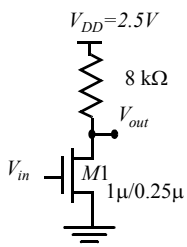


Figure 0.7 MOS circuit.

12. [E, None, 3.3.2]Below in Figure 0.8 is an I-V transfer curve for an NMOS transistor. In this problem, the objective is to use this I-V curve to obtain information about the transistor. The transistor has (W/L)=(1μ/1μ). It may also be assumed that velocity saturation does not play a role in this example. Also assume $-2\Phi_F$ = 0.6V. Using Figure 0.8 determine the following parameters: device V_{TO}, γ, λ.

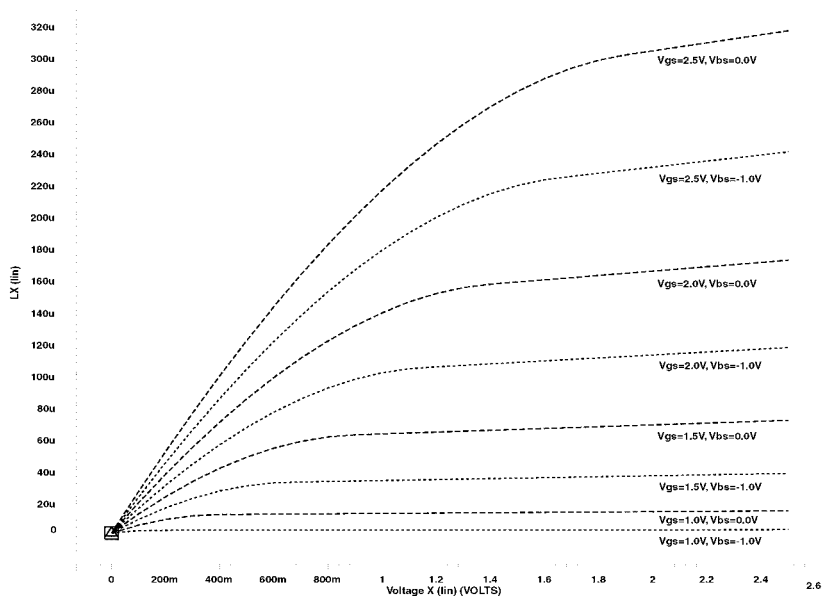


Figure 0.8 I-V curves

13. [E, None , 3.3.2]The curves below in Figure 0.9 represent the gate voltage(V_{GS}) vs. drain current (I_{DS}) of two NMOS devices which are on the same die and operate in subthreshold region. Due to process variations on the same die the curves do not overlap.

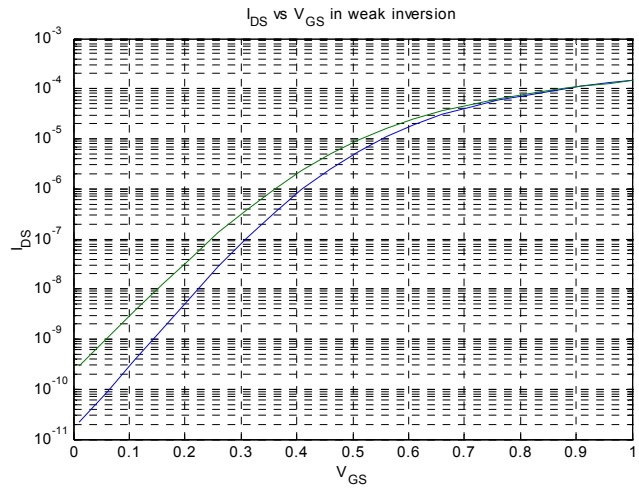


Figure 0.9 Subthreshold current curves. Difference is due to process variations

Also assume that the transistors are within the same circuit configurations as Figure 0.10 in If the in put voltages are both V_{in} = 0.2V. What would be the respective durations to discharge the load of C_L = 1pF attached to the drains of these devices.

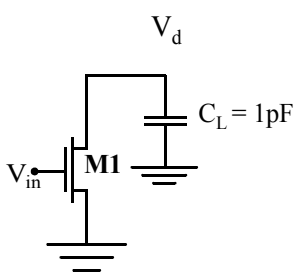


Figure 0.10 The circuit for testing the time to discharge the load capacitance through a device operating in subthreshold region.

14. [M, None, 3.3.2] Short-channel effects:

a. Use the fact that current can be expressed as the product of the carrier charge per unit length and the velocity of carriers (I_{DS} = Qv) to derive I_{DS} as a function of W, C_{ox}, $V_{GS} - V_T$, and carrier velocity v.

b. For a long-channel device, the carrier velocity is the mobility times the applied electric field. The electrical field, which has dimensions of V/m, is simply $(V_{GS} - V_T) / 2L$. Derive I_{DS} for a long-channel device.

c. From the equation derived in *a*, find I_{DS} for a short-channel device in terms of the maximum carrier velocity, v_{max}.

Based on the results of *b* and *c* describe the most important differences between short-channel and long-channel devices.

15. [C, None, 3.3.2] Another equation, which models the velocity-saturated drain current of an MOS transistor is given by

$$I_{dsat} = \frac{1}{1 + (V_{GS} - V_t)/(E_{sat}L)}\left(\frac{\mu_0 C_{ox}}{2}\right)\frac{W}{L}(V_{GS} - V_T)^2$$

Using this equation it is possible to see that velocity saturation can be modeled by a MOS device with a source-degeneration resistor (see Figure 0.11).

a. Find the value of R_S such that $I_{DSAT}(V_{GS}, V_{DS})$ for the composite transistor in the figure matches the above velocity-saturated drain current equation. *Hint: the voltage drop across R_S is typically small.*

b. Given $E_{sat} = 1.5$ V/μm and $k' = \mu_0 C_{ox} = 20$ μA/V^2, what value of R_S is required to model velocity saturation. How does this value depend on W and L?

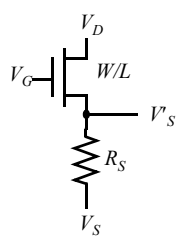


Figure 0.11 Source-degeneration model of velocity saturation.

16. [E, None, 3.3.2] The equivalent resistances of two different devices need to be computed.

a. First, consider the fictive device whose I-V characteristic is given in Figure 0.12. Constant k has the dimension of S (or1/Ω). V_0 is a voltage characteristic to the device. Calculate the equivalent resistance for an output voltage transition from 0 to $2V_0$ by integrating the resistance as a function of the voltage.

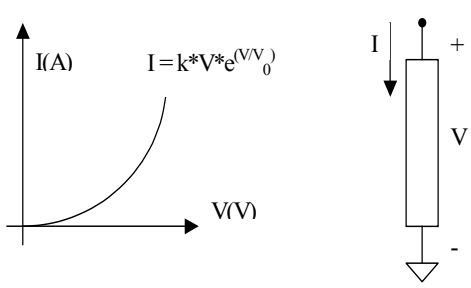


Figure 0.12 Fictive device whose equivalent resistance is to be calculated.

b. Next, obtain the resistance equation 3.43 using the Figure 0.13. Assuming the V_{GS} is kept at V_{DD}, Calculate the Req as output (V_{DS}) transitions from V_{DD} to $V_{DD}/2$.(Figure 0.13).

Hint: Make sure you use the Short channel Unified MOS Model equations. **Hint:** You will need to use the expansion. $ln(1+x) \approx x - x^2/2 + x^3/3$

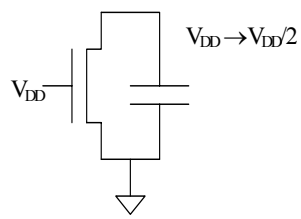


Figure 0.13 The equivalent resistance is to be computed for the H→L transition.

17. [M, None, 3.3.3] Compute the gate and diffusion capacitances for transistor *M*1 of Figure 0.7. Assume that drain and source areas are rectangular, and are 1 μm wide and 0.5 μm long. Use the parameters of Example 3.5 to determine the capacitance values. Assume m_j = 0.5 and m_{jsw} = 0.44. Also compute the total charge stored at node *In*, for the following initial conditions:

a. V_{in} = 2.5 V, V_{out} = 2.5 V, 0.5 V, and 0 V.

b. V_{in} = 0 V, V_{out} = 2.5 V, 0.5 V, and 0 V.

18. [E, None, 3.3.3]Consider a CMOS process with the following capacitive parameters for the NMOS transistor: C_{GSO}, C_{GDO}, C_{OX}, C_J, m_j, C_{jsw}, m_{jsw}, and PB, with the lateral diffusion equal to L_D. The MOS transistor M1 is characterized by the following parameters: W, L, AD, PD, AS, PS.

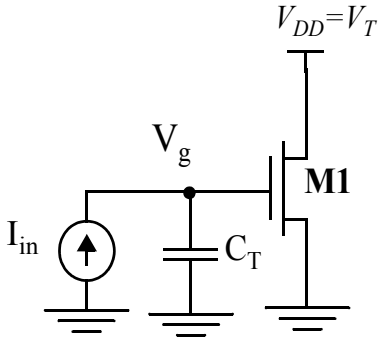


Figure 0.14 Circuit to measure total input capacitance

a. Consider the configuration of Figure 0.14. V_{DD} is equal to V_T (the threshold voltage of the transistor) Assume that the initial value of V_g equals 0. A current source with value I_{in} is applied at time 0. Assuming that all the capacitance at the gate node can be lumped into a single, grounded, linear capacitance C_T, derive an expression for the time it will take for V_g to reach 2 V_T

b. The obvious question is now how to compute C_T. Among, C_{db}, C_{sb}, C_{gs}, C_{gd}, C_{gb} which of these parasatic capacitances of the MOS transistor contribute to C_T. For those that contribute to C_T write down the expression that determines the value of the contribution. Use only the parameters given above. If the transistor goes through different operation regions and this impacts the value of the capacitor, **determine the expression of the contribution for each region (and indicate the region)**.

c. Consider now the case depicted in Figure 0.15. Asssume that V_d is initially at 0 and we want to charge it up to 2 V_T. Again among, C_{db}, C_{sb}, C_{gs}, C_{gd}, C_{gb} which device capacitances contribute to the total drain capacitance? Once again, make sure you differentiate between different operation regions..

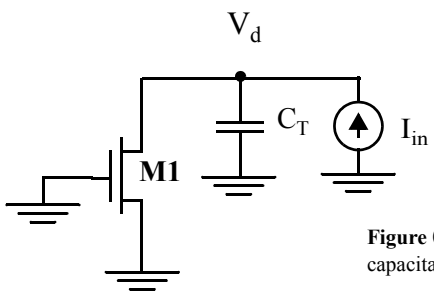


Figure 0.15 Circuit to measure the total drain capacitance

19. [M, None, 3.3.3]For the NMOS transistor in Figure 0.16, **sketch the voltages** at the source and at the drain as a function of time. Initially, both source and drain are at +2.5 volts. Note that the drain is open circuited. The 10 μA current source is turned on at t=0. Device parameters:W/L_{eff}= 125μ/0.25μ ; μC_{ox} = 100 μA/V^2 ; C_{ox} = 6fF/μ^2 ;C_{OL}(per width) = 0.3 fF/μ ; C_{sb} = 100 fF; Cdb = 100fF;V_{DSAT} =1V.

HINT: Do not try to solve this analytically. Just use a qualitative analysis to derive the different operation modes of the circuit and the devices**.**

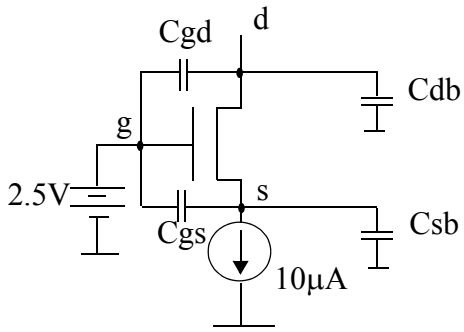


Figure 0.16 Device going through different operation regions over time

20. [C, SPICE, 3.4] Though impossible to quantify exactly by hand, it is a good idea to understand process variations and be able to at least get a rough estimate for the limits of their effects.

a. For the circuit of Figure 0.7, calculate nominal, minimum, and maximum currents in the NMOS device with V_{in} = 0 V, 2.5 V and 5 V. Assume 3σ variations in V_{T0} of 25 mV, in k' of 15%, and in lithographic etching of 0.15 μm.

b. Analyze the impact of these current variations on the output voltage. Assume that the load resistor also can vary by 10%. Verify the result with SPICE.

21. [E, None, 3.5] A state-of-the-art, synthesizable, embedded microprocessor consumes 0.4mW/MHz when fabricated using a 0.18 μm process. With typical standard cells (gates), the area of the processor is 0.7 mm2. Assuming a 100 Mhz clock frequency, and 1.8 V power

supply. Assume short channel devices, but ignore second order effects like mobility degradation, series resistance, etc.

a. Using fixed voltage scaling and constant frequency, what will the area, power consumption, and power density of the same processor be, if scaled to 0.12 μm technology, assuming the same clock frequency?

b. If the supply voltage in the scaled 0.12 μm part is reduced to 1.5 V what will the power consumption and power density be?

c. How fast could the scaled processor in Part (b) be clocked? What would the power and power density be at this new clock frequency?

d. Power density is important for cooling the chip and packaging. What would the supply voltage have to be to maintain the same power density as the original processor?

22. The superscalar, superpipelined, out-of-order executing, highly parallel, fully x86 compatible JMRII microprocessor was fabricated in a 0.25 m technology and was able to operate at 100 MHZ, consuming 10 watts using a 2.5 V power supply.

a. Using fixed voltage scaling, what will the speed and power consumption of the same processor be if scaled to 0.1 μm technology?

b. If the supply voltage on the 0.1 μm part were scaled to 1.0 V, what will the power consumption and speed be?

c. What supply should be used to fix the power consumption at 10 watts? At what speed would the processor operate?

CHAPTER 4

THE WIRE

Determining and quantifying interconnect parameters

n

Introducing circuit models for interconnect wires

n

Detailed wire models for SPICE

n

Technology scaling and its impact on interconnect

4.1 Introduction

4.2 A First Glance

4.3 Interconnect Parameters — Capacitance, Resistance, and Inductance

- 4.3.1 Capacitance
- 4.3.2 Resistance
- 4.3.3 Inductance

4.4 Electrical Wire Models

- 4.4.1 The Ideal Wire
- 4.4.2 The Lumped Model
- 4.4.3 The Lumped RC model
- 4.4.4 The Distributed rc_{Line}
- 4.4.5 The Transmission Line

4.5 SPICE Wire Models

- 4.5.1 Distributed rc Lines in SPICE
- 4.5.2 Transmission Line Models in SPICE

4.6 Perspective: A Look into the Future

4.1 Introduction

Throughout most of the past history of integrated circuits, on-chip interconnect wires were considered to be second class citizens that had only to be considered in special cases or when performing high-precision analysis. With the introduction of deep-submicron semiconductor technologies, this picture is undergoing rapid changes. The parasitics effects introduced by the wires display a scaling behavior that differs from the active devices such as transistors, and tend to gain in importance as device dimensions are reduced and circuit speed is increased. In fact, they start to dominate some of the relevant metrics of digital integrated circuits such as speed, energy-consumption, and reliability. This situation is aggravated by the fact that improvements in technology make the production of ever-larger die sizes economically feasible, which results in an increase in the average length of an interconnect wire and in the associated parasitic effects. A careful and in-depth analysis of the role and the behavior of the interconnect wire in a semiconductor technology is therefore not only desirable, but even essential.

4.2 A First Glance

The designer of an electronic circuit has multiple choices in realizing the interconnections between the various devices that make up the circuit. State-of-the-art processes offer multiple layers of Aluminum, and at least one layer of polysilicon. Even the heavily doped n^+ or p^+ layers, typically used for the realization of source and drain regions, can be employed for wiring purposes. These wires appear in the schematic diagrams of electronic circuits as simple lines with no apparent impact on the circuit performance. In our discussion on the integrated-circuit manufacturing process, it became clear that this picture is overly simplistic, and that the wiring of today's integrated circuits forms a complex geometry that introduces capacitive, resistive, and inductive parasitics. All three have multiple effects on the circuit behavior.

1. An increase in propagation delay, or, equivalently, a drop in performance.
2. An impact on the energy dissipation and the power distribution.
3. An introduction of extra noise sources, which affects the reliability of the circuit.

A designer can decide to play it safe and include all these parasitic effects in her analysis and design optimization process. This conservative approach is non-constructive and even unfeasible. First of all, a "complete" model is dauntingly complex and is only applicable to very small topologies. It is hence totally useless for today's integrated circuits with their millions of circuit nodes. Furthermore, this approach has the disadvantage that the "forest gets lost between the trees". The circuit behavior at a given circuit node is only determined by a few dominant parameters. Bringing all possible effects to bear, only obscures the picture and turns the optimization and design process a "trial-and-error" operation rather than an enlightened and focused search.

To achieve the latter, it is important that the designer has a clear insight in the parasitic wiring effects, their relative importance, and their models. This is best illustrated with the simple example, shown in Figure 4.1. Each wire in a bus network connects a transmit-

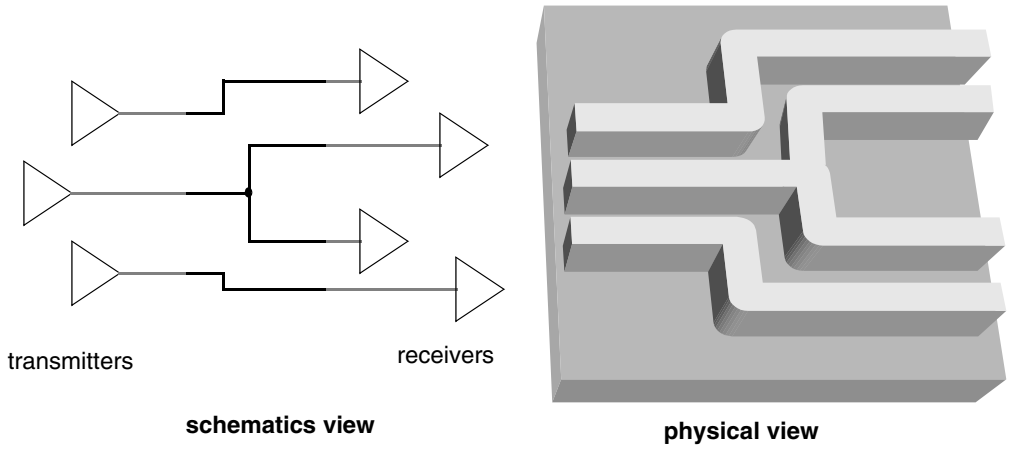


Figure 4.1 Schematic and physical views of wiring of bus-network. The latter shows only a limited area (as indicated by the shadings in the schematics).

ter (or transmitters) to a set of receivers and is implemented as a link of wire segments of various lengths and geometries. Assume that all segments are implemented on a single interconnect layer, isolated from the silicon substrate and from each other by a layer of dielectric material. Be aware that the reality may be far more complex.

A full-fledged circuit model, taking into account the parasitic capacitance, resistance, and the inductance of the interconnections, is shown in Figure 4.2a. Observe that these extra circuit elements are not located in a single physical point, but are distributed over the length of the wire. This is a necessity when the length of the wire becomes significantly larger than its width. Notice also that some parasitics are inter-wire, hence creating coupling effects between the different bus-signals that were not present in the original schematics.

Analyzing the behavior of this schematic, which only models a small part of the circuit, is slow and cumbersome. Fortunately, substantial simplifications can often be made, some of which are enumerated below.

- Inductive effects can be ignored if the resistance of the wire is substantial — this is for instance the case for long Aluminum wires with a small cross-section — or if the rise and fall times of the applied signals are slow.
- When the wires are short, the cross-section of the wire is large, or the interconnect material used has a low resistivity, a capacitance-only model can be used (Figure 4.2b).
- Finally, when the separation between neighboring wires is large, or when the wires only run together for a short distance, inter-wire capacitance can be ignored, and all the parasitic capacitance can be modeled as capacitance to ground.

Obviously, the latter problems are the easiest to model, analyze, and optimize. The experienced designer knows to differentiate between dominant and secondary effects. The goal of this chapter is to present the reader the basic techniques to estimate the values of

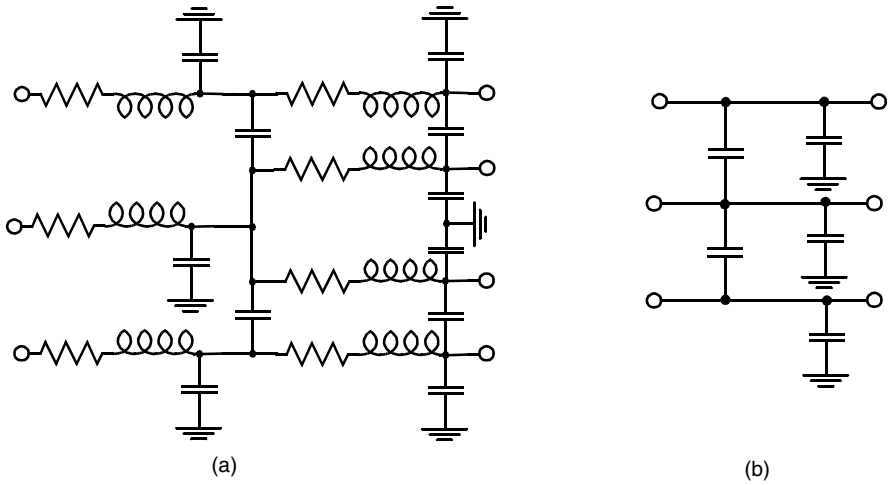


Figure 4.2 Wire models for the circuit of Figure 4.1. Model (a) considers most of the wire parasitics (with the exception of inter-wire resistance and mutual inductance), while model (b) only considers capacitance.

the various interconnect parameters, simple models to evaluate their impact, and a set of rules-of-thumb to decide when and where a particular model or effect should be considered.

4.3 Interconnect Parameters — Capacitance, Resistance, and Inductance

4.3.1 Capacitance

An accurate modeling of the wire capacitance(s) in a state-of-the-art integrated circuit is a non-trivial task and is even today the subject of advanced research. The task is complicated by the fact that the interconnect structure of contemporary integrated circuits is three-dimensional, as was clearly demonstrated in the process cross-section of Figure 2.8. The capacitance of such a wire is a function of its shape, its environment, its distance to the substrate, and the distance to surrounding wires. Rather than getting lost in complex equations and models, a designer typically will use an advanced extraction tool to get precise values of the interconnect capacitances of a completed layout. Most semiconductor manufacturers also provide empirical data for the various capacitance contributions, as measured from a number of test dies. Yet, some simple first-order models come in handy to provide a basic understanding of the nature of interconnect capacitance and its parameters, and of how wire capacitance will evolve with future technologies.

Consider first a simple rectangular wire placed above the semiconductor substrate, as shown in Figure 4.3. If the width of the wire is substantially larger than the thickness of the insulating material, it may be assumed that the electrical-field lines are orthogonal to the capacitor plates, and that its capacitance can be modeled by the *parallel-plate capaci-*

tor model (also called *area capacitance*). Under those circumstances, the total capacitance of the wire can be approximated as[1]

$$c_{int} = \frac{\varepsilon_{di}}{t_{di}} WL \tag{4.1}$$

where W and L are respectively the width and length of the wire, and t_{di} and ε_{di} represent the thickness of the dielectric layer and its permittivity. SiO_2 is the dielectric material of choice in integrated circuits, although some materials with lower permittivity, and hence lower capacitance, are coming in use. Examples of the latter are organic polyimides and aerogels. ε is typically expressed as the product of two terms, or $\varepsilon = \varepsilon_r \varepsilon_0$. $\varepsilon_0 = 8.854 \times 10^{-12}$ F/m is the permittivity of free space, and ε_r the relative permittivity of the insulating material. Table 4.1 presents the relative permittivity of several dielectrics used in integrated circuits. In summary, the important message from Eq. (4.1) is that the capacitance is proportional to the overlap between the conductors and inversely proportional to their separation.

Table 4.1 Relative permittivity of some typical dielectric materials.

Material	ε_r
Free space	1
Aerogels	~1.5
Polyimides (organic)	3-4
Silicon dioxide	3.9
Glass-epoxy (PC board)	5
Silicon Nitride (Si_3N_4)	7.5
Alumina (package)	9.5
Silicon	11.7

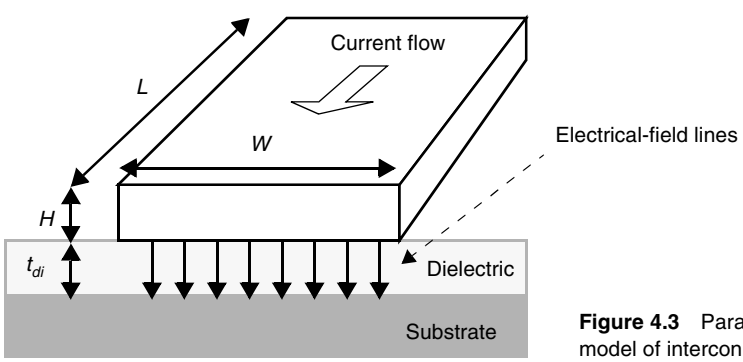
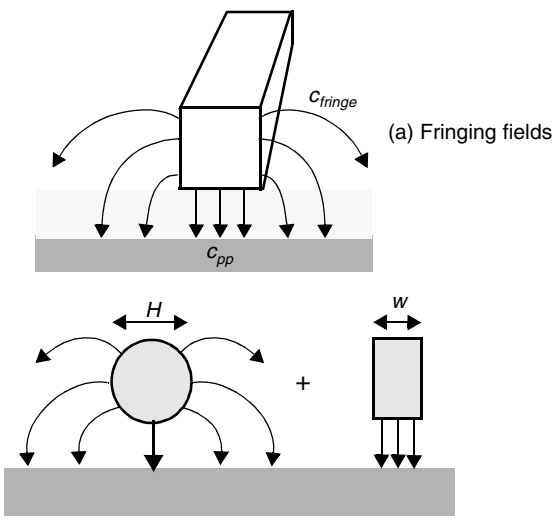


Figure 4.3 Parallel-plate capacitance model of interconnect wire.

[1] To differentiate between distributed (per unit length) wire parameters versus total lumped values, we will use lowercase to denote the former and uppercase for the latter.

In actuality, this model is too simplistic. To minimize the resistance of the wires while scaling technology, it is desirable to keep the cross-section of the wire ($W \times H$) as large as possible — as will become apparent in a later section. On the other hand, small values of W lead to denser wiring and less area overhead. As a result, we have over the years witnessed a steady reduction in the W/H-ratio, such that it has even dropped below unity in advanced processes. This is clearly visible on the process cross-section of FIGURE. Under those circumstances, the parallel-plate model assumed above becomes inaccurate. The capacitance between the side-walls of the wires and the substrate, called the *fringing capacitance*, can no longer be ignored and contributes to the overall capacitance. This effect is illustrated in Figure 4.4a. Presenting an exact model for this difficult geometry is hard. So, as good engineering practice dictates, we will use a simplified model that approximates the capacitance as the sum of two components (Figure 4.4b): a parallel-plate capacitance determined by the orthogonal field between a wire of width w and the ground plane, in parallel with the fringing capacitance modeled by a cylindrical wire with a dimension equal to the interconnect thickness H. The resulting approximation is simple and works fairly well in practice.



(b) Model of fringing-field capacitance.

Figure 4.4 The fringing-field capacitance. The model decomposes the capacitance into two contributions: a parallel-plate capacitance, and a fringing capacitance, modeled by a cylindrical wire with a diameter equal to the thickness of the wire.

$$c_{wire} = c_{pp} + c_{fringe} = \frac{w\varepsilon_{di}}{t_{di}} + \frac{2\pi\varepsilon_{di}}{\log(t_{di}/H)} \tag{4.2}$$

with $w = W - H/2$ a good approximation for the width of the parallel-plate capacitor. Numerous more accurate models (e.g. [Vdmeijs84]) have been developed over time, but these tend to be substantially more complex, and defeat our goal of developing a conceptual understanding.

To illustrate the importance of the fringing-field component, Figure 4.5 plots the value of the wiring capacitance as a function of (W/H). For larger values of (W/H) the total capacitance approaches the parallel-plate model. For (W/H) smaller than 1.5, the fringing

component actually becomes the dominant component. The fringing capacitance can increase the overall capacitance by a factor of more than 10 for small line widths. It is interesting to observe that the total capacitance levels off to a constant value of approximately 1 pF/cm for line widths smaller than the insulator thickness. In other words, the capacitance is no longer a function of the width.

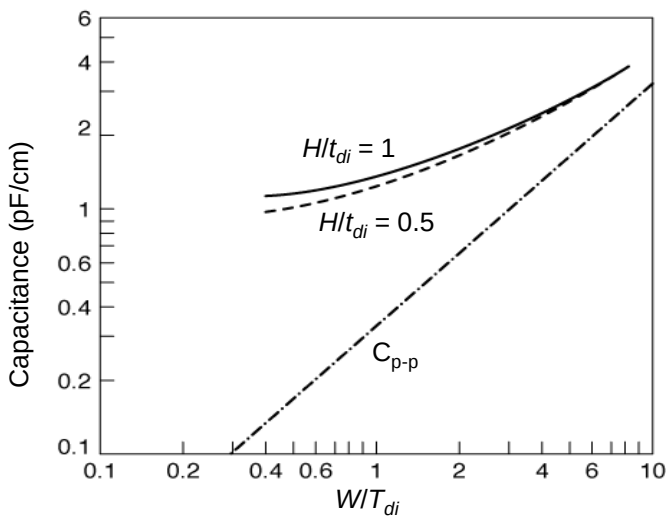


Figure 4.5 Capacitance of interconnect wire as a function of (W/t_{di}), including fringing-field effects (from [Schaper83]). Two values of H/t_{di} are considered. Silicon-dioxide with ε_r = 3.9 is used as dielectric.

So far, we have restricted our analysis to the case of a single rectangular conductor placed over a ground plane. This structure, called a *microstripline*, used to be a good model for semiconductor interconnections when the number of interconnect layers was restricted to 1 or 2. Today's processes offer many more layers of interconnect, which are packed quite densely in addition. In this scenario, the assumption that a wire is completely isolated from its surrounding structures and is only capacitively coupled to ground, becomes untenable. This is illustrated in Figure 4.6, where the capacitance components of a wire embedded in an interconnect hierarchy are identified. Each wire is not only coupled to the grounded substrate, but also to the neighboring wires on the same layer and on adjacent layers. To a first order, this does not change the total capacitance connected to a given wire. The main difference is that not all its capacitive components do terminate at the grounded substrate, but that a large number of them connect to other wires, which have dynamically varying voltage levels. We will later see that these *floating capacitors* form not only a source of noise (crosstalk), but also can have a negative impact on the performance of the circuit.

In summary, inter-wire capacitances become a dominant factor in multi-layer interconnect structures. This effect is more outspoken for wires in the higher interconnect layers, as these wires are farther away from the substrate. The increasing contribution of the inter-wire capacitance to the total capacitance with decreasing feature sizes is best illustrated by Figure 4.7. In this graph, which plots the capacitive components of a set of parallel wires routed above a ground plane, it is assumed that dielectric and wire thickness are

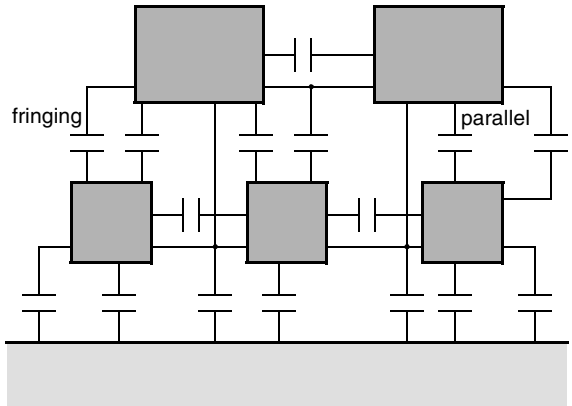


Figure 4.6 Capacitive coupling between wires in interconnect hierarchy.

held constant while scaling all other dimensions. When *W* becomes smaller than 1.75 *H*, the inter-wire capacitance starts to dominate.

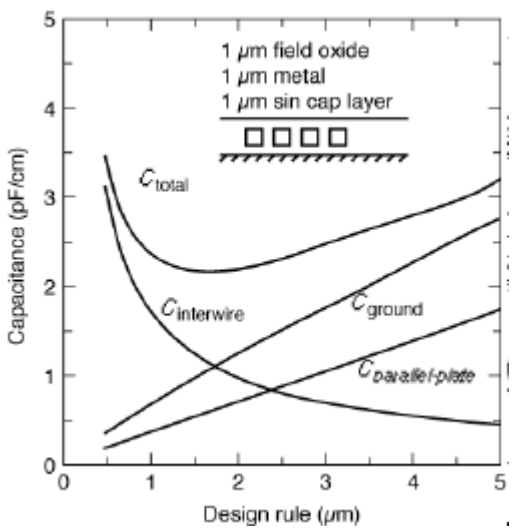


Figure 4.7 Interconnect capacitance as a function of design rules. It consists of a capacitance to ground and an inter-wire capacitance (from [Schaper83]).

Interconnect Capacitance Design Data

A set of typical interconnect capacitances for a standard 0.25 μm CMOS process are given in Table 4.2. The process supports 1 layer of polysilicon and 5 layers of Aluminum. Metal layers 1 to 4 have the same thickness and use a similar dielectric, while the wires at metal layer 5 are almost twice as thick and are embedded in a dielectric with a higher permittivity. When placing the wires over the thick field oxide that is used to isolate different transistors, use the “Field” column in the table, while wires routed over the active area see a higher capacitance as seen in the “Active” column. Be aware that the presented values are only indicative. To obtain more

accurate results for actual structures, complex 3-dimensional models should be used that take the environment of the wire into account.

Table 4.2 Wire area and fringe capacitance values for typical 0.25 μm CMOS process. The table rows represent the top plate of the capacitor, the columns the bottom plate. The area capacitances are expressed in aF/μm^2, while the fringe capacitances (given in the shaded rows) are in aF/μm.

	Field	Active	Poly	Al1	Al2	Al3	Al4
Poly	88						
	54						
Al1	30	41	57				
	40	47	54				
Al2	13	15	17	36			
	25	27	29	45			
Al3	8.9	9.4	10	15	41		
	18	19	20	27	49		
Al4	6.5	6.8	7	8.9	15	35	
	14	15	15	18	27	45	
Al5	5.2	5.4	5.4	6.6	9.1	14	38
	12	12	12	14	19	27	52

Table 4.3 tabulates indicative values for the capacitances between parallel wires placed on the same layer with a minimum spacing (as dictated by the design rules). Observe that these numbers include both the parallel plate and fringing components. Once again, the capacitances are a strong function of the topology. For instance, a ground plane placed on a neighboring layer terminates a large fraction of the fringing field, and effectively reduces the inter-wire capacitance. The polysilicon wires experience a reduced inter-wire capacitance due to the smaller thickness of the wires. On the other hand, the thick Al5 wires display the highest inter-wire capacitance. It is therefore advisable to either separate wires at this level by an amount that is larger than the minimum allowed, or to use it for global signals that are not that sensitive to interference. The supply rails are an example of the latter.

Table 4.3 inter-wire capacitance per unit wire length for different interconnect layers of typical 0.25 μm CMOS process. The capacitances are expressed in aF/μm, and are for minimally-spaced wires.

Layer	Poly	Al1	Al2	Al3	Al4	Al5
Capacitance	40	95	85	85	85	115

Example 4.1 Capacitance of Metal Wire

Some global signals, such as clocks, are distributed all over the chip. The length of those wires can be substantial. For die sizes between 1 and 2 cm, wires can reach a length of 10 cm and have associated wire capacitances of substantial value. Consider an aluminum wire of 10 cm long and 1 μm wide, routed on the first Aluminum layer. We can compute the value of the total capacitance using the data presented in Table 4.2.

$$\text{Area (parallel-plate) capacitance: } (0.1 \times 10^6\ \mu m^2) \times 30\ aF/\mu m^2 = 3\ pF$$

$$\text{Fringing capacitance: } 2 \times (0.1 \times 10^6\ \mu m) \times 40\ aF/\mu m = 8\ pF$$

$$\text{Total capacitance: } 11\ pF$$

Notice the factor 2 in the computation of the fringing capacitance, which takes the two sides of the wire into account.

Suppose now that a second wire is routed alongside the first one, separated by only the minimum allowed distance. From Table 4.3, we can determine that this wire will couple to the first with a capacitance equal to

$$C_{inter} = (0.1 \times 10^6\ \mu m) \times 95\ aF/\mu m = 9.5\ pF$$

which is almost as large as the total capacitance to ground!

A similar exercise shows that moving the wire to Al4 would reduce the capacitance to ground to 3.45 pF (0.65 pF area and 2.8 pF fringe), while the inter-wire capacitance would remain approximately the same at 8.5 pF.

4.3.2 Resistance

The resistance of a wire is proportional to its length L and inversely proportional to its cross-section A. The resistance of a rectangular conductor in the style of Figure 4.3 can be expressed as

$$R = \frac{\rho L}{A} = \frac{\rho L}{HW} \tag{4.3}$$

where the constant ρ is the resistivity of the material (in Ω-m). The resistivities of some commonly-used conductive materials are tabulated in Table 4.4. Aluminum is the interconnect material most often used in integrated circuits because of its low cost and its compatibility with the standard integrated-circuit fabrication process. Unfortunately, it has a large resistivity compared to materials such as Copper. With ever-increasing performance targets, this is rapidly becoming a liability and top-of-the-line processes are now increasingly using Copper as the conductor of choice.

Table 4.4 Resistivity of commonly-used conductors (at 20 C).

Material	r **(**W-**m)**
Silver (Ag)	1.6×10^{-8}

Table 4.4 Resistivity of commonly-used conductors (at 20 C).

Material	r (W-m)
Copper (Cu)	1.7×10^{-8}
Gold (Au)	2.2×10^{-8}
Aluminum (Al)	2.7×10^{-8}
Tungsten (W)	5.5×10^{-8}

Since H is a constant for a given technology, Eq. (4.3) can be rewritten as follows,

$$R = R_{\square}\frac{L}{W} \tag{4.4}$$

with

$$R_{\square} = \frac{\rho}{H} \tag{4.5}$$

the *sheet resistance* of the material, having units of Ω/❑ (pronounced as Ohm-per-square). This expresses that the resistance of a square conductor is independent of its absolute size, as is apparent from Eq. (4.4). To obtain the resistance of a wire, simply multiply the sheet resistance by its ratio (L/W).

Interconnect Resistance Design Data

Typical values of the sheet resistance of various interconnect materials are given in Table 4.5.

Table 4.5 Sheet resistance values for a typical 0.25 μm CMOS process.

Material	Sheet Resistance (Ω/❑)
n- or p-well diffusion	1000 – 1500
n^+, p^+ diffusion	50 – 150
n^+, p^+ diffusion with silicide	3 – 5
n^+, p^+ polysilicon	150 – 200
n^+, p^+ polysilicon with silicide	4 – 5
Aluminum	0.05 – 0.1

From this table, we conclude that Aluminum is the preferred material for the wiring of long interconnections. Polysilicon should only be used for local interconnect. Although the sheet resistance of the diffusion layer (n^+, p^+) is comparable to that of polysilicon, the use of diffusion wires should be avoided due to its large capacitance and the associated *RC* delay.

Advanced processes also offer silicided polysilicon and diffusion layers. A silicide is a compound material formed using silicon and a refractory metal. This creates a highly conductive material that can withstand high-temperature process steps without melting. Examples of silicides are WSi_2, $TiSi_2$, $PtSi_2$, and TaSi. WSi_2, for instance, has a resistivity ρ of 130 μΩ-cm, which is approximately eight times lower than polysilicon. The silicides are most often used in a configuration called a *polycide*, which is a simple layered combination of polysilicon and a silicide. A typical polycide consists of a lower level of polysilicon with an upper coating of silicide and combines the best properties of both materials—good adherence and coverage (from the poly) and high conductance (from the silicide). A MOSFET fabricated with a polycide gate is shown in Figure 4.8. The advantage of the silicided gate is a reduced gate resistance. Similarly, silicided source and drain regions reduce the source and drain resistance of the device.

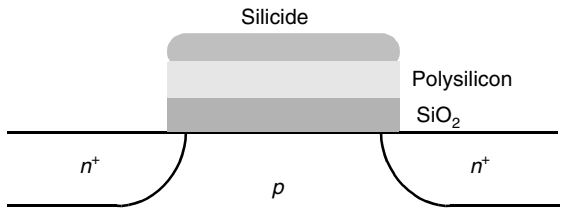


Figure 4.8 A polycide-gate MOSFET.

Transitions between routing layers add extra resistance to a wire, called the *contact resistance*. The preferred routing strategy is thus to keep signal wires on a single layer whenever possible and to avoid excess contacts or via's. It is possible to reduce the contact resistance by making the contact holes larger. Unfortunately, current tends to concentrate around the perimeter in a larger contact hole. This effect, called *current crowding*, puts a practical upper limit on the size of the contact. The following *contact resistances* (for minimum-size contacts) are typical for a 0.25 μm process: 5-20 Ω for metal or polysilicon to n^+, p^+, and metal to polysilicon; 1-5 Ω for via's (metal-to-metal contacts).

Example 4.2 Resistance of a Metal Wire

Consider again the aluminum wire of Example 4.2, which is 10 cm long and 1 μm wide, and is routed on the first Aluminum layer. Assuming a sheet resistance for Al1 of 0.075 Ω/❑, we can compute the total resistance of the wire

$$R_{wire} = 0.075\ \Omega/\square \times (0.1 \times 10^6\ \mu m) / (1\ \mu m) = 7.5\ k\Omega$$

Implementing the wire in polysilicon with a sheet resistance of 175 Ω/❑ raises the overall resistance to 17.5 MΩ, which is clearly unacceptable. Silicided polysilicon with a sheet resistance of 4 Ω/❑ offers a better alternative, but still translates into a wire with a 400 kΩ resistance.

So far, we have considered the resistance of a semiconductor wire to be linear and constant. This is definitely the case for most semiconductor circuits. At very high frequencies however, an additional phenomenon — called the *skin effect* — comes into play such that the resistance becomes frequency-dependent. High-frequency currents tend to flow primarily on the surface of a conductor with the current density falling off exponentially

with depth into the conductor. The *skin depth* δ is defined as the depth where the current falls off to a value of e^{-1} of its nominal value, and is given by

$$\delta = \sqrt{\frac{\rho}{\pi f \mu}} \tag{4.6}$$

with f the frequency of the signal and μ the permeability of the surrounding dielectric (typically equal to the permeability of free space, or $\mu = 4\pi \times 10^{-7}$ H/m). For Aluminum at 1 GHz, the skin depth is equal to 2.6 μm. The obvious question is now if this is something we should be concerned about when designing state-of-the-art digital circuits?

The effect can be approximated by assuming that the current flows uniformly in an outer shell of the conductor with thickness δ, as is illustrated in Figure 4.9 for a rectangular wire. Assuming that the overall cross-section of the wire is now limited to approximately $2(W+H)\delta$, we obtain the following expression for the resistance (per unit length) at high frequencies ($f > f_s$):

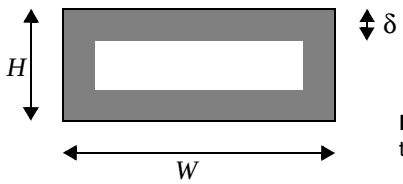


Figure 4.9 The skin-effect reduces the flow of the current to the surface of the wire.

$$r(f) = \frac{\sqrt{\pi f \mu \rho}}{2(H + W)} \tag{4.7}$$

The increased resistance at higher frequencies may cause an extra attenuation — and hence distortion — of the signal being transmitted over the wire. To determine the on-set of the skin-effect, we can find the frequency f_s where the skin depth is equal to half the largest dimension (W or H) of the conductor. Below f_s the whole wire is conducting current, and the resistance is equal to (constant) low-frequency resistance of the wire. From Eq. (4.6), we find the value of f_s:

$$f_s = \frac{4\rho}{\pi \mu (\max(W, H))^2} \tag{4.8}$$

Example 4.3 Skin-effect and Aluminum wires

We determine the impact of the skin-effect on contemporary integrated circuits by analyzing an Aluminum wire with a resistivity of 2.7×10^{-8} Ω-m, embedded in a SiO_2 dielectric with a permeability of $4\pi \times 10^{-7}$ H/m. From Eq. (4.8), we find that the largest dimension of wire should be at least 5.2 μm for the effect to be noticeable at 1 GHz. This is confirmed by the more accurate simulation results of Figure 4.10, which plots the increase in resistance due to skin effects for different width Aluminum conductors. A 30% increase in

resistance can be observed at 1 GHz for a 20 μm wire, while the increase for a 1 μm wire is less than 1%.

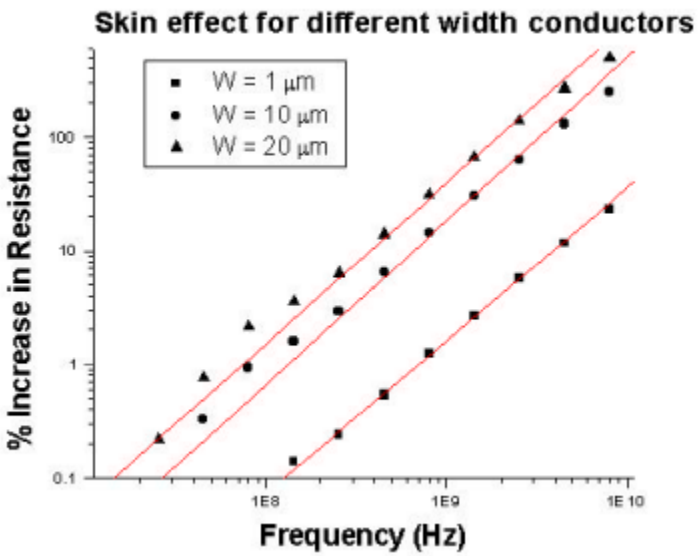


Figure 4.10 Skin-effect induced increase in resistance as a function of frequency and wire width. All simulations were performed for a wire thickness of 0.7 μm [Sylvester97].

In summary, the skin-effect is only an issue for wider wires. Since clocks tend to carry the highest-frequency signals on a chip and also are fairly wide to limit resistance, the skin effect is likely to have its first impact on these lines. This is a real concern for GHz-range design, as clocks determine the overall performance of the chip (cycle time, instructions per second, etc.). Another major design concern is that the adoption of better conductors such as Copper may move the on-set of skin-effects to lower frequencies.

4.3.3 Inductance

Integrated-circuit designers tend to dismiss inductance as something they heard about in their physics classes, but that has no impact on their field. This was definitely the case in the first decades of integrated digital circuit design. Yet with the adoption of low-resistive interconnect materials and the increase of switching frequencies to the super GHz range, inductance starts to play a role even on a chip. Consequences of on-chip inductance include ringing and overshoot effects, reflections of signals due to impedance mismatch, inductive coupling between lines, and switching noise due to Ldi/dt voltage drops.

The inductance of a section of a circuit can always be evaluated with the aid of its definition, which states that a changing current passing through an inductor generates a voltage drop ΔV

$$\Delta V = L\frac{\mathrm{d}i}{\mathrm{d}t} \tag{4.9}$$

It is possible to compute the inductance a wire directly from its geometry and its environment. A simpler approach relies on the fact that the capacitance c and the inductance l (per unit length) of a wire are related by the following expression

$$cl = \varepsilon\mu \tag{4.10}$$

with ε and μ respectively the permittivity and permeability of the surrounding dielectric. The caveat is that for this expression to be valid the conductor must be completely surrounded by a uniform dielectric medium. This is most often not the case. Yet even when the wire is embedded in different dielectric materials, its is possible to adopt "average" dielectric constants such that Eq. (4.10) still can be used to get an approximative value of the inductance.

Some other interesting relations, obtained from Maxwell's laws, can be pointed out. The constant product of permeability and permittivity also defines the speed ν at which an electromagnetic wave can propagate through the medium

$$\nu = \frac{1}{\sqrt{lc}} = \frac{1}{\sqrt{\varepsilon\mu}} = \frac{c_0}{\sqrt{\varepsilon_r\mu_r}} \tag{4.11}$$

c_0 equals the speed of light (30 cm/nsec) in a vacuum. The propagation speeds for a number of materials used in the fabrication of electronic circuits are tabulated in Table 4.6. The propagation speed for SiO_2 is two times slower than in a vacuum.

Table 4.6 Dielectric constants and wave-propagation speeds for various materials used in electronic circuits. The relative permeability μ_r of most dielectrics is approximately equal to 1.

Dielectric	ε_r	Propagation speed (cm/nsec)
Vacuum	1	30
SiO_2	3.9	15
PC board (epoxy glass)	5.0	13
Alumina (ceramic package)	9.5	10

Example 4.4 Inductance of a Semiconductor Wire

Consider an Al1 wire implemented in the 0.25 micron CMOS technology and routed on top of the field oxide. From Table 4.2, we can derive the capacitance of the wire per unit length:

$$c = (W \times 30 + 2 \times 40)\ \text{aF}/\mu\text{m}$$

From Eq. (4.10), we can derive the inductance per unit length of the wire, assuming SiO_2 as the dielectric and assuming a uniform dielectric (make sure to use the correct units!)

$$l = (3.9 \times 8.854 \times 10^{-12}) \times (4\,\pi\,10^{-7}) / C$$

For wire widths of 0.4 mm, 1mm and 10mm, this leads to the following numbers:

$W = 0.4\ \mu m$: $c = 92$ aF/μm; $l = 0.47$ pH/μm
$W = 1\ \mu m$: $c = 110$ aF/μm; $l = 0.39$ pH/μm
$W = 10\ \mu m$: $c = 380$ aF/μm; $l = 0.11$ pH/μm

Assuming a sheet resistance of 0.075 Ω/❑, we can also determine the resistance of the wire,

$$r = 0.075/W\ \Omega/\mu m$$

It is interesting to observe that the inductive part of the wire impedance becomes equal in value to the resistive component at a frequency of 27.5 GHz (for a 1 μm wide wire), as can be obtained from solving the following expression:

$$\omega l = 2\pi f l = r$$

For extra wide wires, this frequency reduces to approximately 11 GHz. For wires with a smaller capacitance and resistance (such as the thicker wires located at the upper interconnect layers), this frequency can become as low as 500 MHz, especially when better interconnect materials such as Copper are being used. Yet, these numbers indicate that inductance only becomes an issue in integrated circuits for frequencies that are well above 1 GHz.

4.4 Electrical Wire Models

In previous sections, we have introduced the electrical properties of the interconnect wire — capacitance, resistance, and inductance — and presented some simple relations and techniques to derive their values from the interconnect geometries and topologies. These parasitic elements have an impact on the electrical behavior of the circuit and influence its delay, power dissipation, and reliability. To study these effects requires the introduction of electrical models that estimate and approximate the real behavior of the wire as a function of its parameters. These models vary from very simple to very complex depending upon the effects that are being studied and the required accuracy. In this section, we first derive models for manual analysis, while how to cope with interconnect wires in the SPICE circuit simulator is the topic follows next.

4.4.1 The Ideal Wire

In schematics, wires occur as simple lines with no attached parameters or parasitics. These wires have no impact on the electrical behavior of the circuit. A voltage change at one end of the wire propagates immediately to its other ends, even if those are some distance away. Hence, it may be assumed that the same voltage is present at every segment of the wire at the every point in time, and that the whole wire is an *equipotential region*. While this *ideal-wire model* is simplistic, it has its value, especially in the early phases of the design process when the designer wants to concentrate on the properties and the behavior of the transistors that are being connected. Also, when studying small circuit components such as gates, the wires tend to be very short and their parasitics ignorable. Taking these into account would just make the analysis unnecessarily complex. More often though, wire parasitics play a role and more complex models should be considered.

4.4.2 The Lumped Model

The circuit parasitics of a wire are distributed along its length and are not lumped into a single position. Yet, when only a single parasitic component is dominant, when the interaction between the components is small, or when looking at only one aspect of the circuit behavior, it is often useful to lump the different fractions into a single circuit element. The advantage of this approach is that the effects of the parasitic then can be described by an ordinary differential equation. As we will see later, the description of a distributed element requires partial differential equations.

As long as the resistive component of the wire is small and the switching frequencies are in the low to medium range, it is meaningful to consider only the capacitive component of the wire, and to lump the distributed capacitance into a single capacitor as shown in Figure 4.11. Observe that in this model the wire still represents an equipotential region, and that the wire itself does not introduce any delay. The only impact on performance is introduced by the loading effect of the capacitor on the driving gate. This capacitive lumped model is simple, yet effective, and is the model of choice for the analysis of most interconnect wires in digital integrated circuits.

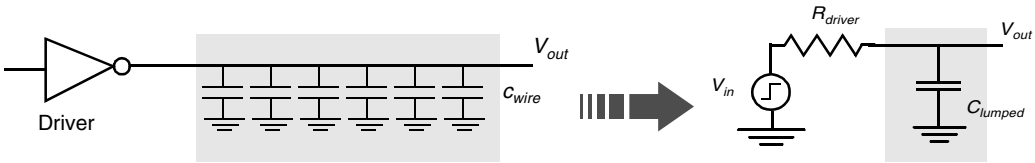


Figure 4.11 Distributed versus lumped capacitance model of wire. $C_{lumped} = L \times c_{wire}$, with L the length of the wire and c_{wire} the capacitance per unit length. The driver is modeled as a voltage source and a source resistance R_{driver}.

Example 4.5 Lumped capacitance model of wire

For the circuit of Figure 4.11, assume that a driver with a source resistance of 10 kΩ is used to drive a 10 cm long, 1 µm wide Al1 wire. In Example 4.1, we have found that the total lumped capacitance for this wire equals 11 pF.

The operation of this simple RC network is described by the following ordinary differential equation (similar to the expression derived in Example 1.6):

$$C_{lumped}\frac{dV_{out}}{dt} + \frac{V_{out} - V_{in}}{R_{driver}} = 0$$

When applying a step input (with V_{in} going from 0 to V), the transient response of this circuit is known to be an exponential function, and is given by the following expression (where $\tau = R_{driver} \acute{} C_{lumped}$, the time constant of the network):

$$V_{out}(t) = (1 - e^{-t/\tau})\ V$$

The time to reach the 50% point is easily computed as $t = \ln(2)\tau = 0.69\tau$. Similarly, it takes $t = \ln(9)\tau = 2.2\tau$ to get to the 90% point. Plugging in the numbers for this specific example yields

$$t_{50\%} = 0.69 \times 10\ \text{K}\Omega \times 11\ \text{pF} = 76\ \text{nsec}$$
$$t_{90\%} = 2.2 \times 10\ \text{K}\Omega \times 11\ \text{pF} = 242\ \text{nsec}$$

These numbers are not even acceptable for the lowest performance digital circuits. Techniques to deal with this bottleneck, such as reducing the source resistance of the driver, will be introduced in Chapter ZZZ.

While the lumped capacitor model is the most popular, sometimes it is also useful to present lumped models of a wire with respect to either resistance and inductance. This is often the case when studying the supply distribution network. Both the resistance and inductance of the supply wires can be interpreted as parasitic noise sources that introduce voltage drops and bounces on the supply rails.

4.4.3 The Lumped *RC* model

On-chip metal wires of over a few mm length have a significant resistance. The equipotential assumption, presented in the lumped-capacitor model, is no longer adequate, and a resistive-capacitive model has to be adopted.

A first approach lumps the total wire resistance of each wire segment into one single *R* and similarly combines the global capacitance into a single capacitor *C*. This simple model, called the *lumped RC model* is pessimistic and inaccurate for long interconnect wires, which are more adequately represented by a *distributed rc-model*. Yet, before analyzing the distributed model, its is worthwhile to spend some time on the analysis and the modeling of lumped *RC* networks for the following reasons:

- The distributed *rc*-model is complex and no closed form solutions exist. The behavior of the distributed *rc*-line can be adequately modeled by a simple *RC* network.
- A common practice in the study of the transient behavior of complex transistor-wire networks is to reduce the circuit to an *RC* network. Having a means to analyze such a network effectively and to predict its first-order response would add a great asset to the designers tool box.

In Example 4.5, we analyzed a single resistor-single capacitor network. The behavior of such a network is fully described by a single differential equation, and its transient waveform is a modeled by an exponential with a single time-constant (or network pole). Unfortunately, deriving the correct waveforms for a network with a larger number of capacitors and resistors rapidly becomes hopelessly complex: describing its behavior requires a set of ordinary differential equations, and the network now contains many time-constants (or poles and zeros). Short of running a full-fledged SPICE simulation, delay calculation methods such as the *Elmore delay formula* come to the rescue [Elmore48].

Consider the resistor-capacitor network of Figure 4.12. This circuit is called an *RC-tree* and has the following properties:

- the network has a single input node (called *s* in Figure 4.12)
- all the capacitors are between a node and the ground
- the network does not contain any resistive loops (which makes it a tree)

An interesting result of this particular circuit topology is that there exists a unique resistive path between the source node *s* and any node *i* of the network. The total resistance along

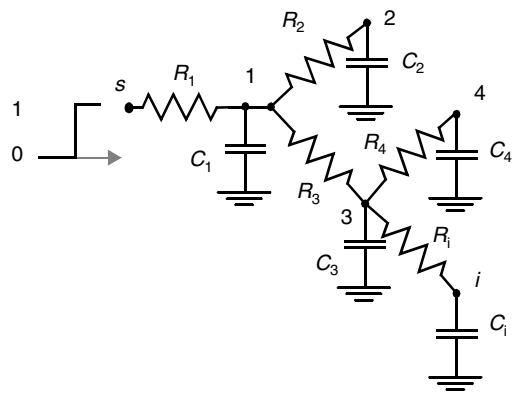


Figure 4.12 Tree-structured *RC* network.

this path is called the *path resistance* R_{ii}. For example, the path resistance between the source node *s* and node 4 in the example of Figure 4.12 equals

$$R_{44} = R_1 + R_3 + R_4$$

The definition of the path resistance can be extended to address the *shared path resistance* R_{ik}, which represents the resistance shared among the paths from the root node *s* to nodes *k* and *i*:

$$R_{ik} = \sum R_j \Rightarrow (R_j \in [path(s \rightarrow i) \cap path(s \rightarrow k)]) \tag{4.12}$$

For the circuit of Figure 4.12, $R_{i4} = R_1 + R_3$ while $R_{i2} = R_1$.

Assume now that each of the *N* nodes of the network is initially discharged to GND, and that a step input is applied at node *s* at time $t = 0$. The Elmore delay at node *i* is then given by the following expression:

$$\tau_{Di} = \sum_{k=1}^{N} C_k R_{ik} \tag{4.13}$$

The Elmore delay is equivalent to the first-order time constant of the network (or the first moment of the impulse response). The designer should be aware that this time-constant represents a simple approximation of the actual delay between source node and node *i*. Yet in most cases this approximation has proven to be quite reasonable and acceptable. It offers the designer a powerful mechanism for providing a quick estimate of the delay of a complex network.

Example 4.6 *RC* delay of a tree-structured network

Using Eq. (4.13), we can compute the Elmore delay for node *i* in the network of Figure 4.12.

$$\tau_{Di} = R_1C_1 + R_1C_2 + (R_1 + R_3)C_3 + (R_1 + R_3)C_4 + (R_1 + R_3 + R_i)C_i$$

As a special case of the *RC* tree network, let us consider the simple, non-branched *RC* chain (or ladder) shown in Figure 4.13. This network is worth analyzing because it is a structure that is often encountered in digital circuits, and also because it represents an

approximative model of a resistive-capacitive wire. The Elmore delay of this chain network can be derived with the aid of Eq. (4.13):

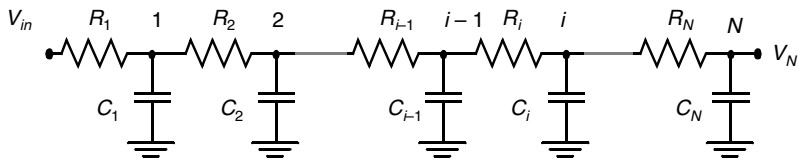


Figure 4.13 *RC* chain.

$$\tau_{DN} = \sum_{i=1}^{N} C_i \sum_{j=1}^{i} R_j = \sum_{i=1}^{N} C_i R_{ii} \tag{4.14}$$

or the shared-path resistance is replaced by simply the path resistance. As an example, consider node 2 in the *RC* chain of Figure 4.13. Its time-constant consists of two components contributed by nodes 1 and 2. The component of node 1 consists of C_1R_1 with R_1 the total resistance between the node and the source, while the contribution of node 2 equals $C_2(R_1 + R_2)$. The equivalent time constant at node *2* equals $C_1R_1 + C_2(R_1 + R_2)$. τ_i of node *i* can be derived in a similar way.

$$\tau_{Di} = C_1R_1 + C_2(R_1 + R_2) + \ldots + C_i(R_1 + R_2 + \ldots + R_i)$$

Example 4.7 Time-Constant of Resistive-Capacitive Wire

The model presented in Figure 4.13 can be used as an approximation of a resistive-capacitive wire. The wire with a total length of *L* is partitioned into *N* identical segments, each with a length of *L*/*N*. The resistance and capacitance of each segment are hence given by *rL*/*N* and *cL*/*N*, respectively. Using the Elmore formula, we can compute the dominant time-constant of the wire:

$$\tau_{DN} = \left(\frac{L}{N}\right)^2 (rc + 2rc + \ldots + Nrc) = (rcL^2)\frac{N(N+1)}{2N^2} = RC\frac{N+1}{2N} \tag{4.15}$$

with *R* (= *rL*) and *C* (= *cL*) the total lumped resistance and capacitance of the wire. For very large values of *N*, this model asymptotically approaches the distributed *rc* line. Eq. (4.15) then simplifies to the following expression:

$$\tau_{DN} = \frac{RC}{2} = \frac{rcL^2}{2} \tag{4.16}$$

Eq. (4.16) leads to two important conclusions:

- The delay of a wire is **a quadratic function of its length**! This means that doubling the length of the wire quadruples its delay.
- The delay of the distributed *rc*-line is **one half of the delay** that would have been predicted by the lumped *RC* model. The latter combines the total resistance and capacitance into single elements, and has a time-constant equal to *RC* (as is also

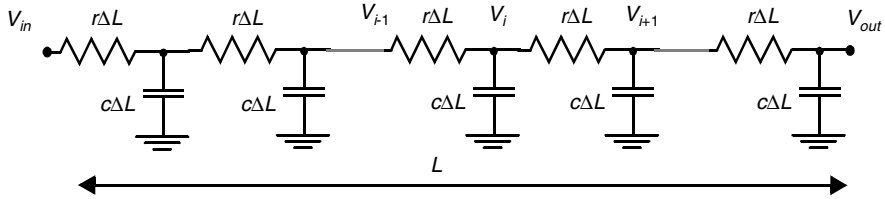
obtained by setting $N = 1$ in Eq. (4.15)). This confirms the observation made earlier that the lumped model presents a pessimistic view on the delay of resistive wire.

WARNING: Be aware that an *RC*-chain is characterized by a number of time-constants. The Elmore expression determines the value of only the dominant one, and presents thus a first-order approximation.

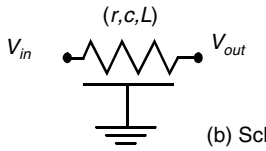
The Elmore delay formula has proven to be extremely useful. Besides making it possible to analyze wires, the formula can also be used to approximate the propagation delay of complex transistor networks. In the switch model, transistors are replaced by their equivalent, linearized on-resistance. The evaluation of the propagation delay is then reduced to the analysis of the resulting *RC* network. More precise minimum and maximum bounds on the voltage waveforms in an *RC* tree have further been established [Rubinstein83]. These bounds have formed the base for most computer-aided timing analyzers at the switch and functional level [Horowitz83]. An interesting result [Lin84] is that the exponential voltage waveform with the Elmore delay as time constant is always situated between these min and max bounds, which demonstrates the validity of the Elmore approximation.

4.4.4 The Distributed *rc* Line

In the previous paragraphs, we have shown that the lumped *RC* model is a pessimistic model for a resistive-capacitive wire, and that a distributed *rc* model (Figure 4.14a) is more appropriate. As before, *L* represents the total length of the wire, while *r* and *c* stand for the resistance and capacitance per unit length. A schematic representation of the distributed *rc* line is given in Figure 4.14b.



(a) Distributed model



(b) Schematic symbol for distributed *RC* line

Figure 4.14 Distributed RC line wire-model and its schematic symbol.

The voltage at node *i* of this network can be determined by solving the following set of partial differential equations:

$$c\Delta L\frac{\partial V_i}{\partial t} = \frac{(V_{i+1} - V_i) + (V_{i-1} - V_i)}{r\Delta L} \tag{4.17}$$

The correct behavior of the distributed *rc* line is then obtained by reducing ΔL asymptotically to 0. For $\Delta L \rightarrow 0$, Eq. (4.17) becomes the well-known *diffusion equation*:

$$rc\frac{\partial V}{\partial t} = \frac{\partial^2 V}{\partial x^2} \tag{4.18}$$

where V is the voltage at a particular point in the wire, and x is the distance between this point and the signal source. No closed-form solution exists for this equation, but approximative expressions such as the formula presented in Eq. (4.19) can be derived [Bakoglu90]. These equations are difficult to use for ordinary circuit analysis. It is known however that the distributed *rc* line can be approximated by a lumped *RC* ladder network, which can be easily used in computer-aided analysis. Some of these models will be presented in a later section, discussing SPICE wire models.

$$\begin{aligned} V_{out}(t) &= 2erfc\left(\sqrt{\frac{RC}{4t}}\right) && t \ll RC \\ &= 1.0 - 1.366e^{-2.5359\frac{t}{RC}} + 0.366e^{-9.4641\frac{t}{RC}} && t \gg RC \end{aligned} \tag{4.19}$$

Figure 4.15 shows the response of a wire to a step input, plotting the waveforms at different points in the wire as a function of time. Observe how the step waveform "diffuses" from the start to the end of the wire, and the waveform rapidly degrades, resulting in a considerable delay for long wires. Driving these *rc* lines and minimizing the delay and signal degradation is one of the trickiest problems in modern digital integrated circuit design. It hence will receive considerable attention in later chapters.

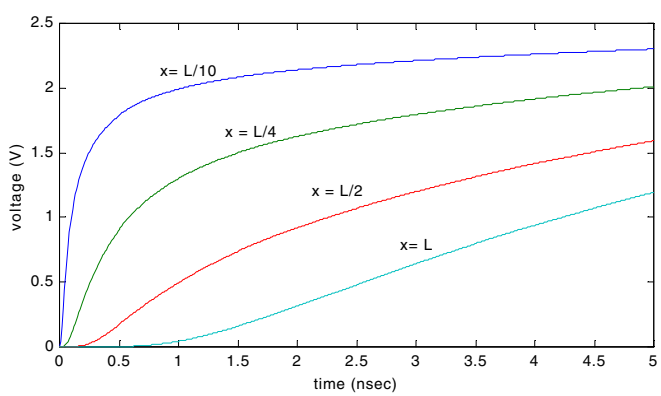


Figure 4.15 Simulated step response of resistive-capacitive wire as a function of time and place.

Some of the important reference points in the step response of the lumped and the distributed RC model of the wire are tabulated in Table 4.7. For instance, the propagation delay (defined at 50% of the final value) of the lumped network not surprisingly equals 0.69 *RC*. The distributed network, on the other hand, has a delay of only 0.38 *RC*, with *R* and *C* the total resistance and capacitance of the wire. This confirms the result of Eq. (4.16).

Table 4.7 Step response of lumped and distributed *RC* networks—points of Interest.

Voltage range	Lumped *RC* network	Distributed *RC* network
$0 \rightarrow 50\%$ (t_p)	0.69 *RC*	0.38 *RC*
$0 \rightarrow 63\%$ (τ)	*RC*	0.5 *RC*
$10\% \rightarrow 90\%$ (t_r)	2.2 *RC*	0.9 *RC*
$0\% \rightarrow 90\%$	2.3 *RC*	1.0 *RC*

Example 4.8 *RC* delay of Aluminum Wire

Let us consider again the 10 cm long, 1 μm wide Al1 wire of Example 4.1. In Example 4.4, we derived the following values for *r* and *c*:

$$c = 110 \text{ aF/}\mu\text{m}; \; r = 0.075\ \Omega/\mu\text{m};$$

Using the entry of Table 4.7, we derive the propagation delay of the wire:

$$t_p = 0.38\ RC = 0.38 \times (0.075\ \Omega/\mu\text{m}) \times (110 \text{ aF/}\mu\text{m}) \times (10^5\ \mu\text{m})^2 = 31.4 \text{ nsec}$$

We can also deduce the propagation delays of an identical wire implemented in polysilicon and Al5. The values of the capacitances are obtained from Table 4.2, while the resistances are assumed to be respectively 150 Ω/μm and 0.0375 Ω/μm for Poly and Al5:

$$\text{Poly: } t_p = 0.38 \times (150\ \Omega/\mu\text{m}) \times (88 + 2 \times 54 \text{ aF/}\mu\text{m}) \times (10^5\ \mu\text{m})^2 = 112\ \mu\text{sec!}$$
$$\text{Al5: } t_p = 0.38 \times (0.0375\ \Omega/\mu\text{m}) \times (5.2 + 2 \times 12 \text{ aF/}\mu\text{m}) \times (10^5\ \mu\text{m})^2 = 4.2 \text{ nsec}$$

Obviously, the choice of the interconnect material and layer has a dramatic impact on the delay of the wire.

An important question for a designer to answer when analyzing an interconnect network whether the effects of *RC* delays should be considered, or whether she can get away with a simpler lumped capacitive model. A simple rule of thumb proves to be very useful here.

Design Rules of Thumb

- ***rc* delays should only be considered when $t_{pRC} >> t_{pgate}$ of the driving gate.**

This translates into Eq. (4.20), which determines the critical length *L* of the interconnect wire where *RC* delays become dominant.

$$L_{crit} >> \sqrt{\frac{t_{pgate}}{0.38rc}} \tag{4.20}$$

The actual value of L_{crit} depends upon the sizing of the driving gate and the chosen interconnect material.

- ***rc* delays should only be considered when the rise (fall) time at the line input is smaller than *RC*, the rise (fall) time of the line.**

$$t_{rise} < RC \tag{4.21}$$

with R and C the total resistance and capacitance of the wire. When this condition is not met, the change in signal is slower than the propagation delay of the wire, and a lumped capacitive model suffices.

Example 4.9 *RC* versus Lumped *C*

The presented rule can be illustrated with the aid of the simple circuit shown in Figure 4.16. It is assumed here that the driving gate can be modeled as voltage source with a finite source resistance R_s. The total propagation delay of the network can be approximated by the following expression, obtained by applying the Elmore formula:[2]

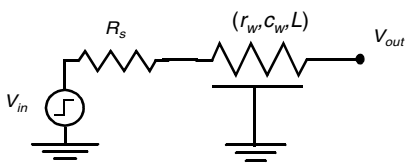


Figure 4.16 *rc*-line of length *L* driven by source with resistance equal to R_s.

$$\tau_D = R_s C_w + \frac{R_w C_w}{2} = R_s C_w + 0.5 r_w c_w L^2$$

and

$$t_p = 0.69 R_s C_w + 0.38 R_w C_w$$

with $R_w = rL$ and $C_w = cL$. The delay introduced by the wire resistance becomes dominant when $(R_w C_w)/2 \geq R_s C_w$, or $L \geq 2R_s/r$. Assume now a driver with a source resistance of 1 kΩ, driving an Al1 wire of 1 μm wide (r = 0.075 Ω/μm). This leads to a critical length of 2.67 cm.

[2] Hint: replace the wire by the lumped RC network of Figure 4.13 and apply the Elmore equation on the resulting network.

4.4.5 The Transmission Line

When the switching speeds of the circuits become sufficiently fast, and the quality of the interconnect material become high enough so that the resistance of the wire is kept within bounds, the inductance of the wire starts to dominate the delay behavior, and transmission line effects must be considered. This is more precisely the case when the rise and fall times of the signal become comparable to the time of flight of the signal waveform across the line as determined by the speed of light. With the advent of Copper interconnect and the high switching speeds enabled by the deep-submicron technologies, transmission line effects are soon to be considered in the fastest CMOS designs.

In this section, we first analyze the transmission line model. Next, we apply it to the current semiconductor technology and determine when those effects should be actively considered in the design process.

Transmission Line Model

Similar to the resistance and capacitance of an interconnect line, the inductance is distributed over the wire. A distributed *rlc* model of a wire, known as the transmission line model, becomes the most accurate approximation of the actual behavior. The transmission line has the prime property that a signal propagates over the interconnection medium as a *wave*. This is in contrast to the distributed *rc* model, where the signal *diffuses* from the source to the destination governed by the diffusion equation, Eq. (4.18). In the wave mode, a signal propagates by alternatively transferring energy from the electric to the magnetic fields, or equivalently from the capacitive to the inductive modes.

Consider the point *x* along the transmission line of Figure 4.17 at time *t*. The following set of equations holds:

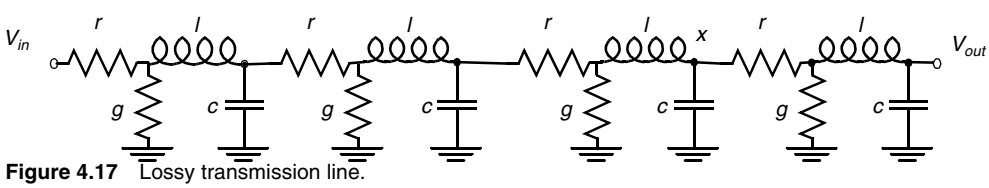


Figure 4.17 Lossy transmission line.

$$\frac{\partial v}{\partial x} = -ri - l\frac{\partial i}{\partial t}$$
$$\frac{\partial i}{\partial x} = -gv - c\frac{\partial v}{\partial t} \tag{4.22}$$

Assuming that the leakage conductance *g* equals 0, which is true for most insulating materials, and eliminating the current *i* yields the *wave propagation equation*, Eq. (4.23).

$$\frac{\partial^2 v}{\partial x^2} = rc\frac{\partial v}{\partial t} + lc\frac{\partial^2 v}{\partial t^2} \tag{4.23}$$

where r, c, and l are the resistance, capacitance, and inductance per unit length, respectively.

To understand the behavior of the transmission line, we will first assume that the resistance of the line is small. In this case, a simplified capacitive/inductive model, called the *lossless transmission line*, is appropriate. This model is applicable for wires at the printed-circuit board level. Due to the high conductivity of the Copper interconnect material used there, the resistance of the transmission line can be ignored. On the other hand, resistance plays an important role in integrated circuits, and a more complex model, called the *lossy transmission line* should be considered. The lossy model is only discussed briefly at the end.

The Lossless Transmission Line

For the lossless line, Eq. (4.23) simplifies to the *ideal wave* equation:

$$\frac{\partial^2 v}{\partial x^2} = lc\frac{\partial^2 v}{\partial t^2} = \frac{1}{\nu^2}\frac{\partial^2 v}{\partial t^2} \tag{4.24}$$

A step input applied to a lossless transmission line propagates along the line with a speed ν, given by Eq. (4.11) and repeated below.

$$\nu = \frac{1}{\sqrt{lc}} = \frac{1}{\sqrt{\varepsilon\mu}} = \frac{c_0}{\sqrt{\varepsilon_r\mu_r}} \tag{4.25}$$

Even though the values of both l and c depend on the geometric shape of the wire, their product is a constant and is only a function of the surrounding media. The propagation delay per unit wire length (t_p) of a transmission line is the inverse of the speed:

$$t_p = \sqrt{lc} \tag{4.26}$$

Let us now analyze how a wave propagates along a lossless transmission line. Suppose that a voltage step V has been applied at the input and has propagated to point x of the line (Figure 4.18). All currents are equal to 0 at the right side of x, while the voltage over the line equals V at the left side. An additional capacitance cdx must be charged for the wave to propagate over an additional distance dx. This requires the following current:

$$I = \frac{\mathrm{d}Q}{\mathrm{d}t} = c\frac{\mathrm{d}x}{\mathrm{d}t}V = c\nu V = \sqrt{\frac{c}{l}}V \tag{4.27}$$

since the propagation speed of the signal dx/dt equals ν. This means that the signal sees the remainder of the line as areal impedance,

$$Z_0 = \frac{V}{I} = \sqrt{\frac{l}{c}} = \frac{\sqrt{\varepsilon\mu}}{c} = \frac{1}{c\nu}. \tag{4.28}$$

This impedance, called the *characteristic impedance* of the line, is a function of the dielectric medium and the geometry of the conducting wire and isolator (Eq. (4.28)), and is independent of the length of the wire and the frequency. That a line of arbitrary length has a constant, real impedance is a great feature as it simplifies the design of the driver cir-

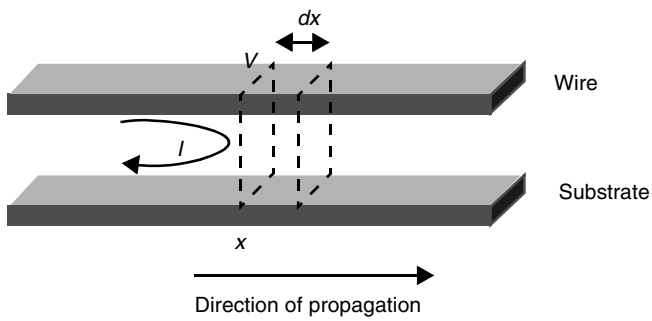


Figure 4.18 Propagation of voltage step along a lossless transmission line.

cuitry. Typical values of the characteristic impedance of wires in semiconductor circuits range from 10 to 200 Ω.

Example 4.10 Propagation Speeds of Signal Waveforms

The information of Table 4.6 shows that it takes 1.5 nsec for a signal wave to propagate from source-to-destination on a 20 cm wire deposited on an epoxy printed-circuit board. If transmission line effects were an issue on silicon integrated circuits, it would take 0.67 nsec for the signal to reach the end of a 10 cm wire.

WARNING: The characteristic impedance of a wire is a function of the overall interconnect topology. The electro-magnetic fields in complex interconnect structures tend to be irregular, and are strongly influenced by issues such as the current return path. Providing a general answer to the latter problem has so far proven to be illusive, and no closed-formed analytical solutions are typically available. Hence, accurate inductance and characteristic impedance extraction is still an active research topic. For some simplified structures, approximative expressions have been derived. For instance, the characteristic impedances of a triplate strip-line (a wire embedded in between two ground planes) and a semiconductor micro strip-line (wire above a semiconductor substrate) are approximated by Eq. (4.29) and Eq. (4.30), respectively.

$$Z_0(\text{triplate}) \approx 94\Omega \sqrt{\frac{\mu_r}{\varepsilon_r}} \ln\left(\frac{2t + W}{H + W}\right) \tag{4.29}$$

and

$$Z_0(\text{microstrip}) \approx 60\Omega \sqrt{\frac{\mu_r}{0.475\varepsilon_r + 0.67}} \ln\left(\frac{4t}{0.536W + 0.67H}\right) \tag{4.30}$$

Termination

The behavior of the transmission line is strongly influenced by the termination of the line. The termination determines how much of the wave is reflected upon arrival at the wire

end. This is expressed by the *reflection coefficient* ρ that determines the relationship between the voltages and currents of the incident and reflected waveforms.

$$\rho = \frac{V_{refl}}{V_{inc}} = \frac{I_{refl}}{I_{inc}} = \frac{R - Z_0}{R + Z_o} \tag{4.31}$$

where R is the value of the termination resistance. The total voltages and currents at the termination end are the sum of incident and reflected waveforms.

$$\begin{aligned} V &= V_{inc}(1 + \rho) \\ I &= I_{inc}(1 - \rho) \end{aligned} \tag{4.32}$$

Three interesting cases can be distinguished, as illustrated in Figure 4.19. In case (a) the terminating resistance is equal to the characteristic impedance of the line. The termination appears as an infinite extension of the line, and no waveform is reflected. This is also demonstrated by the value of ρ, which equals 0. In case (b), the line termination is

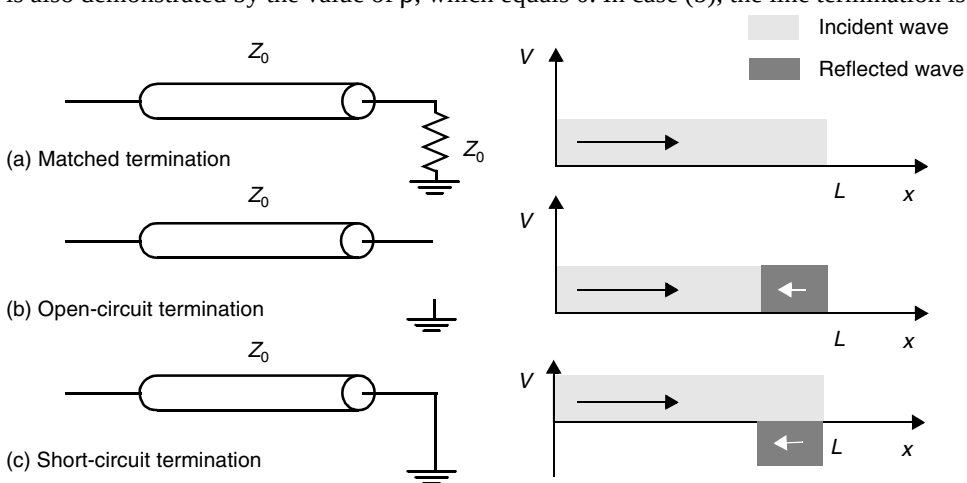


Figure 4.19 Behavior of various transmission line terminations.

an open circuit ($R = \infty$), and $\rho = 1$. The total voltage waveform after reflection is twice the incident one as predicted by Eq. (4.32). Finally, in case (c) where the line termination is a short circuit, $R = 0$, and $\rho = -1$. The total voltage waveform after reflection equals zero.

The transient behavior of a complete transmission line can now be examined. It is influenced by the characteristic impedance of the line, the series impedance of the source Z_S, and the loading impedance Z_L at the destination end, as shown in Figure 4.20.

Consider first the case where the wire is open at the destination end, or $Z_L = \infty$, and $\rho_L = 1$. An incoming wave is completely reflected without phase reversal. Under the assumption that the source impedance is resistive, three possible scenarios are sketched in Figure 4.21: $R_S = 5\ Z_0$, $R_S = Z_0$, and $R_S = 1/5\ Z_0$.

1. **Large source resistance**—$R_S = 5\ Z_0$ (Figure 4.21a)

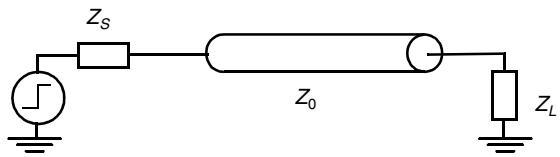


Figure 4.20 Transmission line with terminating impedances.

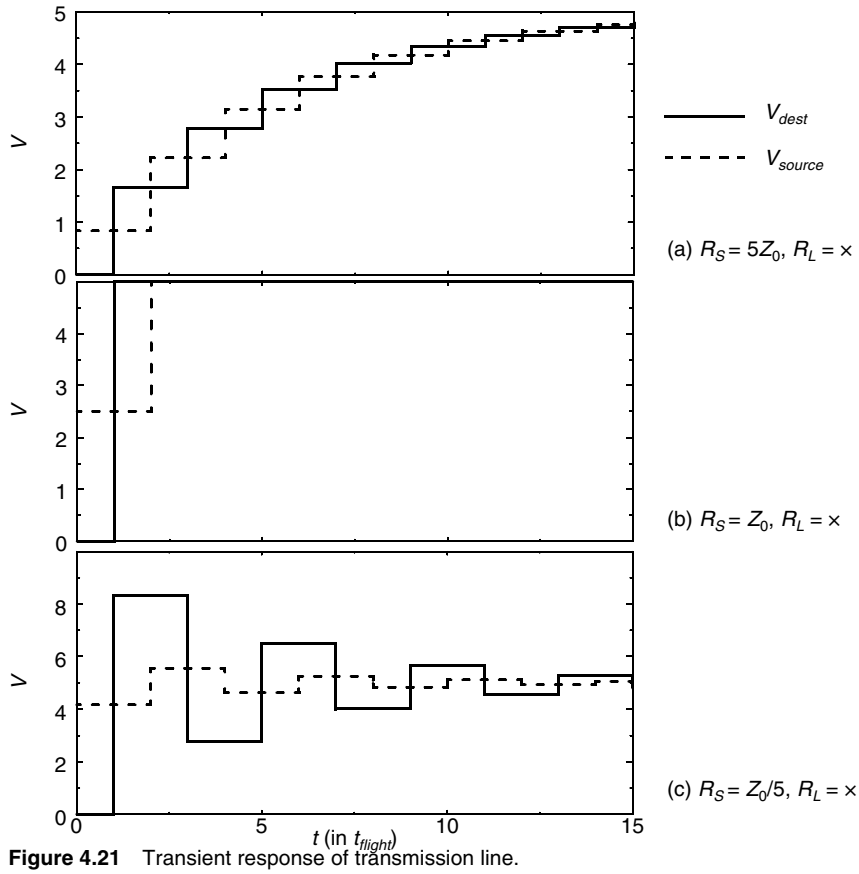


Figure 4.21 Transient response of transmission line.

Only a small fraction of the incoming signal V_{in} is injected into the transmission line. The amount injected is determined by the resistive divider formed by the source resistance and the characteristic impedance Z_0.

$$V_{source} = (Z_0 / (Z_0 + R_S))\, V_{in} = 1/6 \times 5 \text{ V} = 0.83 \text{ V} \tag{4.33}$$

This signal reaches the end of the line after L/v sec, where L stands for the length of the wire and is fully reflected, which effectively doubles the amplitude of the wave (V_{dest} = 1.67 V). The time it takes for the wave to propagate from one end of the wire to

the other is called the *time-of-flight*, $t_{flight} = L/v$. Approximately the same happens when the wave reaches the source node again. The incident waveform is reflected with an amplitude determined by the source reflection coefficient, which equals 2/3 for this particular case.

$$\rho_S = \frac{5Z_0 - Z_0}{5Z_0 + Z_0} = \frac{2}{3} \tag{4.34}$$

The voltage amplitude at source and destination nodes gradually reaches its final value of V_{in}. The overall rise time is, however, many times L/v.

When multiple reflections are present, as in the above case, keeping track of waves on the line and total voltage levels rapidly becomes cumbersome. Therefore a graphical construction called the *lattice diagram* is often used to keep track of the data (Figure 4.22). The diagram contains the values of the voltages at the source and destination ends, as well as the values of the incident and reflected wave forms. The line voltage at a termination point equals the sum of the previous voltage, the incident, and reflected waves.

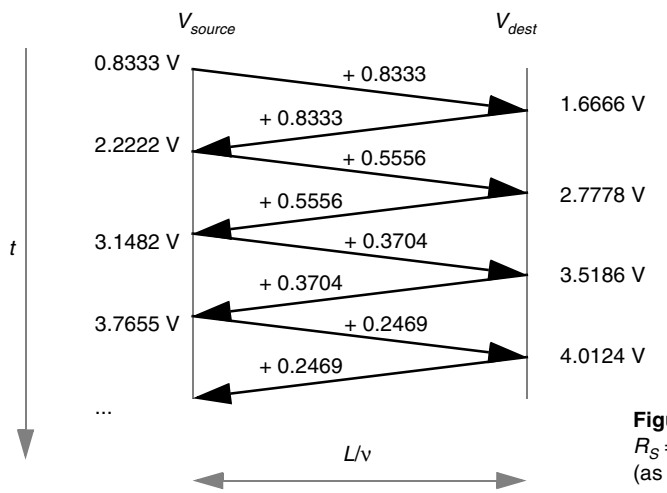


Figure 4.22 Lattice diagram for $R_S = 5\ Z_0$ and $R_L = \infty$. $V_{step} = 5$ V, (as in Figure 4.21a).

2. **Small source resistance**—$R_S = Z_0/5$ (Figure 4.21c)

A large portion of the input is injected in the line. Its value is doubled at the destination end, which causes a severe overshoot. At the source end, the phase of the signal is reversed ($\rho_S = -2/3$). The signal bounces back and forth and exhibits severe ringing. It takes multiple L/v before it settles.

3. **Matched source resistance**—$R_S = Z_0$ (Figure 4.21b)

Half of the input signal is injected at the source. The reflection at the destination end doubles the signal, so that the final value is reached immediately. It is obvious that this is the most effective case.

Note that the above analysis is an ideal one, as it is assumed that the input signal has a zero rise time. In real conditions the signals are substantially smoother, as demonstrated in the simulated response of Figure 4.23 (for $R_S = Z_0/5$ and $t_r = t_{flight}$).

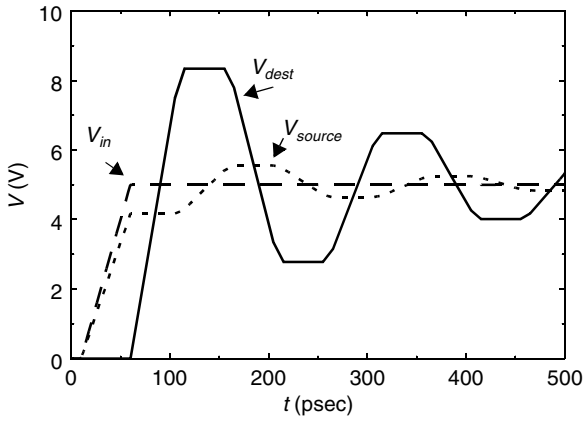


Figure 4.23 Simulated transient response of lossless transmission line for finite input rise times ($R_S = Z_0/5$, $t_r = t_{flight}$).

Problem 4.1 Transmission Line Response

Derive the lattice diagram of the above transmission line for $R_S = Z_0/5$, $R_L = \infty$, and $V_{step} = 5$ V. Also try the reverse picture—assume that the series resistance of the source equals zero, and consider different load impedances.

Example 4.11 Capacitive Termination

Loads in MOS digital circuits tend to be of a capacitive nature. One might wonder how this influences the transmission line behavior and when the load capacitance should be taken into account.

The characteristic impedance of the transmission line determines the current that can be supplied to charge capacitive load C_L. From the load's point of the view, the line behaves as a resistance with value Z_0. The transient response at the capacitor node, therefore, displays a time constant Z_0C_L. This is illustrated in Figure 4.24, which shows the simulated transient response of a series-terminated transmission line with a characteristic impedance of 50 Ω loaded by a capacitance of 2 pF. The response shows how the output rises to its final value with a time-constant of 100 psec (= 50 Ω × 2 pF) after a delay equal to the time-of-flight of the line.

This asymptotic response causes some interesting artifacts. After 2 t_{flight}, an unexpected voltage dip occurs at the source node that can be explained as follows. Upon reaching the destination node, the incident wave is reflected. This reflected wave also approaches its final value asymptotically. Since V_{dest} equals 0 initially instead of the expected jump to 5 V, the reflection equals –2.5 V rather than the expected 2.5 V. This forces the transmission line temporarily to 0 V, as shown in the simulation. This effect gradually disappears as the output node converges to its final value.

The propagation delay of the line equals the sum of the time-of-flight of the line (= 50 psec) and the time it takes to charge the capacitance (= 0.69 Z_0 C_L = 69 psec). This is exactly what the simulation yields. In general, we can say that the capacitive load should only be considered in the analysis when its value is comparable to or larger than the total capacitance of the transmission line [Bakoglu90].

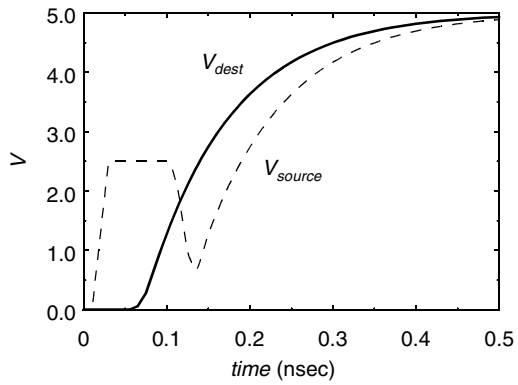


Figure 4.24 Capacitively terminated transmission line: R_S = 50 Ω, $R_L = \infty$, C_L = 2 pF, Z_0 = 50 Ω, t_{flight} = 50 psec.

Lossy Transmission Line

While board and module wires are thick and wide enough to be treated as lossless transmission lines, the same is not entirely true for on-chip interconnect where the resistance of the wire is an important factor. The lossy transmission-line model should be applied instead. Going into detail about the behavior of a lossy line would lead us to far astray. We therefore only discuss the effects of resistive loss on the transmission line behavior in a qualitative fashion.

The response of a lossy *RLC line* to a unit step combines wave propagation with a diffusive component. This is demonstrated in Figure 4.25, which plots the response of the *RLC* transmission line as a function of distance from the source. The step input still propagates as a wave through the line. However, the amplitude of this traveling wave is attenuated along the line:

$$\frac{V_{step}(x)}{V_{step}(0)} = e^{-\frac{r}{2Z_0}x} \tag{4.35}$$

The arrival of the wave is followed by a diffusive relaxation to the steady-state value at point *x*. The farther it is from the source, the more the response resembles the behavior of a distributed *RC* line. In fact, the resistive effect becomes dominant, and the line behaves as a distributed *RC* line when R (= rL, the total resistance of the line) >> 2 Z_0. When $R = 5\ Z_0$, only 8% of the original step reaches the end of the line. At that point, the line is more appropriately modeled as a distributed *rc* line.

Be aware that the actual wires on chips, boards, or substrates behave in a far more complex way than predicted by the above analysis. For instance, branches on wires, often called *transmission line taps,* cause extra reflections and can affect both signal shape and delay. Since the analysis of these effects is very involved, the only meaningful approach is to use computer analysis and simulation techniques. For a more extensive discussion of these effects, we would like to refer the reader to [Bakoglu90] and [Dally98].

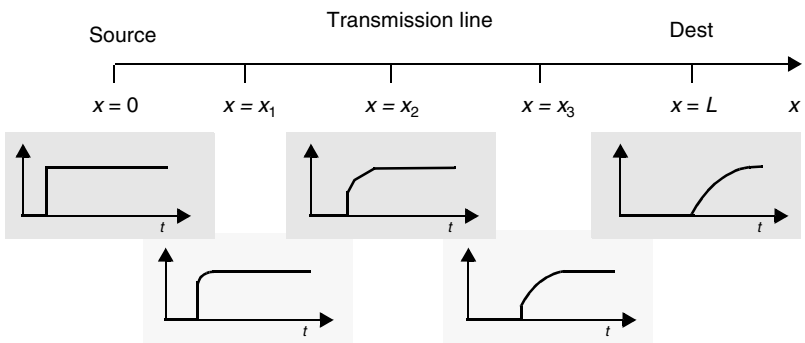


Figure 4.25 Step response of lossy transmission line.

Design Rules of Thumb

Once again, we have to ask ourselves the question when it is appropriate to consider transmission line effects. From the above discussion, we can derive two important constraints:

- **Transmission line effects should be considered when the rise or fall time of the input signal (t_r, t_f) is smaller than the time-of-flight of the transmission line (t_{flight}).**

This leads to the following rule of thumb, which determines when transmission line effects should be considered:

$$t_r(t_f) < 2.5 t_{flight} = 2.5\frac{L}{n} \tag{4.36}$$

For on-chip wires with a maximum length of 1 cm, one should only worry about transmission line effects when t_r < 150 psec. At the board level, where wires can reach a length of up to 50 cm, we should account for the delay of the transmission line when t_r < 8 nsec. This condition is easily achieved with state-of-the-art processes and packaging technologies. Ignoring the inductive component of the propagation delay can easily result in overly optimistic delay predictions.

- **Transmission line effects should only be considered when the total resistance of the wire is limited:**

$$R < 5Z_0 \tag{4.37}$$

If this is not the case, the distributed RC model is more appropriate.
Both constraints can be summarized in the following set of bounds on the wire length:

$$\frac{t_r}{2.5}\frac{1}{\sqrt{lc}} < L < \frac{5}{r}\sqrt{\frac{l}{c}} \tag{4.38}$$

- **The transmission line is considered lossless when the total resistance is substantially smaller than the characteristic impedance, or**

$$R < \frac{Z_0}{2} \tag{4.39}$$

Example 4.12 When to Consider Transmission Line Effects

Consider again our Al1 wire. Using the data from Example 4.4 and Eq. (4.28), we can approximate the value of Z_0 for various wire widths:

W = 0.1 μm: c = 92 aF/μm; Z_0 = 74 Ω
W = 1.0 μm: c = 110 aF/μm; Z_0 = 60 Ω
W = 10 μm: c = 380 aF/μm; Z_0 = 17 Ω

For a wire with a width of 1μm, we can derive the maximum length of the wire for which we should consider transmission line effects using Eq. (4.37):

$$L_{max} = \frac{5Z_0}{r} = \frac{5 \times 60\Omega}{0.075\Omega/\mu\text{m}} = 4000\mu\text{m}$$

From Eq. (4.36), we find a corresponding maximum rise (or fall) time of the input signal equal to

$$t_{rmax} = 2.5 \times (4000\ \mu\text{m})/(15\ \text{cm/nsec}) = 67\ \text{psec}$$

This is hard to accomplish in current technologies. For these wires, a lumped capacitance model is more appropriate. Transmission line effects are more plausible in wider wires. For a 10 μm wide wire, we find a maximum length of 11.3 mm, which corresponds to a maximum rise time of 188 psec.

Assume now a Copper wire, implemented on level 5, with a characteristic impedance of 200 Ω and a resistance of 0.025 Ω/μm. The resulting maximum wire length equals 40 mm. Rise times smaller than 670 psec will cause transmission line effects to occur.

Be aware however that the values for Z_0, derived in this example, are only approximations. In actual designs, more complex expressions or empirical data should be used.

Example 4.13 Simulation of Transmission Line Effects

Show SPICE simulation

4.5 SPICE Wire Models

In previous sections, we have discussed the various interconnect parasitics, and introduced simple models for each of them. Yet, the full and precise impact of these effects can only be found through detailed simulation. In this section, we introduce the models that SPICE provides for the capacitive, resistive, and inductive parasitics.

4.5.1 Distributed *rc* Lines in SPICE

Because of the importance of the distributed *rc*-line in today's design, most circuit simulators have built-in distributed *rc*-models of high accuracy. For instance, the Berkeley SPICE3 simulator supports a uniform-distributed *rc*-line model (URC). This model approximates the *rc*-line as a network of lumped RC segments with internally generated nodes. Parameters include the length of the wire *L* and (optionally) the number of segments used in the model.

Example 4.14 SPICE3 URC Model

A typical example of a SPICE3 instantiation of a distributed rc-line is shown below. N1 and N2 represent the terminal nodes of the line, while N3 is the node the capacitances are connected to. RPERL and CPERL stand for the resistance and capacitance per meter.

```
U1 N1=1 N2=2 N3=0 URCMOD L=50m N=6
.MODEL URCMOD URC(RPERL=75K CPERL=100pF)
```

If your simulator does not support a distributed *rc*-model, or if the computational complexity of these models slows down your simulation too much, you can construct a simple yet accurate model yourself by approximating the distributed *rc* by a lumped *RC* network with a limited number of elements. Figure 4.26 shows some of these approximations ordered along increasing precision and complexity. The accuracy of the model is determined by the number of stages. For instance, the error of the $\pi 3$ model is less than 3%, which is generally sufficient.

4.5.2 Transmission Line Models in SPICE

SPICE supports a lossless transmission line model. The line characteristics are defined by the characteristic impedance Z_0, while the length of the line can be defined in either of two forms. A first approach is to directly define the *transmission delay TD*, which is equivalent to the time-of-flight. Alternatively, a frequency *F* may be given together with *NL*, the dimensionless, normalized electrical length of the transmission line, which is measured with respect to the wavelength in the line at the frequency *F*. The following relation is valid.

$$NL = F \cdot TD \tag{4.40}$$

No lossy transmission line model is currently provided. When necessary, loss can be added by breaking up a long transmission line into shorter sections and adding a small series resistance in each section to model the transmission line loss. Be careful when using this approximation. First of all, the accuracy is still limited. Secondly, the simulation speed might be severely effected, since SPICE chooses a time step that is less than or equal to half of the value of *TD*. For small transmission lines, this time step might be much smaller than what is needed for transistor analysis.

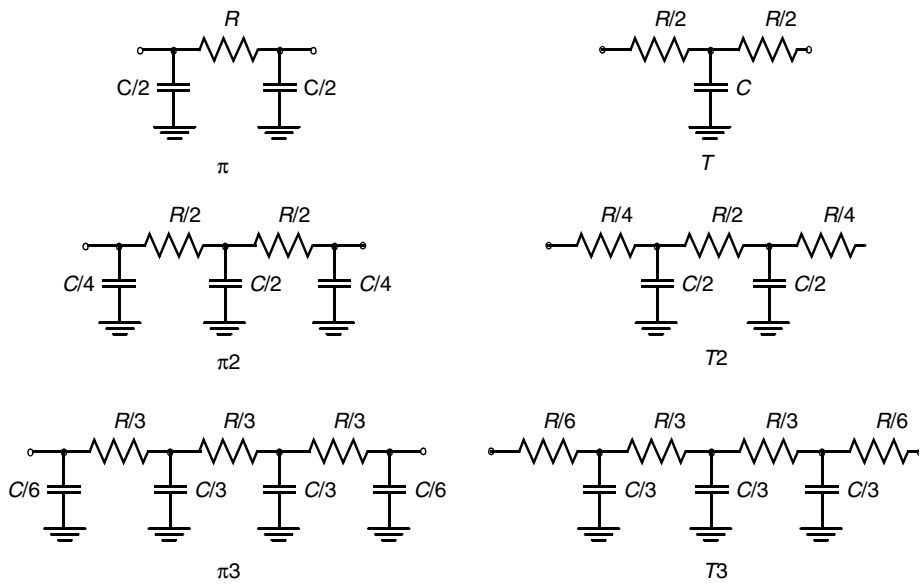


Figure 4.26 Simulation models for distributed *RC* line.

4.6 Perspective: A Look into the Future

Similar to the approach we followed for the MOS transistor, it is worthwhile to explore how the wire parameters will evolve with further scaling of the technology. As transistor dimensions are reduced, the interconnect dimensions must also be reduced to take full advantage of the scaling process.

A straightforward approach is to scale all dimensions of the wire by the same factor *S* as the transistors (*ideal scaling*). This might not be possible for at least one dimension of the wire, being the length. It can be surmised that the length of *local interconnections* — wires that connect closely grouped transistors — scales in the same way as these transistors. On the other hand, *global interconnections*, that provide the connectivity between large modules and the input-output circuitry, display a different scaling behavior. Examples of such wires are clock signals, and data and instruction buses. Figure 4.27 contains a histogram showing the distribution of the wire lengths in an actual microprocessor design, containing approximately 90,000 gate) [Davis98]. While most of the wires tend to be only a couple of gate pitches long, a substantial number of them are much longer and can reach lengths up to 500 gate pitches.

The average length of these long wires is proportional to the die size (or complexity) of the circuit. An interesting trend is that while transistor dimensions have continued to shrink over the last decades, the chip sizes have gradually increased. In fact, the size of the typical die (which is the square root of the die area) is increasing by 6% per year, doubling

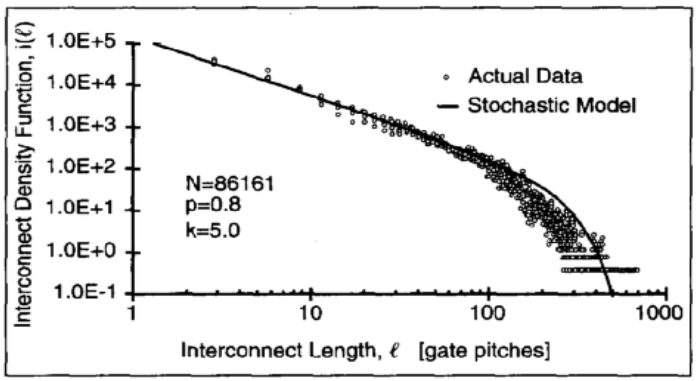


Figure 4.27 Distribution of wire lengths in an advanced microprocessor as a function of the gate pitch.

about every decade. Chips have scaled from 2 mm × 2 mm in the early 1960s to approximately 2 cm × 2 cm in 2000. They are projected to reach 4 cm on the side by 2010!

This argues that when studying the scaling behavior of the wire length, we have to differentiate between local and global wires. In our subsequent analysis, we will therefore consider three models: local wires ($S_L = S > 1$), constant length wires ($S_L = 1$), and global wires ($S_L = S_C < 1$).

Assume now that all other wire dimensions of the interconnect structure (W, H, t) scale with the technology factor S. This leads to the scaling behavior illustrated in Table 4.8. Be aware that this is only a first-order analysis, intended to look at overall trends. Effects such a fringing capacitance are ignored, and breakthroughs in semiconductor technology such as new interconnect and dielectric materials are also not considered.

Table 4.8 Ideal Scaling of Wire Properties

Parameter	Relation	Local Wire	Constant Length	Global Wire
W, H, t		$1/S$	$1/S$	$1/S$
L		$1/S$	1	$1/S_C$
C	LW/t	$1/S$	1	$1/S_C$
R	L/WH	S	S^2	S^2/S_C
CR	L^2/Ht	1	S^2	S^2/S_C^2

The eye-catching conclusion of this exercise is that scaling of the technology does not reduce wire delay (as personified by the RC time-constant). A constant delay is predicted for local wires, while the delay of the global wires goes up with 50% per year (for $S = 1.15$ and $S_C = 0.94$). This is in great contrast with the gate delay, which reduces from year to year. This explains why wire delays are starting to play a predominant role in today's digital integrated circuit design.

The ideal scaling approach clearly has problems, as it causes a rapid increase in wire resistance. This explains why other interconnect scaling techniques are attractive. One

option is to scale the wire thickness at a different rate. The "constant resistance" model of Table 4.9 explores the impact of not scaling the wire thickness at all. While this approach seemingly has a positive impact on the performance, it causes the fringing and inter-wire capacitance components to come to the foreground. We therefore introduce an extra capacitance scaling factor ε_c (> 1), that captures the increasingly horizontal nature of the capacitance when wire widths and pitches are shrunk while the height is kept constant.

Table 4.9 "Constant Resistance" Scaling of Wire Properties

Parameter	Relation	Local Wire	Constant Length	Global Wire
W, t		$1/S$	$1/S$	$1/S$
H		1	1	1
L		$1/S$	1	$1/S_C$
C	e_cLW/t	e_c/S	e_c	e_c/S_C
R	L/WH	1	S	S/S_C
CR	L^2/Ht	e_c/S	e_cS	e_cS/S_C^2

This scaling scenario offers a slightly more optimistic perspective, assuming of course that $\varepsilon_c < S$. Yet, delay is bound to increase substantially for intermediate and long wires, independent of the scaling scenario. To keep these delays from becoming excessive, interconnect technology has to be drastically improved. One option is to use better interconnect (Cu) and insulation materials (polymers and air). The other option is to differentiate between local and global wires. In the former, density and low-capacitance are crucial, while keeping the resistance under control is crucial in the latter. To address these conflicting demands, modern interconnect topologies combine a dense and thin wiring grid at the lower metal layers with fat, widely spaced wires at the higher levels, as is illustrated in Figure 4.28. Even with these advances, it is obvious that interconnect will play a dominant role in both high-performance and low-energy circuits for years to come.

4.7 Summary

This chapter has presented a careful and in-depth analysis of the role and the behavior of the interconnect wire in modern semiconductor technology. The main goal is to identify the dominant parameters that set the values of the wire parasitics (being capacitance, resistance, and inductance), and to present adequate wire models that will aid us in the further analysis and optimization of complex digital circuits.

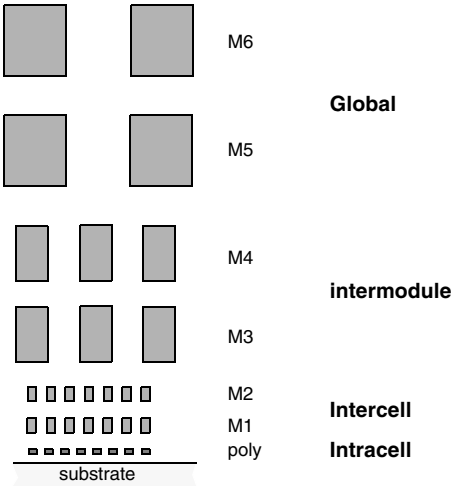


Figure 4.28 Interconnect hierarchy of 0.25 μm CMOS process, drawn to scale.

4.8 To Probe Further

Interconnect and its modeling is a hotly debated topic, that receives major attention in journals and conferences. A number of textbooks and reprint volumes have been published. [Bakoglu90], [Tewksbury94], and [Dally98] present an in-depth coverage of interconnect issues, and are a valuable resource for further browsing.

REFERENCES

[Bakoglu90] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990.

[Dally98] B. Dally, *Digital Systems Engineering*, Cambridge University Press, 1998.

[Davis98] J. Davis and J. Meindl, "Is Interconnect the Weak Link?," *IEEE Circuits and Systems Magazine*, pp. 30-36, March 1998.

[Elmore48] E. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, pp. 55–63, January 1948.

[Etter93] D. Etter, "Engineering Problem Solving with Matlab," Prentice Hall, 1993.

[Horowitz83] M. Horowitz, "Timing Models for MOS Circuits," Ph.D. diss., Stanford University, 1983.

[Lin84] T.. Lin and C. Mead, "Signal delay in general RC networks," IEEE Transactions on Computer-Aided Design, Vol. 3 No 4, pp. 321--349, 1984.

[Rubinstein83] J. Rubinstein, P. Penfield, and M. Horowitz, "Signal Delay in *RC* Networks," *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.

[Tewksbury94] S. Tewksbury, ed., *Microelectronics System Interconnections—Performance and Modeling*, IEEE Press, 1994.

[Vdmeijs84] N. Van De Meijs and J. Fokkema, "VLSI Circuit Reconstruction from Mask Topology," *Integration*, vol. 2, no. 2, pp. 85–119, 1984.

Chapter 4

Problems

1. [M, None, 4.x] Figure 0.1 shows a clock-distribution network. Each segment of the clock network (between the nodes) is 5 mm long, 3 μm wide, and is implemented in polysilicon. At each of the terminal nodes (such as *R*) resides a load capacitance of 100 fF.

a. Determine the average current of the clock driver, given a voltage swing on the clock lines of 5 V and a maximum delay of 5 nsec between clock source and destination node *R*. For this part, you may ignore the resistance and inductance of the network

b. Unfortunately the resistance of the polysilicon cannot be ignored. Assume that each straight segment of the network can be modeled as a Π-network. Draw the equivalent circuit and annotate the values of resistors and capacitors.

c. Determine the dominant time-constant of the clock response at node *R*.

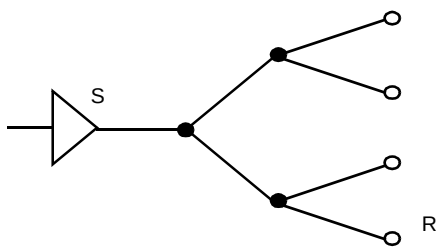


Figure 0.1 Clock-distribution network.

2. [C, SPICE, 4.x] You are designing a clock distribution network in which it is critical to minimize skew between local clocks (*CLK*1, *CLK*2, and *CLK*3). You have extracted the *RC* network of Figure 0.2, which models the routing parasitics of your clock line. Initially, you notice that the path to *CLK*3 is shorter than to *CLK*1 or *CLK*2. In order to compensate for this imbalance, you insert a transmission gate in the path of *CLK*3 to eliminate the skew.

a. Write expressions for the time-constants associated with nodes *CLK*1, *CLK*2 and *CLK*3. Assume the transmission gate can be modeled as a resistance R_3.

b. If $R_1 = R_2 = R_4 = R_5 = R$ and $C_1 = C_2 = C_3 = C_4 = C_5 = C$, what value of R_3 is required to balance the delays to *CLK*1, *CLK*2, and *CLK*3?

c. For $R = 750\Omega$ and $C = 200$fF, what (W/L)'s are required in the transmission gate to eliminate skew? Determine the value of the propagation delay.

d. Simulate the network using SPICE, and compare the obtained results with the manually obtained numbers.

3. [M, None, 4.x]Consider a CMOS inverter followed by a wire of length *L*. Assume that in the reference design, inverter and wire contribute equally to the total propagation delay t_{pref}. You may assume that the transistors are velocity-saturated. The wire is scaled in line with the **ideal wire scaling model**. Assume initially that the wire is **a local wire**.

a. Determine the new (total) propagation delay as a a function of t_{pref}, assuming that technology and supply voltage scale with a factor 2. Consider only first-order effects.

b. Perform the same analysis, assuming now that the wire scales a **global wire**, and the wire length scales inversely proportional to the technology.

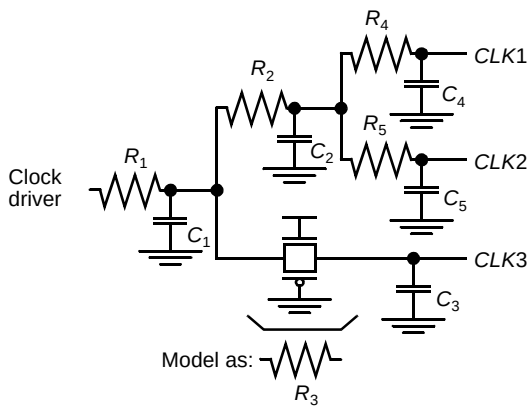


Figure 0.2 *RC* clock-distribution network.

c. Repeat b, but assume now that the wire is scaled along the constant resistance model. You may ignore the effect of the fringing capacitance.

d. Repeat b, but assume that the new technology uses a better wiring material that reduces the resistivity by half, and a dielectric with a 25% smaller permittivity.

e. Discuss the energy dissipation of part a. as a function of the energy dissipation of the original design E_{ref}.

f. Determine for each of the statements below if it is true, false, or undefined, and explain in one line your answer.

- When driving a small fan-out, increasing the driver transistor sizes raises the short-circuit power dissipation.
- Reducing the supply voltage, while keeping the threshold voltage constant decreases the short-circuit power dissipation.
- Moving to Copper wires on a chip will enable us to build faster adders.
- Making a wire wider helps to reduce its RC delay.
- Going to dielectrics with a lower permittivity will make RC wire delay more important.

4. [M, None, 4.x] A two-stage buffer is used to drive a metal wire of 1 cm. The first inverter is of minimum size with an input capacitance Ci=10 fF and an internal propagation delay t_{p0}=50 ps and load dependent delay of 5ps/fF. The width of the metal wire is 3.6 µm. The sheet resistance of the metal is 0.08 Ω/ , the capacitance value is 0.03 fF/µm2 and the fringing field capacitance is 0.04fF/µm.

a. What is the propagation delay of the metal wire?

b. Compute the optimal size of the second inverter. What is the minimum delay through the buffer?

c. If the input to the first inverter has 25% chance of making a 0-to-1 transition, and the whole chip is running at 20MHz with a 2.5 supply voltage, then what's the power consumed by the metal wire?

5. [M, None, 4.x]To connect a processor to an external memory an off -chip connection is necessary. The copper wire on the board is 15 cm long and acts as a transmission line with a characteristic impedance of 100Ω(See Figure 0.3). The memory input pins present a very high impedance which can be considered infinite. The bus driver is a CMOS inverter consisting of very large devices: (50/0.25) for the NMOS and (150/0.25) for the PMOS, where all sizes are

in µm. The minimum size device, (0.25/0.25) for NMOS and (0.75/0.25) for PMOS, has the on resistance 35 kΩ.

a. Determine the time it takes for a change in the signal to propagate from source to destination (time of flight). The wire inductance per unit length equals $75*10^{-8}$ H/m.

b. Determine how long it will take the output signal to stay within 10% of its final value. You can model the driver as a voltage source with the driving device acting as a series resistance. Assume a supply and step voltage of 2.5V. Hint: draw the lattice diagram for the transmission line.

c. Resize the dimensions of the driver to minimize the total delay.

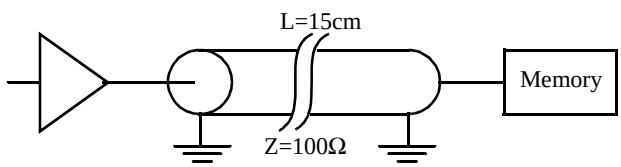


Figure 0.3 The driver, the connecting copper wire and the memory block being accessed.

6. [M, None, 4.x] A two stage buffer is used to drive a metal wire of 1 cm. The first inverter is a minimum size with an input capacitance C_i=10 fF and a propagation delay t_{p0}=175 ps when loaded with an identical gate. The width of the metal wire is 3.6 µm. The sheet resistance of the metal is 0.08 Ω/ , the capacitance value is 0.03 fF/µm2 and the fringing field capacitance is 0.04 fF/µm.

a. What is the propagation delay of the metal wire?

b. Compute the optimal size of the second inverter. What is the minimum delay through the buffer?

7. [M, None, 4.x] For the RC tree given in Figure 0.4 calculate the Elmore delay from node A to node B using the values for the resistors and capacitors given in the below in Table 0.1.

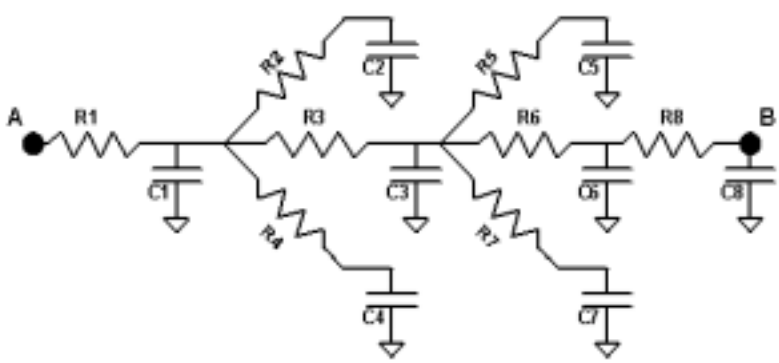


Figure 0.4 RC tree for calculating the delay

Table 0.1 Values of the components in the RC tree of Figure 0.4

Resistor	Value(Ω)	Capacitor	Value(fF)
R1	0.25	C1	250
R2	0.25	C2	750
R3	0.50	C3	250
R4	100	C4	250
R5	0.25	C5	1000
R6	1.00	C6	250
R7	0.75	C7	500
R8	1000	C8	250

8. [M, SPICE, 4.x] In this problem the various wire models and their respective accuracies will be studied.

a. Compute the 0%-50% delay of a 500um x 0.5um wire with resistance of 0.08 Ω/ , with area capacitance of 30aF/um2, and fringing capacitance of 40aF/um. Assume the driver has a 100Ω resistance and negligible output capacitance.

- Using a lumped model for the wire.
- Using a PI model for the wire, and the Elmore equations to find tau. (see Chapter 4, figure 4.26).
- Using the distributed RC line equations from Chapter 4, section 4.4.4.

b. Compare your results in part a. using spice (be sure to include the source resistance). For each simulation, measure the 0%-50% time for the output

- First, simulate a step input to a lumped R-C circuit.
- Next, simulate a step input to your wire as a PI model.
- Unfortunately, our version of SPICE does not support the distributed RC model as described in your book (Chapter 4, section 4.5.1). Instead, simulate a step input to your wire using a PI3 distributed RC model.

9. [M, None, 4.x] A standard CMOS inverter drives an aluminum wire on the first metal layer. Assume Rn=4kΩ, Rp=6kΩ. Also, assume that the output capacitance of the inverter is negligible in comparison with the wire capacitance. The wire is .5um wide, and the resistivity is 0.08 Ω/ ..

a. What is the "critical length" of the wire?

b. What is the equivalent capacitance of a wire of this length? (For your capacitance calculations, use Table 4.2 of your book , assume there's field oxide underneath and nothing above the aluminum wire)

10. [M, None, 4.x] A 10cm long lossless transmission line on a PC board (relative dielectric constant = 9, relative permeability = 1) with characteristic impedance of 50Ω is driven by a 2.5V pulse coming from a source with 150Ω resistance.

a. If the load resistance is infinite, determine the time it takes for a change at the source to reach the load (time of flight).

Now a 200Ω load is attached at the end of the transmission line.

b. What is the voltage at the load at t = 3ns?

c. Draw lattice diagram and sketch the voltage at the load as a function of time. Determine how long does it take for the output to be within 1 percent of its final value.

11. [C, SPICE, 4.x] Assume V_{DD}=1.5V. Also, use short-channel transistor models forhand analysis.

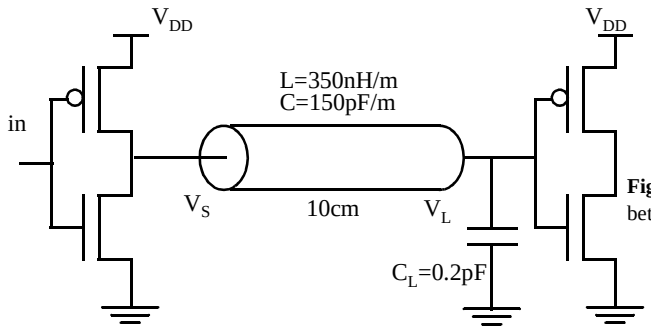


Figure 0.5 Transmission line between two inverters

a. The Figure 0.5 shows an output driver feeding a 0.2 pF effective fan-out of CMOS gates through a transmission line. Size the two transistors of the driver to optimize the delay. Sketch waveforms of V_S and V_L, assuming a square wave input. Label critical voltages and times.

b. Size down the transistors by m times (m is to be treated as a parameter). Derive a first order expression for the time it takes for V_L to settle down within 10% of its final voltage level.Compare the obtained result with the case where no inductance is associated with the wire.Please draw the waveforms of V_L for both cases, and comment.

c. Use the transistors as in part a). Suppose C_L is changed to 20pF. Sketch waveforms of V_S and V_L, assuming a square wave input. Label critical voltages and instants.

d. Assume now that the transmission line is lossy. Perform Hspice simulation for three cases: R=100 Ω/cm; R=2.5 Ω/cm; R=0.5 Ω/cm. Get the waveforms of V_S, V_L and the middle point of the line. Discuss the results.

12. [M, None, 4.x] Consider an isolated 2mm long and 1μm wide M1(Metal1)wire over a silicon substrate driven by an inverter that has zero resistance and parasitic output capccitance. How will the wire delay change for the following cases? Explain your reasoning in each case.

a. If the wire width is doubled.

b. If the wire length is halved.

c. If the wire thickness is doubled.

d. If thickness of the oxide between the M1 and the substrate is doubled.

13. [E, None, 4.x] In an ideal scaling model, where all dimensions and voltages scale with a factor of S >1 :

a. How does the delay of an inverter scale?

b. If a chip is scaled from one technology to another where all wire dimensions,including the vertical one and spacing, scale with a factor of S, how does the wire delayscale? How does the overall operating frequency of a chip scale?

c. Repeat b) for the case where everything scales, except the vertical dimension of wires (it stays constant).

CHAPTER 5

THE CMOS INVERTER

Quantification of integrity, performance, and energy metrics of an inverter
Optimization of an inverter design

5.1 Introduction

5.2 The Static CMOS Inverter — An Intuitive Perspective

5.3 Evaluating the Robustness of the CMOS Inverter: The Static Behavior

- 5.3.1 Switching Threshold
- 5.3.2 Noise Margins
- 5.3.3 Robustness Revisited

5.4 Performance of CMOS Inverter: The Dynamic Behavior

- 5.4.1 Computing the Capacitances
- 5.4.2 Propagation Delay: First-Order Analysis
- 5.4.3 Propagation Delay from a Design Perspective

5.5 Power, Energy, and Energy-Delay

- 5.5.1 Dynamic Power Consumption
- 5.5.2 Static Consumption
- 5.5.3 Putting It All Together
- 5.5.4 Analyzing Power Consumption Using SPICE

5.6 Perspective: Technology Scaling and its Impact on the Inverter Metrics

5.1 Introduction

The inverter is truly the nucleus of all digital designs. Once its operation and properties are clearly understood, designing more intricate structures such as NAND gates, adders, multipliers, and microprocessors is greatly simplified. The electrical behavior of these complex circuits can be almost completely derived by extrapolating the results obtained for inverters. The analysis of inverters can be extended to explain the behavior of more complex gates such as NAND, NOR, or XOR, which in turn form the building blocks for modules such as multipliers and processors.

In this chapter, we focus on one single incarnation of the inverter gate, being the static CMOS inverter — or the CMOS inverter, in short. This is certainly the most popular at present, and therefore deserves our special attention. We analyze the gate with respect to the different design metrics that were outlined in Chapter 1:

- *cost*, expressed by the complexity and area
- *integrity and robustness*, expressed by the static (or steady-state) behavior
- *performance*, determined by the dynamic (or transient) response
- *energy efficiency*, set by the energy and power consumption

From this analysis arises a model of the gate that will help us to identify the parameters of the gate and to choose their values so that the resulting design meets desired specifications. While each of these parameters can be easily quantified for a given technology, we also discuss how they are affected by *scaling of the technology*.

While this Chapter focuses uniquely on the CMOS inverter, we will see in the following Chapter that the same methodology also applies to other gate topologies.

5.2 The Static CMOS Inverter — An Intuitive Perspective

Figure 5.1 shows the circuit diagram of a static CMOS inverter. Its operation is readily understood with the aid of the simple switch model of the MOS transistor, introduced in Chapter 3 (Figure 3.25): the transistor is nothing more than a switch with an infinite off-resistance (for $|V_{GS}| < |V_T|$), and a finite on-resistance (for $|V_{GS}| > |V_T|$). This leads to the

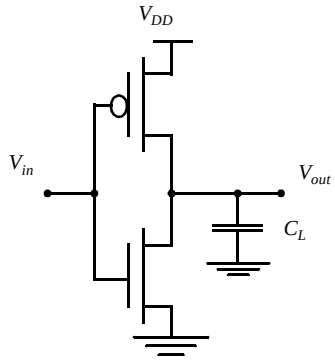
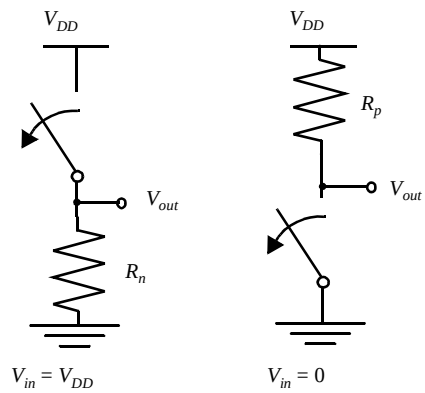


Figure 5.1 Static CMOS inverter. V_{DD} stands for the supply voltage.

following interpretation of the inverter. When V_{in} is high and equal to V_{DD}, the NMOS transistor is on, while the PMOS is off. This yields the equivalent circuit of Figure 5.2a. A direct path exists between V_{out} and the ground node, resulting in a steady-state value of 0 V. On the other hand, when the input voltage is low (0 V), NMOS and PMOS transistors are off and on, respectively. The equivalent circuit of Figure 5.2b shows that a path exists between V_{DD} and V_{out}, yielding a high output voltage. The gate clearly functions as an inverter.



(a) Model for high input (b) Model for low input

Figure 5.2 Switch models of CMOS inverter.

A number of other important properties of static CMOS can be derived from this switch-level view:

- The high and low output levels equal V_{DD} and *GND*, respectively; in other words, the voltage swing is equal to the supply voltage. This results in high noise margins.
- The logic levels are not dependent upon the relative device sizes, so that the transistors can be minimum size. Gates with this property are called *ratioless*. This is in contrast with *ratioed logic*, where logic levels are determined by the relative dimensions of the composing transistors.
- In steady state, there always exists a path with finite resistance between the output and either V_{DD} or *GND*. A well-designed CMOS inverter, therefore, has a *low output impedance*, which makes it less sensitive to noise and disturbances. Typical values of the output resistance are in kΩ range.
- The *input resistance* of the CMOS inverter is extremely high, as the gate of an MOS transistor is a virtually perfect insulator and draws no dc input current. Since the input node of the inverter only connects to transistor gates, the steady-state input current is nearly zero. A single inverter can theoretically drive an infinite number of gates (or have an infinite fan-out) and still be functionally operational; however, increasing the fan-out also increases the propagation delay, as will become clear below. So, although fan-out does not have any effect on the steady-state behavior, it degrades the transient response.

- No direct path exists between the supply and ground rails under steady-state operating conditions (this is, when the input and outputs remain constant). The absence of current flow (ignoring leakage currents) means that the gate does not consume any static power.

SIDELINE: The above observation, while seemingly obvious, is of crucial importance, and is one of the primary reasons CMOS is the digital technology of choice at present. The situation was very different in the 1970s and early 1980s. All early microprocessors, such as the Intel 4004, were implemented in a pure NMOS technology. The lack of complementary devices (such as the NMOS and PMOS transistor) in such a technology makes the realization of inverters with zero static power non-trivial. The resulting static power consumption puts a firm upper bound on the number of gates that can be integrated on a single die; hence the forced move to CMOS in the 1980s, when scaling of the technology allowed for higher integration densities.

The nature and the form of the voltage-transfer characteristic (VTC) can be graphically deduced by superimposing the current characteristics of the NMOS and the PMOS devices. Such a graphical construction is traditionally called *a load-line plot.* It requires that the *I-V* curves of the NMOS and PMOS devices are transformed onto a common coordinate set. We have selected the input voltage V_{in}, the output voltage V_{out} and the NMOS drain current I_{DN} as the variables of choice. The PMOS *I-V* relations can be translated into this variable space by the following relations (the subscripts *n* and *p* denote the NMOS and PMOS devices, respectively):

$$\begin{aligned} I_{DSp} &= -I_{DSn} \\ V_{GSn} = V_{in} \;;\; V_{GSp} &= V_{in} - V_{DD} \\ V_{DSn} = V_{out} \;;\; V_{DSp} &= V_{out} - V_{DD} \end{aligned} \tag{5.1}$$

The load-line curves of the PMOS device are obtained by a mirroring around the *x*-axis and a horizontal shift over V_{DD}. This procedure is outlined in Figure 5.3, where the subsequent steps to adjust the original PMOS *I-V* curves to the common coordinate set V_{in}, V_{out} and I_{Dn} are illustrated.

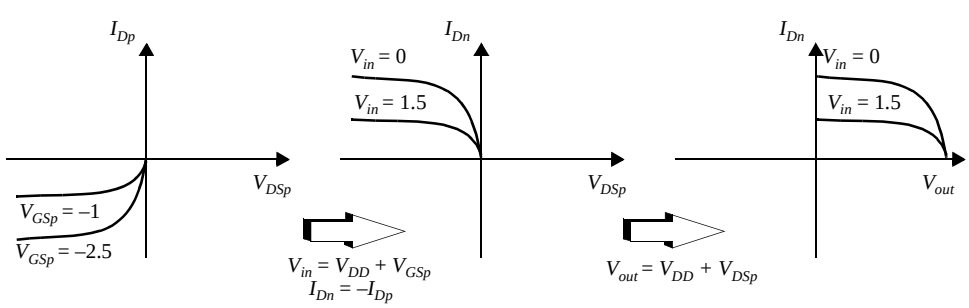


Figure 5.3 Transforming PMOS *I-V* characteristic to a common coordinate set (assuming VDD = 2.5 V).

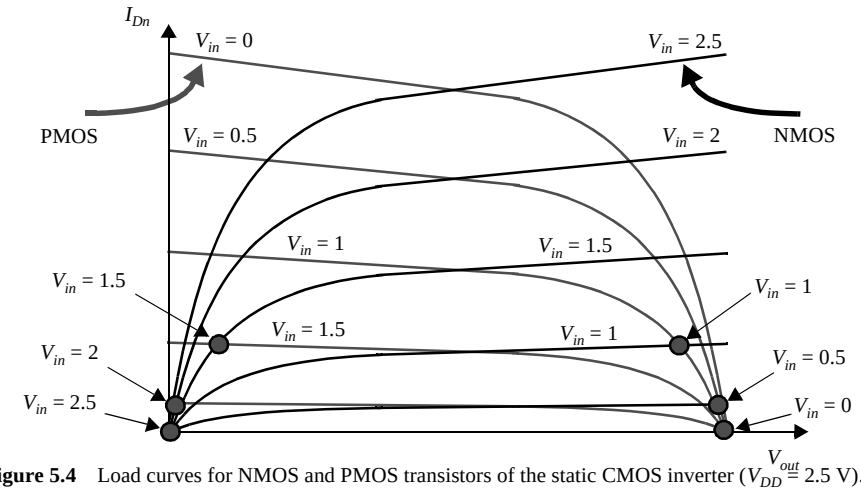


Figure 5.4 Load curves for NMOS and PMOS transistors of the static CMOS inverter (V_{DD} = 2.5 V). The dots represent the dc operation points for various input voltages.

The resulting load lines are plotted in Figure 5.4. For a dc operating points to be valid, the currents through the NMOS and PMOS devices must be equal. Graphically, this means that the dc points must be located at the intersection of corresponding load lines. A number of those points (for V_{in} = 0, 0.5, 1, 1.5, 2, and 2.5 V) are marked on the graph. As can be observed, all operating points are located either at the high or low output levels. The VTC of the inverter hence exhibits a very narrow transition zone. This results from the high gain during the switching transient, when both NMOS and PMOS are simultaneously on, and in saturation. In that operation region, a small change in the input voltage results in a large output variation. All these observations translate into the VTC of Figure 5.5.

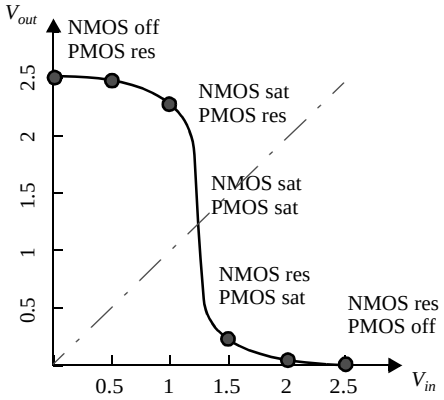
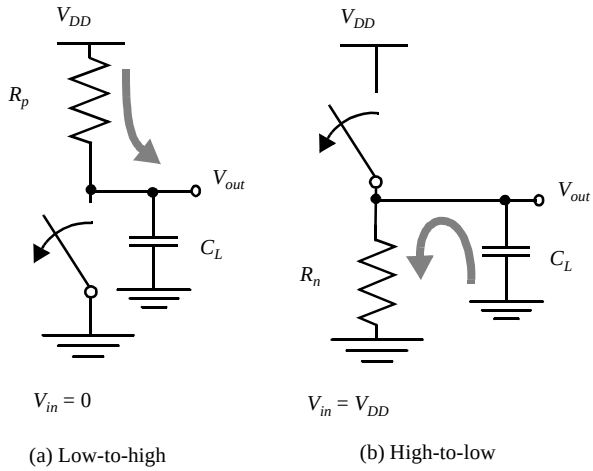


Figure 5.5 VTC of static CMOS inverter, derived from Figure 5.4 (V_{DD} = 2.5 V). For each operation region, the modes of the transistors are annotated — off, res(istive), or sat(urated).

Before going into the analytical details of the operation of the CMOS inverter, a qualitative analysis of the transient behavior of the gate is appropriate as well. This response is dominated mainly by the output capacitance of the gate, C_L, which is com-



(a) Low-to-high (b) High-to-low

Figure 5.6 Switch model of dynamic behavior of static CMOS inverter.

posed of the drain diffusion capacitances of the NMOS and PMOS transistors, the capacitance of the connecting wires, and the input capacitance of the fan-out gates. Assuming temporarily that the transistors switch instantaneously, we can get an approximate idea of the transient response by using the simplified switch model again (Figure 5.6). Let us consider the low-to-high transition first (Figure 5.6a). The gate response time is simply determined by the time it takes to charge the capacitor C_L through the resistor R_p. In Example 4.5, we learned that the propagation delay of such a network is proportional to the its time constant R_pC_L. **Hence, a fast gate is built either by keeping the output capacitance small or by decreasing the on-resistance of the transistor.** The latter is achieved by increasing the W/L ratio of the device. Similar considerations are valid for the high-to-low transition (Figure 5.6b), which is dominated by the R_nC_L time-constant. The reader should be aware that the on-resistance of the NMOS and PMOS transistor is not constant, but is a nonlinear function of the voltage across the transistor. This complicates the exact determination of the propagation delay. An in-depth analysis of how to analyze and optimize the performance of the static CMOS inverter is offered in Section 5.4.

5.3 Evaluating the Robustness of the CMOS Inverter: The Static Behavior

In the qualitative discussion above, the overall shape of the voltage-transfer characteristic of the static CMOS inverter was derived, as were the values of V_{OH} and V_{OL} (V_{DD} and *GND*, respectively). It remains to determine the precise values of V_M, V_{IH}, and V_{IL} as well as the noise margins.

5.3.1 Switching Threshold

The switching threshold, V_M, is defined as the point where $V_{in} = V_{out}$. Its value can be obtained graphically from the intersection of the VTC with the line given by $V_{in} = V_{out}$ (see Figure 5.5). In this region, both PMOS and NMOS are always saturated, since $V_{DS} = V_{GS}$. An analytical expression for V_M is obtained by equating the currents through the tran-

sistors. We solve the case where the supply voltage is high so that the devices can be assumed to be velocity-saturated (or $V_{DSAT} < V_M - V_T$). We furthermore ignore the channel-length modulation effects.

$$k_n V_{DSATn}\left(V_M - V_{Tn} - \frac{V_{DSATn}}{2}\right) + k_p V_{DSATp}\left(V_M - V_{DD} - V_{Tp} - \frac{V_{DSATp}}{2}\right) = 0 \qquad (5.2)$$

Solving for V_M yields

$$V_M = \frac{\left(V_{Tn} + \frac{V_{DSATn}}{2}\right) + r\left(V_{DD} + V_{Tp} + \frac{V_{DSATp}}{2}\right)}{1 + r} \quad \text{with} \quad r = \frac{k_p V_{DSATp}}{k_n V_{DSATn}} = \frac{\upsilon_{satp} W_p}{\upsilon_{satn} W_n} \qquad (5.3)$$

assuming identical oxide thicknesses for PMOS and NMOS transistors. For large values of V_{DD} (compared to threshold and saturation voltages), Eq. (5.3) can be simplified:

$$V_M \approx \frac{r V_{DD}}{1 + r} \qquad (5.4)$$

Eq. (5.4) states that the switching threshold is set by the ratio r, which compares the relative driving strengths of the PMOS and NMOS transistors. It is generally considered to be desirable for V_M to be located around the middle of the available voltage swing (or at $V_{DD}/2$), since this results in comparable values for the low and high noise margins. This requires r to be approximately 1, which is equivalent to sizing the PMOS device so that $(W/L)_p = (W/L)_n \times (V_{DSATn} k'_n)/(V_{DSATn} k'_p)$. To move V_M upwards, a larger value of r is required, which means making the PMOS wider. Increasing the strength of the NMOS, on the other hand, moves the switching threshold closer to GND.

From Eq. (5.2), we can derive the required ratio of PMOS versus NMOS transistor sizes such that the switching threshold is set to a desired value V_M. When using this expression, please make sure that the assumption that both devices are velocity-saturated still holds for the chosen operation point.

$$\frac{(W/L)_p}{(W/L)_n} = \frac{k'_n V_{DSATn}(V_M - V_{Tn} - V_{DSATn}/2)}{k'_p V_{DSATp}(V_{DD} - V_M + V_{Tp} + V_{DSATp}/2)} \qquad (5.5)$$

Problem 5.1 Inverter switching threshold for long-channel devices, or low supply-voltages.

The above expressions were derived under the assumption that the transistors are velocity-saturated. When the PMOS and NMOS are long-channel devices, or when the supply voltage is low, velocity saturation does not occur (V_M-$V_T < V_{DSAT}$). Under these circumstances, Eq. (5.6) holds for V_M. Derive.

$$V_M = \frac{V_{Tn} + r(V_{DD} + V_{Tp})}{1 + r} \quad \text{with} \quad r = \sqrt{\frac{-k_p}{k_n}} \qquad (5.6)$$

Design Technique — Maximizing the noise margins

When designing static CMOS circuits, it is advisable to balance the driving strengths of the transistors by making the PMOS section wider than the NMOS section, if one wants to maximize the noise margins and obtain symmetrical characteristics. The required ratio is given by Eq. (5.5).

Example 5.1 Switching threshold of CMOS inverter

We derive the sizes of PMOS and NMOS transistors such that the switching threshold of a CMOS inverter, implemented in our generic 0.25 μm CMOS process, is located in the middle between the supply rails. We use the process parameters presented in Example 3.7, and assume a supply voltage of 2.5 V. The minimum size device has a width/length ratio of 1.5. With the aid of Eq. (5.5), we find

$$\frac{(W/L)_p}{(W/L)_n} = \frac{115 \times 10^{-6}}{30 \times 10^{-6}} \times \frac{0.63}{1.0} \times \frac{(1.25 - 0.43 - 0.63/2)}{(1.25 - 0.4 - 1.0/2)} = 3.5$$

Figure 5.7 plots the values of switching threshold as a function of the PMOS/NMOS ratio, as obtained by circuit simulation. The simulated PMOS/NMOS ratio of 3.4 for a 1.25 V switching threshold confirms the value predicted by Eq. (5.5).

An analysis of the curve of Figure 5.7 produces some interesting observations:

1. V_M is relatively insensitive to variations in the device ratio. This means that small variations of the ratio (e.g., making it 3 or 2.5) do not disturb the transfer characteristic that much. It is therefore an accepted practice in industrial designs to set the width of the PMOS transistor to values smaller than those required for exact symmetry. For the above example, setting the ratio to 3, 2.5, and 2 yields switching thresholds of 1.22 V, 1.18 V, and 1.13 V, respectively.

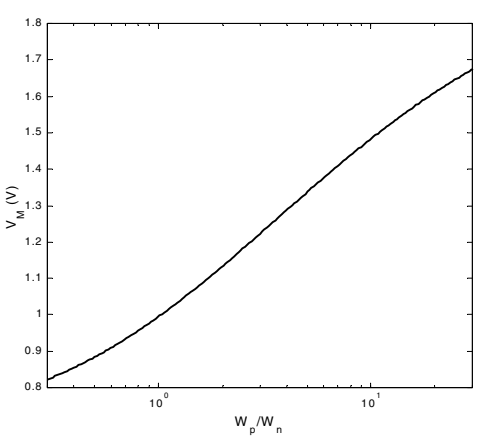
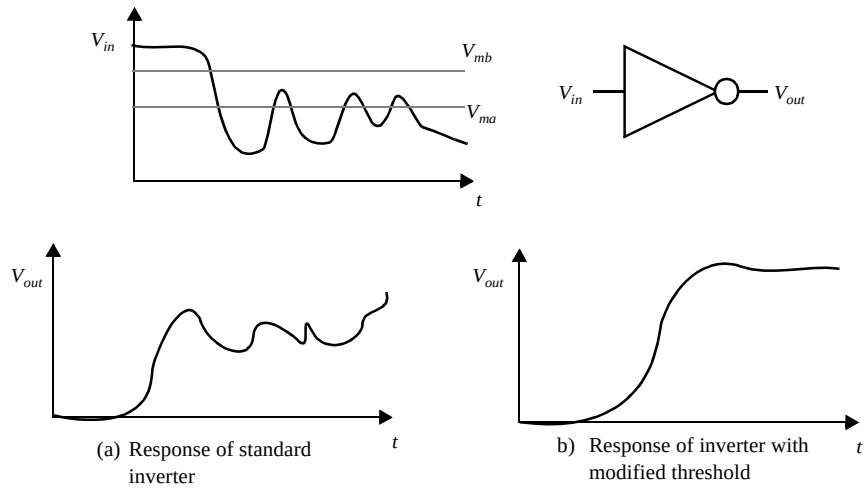


Figure 5.7 Simulated inverter switching threshold versus PMOS/NMOS ratio (0.25 μm CMOS, V_{DD} = 2.5 V)



(a) Response of standard inverter

b) Response of inverter with modified threshold

Figure 5.8 Changing the inverter threshold can improve the circuit reliability.

2. The effect of changing the W_p/W_n ratio is to shift the transient region of the VTC. Increasing the width of the PMOS or the NMOS moves V_M towards V_{DD} or *GND* respectively. This property can be very useful, as asymmetrical transfer characteristics are actually desirable in some designs. This is demonstrated by the example of Figure 5.8. The incoming signal V_{in} has a very noisy zero value. Passing this signal through a symmetrical inverter would lead to erroneous values (Figure 5.8a). This can be addressed by raising the threshold of the inverter, which results in a correct response (Figure 5.8b). Further in the text, we will see other circuit instances where inverters with asymmetrical switching thresholds are desirable.
Changing the switching threshold by a considerable amount is however not easy, especially when the ratio of supply voltage to transistor threshold is relatively small (2.5/0.4 = 6 for our particular example). To move the threshold to 1.5 V requires a transistor ratio of 11, and further increases are prohibitively expensive. Observe that Figure 5.7 is plotted in a semi-log format.

5.3.2 Noise Margins

By definition, V_{IH} and V_{IL} are the operational points of the inverter where $\frac{dV_{out}}{dV_{in}} = -1$. In the terminology of the analog circuit designer, these are the points where the gain g of the amplifier, formed by the inverter, is equal to –1. While it is indeed possible to derive analytical expressions for V_{IH} and V_{IL}, these tend to be unwieldy and provide little insight in what parameters are instrumental in setting the noise margins.

A simpler approach is to use a piece wise linear approximation for the VTC, as shown in Figure 5.9. The transition region is approximated by a straight line, the gain of which equals the gain g at the switching threshold V_M. The crossover with the V_{OH} and the V_{OL} lines is used to define V_{IH} and V_{IL} points. The error introduced is small and well

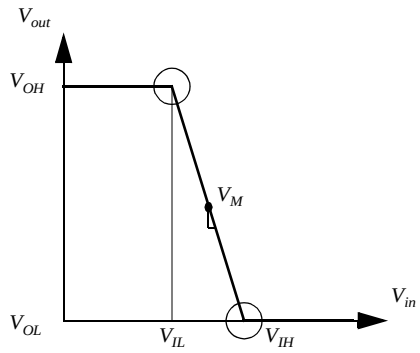


Figure 5.9 A piece-wise linear approximation of the VTC simplifies the derivation of V_{IL} and V_{IH}.

within the range of what is required for an initial design. This approach yields the following expressions for the width of the transition region V_{IH} - V_{IL}, V_{IH}, V_{IL}, and the noise margins NM_H and NM_L.

$$\begin{aligned} V_{IH} - V_{IL} &= -\frac{(V_{OH} - V_{OL})}{g} = \frac{-V_{DD}}{g} \\ V_{IH} &= V_M - \frac{V_M}{g} \qquad V_{IL} = V_M + \frac{V_{DD} - V_M}{g} \\ NM_H &= V_{DD} - V_{IH} \qquad NM_L = V_{IL} \end{aligned} \tag{5.7}$$

These expressions make it increasingly clear that a high gain in the transition region is very desirable. In the extreme case of an infinite gain, the noise margins simplify to V_{OH} - V_M and V_M - V_{OL} for NM_H and NM_L, respectively, and span the complete voltage swing.

Remains us to determine the midpoint gain of the static CMOS inverter. We assume once again that both PMOS and NMOS are velocity-saturated. It is apparent from Figure 5.4 that the gain is a strong function of the slopes of the currents in the saturation region. The channel-length modulation factor hence cannot be ignored in this analysis — doing so would lead to an infinite gain. The gain can now be derived by differentiating the current equation (5.8), valid around the switching threshold, with respect to V_{in}.

$$\begin{aligned} &k_n V_{DSATn}\left(V_{in} - V_{Tn} - \frac{V_{DSATn}}{2}\right)(1 + \lambda_n V_{out}) + \\ &k_p V_{DSATp}\left(V_{in} - V_{DD} - V_{Tp} - \frac{V_{DSATp}}{2}\right)(1 + \lambda_p V_{out} - \lambda_p V_{DD}) = 0 \end{aligned} \tag{5.8}$$

Differentiation and solving for dV_{out}/dV_{in} yields

$$\frac{dV_{out}}{dV_{in}} = -\frac{k_n V_{DSATn}(1 + \lambda_n V_{out}) + k_p V_{DSATp}(1 + \lambda_p V_{out} - \lambda_p V_{DD})}{\lambda_n k_n V_{DSATn}(V_{in} - V_{Tn} - V_{DSATn}/2) + \lambda_p k_p V_{DSATp}(V_{in} - V_{DD} - V_{Tp} - V_{DSATp}/2)} \tag{5.9}$$

Ignoring some second-order terms, and setting $V_{in} = V_M$ results in the gain expression,

$$g = -\frac{1}{I_D(V_M)}\frac{k_n V_{DSATn} + k_p V_{DSATp}}{\lambda_n - \lambda_p} \approx \frac{1+r}{(V_M - V_{Tn} - V_{DSATn}/2)(\lambda_n - \lambda_p)} \tag{5.10}$$

with $I_D(V_M)$ the current flowing through the inverter for $V_{in} = V_M$. The gain is almost purely determined by technology parameters, especially the channel length modulation. It can only in a minor way be influenced by the designer through the choice of supply and switching threshold voltages.

Example 5.2 Voltage transfer characteristic and noise margins of CMOS Inverter

Assume an inverter in the generic 0.25 μm CMOS technology designed with a PMOS/NMOS ratio of 3.4 and with the NMOS transistor minimum size (W = 0.375 μm, L = 0.25 μm, W/L = 1.5). We first compute the gain at V_M (= 1.25 V),

$$I_D(V_M) = 1.5 \times 115 \times 10^{-6} \times 0.63 \times (1.25 - 0.43 - 0.63/2) \times (1 + 0.06 \times 1.25) = 59 \times 10^{-6}\ \text{A}$$

$$g = -\frac{1}{59 \times 10^{-6}}\frac{1.5 \times 115 \times 10^{-6} \times 0.63 + 1.5 \times 3.4 \times 30 \times 10^{-6} \times 1.0}{0.06 + 0.1} = -27.5 \quad \text{(Eq. 5.10)}$$

This yields the following values for V_{IL}, V_{IH}, NM_L, NM_H:

$$V_{IL} = 1.2\ \text{V},\ V_{IH} = 1.3\ \text{V},\ NM_L = NM_H = 1.2.$$

Figure 5.10 plots the simulated VTC of the inverter, as well as its derivative, the gain. A close to ideal characteristic is obtained. The actual values of V_{IL} and V_{IH} are 1.03 V and 1.45 V, respectively, which leads to noise margins of 1.03 V and 1.05 V. These values are lower than those predicted for two reasons:

• Eq. (5.10) overestimates the gain. As observed in Figure 5.10b, the maximum gain (at V_M) equals only 17. This reduced gain would yield values for V_{IL} and V_{IH} of 1.17 V, and 1.33 V, respectively.

• The most important deviation is due to the piecewise linear approximation of the VTC, which is optimistic with respect to the actual noise margins.

The obtained expressions are however perfectly useful as first-order estimations as well as means of identifying the relevant parameters and their impact.

To conclude this example, we also extracted from simulations the output resistance of the inverter in the low- and high-output states. Low values of 2.4 kΩ and 3.3 kΩ were observed, respectively. The output resistance is a good measure of the sensitivity of the gate in respect to noise induced at the output, and is preferably as low as possible.

SIDELINE: Surprisingly (or not so surprisingly), the static CMOS inverter can also be used as an analog amplifier, as it has a fairly high gain in its transition region. This region is very narrow however, as is apparent in the graph of Figure 5.10b. It also receives poor marks on other amplifier properties such as supply noise rejection. Yet, this observation can be used to demonstrate one of the major differences between analog and digital design. Where the analog designer would bias the amplifier in the middle of the transient region, so that a maximum linearity is obtained, the digital designer will operate the

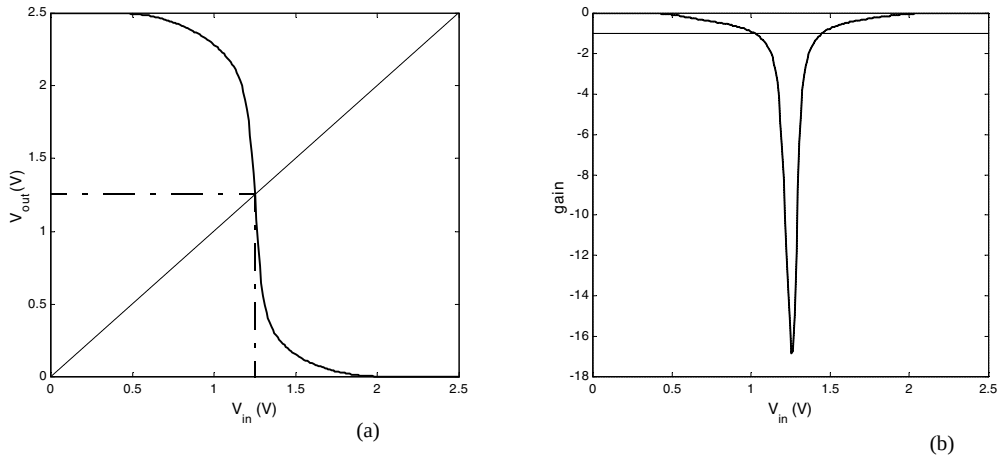


Figure 5.10 Simulated Voltage Transfer Characteristic (a) and voltage gain (b) of CMOS inverter (0.25 μm CMOS, V_{DD} = 2.5 V).

device in the regions of extreme nonlinearity, resulting in well-defined and well-separated high and low signals.

Problem 5.2 Inverter noise margins for long-channel devices

Derive expressions for the gain and noise margins assuming that PMOS and NMOS are long-channel devices (or that the supply voltage is low), so that velocity saturation does not occur.

5.3.3 Robustness Revisited

Device Variations

While we design a gate for nominal operation conditions and typical device parameters, we should always be aware that the actual operating temperature might very over a large range, and that the device parameters after fabrication probably will deviate from the nominal values we used in our design optimization process. Fortunately, the dc-characteristics of the static CMOS inverter turn out to be rather insensitive to these variations, and the gate remains functional over a wide range of operating conditions. This already became apparent in Figure 5.7, which shows that variations in the device sizes have only a minor impact on the switching threshold of the inverter. To further confirm the assumed robustness of the gate, we have re-simulated the voltage transfer characteristic by replacing the nominal devices by their worst- or best-case incarnations. Two corner-cases are plotted in Figure 5.11: a better-than-expected NMOS combined with an inferior PMOS, and the opposite scenario. Comparing the resulting curves with the nominal response shows that the variations mostly cause a shift in the switching threshold, but that the operation of the

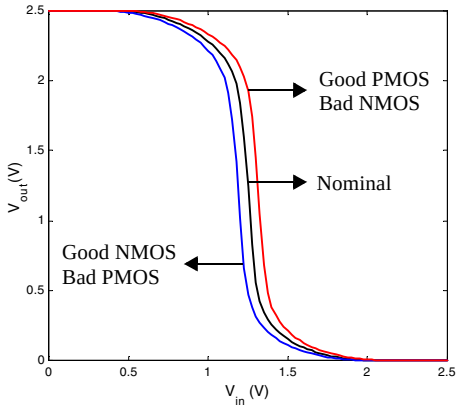


Figure 5.11 Impact of device variations on static CMOS inverter VTC. The "good" device has a smaller oxide thickness (- 3nm), a smaller length (-25 nm), a higher width (+30 nm), and a smaller threshold (-60 mV). The opposite is true for the "bad" transistor.

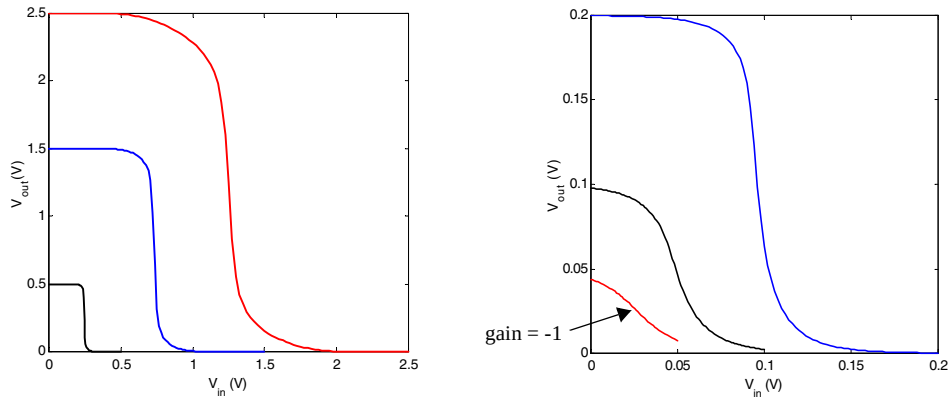
gate is by no means affected. This robust behavior that ensures functionality of the gate over a wide range of conditions has contributed in a big way to the popularity of the static CMOS gate.

Scaling the Supply Voltage

In Chapter 3, we observed that continuing technology scaling forces the supply voltages to reduce at rates similar to the device dimensions. At the same time, device threshold voltages are virtually kept constant. The reader probably wonders about the impact of this trend on the integrity parameters of the CMOS inverter. Do inverters keep on working when the voltages are scaled and are there potential limits to the supply scaling?

A first hint on what might happen was offered in Eq. (5.10), which indicates that the gain of the inverter in the transition region actually increases with a reduction of the supply voltage! Note that for a fixed transistor ratio r, V_M is approximately proportional to V_{DD}. Plotting the (normalized) VTC for different supply voltages not only confirms this conjecture, but even shows that the inverter is well and alive for supply voltages close to the threshold voltage of the composing transistors (Figure 5.12a). At a voltage of 0.5 V — which is just 100 mV above the threshold of the transistors — the width of the transition region measures only 10% of the supply voltage (for a maximum gain of 35), while it widens to 17% for 2.5 V. So, given this improvement in dc characteristics, why do we not choose to operate all our digital circuits at these low supply voltages? Three important arguments come to mind:

- In the following sections, we will learn that reducing the supply voltage indiscriminately has a positive impact on the energy dissipation, but is absolutely detrimental to the performance on the gate.
- The dc-characteristic becomes increasingly sensitive to variations in the device parameters such as the transistor threshold, once supply voltages and intrinsic voltages become comparable.
- Scaling the supply voltage means reducing the signal swing. While this typically helps to reduce the internal noise in the system (such as caused by crosstalk), it makes the design more sensitive to external noise sources that do not scale.



(a) Reducing V_{DD} improves the gain...

(b) but it detoriates for very-low supply voltages.

Figure 5.12 VTC of CMOS inverter as a function of supply voltage (0.25 µm CMOS technology).

To provide an insight into the question on potential limits to the voltage scaling, we have plotted in Figure 5.12b the voltage transfer characteristic of the same inverter for the even-lower supply voltages of 200 mV, 100 mV, and 50 mV (while keeping the transistor thresholds at the same level). Amazingly enough, we still obtain an inverter characteristic, this while the supply voltage is not even large enough to turn the transistors on! The explanation can be found in the sub-threshold operation of the transistors. The sub-threshold currents are sufficient to switch the gate between low and high levels, and provide enough gain to produce acceptable VTCs. The very low value of the switching currents ensures a very slow operation but this might be acceptable for some applications (such as watches, for example).

At around 100 mV, we start observing a major deterioration of the gate characteristic. V_{OL} and V_{OH} are no longer at the supply rails and the transition-region gain approaches 1. The latter turns out to be a fundamental show-stopper. To achieving sufficient gain for use in a digital circuit, it is necessary that the supply must be at least a couple times $\phi_T = kT/q$ (=25 mV at room temperature), the thermal voltage introduced in Chapter 3 [Swanson72]. It turns out that below this same voltage, thermal noise becomes an issue as well, potentially resulting in unreliable operation.

$$V_{DDmin} > 2\ldots4\frac{kT}{q} \tag{5.11}$$

Eq. (5.11) presents a true lower bound on supply scaling. It suggests that the only way to get CMOS inverters to operate below 100 mV is to reduce the ambient temperature, or in other words to cool the circuit.

Problem 5.3 Minimum supply voltage of CMOS inverter

Once the supply voltage drops below the threshold voltage, the transistors operate the sub-threshold region, and display an exponential current-voltage relationship (as expressed in Eq. (3.40)). Derive an expression for the gain of the inverter under these circumstances

(assume symmetrical NMOS and PMOS transistors, and a maximum gain at $V_M = V_{DD}/2$). The resulting expression demonstrates that the minimum voltage is a function of the slope factor n of the transistor.

$$g = -\left(\frac{1}{n}\right)(e^{V_{DD}/2\phi_T} - 1) \tag{5.12}$$

According to this expression, the gain drops to -1 at V_{DD} = 48 mV (for n = 1.5 and ϕ_T = 25 mV).

5.4 Performance of CMOS Inverter: The Dynamic Behavior

The qualitative analysis presented earlier concluded that the propagation delay of the CMOS inverter is determined by the time it takes to charge and discharge the load capacitor C_L through the PMOS and NMOS transistors, respectively. This observation suggests that **getting C_L as small as possible is crucial to the realization of high-performance CMOS circuits**. It is hence worthwhile to first study the major components of the load capacitance before embarking onto an in-depth analysis of the propagation delay of the gate. In addition to this detailed analysis, the section also presents a summary of techniques that a designer might use to optimize the performance of the inverter.

5.4.1 Computing the Capacitances

Manual analysis of MOS circuits where each capacitor is considered individually is virtually impossible and is exacerbated by the many nonlinear capacitances in the MOS transistor model. To make the analysis tractable, we assume that all capacitances are lumped together into one single capacitor C_L, located between V_{out} and *GND*. Be aware that this is a considerable simplification of the actual situation, even in the case of a simple inverter.

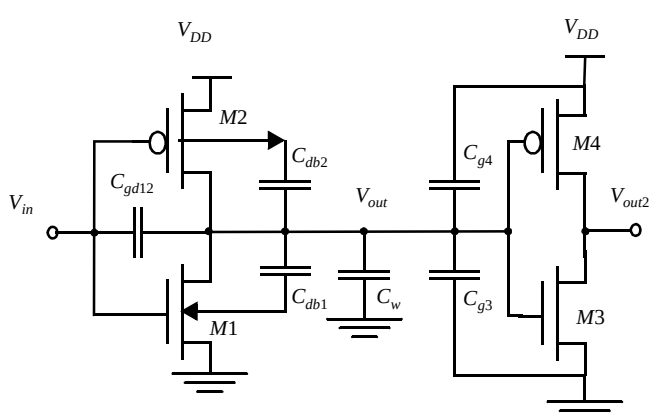


Figure 5.13 Parasitic capacitances, influencing the transient behavior of the cascaded inverter pair.

Figure 5.13 shows the schematic of a cascaded inverter pair. It includes all the capacitances influencing the transient response of node V_{out}. It is initially assumed that the input V_{in} is driven by an *ideal voltage source with zero rise and fall times*. Accounting only for capacitances connected to the output node, C_L breaks down into the following components.

Gate-Drain Capacitance C_{gd12}

M1 and M2 are either in cut-off or in the saturation mode during the first half (up to 50% point) of the output transient. Under these circumstances, the only contributions to C_{gd12} are the overlap capacitances of both M1 and M2. The channel capacitance of the MOS transistors does not play a role here, as it is located either completely between gate and bulk (cut-off) or gate and source (saturation) (see Chapter 3).

The lumped capacitor model now requires that this floating gate-drain capacitor be replaced by a capacitance-to-ground. This is accomplished by taking the so-called Miller effect into account. During a low-high or high-low transition, the terminals of the gate-drain capacitor are moving in opposite directions (Figure 5.14). The voltage change over the floating capacitor is hence twice the actual output voltage swing. To present an identical load to the output node, the capacitance-to-ground must have a value that is twice as large as the floating capacitance.

We use the following equation for the gate-drain capacitors: $C_{gd} = 2\ C_{GD0}W$ (with C_{GD0} the overlap capacitance per unit width as used in the SPICE model). For an in-depth discussion of the Miller effect, please refer to textbooks such as Sedra and Smith ([Sedra87], p. 57).[1]

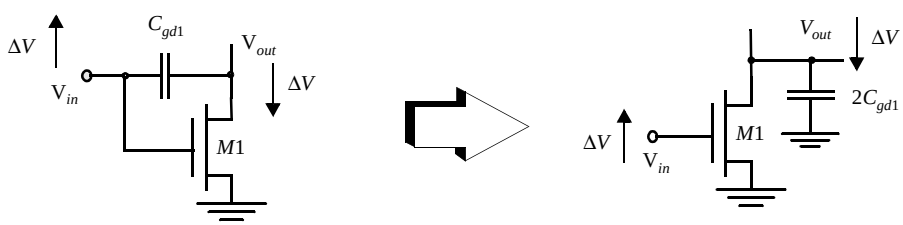


Figure 5.14 The Miller effect—A capacitor experiencing identical but opposite voltage swings at both its terminals can be replaced by a capacitor to ground, whose value is two times the original value.

Diffusion Capacitances C_{db1} and C_{db2}

The capacitance between drain and bulk is due to the reverse-biased *pn*-junction. Such a capacitor is, unfortunately, quite nonlinear and depends heavily on the applied voltage. We argued in Chapter 3 that the best approach towards simplifying the analysis is to replace the nonlinear capacitor by a linear one with the same change in charge for the voltage range of interest. A multiplication factor K_{eq} is introduced to relate the linearized capacitor to the value of the junction capacitance under zero-bias conditions.

[1] The Miller effect discussed in this context is a simplified version of the general analog case. In a digital inverter, the large scale gain between input and output always equals -1.

$$C_{eq} = K_{eq} C_{j0} \tag{5.13}$$

with C_{j0} the junction capacitance per unit area under zero-bias conditions. An expression for K_{eq} was derived in Eq. (3.11) and is repeated here for convenience

$$K_{eq} = \frac{-\phi_0^m}{(V_{high} - V_{low})(1-m)} [(\phi_0 - V_{high})^{1-m} - (\phi_0 - V_{low})^{1-m}] \tag{5.14}$$

with ϕ_0 the built-in junction potential and m the grading coefficient of the junction. Observe that the junction voltage is defined to be negative for reverse-biased junctions.

Example 5.3 K_{eq} for a 2.5 V CMOS Inverter

Consider the inverter of Figure 5.13 designed in the generic 0.25 μm CMOS technology. The relevant capacitance parameters for this process were summarized in Table 3.5.

Let us first analyze the NMOS transistor (C_{db1} in Figure 5.13). The propagation delay is defined by the time between the 50% transitions of the input and the output. For the CMOS inverter, this is the time-instance where V_{out} reaches 1.25 V, as the output voltage swing goes from rail to rail or equals 2.5 V. We, therefore, linearize the junction capacitance over the interval {2.5 V, 1.25 V} for the high-to-low transition, and {0, 1.25 V} for the low-to-high transition.

During the high-to-low transition at the output, V_{out} initially equals 2.5 V. Because the bulk of the NMOS device is connected to *GND*, this translates into a reverse voltage of 2.5 V over the drain junction or $V_{high} = -2.5$ V. At the 50% point, $V_{out} = 1.25$ V or $V_{low} = -1.25$ V. Evaluating Eq. (5.14) for the bottom plate and sidewall components of the diffusion capacitance yields

Bottom plate: K_{eq} ($m = 0.5$, $\phi_0 = 0.9$) = 0.57,
Sidewall: K_{eqsw} ($m = 0.44$, $\phi_0 = 0.9$) = 0.61

During the low-to-high transition, V_{low} and V_{high} equal 0 V and –1.25 V, respectively, resulting in higher values for K_{eq},

Bottom plate: K_{eq} ($m = 0.5$, $\phi_0 = 0.9$) = 0.79,
Sidewall: K_{eqsw} ($m = 0.44$, $\phi_0 = 0.9$) = 0.81

The PMOS transistor displays a reverse behavior, as its substrate is connected to 2.5 V. Hence, for the high-to-low transition ($V_{low} = 0$, $V_{high} = -1.25$ V),

Bottom plate: K_{eq} ($m = 0.48$, $\phi_0 = 0.9$) = 0.79,
Sidewall: K_{eqsw} ($m = 0.32$, $\phi_0 = 0.9$) = 0.86

and for the low-to-high transition ($V_{low} = -1.25$ V, $V_{high} = -2.5$ V)

Bottom plate: K_{eq} ($m = 0.48$, $\phi_0 = 0.9$) = 0.59,
Sidewall: K_{eqsw} ($m = 0.32$, $\phi_0 = 0.9$) = 0.7

Using this approach, the junction capacitance can be replaced by a linear component and treated as any other device capacitance. The result of the linearization is a minor distortion of the voltage waveforms. The logic delays are not significantly influenced by this simplification.

Wiring Capacitance C_w

The capacitance due to the wiring depends upon the length and width of the connecting wires, and is a function of the distance of the fanout from the driving gate and the number of fanout gates. As argued in Chapter 4, this component is growing in importance with the scaling of the technology.

Gate Capacitance of Fanout C_{g3} and C_{g4}

We assume that the fanout capacitance equals the total gate capacitance of the loading gates M3 and M4. Hence,

$$\begin{aligned} C_{fanout} &= C_{gate}(\text{NMOS}) + C_{gate}(\text{PMOS}) \\ &= (C_{GSOn} + C_{GDOn} + W_n L_n C_{ox}) + (C_{GSOp} + C_{GDOp} + W_p L_p C_{ox}) \end{aligned} \tag{5.15}$$

This expression simplifies the actual situation in two ways:

- It assumes that all components of the gate capacitance are connected between V_{out} and GND (or V_{DD}), and ignores the Miller effect on the gate-drain capacitances. This has a relatively minor effect on the accuracy, since we can safely assume that the connecting gate does not switch before the 50% point is reached, and V_{out2}, therefore, remains constant in the interval of interest.
- A second approximation is that the channel capacitance of the connecting gate is constant over the interval of interest. This is not exactly the case as we discovered in Chapter 3. The total channel capacitance is a function of the operation mode of the device, and varies from approximately 2/3 WLC_{ox} (saturation) to the full WLC_{ox} (linear and cut-off). A drop in overall gate capacitance also occurs just before the transistor turns on (Figure 3.30). During the first half of the transient, it may be assumed that one of the load devices is always in linear mode, while the other transistor evolves from the off-mode to saturation. Ignoring the capacitance variation results in a pessimistic estimation with an error of approximately 10%, which is acceptable for a first order analysis.

Example 5.4 Capacitances of a 0.25 μm CMOS Inverter

A minimum-size, symmetrical CMOS inverter has been designed in the 0.25 μm CMOS technology. The layout is shown in Figure 5.15. The supply voltage V_{DD} is set to 2.5 V. From the layout, we derive the transistor sizes, diffusion areas, and perimeters. This data is summarized in Table 5.1. As an example, we will derive the drain area and perimeter for the NMOS transistor. The drain area is formed by the metal-diffusion contact, which has an area of $4 \times 4\ \lambda^2$, and the rectangle between contact and gate, which has an area of $3 \times 1\ \lambda^2$. This results in a total area of $19\ \lambda^2$, or 0.30 μm^2 (as $\lambda = 0.125$ μm). The perimeter of the drain area is rather involved and consists of the following components (going counterclockwise): 5 + 4 + 4 + 1 + 1 = 15 λ or PD = 15 × 0.125 = 1.875 μm. Notice that the gate side of the drain perimeter is not included, as this is not considered a part of the side-wall. The drain area and perimeter of the PMOS transistor are derived similarly (the rectangular shape makes the exercise considerably simpler): AD = $5 \times 9\ \lambda^2 = 45\ \lambda^2$, or 0.7 μm^2; PD = 5 + 9 + 5 = 19 λ, or 2.375 μm.

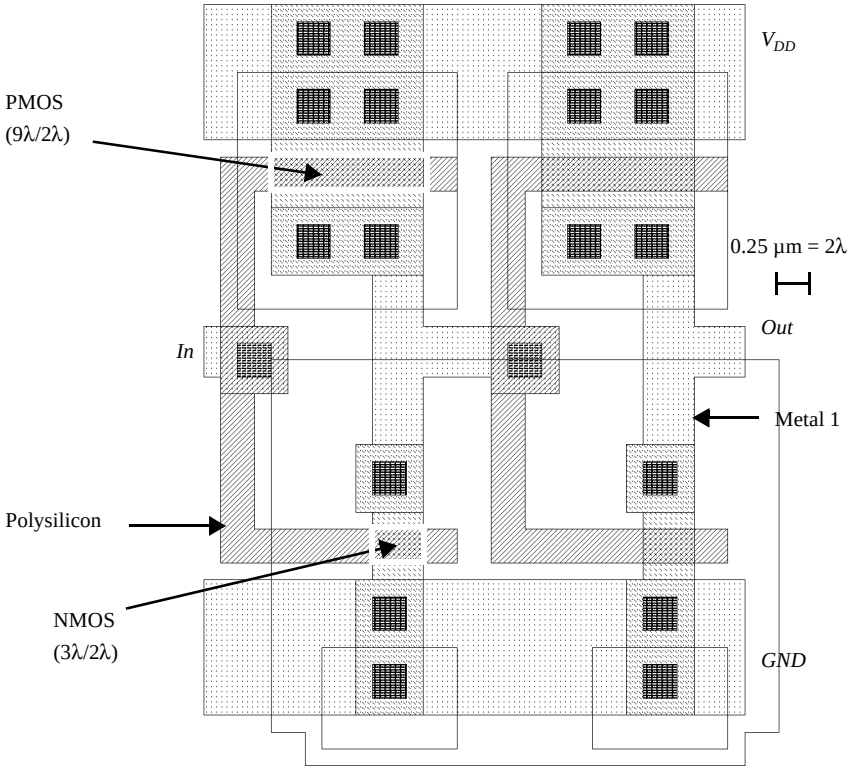


Figure 5.15 Layout of two chained, minimum-size inverters using SCMOS Design Rules (see also Color-plate 6).

Table 5.1 Inverter transistor data.

	W/L	**AD (μm^2)**	**PD (μm)**	**AS (μm^2)**	**PS (μm)**
NMOS	0.375/0.25	0.3 (19 λ^2)	1.875 (15λ)	0.3 (19 λ^2)	1.875 (15λ)
PMOS	1.125/0.25	0.7 (45 λ^2)	2.375 (19λ)	0.7 (45 λ^2)	2.375 (19λ)

This physical information can be combined with the approximations derived above to come up with an estimation of C_L. The capacitor parameters for our generic process were summarized in Table 3.5, and repeated here for convenience:

Overlap capacitance: CGD0(NMOS) = 0.31 fF/μm; CGDO(PMOS) = 0.27 fF/μm
Bottom junction capacitance: CJ(NMOS) = 2 fF/μm^2; CJ(PMOS) = 1.9 fF/μm^2
Side-wall junction capacitance: CJSW(NMOS) = 0.28 fF/μm; CJSW(PMOS) = 0.22 fF/μm
Gate capacitance: C_{ox}(NMOS) = C_{ox}(PMOS) = 6 fF/μm^2

Finally, we should also consider the capacitance contributed by the wire, connecting the gates and implemented in metal 1 and polysilicon. A layout extraction program typically

will deliver us precise values for this parasitic capacitance. Inspection of the layout helps us to form a first-order estimate and yields that the metal-1 and polysilicon areas of the wire, that are not over active diffusion, equal 42 λ^2 and 72 λ^2, respectively. With the aid of the interconnect parameters of Table 4.2, we find the wire capacitance — observe that we ignore the fringing capacitance in this simple exercise. Due to the short length of the wire, this contribution is ignorable compared to the other parasitics.

$$C_{wire} = 42/8^2\ \mu m^2 \times 30\ aF/\mu m^2 + 72/8^2\ \mu m^2 \times 88\ aF/\mu m^2 = 0.12\ fF$$

Bringing all the components together results in Table 5.2. We use the values of K_{eq} derived in Example 5.3 for the computation of the diffusion capacitances. Notice that the load capacitance is almost evenly split between its two major components: the intrinsic capacitance, composed of diffusion and overlap capacitances, and the extrinsic load capacitance, contributed by wire and connecting gate.

Table 5.2 Components of C_L (for high-to-low and low-to-high transitions).

Capacitor	Expression	Value (fF) (H→L)	Value (fF) (L→H)
C_{gd1}	$2\ CGD0_n\ W_n$	0.23	0.23
C_{gd2}	$2\ CGD0_p\ W_p$	0.61	0.61
C_{db1}	$K_{eqn}\ AD_n\ CJ + K_{eqswn}\ PD_n\ CJSW$	0.66	0.90
C_{db2}	$K_{eqp}\ AD_p\ CJ + K_{eqswp}\ PD_p\ CJSW)$	1.5	1.15
C_{g3}	$(CGD0_n+CGSO_n)\ W_n + C_{ox}\ W_n\ L_n$	0.76	0.76
C_{g4}	$(CGD0_p+CGSO_p)\ W_p + C_{ox}\ W_p\ L_p$	2.28	2.28
C_w	From Extraction	0.12	0.12
C_L	Σ	6.1	6.0

5.4.2 Propagation Delay: First-Order Analysis

One way to compute the propagation delay of the inverter is to integrate the capacitor (dis)charge current. This results in the expression of Eq. (5.16).

$$t_p = \int_{v_1}^{v_2} \frac{C_L(v)}{i(v)} dv \tag{5.16}$$

with i the (dis)charging current, v the voltage over the capacitor, and v_1 and v_2 the initial and final voltage. An exact computation of this equation is intractable, as both $C_L(v)$ and $i(v)$ are nonlinear functions of v. We rather fall back to the simplified switch-model of the inverter introduced in Figure 5.6 to derive a reasonable approximation of the propagation delay adequate for manual analysis. The voltage-dependencies of the on-resistance and the load capacitor are addressed by replacing both by a constant linear element with a value averaged over the interval of interest. The preceding section derived precisely this value

for the load capacitance. An expression for the average on-resistance of the MOS transistor was already derived in Example 3.8, and is repeated here for convenience.

$$R_{eq} = \frac{1}{V_{DD}/2}\int_{V_{DD}/2}^{V_{DD}} \frac{V}{I_{DSAT}(1+\lambda V)}dV \approx \frac{3}{4}\frac{V_{DD}}{I_{DSAT}}\left(1-\frac{7}{9}\lambda V_{DD}\right)$$
$$\text{with} \quad I_{DSAT} = k'\frac{W}{L}\left((V_{DD}-V_T)V_{DSAT}-\frac{V_{DSAT}^2}{2}\right) \tag{5.17}$$

Deriving the propagation delay of the resulting circuit is now straightforward, and is nothing more than the analysis of a first-order linear *RC*-network, identical to the exercise of Example 4.5. There, we learned that the propagation delay of such a network for a voltage step at the input is proportional to the time-constant of the network, formed by pull-down resistor and load capacitance. Hence,

$$t_{pHL} = \ln(2)R_{eqn}C_L = 0.69R_{eqn}C_L \tag{5.18}$$

Similarly, we can obtain the propagation delay for the low-to-high transition,

$$t_{pLH} = 0.69R_{eqp}C_L \tag{5.19}$$

with R_{eqp} the equivalent on-resistance of the PMOS transistor over the interval of interest. This analysis assumes that the equivalent load-capacitance is identical for both the high-to-low and low-to-high transitions. This has been shown to be approximately the case in the example of the previous section. The overall propagation delay of the inverter is defined as the average of the two values, or

$$t_p = \frac{t_{pHL}+t_{pLH}}{2} = 0.69C_L\left(\frac{R_{eqn}+R_{eqp}}{2}\right) \tag{5.20}$$

Very often, it is desirable for a gate to have identical propagation delays for both rising and falling inputs. This condition can be achieved by making the on-resistance of the NMOS and PMOS approximately equal. Remember that this condition is identical to the requirement for a symmetrical VTC.

Example 5.5 Propagation Delay of a 0.25 μm CMOS Inverter

To derive the propagation delays of the CMOS inverter of Figure 5.15, we make use of Eq. (5.18) and Eq. (5.19). The load capacitance C_L was already computed in Example 5.4, while the equivalent on-resistances of the transistors for the generic 0.25 μm CMOS process were derived in Table 3.3. For a supply voltage of 2.5 V, the normalized on-resistances of NMOS and PMOS transistors equal 13 kΩ and 31 kΩ, respectively. From the layout, we determine the (W/L) ratios of the transistors to be 1.5 for the NMOS, and 4.5 for the PMOS. We assume that the difference between drawn and effective dimensions is small enough to be ignorable. This leads to the following values for the delays:

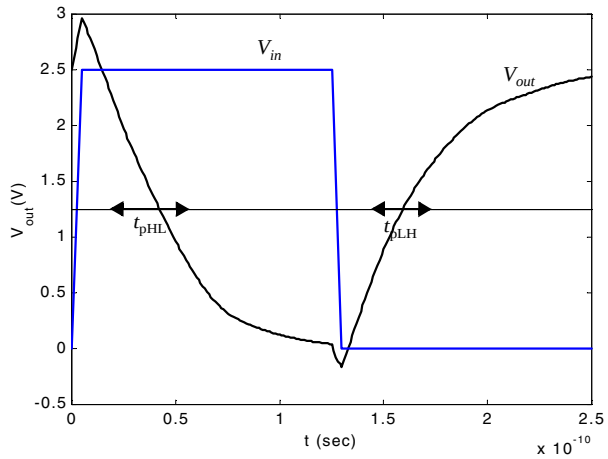


Figure 5.16 Simulated transient response of the inverter of Figure 5.15.

$$t_{pHL} = 0.69 \times \left(\frac{13\text{k}\Omega}{1.5}\right) \times 6.1\text{fF} = 36 \text{ psec}$$

$$t_{pLH} = 0.69 \times \left(\frac{31\text{k}\Omega}{4.5}\right) \times 6.0\text{fF} = 29 \text{ psec}$$

and

$$t_p = \left(\frac{36 + 29}{2}\right) = 32.5 \text{ psec}$$

The accuracy of this analysis is checked by performing a SPICE transient simulation on the circuit schematic, extracted from the layout of Figure 5.15. The computed transient response of the circuit is plotted in Figure 5.16, and determines the propagation delays to be 39.9 psec and 31.7 for the HL and LH transitions, respectively. The manual results are good considering the many simplifications made during their derivation. Notice especially the overshoots on the simulated output signals. These are caused by the gate-drain capacitances of the inverter transistors, which couple the steep voltage step at the input node directly to the output before the transistors can even start to react to the changes at the input. These overshoots clearly have a negative impact on the performance of the gate, and explain why the simulated delays are larger than the estimations.

WARNING: This example might give the impression that manual analysis always leads to close approximations of the actual response. This is not necessarily the case. Large deviations can often be observed between first- and higher-order models. The purpose of the manual analysis is to get a basic insight in the behavior of the circuit and to determine the dominant parameters. A detailed simulation is indispensable when quantitative data is required. Consider the example above a stroke of good luck.

The obvious question a designer asks herself at this point is how she can manipulate and/or optimize the delay of a gate. To provide an answer to this question, it is necessary to make the parameters governing the delay explicit by expanding R_{eq} in the delay equation. Combining Eq. (5.18) and Eq. (5.17), and assuming for the time being that the channel-length modulation factor λ is ignorable, yields the following expression for t_{pHL} (a similar analysis holds for t_{pLH})

$$t_{pHL} = 0.69\frac{3}{4}\frac{C_L V_{DD}}{I_{DSATn}} = 0.52\frac{C_L V_{DD}}{(W/L)_n k'_n V_{DSATn}(V_{DD} - V_{Tn} - V_{DSATn}/2)} \tag{5.21}$$

In the majority of designs, the supply voltage is chosen high enough so that $V_{DD} >> V_{Tn} + V_{DSATn}/2$. Under these conditions, the delay becomes virtually independent of the supply voltage (Eq. (5.22)). Observe that this is a first-order approximation, and that increasing the supply voltage yields an observable, albeit small, improvement in performance due to a non-zero channel-length modulation factor.

$$t_{pHL} \approx 0.52\frac{C_L}{(W/L)_n k'_n V_{DSATn}} \tag{5.22}$$

This analysis is confirmed in Figure 5.17, which plots the propagation delay of the inverter as a function of the supply voltage. It comes as no surprise that this curve is virtually identical in shape to the one of Figure 3.27, which charts the equivalent on-resistance of the MOS transistor as a function of V_{DD}. While the delay is relative insensitive to supply variations for higher values of V_{DD}, a sharp increase can be observed starting around

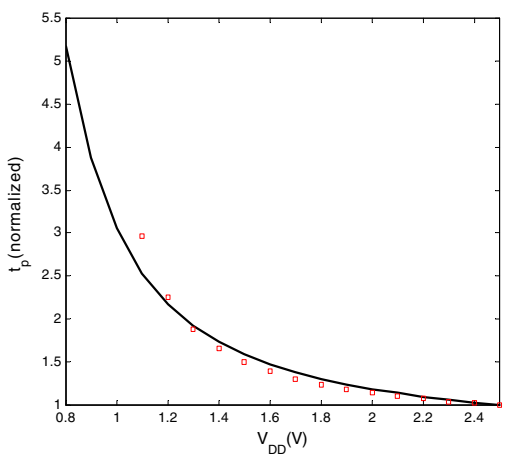


Figure 5.17 Propagation delay of CMOS inverter as a function of supply voltage (normalized with respect to the delay at 2.5 V). The dots indicate the delay values predicted by Eq. (5.21). Observe that this equation is only valid when the devices are velocity-saturated. Hence, the deviation at low supply voltages.

$\approx 2V_T$. This operation region should clearly be avoided if achieving high performance is a premier design goal.

Design Techniques

From the above, we deduce that the propagation delay of a gate can be minimized in the following ways:

- *Reduce* C_L. Remember that three major factors contribute to the load capacitance: the internal diffusion capacitance of the gate itself, the interconnect capacitance, and the fan-out. Careful layout helps to reduce the diffusion and interconnect capacitances. **Good design practice requires keeping the drain diffusion areas as small as possible.**
- *Increase the W/L ratio of the transistors*. This is the most powerful and effective performance optimization tool in the hands of the designer. Proceed however with caution when applying this approach. Increasing the transistor size also raises the diffusion capacitance and hence C_L. In fact, once the intrinsic capacitance (i.e. the diffusion capacitance) starts to dominate the extrinsic load formed by wiring and fanout, increasing the gate size does not longer help in reducing the delay, and only makes the gate larger in area. This effect is called "*self-loading*". In addition, wide transistors have a larger gate capacitance, which increases the fan-out factor of the driving gate and adversely affects its speed.
- *Increase* V_{DD}. As illustrated in Figure 5.17, the delay of a gate can be modulated by modifying the supply voltage. This flexibility allows the designer to trade-off energy dissipation for performance, as we will see in a later section. However, increasing the supply voltage above a certain level yields only very minimal improvement and hence should be avoided. Also, reliability concerns (oxide breakdown, hot-electron effects) enforce firm upper-bounds on the supply voltage in deep sub-micron processes.

Problem 5.4 Propagation Delay as a Function of (dis)charge Current

So far, we have expressed the propagation delay as a function of the equivalent resistance of the transistors. Another approach would be replace the transistor by a current source with value equal to the average (dis)charge current over the interval of interest. Derive an expression of the propagation delay using this alternative approach.

5.4.3 Propagation Delay from a Design Perspective

Some interesting design considerations and trade-off's can be derived from the delay expressions we have derived so far. Most importantly, they lead to a general approach towards transistor sizing that will prove to be extremely useful.

NMOS/PMOS Ratio

So far, we have consistently widened the PMOS transistor so that its resistance matches that of the pull-down NMOS device. This typically requires a ratio of 3 to 3.5 between PMOS and NMOS width. The motivation behind this approach is to create an inverter with a symmetrical VTC, and to equate the high-to-low and low-to-high propagation delays. However, this does not imply that this ratio also yields the minimum overall propagation delay. If symmetry and reduced noise margins are not of prime concern, it is actually possible to speed up the inverter by reducing the width of the PMOS device!

The reasoning behind this statement is that, while widening the PMOS improves the t_{pLH} of the inverter by increasing the charging current, it also degrades the t_{pHL} by cause of a larger parasitic capacitance. When two contradictory effects are present, there must exist a transistor ratio that optimizes the propagation delay of the inverter.

This optimum ratio can be derived through the following simple analysis. Consider two identical, cascaded CMOS inverters. The load capacitance of the first gate equals approximately

$$C_L = (C_{dp1} + C_{dn1}) + (C_{gp2} + C_{gn2}) + C_W \tag{5.23}$$

where C_{dp1} and C_{dn1} are the equivalent drain diffusion capacitances of PMOS and NMOS transistors of the first inverter, while C_{gp2} and C_{gn2} are the gate capacitances of the second gate. C_W represents the wiring capacitance.

When the PMOS devices are made β times larger than the NMOS ones ($\beta = (W/L)_p / (W/L)_n$), all transistor capacitances will scale in approximately the same way, or $C_{dp1} \approx \beta\, C_{dn1}$, and $C_{gp2} \approx \beta\, C_{gn2}$. Eq. (5.23) can then be rewritten:

$$C_L = (1 + \beta)(C_{dn1} + C_{gn2}) + C_W \tag{5.24}$$

An expression for the propagation delay can be derived, based on Eq. (5.20).

$$\begin{aligned} t_p &= \frac{0.69}{2}((1 + \beta)(C_{dn1} + C_{gn2}) + C_W)\left(R_{eqn} + \frac{R_{eqp}}{\beta}\right) \\ &= 0.345((1 + \beta)(C_{dn1} + C_{gn2}) + C_W)R_{eqn}\left(1 + \frac{r}{\beta}\right) \end{aligned} \tag{5.25}$$

r (= R_{eqp}/R_{eqn}) represents the resistance ratio of identically-sized PMOS and NMOS transistors. The optimal value of β can be found by setting $\frac{\partial t_p}{\partial \beta}$ to 0, which yields

$$\beta_{opt} = \sqrt{r\left(1 + \frac{C_w}{C_{dn1} + C_{gn2}}\right)} \tag{5.26}$$

This means that when the wiring capacitance is negligible ($C_{dn1}+C_{gn2} >> C_W$), β_{opt} equals $\sqrt{r}$, in contrast to the factor r normally used in the noncascaded case. If the wiring capacitance dominates, larger values of β should be used. The surprising result of this analysis is that smaller device sizes (and hence smaller design area) yield a faster design at the expense of symmetry and noise margin.

Example 5.6 Sizing of CMOS Inverter Loaded by an Identical Gate

Consider again our standard design example. From the values of the equivalent resistances (Table 3.3), we find that a ratio β of 2.4 (= 31 kΩ / 13 kΩ) would yield a symmetrical transient response. Eq. (5.26) now predicts that the device ratio for an optimal performance should equal 1.6. These results are verified in Figure 5.18, which plots the simulated propagation delay as a function of the transistor ratio β. The graph clearly illustrates how a changing β trades off between t_{pLH} and t_{pHL}. The optimum point occurs around β = 1.9, which is some-

what higher than predicted. Observe also that the rising and falling delays are identical at the predicted point of β equal to 2.4.

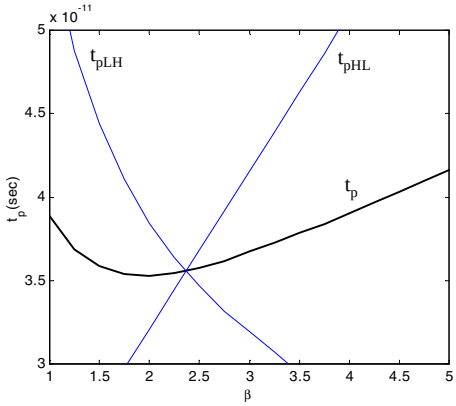


Figure 5.18 Propagation delay of CMOS inverter as a function of the PMOS/NMOS transistor ratio β.

Sizing Inverters for Performance

In this analysis, we assume a symmetrical inverter, this is an inverter where PMOS and NMOS are sized such that the rise and fall delays are identical. The load capacitance of the inverter can be divided into an intrinsic and an extrinsic component, or $C_L = C_{int} + C_{ext}$. C_{int} represents the self-loading or intrinsic output capacitance of the inverter, and is associated with the diffusion capacitances of the NMOS and PMOS transistors as well as the gate-drain overlap (Miller) capacitances. C_{ext} is the extrinsic load capacitance, attributable to fanout and wiring capacitance. Assuming that R_{eq} stands for the equivalent resistance of the gate, we can express the propagation delay as follows

$$\begin{aligned} t_p &= 0.69 R_{eq}(C_{int} + C_{ext}) \\ &= 0.69 R_{eq} C_{int}(1 + C_{ext}/C_{int}) = t_{p0}(1 + C_{ext}/C_{int}) \end{aligned} \tag{5.27}$$

$t_{p0} = 0.69\ R_{eq}C_{int}$ represents the delay of the inverter only loaded by its own intrinsic capacitance ($C_{ext} = 0$), and is called the *intrinsic or unloaded delay*.

The next question is how transistor sizing impacts the performance of the gate. To do so, we must establish the relationship between the various parameters in Eq. (5.27) and the sizing factor S, which relates the transistor sizes of our inverter to a reference gate—typically a minimum-sized inverter. The intrinsic capacitance C_{int} consists of the diffusion and Miller capacitances, both of which are proportional to the width of the transistors. Hence, $C_{int} = SC_{iref}$. The resistance of the gate relates to the reference gate as $R_{eq} = R_{ref}/S$. We can now rewrite Eq. (5.27),

$$\begin{aligned} t_p &= 0.69(R_{ref}/S)(SC_{iref})(1 + C_{ext}/(SC_{iref})) \\ &= 0.69 R_{ref} C_{iref}\left(1 + \frac{C_{ext}}{SC_{iref}}\right) = t_{p0}\left(1 + \frac{C_{ext}}{SC_{iref}}\right) \end{aligned} \tag{5.28}$$

This leads to two important conclusions:

- The intrinsic delay of the inverter t_{p0} is independent of the sizing of the gate, and is purely determined by technology and inverter layout. When no load is present, an increase in the drive of the gate is totally offset by the increased capacitance.
- Making S infinitely large yields the maximum obtainable performance gain, eliminating the impact of any external load, and reducing the delay to the intrinsic one. Yet, any sizing factor S that is sufficiently larger than (C_{ext}/C_{int}) produces similar results at a substantial gain in silicon area.

Example 5.7 Device Sizing for Performance

Let us explore the performance improvement that can be obtained by device sizing in the design of Example 5.5. We find from Table 5.2 that $C_{int}/C_{ext} \approx 1.05$ (C_{int} = 3.0 fF, C_{ext} = 3.15 fF). This would predict a maximum performance gain of 2.05. A scaling factor of 10 allows us to get within 10% of this optimal performance, while larger device sizes only yield ignorable performance gains.

This is confirmed by simulation results, which predict a maximum obtainable performance improvement of 1.9 (t_{p0} = 19.3 psec). From the graph of Figure 5.19, we observe that the bulk of the improvement is already obtained for S = 5, and that sizing factors larger than 10 barely yield any extra gain.

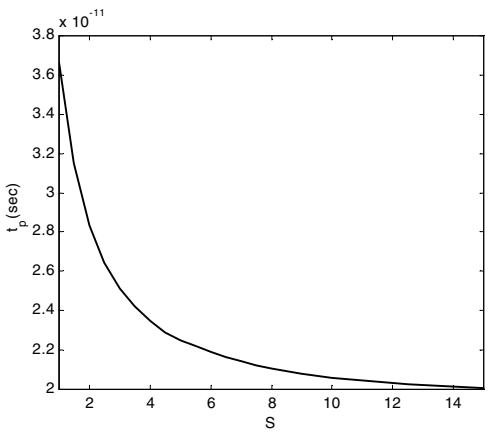


Figure 5.19 Increasing inverter performance by sizing the NMOS and PMOS transistor with an identical factor S for a fixed fanout (inverter of Figure 5.15).

Sizing A Chain of Inverters

While sizing up an inverter reduces its delay, it also increases its input capacitance. Gate sizing in an isolated fashion without taking into account its impact on the delay of the preceding gates is a purely academic enterprise. Therefore, a more relevant problem is determining the optimum sizing of a gate when embedded in a real environment. A simple chain of inverters is a good first case to study. To determine the input loading effect, the relationship between the input gate capacitance C_g and the intrinsic output capacitance of the inverter has to be established. Both are proportional to the gate sizing. Hence, the following relationship holds, independent of gate sizing

$$C_{int} = \gamma C_g \tag{5.29}$$

γ is a proportionality factor, which is only a function of technology and is close to 1 for most sub-micron processes. Rewriting Eq. (5.28),

$$t_p = t_{p0}\left(1 + \frac{C_{ext}}{\gamma C_g}\right) = t_{p0}(1 + f/\gamma) \tag{5.30}$$

shows that the delay if the inverter is only a function of the ratio between its external load capacitance and input capacitance. This ratio is called the *effective fanout f*.

Let us consider the circuit of Eq. Figure 5.20. The goal is to minimize the delay through the inverter chain, with the input capacitance of the first inverter C_{g1}—typically a minimally-sized device— and the load capacitance C_L fixed.

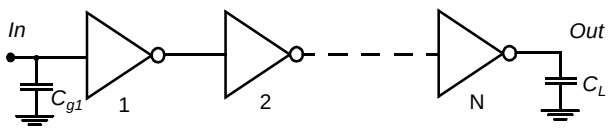


Figure 5.20 Chain of *N* inverters with fixed input and output capacitance.

Given the delay expression for the j-th inverter stage,[2]

$$t_{p,j} = t_{p0}\left(1 + \frac{C_{g,j+1}}{\gamma C_{g,j}}\right) = t_{p0}(1 + f_j/\gamma) \tag{5.31}$$

we can derive the total delay of the chain.

$$t_p = \sum_{j=1}^{N} t_{p,j} = t_{p0}\sum_{j=1}^{N}\left(1 + \frac{C_{g,j+1}}{\gamma C_{g,j}}\right), \text{ with } C_{g,N+1} = C_L \tag{5.32}$$

This equation has *N*-1 unknowns, being $C_{g,2}$, $C_{g,3}$, …, $C_{g,N}$. The minimum delay can be found by taking *N*-1 partial derivatives, and equating them to 0, or $\partial t_p/\partial C_{g,j} = 0$. The result is a set of constraints, $C_{g,j+1}/C_{g,j} = C_{g,j}/C_{g,j-1}$. In other words, the optimum size of each inverter is the geometric mean of its neighbors sizes,

$$C_{g,j} = \sqrt{C_{g,j-1}C_{g,j+1}}\,. \tag{5.33}$$

Overall, this means that each inverter is sized up by the same factor *f* with respect to the preceding gate, has the same effective fanout ($f_j = f$), and hence the same delay. With $C_{g,1}$ and C_L given, we can derive the sizing factor,

$$f = \sqrt[N]{C_L/C_{g,1}} = \sqrt[N]{F} \tag{5.34}$$

and the minimum delay through the chain,

$$t_p = Nt_{p0}(1 + \sqrt[N]{F}/\gamma)\,. \tag{5.35}$$

[2] This expression ignores the wiring capacitance, which is a fair assumption for the time being.

F represents the *overall effective fanout* of the circuit, and equals $C_L/C_{g,1}$. Observe how the relationship between t_p and F is a very strong function of the number of stages. As expected, the relationship is linear when only 1 stage is present. Introducing a second stage turns it into square root, and so on. The obvious question now is how to choose the number of stages so that the delay is minimized for a given value of F.

Choosing the Right Number of Stages in an Inverter Chain

Evaluation of Eq. (5.35) reveals the trade-off's in choosing the number of stages for a given $F\ (=f^N)$. When the number of stages is too large, the first component of the equation, representing the intrinsic delay of the stages, becomes dominant. If the number of stages is too small, the effective fanout of each stage becomes large, and the second component is dominant. The optimum value can be found by differentiating the minimum delay expression by the number of stages, and setting the result to 0.

$$\gamma + \sqrt[N]{F} - \frac{\sqrt[N]{F}\ln F}{N} = 0$$

or equivalently

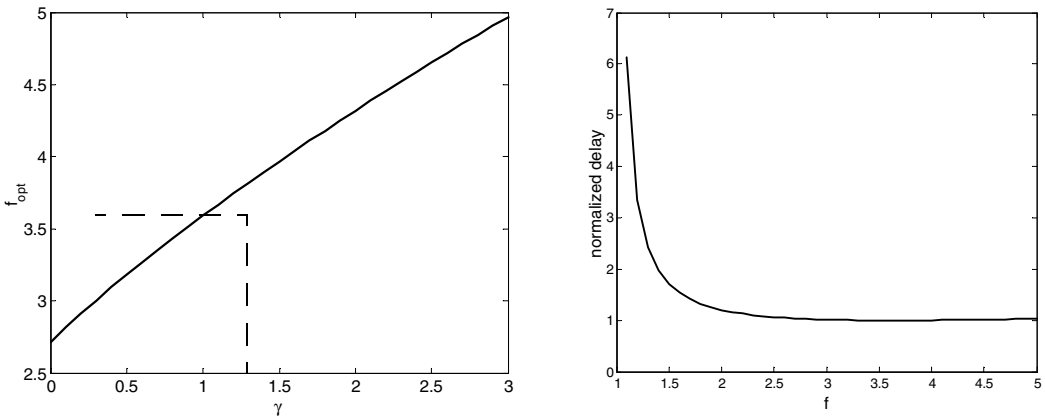
$$f = e^{(1+\gamma/f)} \tag{5.36}$$

This equation only has a closed-form solution for $\gamma = 0$, this is when the self-loading is ignored and the load capacitance only consists of the fanout. Under these simplified conditions, it is found that the optimal number of stages equals $N = \ln(F)$, and the effective fanout of each stage is set to $f = 2.71828 = e$. This optimal buffer design scales consecutive stages in an exponential fashion, and is hence called an exponential horn [Mead79]. When self-loading is included, Eq. (5.36) can only be solved numerically. The results are plotted in Figure 5.21a. For the typical case of $\gamma \approx 1$, the optimum scaler factor turns out to be close to 3.6. Figure 5.21b plots the (normalized) propagation delay of the inverter chain as a function of the effective fanout for $\gamma = 1$. Choosing values of the fanout that are higher than the optimum does not impact the delay that much, and reduces the required number of buffer stages and the implementation area. A common practice is to select an optimum fanout of 4. The use of too many stages ($f < f_{opt}$), on the other hand, has a substantial negative impact on the delay, and should be avoided.

Example 5.8 The Impact of Introducing Buffer Stages

Table 5.3 enumerates the values of $t_{p,opt}/t_{p0}$ for the unbuffered design, the dual stage, and optimized inverter chain for a variety of values of F (for $\gamma = 1$). Observe the impressive speed-up obtained with cascaded inverters when driving very large capacitive loads.

The above analysis can be extended to not only cover chains of inverters, but also networks of inverters that contain actual fanout, an example of which is shown in Figure 5.22. We solely have to adjusting the expression for C_{ext} to incorporate the additional fanout factors.



(a) Optimum effective fanout f (or inverter scaling factor) as a function of the self-loading factor γ in an inverter chain.

(b) Normalized propagation delay ($t_p/(t_{popt})$) as a function of the effective fanout f for γ=1.

Figure 5.21 Optimizing the number of stages in an inverter chain.

Table 5.3 t_{opt}/t_{p0} versus x for various driver configurations.

F	Unbuffered	Two Stage	Inverter Chain
10	11	8.3	8.3
100	101	22	16.5
1000	1001	65	24.8
10,000	10,001	202	33.1

Problem 5.5 Sizing an Inverter Network

Determine the sizes of the inverters in the circuit of Figure 5.22, such that the delay between nodes *Out* and *In* is minimized. You may assume that $C_L = 64\ C_{g,1}$.

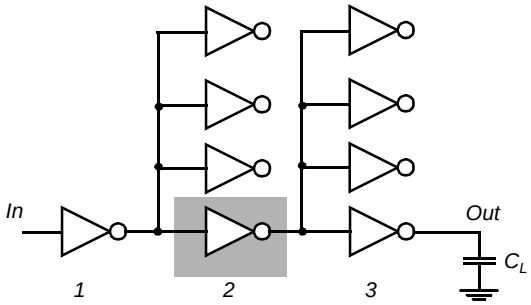


Figure 5.22 Inverter network, in which each gate has a fanout of 4 gates, distributing a single input to 16 output signals in a tree-like fashion.

Hints: Determine first the ratio's between the devices that minimize the delay. You should find that the following must hold,

$$\frac{4C_{g,2}}{C_{g,1}} = \frac{4C_{g,3}}{C_{g,2}} = \frac{C_L}{C_{g,3}}$$

Finding the actual gate sizes ($C_{g,3} = 2.52C_{g,2} = 6.35C_{g,1}$) is a relatively straightforward task. Straightforward sizing of the inverter chain, without taking the fanout into account, would have led to a sizing factor of 4 instead of 2.52.

The rise/fall time of the input signal

All the above expressions were derived under the assumption that the input signal to the inverter abruptly changed from 0 to V_{DD} or vice-versa. Only one of the devices is assumed to be on during the (dis)charging process. In reality, the input signal changes gradually and, temporarily, PMOS and NMOS transistors conduct simultaneously. This affects the total current available for (dis)charging and impacts the propagation delay. Figure 5.23 plots the propagation delay of a minimum-size inverter as a function of the input signal slope—as obtained from SPICE. It can be observed that t_p increases (approximately) linearly with increasing input slope, once $t_s > t_p(t_s=0)$.

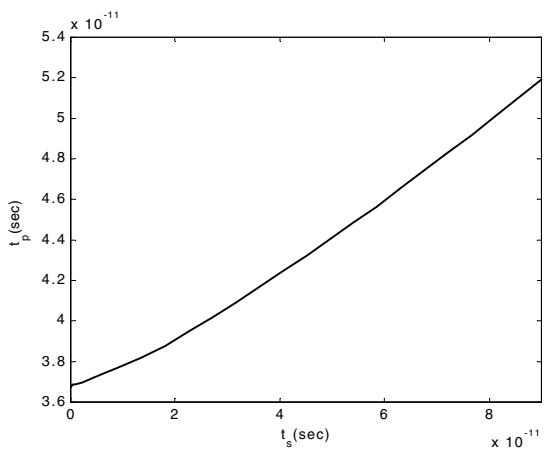


Figure 5.23 t_p as a function of the input signal slope (10-90% rise or fall time) for minimum-size inverter with fan-out of a single gate.

While it is possible to derive an analytical expression describing the relationship between input signal slope and propagation delay, the result tends to be complex and of limited value. From a design perspective, it is more valuable to relate the impact of the finite slope on the performance directly to its cause, which is the limited driving capability of the preceding gate. If the latter would be infinitely strong, its output slope would be zero, and the performance of the gate under examination would be unaffected. The strength of this approach is that it realizes that a gate is never designed in isolation, and that its performance is both affected by the fanout, and the driving strength of the gate(s) feeding into its inputs. This leads to a revised expression for the propagation delay of an inverter *i* in a chain of inverters [Hedenstierna87]:

$$t_p^i = t_{step}^i + \eta t_{step}^{i-1} \tag{5.37}$$

Eq. (5.37) states that the propagation delay of inverter *i* equals the sum of the delay of the same gate for a step input (t_{step}^i) (i.e. zero input slope) augmented with a fraction of the step-input delay of the preceding gate (*i*-1). The fraction η is an empirical constant, which typically has values around 0.25. This expression has the advantage of being very simple, while exposing all relationships necessary for global delay computations of complex circuits.

Example 5.9 Delay of Inverter embedded in Network

Consider for instance the circuit of Figure 5.22. With the aid of Eq. (5.31) and Eq. (5.37), we can derive an expression for the delay of the stage-2 inverter, marked by the gray box.

$$t_{p,2} = t_{p0}\left(1 + \frac{4C_{g,3}}{\gamma C_{g,2}}\right) + \eta t_{p0}\left(1 + \frac{4C_{g,2}}{\gamma C_{g,1}}\right)$$

An analysis of the overall propagation delay in the style of Problem 5.5, leads to the following revised sizing requirements for minimum delay,

$$\frac{4(1+\eta)C_{g,2}}{C_{g,1}} = \frac{4(1+\eta)C_{g,3}}{C_{g,2}} = \frac{C_L}{C_{g,3}}$$

or $f_2 = f_1 = 2.47$ (assuming $\eta = 0.25$).

Design Challenge

It is advantageous to keep the signal rise times smaller than or equal to the gate propagation delays. This proves to be true not only for performance, but also for power consumption considerations as will be discussed later. Keeping the rise and fall times of the signals small and of approximately equal values is one of the major challenges in high-performance design, and is often called '*slope engineering*'.

Problem 5.6 Impact of input slope

Determine if reducing the supply voltage increases or decreases the influence of the input signal slope on the propagation delay. Explain your answer.

Delay in the Presence of (Long) Interconnect Wires

The interconnect wire has played a minimal role in our analysis so far. When gates get farther apart, the wire capacitance and resistance can no longer be ignored, and may even dominate the transient response. Earlier delay expressions can be adjusted to accommodate these extra contributions by employing the wire modeling techniques introduced in

the previous Chapter. The analysis detailed in Example 4.9 is directly applicable to the problem at hand. Consider the circuit of Figure 5.24, where an inverter drives a single fanout through a wire of length L. The driver is represented by a single resistance R_{dr}, which is the average between R_{eqn} and R_{eqp}. C_{int} and C_{fan} account for the intrinsic capacitance of the driver, and the input capacitance of the fanout gate, respectively.

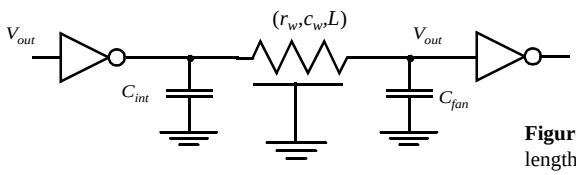


Figure 5.24 Inverter driving single fanout through wire of length L.

The propagation delay of the circuit can be obtained by applying the Ellmore delay expression.

$$\begin{aligned} t_p &= 0.69R_{dr}C_{int} + (0.69R_{dr} + 0.38R_w)C_w + 0.69(R_{dr} + R_w)C_{fan} \\ &= 0.69R_{dr}(C_{int} + C_{fan}) + 0.69(R_{dr}c_w + r_wC_{fan})L + 0.38r_wc_wL^2 \end{aligned} \tag{5.38}$$

The 0.38 factor accounts for the fact that the wire represents a distributed delay. C_w and R_w stand for the total capacitance and resistance of the wire, respectively. The delay expressions contains a component that is linear with the wire length, as well a quadratic one. It is the latter that causes the wire delay to rapidly become the dominant factor in the delay budget for longer wires.

Example 5.10 Inverter delay in presence of interconnect

Consider the circuit of Figure 5.24, and assume the device parameters of Example 5.5: C_{int} = 3 fF, C_{fan} = 3 fF, and R_{dr} = 0.5(13/1.5 + 31/4.5) = 7.8 kΩ. The wire is implemented in metal1 and has a width of 0.4 μm—the minimum allowed. This yields the following parameters: c_w = 92 aF/μm, and r_w = 0.19 Ω/μm (Example 4.4). With the aid of Eq. (5.38), we can compute at what wire length the delay of the interconnect becomes equal to the intrinsic delay caused purely by device parasitics. Solving the following quadratic equation yields a single (meaningful) solution.

$$6.6 \times 10^{-18}L^2 + 0.5 \times 10^{-12}L = 32.29 \times 10^{-12}$$

or

$$L = 65\ \mu\text{m}$$

Observe that the extra delay is solely due to the linear factor in the equation, and more specifically due to the extra capacitance introduced by the wire. The quadratic factor (this is, the distributed wire delay) only becomes dominant at much larger wire lengths (> 7 cm). This is due to the high resistance of the (minimum-size) driver transistors. A different balance emerges when wider transistors are used. Analyze, for instance, the same problem with the driver transistors 100 times wider, as is typical for high-speed, large fan-out drivers.

5.5 Power, Energy, and Energy-Delay

So far, we have seen that the static CMOS inverter with its almost ideal VTC—symmetrical shape, full logic swing, and high noise margins—offers a superior robustness, which simplifies the design process considerably and opens the door for design automation. Another major attractor for static CMOS is the almost complete absence of power consumption in steady-state operation mode. It is this combination of robustness and low static power that has made static CMOS the technology of choice of most contemporary digital designs. The power dissipation of a CMOS circuit is instead dominated by the dynamic dissipation resulting from charging and discharging capacitances.

5.5.1 Dynamic Power Consumption

Dynamic Dissipation due to Charging and Discharging Capacitances

Each time the capacitor C_L gets charged through the PMOS transistor, its voltage rises from 0 to V_{DD}, and a certain amount of energy is drawn from the power supply. Part of this energy is dissipated in the PMOS device, while the remainder is stored on the load capacitor. During the high-to-low transition, this capacitor is discharged, and the stored energy is dissipated in the NMOS transistor.[3]

A precise measure for this energy consumption can be derived. Let us first consider the low-to-high transition. We assume, initially, that the input waveform has zero rise and fall times, or, in other words, that the NMOS and PMOS devices are never on simultaneously. Therefore, the equivalent circuit of Figure 5.25 is valid. The values of the energy E_{VDD}, taken from the supply during the transition, as well as the energy E_C, stored on the capacitor at the end of the transition, can be derived by integrating the instantaneous power over the period of interest. The corresponding waveforms of $v_{out}(t)$ and $i_{VDD}(t)$ are pictured in Figure 5.26.

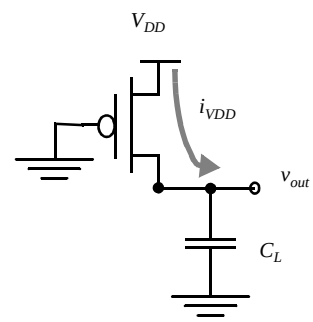


Figure 5.25 Equivalent circuit during the low-to-high transition.

$$E_{VDD} = \int_0^{\infty} i_{VDD}(t) V_{DD} dt = V_{DD} \int_0^{\infty} C_L \frac{dv_{out}}{dt} dt = C_L V_{DD} \int_0^{V_{DD}} dv_{out} = C_L V_{DD}^2 \qquad (5.39)$$

[3] Observe that this model is a simplification of the actual circuit. In reality, the load capacitance consists of multiple components some of which are located between the output node and GND, others between output node and V_{DD}. The latter experience a charge-discharge cycle that is out of phase with the capacitances to GND, i.e. they get charged when V_{out} goes low and discharged when V_{out} rises. While this distributes the energy delivery by the supply over the two phases, it does not impact the overall dissipation, and the results presented in this section are still valid.

$$E_C = \int_0^\infty i_{VDD}(t) v_{out} dt = \int_0^\infty C_L \frac{dv_{out}}{dt} v_{out} dt = C_L \int_0^{V_{DD}} v_{out} dv_{out} = \frac{C_L V_{DD}^2}{2} \tag{5.40}$$

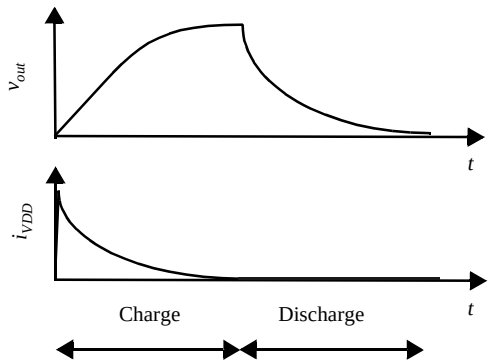


Figure 5.26 Output voltages and supply current during (dis)charge of C_L.

These results can also be derived by observing that during the low-to-high transition, C_L is loaded with a charge $C_L V_{DD}$. Providing this charge requires an energy from the supply equal to $C_L V_{DD}^2$ $(= Q \times V_{DD})$. The energy stored on the capacitor equals $C_L V_{DD}^2/2$. This means that only half of the energy supplied by the power source is stored on C_L. The other half has been dissipated by the PMOS transistor. Notice that this energy dissipation is independent of the size (and hence the resistance) of the PMOS device! During the discharge phase, the charge is removed from the capacitor, and its energy is dissipated in the NMOS device. Once again, there is no dependence on the size of the device. In summary, each switching cycle (consisting of an L→H and an H→L transition) takes a fixed amount of energy, equal to $C_L V_{DD}^2$. In order to compute the power consumption, we have to take into account how often the device is switched. If the gate is switched **on and off** $f_{0\to1}$ times per second, the power consumption equals

$$P_{dyn} = C_L V_{DD}^2 f_{0\to1} \tag{5.41}$$

$f_{0\to1}$ represents the frequency of energy-consuming transitions, this is $0 \to 1$ transitions for static CMOS.

Advances in technology result in ever-higher of values of $f_{0\to1}$ (as t_p decreases). At the same time, the total capacitance on the chip (C_L) increases as more and more gates are placed on a single die. Consider for instance a 0.25 μm CMOS chip with a clock rate of 500 Mhz and an average load capacitance of 15 fF/gate, assuming a fanout of 4. The power consumption per gate for a 2.5 V supply then equals approximately 50 μW. For a design with 1 million gates and assuming that a transition occurs at every clock edge, this would result in a power consumption of 50 W! This evaluation presents, fortunately, a pessimistic perspective. In reality, not all gates in the complete IC switch at the full rate of 500 Mhz. The actual activity in the circuit is substantially lower.

Example 5.11 Capacitive power dissipation of inverter

The capacitive dissipation of the CMOS inverter of Example 5.4 is now easily computed. In Table 5.2, the value of the load capacitance was determined to equal 6 fF. For a supply voltage of 2.5 V, the amount of energy needed to charge and discharge that capacitance equals

$$E_{dyn} = C_L V_{DD}^2 = 37.5 \text{ fJ}$$

Assume that the inverter is switched at the maximum possible rate ($T = 1/f = t_{pLH} + t_{pHL} = 2\,t_p$). For a t_p of 32.5 psec (Example 5.5), we find that the dynamic power dissipation of the circuit is

$$P_{dyn} = E_{dyn}/(2t_p) = 580\ \mu\text{W}$$

Of course, an inverter in an actual circuit is rarely switched at this maximum rate, and even if done so, the output does not swing from rail-to-rail. The power dissipation will hence be substantially lower. For a rate of 4 GHz (T = 250 psec), the dissipation reduces to 150 μW. This is confirmed by simulations, which yield a power consumption of 155 μW.

Computing the dissipation of a complex circuit is complicated by the $f_{0\to1}$ factor, also called the *switching activity*. While the switching activity is easily computed for an inverter, it turns out to be far more complex in the case of higher-order gates and circuits. One concern is that the switching activity of a network is a function of the nature and the statistics of the input signals: If the input signals remain unchanged, no switching happens, and the dynamic power consumption is zero! On the other hand, rapidly changing signals provoke plenty of switching and hence dissipation. Other factors influencing the activity are the overall network topology and the function to be implemented. We can accommodate this by another rewrite of the equation, or

$$P_{dyn} = C_L V_{DD}^2 f_{0\to1} = C_L V_{DD}^2 P_{0\to1} f = C_{EFF} V_{DD}^2 f \tag{5.42}$$

where f now presents the maximum possible event rate of the inputs (which is often the clock rate) and $P_{0\to1}$ the probability that a clock event results in a $0 \to 1$ (or power-consuming) event at the output of the gate. $C_{EFF} = P_{0\to1}C_L$ is called the *effective capacitance* and represents the average capacitance switched every clock cycle. For our example, an activity factor of 10% ($P_{0\to1} = 0.1$) reduces the average consumption to 5 W.

Example 5.12 Switching activity

Consider the waveforms on the right where the upper waveform represents the idealized clock signal, and the bottom one shows the signal at the output of the gate. Power consuming transitions occur 2 out of 8 times, which is equivalent to a transition probability of 0.25 (or 25%).

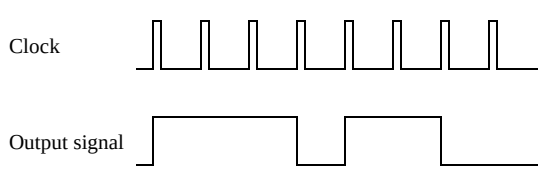


Figure 5.27 Clock and signal waveforms

Low Energy/Power Design Techniques

With the increasing complexity of the digital integrated circuits, it is anticipated that the power problem will only worsen in future technologies. This is one of the reasons that lower supply

voltages are becoming more and more attractive. **Reducing V_{DD} has a quadratic effect on P_{dyn}.** For instance, reducing V_{DD} from 2.5 V to 1.25 V for our example drops the power dissipation from 5 W to 1.25 W. This assumes that the same clock rate can be sustained. Figure 5.17 demonstrates that this assumption is not that unrealistic as long as the supply voltage is substantially higher than the threshold voltage. An important performance penalty occurs once V_{DD} approaches 2 V_T.

When a lower bound on the supply voltage is set by external constraints (as often happens in real-world designs), or when the performance degradation due to lowering the supply voltage is intolerable, the only means of reducing the dissipation is by lowering the effective capacitance. This can be achieved by addressing both of its components: the physical capacitance and the switching activity.

A *reduction in the switching activity* can only be accomplished at the logic and architectural abstraction levels, and will be discussed in more detail in later Chapters. *Lowering the physical capacitance* is an overall worthwhile goal, which also helps to improve the performance of the circuit. As most of the capacitance in a combinational logic circuit is due to transistor capacitances (gate and diffusion), it makes sense to keep those contributions to a minimum when designing for low power. This means that transistors should be kept to *minimal size* whenever possible or reasonable. This definitely affects the performance of the circuit, but the effect can be offset by using logic or architectural speed-up techniques. The only instances where transistors should be sized up is when the load capacitance is dominated by extrinsic capacitances (such as fan-out or wiring capacitance). This is contrary to common design practices used in cell libraries, where transistors are generally made large to accommodate a range of loading and performance requirements.

The above observations lead to an interesting design challenge. Assume we have to minimize the energy dissipation of a circuit with a specified lower-bound on the performance. An attractive approach is to lower the supply voltage as much as possible, and to compensate the loss in performance by increasing the transistor sizes. Yet, the latter causes the capacitance to increase. It may be foreseen that at a low enough supply voltage, the latter factor may start to dominate and cause energy to increase with a further drop in the supply voltage.

Example 5.13 Transistor Sizing for Energy Minimization

To analyze the transistor-sizing for minimum energy problem, we examine the simple case of a static CMOS inverter driving an external load capacitance C_{ext}. To take the input loading effects into account, we assume that the inverter itself is driven by a minimum-sized device (Figure 5.28). The goal is to minimize the energy dissipation of the complete circuit, while maintaining a lower-bound on performance. The degrees of freedom are the size factor f of the inverter and the supply voltage V_{dd} of the circuit. The propagation delay of the optimized circuit should not be larger than that of a reference circuit, chosen to have as parameters $f = 1$ and $V_{dd} = V_{ref}$.

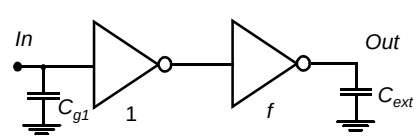


Figure 5.28 CMOS inverter driving an external load capacitance C_{ext}, while being driven by a minimum sized gate.

Using the approach introduced in Section 5.4.3 (*Sizing a Chain of Inverters*), we can derive an expression for the propagation delay of the circuit,

$$t_p = t_{p0}\left(\left(1+\frac{f}{\gamma}\right)+\left(1+\frac{F}{f\gamma}\right)\right) \tag{5.43}$$

with $F = (C_{ext}/C_{g1})$ the overall effective fanout of the circuit t_{p0} is the intrinsic delay of the inverter. Its dependence upon V_{DD} is approximated by the following expression, derived from Eq. (5.21).

$$t_{p0} \sim \frac{V_{DD}}{V_{DD}-V_{TE}} \tag{5.44}$$

The energy dissipation for a single transition at the input is easily found once the total capacitance of the circuit is known, or

$$E = V_{dd}^2 C_{g1}((1+\gamma))(1+f)+F) \tag{5.45}$$

The performance constraint now states that the propagation delay of the scaled circuit should be equal (or smaller) to the delay of the reference circuit (f=1, $V_{dd} = V_{ref}$). To simplify the subsequent analysis, we make the simplifying assumption that the intrinsic output capacitance of the gate equals its gate capacitance, or $\gamma = 1$. Hence,

$$\frac{t_p}{t_{pref}} = \frac{t_{p0}\left(2+f+\frac{F}{f}\right)}{t_{p0ref}(3+F)} = \left(\frac{V_{DD}}{V_{ref}}\right)\left(\frac{V_{ref}-V_{TE}}{V_{DD}-V_{TE}}\right)\left(\frac{2+f+\frac{F}{f}}{3+F}\right) = 1 \tag{5.46}$$

Eq. (5.46) establishes a relationship between the sizing factor f and the supply voltage, plotted in Figure 5.29a for different values of F. Those curves show a clear minimum. Increasing the size of the inverter from the minimum initially increases the performance, and hence allows for a lowering of the supply voltage. This is fruitful until the optimum sizing factor of $f = \sqrt{F}$ is reached, which should not surprise careful readers of the previous sections. Further increases in the device sizes only increase the self-loading factor, deteriorate the performance, and require an increase in supply voltage. Also observe that for the case of F=1, the reference case is the best solution; any resizing just increases the self-loading.

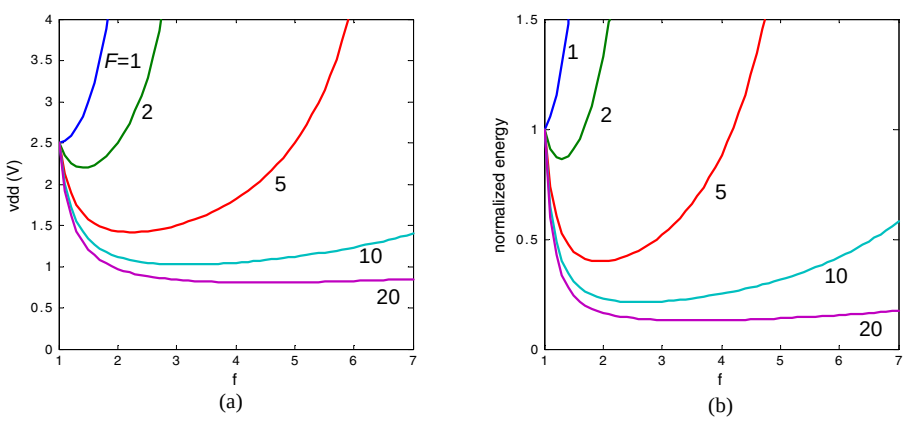


Figure 5.29 Sizing of an inverter for energy-minimization. (a) Required supply voltage as a function of the sizing factor f for different values of the overall effective fanout F; (b) Energy of scaled circuit (normalized with respect to the reference case) as a function of f. V_{ref} = 2.5V, V_{TE} = 0.5V.

With the $V_{DD}(f)$ relationship in hand, we can derive the energy of the scaled circuit (normalized with respect to the reference circuit) as a function of the sizing factor f.

$$\frac{E}{E_{ref}} = \left(\frac{V_{DD}}{V_{ref}}\right)^2 \left(\frac{2 + 2f + F}{4 + F}\right) \tag{5.47}$$

Finding an analytical expression for the optimal sizing factor is possible, but yields a complex and messy equation. A graphical approach is just as effective. The resulting charts are plotted in Figure 5.29b, from which a number of conclusions can be drawn:

- **Device sizing, combined with supply voltage reduction, is a very effective approach in reducing the energy consumption of a logic network.** This is especially true for networks with large effective fanouts, where energy reductions with almost a factor of 10 can be observed. But the gain is also sizable for smaller values of F. The only exception is the F=1 case, where the minimum size device is also the most effective one.
- Oversizing the transistors beyond the optimal value comes at a hefty price in energy. This is unfortunately a common approach in many of today's designs.
- The optimal sizing factor for energy is smaller than the one for performance, especially for large values of F. For example, for a fanout of 20, f_{opt}(energy) = 3.53, while f_{opt}(performance) = 4.47. Increasing the device sizes only leads to a minimal supply reduction once V_{DD} starts approaching V_{TE}, hence leading to very minimal energy gains.

Dissipation Due to Direct-Path Currents

In actual designs, the assumption of the zero rise and fall times of the input wave forms is not correct. The finite slope of the input signal causes a direct current path between V_{DD} and *GND* for a short period of time during switching, while the NMOS and the PMOS transistors are conducting simultaneously. This is illustrated in Figure 5.30. Under the (reasonable) assumption that the resulting current spikes can be approximated as triangles and that the inverter is symmetrical in its rising and falling responses, we can compute the energy consumed per switching period,

$$E_{dp} = V_{DD}\frac{I_{peak}t_{sc}}{2} + V_{DD}\frac{I_{peak}t_{sc}}{2} = t_{sc}\,V_{DD}I_{peak} \tag{5.48}$$

as well as the average power consumption

$$P_{dp} = t_{sc}\,V_{DD}I_{peak}\,f = C_{sc}\,V_{DD}^2 f \tag{5.49}$$

The direct-path power dissipation is proportional to the switching activity, similar to the capacitive power dissipation. t_{sc} represents the time both devices are conducting. For a linear input slope, this time is reasonably well approximated by Eq. (5.50) where t_s represents the 0-100% transition time.

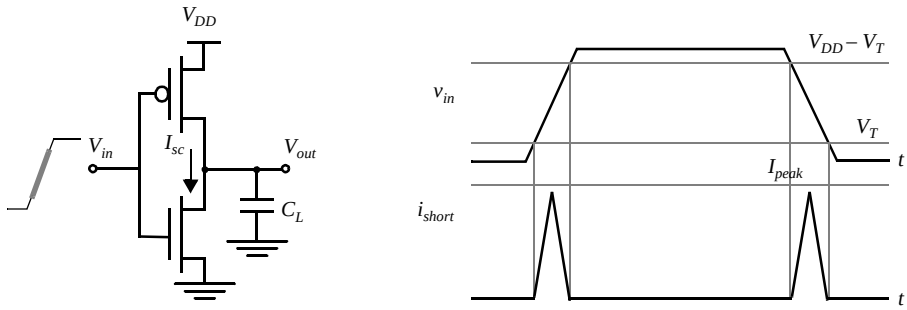
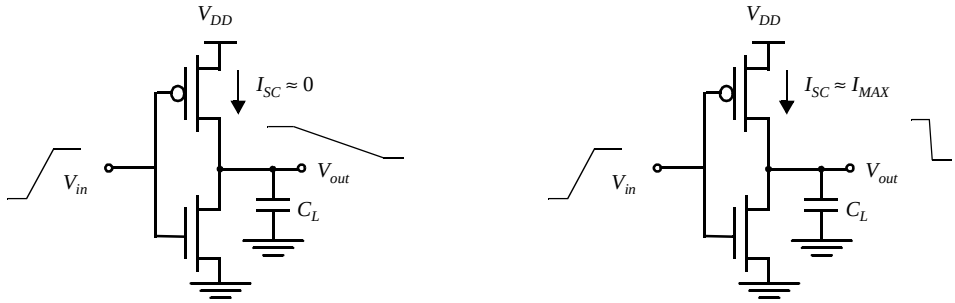


Figure 5.30 Short-circuit currents during transients.

$$t_{sc} = \frac{V_{DD} - 2V_T}{V_{DD}} t_s \approx \frac{V_{DD} - 2V_T}{V_{DD}} \times \frac{t_{r(f)}}{0.8} \tag{5.50}$$

I_{peak} is determined by the saturation current of the devices and is hence directly proportional to the sizes of the transistors. The peak current is also **a strong function of the ratio between input and output slopes**. This relationship is best illustrated by the following simple analysis: Consider a static CMOS inverter with a $0 \rightarrow 1$ transition at the input. Assume first that the load capacitance is very large, so that the output fall time is significantly larger than the input rise time (Figure 5.31a). Under those circumstances, the input



(a) Large capacitive load (b) Small capacitive load

Figure 5.31 Impact of load capacitance on short-circuit current.

moves through the transient region before the output starts to change. As the source-drain voltage of the PMOS device is approximately 0 during that period, the device shuts off without ever delivering any current. The short-circuit current is close to zero in this case. Consider now the reverse case, where the output capacitance is very small, and the output fall time is substantially smaller than the input rise time (Figure 5.31b). The drain-source voltage of the PMOS device equals V_{DD} for most of the transition period, guaranteeing the maximal short-circuit current (equal to the saturation current of the PMOS). This clearly

represents the worst-case condition. The conclusions of the above analysis are confirmed in Figure 5.32, which plots the short-circuit current through the NMOS transistor during a low-to-high transition as a function of the load capacitance.

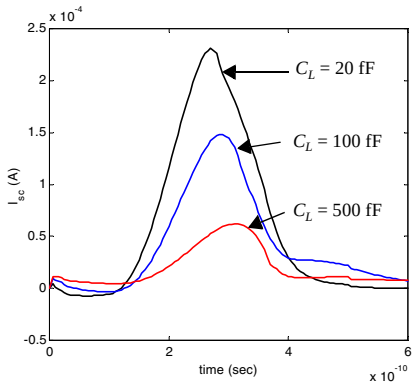


Figure 5.32 CMOS inverter short-circuit current through NMOS transistor as a function of the load capacitance (for a fixed input slope of 500 psec).

This analysis leads to the conclusion that the short-circuit dissipation is minimized by making the output rise/fall time larger than the input rise/fall time. On the other hand, making the output rise/fall time too large slows down the circuit and can cause short-circuit currents in the fan-out gates. This presents a perfect example of how local optimization and forgetting the global picture can lead to an inferior solution.

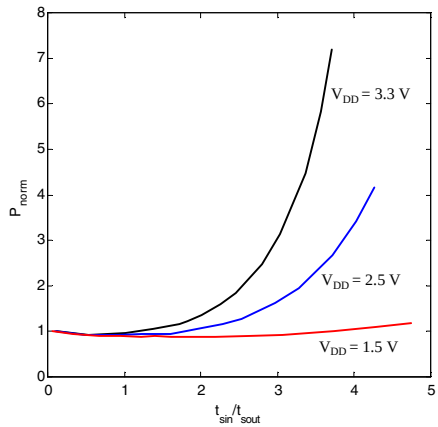
Design Techniques

A more practical rule, which optimizes the power consumption in a global way, can be formulated (Veendrick84]):

> The power dissipation due to short-circuit currents is minimized by matching the rise/fall times of the input and output signals. At the overall circuit level, this means that rise/fall times of all signals should be kept constant within a range.

Making the input and output rise times of a gate identical is not the optimum solution for that particular gate on its own, but keeps the overall short-circuit current within bounds. This is shown in Figure 5.33, which plots the short-circuit energy dissipation of an inverter (normalized with respect to the zero-input rise time dissipation) as a function of the ratio r between input and output rise/fall times. When the load capacitance is too small for a given inverter size ($r > 2\ldots3$ for $V_{DD} = 5$ V), the power is dominated by the short-circuit current. For very large capacitance values, all power dissipation is devoted to charging and discharging the load capacitance. When the rise/fall times of inputs and outputs are equalized, most power dissipation is associated with the dynamic power and only a minor fraction (< 10%) is devoted to short-circuit currents.

Observe also that the impact of **short-circuit current is reduced when we lower the supply voltage**, as is apparent from Eq. (5.50). In the extreme case, when $V_{DD} < V_{Tn} + |V_{Tp}|$, short-circuit dissipation is completely eliminated, because both devices are never on simultaneously. With threshold voltages scaling at a slower rate than the supply voltage, short-circuit power dissipation is becoming of a lesser importance in deep-submicron technologies.



$W/L|_P$ = 1.125 µm/0.25 µm
$W/L|_N$ = 0.375 µm/0.25 µm
C_L = 30 fF

Figure 5.33 Power dissipation of a static CMOS inverter as a function of the ratio between input and output rise/fall times. The power is normalized with respect to zero input rise-time dissipation. At low values of the slope ratio, input-output coupling leads to some extra dissipation.

At a supply voltage of 2.5 V and thresholds around 0.5 V, an input/output slope ratio of 2 is needed to cause a 10% degradation in dissipation.

Finally, its is worth observing that the short-circuit power dissipation can be modeled by adding a load capacitance $C_{sc} = t_{sc}I_{peak}/V_{DD}$ in parallel with C_L, as is apparent in Eq. (5.49). The value of this short-circuit capacitance is a function of V_{DD}, the transistor sizes, and the input-output slope ratio.

5.5.2 Static Consumption

The static (or steady-state) power dissipation of a circuit is expressed by Eq. (5.51), where I_{stat} is the current that flows between the supply rails in the absence of switching activity

$$P_{stat} = I_{stat}V_{DD} \tag{5.51}$$

Ideally, the static current of the CMOS inverter is equal to zero, as the PMOS and NMOS devices are never on simultaneously in steady-state operation. There is, unfortunately, a leakage current flowing through the reverse-biased diode junctions of the transistors, located between the source or drain and the substrate as shown in Figure 5.34. This contribution is, in general, very small and can be ignored. For the device sizes under consideration, the leakage current per unit drain area typically ranges between 10-100 pA/µm^2 at room temperature. For a die with 1 million gates, each with a drain area of 0.5 µm^2 and operated at a supply voltage of 2.5 V, the worst-case power consumption due to diode leakage equals 0.125 mW, which is clearly not much of an issue.

However, be aware that the junction leakage currents are caused by thermally generated carriers. Their value increases with increasing junction temperature, and this occurs in an exponential fashion. At 85°C (a common junction temperature limit for commercial hardware), the leakage currents increase by a factor of 60 over their room-temperature val-

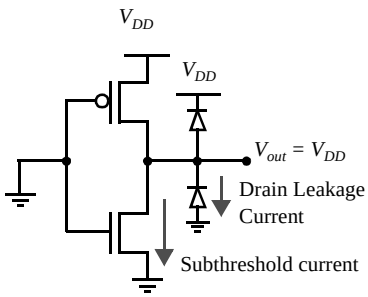


Figure 5.34 Sources of leakage currents in CMOS inverter (for V_{in} = 0 V).

ues. Keeping the overall operation temperature of a circuit low is consequently a desirable goal. As the temperature is a strong function of the dissipated heat and its removal mechanisms, this can only be accomplished by limiting the power dissipation of the circuit and/or by using chip packages that support efficient heat removal.

An emerging source of leakage current is the subthreshold current of the transistors. As discussed in Chapter 3, an MOS transistor can experience a drain-source current, even when V_{GS} is smaller than the threshold voltage (Figure 5.35). The closer the threshold voltage is to zero volts, the larger the leakage current at V_{GS} = 0 V and the larger the static power consumption. To offset this effect, the threshold voltage of the device has generally been kept high enough. Standard processes feature V_T values that are never smaller than 0.5-0.6V and that in some cases are even substantially higher (~ 0.75V).

This approach is being challenged by the reduction in supply voltages that typically goes with deep-submicron technology scaling as became apparent in Figure 3.40. We concluded earlier (Figure 5.17) that scaling the supply voltages while keeping the threshold voltage constant results in an important loss in performance, especially when V_{DD} approaches 2 V_T. One approach to address this performance issue is to scale the device thresholds down as well. This moves the curve of Figure 5.17 to the left, which means that the performance penalty for lowering the supply voltage is reduced. Unfortunately, the threshold voltages are lower-bounded by the amount of allowable subthreshold leakage current, as demonstrated in Figure 5.35. The choice of the threshold voltage hence represents a trade-off between performance and static power dissipation. The continued scaling of the supply voltage predicted for the next generations of CMOS technologies however forces the threshold voltages ever downwards, and makes subthreshold conduction a dominant source of power dissipation. Process technologies that contain devices with sharper turn-off characteristic will therefore become more attractive. An example of the latter is the SOI (Silicon-on-Insulator) technology whose MOS transistors have slope-factors that are close to the ideal 60 mV/decade.

Example 5.14 Impact of threshold reduction on performance and static power dissipation

Consider a minimum size NMOS transistor in the 0.25 μm CMOS technology. In Chapter 3, we derived that the slope factor S for this device equals 90 mV/decade. The off-current (at V_{GS} = 0) of the transistor for a V_T of approximately 0.5V equals 10^{-11}A (Figure 3.22). Reducing the threshold with 200 mV to 0.3 V multiplies the off-current of the transistors with a factor of 170! Assuming a million gate design with a supply voltage of 1.5 V, this translates into a static power dissipation of $10^6 \times 170 \times 10^{-11} \times 1.5$ = 2.6 mW. A further reduction of the thresh-

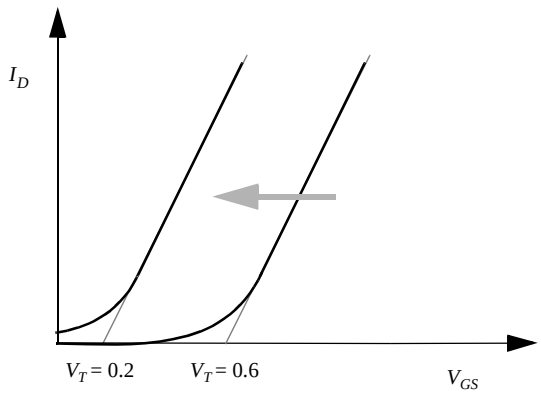


Figure 5.35 Decreasing the threshold increases the subthreshold current at V_{GS} = 0.

old to 100 mV results in an unacceptable dissipation of almost 0.5 W! At that supply voltage, the threshold reductions correspond to a performance improvement of 25% and 40%, respectively.

This lower bound on the thresholds is in some sense artificial. The idea that the leakage current in a static CMOS circuit has to be zero is a preconception. Certainly, the presence of leakage currents degrades the noise margins, because the logic levels are no longer equal to the supply rails. As long as the noise margins are within range, this is not a compelling issue. The leakage currents, of course, cause an increase in static power dissipation. This is offset by the drop in supply voltage, that is enabled by the reduced thresholds at no cost in performance, and results in a quadratic reduction in dynamic power. For a 0.25 μm CMOS process, the following circuit configurations obtain the same performance: 3 V supply–0.7 V V_T; and 0.45 V supply–0.1 V V_T. The dynamic power consumption of the latter is, however, 45 times smaller [Liu93]! Choosing the correct values of supply and threshold voltages once again requires a trade-off. The optimal operation point depends upon the activity of the circuit. In the presence of a sizable static power dissipation, it is essential that non-active modules are *powered down*, lest static power dissipation would become dominant. Power-down (also called *standby*) can be accomplished by disconnecting the unit from the supply rails, or by lowering the supply voltage.

5.5.3 Putting It All Together

The total power consumption of the CMOS inverter is now expressed as the sum of its three components:

$$P_{tot} = P_{dyn} + P_{dp} + P_{stat} = (C_L V_{DD}^2 + V_{DD} I_{peak} t_s) f_{0 \to 1} + V_{DD} I_{leak} \tag{5.52}$$

In typical CMOS circuits, the capacitive dissipation is by far the dominant factor. The direct-path consumption can be kept within bounds by careful design, and should hence not be an issue. Leakage is ignorable at present, but this might change in the not too distant future.

The Power-Delay Product, or Energy per Operation

In Chapter 1, we introduced the *power-delay product* (PDP) as a quality measure for a logic gate.

$$PDP = P_{av} t_p \tag{5.53}$$

The PDP presents a measure of energy, as is apparent from the units (Wsec = Joule). Assuming that the gate is switched at its maximum possible rate of $f_{max} = 1/(2t_p)$, and ignoring the contributions of the static and direct-path currents to the power consumption, we find

$$PDP = C_L V_{DD}^2 f_{max} t_p = \frac{C_L V_{DD}^2}{2} \tag{5.54}$$

The PDP stands for the **average energy consumed per switching event** (this is, for a 0→1, or a 1→0 transition). Remember that earlier we had defined E_{av} as the average energy per switching cycle (or per energy-consuming event). As each inverter cycle contains a 0→1, and a 1→0 transition, E_{av} hence is twice the PDP.

Energy-Delay Product

The validity of the PDP as a quality metric for a process technology or gate topology is questionable. It measures the energy needed to switch the gate, which is an important property for sure. Yet for a given structure, this number can be made arbitrarily low by reducing the supply voltage. From this perspective, the optimum voltage to run the circuit at would be the lowest possible value that still ensures functionality. This comes at the major expense in performance, at discussed earlier. A more relevant metric should combine a measure of performance and energy. The energy-delay product (EDP) does exactly that.

$$EDP = PDP \times t_p = P_{av} t_p^2 = \frac{C_L V_{DD}^2}{2} t_p \tag{5.55}$$

It is worth analyzing the voltage dependence of the EDP. Higher supply voltages reduce delay, but harm the energy, and the opposite is true for low voltages. An optimum operation point should hence exist. Assuming that NMOS and PMOS transistors have comparable threshold and saturation voltages, we can simplify the propagation delay expression Eq. (5.21).

$$t_p \approx \frac{\alpha C_L V_{DD}}{V_{DD} - V_{Te}} \tag{5.56}$$

where $V_{Te} = V_T + V_{DSAT}/2$, and α technology parameter. Combining Eq. (5.55) and Eq. (5.56), [4]

[4] This equation is only accurate as long as the devices remain in velocity saturation, which is probably not the case for the lower supply voltages. This introduces some inaccuracy in the analysis, but will not distort the overall result.

$$EDP = \frac{\alpha C_L^2 V_{DD}^3}{2(V_{DD} - V_{TE})} \tag{5.57}$$

The optimum supply voltage can be obtained by taking the derivative of Eq. (5.57) with respect to V_{DD}, and equating the result to 0.

$$V_{DDopt} = \frac{3}{2} V_{TE} \tag{5.58}$$

The remarkable outcome from this analysis is the low value of the supply voltage that simultaneously optimizes performance and energy. For sub-micron technologies with thresholds in the range of 0.5 V, the optimum supply is situated around 1 V.

Example 5.15 Optimum supply voltage for 0.25 μm CMOS inverter

From the technology parameters for our generic CMOS process presented in Chapter 3, the value of V_{TE} can be derived.

$$V_{Tn} = 0.43 \text{ V}, V_{Dsatn} = 0.63 \text{ V}, V_{TEn} = 0.74 \text{ V}.$$
$$V_{Tp} = -0.4 \text{ V}, V_{Dsatp} = -1 \text{ V}, V_{TEp} = -0.9 \text{ V}.$$
$$V_{TE} \approx (V_{TEn} + |V_{TEp}|)/2 = 0.8 \text{ V}$$

Hence, $V_{DDopt} = (3/2) \times 0.8$ V = 1.2 V. The simulated graphs of Figure 5.36, plotting normalized delay, energy, and energy-delay product, confirm this result. The optimum supply voltage is predicted to equal 1.1 V. The charts clearly illustrate the trade-off between delay and energy.

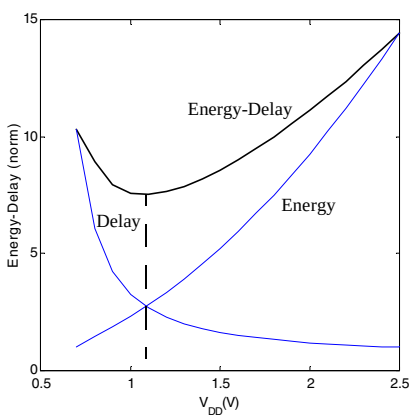


Figure 5.36 Normalized delay, energy, and energy-delay plots for CMOS inverter in 0.25 μm CMOS technology.

WARNING: While the above example demonstrates that there exists a supply voltage that minimizes the energy-delay product of a gate, this voltage does not necessarily represent the optimum voltage for a given design problem. For instance, some designs require a minimum performance, which requires a higher voltage at the expense of energy. Similarly, a lower-energy design is possible by operating by circuit at a lower voltage and by

obtaining the overall system performance through the use of architectural techniques such as pipelining or concurrency.

5.5.4 Analyzing Power Consumption Using SPICE

A definition of the average power consumption of a circuit was provided in Chapter 1, and is repeated here for the sake of convenience.

$$P_{av} = \frac{1}{T}\int_0^T p(t)dt = \frac{V_{DD}}{T}\int_0^T i_{DD}(t)dt \tag{5.59}$$

with T the period of interest, and V_{DD} and i_{DD} the supply voltage and current, respectively. Some implementations of SPICE provide built-in functions to measure the average value of a circuit signal. For instance, the HSPICE *.MEASURE TRAN I(VDD) AVG* command computes the area under a computed transient response (*I*(*VDD*)) and divides it by the period of interest. This is identical to the definition given in Eq. (5.59). Other implementations of SPICE are, unfortunately, not as extensive. This is not as bad as it seems, as long as one realizes that SPICE is actually a differential equation solver. A small circuit can easily be conceived that acts as an integrator and whose output signal is nothing but the average power.

Consider, for instance, the circuit of Figure 5.37. The current delivered by the power supply is measured by the current-controlled current source and integrated on the capacitor C. The resistance R is only provided for DC-convergence reasons and should be chosen as high as possible to minimize leakage. A clever choice of the element parameter ensures that the output voltage P_{av} equals the average power consumption. The operation of the circuit is summarized in Eq. (5.60) under the assumption that the initial voltage on the capacitor C is zero.

$$\begin{gathered} C\frac{dP_{av}}{dt} = ki_{DD} \\ or \\ P_{av} = \frac{k}{C}\int_0^T i_{DD}dt \end{gathered} \tag{5.60}$$

Equating Eq. (5.59) and Eq. (5.60) yields the necessary conditions for the equivalent circuit parameters: $k/C = V_{DD}/T$. Under these circumstances, the equivalent circuit shown presents a convenient means of tracking the average power in a digital circuit.

Example 5.16 Average Power of Inverter

The average power consumption of the inverter of Example 5.4 is analyzed using the above technique for a toggle period of 250 psec (T = 250 psec, k = 1, V_{DD} = 2.5 V, hence C = 100 pF). The resulting power consumption is plotted in Figure 5.38, showing an average power consumption of approximately 157.3 μW. The .MEAS AVG command yields a value of

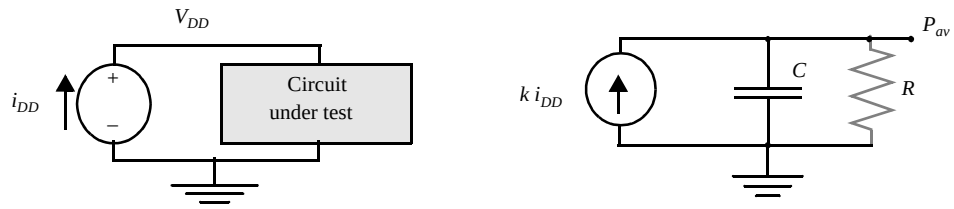


Figure 5.37 Equivalent circuit to measure average power in SPICE.

160.32 μW, which demonstrates the approximate equivalence of both methods. These numbers are equivalent to an energy of 39 fJ (which is close to the 37.5 fJ derived in Example 5.11). Observe the slightly negative dip during the high-to-low transition. This is due to the injection of current into the supply, when the output briefly overshoots V_{DD} as a result of the capacitive coupling between input and output (as is apparent from in the transient response of Figure 5.16).

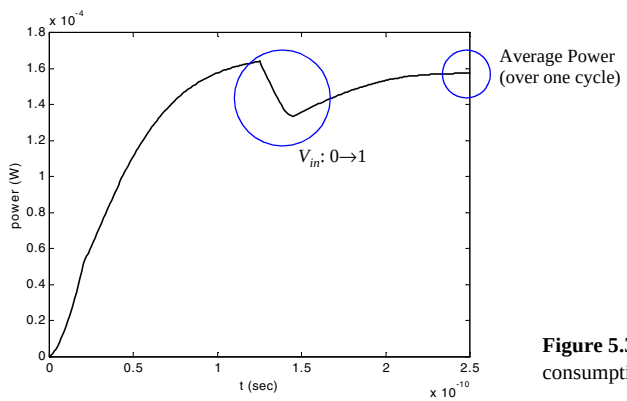


Figure 5.38 Deriving the power consumption using SPICE.

5.6 Perspective: Technology Scaling and its Impact on the Inverter Metrics

In section 3.5, we have explored the impact of the scaling of technology on the some of the important design parameters such as area, delay, and power. For the sake of clarity, we repeat here some of the most important entries in the resulting scaling table (Table 3.8).

Table 5.4 Scaling scenarios for short-channel devices (*S* and *U* represent the technology and voltage scaling parameters, respectively).

Parameter	Relation	Full Scaling	General Scaling	Fixed-Voltage Scaling
Area/Device	WL	$1/S^2$	$1/S^2$	$1/S^2$
Intrinsic Delay	$R_{on}C_{gate}$	$1/S$	$1/S$	$1/S$

Table 5.4 Scaling scenarios for short-channel devices (*S* and *U* represent the technology and voltage scaling parameters, respectively).

Parameter	Relation	Full Scaling	General Scaling	Fixed-Voltage Scaling
Intrinsic Energy	$C_{gate}V^2$	$1/S^3$	$1/SU^2$	$1/S$
Intrinsic Power	*Energy/Delay*	$1/S^2$	$1/U^2$	1
Power Density	*P/Area*	1	S^2/U^2	S^2

To validity of these theoretical projections can be verified by looking back and observing the trends during the past decades. From Figure 5.39, we can derive that the gate delay indeed decreases exponentially at a rate of 13%/year, or halving every five years. This rate is on course with the prediction of Table 5.4, since *S* averages approximately 1.15 as we had already observed in Figure 3.39. The delay of a 2-input NAND gate with a fanout of four has gone from tens of nanoseconds in the 1960s to a tenth of a nanosecond in the year 2000, and is projected to be a few tens of picoseconds by 2010.

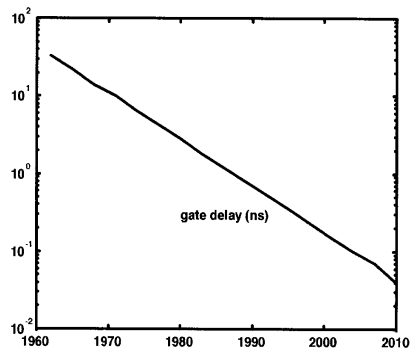


Figure 5.39 Scaling of the gate delay (from [Dally98]).

Reducing power dissipation has only been a second-order priority until recently. Hence, statistics on dissipation-per-gate or design are only marginally available. An interesting chart is shown in Figure 5.40, which plots the power density measured over a large number of designs produced between 1980 and 1995. Although the variation is large—even for a fixed technology—it shows the power density to increase approximately with S^2. This is in correspondence with the fixed-voltage scaling scenario presented in Table 5.4. For more recent years, we expect a scenario more in line with the full-scaling model—which predicts a constant power density—due to the accelerated supply-voltage scaling and the increased attention to power-reducing design techniques. Even under these circumstances, power dissipation-per-chip will continue to increase due to the ever-larger die sizes.

The presented scaling model has one fatal flaw however: the performance and power predictions produce purely "intrinsic" numbers that take only device parameters into account. In Chapter 4, it was concluded that the interconnect wires exhibit a different scaling behavior, and that wire parasitics may come to dominate the overall performance. Similarly, charging and discharging the wire capacitances may dominate the energy budget. To get a crisper perspective, one has to construct a combined model that considers device and wire scaling models simultaneously. The impact of the wire capacitance and its scaling behavior is summarized in Table 5.5. We adopt the fixed-resistance model introduced in Chapter 4. We furthermore assume that the resistance of the driver dominates the wire resistance, which is definitely the case for short to medium-long wires.

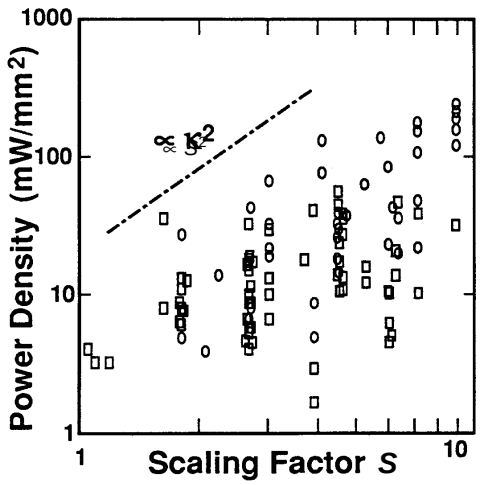


Figure 5.40 Evolution of power-density in micro- and DSP processors, as a function of the scaling factor S ([Sakurai97]). *S* is normalized to 1 for a 4 µm process.

Table 5.5 Scaling scenarios for wire capacitance. *S* and *U* represent the technology and voltage scaling parameters, respectively, while S_L stands for the wire-length scaling factor. ε_c represents the impact of fringing and inter-wire capacitances.

Parameter	Relation	General Scaling
Wire Capacitance	WL/t	ε_c/S_L
Wire Delay	$R_{on}C_{int}$	ε_c/S_L
Wire Energy	$C_{int}V^2$	$\varepsilon_c/S_L U^2$
Wire Delay / Intrinsic Delay		$\varepsilon_c S/S_L$
Wire Energy / Intrinsic Energy		$\varepsilon_c S/S_L$

The model predicts that the interconnect-caused delay (and energy) gain in importance with the scaling of technology. This impact is limited to an increase with ε_c for short wires ($S = S_L$), but it becomes increasingly more outspoken for medium-range and long wires ($S_L < S$). These conclusions have been confirmed by a number of studies, an example of which is shown in Figure 5.41. How the ratio of wire over intrinsic contributions will actually evolve is debatable, as it depends upon a wide range of independent parameters such as system architecture, design methodology, transistor sizing, and interconnect materials. The doom-day scenario that interconnect may cause CMOS performance to saturate in the very near future hence may be exaggerated. Yet, it is clear to that increased attention to interconnect is an absolute necessity, and may change the way the next-generation circuits are designed and optimized (e.g. [Sylvester99]).

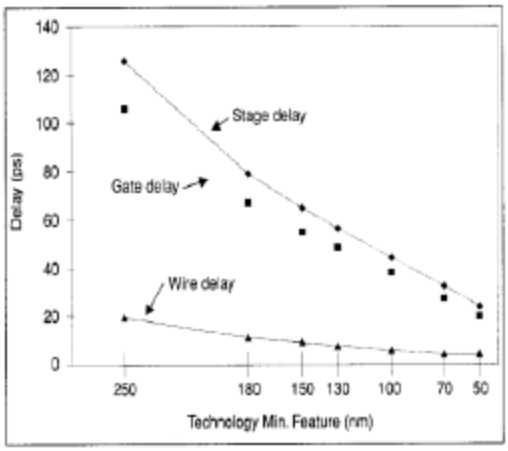


Figure 5.41 Evolution of wire delay / gate delay ratio with respect to technology (from [Fisher98]).

5.7 Summary

This chapter presented a rigorous and in-depth analysis of the static CMOS inverter. The key characteristics of the gate are summarized:

- The static CMOS inverter combines a pull-up PMOS section with a pull-down NMOS device. The PMOS is normally made wider than the NMOS due to its inferior current-driving capabilities.
- The gate has an almost ideal voltage-transfer characteristic. The logic swing is equal to the supply voltage and is not a function of the transistor sizes. The noise margins of a symmetrical inverter (where PMOS and NMOS transistor have equal current-driving strength) approach $V_{DD}/2$. The steady-state response is not affected by fan-out.
- Its propagation delay is dominated by the time it takes to charge or discharge the load capacitor C_L. To a first order, it can be approximated as

$$t_p = 0.69 C_L \left(\frac{R_{eqn} + R_{eqp}}{2} \right)$$

  Keeping the load capacitance small is the most effective means of implementing high-performance circuits. Transistor sizing may help to improve performance as long as the delay is dominated by the extrinsic (or load) capacitance of fanout and wiring.
- The power dissipation is dominated by the dynamic power consumed in charging and discharging the load capacitor. It is given by $P_{0\to1}\, C_L V_{DD}^2 f$. The dissipation is proportional to the activity in the network. The dissipation due to the direct-path currents occurring during switching can be limited by careful tailoring of the signal

slopes. The static dissipation can usually be ignored but might become a major factor in the future as a result of subthreshold currents.

- Scaling the technology is an effective means of reducing the area, propagation delay and power consumption of a gate. The impact is even more striking if the supply voltage is scaled simultaneously.
- The interconnect component is gradually taking a larger fraction of the delay and performance budget.

5.8 To Probe Further

The operation of the CMOS inverter has been the topic of numerous publications and textbooks. Virtually every book on digital design devotes a substantial number of pages to the analysis of the basic inverter gate. An extensive list of references was presented in Chapter 1. Some references of particular interest that were explicitly quoted in this chapter are given below.

REFERENCES

[Dally98] W. Dally and J. Poulton, *Digital Systems Engineering*, Cambridge University Press, 1998.

[Fisher98] P. D. Fisher and R. Nesbitt, ``The Test of Time: Clock-Cycle Estimation and Test Challenges for Future Microprocessors," *IEEE Circuits and Devices Magazine*, 14(2), pp. 37-44, 1998.

[Hedenstierna87] N. Hedenstierna and K. Jeppson, “CMOS Circuit Speed and Buffer Optimization,” *IEEE Transactions on CAD*, Vol CAD-6, No 2, pp. 270-281, March 1987.

[Liu93] D. Liu and C Svensson, “Trading speed for low power by choice of supply and threshold voltages”, IEEE Journal of Solid State Circuits, Vol.28, no.1, pp. 10-17, Jan. 1993, p.10-17.

[Mead80] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[Sakurai97] T.Sakurai, H.Kawaguchi, T.Kuroda, “Low-Power CMOS Design through VTH Control and Low-Swing Circuits,” *Digest International Symp.on Low-Power Electronics and Design*, pp.1-6, Sept. 1997. Also in T.Sakurai, T.Kuroda, “Low Voltage Technology and Circuits,” *Mead Microelectronics Conference*, Lausanne, Switzerland, June 1997.

[Sedra87] Sedra and Smith, *MicroElectronic Circuits*, Holt, Rinehart and Winston, 1987.

[Swanson72] R. Swanson and J. Meindl, “Ion-Implanted Complementary CMOS transistors in Low-Voltage Circuits,” *IEEE Journal of Solid-State Circuits*, Vol. SC-7, No. 2, pp.146-152, April 1972.

[Veendrick84] H. Veendrick, “Short-Circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer Circuits,” *IEEE Journal of Solid-State Circuits*, Vol. SC-19, no. 4, pp. 468–473, 1984.

5.9 Exercises and Design Problems

For all problems, use the device parameters provided in Chapter 3 (as well as the inside back cover), unless otherwise mentioned.

DESIGN PROBLEM

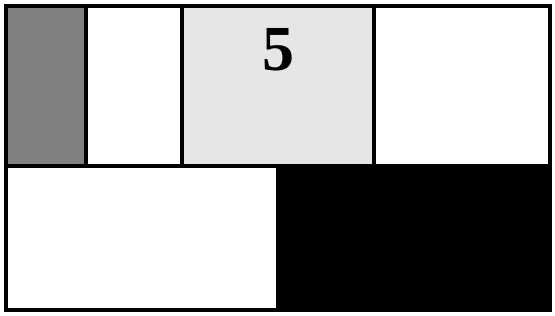
Using the 1.2 μm CMOS introduced in Chapter 2, design a static CMOS inverter that meets the following requirements:

1. Matched pull-up and pull-down times (i.e., $t_{pHL} = t_{pLH}$).
2. $t_p = 5$ nsec (± 0.1 nsec).

The load capacitance connected to the output is equal to 4 pF. Notice that this capacitance is substantially larger than the internal capacitances of the gate.

Determine the W and L of the transistors. To reduce the parasitics, use minimal lengths ($L = 1.2$ μm) for all transistors. Verify and optimize the design using SPICE after proposing a first design using manual computations. Compute also the energy consumed per transition. If you have a layout editor (such as MAGIC) available, perform the physical design, extract the real circuit parameters, and compare the simulated results with the ones obtained earlier.

CHAPTER



THE CMOS INVERTER

Quantification of integrity, performance, and energy metrics of an inverter
Optimization of an inverter design

5.1 Exercises and Design Problems

5.2 The Static CMOS Inverter — An Intuitive Perspective

5.3 Evaluating the Robustness of the CMOS Inverter: The Static Behavior

- 5.3.1 Switching Threshold
- 5.3.2 Noise Margins
- 5.3.3 Robustness Revisited

5.4 Performance of CMOS Inverter: The Dynamic Behavior

- 5.4.1 Computing the Capacitances
- 5.4.2 Propagation Delay: First-Order Analysis
- 5.4.3 Propagation Delay from a Design Perspective

5.5 Power, Energy, and Energy-Delay

- 5.5.1 Dynamic Power Consumption
- 5.5.2 Static Consumption
- 5.5.3 Putting It All Together
- 5.5.4 Analyzing Power Consumption Using SPICE

5.6 Perspective: Technology Scaling and its Impact on the Inverter Metrics

5.1 Exercises and Design Problems

1. [M, SPICE, 3.3.2] The layout of a static CMOS inverter is given in Figure 5.1. (λ = 0.125 µm).

a. Determine the sizes of the NMOS and PMOS transistors.

b. Plot the VTC (using HSPICE) and derive its parameters (V_{OH}, V_{OL}, V_M, V_{IH}, and V_{IL}).

c. Is the VTC affected when the output of the gates is connected to the inputs of 4 similar gates?.

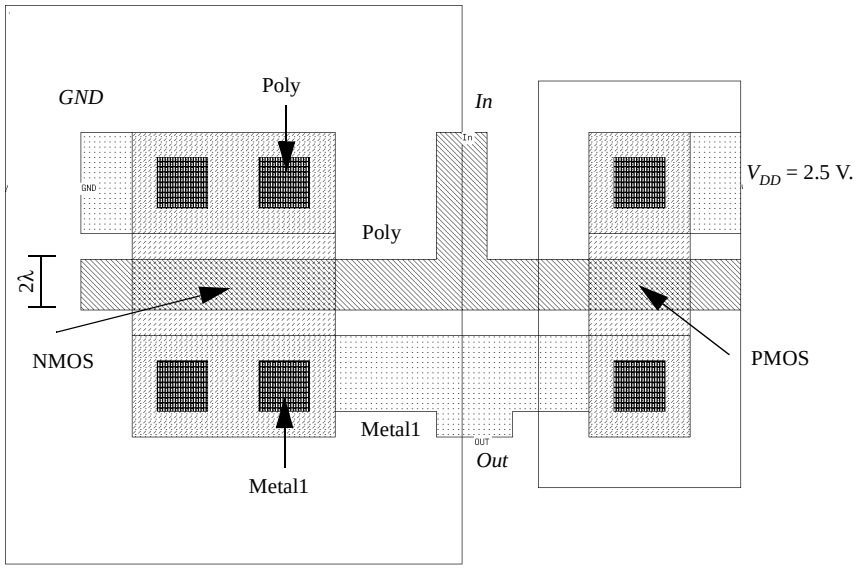


Figure 5.1 CMOS inverter layout.

d. Resize the inverter to achieve a switching threshold of approximately 0.75 V. Do not layout the new inverter, use HSPICE for your simulations. How are the noise margins affected by this modification?

2. Figure 5.2 shows a piecewise linear approximation for the VTC. The transition region is approximated by a straight line with a slope equal to the inverter gain at V_M. The intersection of this line with the V_{OH} and the V_{OL} lines defines V_{IH} and V_{IL}.

a. The noise margins of a CMOS inverter are highly dependent on the sizing ratio, $r = k_p/k_n$, of the NMOS and PMOS transistors. Use HSPICE with $V_{Tn} = |V_{Tp}|$ to determine the value of r that results in equal noise margins? Give a qualitative explanation.

b. Section 5.3.2 of the text uses this piecewise linear approximation to derive simplified expressions for NM_H and NM_L in terms of the inverter gain. The derivation of the gain is based on the assumption that both the NMOS and the PMOS devices are velocity saturated at V_M. For what range of r is this assumption valid? What is the resulting range of V_M?

c. Derive expressions for the inverter gain at V_M for the cases when the sizing ratio is just above and just below the limits of the range where both devices are velocity saturated. What are the operating regions of the NMOS and the PMOS for each case? Consider the effect of channel-length modulation by using the following expression for the small-signal resistance in the saturation region: $r_{o,sat} = 1/(\lambda I_D)$.

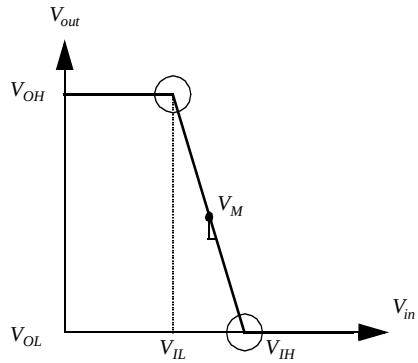


Figure 5.2 A different approach to derive V_{IL} and V_{IH}.

3. [M, SPICE, 3.3.2] Figure 5.3 shows an NMOS inverter with resistive load.

a. Qualitatively discuss why this circuit behaves as an inverter.

b. Find V_{OH} and V_{OL} calculate V_{IH} and V_{IL}.

c. Find NM_L and NM_H, and plot the VTC using HSPICE.

d. Compute the average power dissipation for: (i) $V_{in} = 0$ V and (ii) $V_{in} = 2.5$ V

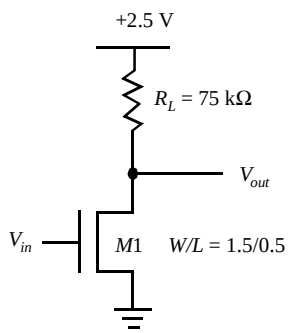


Figure 5.3 Resistive-load inverter

e. Use HSPICE to sketch the VTCs for R_L = 37k, 75k, and 150k on a single graph.

f. Comment on the relationship between the critical VTC voltages (i.e., V_{OL}, V_{OH}, V_{IL}, V_{IH}) and the load resistance, R_L.

g. Do high or low impedance loads seem to produce more ideal inverter characteristics?

4. [E, None, 3.3.3] For the inverter of Figure 5.3 and an output load of 3 pF:

a. Calculate t_{plh}, t_{phl}, and t_p.

b. Are the rising and falling delays equal? Why or why not?

c. Compute the static and dynamic power dissipation assuming the gate is clocked as fast as possible.

5. The next figure shows two implementations of MOS inverters. The first inverter uses only NMOS transistors.

a. Calculate V_{OH}, V_{OL}, V_M for each case.

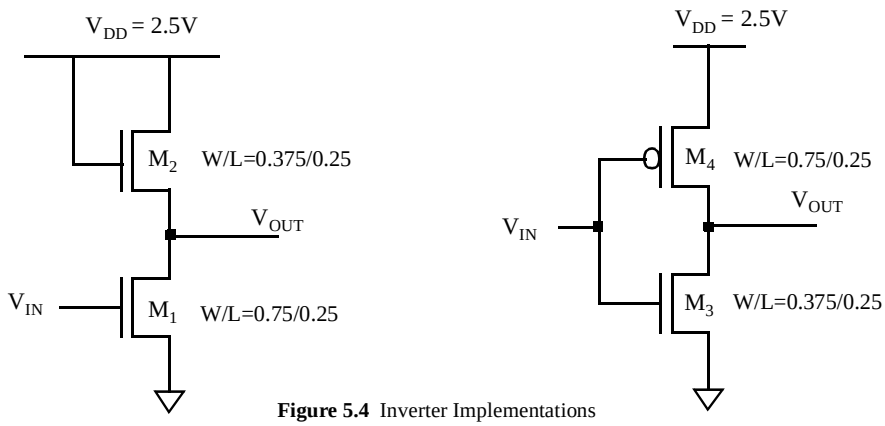


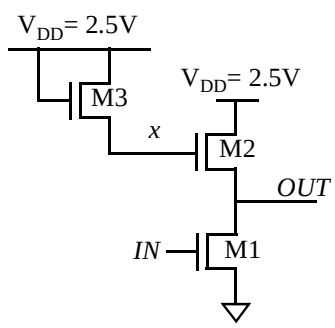
Figure 5.4 Inverter Implementations

b. Use HSPICE to obtain the two VTCs. You must assume certain values for the source/drain areas and perimeters since there is no layout. For our scalable CMOS process, λ = 0.125 µm, and the source/drain extensions are 5λ for the PMOS; for the NMOS the source/drain contact regions are 5λx5λ.

c. Find V_{IH}, V_{IL}, NM_L and NM_H for each inverter and comment on the results. How can you increase the noise margins and reduce the undefined region?

d. Comment on the differences in the VTCs, robustness and regeneration of each inverter.

6. Consider the following NMOS inverter. Assume that the bulk terminals of all NMOS device are connected to GND. Assume that the input IN has a 0V to 2.5V swing.



a. Set up the equation(s) to compute the voltage on node *x*. Assume γ=0.5.

b. What are the modes of operation of device M2? Assume γ=0.

c. What is the value on the output node *OUT* for the case when *IN* =0V?Assume γ=0.

d. Assuming γ=0, derive an expression for the switching threshold (V_M) of the inverter. Recall that the switching threshold is the point where V_{IN}= V_{OUT}. Assume that the device sizes for M1, M2 and M3 are $(W/L)_1$, $(W/L)_2$, and $(W/L)_3$ respectively. What are the limits on the switching threshold?

For this, consider two cases:

i) $(W/L)_1 >> (W/L)_2$

ii) $(W/L)_2 >> (W/L)_1$

7. Consider the circuit in Figure 5.5. Device M1 is a standard NMOS device. Device M2 has all the same properties as M1, except that its device threshold voltage is *negative* and has a value of -0.4V. Assume that all the current equations and inequality equations (to determine the mode of operation) for the depletion device M2 are the same as a regular NMOS. Assume that the input *IN* has a 0V to 2.5V swing.

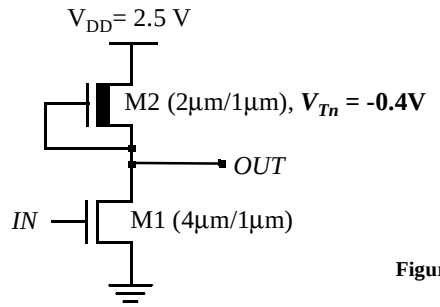


Figure 5.5 A depletion load NMOS inverter

a. Device M2 has its gate terminal connected to its source terminal. If V_{IN} = 0V, what is the output voltage? In steady state, what is the mode of operation of device M2 for this input?

b. Compute the output voltage for V_{IN} = 2.5V. You may assume that $\boldsymbol{V_{OUT}}$ is small to simplify your calculation. In steady state, what is the mode of operation of device M2 for this input?

c. Assuming $Pr_{(IN=0)}$= 0.3, what is the static power dissipation of this circuit?

8. [M, None, 3.3.3] An NMOS transistor is used to charge a large capacitor, as shown in Figure 5.6.

a. Determine the t_{pLH} of this circuit, assuming an ideal step from 0 to 2.5V at the input node.

b. Assume that a resistor R_S of 5 kΩ is used to discharge the capacitance to ground. Determine t_{pHL}.

c. Determine how much energy is taken from the supply during the charging of the capacitor. How much of this is dissipated in M1. How much is dissipated in the pull-down resistance during discharge? How does this change when R_S is reduced to 1 kΩ.

d. The NMOS transistor is replaced by a PMOS device, sized so that k_p is equal to the k_n of the original NMOS. Will the resulting structure be faster? Explain why or why not.

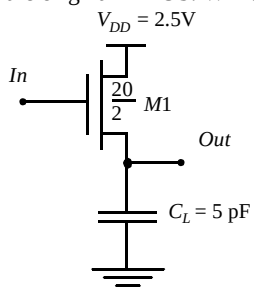


Figure 5.6 Circuit diagram with annotated *W/L* ratios

9. The circuit in Figure 5.7 is known as the *source follower* configuration. It achieves a DC level shift between the input and the output. The value of this shift is determined by the current I_0. Assume x_d=0, γ=0.4, 2|ϕ_f|=0.6V, V_{T0}=0.43V, k_n'=115μA/V^2 and λ=0.

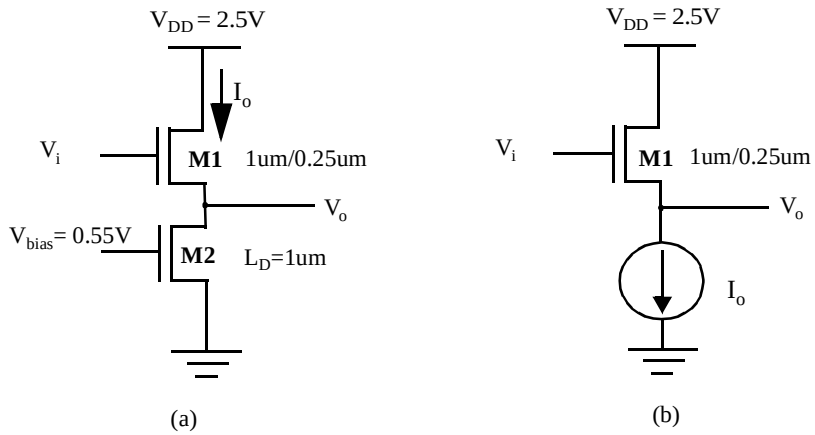


Figure 5.7 NMOS source follower configuration

a. Suppose we want the nominal level shift between V_i and V_o to be 0.6V in the circuit in Figure 5.7 (a). Neglecting the backgate effect, calculate the width of M2 to provide this level shift (Hint: first relate V_i to V_o in terms of I_o).

b. Now assume that an ideal current source replaces M2 (Figure 5.7 (b)). The NMOS transistor M1 experiences a shift in V_T due to the backgate effect. Find V_T as a function of V_o for V_o ranging from 0 to 2.5V with 0.5V intervals. Plot V_T vs. V_o

c. Plot V_o vs. V_i as V_o varies from 0 to 2.5V with 0.5 V intervals. Plot two curves: one neglecting the body effect and one accounting for it. How does the body effect influence the operation of the level converter?

d. At V_o(with body effect) = 2.5V, find V_o(ideal) and thus determine the maximum error introduced by the body effect.

10. For this problem assume:

V_{DD} = 2.5V, W_P/L = 1.25/0.25, W_N/L = 0.375/0.25, $L=L_{eff}$ =0.25µm (i.e. x_d= 0µm), $C_L=C_{inv-gate}$, k_n' = 115µA/V^2, k_p'= -30µA/V^2, V_{tn0} = | V_{tp0} | = 0.4V, λ = 0V^{-1}, γ = 0.4, 2|ϕ_f|=0.6V, and t_{ox} = 58A. Use the HSPICE model parameters for parasitic capacitance given below (i.e. C_{gd0}, C_j, C_{jsw}), and assume that V_{SB}=0V for all problems except part **(e)**.

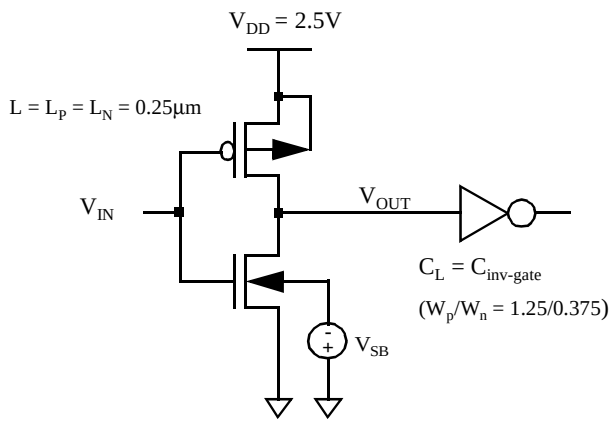


Figure 5.8 CMOS inverter with capacitive

Parasitic Capacitance Parameters (F/m)##
NMOS: CGDO=3.11x10^{-10}, CGSO=3.11x10^{-10}, CJ=2.02x10^{-3}, CJSW=2.75x10^{-10}
PMOS: CGDO=2.68x10^{-10}, CGSO=2.68x10^{-10}, CJ=1.93x10^{-3}, CJSW=2.23x10^{-10}

a. What is the V_m for this inverter?

b. What is the effective load capacitance C_{Leff} of this inverter? (include parasitic capacitance, refer to the text for K_{eq} and m.) **Hint:** You must assume certain values for the source/drain areas and perimeters since there is no layout. For our scalable CMOS process, λ = 0.125 μm, and the source/drain extensions are 5λ for the PMOS; for the NMOS the source/drain contact regions are 5λx5λ.

c. Calculate t_{PHL}, t_{PLH} assuming the result of (b) is 'C_{Leff} = 6.5fF'. (Assume an ideal step input, i.e. t_{rise}=t_{fall}=0. Do this part by computing the average current used to charge/discharge C_{Leff}.)

d. Find (W_p/W_n) such that $t_{PHL} = t_{PLH}$.

e. Suppose we increase the width of the transistors to reduce the t_{PHL}, t_{PLH}. Do we get a proportional decrease in the delay times? Justify your answer.

f. Suppose V_{SB} = 1V, what is the value of V_{tn}, V_{tp}, V_m? How does this qualitatively affect C_{Leff}?

11. Using Hspice answer the following questions.

a. Simulate the circuit in Problem 10 and measure t_P and the average power for input V_{in}: pulse(0 V_{DD} 5n 0.1n 0.1n 9n 20n), as V_{DD} varies from 1V - 2.5V with a 0.25V interval. [t_P = (t_{PHL} + t_{PLH}) / 2]. Using this data, plot 't_P vs. V_{DD}', and 'Power vs. V_{DD}'.

Specify AS, AD, PS, PD in your spice deck, and manually add C_L = 6.5fF. Set V_{SB} = 0V for this problem.

b. For Vdd equal to 2.5V determine the maximum fan-out of identical inverters this gate can drive before its delay becomes larger than 2 ns.

c. Simulate the same circuit for a set of 'pulse' inputs with rise and fall times of $t_{in\_rise,fall}$ =1ns, 2ns, 5ns, 10ns, 20ns. For each input, measure (1) the rise and fall times t_{out_rise} and

t_{out_fall} of the inverter output, (2) the total energy lost E_{total}, and (3) the energy lost due to short circuit current E_{short}.

Using this data, prepare a plot of (1) $(t_{out\_rise}+t_{out\_fall})/2$ vs. $t_{in\_rise,fall}$, (2) E_{total} vs. $t_{in\_rise,fall}$, (3) E_{short} vs. $t_{in\_rise,fall}$ and (4) E_{short}/E_{total} vs. $t_{in\_rise,fall.}$

d. Provide simple explanations for:

(i) Why the slope for (1) is less than 1?
(ii) Why E_{short} increases with $t_{in\_rise,fall}$?
(iii) Why E_{total} increases with $t_{in\_rise,fall}$?

12. Consider the low swing driver of Figure 5.9:

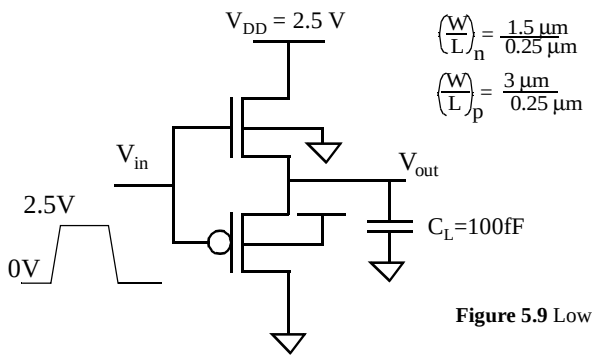


Figure 5.9 Low Swing Driver

a. What is the voltage swing on the output node (V_{out})? Assume γ=0.

b. Estimate (i) the energy drawn from the supply and (ii) energy dissipated for a 0V to 2.5V transition at the input. Assume that the rise and fall times at the input are 0. Repeat the analysis for a 2.5V to 0V transition at the input.

c. Compute t_{pLH} (i.e. the time to transition from V_{OL} to $(V_{OH} + V_{OL})/2$). Assume the input rise time to be 0. V_{OL} is the output voltage with the input at 0V and V_{OH} is the output voltage with the input at 2.5V.

d. Compute V_{OH} taking into account body effect. Assume $\gamma = 0.5V^{1/2}$ for both NMOS and PMOS.

13. Consider the following low swing driver consisting of NMOS devices M1 and M2. Assume an NWELL implementation. Assume that the inputs IN and $\overline{IN}$ have a 0V to 2.5V swing and that V_{IN} = 0V when $V_{\overline{IN}}$ = 2.5V and vice-versa. Also assume that there is no skew between IN and $\overline{IN}$ (i.e., the inverter delay to derive $\overline{IN}$ from IN is zero).

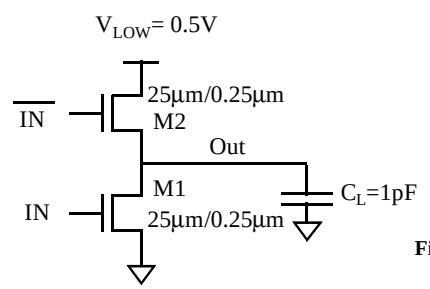


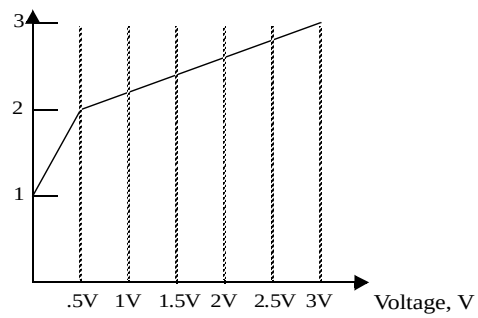
Figure 5.10 Low Swing Driver

a. What voltage is the bulk terminal of M2 connected to?

b. What is the voltage swing on the output node as the inputs swing from 0V to 2.5V. Show the low value and the high value.

c. Assume that the inputs IN and $\overline{IN}$ have zero rise and fall times. Assume a zero skew between IN and $\overline{IN}$. Determine the low to high propagation delay for charging the output node measured from the the 50% point of the input to the 50% point of the output. Assume that the total load capacitance is 1pF, including the transistor parasitics.

d. Assume that, instead of the 1pF load, the low swing driver drives a non-linear capacitor, whose capacitance vs. voltage is plotted below. Compute the energy drawn from the low supply for charging up the load capacitor. Ignore the parasitic capacitance of the driver circuit itself.



14. The inverter below operates with V_{DD}=0.4V and is composed of |Vt| = 0.5V devices. The devices have identical I_0 and n.

a. Calculate the switching threshold (V_M) of this inverter.

b. Calculate V_{IL} and V_{IH} of the inverter.

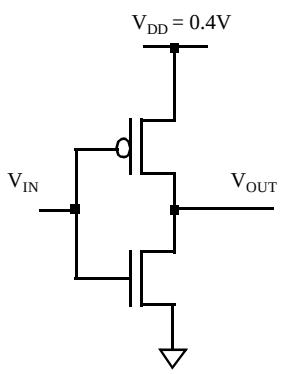


Figure 5.11 Inverter in Weak Inversion Regime

15. Sizing a chain of inverters.

a. In order to drive a large capacitance (C_L = 20 pF) from a minimum size gate (with input capacitance C_i = 10fF), you decide to introduce a two-staged buffer as shown in Figure 5.12. Assume that the propagation delay of a minimum size inverter is 70 ps. Also assume

that the input capacitance of a gate is proportional to its size. Determine the sizing of the two additional buffer stages that will minimize the propagation delay.

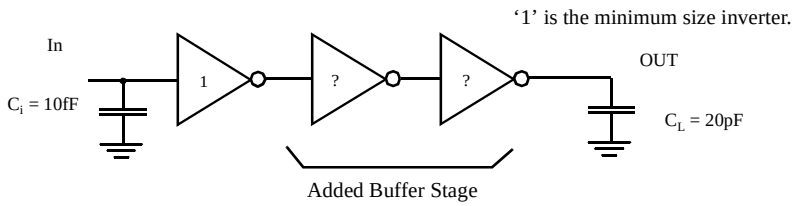


Figure 5.12 Buffer insertion for driving large loads.

b. If you could add any number of stages to achieve the minimum delay, how many stages would you insert? What is the propagation delay in this case?

c. Describe the advantages and disadvantages of the methods shown in (a) and (b).

d. Determine a closed form expression for the power consumption in the circuit. Consider only gate capacitances in your analysis. What is the power consumption for a supply voltage of 2.5V and an activity factor of 1?

16. [M, None, 3.3.5] Consider scaling a CMOS technology by S > 1. In order to maintain compatibility with existing system components, you decide to use constant voltage scaling.

a. In traditional constant voltage scaling, transistor widths scale inversely with S, $W \propto 1/S$. To avoid the power increases associated with constant voltage scaling, however, you decide to change the scaling factor for *W*. What should this new scaling factor be to maintain approximately constant power. Assume long-channel devices (i.e., neglect velocity saturation).

b. How does delay scale under this new methodology?

c. Assuming short-channel devices (i.e., velocity saturation), how would transistor widths have to scale to maintain the constant power requirement?

DESIGN PROBLEM

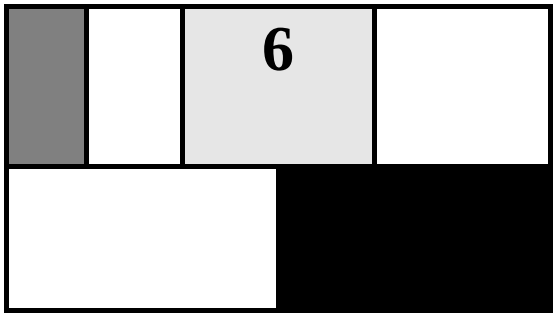
Using the 0.25 μm CMOS introduced in Chapter 2, design a static CMOS inverter that meets the following requirements:

1. Matched pull-up and pull-down times (i.e., $t_{pHL} = t_{pLH}$).
2. $t_p = 5$ nsec ($\pm$ 0.1 nsec).

The load capacitance connected to the output is equal to 4 pF. Notice that this capacitance is substantially larger than the internal capacitances of the gate.

Determine the W and L of the transistors. To reduce the parasitics, use minimal lengths ($L = 0.25$ μm) for all transistors. Verify and optimize the design using SPICE after proposing a first design using manual computations. Compute also the energy consumed per transition. If you have a layout editor (such as MAGIC) available, perform the physical design, extract the real circuit parameters, and compare the simulated results with the ones obtained earlier.

CHAPTER



DESIGNING COMBINATIONAL LOGIC GATES IN CMOS

In-depth discussion of logic families in CMOS—
static and dynamic, pass-transistor, non-ratioed and ratioed logic

n

Optimizing a logic gate for area, speed, energy, or robustness

n

Low-power and high-performance circuit-design techniques

6.1 Introduction

6.2 Static CMOS Design

- 6.2.1 Complementary CMOS
- 6.2.2 Ratioed Logic
- 6.2.3 Pass-Transistor Logic

6.3 Dynamic CMOS Design

- 6.3.1 Dynamic Logic: Basic Principles
- 6.3.2 Speed and Power Dissipation of Dynamic Logic
- 6.3.3 Issues in Dynamic Design
- 6.3.4 Cascading Dynamic Gates

6.4 Perspectives

- 6.4.1 How to Choose a Logic Style?
- 6.4.2 Designing Logic for Reduced Supply Voltages

6.5 Summary

6.6 To Probe Further

6.1 Introduction

The design considerations for a simple inverter circuit were presented in the previous chapter. Now, we will extend this discussion to address the synthesis of arbitrary digital gates such as NOR, NAND and XOR. The focus is on *combinational logic* (or *non-regenerative*) circuits; this is, circuits that have the property that at any point in time, the output of the circuit is related to its current input signals by some Boolean expression (assuming that the transients through the logic gates have settled). No intentional connection between outputs and inputs is present.

This is in contrast to another class of circuits, known as *sequential* or *regenerative*, for which the output is not only a function of the current input data, but also of previous values of the input signals (Figure 6.1). This is accomplished by connecting one or more outputs intentionally back to some inputs. Consequently, the circuit "remembers" past events and has a sense of *history*. A sequential circuit includes a combinational logic portion and a module that holds the state. Example circuits are registers, counters, oscillators, and memory. Sequential circuits are the topic of the next Chapter.

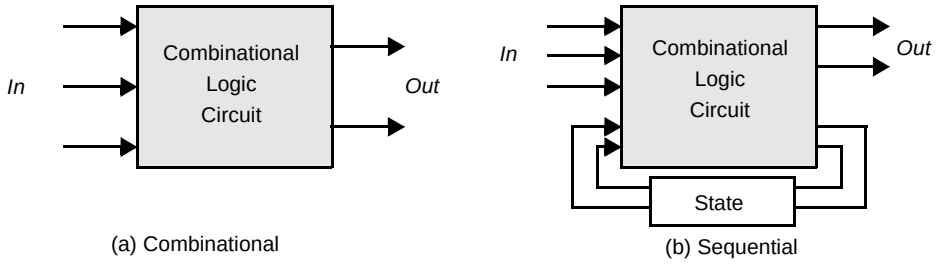


Figure 6.1 High level classification of logic circuits.

There are numerous circuit styles to implement a given logic function. As with the inverter, the common design metrics by which a gate is evaluated are area, speed, energy and power. Depending on the application, the emphasis will be on different metrics. For instance, the switching speed of digital circuits is the primary metric in a high-performance processor, while it is energy dissipation in a battery operated circuit. In addition to these metrics, robustness to noise and reliability are also very important considerations. We will see that certain logic styles can significantly improve performance, but are more sensitive to noise. Recently, power dissipation has also become a very important requirement and significant emphasis is placed on understanding the sources of power and approaches to deal with power.

6.2 Static CMOS Design

The most widely used logic style is static complementary CMOS. The static CMOS style is really an extension of the static CMOS inverter to multiple inputs. In review, the primary advantage of the CMOS structure is robustness (i.e, low sensitivity to noise), good performance, and low power consumption with no static power dissipation. Most of those

properties are carried over to large fan-in logic gates implemented using a similar circuit topology.

The complementary CMOS circuit style falls under a broad class of logic circuits called *static* circuits in which at every point in time (except during the switching transients), each gate output is connected to either V_{DD} or V_{ss} via a low-resistance path. Also, the outputs of the gates assume at all times the value of the Boolean function implemented by the circuit (ignoring, once again, the transient effects during switching periods). This is in contrast to the *dynamic* circuit class, which relies on temporary storage of signal values on the capacitance of high-impedance circuit nodes. The latter approach has the advantage that the resulting gate is simpler and faster. Its design and operation are however more involved and prone to failure due to an increased sensitivity to noise.

In this section, we sequentially address the design of various static circuit flavors including complementary CMOS, ratioed logic (pseudo-NMOS and DCVSL), and pass-transistor logic. The issues of scaling to lower power supply voltages and threshold voltages will also be dealt with.

6.2.1 Complementary CMOS

Concept

A static CMOS gate is a combination of two networks, called the *pull-up network* (PUN) and the *pull-down* network (PDN) (Figure 6.2). The figure shows a generic *N* input logic gate where all inputs are distributed to both the pull-up and pull-down networks. The function of the PUN is to provide a connection between the output and V_{DD} anytime the output of the logic gate is meant to be 1 (based on the inputs). Similarly, the function of the PDN is to connect the output to V_{SS} when the output of the logic gate is meant to be 0. The PUN and PDN networks are constructed in a mutually exclusive fashion such that *one and only one* of the networks is conducting in steady state. In this way, once the transients have settled, a path always exists between V_{DD} and the output *F*, realizing a high output ("one"), or, alternatively, between V_{SS} and *F* for a low output ("zero"). This is equivalent to stating that the output node is always a *low-impedance* node in steady state.

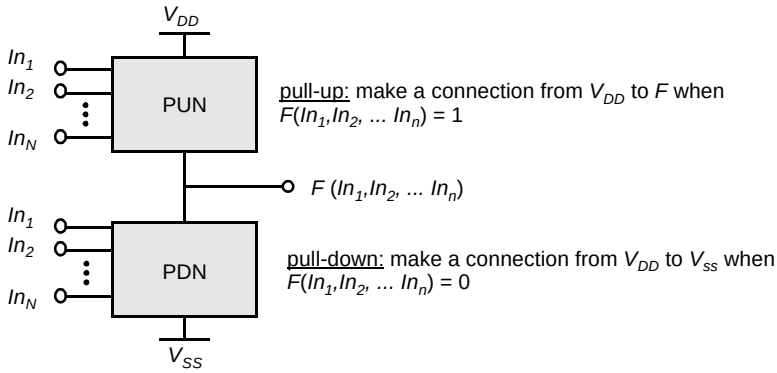


Figure 6.2 Complementary logic gate as a combination of a PUN (pull-up network) and a PDN (pull-down network).

In constructing the PDN and PUN networks, the following observations should be kept in mind:

- A transistor can be thought of as a switch controlled by its gate signal. An NMOS switch is *on* when the controlling signal is high and is *off* when the controlling signal is low. A PMOS transistor acts as an inverse switch that is *on* when the controlling signal is low and *off* when the controlling signal is high.
- The PDN is constructed using NMOS devices, while PMOS transistors are used in the PUN. The primary reason for this choice is that NMOS transistors produce "strong zeros," and PMOS devices generate "strong ones". To illustrate this, consider the examples shown in Figure 6.3. In Figure 6.3a, the output capacitance is initially charged to V_{DD}. Two possible discharge scenarios are shown. An NMOS device pulls the output all the way down to GND, while a PMOS lowers the output no further than $|V_{Tp}|$ — the PMOS turns *off* at that point, and stops contributing discharge current. NMOS transistors are hence the preferred devices in the PDN. Similarly, two alternative approaches to charging up a capacitor are shown in Figure 6.3b, with the output initially at GND. A PMOS switch succeeds in charging the output all the way to V_{DD}, while the NMOS device fails to raise the output above V_{DD}-V_{Tn}. This explains why PMOS transistors are preferentially used in a PUN.

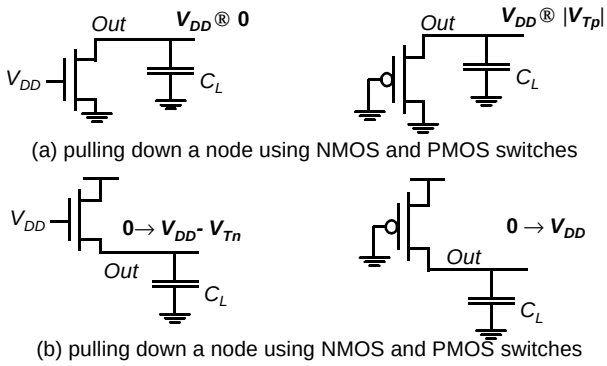


Figure 6.3 Simple examples illustrate why an NMOS should be used as a pull-down, and a PMOS should be used as a pull-up device.

- A set of construction rules can be derived to construct logic functions (Figure 6.4). NMOS devices connected in series corresponds to an AND function. With all the inputs high, the series combination conducts and the value at one end of the chain is transferred to the other end. Similarly, NMOS transistors connected in parallel represent an OR function. A conducting path exists between the output and input terminal if at least one of the inputs is high. Using similar arguments, construction rules for PMOS networks can be formulated. A series connection of PMOS conducts if both inputs are low, representing a NOR function ($\overline{A}.\overline{B} = \overline{A+B}$), while PMOS transistors in parallel implement a NAND ($\overline{A}+\overline{B} = \overline{A \cdot B}$.
- Using De Morgan's theorems ($(\overline{A + B}) = A \cdot B$ and $\overline{A \cdot B} = \overline{A} + \overline{B}$), it can be shown that the pull-up and pull-down networks of a complementary CMOS structure are *dual* networks. This means that a parallel connection of transistors in the pull-up network corresponds to a series connection of the corresponding devices in the pull-down

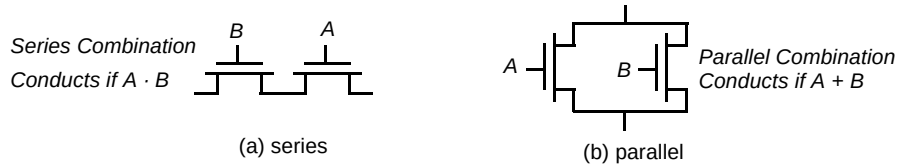


Figure 6.4 NMOS logic rules — series devices implement an AND, and parallel devices implement an OR.

network, and vice versa. Therefore, to construct a CMOS gate, one of the networks (e.g., PDN) is implemented using combinations of series and parallel devices. The other network (i.e., PUN) is obtained using duality principle by walking the hierarchy, replacing series sub-nets with parallel sub-nets, and parallel sub-nets with series sub-nets. The complete CMOS gate is constructed by combining the PDN with the PUN.

- The complementary gate is naturally *inverting*, implementing only functions such as NAND, NOR, and XNOR. The realization of a non-inverting Boolean function (such as AND OR, or XOR) in a single stage is not possible, and requires the addition of an extra inverter stage.
- The number of transistors required to implement an N-input logic gate is $2N$.

Example 6.1 Two-input NAND Gate

Figure 6.5 shows a two-input NAND gate ($F = \overline{A \cdot B}$). The PDN network consists of two NMOS devices in series that conduct when both A and B are high. The PUN is the dual network, and consists of two parallel PMOS transistors. This means that F is 1 if $A = 0$ or $B = 0$, which is equivalent to $F = \overline{A \cdot B}$. The truth table for the simple two input NAND gate is given in Table 6.1. It can be verified that the output F is always connected to either V_{DD} or GND, but never to both at the same time.

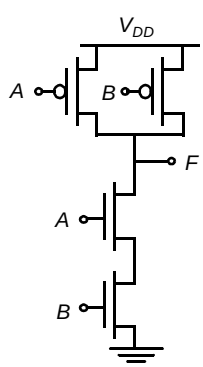


Table 6.1 Truth Table for 2 input NAND

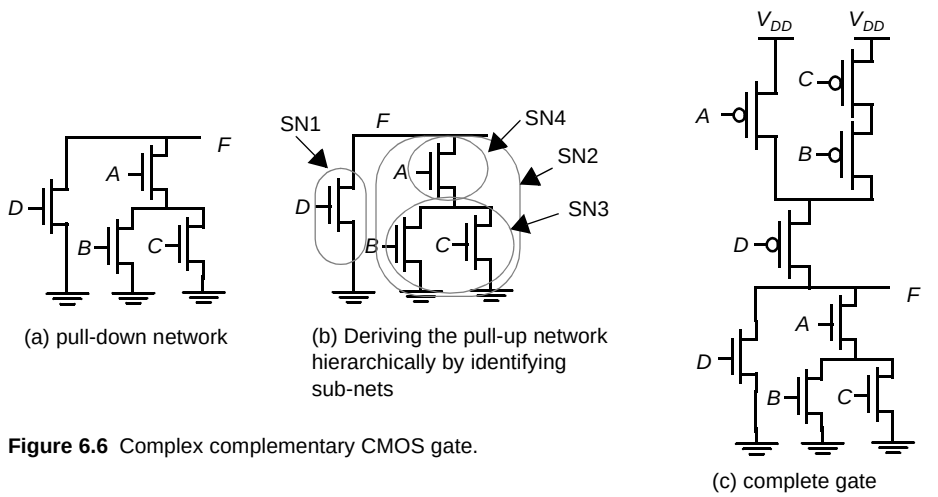
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Figure 6.5 Two-input NAND gate in complementary static CMOS style.

Example 6.2 Synthesis of complex CMOS Gate

Using complementary CMOS logic, consider the synthesis of a complex CMOS gate whose function is $F = \overline{D + A \cdot (B + C)}$. The first step in the synthesis of the logic gate is to derive the pull-down network as shown in Figure 6.6a by using the fact that NMOS devices in series

implements the AND function and parallel device implements the OR function. The next step is to use duality to derive the PUN in a hierarchical fashion. The PDN network is broken into smaller networks (i.e., subset of the PDN) called sub-nets that simplify the derivation of the PUN. In Figure 6.6b, the sub-nets (SN) for the pull-down network are identified At the top level, SN1 and SN2 are in parallel so in the dual network, they will be in series. Since SN1 consists of a single transistor, it maps directly to the pull-up network. On the other hand, we need to recursively apply the duality rules to SN2. Inside SN2, we have SN3 and SN4 in series so in the PUN they will appear in parallel. Finally, inside SN3, the devices are in parallel so they appear in series in the PUN. The complete gate is shown in Figure 6.6c. The reader can verify that for every possible input combination, there always exists a path to either V_{DD} or *GND*.



(a) pull-down network

(b) Deriving the pull-up network hierarchically by identifying sub-nets

(c) complete gate

Figure 6.6 Complex complementary CMOS gate.

Static Properties of Complementary CMOS Gates

Complementary CMOS gates inherit all the nice properties of the basic CMOS inverter. They exhibit rail to rail swing with $V_{OH} = V_{DD}$ and $V_{OL} = GND$. The circuits also have no static power dissipation, since the circuits are designed such that the pull-down and pull-up networks are mutually exclusive. The analysis of the DC voltage transfer characteristics and the noise margins is more complicated then for the inverter, as these parameters depend upon the data input patterns applied to gate.

Consider the static two-input NAND gate shown in Figure 6.7. Three possible input combinations switch the output of the gate from high-to-low: (a) $A = B = 0 \rightarrow 1$, (b) $A= 1$, $B = 0 \rightarrow 1$, and (c) $B= 1$, $A = 0 \rightarrow 1$. The resulting voltage transfer curves display significant differences. The large variation between case (a) and the others (b & c) is explained by the fact that in the former case both transistors in the pull-up network are on simultaneously for A=B=0, representing a strong pull-up. In the latter cases, only one of the pull-up devices is on. The VTC is shifted to the left as a result of the weaker PUN.

The difference between (b) and (c) results mainly from the state of the internal node *int* between the two NMOS devices. For the NMOS devices to turn on, both gate-to-source voltages must be above V_{Tn}, with $V_{GS2} = V_A - V_{DS1}$ and $V_{GS1} = V_B$. The threshold

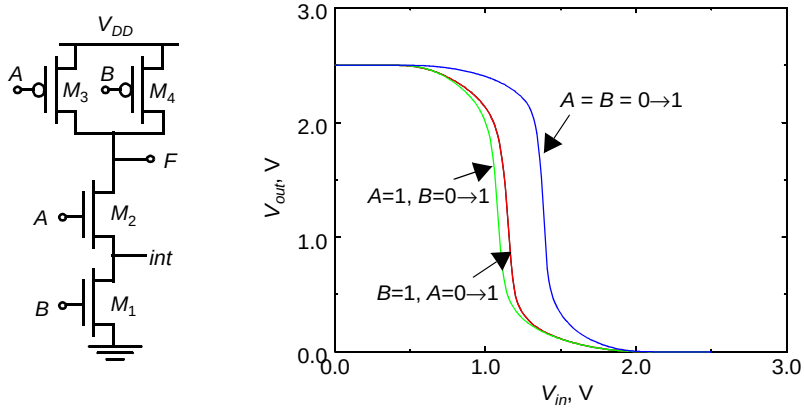


Figure 6.7 The VTC of a two-input NAND is data-dependent. NMOS devices are 0.5µm/0.25µm while the PMOS devices are sized at 0.75µm/0.25µm.

voltage of transistor M_2 will be higher than transistor M_1 due to the body effect. The threshold voltages of the two devices are given by:

$$V_{Tn2} = V_{tn0} + \gamma((\sqrt{|2\phi_f| + V_{int}}) - \sqrt{|2\phi_f|}) \tag{6.1}$$

$$V_{Tn1} = V_{tn0} \tag{6.2}$$

For case (b), M_3 is turned *off*, and the gate voltage of M_2 is set to V_{DD}. To a first order, M_2 may be considered as a resistor in series with M_1. Since the drive on M_2 is large, this resistance is small and has only a small effect on the voltage transfer characteristics. In case (c), transistor M1 acts as a resistor, causing body effect in M_2. The overall impact is quite small as seen from the plot.

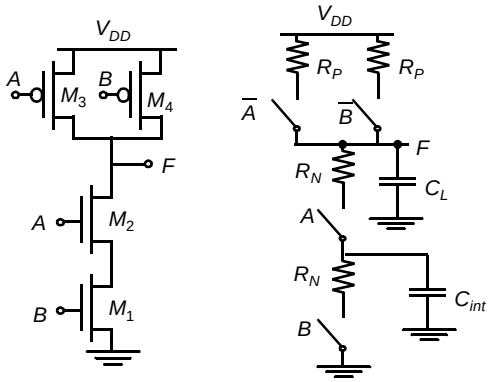
Design Consideration

The important point to take away from the above discussion is that the **noise margins are input-pattern dependent.** For the above example, a glitch on only one of the two inputs has a larger chance of creating a false transition at the output than when the glitch would occur on both inputs simultaneously. Therefore, the former condition has a lower low noise margin. A common practice when characterizing gates such as NAND and NOR is to connect all the inputs together. This unfortunately does not represent the worst-case static behavior. The data dependencies should be carefully modeled.

Propagation Delay of Complementary CMOS Gates

The computation of propagation delay proceeds in a fashion similar to the static inverter. For the purpose of delay analysis, each transistor is modeled as a resistor in series with an ideal switch. The value of the resistance is dependent on the power supply voltage and an equivalent large signal resistance, scaled by the ratio of device width over length, must be

used. The logic is transformed into an equivalent RC network that includes the effect of internal node capacitances. Figure 6.8 shows the two-input NAND gate and its equivalent RC switch level model. Note that the internal node capacitance C_{int} —attributable to the source/drain regions and the gate overlap capacitance of M_2/M_1— is included. While complicating the analysis, the capacitance of the internal nodes can have quite an impact in some networks such as large fan-in gates. In a first pass, we ignore the effect of the internal capacitance.



(a) Two-input NAND (b) RC equivalent model

Figure 6.8 Equivalent RC model for a 2-input NAND gate.

A simple analysis of the model shows that—similar to the noise margins—**the propagation delay depends upon the input patterns**. Consider for instance the low-to-high transition. Three possible input scenarios can be identified for charging the output to V_{DD}. If both inputs are driven low, the two PMOS devices are on. The delay in this case is $0.69 \times (R_p/2) \times C_L$, since the two resistors are in parallel. This is not the worst-case low-to-high transition, which occurs when only one device turns on, and is given by $0.69 \times R_p \times C_L$. For the pull-down path, the output is discharged only if both A and B are switched high, and the delay is given by $0.69 \times (2R_N) \times C_L$ to a first order. In other words, adding devices in series slows down the circuit, and devices must be made wider to avoid a performance penalty. When sizing the transistors in a gate with multiple fan-in's, we should pick the combination of inputs that triggers the worst-case conditions.

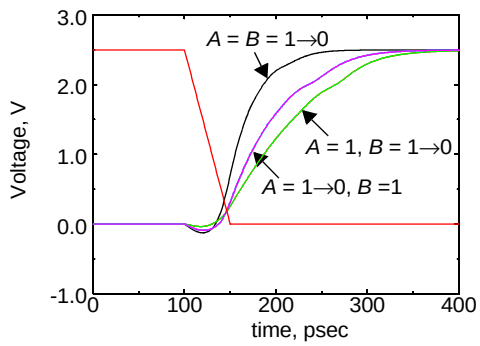
For example, for a NAND gate to have the same pull-down delay (t_{phl}) as a minimum-sized inverter, the NMOS devices in the NAND stack must be made twice as wide so that the equivalent resistance the NAND pull-down is the same as the inverter. The PMOS devices can remain unchanged.

This first-order analysis assumes that the extra capacitance introduced by widening the transistors can be ignored. This is not a good assumption in general, but allows for a reasonable first cut at device sizing.

Example 6.3 Delay dependence on input patterns

Consider the NAND gate of Figure 6.8a. Assume NMOS and PMOS devices of 0.5μm/0.25μm and 0.75μm/0.25μm, respectively. This sizing should result in approximately equal worst-case rise and fall times (since the effective resistance of the pull-down is designed to be equal to the pull-up resistance).

Figure 6.9 shows the simulated low-to-high delay for different input patterns. As expected, the case where both inputs transition go low ($A = B = 1 \rightarrow 0$) results in a smaller delay, compared to the case where only one input is driven low. Notice that the worst-case low-to-high delay depends upon which input (A or B) goes low. The reason for this involves the internal node capacitance of the pull-down stack (i.e., the source of M_2). For the case that $B = 1$ and A transitions from $1 \rightarrow 0$, the pull-up PMOS device only has to charge up the output node capacitance since M_2 is turned off. On the other hand, for the case where A=1 and B transitions from $1 \rightarrow 0$, the pull-up PMOS device has to charge up the sum of the output and the internal node capacitances, which slows down the transition.



Input Data Pattern	Delay (psec)
***A* = *B*= 0®1**	69
***A* = 1, *B*= 0®1**	62
***A*= 0®1, *B* = 1**	50
***A*=*B*=1®0**	35
***A*=1, *B* = 1®0**	76
***A*= 1®0, *B* = 1**	57

Figure 6.9 Example showing the delay dependence on input patterns.

The table in Figure 6.9 shows a compilation of various delays for this circuit. The first-order transistor sizing indeed provides approximately equal rise and fall delays. An important point to note is that the high-to-low propagation delay depends on the state of the internal nodes. For example, when both inputs transition from $0 \rightarrow 1$, it is important to establish the state of the internal node. The worst-case happens when the internal node is charged up to V_{DD}-V_{Tn}. The worst case can be ensured by pulsing the A input from $1 \rightarrow 0 \rightarrow 1$, while input B only makes the $0 \rightarrow 1$. In this way, the internal node is initialized properly.

The important point to take away from this example is that estimation of delay can be fairly complex, and requires a careful consideration of internal node capacitances and data patterns. Care must be taken to model the worst-case scenario in the simulations. A brute force approach that applies all possible input patterns, may not always work as it is important to consider the state of internal nodes.

The CMOS implementation of a NOR gate ($F = \overline{A + B}$) is shown in Figure 6.10. The output of this network is high, if and only if both inputs A and B are low. The worst-case pull-down transition happens when only one of the NMOS devices turns on (i.e., if either A or B is high). Assume that the goal is to size the NOR gate such that it has approximately the same delay as an inverter with the following device sizes: NMOS 0.5µm/0.25µm and PMOS 1.5µm/0.25µm. Since the pull-down path in the worst case is a single device, the NMOS devices (M_1 and M_2) can have the same device widths as the NMOS device in the inverter. For the output to be pulled high, both devices must be turned on. Since the resistances add, the devices must be made two times larger compared to the PMOS in the inverter (i.e., M_3 and M_4 must have a size of 3µm/0.25µm). Since PMOS devices have a lower mobility relative to NMOS devices, stacking devices in series

must be avoided as much as possible. A NAND implementation is clearly preferred over a NOR implementation for implementing generic logic.

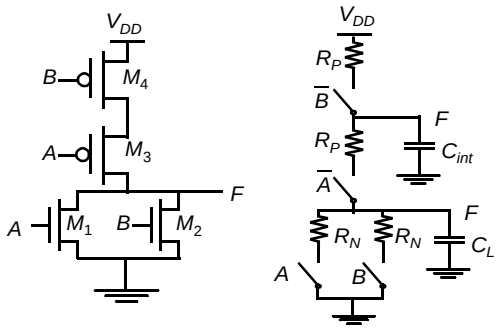


Figure 6.10 Sizing of a NOR gate to produce the same delay as an inverter with size of NMOS: 0.5µm/0.25µm and PMOS: 1.5µm/0.25µm.

Problem 6.1 Transistor Sizing in Complementary CMOS Gates

Determine the transistor sizes of the individual transistors in Figure 6.6c such that it has approximately the same t_{plh} and t_{phl} as a inverter with the following sizes: NMOS: 0.5µm/0.25µm and PMOS: 1.5µm/0.25µm.

So far in the analysis of propagation delay, we have ignored the effect of internal node capacitances. This is often a reasonable assumption for a first-order analysis. However, in more complex logic gates that have large *fan-in*, the internal node capacitances can become significant. Consider a 4-input NAND gate as shown in Figure 6.11, which shows the equivalent RC model of the gate, including the internal node capacitances. The internal capacitances consist of the junction capacitance of the transistors, as well as the gate-to-source and gate-to-drain capacitances. The latter are turned into capacitances to ground using the Miller equivalence. The delay analysis for such a circuit involves solving distributed RC networks, a problem we already encountered when analyzing the delay of interconnect networks. Consider the pull-down delay of the circuit. The output is discharged when all inputs are driven high. The proper initial conditions must be placed on the internal nodes (this is, the internal nodes must be charged to V_{DD}-V_{TN}) before the inputs are driven high.

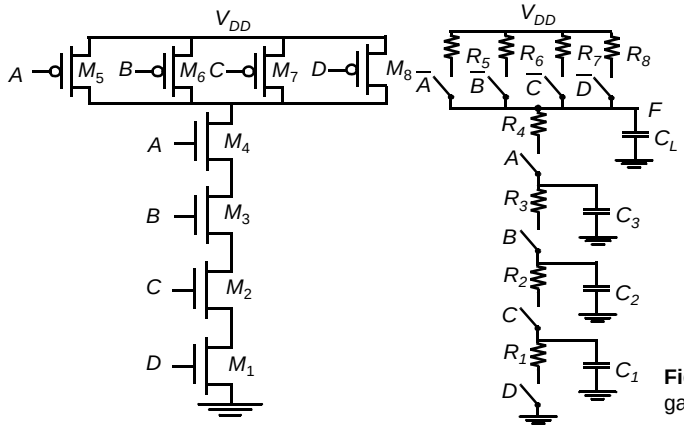


Figure 6.11 Four input NAND gate and its RC model.

The propagation delay can be computed using the Elmore delay model and is approximated as:

$$t_{pHL} = 0.69(R_1 \cdot C_1 + (R_1 + R_2) \cdot C_2 + (R_1 + R_2 + R_3) \cdot C_3 + (R_1 + R_2 + R_3 + R_4) \cdot C_L) \tag{6.3}$$

Notice that the resistance of M_1 appears in all the terms, which makes this device especially important when attempting to minimize delay. Assuming that all NMOS devices have an equal size, Eq. (6.3) simplifies to

$$t_{pHL} = 0.69R_N(C_1 + 2 \cdot C_2 + 3 \cdot C_3 + 4 \cdot C_L) \tag{6.4}$$

Example 6.4 A Four-Input Complementary CMOS NAND Gate

In this example, the intrinsic *propagation delay* of the 4 input NAND gate (without any loading) is evaluated using hand analysis and simulation. Assume that all NMOS devices have a *W*/*L* of 0.5µm/0.25µm, and all PMOS devices have a device size of 0.375µm/0.25µm. The layout of a four-input NAND gate is shown in Figure 6.12. The devices are sized such that the worst case rise and fall time are approximately equal (to first order ignoring the internal node capacitances).

Using techniques similar to those employed for the CMOS inverter in Chapter 3, the capacitances values can be computed from the layout. Notice that in the pull-up path, the PMOS devices share the drain terminal in order to reduce the overall parasitic contribution to the output. Using our standard design rules, the area and perimeter for various devices can be easily computed as shown in Table 6.1

In this example, we will focus on the pull-down delay, and the capacitances will be computed for the high-to-low transition at the output. While the output makes a transition from V_{DD} to 0, the internal nodes only transition from V_{DD}-V_{Tn} to GND. We would need to linearize the internal junction capacitances for this voltage transition, but, to simplify the analysis, we will use the same K_{eff} for the internal nodes as for the output node.

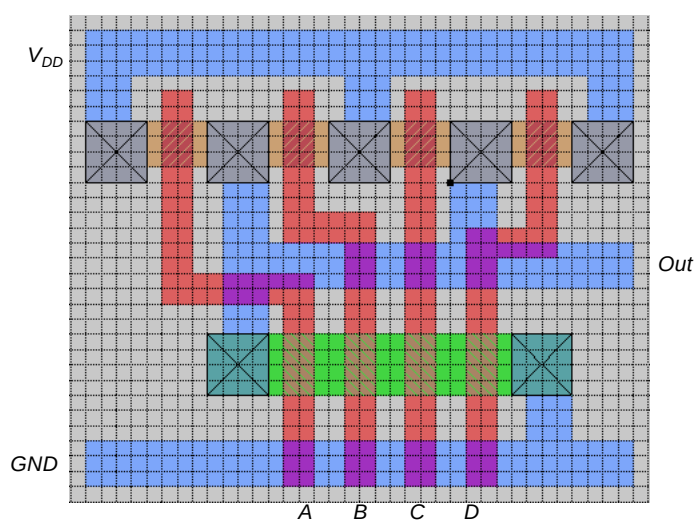


Figure 6.12 Layout a four-input NAND gate in complementary CMOS.

Table 6.1 Area and perimeter of transistors in 4 input NAND gate.

Transistor	W (mm)	AS (mm^2)	AD (mm^2)	PS (mm)	PD(mm)
1	0.5	0.3125	0.0625	1.75	0.25
2	0.5	0.0625	0.0625	0.25	0.25
3	0.5	0.0625	0.0625	0.25	0.25
4	0.5	0.0625	0.3125	0.25	1.75
5	0.375	0.296875	0.171875	1.875	0.875
6	0.375	0.171875	0.171875	0.875	0.875
7	0.375	0.171875	0.171875	0.875	0.875
8	0.375	0.296875	0.171875	1.875	0.875

It is assumed that the output connects to a single, minimum-size inverter. The effect of intra-cell routing, which is small, is ignored. The various contributions are summarized in Table 6.2. For the NMOS and PMOS junctions, we use K_{eq} = 0.57, K_{eqsw} = 0.61, and K_{eq} = 0.79, K_{eqsw} = 0.86, respectively. Notice that the gate-to-drain capacitance is multiplied by a factor of two for all internal nodes and the output node to account for the Miller effect (this ignores the fact that the internal nodes have a slightly smaller swing due to the threshold drop).

Table 6.2 Computation of capacitances for high-to-low transition at the output. The table shows the intrinsic delay of the gate without extra loading. Any *fan-out* capacitance would simply be added to the C_L term.

Capacitor	Contributions (H→L)	Value (fF) (H→L)
C_1	$C_{d1} + C_{s2} + 2 * C_{gd1} + 2 * C_{gs2}$	(0.57 * 0.0625 * 2+ 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2+ 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85fF
C_2	$C_{d2} + C_{s3} + 2 * C_{gd2} + 2 * C_{gs3}$	(0.57 * 0.0625 * 2+ 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2+ 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85fF
C_3	$C_{d3} + C_{s4} + 2 * C_{gd3} + 2 * C_{gs4}$	(0.57 * 0.0625 * 2+ 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2+ 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85fF
C_L	$C_{d4} + 2 * C_{gd4} + C_{d5} + C_{d6} + C_{d7} + C_{d8} + 2 * C_{gd5} + 2 * C_{gd6} + 2 * C_{gd7} + 2 * C_{gd8} = C_{d4} + 4 * C_{d5} + 4 * 2 * C_{gd6}$	(0.57 * 0.3125 * 2 + 0.61 * 1.75 *0.28) + 2 * (0.31 * 0.5)+ 4 * (0.79 * 0.171875* 1.9+ 0.86 * 0.875 * 0.22)+ 4 * 2 * (0.27 * 0.375) = 3.47fF

Using Eq. (6.4), we can compute the propagation delay as:

$$t_{pHL} = 0.69\left(\frac{13K\Omega}{2}\right)(0.85fF + 2 \cdot 0.85fF + 3 \cdot 0.85fF + 4 \cdot 3.47fF) = 85ps$$

The simulated delay for this particular transition was found to be 86 psec! The hand analysis gives a fairly accurate estimate given all assumptions and linearizations made. For example, we assume that the gate-source (or gate-drain) capacitance only consists of the overlap component. This is not entirely the case, as during the transition some other contributions come in place depending upon the operating region. Once again, the goal of hand analysis is not to

provide a totally accurate delay prediction, but rather to give intuition into what factors influence the delay and to aide in initial transistor sizing. Accurate timing analysis and transistor optimization is usually done using SPICE. The simulated worst-case low-to-high delay time for this gate was 106ps.

While complementary CMOS is a very robust and simple approach for implementing logic gates, there are two major problems associated with using this style as the complexity of the gate (i.e., *fan-in*) increases. First, the number of transistors required to implement an *N* fan-in gate is *2N*. This can result in significant implementation area. The second problem is that propagation delay of a complementary CMOS gate deteriorates rapidly as a function of the fan-in. The large number of transistors (*2N*) increases the overall capacitance of the gate. For an *N*-input NAND gate, the output capacitance increases linearly with the fan-in since the number of PMOS devices connected to the output node increases linearly with the fan-in. Also, a series connection of transistors in either the PUN or PDN slows the gate as well, because the effective (dis)charging resistance is increased. For the same *N*-input NAND gate, the effective resistance of the PDN path increases linearly with the fan-in. Since the output capacitance increase linearly and the pull-down resistance increases linearly, the high-to-low delay can increase in a quadratic fashion.

The *fan-out* has a large impact on the delay of complementary CMOS logic as well. Each input to a CMOS gate connects to both an NMOS and a PMOS device, and presents a load to the driving gate equal to the sum of the gate capacitances.

The above observations are summarized by the following formula, which approximates the influence of *fan-in* and *fan-out* on the propagation delay of the complementary CMOS gate

$$t_p = a_1 FI + a_2 FI^2 + a_3 FO \tag{6.5}$$

where *FI* and *FO* are the *fan-in* and *fan-out* of the gate, respectively, and a_1, a_2 and a_3 are weighting factors that are a function of the technology.

At first glance, it would appear that the increase in resistance for larger *fan-in* can be solved by making the devices in the transistor chain wider. Unfortunately, this does not improve the performance as much as expected, since widening a device also increases its gate and diffusion capacitances, and has an adverse affect on the gate performance. For the *N*-input NAND gate, the low-to-high delay only increases linearly since the pull-up resistance remains unchanged and only the capacitance increases linearly.

Figure 6.13 show the propagation delay for both transitions as a function of fan-in assuming a fixed fan-out (NMOS: 0.5μm and PMOS: 1.5μm). As predicted above, the t_{pLH} increases linearly due to the linearly-increasing value of the output capacitance. The simultaneous increase in the pull-down resistance and the load capacitance results in an approximately quadratic relationship for t_{pHL}. Gates with a *fan-in* greater than or equal to 4 become excessively slow and must be avoided.

Design Techniques for Large Fan-in

Several approaches may be used to reduce delays in large fan-in circuits.

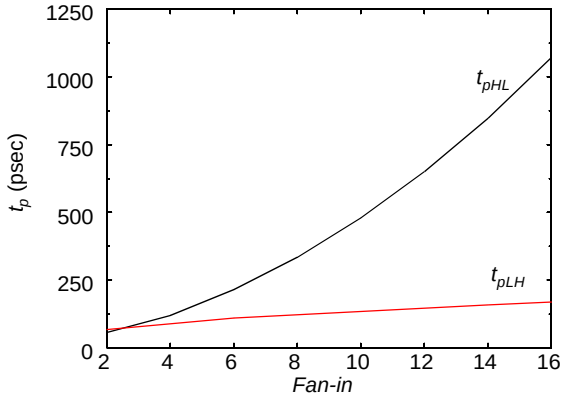


Figure 6.13 Propagation delay of CMOS NAND gate as a function of fan-in. A fan-out of one inverter is assumed, and all pull-down transistors are minimal size.

1. Transistor Sizing

The most obvious solution is to increase the overall transistor size. This lowers the resistance of devices in series and lowers the time constant. However, increasing the transistor size, results in larger parasitic capacitors, which do not only affect the *propagation delay* of the gate in question, but also present a larger load to the preceding gate. This technique should, therefore, be used with caution. If the load capacitance is dominated by the intrinsic capacitance of the gate, widening the device only creates a "self-loading" effect, and the *propagation delay* is unaffected. A more comprehensive approach towards sizing transistors in complex CMOS gates is discussed in the next section.

2. Progressive Transistor Sizing

An alternate approach to uniform sizing (in which each transistor is scaled up uniformly), is to use progressive transistor sizing (Figure 6.14). Referring back to Eq. (6.3), we see that the resistance of M_1 (R_1) appears N times in the delay equation, the resistance of M_2 (R_2) appears N-1 times, etc. From the equation, it is clear that R_1 should be made the smallest, R_2 the next smallest, etc. Consequently, a progressive scaling of the transistors is beneficial: $M_1 > M_2 > M_3 > M_N$. Basically, in this approach, the important resistance is reduced while reducing capacitance. For an excellent treatment on the optimal sizing of transistors in a complex network, we refer the interested reader to [Shoji88, pp. 131–143]. The reader should be aware of

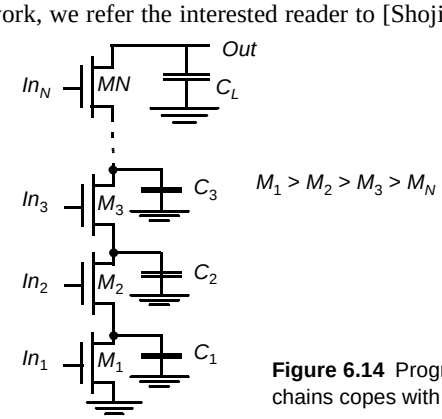


Figure 6.14 Progressive sizing of transistors in large transistor chains copes with the extra load of internal capacitances.

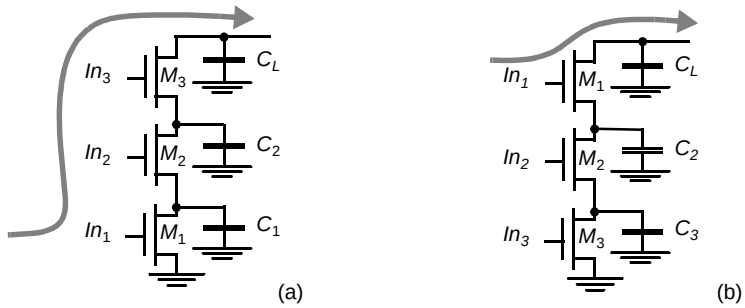


Figure 6.15 Influence of transistor ordering on delay. Signal In_1 is the critical signal.

one important pitfall of this approach. While progressive resizing of transistors is relatively easy in a schematic diagram, it is not as simple in a real layout. Very often, design-rule considerations force the designer to push the transistors apart, which causes the internal capacitance to grow. This may offset all the gains of the resizing!

3. Input Re-Ordering

Some signals in complex combinational logic blocks might be more critical than others. Not all inputs of a gate arrive at the same time (due, for instance, to the propagation delays of the preceding logical gates). An input signal to a gate is called *critical* if it is the last signal of all inputs to assume a stable value. The path through the logic which determines the ultimate speed of the structure is called the *critical path*.

Putting the critical-path transistors closer to the output of the gate can result in a speed-up. This is demonstrated in Figure 6.15. Signal In_1 is assumed to be a critical signal. Suppose further that In_2 and In_3 are high and that In_1 undergoes a $0 \rightarrow 1$ transition. Assume also that C_L is initially charged high. In case (a), no path to *GND* exists until M_1 is turned on, which is unfortunately the last event to happen. The delay between the arrival of In_1 and the output is therefore determined by the time it takes to discharge C_L, C_1 and C_2. In the second case, C_1 and C_2 are already discharged when In_1 changes. Only C_L still has to be discharged, resulting in a smaller delay.

4. Logic Restructuring

Manipulating the logic equations can reduce the fan-in requirements and hence reduce the gate delay, as illustrated in Figure 6.16. The quadratic dependency of the gate delay on *fan-in* makes the six-input NOR gate extremely slow. Partitioning the NOR-gate into two three-input gates results in a significant speed-up, which offsets by far the extra delay incurred by turning the inverter into a two-input NAND gate.

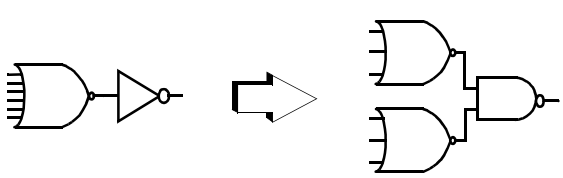


Figure 6.16 Logic restructuring can reduce the gate fan-in.

Transistor Sizing for Performance in Combinational Networks

Earlier, we established that minimization of the propagation delay of a gate in isolation is a purely academic effort. The sizing of devices should happen in its proper context. In Chapter 5, we developed a methodology to do so for inverters. In Chapter 5 we found out that an optimal fanout for a chain of inverters driving a load C_L is $(C_L/C_{in})^{1/N}$, where N is the number of stages in the chain, and C_{in} the input capacitance of the first gate in the chain. If we have an opportunity to select the number of stages, we found out that we would like to keep the fanout per stage around 4. Can this result be extended to determine the size of any combinational path for minimal delay? By extending our previous approach to address complex logic networks, we will find out that this is indeed possible [Sutherland99].[1]

To do so, we modify the basic delay equation of the inverter, introduced in Chapter 5, and repeated here for the sake of clarity,

$$t_p = t_{p0}\left(1 + \frac{C_{ext}}{\gamma C_g}\right) = t_{p0}(1 + f/\gamma) \tag{6.6}$$

to

$$t_p = t_{p0}(p + gf/\gamma) \tag{6.7}$$

with t_{p0} still representing the intrinsic delay of an inverter, and f the ratio between the external load and the input capacitance of the gate. In this context, f is often called the *electrical effort*. p represents the ratio of the intrinsic (or unloaded) delays of the complex gate and the simple inverter. The more involved structure of the multiple-input gate, combined with its series devices, increases its intrinsic delay. p is a function of gate topology as well as layout style. Table 6.3 enumerates the values of p for some standard gates, assuming simple layout styles, and ignoring second-order effects such as internal node capacitances.

Table 6.3 Estimates of intrinsic delay factors of various logic types assuming simple layout styles, and a fixed PMOS/NMOS ratio.

Gate type	p
Inverter	1
n-input NAND	n
n-input NOR	n
n-way multiplexer	$2n$
XOR, NXOR	$n2^{n-1}$

[1] The approach introduced in this section is commonly called logical effort, and was first introduced in [Sutherland99], which presents an extensive treatment of the topic. The treatment offered here represents only a glance-over of the overall approach.

The factor g is called the *logical effort*, and represents the fact that, for a given load, complex gates have to work harder than an inverter to produce a similar response. In other words, the logical effort of a logic gate tells how much worse it is at producing output current than an inverter, given that each of its inputs may contain only the same input capacitance as the inverter. Equivalently, logical effort is how much more input capacitance a gate presents to deliver the same output current as an inverter. Logical effort is a useful parameter, because it depends only on circuit topology. The logical efforts of some common logic gates are given in Table 6.4.

Table 6.4 Logic efforts of common logic gates, assuming a PMOS/NMOS ratio of 2.

Gate Type	Number of Inputs			
	1	**2**	**3**	**n**
Inverter	1			
NAND		4/3	5/3	(n+2)/3
NOR		5/3	7/3	(2n+1)/3
Multiplexer		2	2	2
XOR		4	12	

Example 6.5 Logical effort of complex gates

Consider the gates shown in Figure 6.17. Assuming an PMOS/NMOS ratio of 2, the input capacitance of a minimum-sized symmetrical inverter equals 3 times the gate capacitance of a minimum-sized NMOS (called C_{unit}). We size the 2-input NAND and NOR such that their equivalent resistances equal the resistance of the inverter (using the techniques described earlier). This increases the input capacitance of the 2-input NOR to 4 C_{unit}, or 4/3 the capacitance of the inverter.The input capacitance of the 2-input NOR is 5/3 that of the inverter. Equivalently, for the same input capacitance, the NAND and NOR gate have 4/3 and 5/3 less driving strength than the inverter. This affects the delay component that corresponds to the load, increasing it by this same factor, called 'logical effort.' Hence, g_{NAND} = 4/3, and g_{NOR} = 5/3.

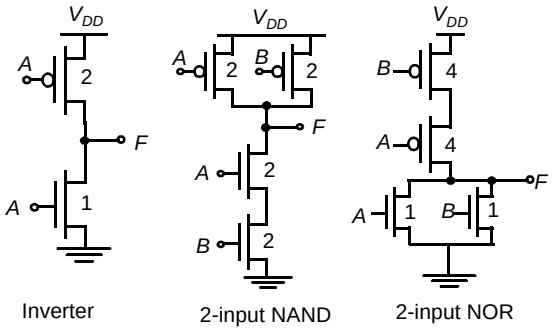


Figure 6.17 Logical effort of 2-input NAND and NOR gates.

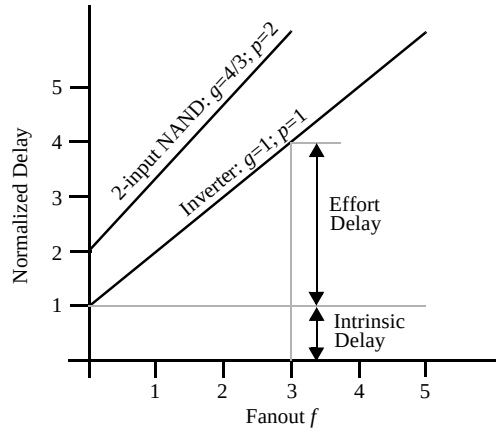


Figure 6.18 Delay as a function of fanout for an inverter and a 2-input NAND.

The delay model of a logic gate, as represented in Eq. (6.7), is a simple linear relationship. Figure 6.18 shows this relationship graphically: the delay is plotted as a function of the fanout (electrical effort) for an inverter and for a 2-input NAND gate. The slope of the line is the logical effort of the gate; its intercept is the intrinsic delay. The graph shows that we can adjust the delay by adjusting the effective fanout (by transistor sizing) or by choosing a logic gate with a different logical effort. Observe also that fanout and logical effort contribute to the delay in a similar way. We call the product of the two $h = fg$ the *gate effort*.

The total delay of a path through a combinational logic block can now be expressed as

$$t_p = \sum_{j=1}^{N} t_{p,j} = t_{p0} \sum_{j=1}^{N} \left(p_j + \frac{f_j g_j}{\gamma} \right) \tag{6.8}$$

We use a similar procedure as we did for the inverter chain in Chapter 5 to determine the minimum delay of the path. By finding $N - 1$ partial derivatives and setting theme to zero, we find that each stage should bear the same 'effort':

$$f_1 g_1 = f_2 g_2 = \ldots = f_N g_N \tag{6.9}$$

The fanouts along the path can be multiplied to get a *path effective fanout*, and so can the logical efforts.

$$\begin{aligned} F &= f_1 f_2 \ldots f_N = C_L / C_{g1} \\ G &= g_1 g_2 \ldots g_N \end{aligned} \tag{6.10}$$

The path effort can then be defined as the product of the two, or $H = FG$. From here on, the analysis proceeds along the same lines as the inverter chain. The gate effort that minimizes the path delay is found to equal

$$h = \sqrt[N]{FG} = \sqrt[N]{H}, \tag{6.11}$$

and the minimum delay through the path is

$$D = t_{p0} \left(\sum_{j=1}^{N} p_j + \frac{N(\sqrt[N]{H})}{\gamma} \right) \tag{6.12}$$

Note that the overall intrinsic delay is a function of the types of logic gates in the path, and is not affected by the sizing.

Example 6.6 Sizing combinational logic for minimum delay

Consider the logic network of Figure 6.19, which may represent the critical path of a more complex logic block. The output of the network is loaded with a capacitance which is 5 times larger than the input capacitance of the first gate, which is a minimum-sized inverter. The effective fanout of the path hence equals $F = C_L/C_{g1} = 5$. Using the entries in Table 6.4, we find the path logical effort

$$G = 1 \times \frac{5}{3} \times \frac{5}{3} \times 1 = \frac{25}{9}$$

$H = FG = 125/9$, and the optimal stage effort h is $\sqrt[4]{H} = 1.93$. Taking into account the gate types, we derive the fanout factors: $f_1 = 1.93$; $f_2 = 1.93\times(3/5) = 1.16$; $f_3 = 1.16$; $f_4=1.93$. Notice that the inverters are assigned larger electrical efforts than the more complex gates because they are better at driving loads. From this, we can derive the sizes of the gates (with respect to their minimum-sized versions): $a = f_1/g_2 = 1.16$; $b = f_1f_2/g_3= 1.34$; $c = f_1f_2f_3/g_4=2.60$.

These calculations do not have to be very precise. As discussed in the Chapter 5, sizing a gate too large or too small by a factor of 1.5 still result in circuits within 5% of minimum delay. Therefore, the "back of the envelope" hand calculations using this technique are quite effective.

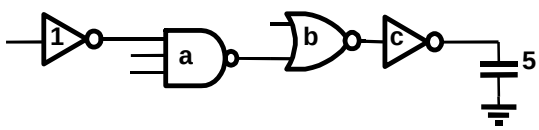


Figure 6.19 Critical path of combinational network.

Power Consumption in CMOS Logic Gates

The sources of power consumption in a complementary CMOS inverter were discussed in detail in Chapter 5. Many of these issues apply directly to complex CMOS gates. The power dissipation is a strong function of transistor sizing (which affects physical capacitance), input and output rise/fall times (which affects the short-circuit power), device thresholds and temperature (which affect leakage power), and switching activity. The dynamic power dissipation is given by $a_{0\to1}\, C_L\, V_{DD}^2\, f$. Making a gate more complex mostly affects the *switching activity* $a_{0\to1}$, which has two components: a static component that is only a function of the topology of the logic network, and a dynamic one that results from the timing behavior of the circuit—the latter factor is also called glitching.

Logic Function—The transition activity is a strong function of the logic function being implemented. For static CMOS gates with statistically independent inputs, the static transition probability is the probability p_0 that the output will be in the *zero* state in one cycle, multiplied by the probability p_1 that the output will be in the *one* state in the next cycle:

$$\alpha_{0\to1} = p_0 \bullet p_1 = p_0 \bullet (1 - p_0) \tag{6.13}$$

Assuming that the inputs are independent and uniformly distributed, any *N*-input static gate has a transition probability that corresponds to

$$\alpha_{0 \to 1} = \frac{N_0}{2^N} \bullet \frac{N_1}{2^N} = \frac{N_0 \bullet \left(2^N - N_0\right)}{2^{2N}} \tag{6.14}$$

where N_0 is the number of *zero* entries and N_1 is the number of *one* entries in the output column of the truth table of the function. To illustrate, consider a static 2-input NOR gate whose truth table is shown in Table 6.5. Assume that only one input transition is possible during a clock cycle, and that the inputs to the NOR gate have a uniform input distribution —this is, the four possible states for inputs *A* and *B* (00, 01, 10, 11) are equally likely.

Table 6.5 Truth table of a 2 input NOR gate.

A	*B*	*Out*
0	0	1
0	1	0
1	0	0
1	1	0

From Table 6.5 and Eq. (6.14), the output transition probability of a 2-input static CMOS NOR gate can be derived:

$$\alpha_{0 \to 1} = \frac{N_0 \bullet \left(2^N - N_0\right)}{2^{2N}} = \frac{3 \bullet \left(2^2 - 3\right)}{2^{2 \bullet 2}} = \frac{3}{16} \tag{6.15}$$

Problem 6.2 *N* input XOR gate

Assuming the inputs to an *N*-input XOR gate are uncorrelated and uniformly distributed, derive the expression for the switching activity factor.

***Signal Statistics*—**The switching activity of a logic gate is a strong function of the input signal statistics. Using a uniform input distribution to compute activity is not a good one since the propagation through logic gates can significantly modify the signal statistics. For example, consider once again a 2-input static NOR gate, and let p_a and p_b be the probabilities that the inputs *A* and *B* are *one*. Assume further that the inputs are not correlated. The probability that the output node equals *one* is given by

$$p_1 = (1\text{-}p_a)\ (1\text{-}p_b) \tag{6.16}$$

Therefore, the probability of a transition from 0 to 1 is

$$a_{0\text{->}1} = p_0\, p_1 = (1\text{-}(1\text{-}p_a)\ (1\text{-}p_b))\ (1\text{-}p_a)\ (1\text{-}p_b) \tag{6.17}$$

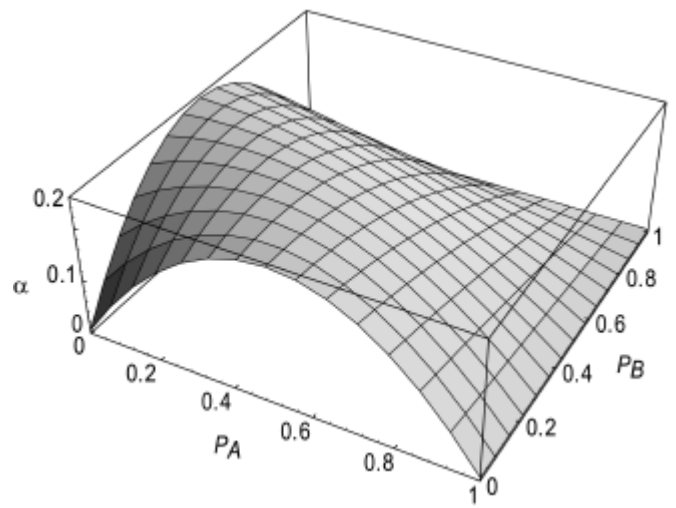


Figure 6.20 Transition activity of a two-input NOR gate as a function of the input probabilities (p_A,p_B)

Figure 6.20 shows the transition probability as a function of p_a and p_b. Observe how this graph degrades into the simple inverter case when one of the input probabilities is set to 0. From this plot, it is clear that understanding the signal statistics and their impact on switching events can be used to significantly impact the power dissipation.

Problem 6.3 Power Dissipation of Basic Logic Gates

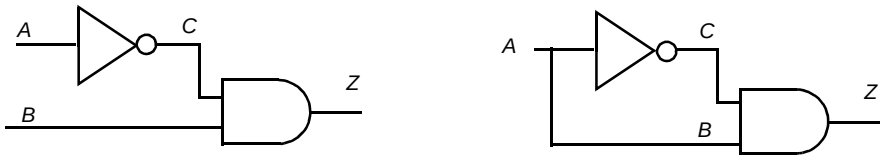
Derive the 0 → 1 output transition probabilities for the basic logic gates (AND, OR, XOR). The results to be obtained are given in Table 6.6.

Table 6.6 Output transition probabilities for static logic gates.

	$a_{0\to1}$
AND	$(1 - p_A p_B)p_A p_B$
OR	$(1 - p_A)(1 - p_B)[1 - (1 - p_A)(1 - p_B)]$
XOR	$[1 - (p_A + p_B - 2p_A p_B)](p_A + p_B - 2p_A p_B)$

***Inter-signal Correlations*—**The evaluation of the switching activity is further complicated by the fact that signals exhibit correlation in space and time. Even if the primary inputs to a logic network are uncorrelated, the signals become correlated or "colored", as they propagate through the logic network. This is best illustrated with a simple example**.** Consider first the circuit shown in Figure 6.21a, and assume that the primary inputs, *A* and *B*, are uncorrelated and uniformly distributed. Node *C* has a **1** (**0**) probability of 1/2, and a 0->1 transition probability of 1/4. The probability that the node *Z* undergoes a power consuming transition is then determined using the AND-gate expression of Table 6.6.

$$p_{0\text{->}1} = (1\text{-}\ p_a\, p_b)\ p_a\, p_b = (1\text{-}1/2 \bullet 1/2)\ 1/2 \bullet 1/2 = 3/16 \qquad (6.18)$$



(a) Logic circuit without reconvergent fanout

(b) Logic circuit with reconvergent fanout

Figure 6.21 Example illustrating the effect of signal correlations.

The computation of the probabilities is straightforward: signal and transition probabilities are evaluated in an ordered fashion, progressing from the input to the output node. This approach, however, has two major limitations: (1) it does not deal with circuits with feedback as found in sequential circuits; (2) it assumes that the signal probabilities at the input of each gate are independent. This is rarely the case in actual circuits, where reconvergent fanout often causes inter-signal dependencies. For instance, the inputs to the AND gate in Figure 6.21b (C and B) are inter-dependent as both are a function of A. The approach to compute probabilities, presented previously, fails under these circumstances. Traversing from inputs to outputs yields a transition probability of 3/16 for node Z, similar to the previous analysis. This value for transition probability is clearly false, as logic transformations show that the network can be reduced to $Z = C \bullet B = A \bullet \overline{A} = 0$, and no transition will ever take place.

To get the precise results in the progressive analysis approach, its is essential to take signal inter-dependencies into account. This can be accomplished with the aid of conditional probabilities. For an AND gate, Z equals 1 if and only if B and C are equal to 1.

$$p_Z = p(Z{=}1) = p(B{=}1, C{=}1) \tag{6.19}$$

where $p(B{=}1,C{=}1)$ represents the probability that B and C are equal to 1 simultaneously. If B and C are independent, $p(B{=}1,C{=}1)$ can be decomposed into $p(B{=}1) \bullet p(C{=}1)$, and this yields the expression for the AND-gate, derived earlier: $p_Z = p(B{=}1) \bullet p(C{=}1) = p_B\, p_C$. If a dependency between the two exists (as is the case in Figure 6.21b), a conditional probability has to be employed, such as

$$p_Z = p(C{=}1|B{=}1) \bullet p(B{=}1) \tag{6.20}$$

The first factor in Eq. (6.20) represents the probability that C=1 given that B=1. The extra condition is necessary as C is dependent upon B. Inspection of the network shows that this probability is equal to 0, since C and B are logical inversions of each other, resulting in the signal probability for Z, $p_Z = 0$.

Deriving those expressions in a structured way for large networks with reconvergent fanout is complex, especially when the networks contain feedback loops. Computer support is therefore essential. To be meaningful, the analysis program has to process a typical sequence of input signals, as the power dissipation is a strong function of statistics of those signals.

***Dynamic or Glitching Transitions*—**When analyzing the transition probabilities of complex, multistage logic networks in the preceding section, we ignored the fact that the gates have a non-zero propagation delay. In reality, the finite propagation delay from one

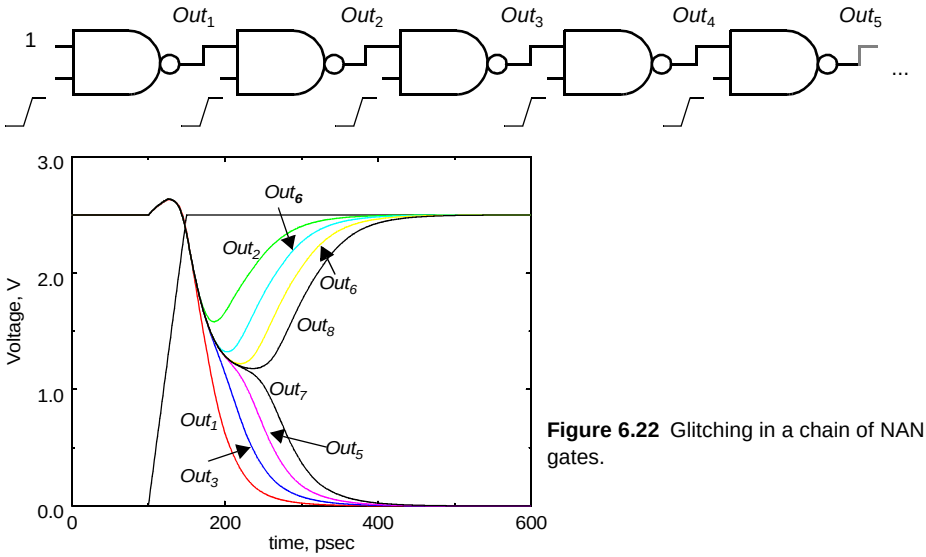


Figure 6.22 Glitching in a chain of NAND gates.

logic block to the next can cause spurious transitions, called *glitches, critical races, or dynamic hazards*, to occur: a node can exhibit multiple transitions in a single clock cycle before settling to the correct logic level.

A typical example of the effect of glitching is shown in Figure 6.22, which displays the simulated response of a chain of NAND gates for all inputs going simultaneously from 0 to 1. Initially, all the outputs are 1 since one of the inputs was 0. For this particular transition, all the odd bits must transition to 0 while the even bits remain at the value of 1. However, due to the finite propagation delay, the higher order even outputs start to discharge and the voltage drops. When the correct input ripples through the network, the output goes high. The glitch on the even bits causes extra power dissipation beyond what is required to strictly implement the logic function. Although the glitches in this example are only partial (i.e., not from rail to rail), they contribute significantly to the power dissipation. Long chains of gates often occur in important structures such as adders and multipliers and the glitching component can easily dominate the overall power consumption.

Design Techniques to Reduce Switching Activity

The dynamic power of a logic gate can be reduced by minimizing the physical capacitance and the switching activity. The physical capacitance can be minimized in a number ways, including circuit style selection, transistor sizing, placement and routing, and architectural optimizations. The switching activity, on the other hand, can be minimized at all level of the design abstraction, and is the focus of this section. Logic structures can be optimized to minimize both the fundamental transitions required to implement a given function, and the spurious transitions.

1. Logic Restructuring

Changing the topology of a logic network may reduce its power dissipation. Consider for instance two alternate implementations of $F = A \bullet B \bullet C \bullet D$, as shown in Figure 6.23. Ignore

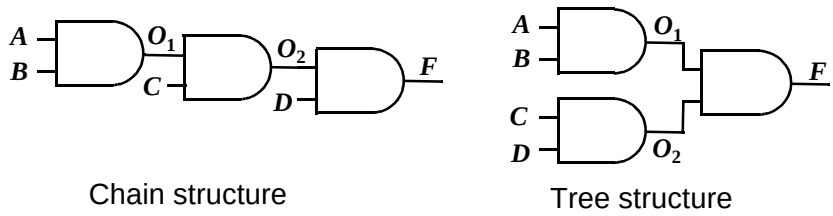


Figure 6.23 Simple example to demonstrate the influence of circuit topology on activity.

glitching and assume that all primary inputs (*A*,*B*,*C*,*D*) are uncorrelated and uniformly distributed (i.e., $p_{1\ (a,b,c,d)} = 0.5$). Using the expressions from Table 6.6, the activity can be computed for the two topologies, as shown in Table 6.7. The results indicate that the chain implementation will have an overall lower switching activity than the tree implementation for random inputs. However, as mentioned before, it is also important to consider the timing behavior to accurately make power trade-offs. In this example the tree topology will have lower (no) glitching activity since the signal paths are balanced to all the gates.

Table 6.7 Probabilities for tree and chain topologies.

	O_1	O_2	F
p_1 (chain)	1/4	1/8	1/16
$p_0 = 1\text{-}p_1$ (chain)	3/4	7/8	15/16
$p_{0\text{->}1}$ (chain)	3/16	**7/64**	15/256
p_1 (tree)	1/4	1/4	1/16
$p_0 = 1\text{-}p_1$ (tree)	3/4	3/4	15/16
$p_{0\text{->}1}$ (tree)	3/16	**3/16**	15/256

2. Input ordering

Consider the two static logic circuits of Figure 6.24. The probabilities of *A*, *B* and C being **1** are listed in the Figure. Since both circuits implement identical logic functionality, it is clear that the activity at the output node *Z* is equal in both cases. The difference is in the activity at the intermediate node. In the first circuit, this activity equals $(1 - 0.5 \times 0.2)\ (0.5 \times 0.2) = 0.09$. In the second case, the probability that a $0 \rightarrow 1$ transition occurs equals $(1 - 0.2 \times 0.1)\ (0.2 \times 0.1) = 0.0196$. This is substantially lower. From this we learn that it is beneficial to postpone the introduction of signals with a high transition rate (i.e., signals with a signal probability close to 0.5). A simple reordering of the input signals is often sufficient to accomplish that goal.

3. Time-multiplexing resources

Time-multiplexing a single hardware resource—such as a logic unit or a bus—over a number functions is an often used technique to minimize the implementation area. Unfortunately, the minimum area solution does not always result in the lowest switching activity. For example, consider the transmission of two input bits (*A* and *B*) using either dedicated resources or a time-multiplexed approach, as shown in Figure 6.25. To first order—ignoring the multiplexer over-

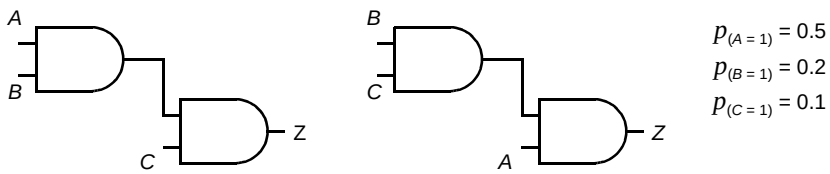
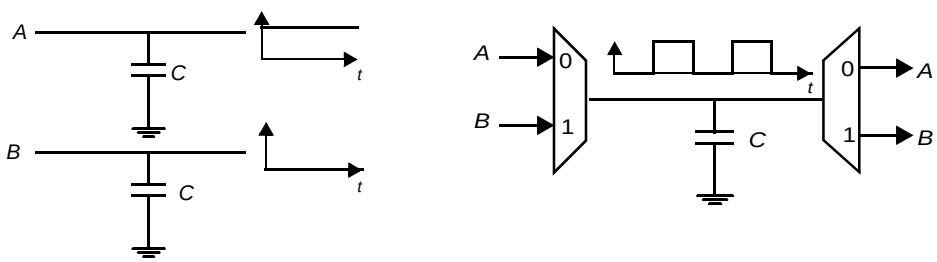


Figure 6.24 Reordering of inputs affects the circuit activity.

head—, it would seem that the degree of time-multiplexing should not affect the switched capacitance, since the time-multiplexed solution has half the capacitance switched at twice the frequency (for a fixed throughput).

If data being transmitted were random, it will make no difference which architecture is used. However if the data signals have some distinct properties (called temporal correlation), the power dissipation of the time-multiplexed solution can be significantly higher. Suppose, for instance, that A is always (or mostly) 1 and B is (mostly) 0. In the parallel solution, the switched capacitance is very low since there are very few transitions on the data bits. However, in the time-multiplexed solution, the bus toggles between 0 and 1. Care must be taken in digital systems to avoid time-multiplexing data streams with very distinct data characteristics.



(a) parallel data transmission (b) serial data transmission

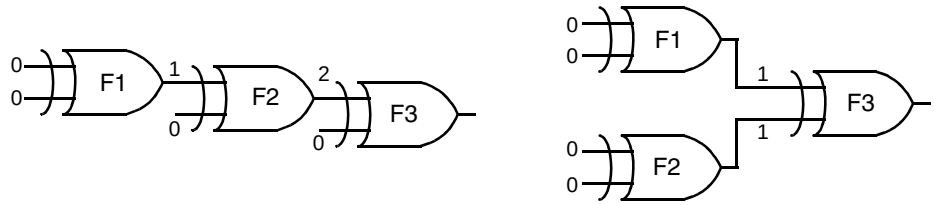
Figure 6.25 Parallel versus time-multiplexed data busses.

4. Glitch Reduction by balancing signal paths

The occurrence of glitching in a circuit is mainly due to a mismatch in the path lengths in the network. If all input signals of a gate change simultaneously, no glitching occurs. On the other hand, if input signals change at different times, a dynamic hazard might develop. Such a mismatch in signal timing is typically the result of different path lengths with respect to the primary inputs of the network. This is illustrated in Figure 6.26. Assume that the XOR gate has a unit delay. The first network (a) suffers from glitching as a result of the wide disparity between the arrival times of the input signals for a gate. For example, for gate F_3, one input settles at time 0, while the second one only arrives at time 2. Redesigning the network so that all arrival times are identical can dramatically reduce the number of superfluous transitions (network b).

Summary

The CMOS logic style described in the previous section is highly robust and scalable with



(a) Network sensitive to glitching (b) Glitch-free network

Figure 6.26 Glitching is influenced by matching of signal path lengths. The annotated numbers indicate the signal arrival times.

technology, but requires 2*N* transistors to implement a *N*-input logic gate. Also, the load capacitance is significant, since each gate drives two devices (a PMOS and an NMOS) per *fan-out*. This has opened the door for alternative logic families that either are simpler or faster.

6.2.2 Ratioed Logic

Concept

Ratioed logic is an attempt to reduce the number of transistors required to implement a given logic function, at the cost of reduced robustness and extra power dissipation. The purpose of the PUN in complementary CMOS is to provide a conditional path between V_{DD} and the output when the PDN is turned *off*. In ratioed logic, the entire PUN is replaced with a single unconditional load device that pulls up the output for a high output (Figure 6.27a). Instead of a combination of active pull-down and pull-up networks, such a gate consists of an NMOS pull-down network that realizes the *logic function*, and a simple *load device*. Figure 6.27b shows an example of ratioed logic, which uses a grounded PMOS load and is referred to as a pseudo-NMOS gate.

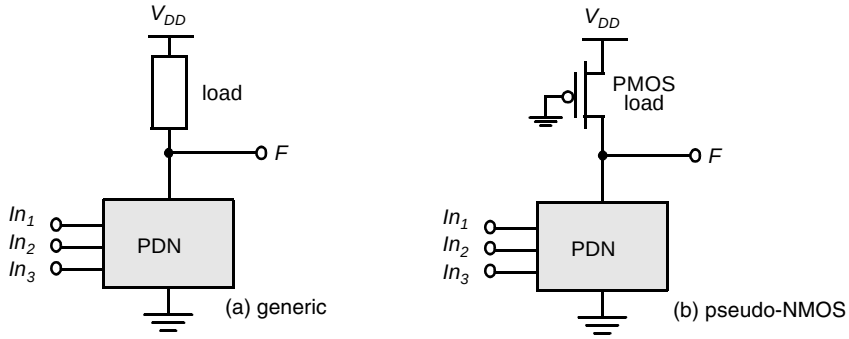


Figure 6.27 Ratioed logic gate.

The clear advantage of pseudo-NMOS is the reduced number of transistors (*N*+1 versus 2*N* for complementary CMOS). The nominal high output voltage (V_{OH}) for this gate is V_{DD} since the pull-down devices are turned *off* when the output is pulled high (assuming that V_{OL} is below V_{Tn}). On the other hand, the **nominal low output voltage is**

not 0 V since there is a fight between the devices in the PDN and the grounded PMOS load device. This results in reduced noise margins and more importantly static power dissipation. The sizing of the load device relative to the pull-down devices can be used to trade-off parameters such a *noise margin*, *propagation delay* and *power dissipation*. Since the voltage swing on the output and the overall functionality of the gate depends upon the ratio between the NMOS and PMOS sizes, the circuit is called *ratioed*. This is in contrast to the *ratioless* logic styles, such as complementary CMOS, where the low and high levels do not depend upon transistor sizes.

Computing the dc-transfer characteristic of the pseudo-NMOS proceeds along paths similar to those used for its complementary CMOS counterpart. The value of V_{OL} is obtained by equating the currents through the driver and load devices for $V_{in} = V_{DD}$. At this operation point, it is reasonable to assume that the NMOS device resides in linear mode (since the output should ideally be close to 0V), while the PMOS load is saturated.

$$k_n\left((V_{DD} - V_{Tn})V_{OL} - \frac{V_{OL}^2}{2}\right) = k_p\left((-V_{DD} - V_{Tp}) \cdot V_{DSAT} - \frac{V_{DSAT}^2}{2}\right) \tag{6.21}$$

Assuming that V_{OL} is small relative to the gate drive (V_{DD}-V_T) and that V_{Tn} is equal to V_{Tp} in magnitude, V_{OL} can be approximated as:

$$V_{OL} \approx \frac{k_p(-V_{DD} - V_{Tp}) \cdot V_{DSAT}}{k_n(V_{DD} - V_{Tn})} \approx \frac{\mu_p \cdot W_p}{\mu_n \cdot W_n} \cdot |V_{DSAT}| \tag{6.22}$$

In order to make V_{OL} as small as possible, the PMOS device should be sized much smaller than the NMOS pull-down devices. Unfortunately, this has a negative impact on the *propagation delay* for charging up the output node since the current provided by the PMOS device is limited.

A major disadvantage of the pseudo-NMOS gate is the static power that is dissipated when the output is low through the direct current path that exists between V_{DD} and *GND*. The static power consumption in the low-output mode is easily derived

$$P_{low} = V_{DD}I_{low} \approx V_{DD} \cdot k_p\left((-V_{DD} - V_{Tp}) \cdot V_{DSATp} - \frac{V_{DSATp}^2}{2}\right) \tag{6.23}$$

Example 6.7 Pseudo-NMOS Inverter

Consider a simple pseudo-NMOS inverter (where the PDN network in Figure 6.27 degenerates to a single transistor) with an NMOS size of 0.5μm/0.25μm. The effect of sizing the PMOS device is studied in this example to demonstrate the impact on various parameters. The *W*/*L* ratio of the grounded PMOS is varied over values from 4, 2, 1, 0.5 to 0.25. Devices with a *W*/*L* < 1 are constructed by making the length longer than the width. The voltage transfer curve for the different sizes is plotted in Figure 6.28.

Table 6.8 summarizes the nominal output voltage (V_{OL}), static power dissipation, and the low-to-high propagation delay. The low-to-high delay is measured as the time to reach 1.25V from V_{OL} (which is not 0V for this inverter). This is chosen since the load gate is a CMOS inverter with a switching threshold of 1.25V. The trade-off between the static and dynamic properties is apparent. A larger pull-up device improves performance, but increases static power dissipation and lowers noise margins (i.e., increases V_{OL}).

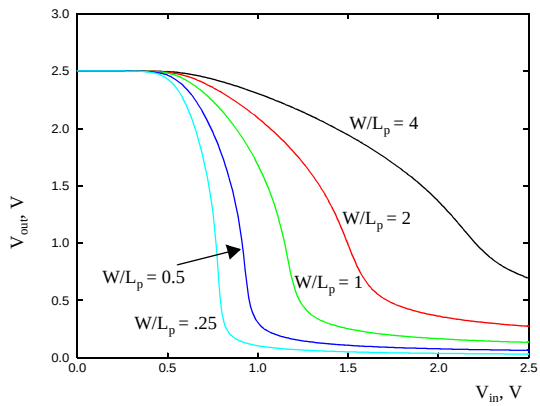


Figure 6.28 Voltage-transfer curves of the pseudo-NMOS inverter as a function of the PMOS size.

Table 6.8 Performance of a pseudo-NMOS inverter.

Size	V_{OL}	Static Power Dissipation	t_{plh}
4	0.693V	564µW	14ps
2	0.273V	298µW	56ps
1	.133V	160µW	123ps
0.5	0.064V	80µW	268ps
0.25	0.031V	41µW	569ps

Notice that the simple first-order model to predict V_{OL} is quite effective. For a PMOS W/L of 4, V_{OL} is given by (30/115) (4) (0.63V) = 0.66V.

The static power dissipation of pseudo-NMOS limits its use. However, pseudo-NMOS still finds use in large fan-in circuits. Figure 6.29 shows the schematics of pseudo-NMOS NOR and NAND gates. When area is most important, the reduced transistor count compared to complimentary CMOS is quite attractive.

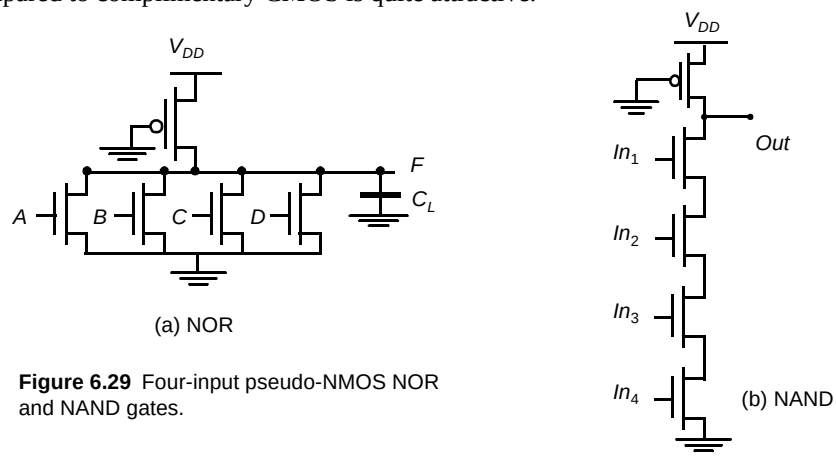


Figure 6.29 Four-input pseudo-NMOS NOR and NAND gates.

Problem 6.4 NAND Versus NOR in Pseudo-NMOS

Given the choice between NOR or NAND logic, which one would you prefer for implementation in pseudo-NMOS?

How to Build Even Better Loads

It is possible to create a ratioed logic style that completely eliminates static currents and provides rail-to-rail swing. Such a gate combines two concepts: *differential logic* and *positive feedback*. A differential gate requires that each input is provided in complementary format, and produces complementary outputs in turn. The feedback mechanism ensures that the load device is turned off when not needed. A example of such a logic family, called *Differential Cascode Voltage Switch Logic* (or DCVSL), is presented conceptually in Figure 6.30a [Heller84].

The pull-down networks PDN1 and PDN2 use NMOS devices and are mutually exclusive (this is, when PDN1 conducts, PDN2 is off, and when PDN1 is off, PDN2 conducts), such that the required logic function and its inverse are simultaneously implemented. Assume now that, for a given set of inputs, PDN1 conducts while PDN2 does not, and that *Out* and $\overline{Out}$ are initially high and low, respectively. Turning *on PDN1,* causes *Out* to be pulled down, although there is still a fight between M_1 and PDN1. $\overline{Out}$ is in a high impedance state, as M_2 and PDN2 are both turned *off*. PDN1 must be strong enough to bring *Out* below V_{DD}-$|V_{Tp}|$, the point at which M_2 turns *on* and starts charging $\overline{Out}$ to V_{DD}—eventually turning off M_1. This in turn enables *Out* to discharge all the way to *GND*. Figure 6.30b shows an example of an XOR/XNOR gate. Notice that it is possible to share transistors among the two pull-down networks, which reduces the implementation overhead

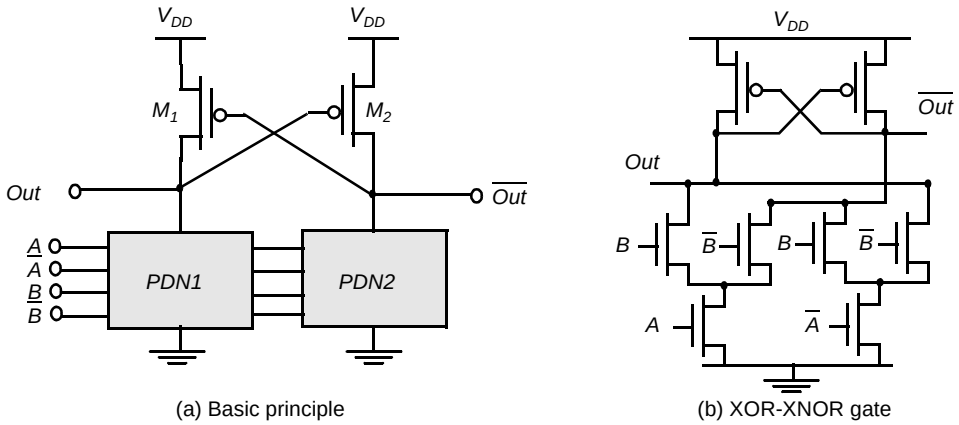


Figure 6.30 DCVSL logic gate.

The resulting circuit exhibits a rail-to-rail swing, and the static power dissipation is eliminated: in steady state, none of the stacked pull-down networks and load devices are simultaneously conducting. However, the circuit is still ratioed since the sizing of the PMOS devices relative to the pull-down devices is critical to functionality, not just perfor-

mance. In addition to the problem of increase complexity in design, this circuit style still has a power-dissipation problem that is due to cross-over currents. During the transition, there is a period of time when PMOS and PDN are turned on simultaneously, producing a short circuit path.

Example 6.8 DCVSL Transient Response

An example transient response is shown for an AND/NAND gate in DCVSL. Notice that as *Out* is pulled down to V_{DD}-$|V_{Tp}|$, $\overline{Out}$ starts to charge up to V_{DD} quickly. The delay from the input to *Out* is 197 psec and to $\overline{Out}$ is 321 psec. A static CMOS AND gate (NAND followed by an inverter) has a delay of 200ps.

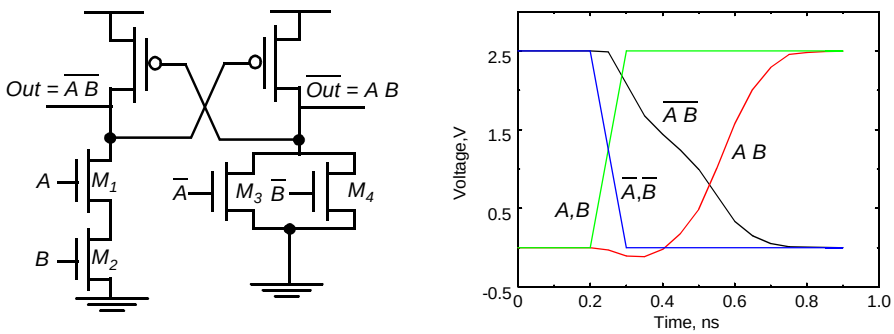


Figure 6.31 Transient response of a simple AND/NAND DCVSL gate. M_1 and M_2 1µm/0.25µm, M_3 and M_4 are 0.5µm/0.25µm and the cross-coupled PMOS devices are 1.5µm/0.25µm.

Design Consideration: Single-ended versus Differential

The DCVSL gate provides *differential (or complementary) outputs*. Both the output signal (V_{out1}) and its inverted value (V_{out2}) are simultaneously available. This is a distinct advantage, as it eliminates the need for an extra inverter to produce the complementary signal. It has been observed that a differential implementation of a complex function may reduce the number of gates required by a factor of two! The number of gates in the critical timing path is often reduced as well. Finally, the approach prevents some of the time-differential problems introduced by additional inverters. For example, in logic design it often happens that both a signal and its complement are needed simultaneously. When the complementary signal is generated using an inverter, the inverted signal is delayed with respect to the original (Figure 6.32a). This causes timing problems, especially in very high-speed designs. The differential output capability avoids this problem (Figure 6.32b).

With all these positive properties, why not always use differential logic? Well, the differential nature virtually doubles the number of wires that has to be routed, leading very often to unwieldy designs (on top of the additional implementation overhead in the individual gates). Additionally, the dynamic power dissipation is high.

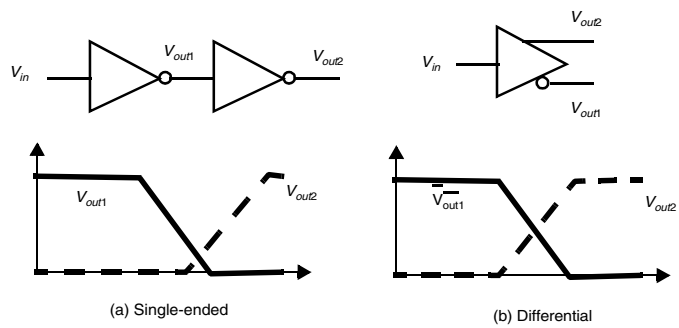


Figure 6.32 Advantage of differential (b) over single-ended (a) gate.

6.2.3 Pass-Transistor Logic

Pass-Transistor Basics

A popular and widely-used alternative to complementary CMOS is pass-transistor logic, which attempts to reduce the number of transistors required to implement logic by allowing the primary inputs to drive gate terminals as well as source/drain terminals [Radhakrishnan85]. This is in contrast to logic families that we have studied so far, which only allow primary inputs to drive the gate terminals of MOSFETS.

Figure 6.33 shows an implementation of the AND function constructed that way, using only NMOS transistors. In this gate, if the B input is high, the top transistor is turned on and copies the input A to the output F. When B is low, the bottom pass transistor is turned on and passes a 0. The switch driven by $\overline{B}$ seems to be redundant at first glance. Its presence is essential to ensure that the gate is static, this is that a low-impedance path exists to the supply rails under all circumstances, or, in this particular case, when B is low.

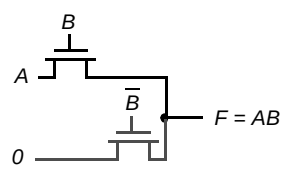


Figure 6.33 Pass-transistor implementation of an AND gate.

The promise of this approach is that fewer transistors are required to implement a given function. For example, the implementation of the AND gate in Figure 6.33 requires 4 transistors (including the inverter required to invert B), while a complementary CMOS implementation would require 6 transistors. The reduced number of devices has the additional advantage of lower capacitance.

Unfortunately, as discussed earlier, an NMOS device is effective at passing a 0 but is poor at pulling a node to V_{DD}. When the pass transistor pulls a node high, the output only charges up to V_{DD} -V_{Tn}. In fact, the situation is worsened by the fact that the devices experience body effect, as there exists a significant source-to-body voltage when pulling high. Consider the case when the pass transistor is charging up a node with the gate and drain terminals set at V_{DD}. Let the source of the NMOS pass transistor be labeled x. Node x will charge up to V_{DD}-$V_{Tn}(V_x)$:

$$V_x = V_{DD} - (V_{tn0} + \gamma((\sqrt{|2\phi_f| + V_x}) - \sqrt{|2\phi_f|})) \tag{6.24}$$

Example 6.9 Voltage swing for pass transistors circuits

Assuming a power supply voltage of 2.5V, the transient response of Figure 6.34 shows the output of a NMOS charging up (where the drain voltage is at V_{DD} and the gate voltage in is ramped from 0V to V_{DD}). Assume that node *x* was initially 0. Also notice that if *IN* is low,

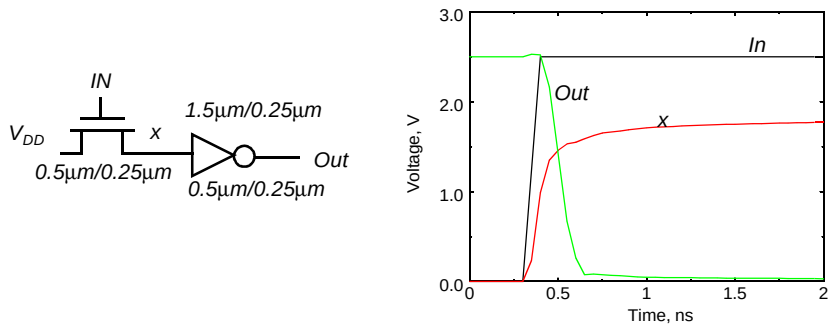


Figure 6.34 Transient response of charging up a node using an *N* device. Notice the slow tail after an initial quick response.

node *x* is in a high impedance state (not driven to one of the rails using a low resistance path). Extra transistors can be added to provide a path to GND, but for this discussion, the simplified circuit is sufficient. Notice that the output charges up quickly initially, but has slow tail. This is attributed to the fact that the drive (gate to source voltage) reduces significantly as the output approaches V_{DD}-V_{Tn} and the current available to charge up node *x* reduces drastically. Hand calculation using Eq. (6.24), results in an output voltage of 1.8V, which comes close to the simulated value.

WARNING:

The above example demonstrates that **pass-transistor gates cannot be cascaded by connecting the output of a pass gate to the gate input of another pass transistor.** This is illustrated in Figure 6.35a, where the output of M_1 (node *x*) drives the gate of another MOS device. Node *x* can charge up to V_{DD}-V_{Tn1}. If node *C* has a rail to rail swing, node *Y* only charges up to the voltage on node *x* - V_{Tn2}, which works out to V_{DD}-V_{Tn1}-V_{Tn2}. Figure 6.35b on the other hand has the output of M_1 (*x*) driving the junction of M_2, and there is only one threshold drop. This is the proper way of cascading pass gates.

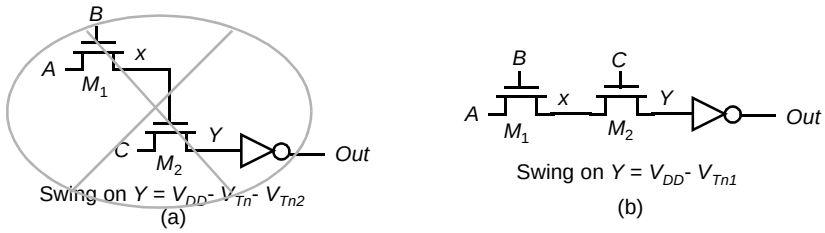


Figure 6.35 Pass transistor output (Drain/Source) terminal should not drive other gate terminals to avoid multiple threshold drops.

Example 6.10 VTC of the pass transistor AND gate

The voltage transfer curve of a pass-transistor gate shows little resemblance to complementary CMOS. Consider the AND gate shown in Figure 6.36. Similar to complementary CMOS, the VTC of pass transistor logic is data-dependent. For the case when $B = V_{DD}$, the top pass transistor is turned *on*, while the bottom one is turned *off*. In this case, the output just follows the input A until the input is high enough to turn *off* the top pass transistor (i.e., reaches V_{DD}-V_{Tn}). Next consider the case when A=V_{DD}, and B makes a transition from 0 → 1. Since the inverter has a threshold of $V_{DD}/2$, the bottom pass transistor is turned *on* till then and the output is close to zero. Once the bottom pass transistor turns *off*, the output follows the input B minus a threshold drop. A similar behavior is observed when both inputs A and B transition from 0 → 1.

Observe that a pure pass-transistor gate is not regenerative. A gradual signal degradation will be observed after passing through a number of subsequent stages. This can be remedied by the occasional insertion of a CMOS inverter. With the inclusion of an inverter in the signal path, the VTC resembles the one of CMOS gates.

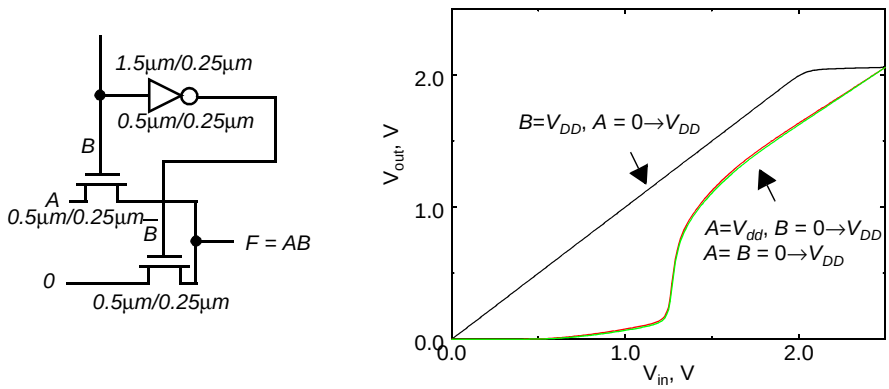


Figure 6.36 Voltage Transfer Characteristic for the pass-transistor AND gate of Figure 6.33.

Pass-transistors require lower switching energy to charge up a node due to the reduced voltage swing. For the pass transistor circuit in Figure 6.34 assume that the drain voltage is at V_{DD} and the gate voltage transitions to V_{DD}. The output node charges from 0V to V_{DD}-V_{Tn} (assuming that node x was initially at 0V) and the energy drawn from the power supply for charging the output of a pass transistor is given by:

$$E_{0\to 1} = \int_0^T P(t)dt = V_{DD}\int_0^T i_{supply}(t)dt = V_{DD}\int_0^{(V_{DD}-V_{Tn})} C_L dV_{out} = C_L \bullet V_{DD} \bullet (V_{DD}-V_{Tn}) \quad (6.25)$$

While the circuit exhibits lower switching power, it may consumes static power when the output is high—the reduced voltage level may be insufficient to turn off the PMOS transistor of the subsequent CMOS inverter.

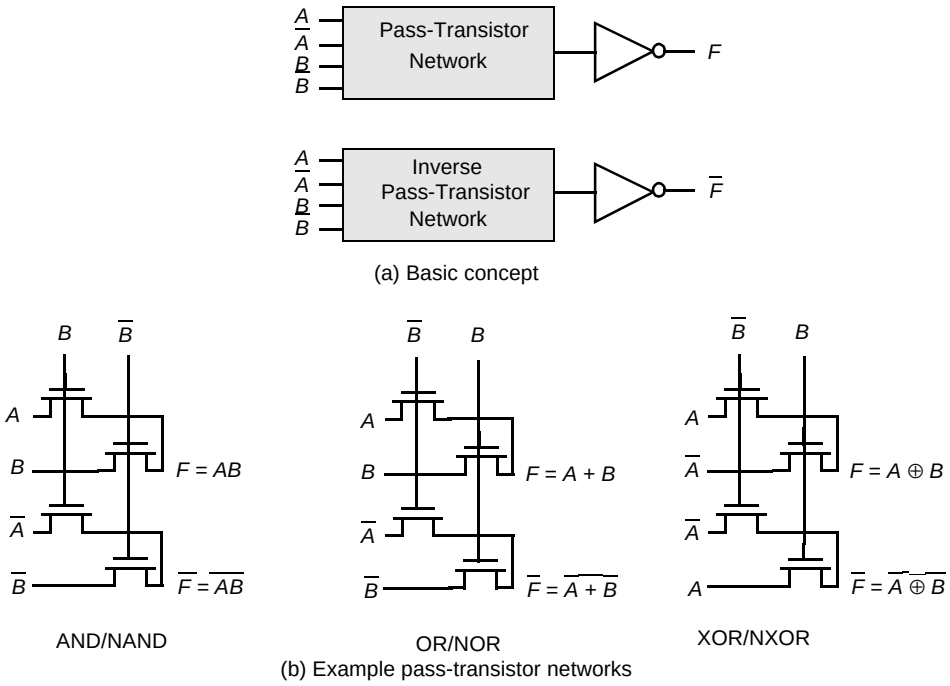


Figure 6.37 Complementary pass-transistor logic (CPL).

Differential Pass Transistor Logic

For high performance design, a differential pass-transistor logic family, called CPL or DPL, is commonly used. The basic idea (similar to DCVSL) is to accept true and complementary inputs and produce true and complementary outputs. A number of CPL gates (AND/NAND, OR/NOR, and XOR/NXOR) are shown in Figure 6.37. These gates possess a number of interesting properties:

- Since the circuits are *differential*, complementary data inputs and outputs are always available. Although generating the differential signals requires extra circuitry, the differential style has the advantage that some complex gates such as XORs and adders can be realized efficiently with a small number of transistors. Furthermore, the availability of both polarities of every signal eliminates the need for extra inverters, as is often the case in static CMOS or pseudo-NMOS.
- CPL belongs to the class of *static* gates, because the output-defining nodes are always connected to either V_{DD} or *GND* through a low resistance path. This is advantageous for the noise resilience.
- The design is very modular. In effect, all gates use exactly the same topology. Only the inputs are permutated. This makes the design of a library of gates very simple. More complex gates can be built by cascading the standard pass-transistor modules.

Example 6.11 Four-input NAND in CPL

Consider the implementation of a four-input AND/NAND gate using CPL. Based on the associativity of the boolean AND operation [$A{\cdot}B{\cdot}C{\cdot}D = (A{\cdot}B){\cdot}(C{\cdot}D)$], a two-stage approach has been adopted to implement the gate (Figure 6.38). The total number of transistors in the gate (including the final buffer) is 14. This is substantially higher than previously discussed gates. This factor, combined with the complicated routing requirements, makes this circuit style not particularly efficient for this gate. One should, however, be aware of the fact that the structure simultaneously implements the AND and the NAND functions, which might reduce the transistor count of the overall circuit.

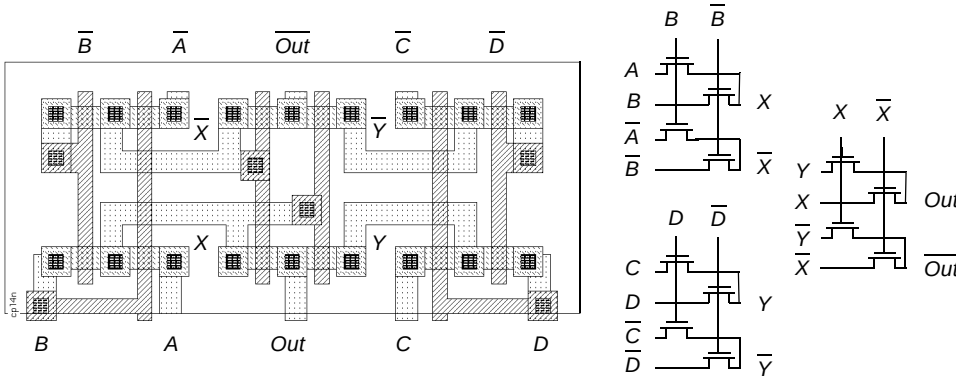


Figure 6.38 Layout and schematics of four-input NAND-gate using CPL (the final inverter stage is omitted). See also Colorplate 9.

In summary, CPL is a conceptually simple and modular logic style. Its applicability depends strongly upon the logic function to be implemented. The availability of a simple XOR as well of the ease of implementing some specific gate structures makes it attractive for structures such as adders and multipliers. Some extremely fast and efficient implementations have been reported in that application domain [Yano90]. When considering CPL, the designer should not ignore the implicit routing overhead of the complementary signals, which is apparent in the layout of Figure 6.38.

Robust and Efficient Pass-Transistor Design

Unfortunately, differential pass-transistor logic, like single-ended pass-transistor logic, suffers from static power dissipation and reduced noise margins, since the high input to the signal-restoring inverter only charges up to V_{DD}-V_{Tn}. There are several solutions proposed to deal with this problem as outlined below.

Solution 1: Level Restoration. A common solution to the voltage drop problem is the use of a *level restorer*, which is a single PMOS configured in a feedback path (Figure 6.39). The gate of the PMOS device is connected to the output of the inverter, its drain connected to the input of the inverter and the source to V_{DD}. Assume that node X is at 0V (*out* is at V_{DD} and the M_r is turned *off*) with $B = V_{DD}$ and $A = 0$. If input A makes a 0 to V_{DD} transition, M_n only charges up node X to V_{DD}-V_{Tn}. This is, however, enough to switch the

output of the inverter low, turning on the feedback device M_r and pulling node X all the way to V_{DD}. This eliminates any static power dissipation in the inverter. Furthermore, no static current path can exist through the level restorer and the pass-transistor, since the restorer is only active when A is high. In summary, this circuit has the advantage that all voltage levels are either at GND or V_{DD}, and no static power is consumed.

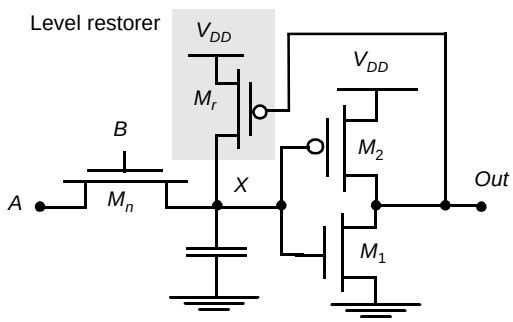


Figure 6.39 Level-restoring circuit.

While this solution is appealing in terms of eliminating static power dissipation, it adds complexity since the circuit is now ratioed. The problem arises during the transition of node X from high-to-low. The pass transistor network attempts to pull-down node X while the level restorer pulls now X to V_{DD}. Therefore, the pull-down device must be stronger than the pull-up device in order to switch node X and the output. Some careful transistor sizing is necessary to make the circuit function correctly. Assume the notation R_1 to denote the equivalent on-resistance of transistor M_1, R_2 for M_2, and so on. When R_r is made too small, it is impossible to bring the voltage at node X below the switching threshold of the inverter. Hence, the inverter output never switches to V_{DD}, and the level-restoring transistor stays on. This sizing problem can be reformulated as follows: the resistance of M_n and M_r must be such that the voltage at node X drops below the threshold of the inverter, $V_M = f(R_1, R_2)$. This condition is sufficient to guarantee a switching of the output voltage V_{out} to V_{DD} and a turning off of the level-restoring transistor.

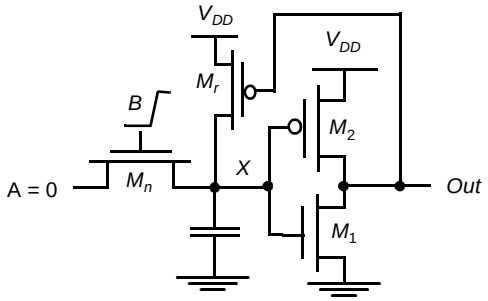


Figure 6.40 Transistor-sizing problem for level-restoring circuit.

Example 6.12 Sizing of a Level Restorer

Analyzing the circuit as a whole is nontrivial, because the restoring transistor acts as a feedback device. One way to simplify the circuit for manual analysis is to open the feedback loop and to ground the gate of the restoring transistor when determining the switching point (this is a reasonable assumption, as the feedback only becomes effective once the inverter starts to

switch). Hence, M_r and M_n form a "pseudo-NMOS-like" configuration, with M_r the load transistor and M_n acting as a pull-down device to *GND*. Assume that the inverter M_1, M_2 is sized to have the switching point at $V_{DD}/2$ (NMOS: 0.5µm/0.25µm and PMOS: 1.5µm/0.25µm). Therefore, node *X* must be pulled below $V_{DD}/2$ in order to switch the inverter and shut off M_r.

This is confirmed in Figure 6.42, which shows the transient response as the size of the level restorer is varied while keeping the size of M_n fixed (0.5µm/0.25µm). As the simulation indicates, for sizes above 1.5µm/0.25µm, node *X* can't be brought below the switching threshold of the inverter and can't switch the output. The detailed derivation of sizing requirement will be presented in the sequential design chapter. An important point to observe here is that the sizing of M_r is critical for DC functionality, not just performance!

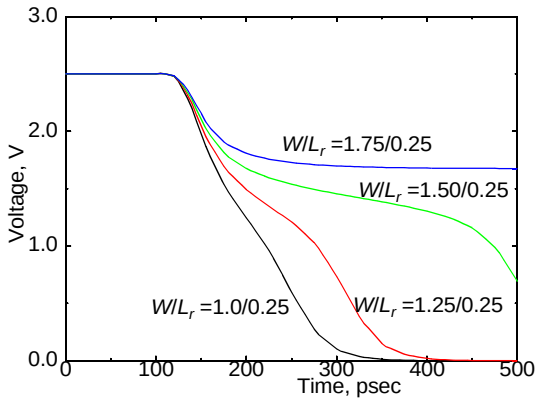


Figure 6.41 Transient response of the circuit in Figure 6.40. A level restorer that is too large can result in incorrect evaluation.

Another concern is the influence of the level restorer on the switching speed of the device. Adding the restoring device increases the capacitance at the internal node *X*, slowing down the gate. The rise time of the gate is further negatively affected, since, the level-restoring transistor M_r fights the decrease in voltage at node *X* before being switched off. On the other hand, the level restorer reduces the fall time, since the PMOS transistor, once turned on, speeds the pull-up action.

Problem 6.5 Device Sizing in Pass Transistors

For the circuit shown in Figure 6.40, assume that the pull-down device consists of 6 pass transistors in series each with a device size of 0.5µm/0.25µm (replacing transistor M_n). Determine the maximum *W*/*L* size for the level restorer transistor for correct functionality.

A modification of the level-restorer, applicable in differential networks and known as swing-restored pass transistor logic, is shown in Figure 6.42. Instead of a simple inverter or half-latch at the output of the pass transistor network, two back-to-back inverters, configured in a cross-coupled fashion, are used for level restoration and performance improvement. Inputs are fed to both the gate and source/drain terminals as in the case of conventional pass transistor networks. Figure 6.42 shows a simple XOR/XNOR gate of three variables *A*, *B* and *C*. Notice that the complementary network can be optimized by sharing transistors between the true and complementary outputs.

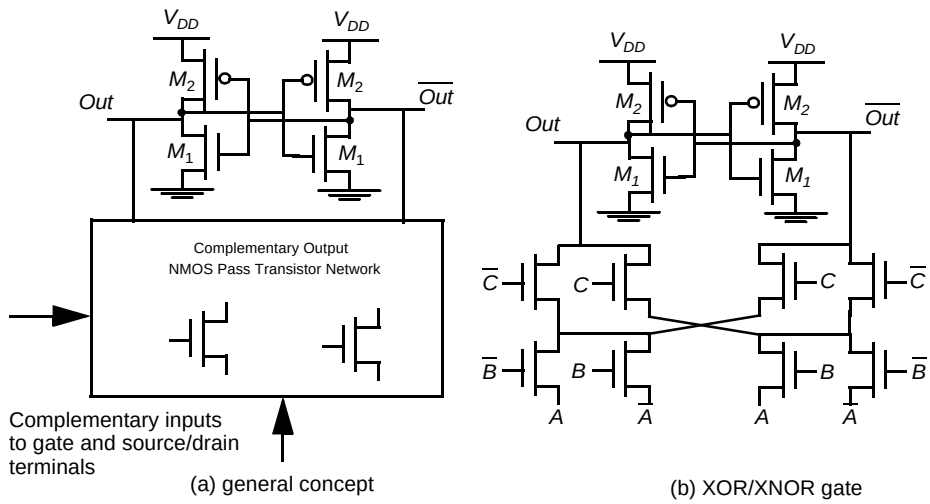


Figure 6.42 Swing-restored pass transistor logic [Parameswar96].

Solution 2: Multiple-Threshold Transistors. A technology solution to the voltage-drop problem associated with pass-transistor logic is the use of multiple-threshold devices. Using *zero threshold* devices for the NMOS pass-transistors eliminates most of the threshold drop, and passes a signal close to V_{DD}. Notice that even if the devices threshold was implanted to be exactly equal to zero, the body effect of the device prevents a swing to V_{DD}. All devices other than the pass transistors (i.e., the inverters) are implemented using standard high-threshold devices. The use of multiple-threshold transistors is becoming more common, and involves simple modifications to existing process flows.

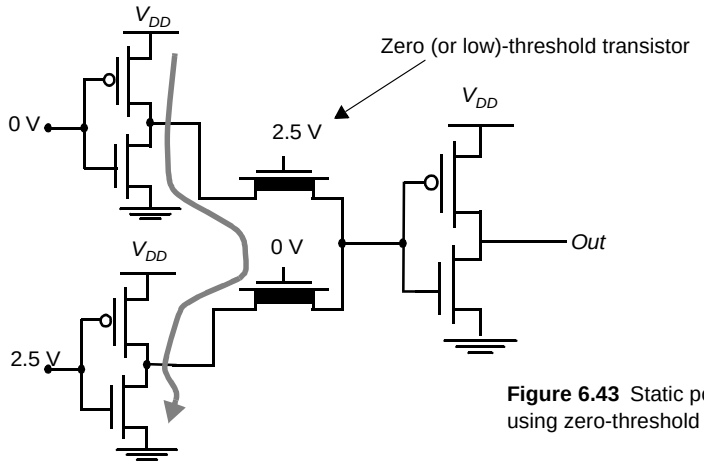


Figure 6.43 Static power consumption when using zero-threshold pass-transistors.

The use of zero-threshold transistors can be dangerous due to the subthreshold currents that can flow through the pass-transistors, even if V_{GS} is slightly below V_T. This is demonstrated in Figure 6.43, which points out a potential sneak dc-current path. While these leakage paths are not critical when the device is switching constantly, they do pose a significant energy-overhead when the circuit is in the idle state.

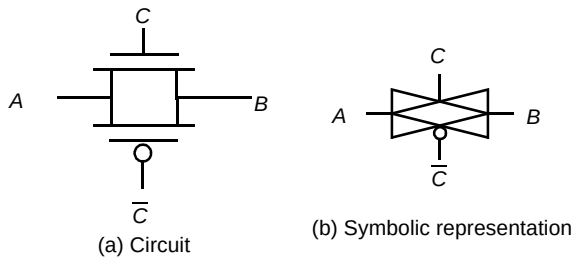


Figure 6.44 CMOS transmission gate.

Solution 3: Transmission Gate Logic. The most widely-used solution to deal with the voltage-drop problem is the use of *transmission gates*. It builds on the complementary properties of NMOS and PMOS transistors: NMOS devices pass a strong 0 but a weak 1, while PMOS transistors pass a strong 1 but a weak 0. The ideal approach is to use an NMOS to pull-down and a PMOS to pull-up. The transmission gate combines the best of both device flavors by placing a NMOS device in parallel with a PMOS device (Figure 6.44a). The control signals to the transmission gate (C and $\overline{C}$) are complementary. The transmission gate acts as a bidirectional switch controlled by the gate signal C. When C = 1, both MOSFETs are on, allowing the signal to pass through the gate. In short,

$$A = B \quad \text{if} \quad C = 1 \tag{6.26}$$

On the other hand, C = 0 places both transistors in cutoff, creating an open circuit between nodes A and B. Figure 6.44b shows a commonly used transmission-gate symbol.

Consider the case of charging node B to V_{DD} for the transmission gate circuit in Figure 6.45a. Node A is driven to V_{DD} and transmission gate is enabled (C = 1 and $\overline{C}$= 0). If only the NMOS pass-device were present, node B charges up to V_{DD}-V_{Tn} at which point the NMOS device turns off. However, since the PMOS device is present and turned on (V_{GSp} = -V_{DD}), charging continues all the way up to V_{DD}. Figure 6.45b shows the opposite case, this is discharging node B to 0. B is initially at V_{DD} when node A is driven low. The PMOS transistor by itself can only pull down node B to V_{Tp} at which point it turns *off*. The parallel NMOS device however stays turned on (since its V_{GS} = V_{DD}) and pulls node B all the way to *GND*. Though the transmission gate requires two transistors and more control signals, it enables rail-to-rail swing.

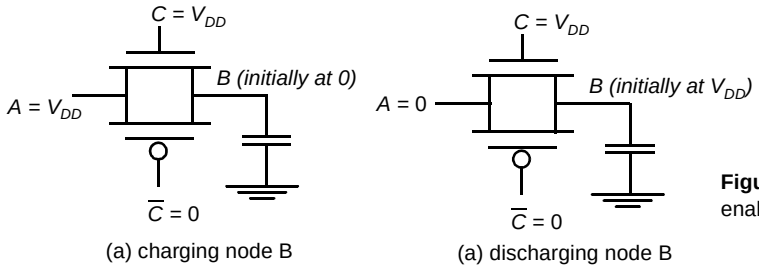


Figure 6.45 Transmission gates enable rail-to-rail switching

Transmission gates can be used to build some complex gates very efficiently. Figure 6.46 shows an example of a simple inverting two-input multiplexer. This gate either

selects input A or B based on the value of the control signal S, which is equivalent to implementing the following Boolean function:

$$\bar{F} = (A \cdot S + B \cdot \bar{S}) \tag{6.27}$$

A complementary implementation of the gate requires eight transistors instead of six.

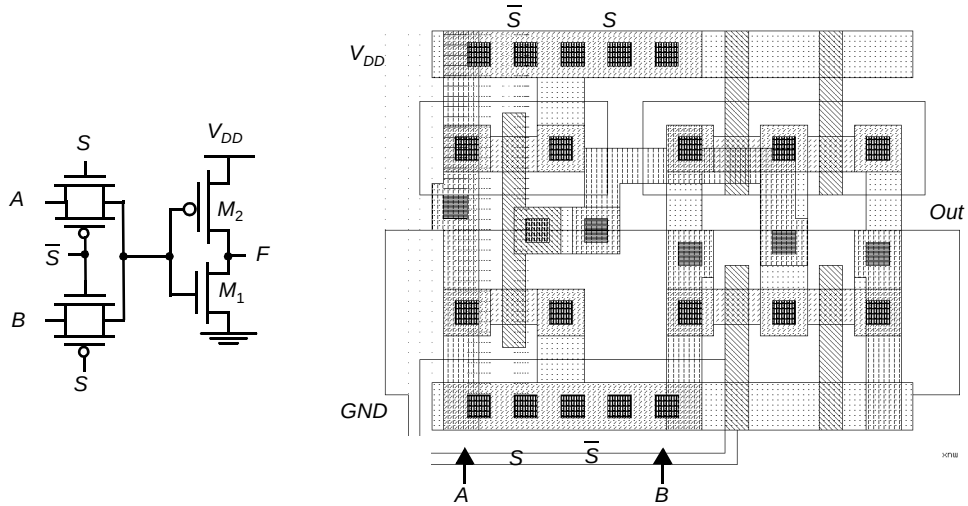


Figure 6.46 Transmission gate multiplexer and its layout.

Another example of the effective use of transmission gates is the popular XOR circuit shown in Figure 6.47. The complete implementation of this gate requires only six transistors (including the inverter used for the generation of $\bar{B}$), compared to the twelve transistors required for a complementary implementation. To understand the operation of this circuit, we have to analyze the $B = 0$ and $B = 1$ cases separately. For $B = 1$, transistors M_1 and M_2 act as an inverter while the transmission gate M_3/M_4 is off; hence $F = \bar{A}B$. In the opposite case, M_1 and M_2 are disabled, and the transmission gate is operational, or $F = A\bar{B}$. The combination of both results in the XOR function. Notice that, regardless of the values of A and B, node F always has a connection to either V_{DD} or GND and is hence a low-impedance node. When designing static-pass transistor networks, it is essential to adhere to the low-impedance rule under all circumstances. Other examples where transmission-gate logic is effectively used are fast adder circuits and registers.

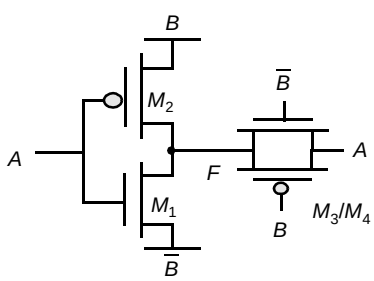


Figure 6.47 Transmission gate XOR.

Performance of Pass-Transistor and Transmission Gate Logic

The pass-transistor and the transmission gate are, unfortunately, not ideal switches, and have a series resistance associated with it. To quantify the resistance, consider the circuit in Figure 6.48, which involves charging a node from 0 V to V_{DD}. In this discussion, we use the large-signal definition of resistance, which involves dividing the voltage across the switch by the drain current. The effective resistance of the switch is modeled as a parallel connection of the resistances R_n and R_p of the NMOS and PMOS devices, defined as $(V_{DD} - V_{out})/I_n$ and $(V_{DD} - V_{out})/I_p$, respectively. The currents through the devices are obviously dependent on the value of V_{out} and the operating mode of the transistors. During the low-to-high transition, the pass-transistors traverse through a number of operation modes. For low values of V_{out}, the NMOS device is saturated and the resistance is approximated as:

$$R_n = \frac{V_{DD} - V_{out}}{I_N} = \frac{V_{DD} - V_{out}}{k'_n\left(\frac{W}{L}\right)_N\left((V_{DD} - V_{out} - V_{Tn})V_{DSAT} - \frac{V_{DSAT}^2}{2}\right)} \approx \frac{V_{DD} - V_{out}}{k_n(V_{DD} - V_{out} - V_{Tn})V_{DSAT}} \tag{6.28}$$

The resistance goes up for increasing values of V_{out}, and approaches infinity when V_{out} reaches V_{DD}-V_{Tn}, this is when the device shuts off. Similarly, we can analyze the behavior of the PMOS transistor. When V_{out} is small, the PMOS is saturated, but it enters the linear mode of operation for V_{out} approaching V_{DD}, giving the following approximated resistance:

$$R_p = \frac{V_{DD} - V_{out}}{I_P} = \frac{V_{DD} - V_{out}}{k_p \cdot \left((-V_{DD} - V_{Tp})(V_{out} - V_{DD}) - \frac{(V_{out} - V_{DD})^2}{2}\right)} \approx \frac{1}{k_p(V_{DD} - |V_{Tp}|)} \tag{6.29}$$

The simulated value of $R_{eq} = R_p \parallel R_n$ as a function of V_{out} is plotted in Figure 6.48. It can be observed that R_{eq} is relatively constant (≈ 8kΩ in this particular case). The same is true in other design instances (for instance, when discharging C_L). When analyzing transmission-gate networks, the simplifying assumption that the switch has a constant resistive value is therefore acceptable.

Problem 6.6 Equivalent Resistance During Discharge

Determine the equivalent resistance by simulation for the high-to-low transition of a transmission gate (this is, produce a plot similar to the one presented in Figure 6.48).

An important consideration is the delay associated with a chain of transmission gates. Figure 6.49 shows a chain of *n* transmission gates. Such a configuration often occurs in circuits such as adders or deep multiplexors. Assume that all transmission gates are turned on and a step is applied at the input. To analyze the propagation delay of this

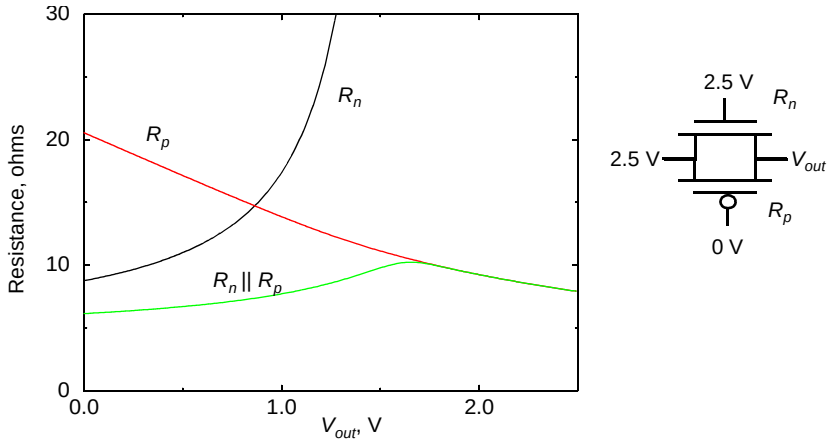


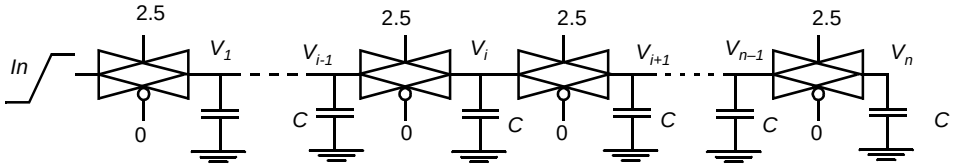
Figure 6.48 Simulated equivalent resistance of transmission gate for low-to-high transition (for $(W/L)_n = (W/L)_p = 0.5\mu m/0.25\mu m$). A similar response for overall resistance is obtained for the high-to-low transition

network, the transmission gates are replaced by their equivalent resistances R_{eq}. This produces the network of Figure 6.49b.

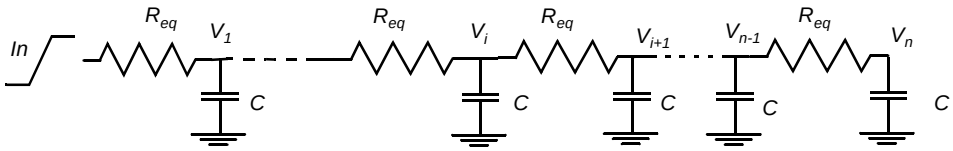
The delay of a network of n transmission gates in sequence can be estimated using the Elmore approximation (see Chapter 4):

$$t_p(V_n) = 0.69 \sum_{k=0}^{n} CR_{eq}k = 0.69CR_{eq}\frac{n(n+1)}{2} \tag{6.30}$$

This means that the propagation delay is proportional to n^2 and increases rapidly with the number of switches in the chain.



(a) A chain of transmission gates



(b) Equivalent *RC* network

Figure 6.49 Speed optimization in transmission-gate networks.

Example 6.13 Delay through 16 transmission gates

Consider 16 cascaded minimum-sized transmission gates, each with an average resistance of 8 kΩ. The node capacitance consists of the capacitance of two NMOS and PMOS devices (junction and gate). Since the gate inputs are assumed to be fixed, there is no Miller multiplication. The capacitance can be calculated to be approximately 3.6 fF for the low-to-high transition. The delay is given by:

$$t_p = 0.69 \cdot CR_{eq}\frac{n(n+1)}{2} = 0.69 \cdot (3.6fF)(8K\Omega)\left(\frac{16(16+1)}{2}\right) \approx 2.7ns \tag{6.31}$$

The transient response for this particular example is shown in Figure 6.50. The simulated delay is 2.7ns. It is remarkable that a simple RC model predicts the delay so accurately. It is also clear that the use of long pass transistor chains causes significant delay degradation.

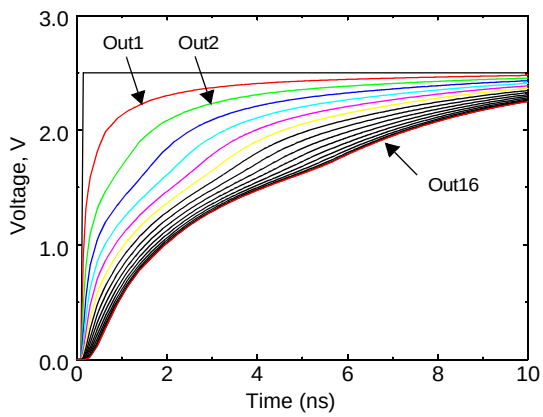


Figure 6.50 Transient response of 16 transmission gates cascaded in series.

The most common approach for dealing with the long delay is to break the chain and by inserting buffers every m switches (Figure 6.51). Assuming a propagation delay t_{buf} for each buffer, the overall propagation delay of the transmission-gate/buffer network is then computed as follows,

$$\begin{aligned} t_p &= 0.69\left[\frac{n}{m}CR_{eq}\frac{m(m+1)}{2}\right] + \left(\frac{n}{m} - 1\right)t_{buf} \\ &= 0.69\left[CR_{eq}\frac{n(m+1)}{2}\right] + \left(\frac{n}{m} - 1\right)t_{buf} \end{aligned} \tag{6.32}$$

The resulting delay exhibits only a linear dependence on the number of switches n, in contrast to the unbuffered circuit, which is quadratic in n. The optimal number of switches m_{opt} between buffers can be found by setting the derivative $\frac{\partial t_p}{\partial m}$ to 0, which yields

$$m_{opt} = 1.7\sqrt{\frac{t_{pbuf}}{CR_{eq}}} \tag{6.33}$$

Obviously, the number of switches per segment grows with increasing values of t_{buf}. In current technologies, m_{opt} is typically around 3.

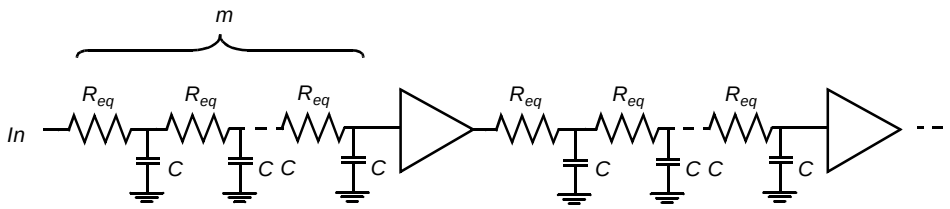


Figure 6.51 Breaking up long transmission gate chains by inserting buffers.

Example 6.14 Transmission Gate Chain

Consider the same 16 transmission gate chain. The buffers shown in Figure 6.51 can be implemented as inverters (instead of two cascaded inverters). In some cases, it might be necessary to add an extra inverter to produce the correct polarity. Assuming that each inverter is sized such that the NMOS is 0.5µm/0.25µm and PMOS is 0.5µm /0.25µm, Eq. (6.33) predicts that an inverter must be inserted every 3 transmission gates. The simulated delay when placing an inverter every two transmission gates equals 154 psec, for every three transmission gates is 154 psec, and for four transmission gates is 164 psec. The insertion of buffering inverters reduces the delay with a factor of almost 2.

CAUTION: Although many of the circuit styles discussed in the previous sections sound very exciting, and might be superior to static CMOS in many respects, none of them has the *robustness and ease of design* of complementary CMOS. Therefore, use them sparingly and with caution. For designs that have no extreme area, complexity, or speed constraints, complementary CMOS is the recommended design style.

6.3 Dynamic CMOS Design

It was noted earlier that static CMOS logic with a fan-in of N requires $2N$ devices. A variety of approaches were presented to reduce the number of transistors required to implement a given logic function including pseudo-NMOS, pass transistor logic, etc. The pseudo-NMOS logic style requires only $N + 1$ transistors to implement an N input logic gate, but unfortunately it has static power dissipation. In this section, an alternate logic style called *dynamic logic* is presented that obtains a similar result, while avoiding static power consumption. With the addition of a clock input, it uses a sequence of *precharge* and conditional *evaluation* phases.

6.3.1 Dynamic Logic: Basic Principles

The basic construction of an (n-type) dynamic logic gate is shown in Figure 6.52a. The PDN (pull-down network) is constructed exactly as in complementary CMOS. The opera-

tion of this circuit is divided into two major phases: *precharge* and *evaluation*, with the mode of operation determined by the *clock signal CLK*.

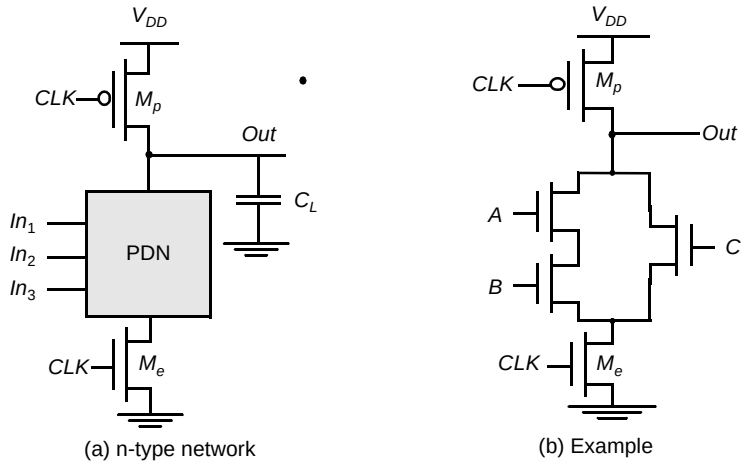


Figure 6.52 Basic concepts of a dynamic gate.

Precharge

When $CLK = 0$, the output node *Out* is precharged to V_{DD} by the PMOS transistor M_p. During that time, the evaluate NMOS transistor M_e is off, so that the pull-down path is disabled. The evaluation FET eliminates any static power that would be consumed during the precharge period (this is, static current would flow between the supplies if both the pull-down and the precharge device were turned on simultaneously).

Evaluation

For $CLK = 1$, the precharge transistor M_p is off, and the evaluation transistor M_e is turned on. The output is conditionally discharged based on the input values and the pull-down topology. If the inputs are such that the PDN conducts, then a low resistance path exists between *Out* and *GND* and the output is discharged to *GND*. If the PDN is turned off, the precharged value remains stored on the output capacitance C_L, which is a combination of junction capacitances, the wiring capacitance, and the input capacitance of the fan-out gates. During the evaluation phase, the only possible path between the output node and a supply rail is to *GND*. Consequently, once *Out* is discharged, it cannot be charged again till then next precharge operation. The inputs to the gate can therefore make *at most one transition during evaluation.* Notice that the output can be in the *high-impedance state* during the evaluation period if the pull-down network is turned off. This behavior is fundamentally different from the static counterpart that always has a low resistance path between the output and one of the power rails.

As as an example, consider the circuit shown in Figure 6.52b. During the precharge phase (CLK=0), the output is precharged to V_{DD} regardless of the input values since the evaluation device is turned off. During evaluation (CLK=1), a conducting path is created

between *Out* and *GND* if (and only if) $A \cdot B + C$ is TRUE. Otherwise, the output remains at the precharged state of V_{DD}. The following function is thus realized:

$$Out = \overline{CLK} + \overline{(A \cdot B + C)} \cdot CLK \tag{6.34}$$

A number of important properties can be derived for the dynamic logic gate:

- The logic function is implemented by the NMOS pull-down network. The construction of the PDN proceeds just as it does for static CMOS.
- The *number of transistors* (for complex gates) is substantially lower than in the static case: $N + 2$ versus $2N$.
- It is *non-ratioed*. The sizing of the PMOS precharge device is not important for realizing proper functionality of the gate. The size of the precharge device can be made large to improve the low-to-high transition time (of course, at a cost to the high-to-low transition time). There is however, a trade-off with power dissipation since a larger precharge device directly increases clock-power dissipation.
- It only consumes *dynamic power*. Ideally, no static current path ever exists between V_{DD} and *GND*. The overall power dissipation, however, can be significantly higher compared to a static logic gate.
- The logic gates have *faster switching speeds.* There are two main reasons for this. The first (obvious) reason is due to the reduced load capacitance attributed to the lower number of transistors per gate and the single-transistor load per *fan-in*. Second, the dynamic gate does not have short circuit current, and all the current provided by the pull-down devices goes towards discharging the load capacitance.

The low and high output levels V_{OL} and V_{OH} are easily identified as *GND* and V_{DD} and are not dependent upon the transistor sizes. The other VTC parameters are dramatically different from static gates. Noise margins and switching thresholds have been defined as static quantities that are not a function of time. To be functional, a dynamic gate requires a periodic sequence of precharges and evaluations. Pure static analysis, therefore, does not apply. During the evaluate period, the pull-down network of a dynamic inverter starts to conduct when the input signal exceeds the threshold voltage (V_{Tn}) of the NMOS pull-down transistor. Therefore, it is reasonable to set the switching threshold (V_M) as well as V_{IH} and V_{IL} of the gate equal to V_{Tn}. This translates to a low value for the NM_L.

Design Consideration

It is also possible to implement dynamic logic using a complimentary approach, where the output node is connected by a pre-discharge NMOS transistor to GND, and the evaluation PUN network is implemented in PMOS. The operation is similar: during precharge, the output node is discharged to GND. During evaluation, the output is conditionally charged to VDD. This p-type dynamic gate has the disadvantage of being slower than the n-type due to the lower current drive of the PMOS transistors.

6.3.2 Speed and Power Dissipation of Dynamic Logic

The main advantages of dynamic logic are increased speed and reduced implementation area. Fewer devices to implement a given logic function implies that the overall load capacitance is much smaller. The analysis of the switching behavior of the gate has some interesting peculiarities to it. After the precharge phase, the output is high. For a low input signal, no additional switching occurs. As a result, $t_{pLH} = 0$! The high-to-low transition, on the other hand, requires the discharging of the output capacitance through the pull-down network. Therefore t_{pHL} is proportional to C_L and the current-sinking capabilities of the pull-down network. The presence of the evaluation transistor slows the gate somewhat, as it presents an extra series resistance. Omitting this transistor, while functionally not forbidden, may result in static power dissipation and potentially a performance loss.

The above analysis is somewhat unfair, because it ignores the influence of the precharge time on the switching speed of the gate. The precharge time is determined by the time it takes to charge C_L through the PMOS precharge transistor. During this time, the logic in the gate cannot be utilized. However, very often, the overall digital system can be designed in such a way that the precharge time coincides with other system functions. For instance, the precharge of the arithmetic unit in a microprocessor can coincide with the instruction decode. The designer has to be aware of this "dead zone" in the use of dynamic logic, and should carefully consider the pros and cons of its usage, taking the overall system requirements into account.

Example 6.15 A Four-Input Dynamic NAND Gate

Figure 6.53 shows the design of a four-input NAND example designed using the dynamic-circuit style. Due to the dynamic nature of the gate, the derivation of the voltage-transfer characteristic diverges from the traditional approach. As we had discussed above, we will assume that the switching threshold of the gate equals the threshold of the NMOS pull-down transistor. This results in asymmetrical noise margins, as shown in Table 6.9.

The dynamic behavior of the gate is simulated with SPICE. It is assumed that all inputs are set high as the clock transitions high. On the rising edge of the clock, the output node is discharged. The resulting transient response is plotted in Figure 6.53, and the propagation delays are summarized in Table 6.9. The duration of the precharge cycle can be adjusted by changing the size of the PMOS precharge transistor. Making the PMOS too large should be avoided, however, as it both slows down the gate, and increases the capacitive load on the clock line. For large designs, the latter factor might become a major design concern as the clock load can become excessive and hard to drive.

Table 6.9 The dc and ac parameters of a four-input dynamic NAND.

Transistors	V_{OH}	V_{OL}	V_M	NM_H	NM_L	t_{pHL}	t_{pLH}	t_{pre}
6	2.5 V	0 V	V_{TN}	2.5-V_{TN}	V_{TN}	110 psec	0 nsec	83psec

As mentioned earlier, the static parameters are time-dependent. To illustrate this, consider the four input NAND gate with all inputs tied together, and making a partial low-to-high transition. Figure 6.54 shows a transient simulation of the output voltage for three different

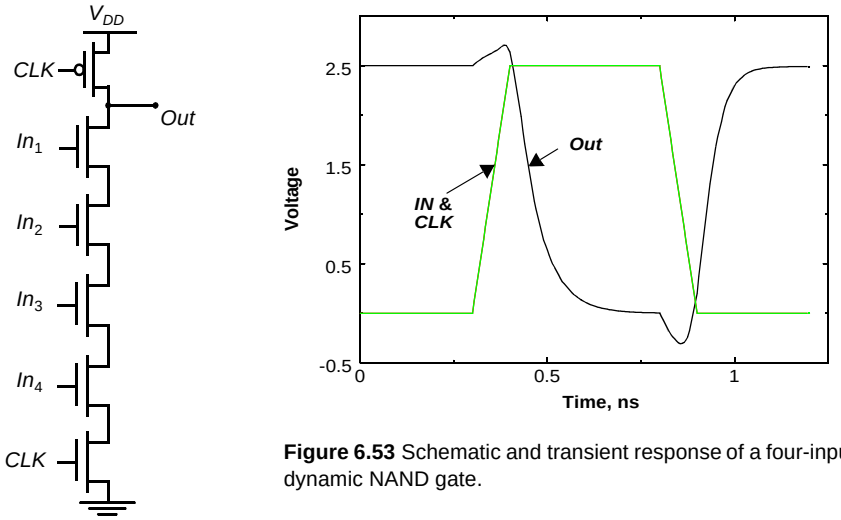


Figure 6.53 Schematic and transient response of a four-input dynamic NAND gate.

input transitions—to 0.45V, 0.5V and 0.55V, respectively. Above, we have defined the switching threshold of the dynamic gate as the device threshold. However, notice that the amount by which the output voltage drops is a strong function of the input voltage and the available evaluation time. The noise voltage needed to corrupt the signal has to be larger if the evaluation time is short. In other words, the switching threshold is really a function of the evaluation time.

When evaluating the power dissipation of a dynamic gate, it would appear that dynamic logic presents a significant advantage. There are three reasons for this. First, the physical capacitance is lower since dynamic logic uses fewer transistors to implement a given function. Also, the load seen for each fanout is one transistor instead of two. Second, dynamic logic gates *by construction* can at most have one transition per clock cycle. Glitching (or dynamic hazards) does not occur in dynamic logic. Finally, dynamic gates do not exhibit short circuit power since the pull-up path is not turned on when the gate is evaluating.

While these arguments are generally true, they are offset by other considerations: (i) the clock power of dynamic logic can be significant, particularly since the clock node has

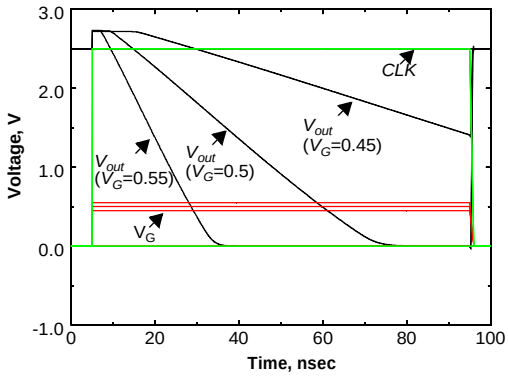


Figure 6.54 Effect of an input glitch on the output. The switching threshold depends on the time for evaluation. A larger glitch is acceptable if the evaluation phase is smaller. In this example, the input glitches high during evaluation and stays high during the whole period.

a guaranteed transition on every single clock cycle; (ii) the number of transistors is higher than the minimal set required for implementing the logic; (iii) short-circuit power may exist when leakage-combatting devices are added (as will be discussed further); (iv) and, most importantly, dynamic logic generally displays a higher switching activity due to the periodic *precharge* and *discharge* operations. Earlier, the transition probability for a static gate was shown to be $p_0\, p_1 = p_0\,(1\text{-}p_0)$. For dynamic logic, the output transition probability does not depend on the state (history) of the inputs, but rather on the signal probabilities only. For an *n*-tree dynamic gate, the output makes a 0→1 transition during the precharge phase only if the output was discharged during the preceding evaluate phase. The 0→1 transition probability for an *n*-type dynamic gate hence equals

$$a_{0 \to 1} = p_0 \tag{6.35}$$

where p_0 is the probability that the output is zero. This number is always larger or equal to $p_0\, p_1$. For uniformly distributed inputs, the transition probability for an *N*-input gate is:

$$a_{0 \to 1} = \frac{N_0}{2^N} \tag{6.36}$$

where N_0 is the number of zero entries in the truth table of the logic function.

Example 6.16 Activity estimation in dynamic logic

To illustrate the increased activity for a dynamic gate, once again consider a 2 input NOR gate. An n-tree dynamic implementation is shown in Figure 6.55 along with its static counterpart. For equi-probable inputs, there is then a 75% probability that the output node of the dynamic gate will discharge immediately after the precharge phase, implying that the activity for such a gate equals 0.75 (i.e P_{NOR}= 0.75 $C_L V_{dd}^2 f_{clk}$). The corresponding activity is a lot smaller, 3/16, for a static implementation. For a dynamic NAND gate, the transition probability is 1/4 (since there is a 25% probability the output will be discharged) while it is 3/16 for a static implementation. Though these example illustrate that the switching activity of dynamic logic is generally higher, it should be noted that dynamic logic has lower physical capacitance. Both factors must be accounted for when choosing a logic style.

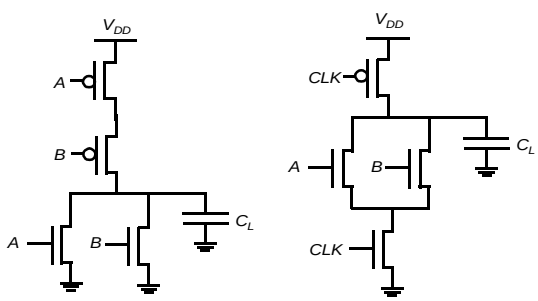


Figure 6.55 Static NOR versus n-type dynamic NOR.

Problem 6.7 Activity Computation

For the 4-input dynamic NAND gate, compute the activity factor with the following assumption for the inputs. Assume that the inputs are independent and $p_{A=1} = 0.2$, $p_{B=1} = 0.3$, $p_{C=1} = 0.5$, and $p_{D=1} = 0.4$.

6.3.3 Issues in Dynamic Design

Dynamic logic clearly can result in high performance solutions compared to static circuits. However, there are several important considerations that must be taken into account if one wants dynamic circuits to function properly. This include charge leakage, charge sharing, backgate (and in general capacitive) coupling, and clock feedthrough. Some of these issues are highlighted in this section.

Charge Leakage

The operation of a dynamic gate relies on the dynamic storage of the output value on a capacitor. If the pull-down network is *off*, the output should ideally remain at the precharged state of V_{DD} during the evaluation phase. However, this charge gradually leaks away due to leakage currents, eventually resulting in a malfunctioning of the gate. Figure 6.56a shows the sources of leakage for the basic dynamic inverter circuit.

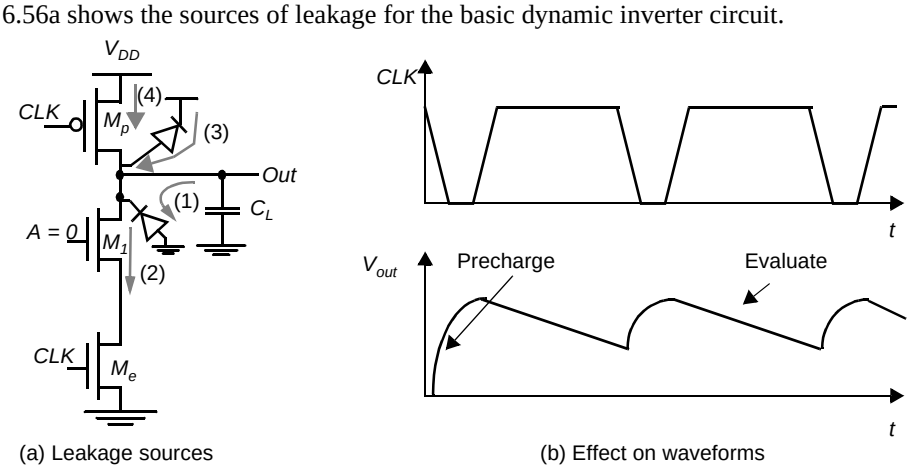


Figure 6.56 Leakage issues in dynamic circuits.

Source 1 and 2 are the *reverse-biased diode* and *sub-threshold leakage* of the NMOS pull-down device M_1, respectively. The charge stored on C_L will slowly leak away due these leakage sources, assuming that the input is at zero during evaluation. Charge leakage causes a degradation in the high level (Figure 6.56b). Dynamic circuits therefore require a minimal clock rate, which is typically on the order of a few kHz. This makes the usage of dynamic techniques unattractive for low performance products such as watches, or processors that use conditional clocks (where there are no guarantees on minimum clock rates). Note that the PMOS precharge device also contributes some leakage current

due to the reverse bias diode (source 3) and the subthreshold conduction (source 4). To some extent, the leakage current of the PMOS counteracts the leakage of the pull-down path. As a result the output voltage is going to be set by the resistive divider composed of the pull-down and pull-up paths.

Example 6.17 Leakage in dynamic circuits

Consider the simple inverter with all devices set at 0.5µm/0.25µm. Assume that the input is low during the evaluation period. Ideally, the output should remain at the precharged state of V_{DD}. However, as seen from Figure 6.57 the output voltage drops. Once the output drops below the switching threshold of the fan-out logic gate, the output is interpreted as a low voltage. Notice that the output settles to an intermediate voltage. This is due to the leakage current provided by the PMOS pull-up.

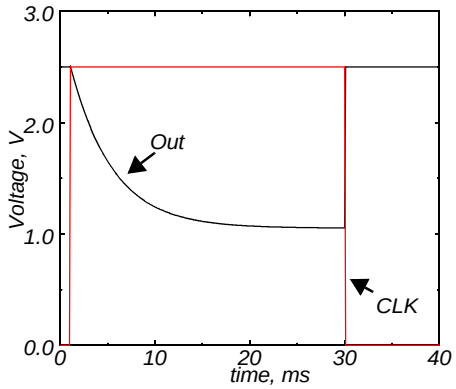


Figure 6.57 Impact of charge leakage. The output settles to an intermediate voltage determined by a resistive divider of the pull-down and pull up devices.

Leakage is caused by the high impedance state of the output node during the evaluate mode, when the pull down path is turned off. The leakage problem van be counteracted by reducing the output impedance on the output node during evaluation. This is often done by adding a *bleeder transistor* as shown in Figure 6.58a. The only function of the bleeder—a pseudo-NMOS-like pull-up device—is to compensate for the charge lost due to the pull-down leakage paths. To avoid the ratio problems associated with this style of circuit and the associated static power consumption, the bleeder resistance is made high, or, in other words, the device is kept small. This allows the (strong) pull-down devices to

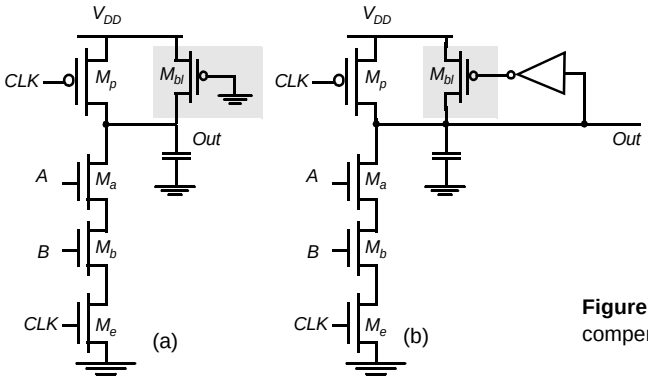


Figure 6.58 Static bleeders compensates for the charge-leakage.

lower the *Out* node substantially below the switching threshold of the inverter. Often, the bleeder is implemented in a feedback configuration to eliminate the static power dissipation (Figure 6.58b).

Charge Sharing

Another important concern in dynamic logic is the impact of charge sharing. Consider the circuit of Figure 6.59. During the precharge phase, the output node is precharged to V_{DD}. Assume that all inputs are set to 0 during precharge, and that the capacitance C_a is discharged. Assume further that input B remains at 0 during evaluation, while input A makes a $0 \rightarrow 1$ transition, turning transistor M_a *on*. The charge stored originally on capacitor C_L is redistributed over C_L and C_a. This causes a drop in the output voltage, which cannot be recovered due to the dynamic nature of the circuit.

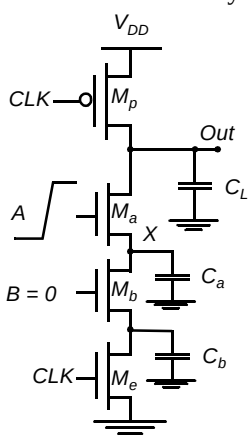


Figure 6.59 Charge sharing in dynamic networks.

The influence on the output voltage is readily calculated. Under the above assumptions, the following initial conditions are valid: $V_{out}(t = 0) = V_{DD}$ and $V_X(t = 0) = 0$. Two possible scenarios must be considered:

1. $\Delta V_{out} < V_{Tn}$ — In this case, the final value of V_X equals $V_{DD} - V_{Tn}(V_X)$. Charge conservation yields

$$C_L V_{DD} = C_L V_{out}(t) + C_a[V_{DD} - V_{Tn}(V_X)]$$

or

$$\Delta V_{out} = V_{out}(t) - V_{DD} = -\frac{C_a}{C_L}[V_{DD} - V_{Tn}(V_X)] \tag{6.37}$$

2. $\Delta V_{out} > V_{Tn}$ — V_{out} and V_X reach the same value:

$$\Delta V_{out} = -V_{DD}\left(\frac{C_a}{C_a + C_L}\right) \tag{6.38}$$

Which of the above scenarios is valid is determined by the capacitance ratio. The boundary condition between the two cases can be determined by setting ΔV_{out} equal to V_{Tn} in Eq. (6.38), yielding

$$\frac{C_a}{C_L} = \frac{V_{Tn}}{V_{DD} - V_{Tn}} \tag{6.39}$$

Overall, it is desirable to keep the value of ΔV_{out} below $|V_{Tp}|$. The output of the dynamic gate might be connected to a static inverter, in which case the low level of V_{out} would cause static power consumption. One major concern is circuit malfunction if the output voltage is brought below the switching threshold of the gate it drives.

Example 6.18 Charge-Sharing

Let us consider the impact of charge sharing on the dynamic logic gate shown in Figure 6.60, which implements a 3-input EXOR function $y = A \oplus B \oplus C$. The first question to be resolved is what conditions cause the worst-case voltage drop on node y. For simplicity, ignore the load inverter, and assume that all inputs are low during the precharge operation and that all isolated internal nodes (V_a, V_b, V_c, and V_d) are initially at 0V.

Inspection of the truth table for this particular logic function shows that the output stays high for 4 out of 8 cases. The worst-case change in output is obtained by exposing the maximum amount of internal capacitance to the output node during the evaluation period. This happens for $\overline{A}\ B\ C$ or $A\ \overline{B}\ C$. The voltage change can then be obtained by equating the initial charge with the final charge as done with equation Eq. (6.38), yielding a worst-case change of 30/(30+50) * 2.5V = 0.94V. To ensure that the circuit functions correctly, the switching threshold of the connecting inverter should be placed below 2.5- 0.94 = 1.56V.

The most common and effective approach to deal with the charge redistribution is to also precharge critical internal nodes, as is shown in Figure 6.61. Since the internal nodes are charged to V_{DD} during precharge, charge sharing does not occur. This solution obviously comes at the cost of increased area and capacitance.

Capacitive Coupling

The high impedance of the output node makes the circuit very sensitive to crosstalk effects. A wire routed over a dynamic node may couple capacitively and destroy the state of the floating node. Another equally important form of capacitive coupling is the *back-gate (or output-to-input) coupling.*Consider the circuit shown in Figure 6.62 in which a dynamic two-input NAND gate drives a static NAND gate. A transition in the input *In* of

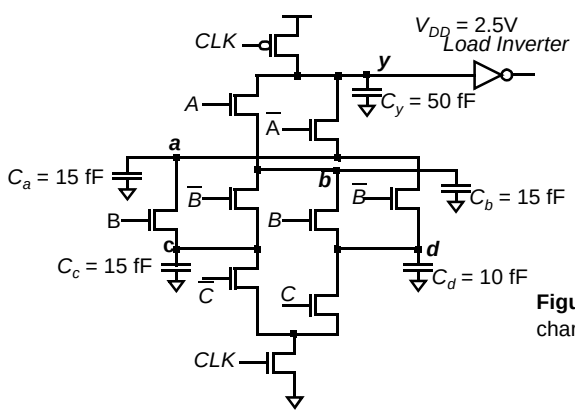


Figure 6.60 Example illustrating the charge sharing effect in dynamic logic.

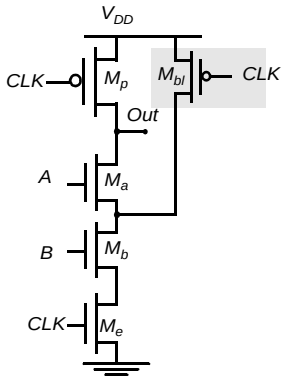


Figure 6.61 Dealing with charge-sharing by precharging internal nodes. An NMOS precharge transistor may also be used, but this requires an inverted clock.

the static gate may cause the output of the gate (Out_2) to go low. This output transition couples capacitively to the other input of the gate, the dynamic node Out_1, through the gate-source and gate-drain capacitances of transistor M4. A simulation of this effect is shown in Figure 6.63, and demonstrates that the output of the dynamic gate can drop significantly. This further causes the output of the static NAND gate not to drop all the way down to 0V, and a small amount of static power is dissipated. If the voltage drop is large enough, the circuit can evaluate incorrectly, and the NAND output may not go low. When designing and laying out dynamic circuits, special care is needed to minimize capacitive coupling.

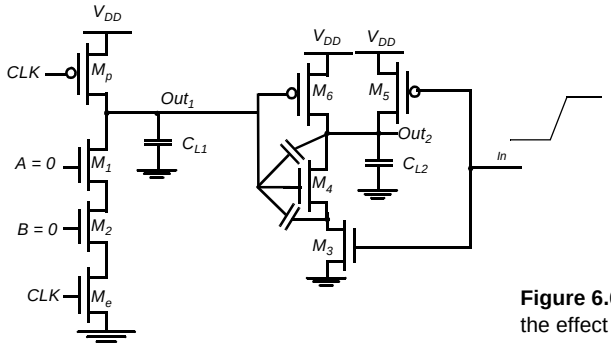


Figure 6.62 Example demonstrating the effect of backgate coupling.

Clock-Feedthrough

A special case of capacitive coupling is clock-feedthrough, an effect caused by the capacitive coupling between the clock input of the precharge device and the dynamic output node. The coupling capacitance consists of the gate-to-drain capacitance of the precharge device, and includes both the overlap and the channel capacitances. This capacitive coupling causes the output of the dynamic node to rise above V_{DD} on the low-to-high transition of the clock, assuming that the pull-down network is turned off. Subsequently, the fast rising and falling edges of the clock couple onto the signal node, as is quite apparent in the simulation of Figure 6.63.

The danger of clock feedthrough is that it may cause the (normally reverse-biased) junction diodes of the precharge transistor to become forward-biased. This causes electron

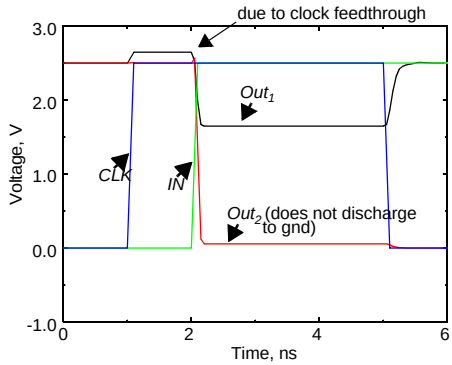


Figure 6.63 Clock feedthrough effect.

injection into the substrate, which can be collected by a nearby high impedance node in the **1** state, eventually resulting in faulty operation. CMOS latchup might be another result of this injection. For all purposes, high-speed dynamic circuits should be carefully simulated to ensure that clock-feedthrough effects stay within bounds.

All the above considerations demonstrate that the design of dynamic circuits is rather tricky and requires extreme care. It should therefore only be attempted when high performance is required.

6.3.4 Cascading Dynamic Gates

Besides the signal integrity issues, there is one major catch that complicates the design of dynamic circuits: straightforward cascading of dynamic gates to create more complex structures does not work. The problem is best illustrated with the two cascaded *n*-type dynamic inverters, shown in Figure 6.64a. During the precharge phase (i.e., *CLK* =0), the outputs of both inverters are precharged to V_{DD}. Assume that the primary input *In* makes a $0 \rightarrow 1$ transition (Figure 6.64b). On the rising edge of the clock, output Out_1 starts to discharge. The second output should remain in the precharged state of V_{DD} as its expected

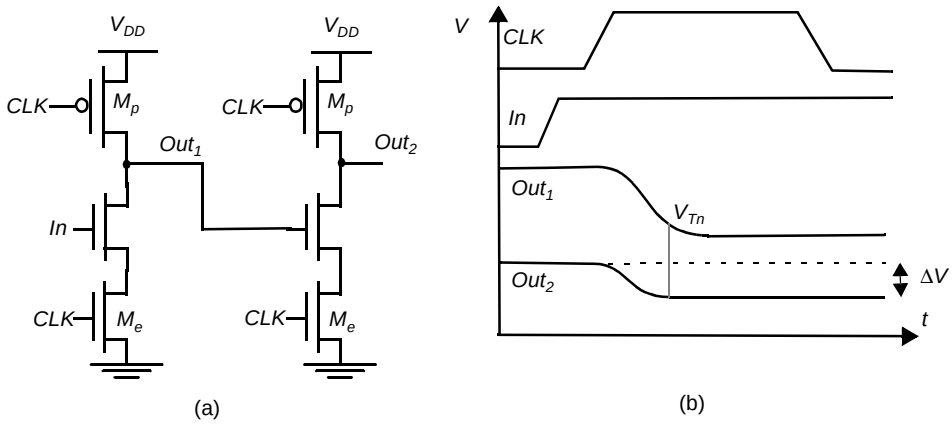


Figure 6.64 Cascade of dynamic *n*–type blocks.

value is 1 (Out_1 transitions to 0 during evaluation). However, there is a finite propagation delay for the input to discharge Out_1 to GND. Therefore, the second output also starts to discharge. As long as Out_1 exceeds the switching threshold of the second gate, which approximately equals V_{Tn}, a conducting path exists between Out_2 and *GND*, and precious charge is lost at Out_2. The conducting path is only disabled once Out_1 reaches V_{Tn}, and turns off the NMOS pull-down transistor. This leaves Out_2 at an intermediate voltage level. The correct level will not be recovered, as dynamic gates rely on capacitive storage in contrast to static gates, which have dc restoration. The charge loss leads to reduced noise margins and potential malfunctioning.

The cascading problem arises because the outputs of each gate—and hence the inputs to the next stages—are precharged to 1. This may cause inadvertent discharge in the beginning of the evaluation cycle. Setting all the inputs to 0 during precharge addresses that concern. When doing so, all transistors in the pull-down network are turned off after precharge, and no inadvertent discharging of the storage capacitors can occur during evaluation. In other words, correct operation is guaranteed as long as **the inputs can only make a single 0 → 1 transition during the evaluation period**[2]. Transistors are only be turned on when needed, and at most once per cycle. A number of design styles complying with this rule have been conceived. The two most important ones are discussed below.

Domino Logic

Concept. A Domino logic module [Krambeck82] consists of an n-type dynamic logic block followed by a static inverter (Figure 6.65). During precharge, the output of the *n*-type dynamic gate is charged up to V_{DD}, and the output of the inverter is set to 0. During evaluation, the dynamic gate conditionally discharges, and the output of the inverter makes a conditional transition from 0 → 1. If one assumes that all the inputs of a Domino gate are outputs of other Domino gates[3], then it is ensured that all inputs are set to 0 at the end of the precharge phase, and that the only transitions during evaluation are 0 → 1 transitions. The formulated rule is hence obeyed. The introduction of the static inverter has the additional advantage that the fan-out of the gate is driven by a static inverter with a low-impedance output, which increases noise immunity. The buffer furthermore reduces the capacitance of the dynamic output node by separating internal and load capacitances.

Consider now the operation of a chain of Domino gates. During precharge, all inputs are set to 0. During evaluation, the output of the first Domino block either stays at 0 or makes a 0 → 1 transition, affecting the second gate. This effect might ripple through the whole chain, one after the other, similar to a line of falling dominoes—hence the name. Domino CMOS has the following properties:

- Since each dynamic gate has a static inverter, only non-inverting logic can be implemented. Although there are ways to deal with this, as is discussed in a subsequent section, this is major limiting factor, and pure Domino design has become rare.

[2] This ignores the impact of charge distribution and leakage effects, discussed earlier.

[3] It is required that all other inputs that do not fall under this classification (for instance, primary inputs) stay constant during evaluation.

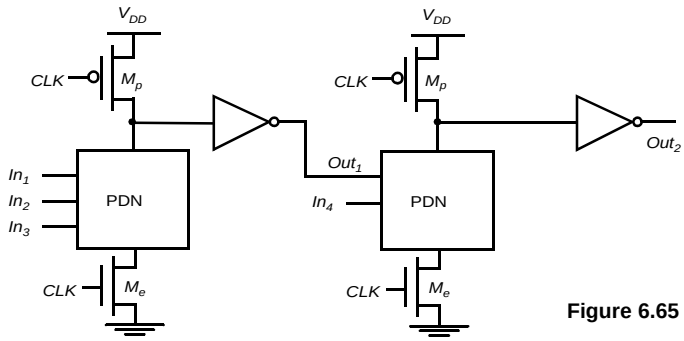


Figure 6.65 DOMINO CMOS logic.

- Very high speeds can be achieved: only a rising edge delay exists, while t_{pHL} equals zero. The inverter can be sized to match the *fan-out*, which is already much smaller than in the complimentary static CMOS case, as only a single gate capacitance has to be accounted for per fan-out gate.

Since the inputs to a Domino gate are low during precharge, it is tempting to eliminate the evaluation transistor as this would reduce clock load and increase pull-down drive. However, eliminating the evaluation device extends the precharge cycle: the precharge now has to ripple through the logic network as well. Consider the logic network shown in Figure 6.66, where the evaluation devices have been eliminated. If the primary input In_1 is 1 during evaluation, the output of each dynamic gate is 0 and the output of each static inverter is 1. On the falling edge of the clock, the precharge operation is started. Assume further that In_1 makes a high-to-low transition. The input to the second gate is initially high, and it takes two gate delays before In_2 is driven low. During that time, the second gate cannot precharge its output, as the pull-down network is fighting the precharge device. Similarly, the third gate has to wait till the second gate precharges before it can start precharging, etc. Therefore the time taken to precharge the logic circuit is equal to its critical path. Another important negative is the extra power dissipation when both pull-up and pull-down devices are on. It is therefore good practice to always utilize evaluation devices.

Dealing with the Non-inverting Property of Domino Logic. A major limitation in Domino logic is that only non-inverting logic can be implemented. This requirement has

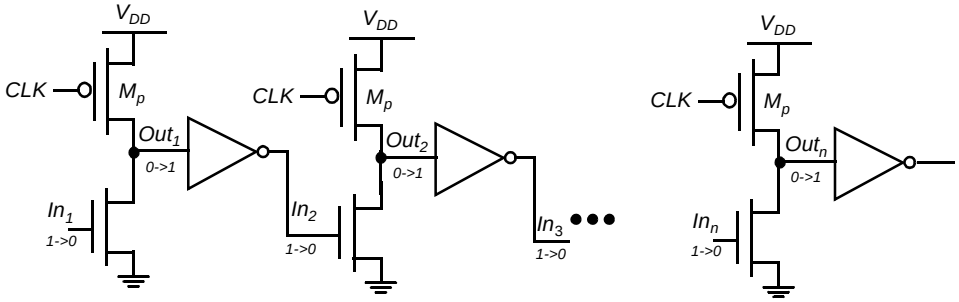


Figure 6.66 Effect of ripple precharge when the evaluation transistor is removed. The circuit also exhibits static power dissipation.

limited the widespread use of pure Domino logic. There are several ways to deal with the non-inverting logic requirement. Figure 6.67 shows one approach to the problem—reorganizing the logic using simple boolean transforms such as De Morgan's Law. Unfortunately, this sort of optimization is not always possible, and more general schemes may have to be used.

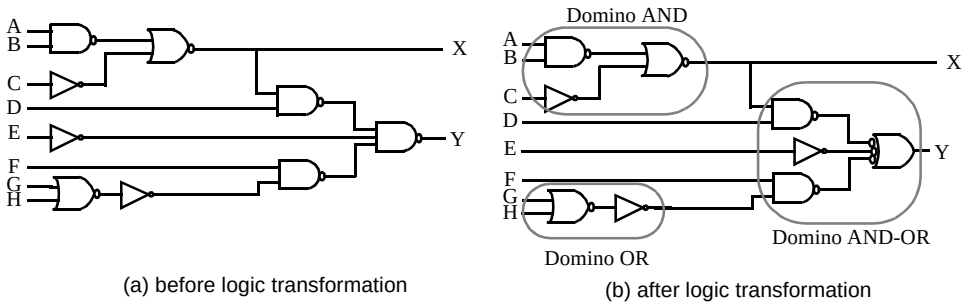


Figure 6.67 Restructuring logic to enable implementation using non-inverting Domino Logic.

A general but expensive approach to solving the problem is the use of differential logic. *Dual-rail Domino* is similar in concept to the DCVSL structure discussed earlier, but uses a precharged load instead of a static cross-coupled PMOS load. Figure 6.68 shows the circuit schematic of an AND/NAND differential logic gate. Note that all inputs come from other differential Domino gates, and are low during the precharge phase, while making a conditional 0→1 transition during evaluation. Using differential Domino, it is possible to implement any arbitrary function. This comes at the expense of an increased power dissipation, since a transition is guaranteed every single clock cycle regardless of the input values—either O or $\overline{O}$ must make a 0→1 transition. The function of transistors M_{f1} and M_{f2} is to keep the circuit static when the clock is high for extended periods of time (*bleeder*). Notice that this circuit is not ratioed, even in the presence of the PMOS pull-up devices! Due to its high-performance, this differential approach is very popular, and is used in several commercial microprocessors.

Optimization of Domino Logic Gates. Several optimizations can be performed on Domino logic gates. The most obvious performance optimization involves the sizing of

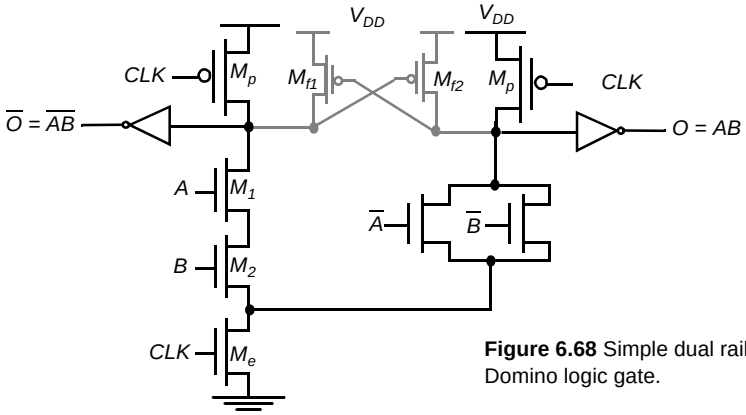


Figure 6.68 Simple dual rail (differential) Domino logic gate.

the transistors in the static inverter. With the inclusion of the evaluation devices in Domino circuits, all gates precharge in parallel, and the precharge operation takes only two gate delays—charging the output of the dynamic gate to V_{DD}, and driving the inverter output low. The critical path during evaluation goes through the pull-down path of the dynamic gate, and the PMOS pull-up transistor of the static inverter. Therefore, to speed up the circuit during evaluation, the beta ratio of the static inverter should be made high so that its switching threshold is close to V_{DD}. This can be accomplished by using a small (minimum) sized NMOS and a large PMOS device. The minimum-sized NMOS only impacts the precharge time, which is limited in general due to the parallel precharging of all gates. The only disadvantage of using a large beta ratio is a reduction in noise margin. A designer should therefor simultaneously consider the reduced noise margin and performance during the device sizing.

Numerous variations of Domino logic have been proposed [Bernstein98]. One optimization that reduces area is *Multiple Output Domino Logic*. The basic concept is illustrated is Figure 6.69. It exploits the fact that certain outputs are subsets of other outputs to generate a number of logical functions in a single gate. In this example, $O3 = C+D$ is used in all three outputs, and hence it is implemented at the bottom of the pull-down network. Since $O2$ equals $B \cdot O3$, it can reuse the logic for $O3$. Notice that the internal nodes have to be precharged to V_{DD} to produce the correct results. Given that the internal nodes precharge to V_{DD}, the number of devices driving precharge devices is not reduced. However, the number of evaluation transistors is drastically reduced as they are amortized over multiple outputs. Additionally, this approach results in a reduction of the fan-out factor, once again due to the reuse of transistors over multiple functions.

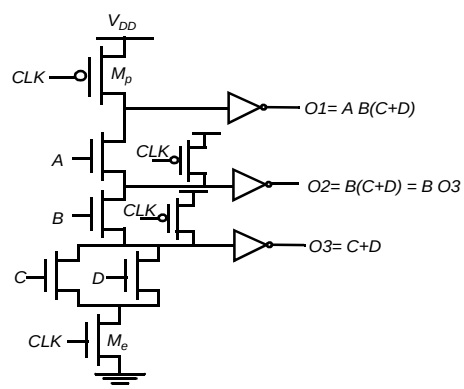


Figure 6.69 Multiple output Domino

Compound Domino (Figure 6.70) represents another optimization of the generic Domino gate, once again minimizing the number of transistors. Instead of each dynamic gate driving a static inverter, it is possible to combine the outputs of multiple dynamic gates with the aid of a complex static CMOS gate, as shown in Figure 6.70. The outputs of three dynamic structures, implementing $O1 = \overline{A\ B\ C}$, $O2 = \overline{D\ E\ F}$ and $O3 = \overline{G\ H}$, are combined using a single complex CMOS static gate that implements $O = \overline{(o1+o2)\ o3}$. The total logic function realized this way equals $O = A\ B\ C\ D\ E\ F + GH$.

Compound Domino is a useful tool for constructing complex dynamic logic gates. Large dynamic stacks are replaced using parallel small fan-in structures and complex CMOS gates. For example, a large *fan-in* Domino AND can be implemented as parallel dynamic NAND structures with lower *fan-in* that are combined using a static NOR gate. One important consideration in Compound Domino is the problem associated with back-

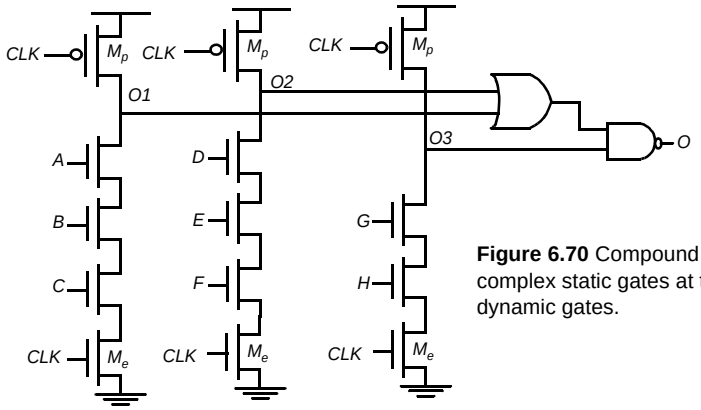


Figure 6.70 Compound Domino logic uses complex static gates at the output of the dynamic gates.

gate coupling. Care must be taken to ensure that the dynamic nodes are not affected by the coupling between the output of the static gates and the output of dynamic nodes.

np-CMOS

The Domino logic presented in the previous section has the disadvantage that each dynamic gate requires an extra static inverter in the critical path to make the circuit functional. *np*-CMOS, provides an alternate approach to cascading dynamic logic by using two flavors (*n*-tree and *p*-tree) of dynamic logic. In a *p*-tree logic gate, PMOS devices are used to build a pull-up logic network, including a PMOS evaluation transistor (Figure 6.71) ([Goncalvez83, Friedman84, Lee86]). The NMOS predischarge transistor drives the output low during precharge. The output conditionally makes a 0 → 1 transition during evaluation depending on its inputs.

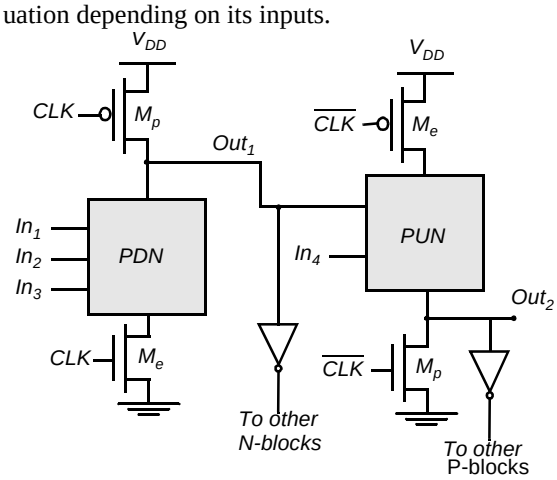


Figure 6.71 The *np*-CMOS logic circuit style.

np-CMOS logic exploits the duality between *n*-tree and *p*-tree logic gates to eliminate the cascading problem. If the *n*-tree gates are controlled by *CLK*, and *p*-tree gates are controlled using $\overline{CLK}$, *n*-tree gates can directly drive *p*-tree gates, and vice-versa. Similar to Domino, *n*-tree outputs must go through an inverter when connecting to another *n*-tree gate. During the precharge phase (*CLK* = 0), the output of the *n*-tree gate, *Out1*, is charged

to V_{DD}, while the output of the *p*-tree gate, *Out2*, is pre-discharged to 0V. Since the *n*-tree gate connects PMOS pull-up devices, the PUN of the *p*-tree is turned off at that time. During evaluation, the output of the n-tree gate can only make a 1→0 transition, conditionally turning on some transistors in the *p*-tree. This ensures that no accidental discharge of Out_2 can occur. Similarly, *n*-tree blocks can follow *p*-tree gates without any problems, as the inputs to the *n*-gate are precharged to 0. A disadvantage of the *np*-CMOS logic style is that the *p*-tree blocks are slower than the *n*-tree modules, due to the lower current drive of the PMOS transistors in the logic network. Equalizing the propagation delays requires extra area.

6.4 Perspectives

6.4.1 How to Choose a Logic Style?

In the preceding sections, we have discussed several gate-implementation approaches using the CMOS technology. Each of the circuit styles has its advantages and disadvantages. Which one to select depends upon the primary requirement: ease of design, robustness, area, speed, or power dissipation. No single style optimizes all these measures at the same time. Even more, the approach of choice may vary from logic function to logic function.

The static approach has the advantage of being robust in the presence of noise. This makes the design process rather trouble-free and amenable to a high degree of automation. This ease-of-design does not come for free: for complex gates with a large fan-in, complementary CMOS becomes expensive in terms of area and performance. Alternative static logic styles have therefore been devised. Pseudo-NMOS is simple and fast at the expense of a reduced noise margin and static power dissipation. Pass-transistor logic is attractive for the implementation of a number of specific circuits, such as multiplexers and XOR-dominated logic such as adders.

Dynamic logic, on the other hand, makes it possible to implement fast and small complex gates. This comes at a price. Parasitic effects such as charge sharing make the design process a precarious job. Charge leakage forces a periodic refresh, which puts a lower bound on the operating frequency of the circuit.

The current trend is towards an increased use of complementary static CMOS. This tendency is inspired by the increased use of design-automation tools at the logic design level. These tools emphasize optimization at the logic rather than the circuit level and put a premium on robustness. Another argument is that static CMOS is more amenable to voltage scaling than some of the other approaches discussed in this chapter.

6.4.2 Designing Logic for Reduced Supply Voltages

In Chapter 3, we projected that the supply voltage for CMOS processes will continue to drop over the coming decade, and may go as low as 0.6V by 2010. To maintain performance under those conditions, it is essential that the device thresholds scale as well. Fig-

ure 6.72a shows a plot of the (V_T, V_{DD}) ratio required to maintain a given performance level (assuming that other device characteristics remain identical).

This trade-off is not without penalty. Reducing the threshold voltage, increases the subthreshold leakage current exponentially as we derived in Eq. (3.40) (repeated here for the sake of clarity).

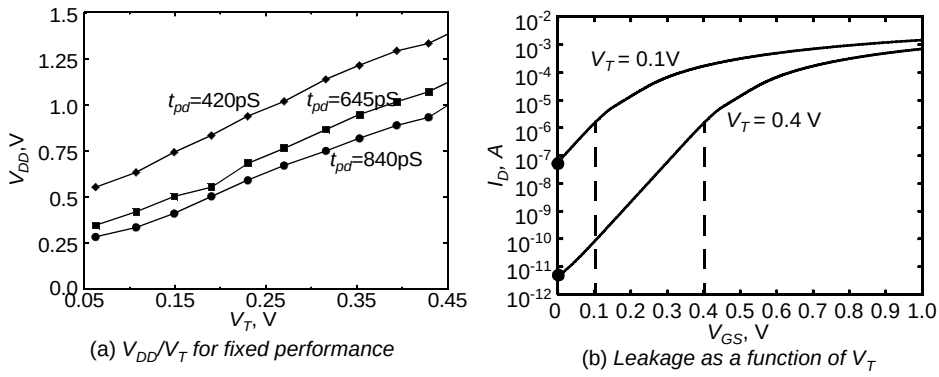


Figure 6.72 Voltage Scaling (V_{DD}/V_T on delay and leakage)

$$I_{leakage} = I_S 10^{\frac{V_{GS}-V_{Th}}{S}} \left(1 - 10^{-\frac{nV_{DS}}{S}}\right) \tag{6.40}$$

with S the *slope factor* of the device. The subthreshold leakage of an inverter is the current of the NMOS for V_{in} = 0V and $V_{out} = V_{DD}$ (or the PMOS current for $V_{in} = V_{DD}$ and V_{out} = 0). The exponential increase in inverter leakage for decreasing thresholds illustrated in Figure 6.72b.

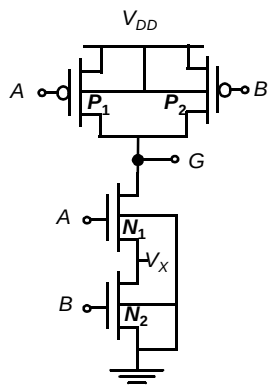
These leakage currents are particularly a concern for designs that feature intermittent computational activity separated by long periods of inactivity. For example, the processor in a cellular phone remains in idle mode for a majority of the time. While the processor is shutdown mode, the system should ideally consume zero or near-zero power. This is only possible if leakage is low—this is, the devices have a high threshold voltage. This is in contradictory to the scaling scenario that we just depicted, where high performance under low supply voltage means reduced thresholds. To satisfy the contradicting requirements of high-performance during active periods, and low leakage during standby, several process modifications or leakage-control techniques have been introduced in CMOS processes. Most processes with feature sizes at and below 0.18 μm CMOS support devices with different thresholds—typically a device with low threshold for high performance circuits, and a transistor with high threshold for leakage control. Another approach that is gaining ground is the dynamic control of the threshold voltage of a device by exploiting the body effect of the transistor. To use this approach for the control of individual devices requires a dual-well process (see Figure 2.2).

Clever circuit design can also help to reduce the leakage current, which is a function of the circuit topology and the value of the inputs applied to the gate. Since V_T depends on body bias (V_{BS}), the sub-threshold leakage of an MOS transistor depends not only on the gate drive (V_{GS}), but also on the body bias. In an inverter with In = 0, the sub-threshold

leakage of the inverter is set by the NMOS transistor with its $V_{GS} = V_{BS} = 0$ V. In more complex CMOS gates, the leakage current depends upon the input vector. For example, the sub-threshold leakage current of a two-input NAND gate is the least when $A = B = 0$. Under these conditions, the intermediate node X settles to,

$$V_X \approx V_{th} \ln(1 + n) \tag{6.41}$$

The NAND gate sub-threshold leakage is then set by the top-most NMOS transistor with $V_{GS} = V_{BS} = -V_X$. Clearly, the sub-threshold leakage under this condition is slightly smaller than that of the inverter. This reduction in sub-threshold leakage due to stacked transistors is called the *stack effect*. Figure 6.73 shows the leakage components for a simple two-input NAND gate.



A	B	V_X	I_{SUB}
0	0	V_{th} ln (1+n)	I^N_{SUB} ($V_{GS} = V_{BS} = -V_X$)
0	1	0	I^N_{SUB} ($V_{GS} = V_{BS} = 0$)
1	0	V_{dd}-V_T	I^N_{SUB} ($V_{GS} = V_{BS} = 0$)
1	1	0	2 I^P_{SUB} ($V_{SG} = V_{SB} = 0$)

Figure 6.73 Sub-threshold leakage reduction due to stack effect in a two-input NAND gate.

In short-channel MOS transistors, the sub-threshold leakage current depends not only on the gate drive (V_{GS}) and the body bias (V_{BS}), but also depends on the drain voltage (V_{DS}). The threshold voltage of a short-channel MOS transistor decreases with increasing V_{DS} due to *drain induced barrier lowering* (DIBL). Typical value for DIBL can range from 20-150 mV change in V_T per volt change in V_{DS}. Figure 6.74 illustrates the impact on the sub-threshold leakage as a result of

- a decrease in gate drive—point *A* to *B*
- an increase in body bias—point *A* to *C*
- an increase in drain voltage—point *A* to *D*.

Because of the latter, the impact of the stack effect is even more significant for short-channel transistors. The intermediate voltage reduces the drain-source voltage of the top-most device, and hence reduces its leakage. Consider again the two-input NAND gate of Figure 6.73, when both $M_1$and M_2 are *off*. From the simulated load lines shown in Figure 6.75, we see that V_X settles to approximately 100 mV in steady state. The steady state sub-threshold leakage in the NAND gate is hence due to $V_{GS} = V_{BS} = -100$ mV and $V_{DS} = V_{DD} - 100$ mV, which is 20 times smaller than that leakage of a stand-alone NMOS transistor with $V_{GS} = V_{BS} = 0$ mV and $V_{DS} = V_{DD}$ [Ye98].

In summary, the sub-threshold leakage in complex stacked circuits can be significantly lower than in individual devices. Observe that the maximum leakage reduction occurs when all the transistors in the stack are *off*, and the intermediate node voltage

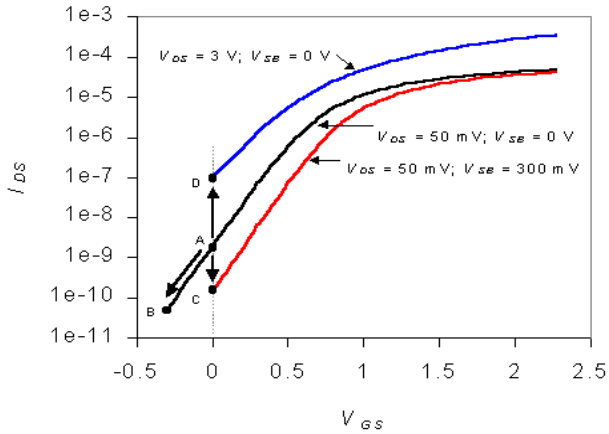


Figure 6.74 Dependence of sub-threshold leakage current on terminal voltages for a typical 0.25 µm NMOS

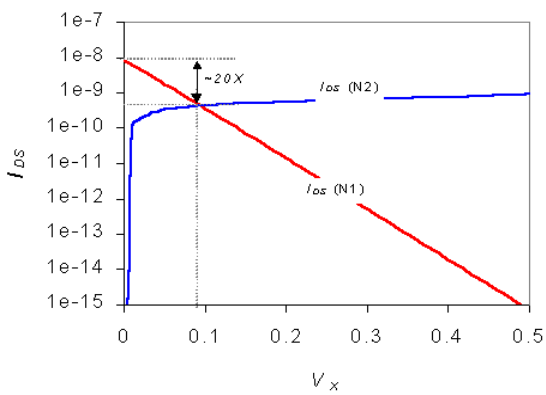


Figure 6.75 Load line indicating the steady state solution for the intermediate node voltage in a transistor stack.

reaches its steady state value. Exploiting this effect requires a careful selection of the input signals to every gate during standby or sleep mode.

Problem 6.8 Computing V_X

Eq. (6.41) represents intermediate node voltage for a two-input NAND with less than 10% error, when $A = B = 0$. Derive Eq. (6.41) assuming (i) V_T and I_o of M_1 and M_2 are approximately equal, (ii) NMOS transistors are identically sized, and (iii) $n < 1.5$. Explain the temperature dependence of stack effect and leakage reduction due to stack effect using equation (4.5).

6.5 Summary

In this chapter, we have extensively analyzed the behavior and performance of combinational CMOS digital circuits with regard to area, speed, and power.

- Static complementary CMOS combines dual pull-down and pull-up networks, only one of which is enabled at any time.
- The performance of a CMOS gate is a strong function of the fan-in. Techniques to deal with fan-in include transistor sizing, input reordering, and partitioning. The speed is also a linear function of the fan-out. Extra buffering is needed for large fan-outs.
- The ratioed logic style consists of an active pull-down (up) network connected to a load device. This results in a substantial reduction in gate complexity at the expense of static power consumption and an asymmetrical response. Careful transistor sizing is necessary to maintain sufficient noise margins. The most popular approaches in this class are the pseudo-NMOS techniques and differential DCVSL, which requires complementary signals.
- Pass-transistor logic implements a logic gate as a simple switch network. This results in very simple implementations for some logic functions. Long cascades of switches are to be avoided due to a quadratic increase in delay with respect to the number of elements in the chain. NMOS-only pass-transistor logic produces even simpler structures, but might suffer from static power consumption and reduced noise margins. This problem can be addressed by adding a level-restoring transistor.
- The operation of dynamic logic is based on the storage of charge on a capacitive node and the conditional discharging of that node as a function of the inputs. This calls for a two-phase scheme, consisting of a precharge followed by an evaluation step. Dynamic logic trades off noise margin for performance. It is sensitive to parasitic effects such as leakage, charge redistribution, and clock feedthrough. Cascading dynamic gates can cause problems, and should be addressed carefully.
- The power consumption of a logic network is strongly related to the switching activity of the network. This activity is a function of the input statistics, the network topology, and the logic style. Sources of power consumption such as glitches and short-circuit currents can be minimized by careful circuit design and transistor sizing.
- Threshold voltage scaling is required for low-voltage operation. Leakage control is critical for low-voltage operation

6.6 To Probe Further

The topic of (C)MOS logic styles is treated extensively in the literature. Numerous texts have been devoted to the issue. Some of the most comprehensive treatments can be found in [Weste93] and [Chandrakasan01]. Regarding the intricacies of high-performance design, [Shoji96] and [Bernstein98] offer the most in-depth discussion of the optimization and analysis of digital MOS circuits. The topic of power minimization is relatively new. Excellent reference works are [Chandrakasan95] and [Rabaey95].

Innovations in the MOS logic area are typically published in the proceedings of the ISSCC Conference and the VLSI circuits symposium, as well as the *IEEE Journal of Solid State Circuits* (especially the November issue).

REFERENCES

[Bernstein98] K. Bernstein et al., *High-Speed CMOS Design Styles*, Kluwer Academic Publishers, 1998.

[Chandrakasan95] A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995.

[Chandrakasan01] A. Chandrakasan, W. Bowhill, and F. Fox, ed., *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2001.

[Friedman84] V. Friedman and S. Liu, "Dynamic Logic CMOS Circuits," *IEEE Journal of Solid State Circuits*, vol. SC-19, no. 2, pp. 263–266, April 1984.

[Goncalvez83] N. Goncalvez and H. De Man, "NORA: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures," *IEEE Journal of Solid State Circuits*, vol. SC-18, no. 3, pp. 261–266, June 1983.

[Heller84] L. Heller et al., "Cascade Voltage Switch Logic: A Differential CMOS Logic Family," *Proc. IEEE ISSCC Conference*, pp. 16–17, February 1984.

[Krambeck82] R. Krambeck et al., "High-Speed Compact Circuits with CMOS," *IEEE Journal of Solid State Circuits*, vol. SC-17, no. 3, pp. 614–619, June 1982.

[Lee86] C. M. Lee and E. Szeto, "Zipper CMOS," *IEEE Circuits and Systems Magazine*, pp. 10–16, May 1986.

[Parameswar96] A.Parameswar, H.Hara, and T.Sakurai, "A swing restored pass-transistor logic-based multiply and accumulate circuit for multimedia applications," *IEEE Journal of Solid State Circuits*, vol.SC-31, no.6, pp.805-809, June 1996.

[Rabaey95] J. Rabaey and M. Pedram, Ed., *Low Power Design Methodologies*, Kluwer, 1995.

[Radhakrishnan85] D. Radhakrishnan, S. Whittaker, and G. Maki, "Formal Design Procedures for Pass-Transistor Switching Circuits," *IEEE Journal of Solid State Circuits*, vol. SC-20, no. 2, pp. 531–536, April 1985.

[Shoji88] M. Shoji, *CMOS Digital Circuit Technology*, Prentice Hall, 1988.

[Shoji96] M. Shoji, *High-Speed Digital Circuits*, Addison-Wesley, 1996.

[Sutherland99] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort*, Morgan Kaufmann, 1999.

[Weste93] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, 1993.

[Yano90] K. Yano et al., "A 3.8 ns CMOS 16 × 16 b Multiplier Using Complimentary Pass-Transistor Logic," *IEEE Journal of Solid State Circuits*, vol. SC-25, no 2, pp. 388–395, April 1990.

[Ye98] Y. Ye, S. Borkar, and V. De, "A new technique for standby leakage reduction in high-performance circuits," *Symposium on VLSI Circuits*, pp 40-41, 1998.

Chapter 6
PROBLEMS

1. [E, None, 4.2] Implement the equation $X = ((\overline{A} + \overline{B})\,(\overline{C} + \overline{D} + \overline{E}) + \overline{F})\,\overline{G}$ using complementary CMOS. Size the devices so that the output resistance is the same as that of an inverter with an NMOS $W/L = 2$ and PMOS $W/L = 6$. Which input pattern(s) would give the worst and best equivalent pull-up or pull-down resistance?

2. Implement the following expression in a full static CMOS logic fashion using no more than 10 transistors:

$$\overline{Y} = (A \cdot B) + (A \cdot C \cdot E) + (D \cdot E) + (D \cdot C \cdot B)$$

3. Consider the circuit of Figure 6.1.

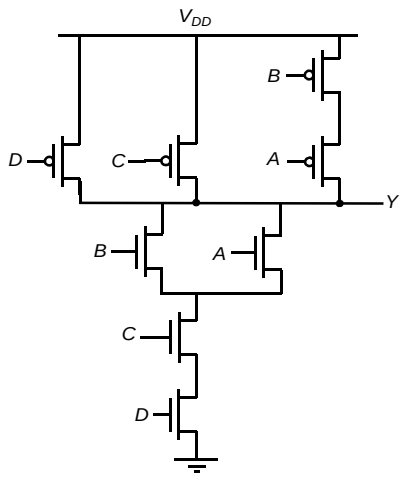


Figure 6.1 CMOS combinational logic gate.

a. What is the logic function implemented by the CMOS transistor network? Size the NMOS and PMOS devices so that the output resistance is the same as that of an inverter with an NMOS $W/L = 4$ and PMOS $W/L = 8$.

b. What are the input patterns that give the worst case t_{pHL} and t_{pLH}. State clearly what are the initial input patterns and which input(s) has to make a transition in order to achieve this maximum propagation delay. Consider the effect of the capacitances at the internal nodes.

c. Verify part (b) with SPICE. Assume all transistors have minimum gate length (0.25μm).

d. If P(A=1)=0.5, P(B=1)=0.2, P(C=1)=0.3 and P(D=1)=1, determine the power dissipation in the logic gate. Assume V_{DD}=2.5V, C_{out}=30fF and f_{clk}=250MHz.

4. [M, None, 4.2] CMOS Logic

a. Do the following two circuits (Figure 6.2) implement the same logic function? If yes, what is that logic function? If no, give Boolean expressions for both circuits.

b. Will these two circuits' output resistances always be equal to each other?

c. Will these two circuits' rise and fall times always be equal to each other? Why or why not?

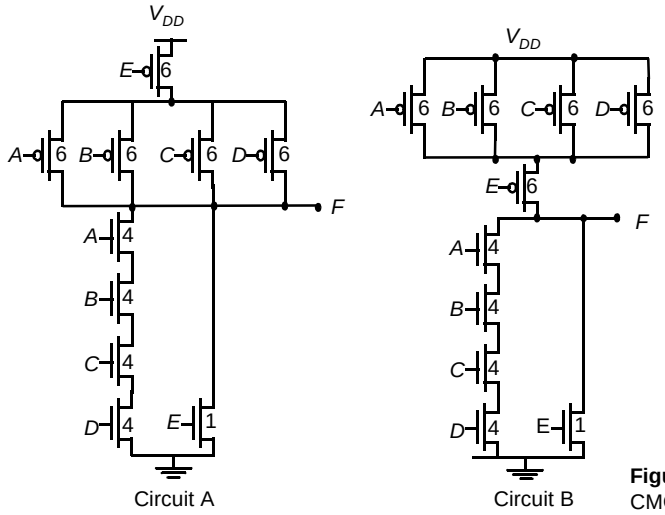


Figure 6.2 Two static CMOS gates.

5. [E, None, 4.2] The transistors in the circuits of the preceding problem have been sized to give an output resistance of 13 kΩ for the worst-case input pattern. This output resistance can vary, however, if other patterns are applied.

a. What input patterns (*A*–*E*) give the lowest output resistance when the output is low? What is the value of that resistance?

b. What input patterns (*A*–*E*) give the lowest output resistance when the output is high? What is the value of that resistance?

6. [E, None, 4.2] What is the logic function of circuits A and B in Figure 6.3? Which one is a dual network and which one is not? Is the nondual network still a valid static logic gate? Explain. List any advantages of one configuration over the other.

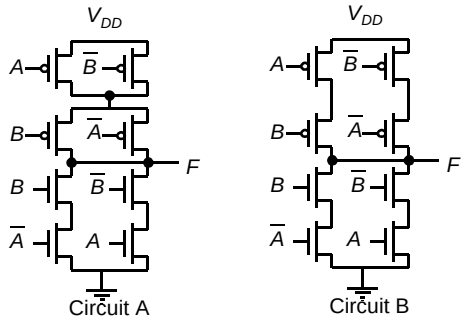


Figure 6.3 Two logic functions.

7. [E, None, 4.2] Compute the following for the pseudo-NMOS inverter shown in Figure 6.4:

a. V_{OL} and V_{OH}

b. NM_L and NM_H

c. The power dissipation: (1) for V_{in} low, and (2) for V_{in} high

d. For an output load of 1 pF, calculate t_{pLH}, t_{pHL}, and t_p. Are the rising and falling delays equal? Why or why not?

8. [M, SPICE, 4.2] Consider the circuit of Figure 6.5.

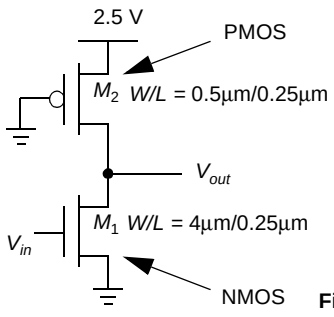


Figure 6.4 Pseudo-NMOS inverter.

a. What is the output voltage if only one input is high? If all four inputs are high?

b. What is the average static power consumption if, at any time, each input turns on with an (independent) probability of 0.5? 0.1?

c. Compare your analytically obtained results to a SPICE simulation.

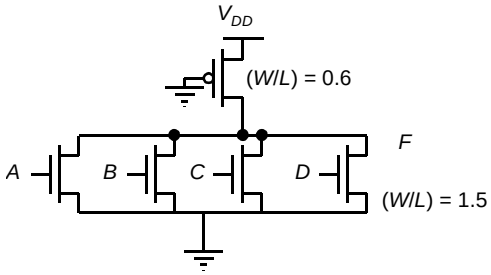


Figure 6.5 Pseudo-NMOS gate.

9. [M, None, 4.2] Implement $F = A\overline{BC} + \overline{A}CD$ (and $\overline{F}$) in DCVSL. Assume A, B, C, D, and their complements are available as inputs. Use the minimum number of transistors.

10. [E, Layout, 4.2] A complex logic gate is shown in Figure 6.6.

a. Write the Boolean equations for outputs F and G. What function does this circuit implement?

b. What logic family does this circuit belong to?

c. Assuming W/L = 0.5u/0.25u for *all nmos* transistors and W/L = 2u/0.25u for the *pmos* transistors, produce a layout of the gate using Magic. Your layout should conform to the following datapath style: (1) Inputs should enter the layout from the left in polysilicon; (2) The outputs should exit the layout at the right in polysilicon (since the outputs would probably be driving transistor gate inputs of the next cell to the right); (3) Power and ground lines should run vertically in metal 1.

d. Extract and netlist the layout. Load both outputs (F,G) with a 30fF capacitance and simulate the circuit. Does the gate function properly? If not, explain why and resize the transistors so that it does. Change the sizes (and areas and perimeters) in the HSPICE netlist.

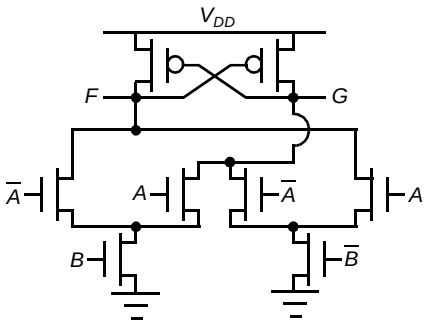
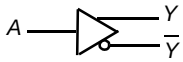


Figure 6.6 Two-input complex logic gate.

11. Design and simulate a circuit that generates an optimal differential signal as shown in Figure 6.7. Make sure the rise and fall times are equal.



A	Y	$\overline{Y}$
0	0	1
1	1	0

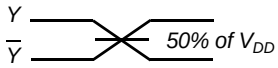


Figure 6.7 Differential Buffer.

12. What is the function of the circuit in Figure 6.8?

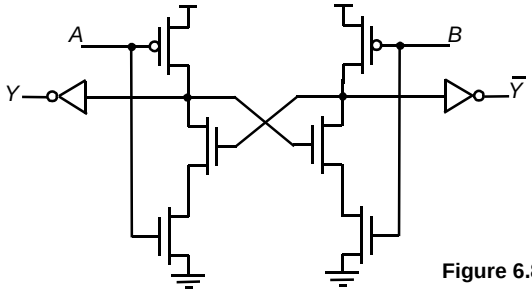


Figure 6.8 Gate.

13. Implement the function $S = ABC + A\overline{BC} + \overline{AB}C + \overline{A}B\overline{C}$, which gives the sum of two inputs with a carry bit, using NMOS pass transistor logic. Design a DCVSL gate which implements the same function. Assume A, B, C, and their complements are available as inputs.

14. Describe the logic function computed by the circuit in Figure 6.9. Note that all transistors (except for the middle inverters) are NMOS. Size and simulate the circuit so that it achieves a

100 ps delay (50-50) using 0.25μm devices, while driving a 100 fF load on both differential outputs. (V_{DD} = 2.5V) Assume *A*, *B* and their complements are available as inputs.

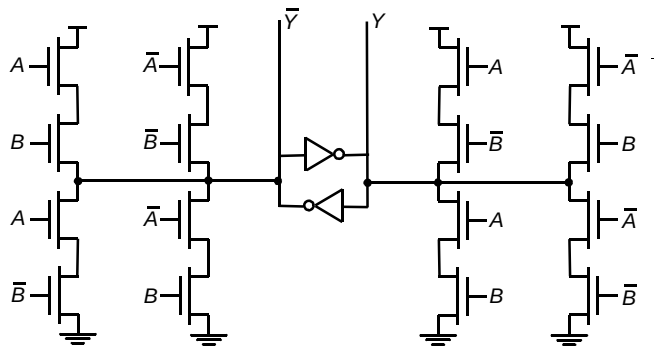


Figure 6.9 Cascoded Logic Styles.

For the drain and source perimeters and areas you can use the following approximations: AS=AD=W*0.625u and PS=PD=W+1.25u.

15. [M, None. 4.2] Figure 6.10 contains a pass-gate logic network.

a. Determine the truth table for the circuit. What logic function does it implement?

b. Assuming 0 and 2.5 V inputs, size the PMOS transistor to achieve a V_{OL} = 0.3 V.

c. If the PMOS were removed, would the circuit still function correctly? Does the PMOS transistor serve any useful purpose?

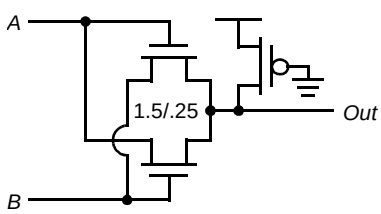


Figure 6.10 Pass-gate network.

16. [M, None, 4.2] This problem considers the effects of process scaling on pass-gate logic.

a. If a process has a t_{buf} of 0.4 ns, R_{eq} of 8 kΩ, and *C* of 12 fF, what is the optimal number of stages between buffers in a pass-gate chain?

b. Suppose that, if the dimension of this process are shrunk by a factor *S*, R_{eq} scales as $1/S^2$, *C* scales as $1/S$, and t_{buf} scales as $1/S^2$. What is the expression for the optimal number of buffers as a function of *S*? What is this value if *S* = 2?

17. [C, None, 4.2] Consider the circuit of Figure 6.11. Let C_x = 50 fF, M_r has W/L = 0.375/0.375, M_n has W/L_{eff} = 0.375/0.25. Assume the output inverter doesn't switch until its input equals $V_{DD}/2$.

a. How long will it take M_n to pull down node *x* from 2.5 V to 1.25 V if *In* is at 0 V and *B* is at 2.5V?

b. How long will it take M_n to pull up node *x* from 0 V to 1.25 V if V_{In} is 2.5 V and V_B is 2.5 V?

c. What is the minimum value of V_B necessary to pull down V_x to 1.25 V when V_{In} = 0 V?

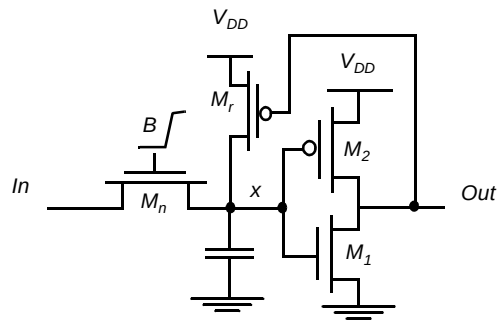
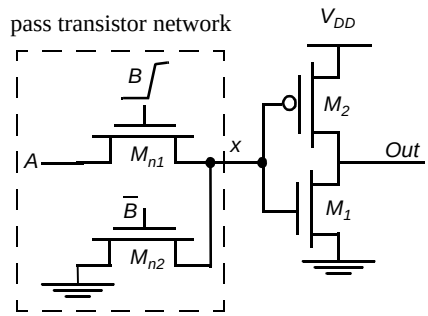


Figure 6.11 Level restorer.

18. Pass Transistor Logic



V_{DD} = 2.5V
$(W/L)_2$ = 1.5um/0.25um
$(W/L)_1$ = 0.5um/0.25um
$(W/L)_{ni}$ = 0.5um/0.25um

k_n' = 115uA/V^2, kp' = -30uA/V^2
V_{tN} = 0.43V, V_{tP} = -0.4V

Figure 6.12 Level restoring circuit.

Consider the circuit of Figure 6.12. Assume the inverter switches ideally at $V_{DD}/2$, neglect body effect, channel length modulation and all parasitic capacitance throughout this problem.

a. What is the logic function performed by this circuit?

b. Explain why this circuit has non-zero static dissipation.

c. Using only just 1 transistor, design a fix so that there will not be any static power dissipation. Explain how you chose the size of the transistor.

d. Implement the same circuit using transmission gates.

e. Replace the pass-transistor network in Figure 6.12 with a pass transistor network that computes the following function: $x = ABC$ at the node x. Assume you have the true and complementary versions of the three inputs A,B and C.

19. [M, None, 4.3] Sketch the waveforms at *x*, *y*, and *z* for the given inputs (Figure 6.13). You may approximate the time scale, but be sure to compute the voltage levels. Assume that $V_T = 0.5$ V when body effect is a factor.

20. [E, None, 4.3] Consider the circuit of Figure 6.14.

a. Give the logic function of *x* and *y* in terms of *A*, *B*, and *C*. Sketch the waveforms at *x* and *y* for the given inputs. Do *x* and *y* evaluate to the values you expected from their logic functions? Explain.

b. Redesign the gates using *np*-CMOS to eliminate any race conditions. Sketch the waveforms at *x* and *y* for your new circuit.

21. [M, None, 4.3] Suppose we wish to implement the two logic functions given by $F = A + B + C$ and $G = A + B + C + D$. Assume both true and complementary signals are available.

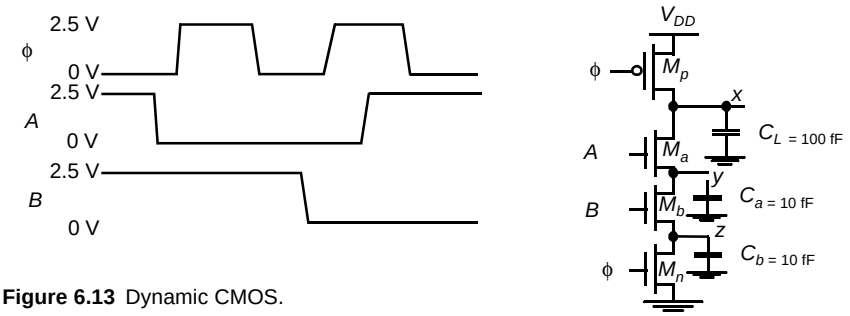


Figure 6.13 Dynamic CMOS.

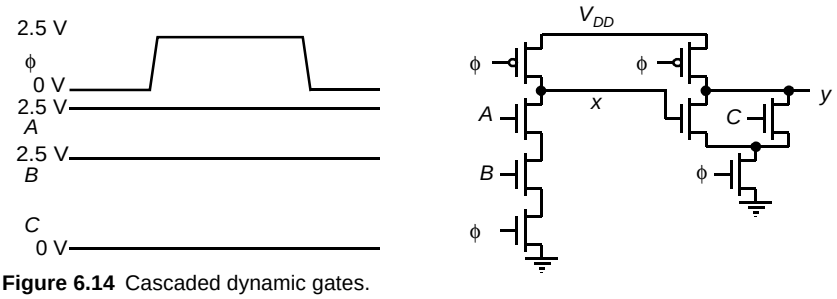


Figure 6.14 Cascaded dynamic gates.

a. Implement these functions in dynamic CMOS as cascaded ϕ stages so as to minimize the total transistor count.

b. Design an *np*-CMOS implementation of the same logic functions. Does this design display any of the difficulties of part (a)?

22. Consider a conventional 4-stage Domino logic circuit as shown in Figure 6.15 in which all precharge and evaluate devices are clocked using a common clock ϕ. For this entire problem, assume that the pulldown network is simply a single NMOS device, so that each Domino stage consists of a dynamic inverter followed by a static inverter. Assume that the precharge time, evaluate time, and propagation delay of the static inverter are all $T/2$. Assume that the transitions are ideal (zero rise/fall times).

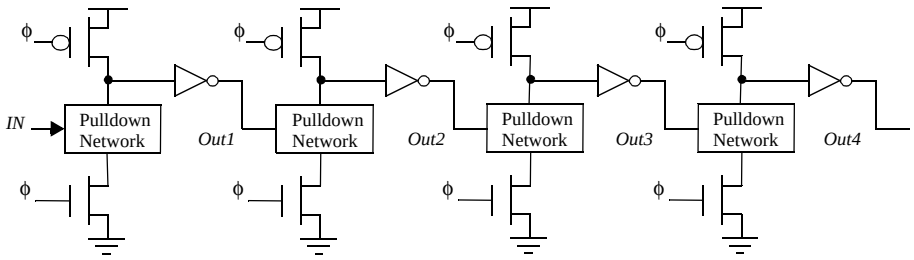


Figure 6.15 Conventional DOMINO Dynamic Logic.

a. Complete the timing diagram for signals Out_1, Out_2, Out_3 and Out_4, when the *IN* signal goes high before the rising edge of the clock ϕ. Assume that the clock period is 10 T time units.

b. Suppose that there are no evaluate switches at the 3 latter stages. Assume that the clock ϕ is initially in the precharge state (ϕ=0 with all nodes settled to the correct precharge states), and the block enters the evaluate period (ϕ=1). Is there a problem during the evaluate period, or is there a benefit? Explain.

c. Assume that the clock ϕ is initially in the evaluate state (ϕ=1), and the block enters the precharge state (ϕ = 0). Is there a problem, or is there any benefit, if the last three evaluate switches are removed? Explain.

23. [C, Spice, 4.3] Figure 6.16 shows a dynamic CMOS circuit in Domino logic. In determining source and drain areas and perimeters, you may use the following approximations: $AD = AS = W \times 0.625\mu m$ and $PD = PS = W + 1.25\mu m$. Assume 0.1 ns rise/fall times for all inputs, including the clock. Furthermore, you may assume that all the inputs and their complements are available, and that all inputs change during the precharge phase of the clock cycle.

a. What Boolean functions are implemented at outputs *F* and *G*? If *A* and *B* are interpreted as two-bit binary words, $A = A_1A_0$ and $B = B_1B_0$, then what interpretation can be applied to output *G*?

b. Which gate (1 or 2) has the highest potential for harmful charge sharing and why? What sequence of inputs (spanning two clock cycles) results in the worst-case charge-sharing scenario? Using SPICE, determine the extent to which charge sharing affects the circuit for this worst case..

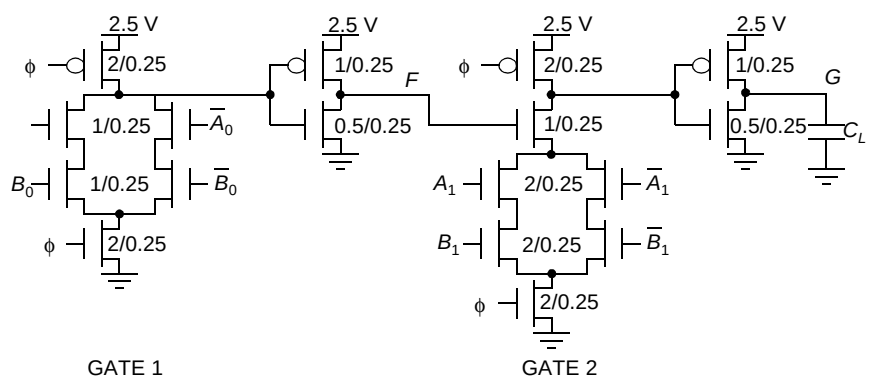


Figure 6.16 DOMINO logic circuit.

24. [M, Spice, 4.3] In this problem you will consider methods for eliminating charge sharing in the circuit of Figure 6.16. You will then determine the performance of the resulting circuit.

a. In problem 24 you determined which gate (1 or 2) suffers the most from charge sharing. Add a single 2/0.25 PMOS precharge transistor (with its gate driven by the clock ϕ and its source connected to V_{DD}) to one of the nodes in that gate to maximally reduce the charge-sharing effect. What effect (if any) will this addition have on the gate delay? Use SPICE to demonstrate that the additional transistor has eliminated charge sharing for the previously determined worst-case sequence of inputs.

b. For the new circuit (including additional precharge transistor), find the sequence of inputs (spanning two clock cycles) that results in the worst-case delay through the circuit.

Remember that precharging is another factor that limits the maximum clocking frequency of the circuit, so your input sequence should address the worst-case precharging delay.

c. Using SPICE on the new circuit and applying the sequence of inputs found in part (b), find the maximum clock frequency for correct operation of the circuit. Remember that the precharge cycle must be long enough to allow all precharged nodes to reach ~90% of their final values before evaluation begins. Also, recall that the inputs (*A*, *B* and their complements) should not begin changing until the clock signal has reached 0 V (precharge phase), and they should reach their final values before the circuit enters the evaluation phase.

25. [C, None, 4.2–3] For this problem, refer to the layout of Figure 6.17.

a. Draw the schematic corresponding to the layout. Include transistor sizes.

b. What logic function does the circuit implement? To which logic family does the circuit belong?

c. Does the circuit have any advantages over fully complementary CMOS?

d. Calculate the worst-case V_{OL} and V_{OH}.

e. Write the expresions for the area and perimeter of the drain and source for all of the FETs in terms of λ. Assume that the capacitance of shared diffuusions divides evenly between the sharing devices. Copy the layout into Magic, extract and simulate to find the worst-case t_{pHL} time. For what input transition(s) does this occur? Name all of the parasitic capacitances that you would need to know to calculate this delay by hand (you do not need to perform the calculation).

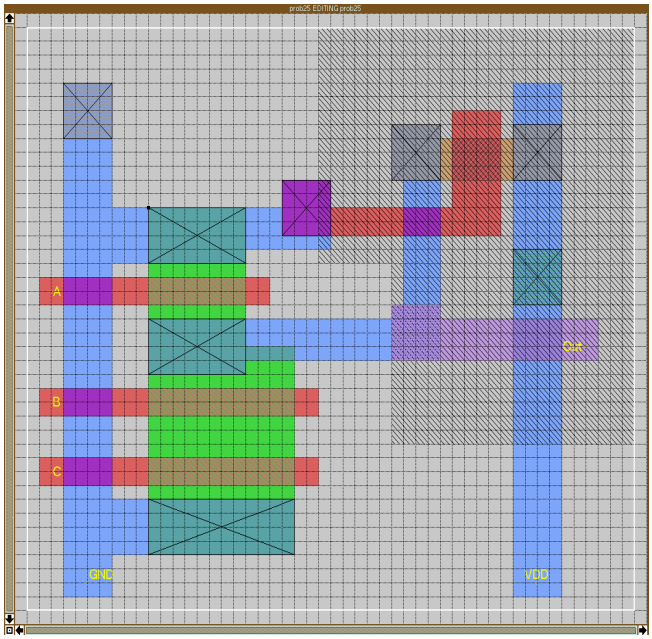


Figure 6.17 Layout of complex gate.

26. [E, None, 4.4] Derive the truth table, state transition graph, and output transition probabilities for a three-input XOR gate with independent, identically distributed, uniform white-noise inputs.

27. [C, None, 4.4] Figure 6.18 shows a two-input multiplexer. For this problem, assume independent, identically-distributed uniform white noise inputs.

a. Does this schematic contain reconvergent fan-out? Explain your answer.

b. Find the exact signal (P_1) and transition ($P_{0 \rightarrow 1}$) formulas for nodes *X*, *Y*, and *Z* for: (1) a static, fully complementary CMOS implementation, and (2) a dynamic CMOS implementation.

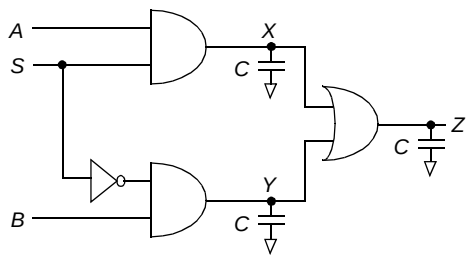


Figure 6.18 Two-input multiplexer

28. [M, None, 4.4] Compute the switching power consumed by the multiplexer of Figure 6.18, assuming that all significant capacitances have been lumped into the three capacitors shown in the figure, where $C = 0.3$ pF. Assume that $V_{DD} = 2.5$ V and independent, identically-distributed uniform white noise inputs, with events occuring at a frequency of 100 MHz. Perform this calculation for the following:

a. A static, fully-complementary CMOS implementation

b. A dynamic CMOS implementation

29. Consider the circuit shown Figure 6.19.

a. What is the logic function implemented by this circuit? Assume that all devices (M1-M6) are 0.5µm/0.25µm.

b. Let the drain current for each device (NMOS and PMOS) be 1µA for NMOS at $V_{GS} = V_T$ and PMOS at $V_{SG} = V_T$. What input vectors cause the worst case leakage power for each output value? Explain (state all the vectors, but do not evaluate the leakage). Ignore DIBL**.**

c. Suppose the circuit is active for a fraction of time *d* and idle for *(1-d)*. When the circuit is active, the inputs arrive at 100 MHz and are uniformly distributed ($Pr_{(A=1)} = 0.5$, $Pr_{(B=1)} = 0.5$, $Pr_{(C=1)} = 0.5$) and independent. When the circuit is in the idle mode, the inputs are fixed to one you chose in part (b). What is the duty cycle *d* for which the active power is equal to the leakage power?

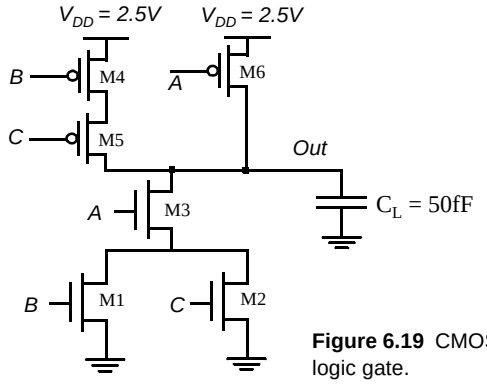


Figure 6.19 CMOS logic gate.

DESIGN PROJECT

Design, lay out, and simulate a CMOS four-input XOR gate in the standard 0.25 micron CMOS process. You can choose any logic circuit style, and you are free to choose how many stages of logic to use: you could use one large logic gate or a combination of smaller logic gates. The supply voltage is set at 2.5 V! Your circuit must drive an external 20 fF load in addition to whatever internal parasitics are present in your circuit.

The primary design objective is to minimize the propagation delay of the worst-case transition for your circuit. The secondary objective is to minimize the area of the layout. At the very worst, your design must have a propagation delay of no more than 0.5 ns and occupy an area of no more than 500 square microns, but the faster and smaller your circuit, the better. Be aware that, when using dynamic logic, the precharge time should be made part of the delay.

The design will be graded on the magnitude of $A \times t_p^2$, the product of the area of your design and the square of the delay for the worst-case transition.

CHAPTER 7

DESIGNING SEQUENTIAL LOGIC CIRCUITS

Implementation techniques for flip-flops, latches, oscillators, pulse generators, and Schmitt triggers

n

Static versus dynamic realization

n

Choosing clocking strategies

7.1 Introduction
- 7.1.1 Timing Metrics for Sequential Circuits
- 7.1.2 Classification of Memory Elements

7.2 Static Latches and Registers
- 7.2.1 The Bistability Principle
- 7.2.2 SR Flip-Flops
- 7.2.3 Multiplexer-Based Latches
- 7.2.4 Master-Slave Edge-Triggered Register
- 7.2.5 Low-Voltage Static Latches

7.3 Dynamic Latches and Registers
- 7.3.1 Dynamic Transmission-Gate Edge-triggered Registers
- 7.3.2 C2MOS—A Clock-Skew Insensitive Approach
- 7.3.3 True Single-Phase Clocked Register (TSPCR)

7.4 Alternative Register Styles
- 7.4.1 Pulse Registers
- 7.4.2 Sense-Amplifier Based Registers

7.5 Pipelining: An approach to optimize sequential circuits
- 7.5.1 Latch- vs. Register-Based Pipelines
- 7.5.2 NORA-CMOS—A Logic Style for Pipelined Structures

7.6 Non-Bistable Sequential Circuits
- 7.6.1 The Schmitt Trigger
- 7.6.2 Monostable Sequential Circuits
- 7.6.3 Astable Circuits

7.7 Perspective: Choosing a Clocking Strategy

7.8 Summary

7.9 To Probe Further

7.1 Introduction

Combinational logic circuits, described earlier, have the property that the output of a logic block is only a function of the *current* input values, assuming that enough time has elapsed for the logic gates to settle. Yet virtually all useful systems require storage of state information, leading to another class of circuits called *sequential logic* circuits. In these circuits, the output not only depends upon the *current* values of the inputs, but also upon *preceding* input values. In other words, a sequential circuit remembers some of the past history of the system—it has memory.

Figure 7.1 shows a block diagram of a generic *finite state machine* (FSM) that consists of combinational logic and registers, which hold the system state. The system depicted here belongs to the class of *synchronous* sequential systems, in which all registers are under control of a single global clock. The outputs of the FSM are a function of the current *Inputs* and the *Current State*. The *Next State* is determined based on the *Current State* and the current *Inputs* and is fed to the inputs of registers. On the rising edge of the clock, the *Next State* bits are copied to the outputs of the registers (after some propagation delay), and a new cycle begins. The register then ignores changes in the input signals until the next rising edge. In general, registers can be *positive edge-triggered* (where the input data is copied on the positive edge of the clock) or *negative edge-triggered* (where the input data is copied on the negative edge, as is indicated by a small circle at the clock input).

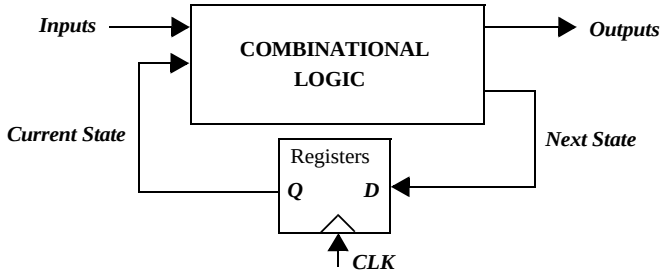


Figure 7.1 Block diagram of a finite state machine using *positive edge-triggered* registers.

This chapter discusses the CMOS implementation of the most important sequential building blocks. A variety of choices in sequential primitives and clocking methodologies exist; making the correct selection is getting increasingly important in modern digital circuits, and can have a great impact on performance, power, and/or design complexity. Before embarking on a detailed discussion on the various design options, a revision of the design metrics, and a classification of the sequential elements is necessary.

7.1.1 Timing Metrics for Sequential Circuits

There are three important timing parameters associated with a register as illustrated in Figure 7.2. The *set-up time* (t_{su}) is the time that the data inputs (*D* input) must be valid before the clock transition (this is, the 0 to 1 transition for a *positive edge-triggered* register). The *hold time* (t_{hold}) is the time the data input must remain valid after the clock edge. Assum-

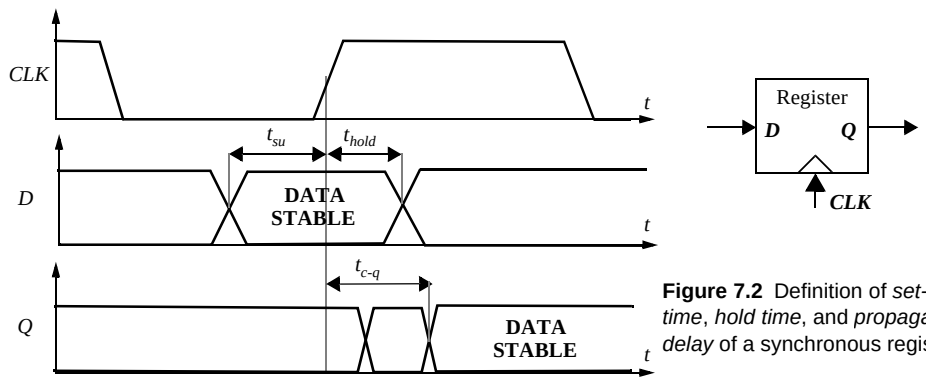


Figure 7.2 Definition of *set-up time*, *hold time*, and *propagation delay* of a synchronous register.

ing that the *set-up* and *hold*-times are met, the data at the *D* input is copied to the *Q* output after a worst-case *propagation delay* (with reference to the clock edge) denoted by $t_{c\text{-}q}$.

Given the timing information for the registers and the combination logic, some system-level timing constraints can be derived. Assume that the worst-case *propagation delay* of the logic equals t_{plogic}, while its minimum delay (also called the *contamination delay*) is t_{cd}. The minimum clock period *T*, required for proper operation of the sequential circuit is given by

$$T \geq t_{c\text{-}q} + t_{plogic} + t_{su} \tag{7.1}$$

The *hold time* of the register imposes an extra constraint for proper operation,

$$t_{cd\text{register}} + t_{cd\text{logic}} \geq t_{hold} \tag{7.2}$$

where $t_{cd\text{register}}$ is the minimum *propagation delay* (or *contamination delay*) of the register.

As seen from Eq. (7.1), it is important to minimize the values of the timing parameters associated with the register, as these directly affect the rate at which a sequential circuit can be clocked. In fact, modern high-performance systems are characterized by a very-low logic depth, and the register *propagation delay* and *set-up* times account for a significant portion of the clock period. For example, the DEC Alpha EV6 microprocessor [Gieseke97] has a maximum logic depth of 12 gates, and the register overhead stands for approximately 15% of the clock period. In general, the requirement of Eq. (7.2) is not hard to meet, although it becomes an issue when there is little or no logic between registers.[1]

7.1.2 Classification of Memory Elements

Foreground versus Background Memory

At a high level, memory is classified into background and foreground memory. Memory that is embedded into logic is foreground memory, and is most often organized as individual registers of register banks. Large amounts of centralized memory core are referred to as background memory. Background memory, discussed later in this book, achieves

[1] (or when the clocks at different registers are somewhat out of phase due to clock skew, as will be discussed in a later Chapter)

higher area densities through efficient use of array structures and by trading off performance and robustness for size. In this chapter, we focus on foreground memories.

Static versus Dynamic Memory

Memories can be static or dynamic. Static memories preserve the state as long as the power is turned on. Static memories are built using *positive feedback* or regeneration, where the circuit topology consists of intentional connections between the output and the input of a combinational circuit. Static memories are most useful when the register won't be updated for extended periods of time. An example of such is configuration data, loaded at power-up time. This condition also holds for most processors that use conditional clocking (i.e., gated clocks) where the clock is turned *off* for unused modules. In that case, there are no guarantees on how frequently the registers will be clocked, and static memories are needed to preserve the state information. Memory based on positive feedback fall under the class of elements called *multivibrator circuits.* The bistable element, is its most popular representative, but other elements such as monostable and astable circuits are also frequently used.

Dynamic memories store state for a short period of time—on the order of milliseconds. They are based on the principle of temporary *charge storage* on parasitic capacitors associated with MOS devices. As with dynamic logic discussed earlier, the capacitors have to be refreshed periodically to annihilate charge leakage. Dynamic memories tend to simpler, resulting in significantly higher performance and lower power dissipation. They are most useful in datapath circuits that require high performance levels and are periodically clocked. It is possible to use dynamic circuitry even when circuits are conditionally clocked, if the state can be discarded when a module goes into idle mode.

Latches versus Registers

A latch is an essential component in the construction of an *edge-triggered* register. It is *level-sensitive* circuit that passes the *D* input to the *Q* output when the clock signal is high. This latch is said to be in *transparent* mode. When the clock is low, the input data sampled on the falling edge of the clock is held stable at the output for the entire phase, and the latch is in *hold* mode. The inputs must be stable for a short period around the falling edge of the clock to meet *set-up* and *hold* requirements. A latch operating under the above conditions is a *positive latch*. Similarly, a *negative latch* passes the *D* input to the *Q* output when the clock signal is low. The signal waveforms for a positive and negative latch are shown in Figure 7.3. A wide variety of static and dynamic implementations exists for the realization of latches.

Contrary to *level-sensitive* latches, *edge-triggered* registers only sample the input on a clock transition — 0-to-1 for a *positive edge-triggered* register, and 1-to-0 for a *negative edge-triggered* register. They are typically built using the latch primitives of Figure 7.3. A most-often recurring configuration is the *master-slave* structure that cascades a positive and negative latch. Registers can also be constructed using one-shot generators of the clock signal ("glitch" registers), or using other specialized structures. Examples of these are shown later in this chapter.

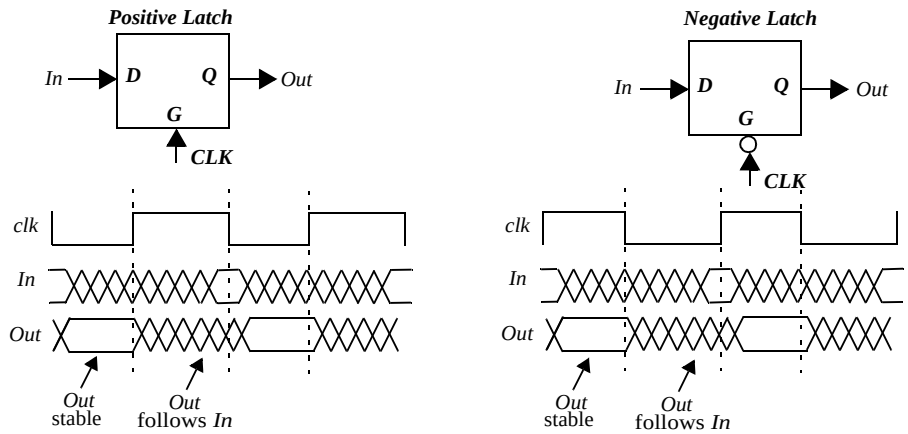


Figure 7.3 Timing of positive and negative latches.

7.2 Static Latches and Registers

7.2.1 The Bistability Principle

Static memories use positive feedback to create a *bistable circuit* — a circuit having two stable states that represent 0 and 1. The basic idea is shown in Figure 7.4a, which shows two inverters connected in cascade along with a voltage-transfer characteristic typical of such a circuit. Also plotted are the VTCs of the first inverter, that is, V_{o1} versus V_{i1}, and the second inverter (V_{o2} versus V_{o1}). The latter plot is rotated to accentuate that $V_{i2} = V_{o1}$. Assume now that the output of the second inverter V_{o2} is connected to the input of the first V_{i1}, as shown by the dotted lines in Figure 7.4a. The resulting circuit has only three possible operation points (*A*, *B*, and *C*), as demonstrated on the combined VTC. The following important conjecture is easily proven to be valid:

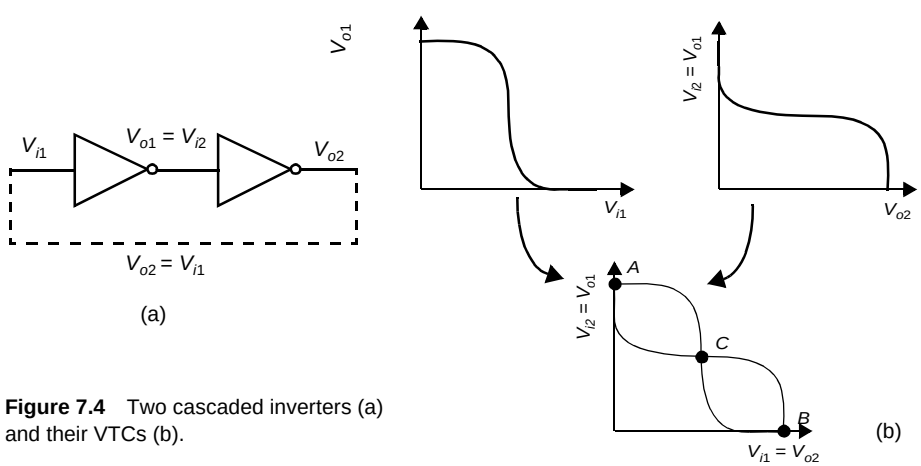


Figure 7.4 Two cascaded inverters (a) and their VTCs (b).

Under the condition that the gain of the inverter in the transient region is larger than 1, only *A* and *B* are stable operation points, and *C* is a metastable operation point.

Suppose that the cross-coupled inverter pair is biased at point *C*. A small deviation from this bias point, possibly caused by noise, is amplified and *regenerated* around the circuit loop. This is a consequence of the gain around the loop being larger than 1. The effect is demonstrated in Figure 7.5a. A small deviation δ is applied to V_{i1} (biased in *C*). This deviation is amplified by the gain of the inverter. The enlarged divergence is applied to the second inverter and amplified once more. The bias point moves away from *C* until one of the operation points *A* or *B* is reached. In conclusion, *C* is an unstable operation point. Every deviation (even the smallest one) causes the operation point to run away from its original bias. The chance is indeed very small that the cross-coupled inverter pair is biased at *C* and stays there. Operation points with this property are termed *metastable*.

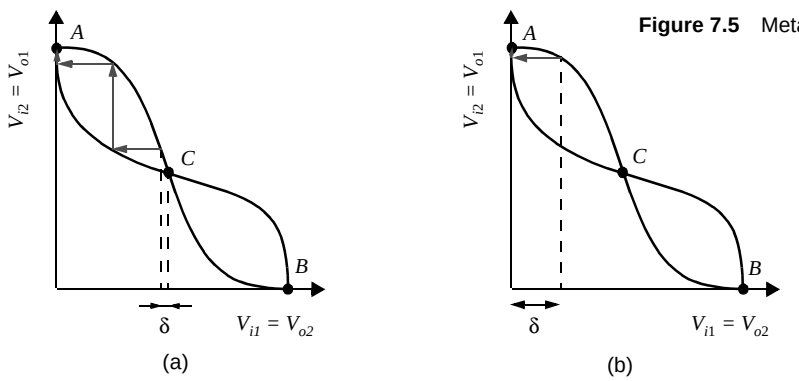


Figure 7.5 Metastability.

On the other hand, *A* and *B* are stable operation points, as demonstrated in Figure 7.5b. In these points, the **loop gain is much smaller than unity.** Even a rather large deviation from the operation point is reduced in size and disappears.

Hence the cross-coupling of two inverters results in a *bistable* circuit, that is, a circuit with two stable states, each corresponding to a logic state. The circuit serves as a memory, storing either a 1 or a 0 (corresponding to positions *A* and *B*).

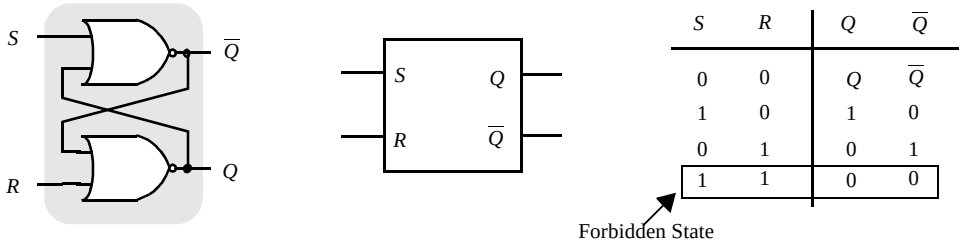
In order to change the stored value, we must be able to bring the circuit from state *A* to *B* and vice-versa. Since the precondition for stability is that the loop gain *G* is smaller than unity, we can achieve this by making *A* (or *B*) temporarily unstable by increasing *G* to a value larger than 1. This is generally done by applying a trigger pulse at V_{i1} or V_{i2}. For instance, assume that the system is in position *A* ($V_{i1} = 0$, $V_{i2} = 1$). Forcing V_{i1} to 1 causes both inverters to be on simultaneously for a short time and the loop gain *G* to be larger than 1. The positive feedback regenerates the effect of the trigger pulse, and the circuit moves to the other state (*B* in this case). The width of the trigger pulse need be only a little larger than the total *propagation delay* around the circuit loop, which is twice the average *propagation delay* of the inverters.

In summary, a bistable circuit has two stable states. In absence of any triggering, the circuit remains in a single state (assuming that the power supply remains applied to the circuit), and hence remembers a value. A trigger pulse must be applied to change the state

of the circuit. Another common name for a bistable circuit is *flip-flop* (unfortunately, an *edge-triggered* register is also referred to as a *flip-flop*).

7.2.2 SR Flip-Flops

The cross-coupled inverter pair shown in the previous section provides an approach to store a binary variable in a stable way. However, extra circuitry must be added to enable control of the memory states. The simplest incarnation accomplishing this is the well-know *SR —or set-reset— flip-flop,* an implementation of which is shown in Figure 7.6a. This circuit is similar to the cross-coupled inverter pair with NOR gates replacing the inverters. The second input of the NOR gates is connected to the trigger inputs (S and R), that make it possible to force the outputs Q and $\overline{Q}$ to a given state. These outputs are complimentary (except for the $SR = 11$ state). When both S and R are 0, the flip-flop is in a quiescent state and both outputs retain their value (a NOR gate with one of its input being 0 looks like an inverter, and the structure looks like a cross coupled inverter). If a positive (or 1) pulse is applied to the S input, the Q output is forced into the 1 state (with $\overline{Q}$ going to 0). Vice versa, a 1 pulse on R resets the flip-flop and the Q output goes to 0.



S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
1	1	0	0

(a) Schematic diagram (b) Logic symbol (c) Characteristic table

Figure 7.6 NOR-based *SR* flip-flop.

These results are summarized in the *characteristic table* of the flip-flop, shown in Figure 7.6c. The characteristic table is the truth table of the gate and lists the output states as functions of all possible input conditions. When both S and R are high, both Q and $\overline{Q}$ are forced to zero. Since this does not correspond with our constraint that Q and $\overline{Q}$ must be complementary, this input mode is considered to be forbidden. An additional problem with this condition is that when the input triggers return to their zero levels, the resulting state of the latch is unpredictable and depends on whatever input is last to go low. Finally, Figure 7.6 shows the schematics symbol of the *SR* flip-flop.

Problem 7.1 *SR* Flip-Flop Using NAND Gates

An *SR* flip-flop can also be implemented using a cross-coupled NAND structure as shown in Figure 7.7. Derive the truth table for a such an implementation.

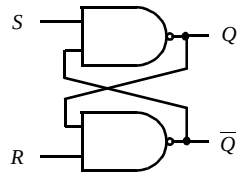


Figure 7.7 NAND-based SR flip-flop.

The SR flip-flops discussed so far are asynchronous, and do not require a clock signal. Most systems operate in a synchronous fashion with transition events referenced to a clock. One possible realization of a clocked *SR* flip-flop— a *level-sensitive* positive latch— is shown in Figure 7.8. It consists of a cross-coupled inverter pair, plus 4 extra transistors to drive the flip-flop from one state to another and to provide clocked operation. Observe that the number of transistors is identical to the implementation of Figure 7.6, but the circuit has the added feature of being clocked. The drawback of saving some transistors over a fully-complimentary CMOS implementation is that transistor sizing becomes critical in ensuring proper functionality. Consider the case where Q is high and an R pulse is applied. The combination of transistors M_4, M_7, and M_8 forms a ratioed inverter. In order to make the latch switch, we must succeed in bringing Q below the switching threshold of the inverter M_1-M_2. Once this is achieved, the positive feedback causes the flip-flop to invert states. This requirement forces us to increase the sizes of transistors M_5, M_6, M_7, and M_8.

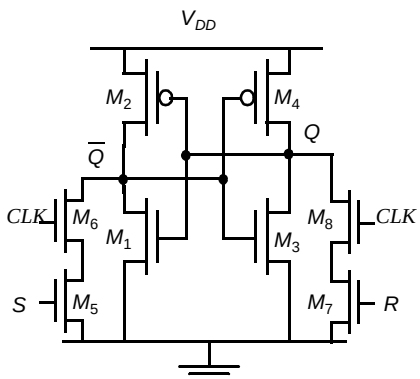


Figure 7.8 CMOS clocked *SR* flip-flop.

The presented flip-flop does not consume any static power. In steady-state, one inverter resides in the high state, while the other one is low. No static paths between V_{DD} and *GND* can exist except during switching.

Example 7.1 Transistor Sizing of Clocked SR Latch

Assume that the cross-coupled inverter pair is designed such that the inverter threshold V_M is located at $V_{DD}/2$. For a 0.25 µm CMOS technology, the following transistor sizes were selected: $(W/L)_{M1} = (W/L)_{M3} = (0.5\mu m/0.25\mu m)$, and $(W/L)_{M2} = (W/L)_{M4} = (1.5\mu m/0.25\mu m)$. Assuming $Q = 0$, we determine the minimum sizes of M_5, M_6, M_7, and M_8 to make the device switchable.

To switch the latch from the $Q = 0$ to the $Q = 1$ state, it is essential that the low level of the ratioed, pseudo-NMOS inverter (M_5-M_6)-M_2 be below the switching threshold of the inverter M_3-M_4 that equals $V_{DD}/2$. It is reasonable to assume that as long as $V_{\overline{Q}} > V_M$, V_Q equals 0, and the gate of transistor M_2 is grounded. The boundary conditions on the transistor sizes can be derived by equating the currents in the inverter for $V_{\overline{Q}} = V_{DD} / 2$, as given in Eq. (7.3) (this ignores channel length modulation). The currents are determined by the saturation current since $V_{SET} = V_{DD} = 2.5V$ and $V_M = 1.25V$. We assume that M_5 and M_6 have identical sizes and that $W/L_{5\text{-}6}$ is the effective ratio of the series-connected devices. Under this condi-

tion, the pull-down network can be modeled by a single transistor M_{56}, whose length is twice the length of the individual devices.

$$k'_n\left(\frac{W}{L}\right)_{5-6}\left((V_{DD}-V_{Tn})V_{DSATn}-\frac{V_{DSATn}^2}{2}\right) = k'_p\left(\frac{W}{L}\right)_2\left((-V_{DD}-V_{Tp})V_{DSATp}-\frac{V_{DSATp}^2}{2}\right) \quad (7.3)$$

Using the parameters for the 0.25 µm process, Eq. (7.3) results in the constraint that the effective $(W/L)_{M5\text{-}6} \geq 2.26$. This implies that the individual device ratio for M_5 or M_6 must be larger that approximately 4.5. Figure 7.9a shows the DC plot of $V_{\overline{Q}}$ as a function of the individual device sizes of M_5 and M_6. We notice that the individual device ratio of greater than 3 is sufficient to bring the $\overline{Q}$ voltage to the inverter switching threshold. The difference between the manual analysis and simulation arises due to second order effects such as DIBL and channel length modulation. Figure 7.9b, which shows the transient response for different device sizes. The plot confirms that an individual *W/L* ratio of greater than 3 is required to overpower the feedback and switch the state of the latch.

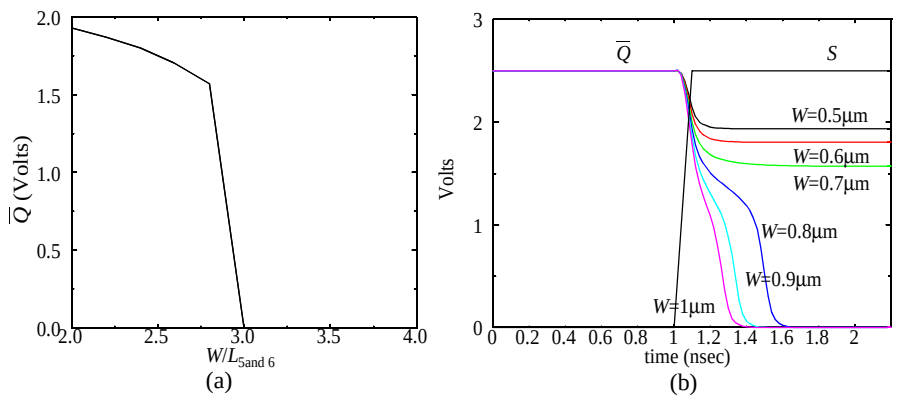


Figure 7.9 Sizing issues for *SR* flip-flop. (a) DC output voltage vs. individual pulldown device size of M_5 and M_6 with W/L_2= 1.5µm/.25µm. (b) Transient response shows that M_5 and M_6 must each have a *W/L* larger than 3 to switch the *SR* flip-flop.

The positive feedback effect makes a manual derivation of *propagation delay* of the *SR* latch quite hard. Some simplifications are therefore necessary. Consider, for instance, the latch of Figure 7.8, where Q and $\overline{Q}$ are set to 0 and 1, respectively. A pulse is applied at node S, causing the latch to toggle. In the first phase of the transient, node $\overline{Q}$ is being pulled down by transistors M_5 and M_6. Since node Q is initially low, the PMOS device M_2 is on while M_1 is off. The transient response is hence determined by the pseudo-NMOS inverter formed by (M_5-M_6) and M_2. Once $\overline{Q}$ reaches the switching threshold of the CMOS inverter M_3-M_4, this inverter reacts and the positive feedback comes into action, turning M_2 off and M_1 on. This accelerates the pulling down of node $\overline{Q}$. From this analysis, we can derive that the *propagation delay* of node $\overline{Q}$ is approximately equal to the delay of the pseudo-NMOS inverter formed by (M_5-M_6) and M_2. To obtain the delay for node Q, it is sufficient to add the delay of the complementary CMOS inverter M_3-M_4.

Example 7.2 ***Propagation Delay* of Static *SR* Flip-Flop**

The transient response of the latch in Figure 7.8, as obtained from simulation, is plotted in Figure 7.10. The devices are sized as described in Example 7.1, and a load of a single inverter

is assumed for each latch output. The flip-flop is initially in the reset state, and an *S*-pulse is applied. As we can observe, this results first in a discharging of the $\overline{Q}$ output while *Q* stays at 0. Once the switching threshold of the inverter M_3-M_4 is reached, the *Q* output starts to rise. The delay of this transient is solely determined by the M_3-M_4 inverter, which is hampered by the slow rise time at its input. From the simulation results, we can derive that $t_{p\overline{Q}}$ and t_{pQ} equal 120psec and 230psec, respectively.

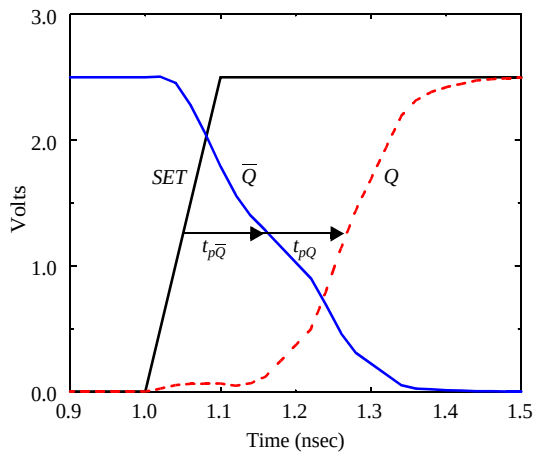


Figure 7.10 Transient response of *SR* flip-flop.

Problem 7.2 Complimentary CMOS *SR* FF

Instead of using the modified *SR* FF of Figure 7.8, it is also possible to use complementary logic to implement the clocked *SR* FF. Derive the transistor schematic (which consists of 12 transistors). This circuit is more complex, but switches faster and consumes less switching power. Explain why.

7.2.3 Multiplexer-Based Latches

There are many approaches for constructing latches. One very common technique involves the use of transmission gate multiplexers. Multiplexer based latches can provide similar functionality to the *SR* latch, but has the important added advantage that the sizing of devices only affects performance and is not critical to the functionality.

Figure 7.11 shows an implementation of static positive and negative latches based on multiplexers. For a negative latch, when the clock signal is low, the input 0 of the multiplexer is selected, and the *D* input is passed to the output. When the clock signal is high, the input 1 of the multiplexer, which connects to the output of the latch, is selected. The feedback holds the output stable while the clock signal is high. Similarly in the positive latch, the *D* input is selected when clock is high, and the output is held (using feedback) when clock is low.

A transistor level implementation of a positive latch based on multiplexers is shown in Figure 7.12. When *CLK* is high, the bottom transmission gate is *on* and the latch is

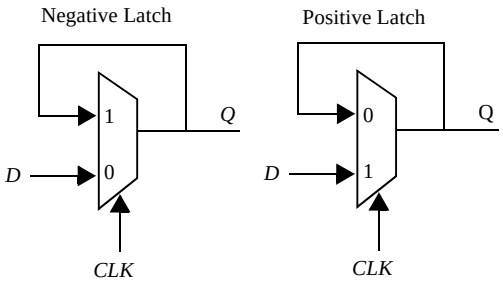


Figure 7.11 Negative and positive latches based on multiplexers.

transparent - that is, the *D* input is copied to the *Q* output. During this phase, the feedback loop is open since the top transmission gate is *off*. Unlike the *SR* FF, the feedback does not have to be overridden to write the memory and hence sizing of transistors is not critical for realizing correct functionality. The number of transistors that the clock touches is important since it has an *activity factor* of 1. This particular latch implementation is not particularly efficient from this metric as it presents a load of 4 transistors to the *CLK* signal.

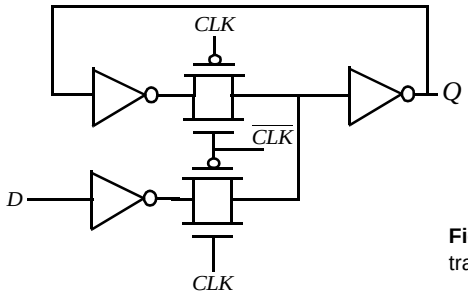
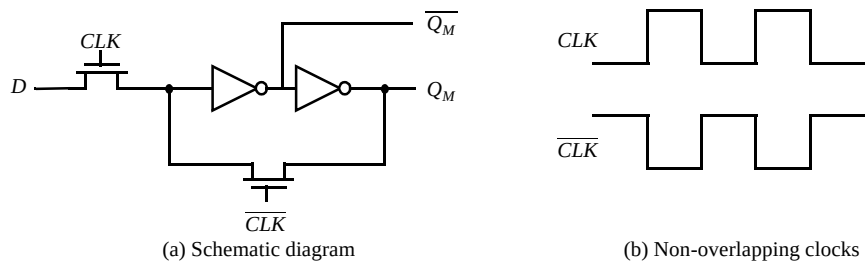


Figure 7.12 Positive latch built using transmission gates.

It is possible to reduce the clock load to two transistors by using implement multiplexers using NMOS only pass transistor as shown in Figure 7.13. The advantage of this approach is the reduced clock load of only two NMOS devices. When *CLK* is high, the latch samples the *D* input, while a low clock-signal enables the feedback-loop, and puts the latch in the hold mode. While attractive for its simplicity, the use of NMOS only pass transistors results in the passing of a degraded high voltage of V_{DD}-V_{Tn} to the input of the first inverter. This impacts both noise margin and the switching performance, especially in the case of low values of V_{DD} and high values of V_{Tn}. It also causes static power dissipation in first inverter, as already pointed out in Chapter 6. Since the maximum input-voltage to the inverter equals V_{DD}-V_{Tn}, the PMOS device of the inverter is never turned off, resulting is a static current flow.

7.2.4 Master-Slave Edge-Triggered Register

The most common approach for constructing an *edge-triggered* register is to use a *master-slave* configuration, as shown in Figure 7.14. The register consists of cascading a negative latch (master stage) with a positive latch (slave stage). A multiplexer-based latch is used in this particular implementation, although any latch could be used. On the low phase of the clock, the master stage is *transparent*, and the *D* input is passed to the master stage output,



(a) Schematic diagram (b) Non-overlapping clocks

Figure 7.13 Multiplexer-based NMOS latch using NMOS-only pass transistors.

Q_M. During this period, the slave stage is in the *hold* mode, keeping its previous value using feedback. On the rising edge of the clock, the master slave stops sampling the input, and the slave stage starts sampling. During the high phase of the clock, the slave stage samples the output of the master stage (Q_M), while the master stage remains in a *hold* mode. Since Q_M is constant during the high phase of the clock, the output Q makes only one transition per cycle. The value of Q is the value of D right before the rising edge of the clock, achieving the *positive edge-triggered* effect. A *negative edge-triggered* register can be constructed using the same principle by simply switching the order of the positive and negative latch (this is, placing the positive latch first).

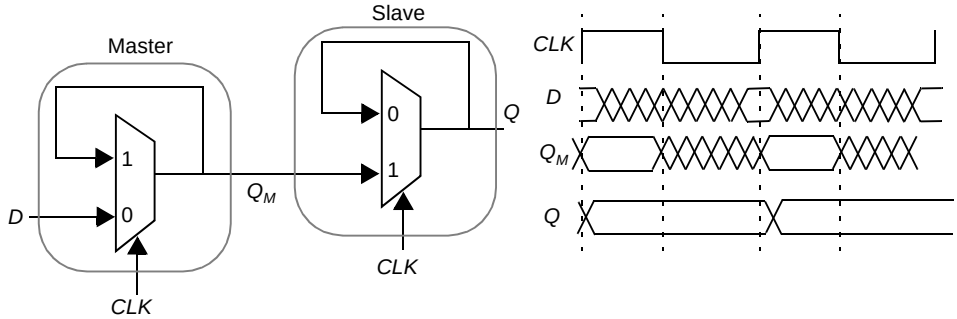


Figure 7.14 Positive edge-triggered register based on a master-slave configuration.

A complete transistor-level implementation of a the *master-slave positive edge-triggered* register is shown in Figure 7.15. The multiplexer is implemented using transmission gates as discussed in the previous section. When the clock is low ($\overline{CLK} = 1$), T_1 is *on* and T_2 is *off*, and the D input is sampled onto node Q_M. During this period, T_3 is *off* and T_4 is *on* and the cross-coupled inverters (I_5, I_6) holds the state of the slave latch. When the clock goes high, the master stage stops sampling the input and goes into a *hold* mode. T_1 is *off* and T_2 is *on*, and the cross coupled inverters I_3 and I_4 holds the state of Q_M. Also, T_3 is *on* and T_4 is *off*, and Q_M is copied to the output Q.

Problem 7.3 Optimization of the Master Slave Register

It is possible to remove the inverters I_1 and I_2 from Figure 7.3 without loss of functionality. Is there any advantage in including these inverters in the implementation?

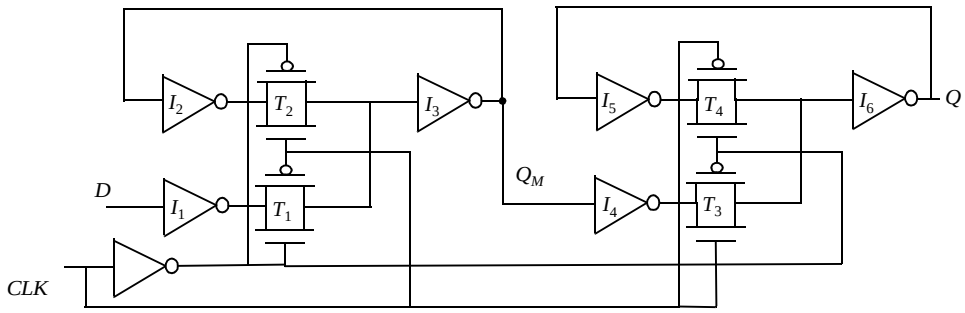


Figure 7.15 Master-slave positive edge-triggered register using multiplexers.

Timing Properties of Multiplexer-based Master-Slave Registers

Registers are characterized by three important timing parameters: the *set-up time*, the *hold time* and the *propagation delay*. It is important to understand the factors that affect these timing parameters, and develop the intuition to manually estimate them. Assume that the *propagation delay* of each inverter is t_{pd_inv}, and the *propagation delay* of the transmission gate is t_{pd_tx}. Also assume that the *contamination delay* is 0 and the inverter delay to derive $\overline{CLK}$ from *CLK* has a delay equal to 0.

The *set-up time* is the time before the rising edge of the clock that the input data *D* must become valid. Another way to ask the question is how long before the rising edge does the *D* input have to be stable such that Q_M samples the value reliably. For the transmission gate multiplexer-based register, the input *D* has to propagate through I_1, T_1, I_3 and I_2 before the rising edge of the clock. This is to ensure that the node voltages on both terminals of the transmission gate T_2 are at the same value. Otherwise, it is possible for the cross-coupled pair I_2 and I_3 to settle to an incorrect value. The *set-up time* is therefore equal to $3 * t_{pd\_inv} + t_{pd\_tx}$.

The *propagation delay* is the time for the value of Q_M to propagate to the output *Q*. Note that since we included the delay of I_2 in the *set-up time*, the output of I_4 is valid before the rising edge of clock. Therefore the delay $t_{c\text{-}q}$ is simply the delay through T_3 and I_6 ($t_{c\text{-}q} = t_{pd\_tx} + t_{pd\_inv}$).

The *hold time* represents the time that the input must be held stable after the rising edge of the clock. In this case, the transmission gate T_1 turns off when clock goes high and therefore any changes in the *D*-input after clock going high are not seen by the input. Therefore, the *hold time* is 0.

Example 7.3 Timing analysis using SPICE.

To obtain the *set-up time* of the register using SPICE, we progressively skew the input with respect to the clock edge until the circuit fails. Figure 7.16 shows the *set-up time* simulation assuming a skew of 210 psec and 200 psec. For the 210 psec case, the correct value of input *D* is sampled (in this case, the *Q* output remains at the value of V_{DD}). For a skew of 200 psec, an incorrect value propagates to the output (in this case, the *Q* output transitions to 0). Node Q_M starts to go high while the output of I_2 (the input to transmission gate T_2) starts to fall. However, the clock is enabled before the two nodes across the transmission gate (T_2) settle to the

same value and therefore, results in an incorrect value written into the master latch.The *set-up time* for this register is therefore 210 psec.

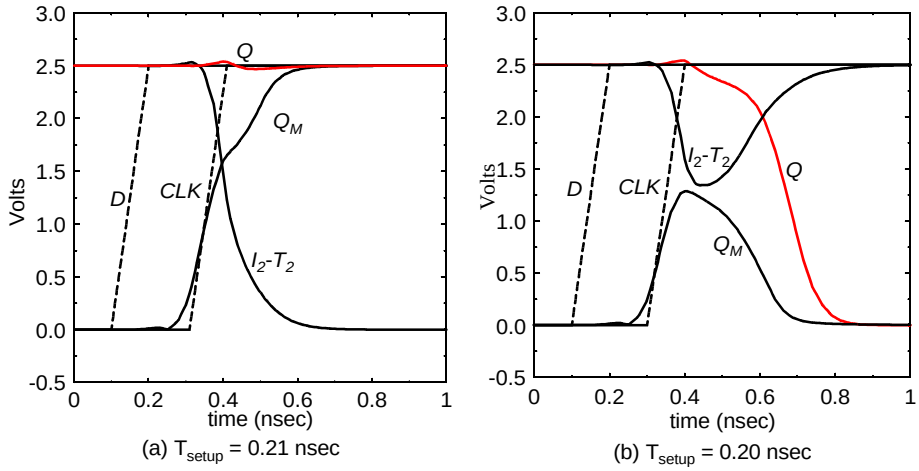


Figure 7.16 *Set-up time* simulation.

In a similar fashion, the hold time can be simulated. The *D* input edge is once again skewed relative to the clock signal till the circuit stop functioning. For this design, the *hold time* is 0 - i.e., the inputs can be changed on the clock edge. Finally, for the *propagation delay*, the inputs transition at least one *set-up time* before the rising edge of the clock and the delay is measured from the 50% point of the *CLK* edge to the 50% point of the *Q* output. From this simulation (Figure 7.17), $t_{c\text{-}q\ (lh)}$ was 160 psec and $t_{c\text{-}q(hl)}$ was 180 psec.

The drawback of the transmission gate register is the high capacitive load presented to the clock signal. The clock load per register is important, since it directly impacts the power dissipation of the clock network. Ignoring the overhead required to invert the clock signal (since the buffer inverter overhead can be amortized over multiple register bits), each register has a clock load of 8 transistors. One approach to reduce the clock load at the cost of robustness is to make the circuit ratioed. Figure 7.18 shows that the feedback transmission gate can be eliminated by directly cross coupling the inverters.

The penalty for the reduced clock load is increased design complexity. The transmission gate (T_1) and its source driver must overpower the feedback inverter (I_2) to switch the state of the cross-coupled inverter.The sizing requirements for the transmission gates

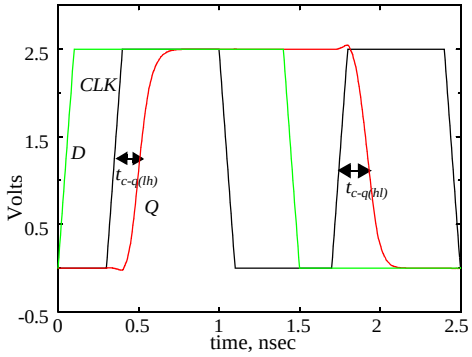


Figure 7.17 Simulation of *propagation delay*.

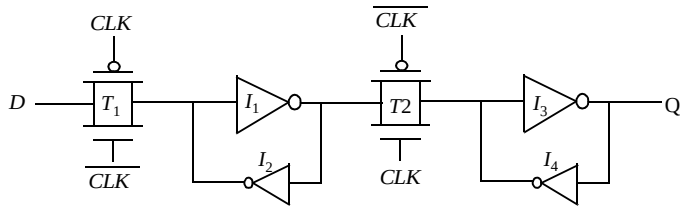


Figure 7.18 Reduced load clock load static *master-slave* register.

can be derived using a similar analysis as performed for the *SR* flip-flop. The input to the inverter I_1 must be brought below its switching threshold in order to make a transition. If minimum-sized devices are to be used in the transmission gates, it is essential that the transistors of inverter I_2 should be made even weaker. This can be accomplished by making their channel-lengths larger than minimum. Using minimum or close-to-minimum-size devices in the transmission gates is desirable to reduce the power dissipation in the latches and the clock distribution network.

Another problem with this scheme is the *reverse conduction* — this is, the second stage can affect the state of the first latch. When the slave stage is *on* (Figure 7.19), it is possible for the combination of T_2 and I_4 to influence the data stored in I_1-I_2 latch. As long as I_4 is a weak device, this is fortunately not a major problem.

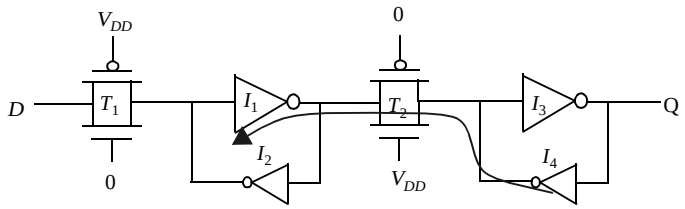
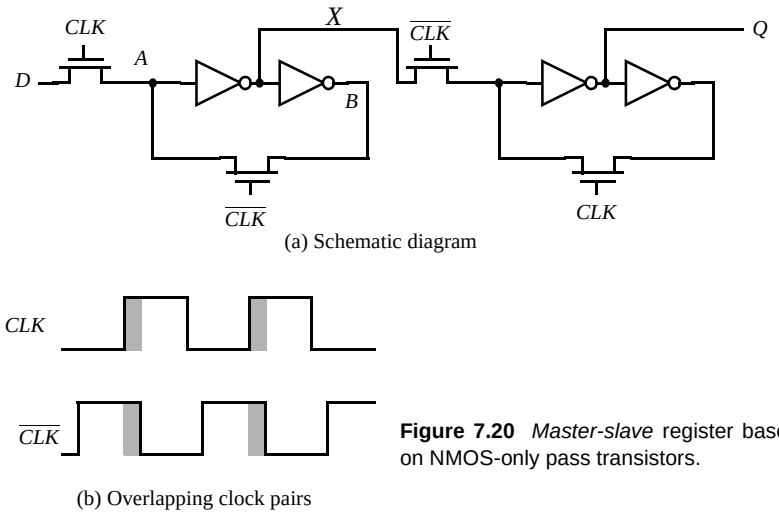


Figure 7.19 Reverse conduction possible in the transmission gate.

Non-ideal clock signals

So far, we have assumed that $\overline{CLK}$ is a perfect inversion of *CLK*, or in other words, that the delay of the generating inverter is zero. Even if this were possible, this would still not be a good assumption. Variations can exist in the wires used to route the two clock signals, or the load capacitances can vary based on data stored in the connecting latches. This effect, known as *clock skew* is a major problem, and causes the two clock signals to overlap as is shown in Figure 7.20b. *Clock-overlap* can cause two types of failures, as illustrated for the NMOS-only negative *master-slave* register of Figure 7.20a.

- When the clock goes high, the slave stage should stop sampling the master stage output and go into a *hold* mode. However, since *CLK* and $\overline{CLK}$ are both high for a short period of time (the *overlap period*), both sampling pass transistors conduct and there is a direct path from the *D* input to the *Q* output. As a result, data at the output can change on the rising edge of the clock, which is undesired for a *negative edge-triggered* register. The is know as a *race* condition in which the value of the output



(a) Schematic diagram

(b) Overlapping clock pairs

Figure 7.20 *Master-slave* register based on NMOS-only pass transistors.

Q is a function of whether the input D arrives at node X before or after the falling edge of $\overline{CLK}$. If node X is sampled in the metastable state, the output will switch to a value determined by noise in the system.

- The primary advantage of the multiplexer-based register is that the feedback loop is open during the sampling period, and therefore sizing of devices is not critical to functionality. However, if there is clock overlap between CLK and $\overline{CLK}$, node A can be driven by both D and B, resulting in an undefined state.

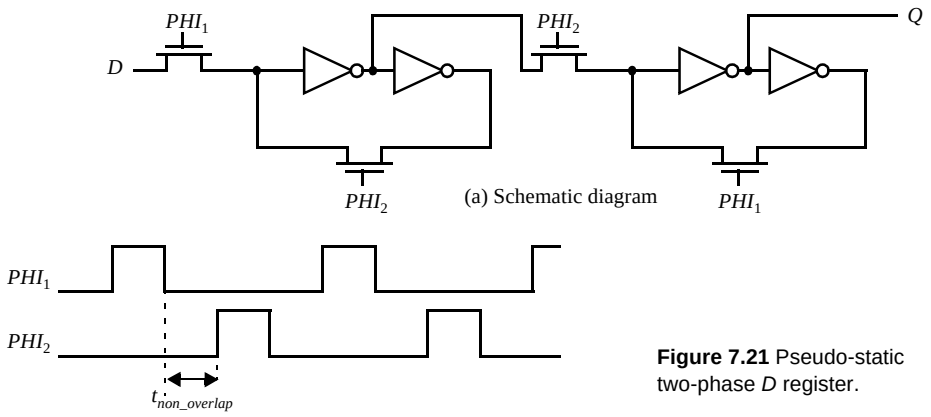
Those problems can be avoided by using two *non-overlapping clocks* PHI_1 and PHI_2 instead (Figure 7.21), and by keeping the nonoverlap time $t_{non\_overlap}$ between the clocks large enough such that no overlap occurs even in the presence of clock-routing delays. During the nonoverlap time, the *FF* is in the high-impedance state—the feedback loop is open, the loop gain is zero, and the input is disconnected. Leakage will destroy the state if this condition holds for too long a time. Hence the name *pseudo-static*: the register employs a combination of static and dynamic storage approaches depending upon the state of the clock.

Problem 7.4 Generating non-overlapping clocks

Figure 7.22 shows one possible implementation of the clock generation circuitry for generating a two-phase non-overlapping clocks. Assuming that each gate has a unit gate delay, derive the timing relationship between the input clock and the two output clocks. What is the non-overlap period? How can this period be increased if needed?

7.2.5 Low-Voltage Static Latches

The scaling of supply voltages is critical for low power operation. Unfortunately, certain latch structures don't function at reduced supply voltages. For example, without the scaling of device thresholds, NMOS only pass transistors (e.g., Figure 7.21) don't



(a) Schematic diagram

(b) Two-phase non-overlapping clocks

Figure 7.21 Pseudo-static two-phase *D* register.

scale well with supply voltage due to its inherent threshold drop. At very low power supply voltages, the input to the inverter cannot be raised above the switching threshold, resulting in incorrect evaluation. Even with the use of transmission gates, performance degrades significantly at reduced supply voltages.

Scaling to low supply voltages hence requires the use of reduced threshold devices. However, this has the negative effect of exponentially increasing the sub-threshold leakage power as discussed in Chapter 6. When the registers are constantly accessed, the leakage energy is typically insignificant compared to the switching power. However, with the use of conditional clocks, it is possible that registers are idle for extended periods and the leakage energy expended by registers can be quite significant.

Many solutions are being explored to address the problem of high leakage during idle periods. One approach for this involves the use of Multiple Threshold devices as shown in Figure 7.23 [Mutoh95]. Only the negative latch is shown here. The shaded inverters and transmission gates are implemented in low-threshold devices. The low-threshold inverters are gated using high threshold devices to eliminate leakage.

During normal mode of operation, the sleep devices are tuned *on*. When clock is low, the *D* input is sampled and propagates to the output. When clock is high, the latch is in the *hold* mode. The feedback transmission gate conducts and the cross-coupled feedback is enabled. Note there is an extra inverter, needed for storage of state when the latch is in the *sleep* state. During idle mode, the high threshold devices in series with the low threshold inverter are turned *off* (the *SLEEP* signal is high), eliminating leakage. It is assumed that clock is in the high state when the latch is in the sleep state. The feedback

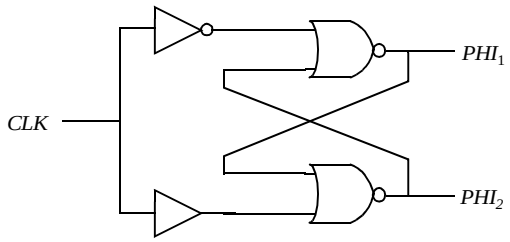


Figure 7.22 Circuitry for generating a two-phase non-overlapping clock.

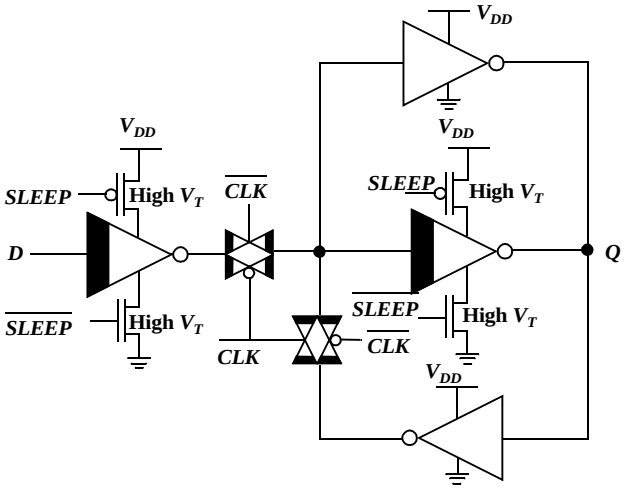


Figure 7.23 Solving the leakage problem using multiple-threshold CMOS.

low-threshold transmission gate is turned *on* and the cross-coupled high-threshold devices maintains the state of the latch.

Problem 7.5 Transistor minimization in the MTCMOS register

Unlike combinational logic, both NMOS and PMOS high-threshold devices are required to eliminate the leakage in low-threshold latches. Explain why this is the case.
Hint: Eliminate the high-V_T NMOS or high-V_T PMOS of the low-threshold inverter on the right of Figure 7.23, and investigate potential leakage paths.

7.3 Dynamic Latches and Registers

Storage in a static sequential circuit relies on the concept that a cross-coupled inverter pair produces a bistable element and can thus be used to memorize binary values. This approach has the useful property that a stored value remains valid as long as the supply voltage is applied to the circuit, hence the name *static*. The major disadvantage of the static gate, however, is its complexity. When registers are used in computational structures that are constantly clocked such as pipelined datapath, the requirement that the memory should hold state for extended periods of time can be significantly relaxed.

This results in a class of circuits based on temporary storage of charge on parasitic capacitors. The principle is exactly identical to the one used in dynamic logic — charge stored on a capacitor can be used to represent a logic signal. The absence of charge denotes a 0, while its presence stands for a stored 1. No capacitor is ideal, unfortunately, and some charge leakage is always present. A stored value can hence only be kept for a limited amount of time, typically in the range of milliseconds. If one wants to preserve signal integrity, a periodic *refresh* of its value is necessary. Hence the name *dynamic* storage. Reading the value of the stored signal from a capacitor without disrupting the charge requires the availability of a device with a high input impedance.

7.3.1 Dynamic Transmission-Gate Edge-triggered Registers

A fully dynamic *positive edge-triggered* register based on the *master-slave* concept is shown in Figure 7.24. When $CLK = 0$, the input data is sampled on storage node 1, which has an equivalent capacitance of C_1 consisting of the gate capacitance of I_1, the junction capacitance of T_1, and the overlap gate capacitance of T_1. During this period, the slave stage is in a *hold* mode, with node 2 in a high-impedance (floating) state. On the rising edge of clock, the transmission gate T_2 turns on, and the value sampled on node 1 right before the rising edge propagates to the output Q (note that node 1 is stable during the high phase of the clock since the first transmission gate is turned *off*). Node 2 now stores the inverted version of node 1. This implementation of an *edge-triggered* register is very efficient as it requires only 8 transistors. The sampling switches can be implemented using NMOS-only pass transistors, resulting in an even-simpler 6 transistor implementation. The reduced transistor count is attractive for high-performance and low-power systems.

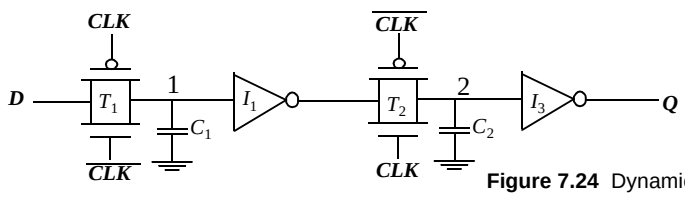


Figure 7.24 Dynamic *edge-triggered* register.

The *set-up time* of this circuit is simply the delay of the transmission gate, and corresponds to the time it takes node 1 to sample the D input. The *hold time* is approximately zero, since the transmission gate is turned *off* on the clock edge and further inputs changes are ignored. The *propagation delay* ($t_{c\text{-}q}$) is equal to two inverter delays plus the delay of the transmission gate T_2.

One important consideration for such a dynamic register is that the storage nodes (i.e., the state) has to be refreshed at periodic intervals to prevent a loss due to charge leakage, due to diode leakage as well as sub-threshold currents. In datapath circuits, the refresh rate is not an issue since the registers are periodically clocked, and the storage nodes are constantly updated.

Clock overlap is an important concern for this register. Consider the clock waveforms shown in Figure 7.25. During the 0-0 overlap period, the NMOS of T_1 and the PMOS of T_2 are simultaneously on, creating a direct path for data to flow from the D input of the register to the Q output. This is known as a *race condition*. The output Q can change on the falling edge if the overlap period is large — obviously an undesirable effect for a *positive edge-triggered* register. The same is true for the 1-1 overlap region, where an input-output path exists through the PMOS of T_1 and the NMOS of T_2. The latter case is taken care off by enforcing a *hold* time constraint. That is, the data must be stable during the high-high overlap period. The former situation (0-0 overlap) can be addressed by making sure that there is enough delay between the D input and node 2 ensuring that new data sampled by the master stage does not propagate through to the slave stage. Generally the built in single inverter delay should be sufficient and the overlap period constraint is given as:

$$t_{overlap0-0} < t_{T1} + t_{I1} + t_{T2} \tag{7.4}$$

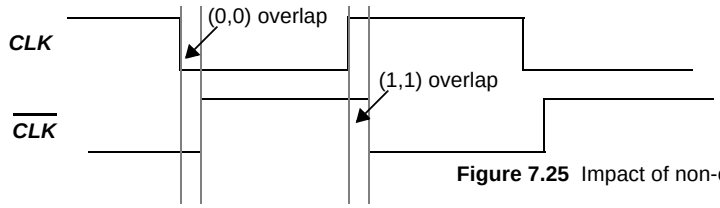


Figure 7.25 Impact of non-overlapping clocks.

Similarly, the constraint for the 1-1 overlap is given as:

$$t_{hold} > t_{overlap1-1} \tag{7.5}$$

7.3.2 C²MOS—A Clock-Skew Insensitive Approach

The C²MOS Register

Figure 7.26 shows an ingenious *positive edge-triggered* register, based on a *master-slave* concept insensitive to clock overlap. This circuit is called the C²MOS (Clocked CMOS) *register* [Suzuki73], and operates in two phases.

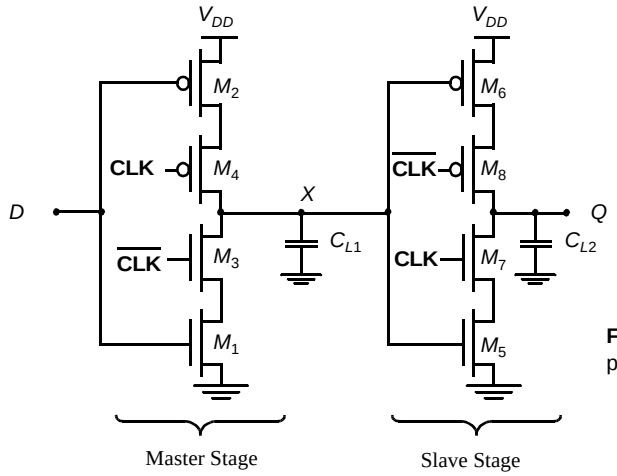


Figure 7.26 C²MOS master-slave positive edge-triggered register.

1. $CLK = 0$ ($\overline{CLK} = 1$): The first tri-state driver is turned on, and the master stage acts as an inverter sampling the inverted version of D on the internal node X. The master stage is in the *evaluation mode*. Meanwhile, the slave section is in a high-impedance mode, or in a *hold mode*. Both transistors M_7 and M_8 are off, decoupling the output from the input. The output Q retains its previous value stored on the output capacitor C_{L2}.
2. The roles are reversed when $CLK = 1$: The master stage section is in hold mode (M_3-M_4 off), while the second section evaluates (M_7-M_8 on). The value stored on C_{L1} propagates to the output node through the slave stage which acts as an inverter.

The overall circuit operates as a *positive edge-triggered master-slave* register — very similar to the transmission-gate based register presented earlier. However, there is an important difference:

> A C^2MOS register with CLK-$\overline{CLK}$ clocking is insensitive to overlap, as long as the rise and fall times of the clock edges are sufficiently small.

To prove the above statement, we examine both the (0-0) and (1-1) overlap cases (Figure 7.25). In the (0-0) overlap case, the circuit simplifies to the network shown in Figure 7.27a in which both PMOS devices are *on* during this period. The question is does any new data sampled during the overlap window propagate to the output Q. This is not desirable since data should not change on the negative edge for a *positive edge-triggered register*. Indeed new data is sampled on node X through the series PMOS devices M_2-M_4, and node X can make a 0-to-1 transition during the overlap period. However, this data cannot propagate to the output since the NMOS device M_7 is turned off. At the end of the overlap period, $\overline{CLK}$=1 and both M_7 and M_8 turn *off*, putting the slave stage is in the *hold* mode. Therefore, any new data sampled on the falling clock edge is not seen at the slave output Q, since the slave state is off till the next rising edge of the clock. As the circuit consists of a cascade of inverters, signal propagation requires one pull-up followed by a pull-down, or vice-versa, which is not feasible in the situation presented.

The (1-1) overlap case (Figure 7.27b), where both NMOS devices M_3 and M_7 are turned on, is somewhat more contentious. The question is again if new data sampled during the overlap period (right after clock goes high) propagates to the Q output. A *positive edge-triggered* register may only pass data that is presented at the input *before* the rising edge. If the D input changes during the overlap period, node X can make a 1-to-0 transition, but cannot propagate to the output. However, as soon as the overlap period is over, the PMOS M_8 is turned on and the 0 propagates to output. This effect is not desirable. The

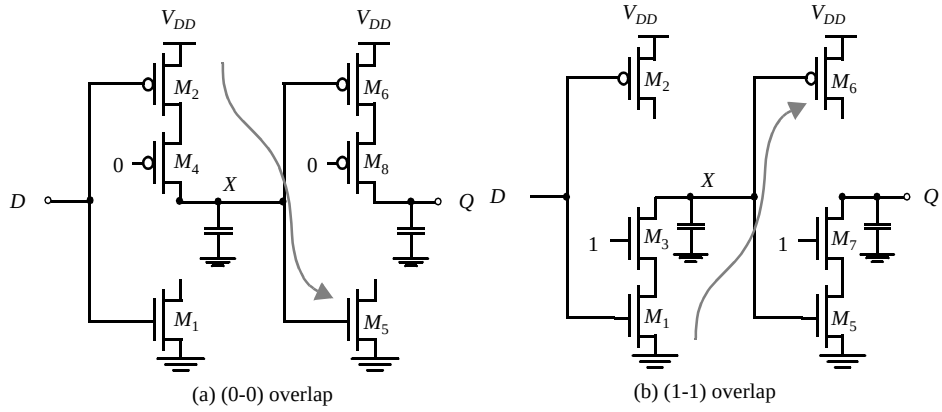


Figure 7.27 C^2MOS *D*-FF during overlap periods. No feasible signal path can exist between *In* and *D,* as illustrated by the arrows.

problem is fixed by imposing a *hold time* constraint on the input data, D, or, in other words, the data D should be stable during the overlap period.

In summary, it can be stated that the C^2MOS latch is insensitive to clock overlaps because those overlaps activate either the pull-up or the pull-down networks of the latches,

but never both of them simultaneously. If the *rise and fall times of the clock* are sufficiently slow, however, there exists a time slot where both the NMOS and PMOS transistors are conducting. This creates a path between input and output that can destroy the state of the circuit. Simulations have shown that the circuit operates correctly as long as the clock rise time (or fall time) is smaller than approximately five times the *propagation delay* of the register. This criterion is not too stringent, and is easily met in practical designs. The impact of the rise and fall times is illustrated in Figure 7.28, which plots the simulated transient response of a C^2MOS *D* FF for clock slopes of respectively 0.1 and 3 nsec. For slow clocks, the potential for a *race condition* exists.

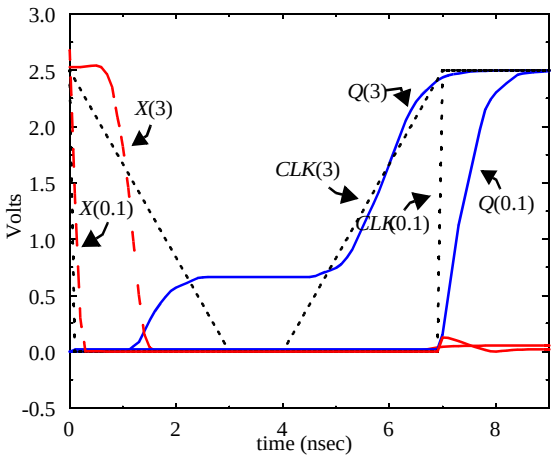


Figure 7.28 Transient response of C^2MOS FF for 0.1 nsec and 3 nsec clock rise (fall) times assuming *In* = 1.

Dual-edge Registers

So far, we have focused on *edge-triggered* registers that sample the input data on only one of the clock edges (rising or falling). It is also possible to design sequential circuits that sample the input on both edges. The advantage of this scheme is that a lower frequency clock (half of the original rate) is distributed for the same functional throughput, resulting in power savings in the clock distribution network. Figure 7.29 shows a modification of the C^2MOS register to enable sampling on both edges. It consists of two parallel *master-slave* based *edge-triggered* registers, whose outputs are multiplexed using the tri-state drivers.

When clock is high, the positive latch composed of transistors M_1-M_4 is sampling the inverted *D* input on node *X*. Node *Y* is held stable, since devices M_9 and M_{10} are turned *off*. On the falling edge of the clock, the top slave latch M_5-M_8 turns on, and drives the inverted value of *X* to the *Q* output. During the low phase, the bottom master latch (M_1, M_4, M_9, M_{10}) is turned on, sampling the inverted *D* input on node *Y*. Note that the devices M_1 and M_4 are reused, reducing the load on the *D* input. On the rising edge, the bottom slave latch conducts, and drives the inverted version of *Y* on node *Q*. Data hence changes on both edges. Note that the slave latches operate in a complementary fashion — this is, only one of them is turned on during each phase of the clock.

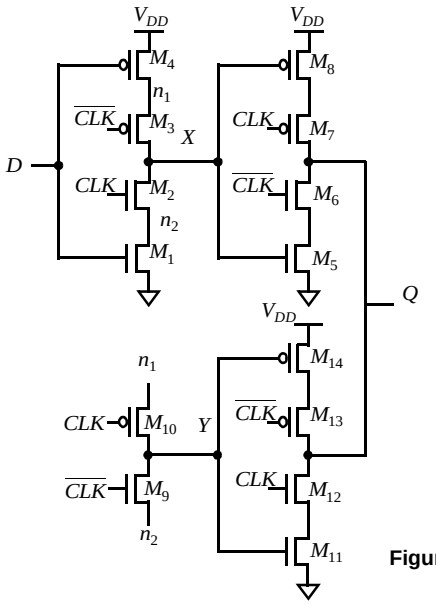


Figure 7.29 C^{2}MOS based dual-edge triggered register.

Problem 7.6 Dual-edge registers

Determine how the adoption of dual-edge registers influences the power-dissipation in the clock-distribution network.

7.3.3 True Single-Phase Clocked Register (TSPCR)

In the two-phase clocking schemes described above, care must be taken in routing the two clock signals to ensure that overlap is minimized. While the C^{2}MOS provides a skew-tolerant solution, it is possible to design registers that only use a single phase clock. The *True Single-Phase Clocked Register* (TSPCR), proposed by Yuan and Svensson, uses a **single clock** [Yuan89]. The basic single-phase positive and negative latches are shown in Figure 7.30. For the positive latch, when *CLK* is high, the latch is in the *transparent mode* and

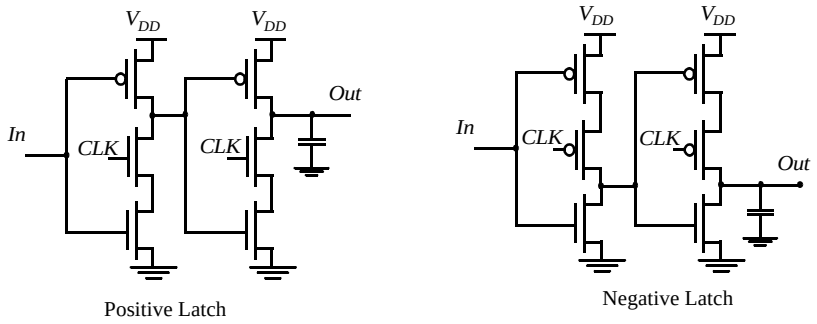
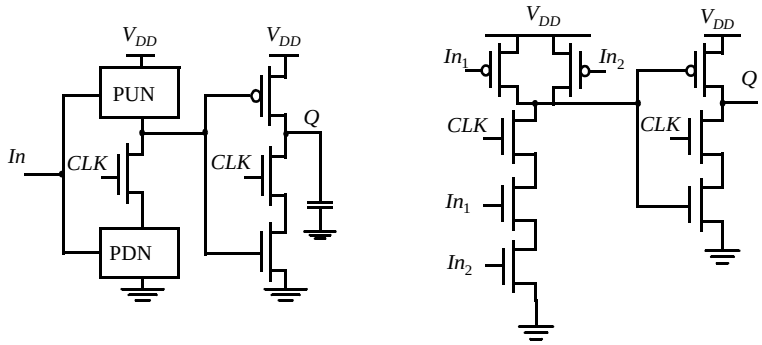


Figure 7.30 True Single Phase Latches.

corresponds to two cascaded inverters; the latch is non-inverting, and propagates the input to the output. On the other hand, when $CLK = 0$, both inverters are disabled, and the latch is in *hold-mode*. Only the pull-up networks are still active, while the pull-down circuits are deactivated. As a result of the dual-stage approach, no signal can ever propagate from the input of the latch to the output in this mode. A register can be constructed by cascading positive and negative latches. The clock load is similar to a conventional transmission gate register, or C^2MOS register. The main advantage is the use of a single clock phase. The disadvantage is the slight increase in the number of transistors — 12 transistors are required.

TSPC offers an additional advantage: the possibility of embedding logic functionality into the latches. This reduces the delay overhead associated with the latches. Figure 7.31a outlines the basic approach for embedding logic, while Figure 7.31b shows an example of a positive latch that implements the AND of In_1 and In_2 in addition to performing the latching function. While the *set-up time* of this latch has increased over the one shown in Figure 7.30, the overall performance of the digital circuit (that is, the clock period of a sequential circuit) has improved: the increase in *set-up time* is typically smaller than the delay of an AND gate. This approach of embedding logic into latches has been used extensively in the design of the EV4 DEC Alpha microprocessor [Dobberpuhl92] and many other high performance processors.



(a) Including logic into the latch (b) AND latch

Figure 7.31 Adding logic to the TSPC approach.

Example 7.4 Impact of embedding logic into latches on performance

Consider embedding an AND gate into the TSPC latch, as shown in Figure 7.31b. In a 0.25 μm, the *set-up time* of such a circuit using minimum-size devices is 140 psec. A conventional approach, composed of an AND gate followed by a positive latch has an effective *set-up time* of 600 psec (we treat the AND plus latch as a black box that performs both functions). The embedded logic approach hence results in significant performance improvements.

The TSPC latch circuits can be further reduced in complexity, as illustrated in Figure 7.32, where only the first inverter is controlled by the clock. Besides the reduced number of transistors, these circuits have the advantage that the clock load is reduced by half. On the other hand, not all node voltages in the latch experience the full logic swing. For instance, the voltage at node A (for $V_{in} = 0$ V) for the positive latch maximally equals $V_{DD} - V_{Tn}$, which results in a reduced drive for the output NMOS transistor and a loss in perfor-

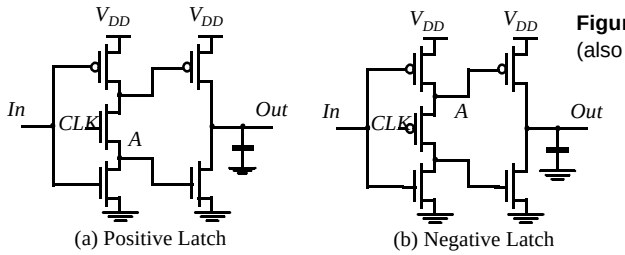


Figure 7.32 Simplified TSPC latch (also called split-output).

mance. Similarly, the voltage on node A (for $V_{in} = V_{DD}$) for the negative latch is only driven down to $|V_{Tp}|$. This also limits the amount of V_{DD} scaling possible on the latch.

Figure 7.33 shows the design of a specialized *single-phase edge-triggered register*. When $CLK = 0$, the input inverter is sampling the inverted D input on node X. The second (dynamic) inverter is in the precharge mode, with M_6 charging up node Y to V_{DD}. The third inverter is in the *hold* mode, since M_8 and M_9 are *off*. Therefore, during the low phase of the clock, the input to the final (static) inverter is holding its previous value and the output Q is stable. On the rising edge of the clock, the dynamic inverter M_4-M_6 evaluates. If X is high on the rising edge, node Y discharges. The third inverter M_7-M_8 is on during the high phase, and the node value on Y is passed to the output Q. On the positive phase of the clock, note that node X transitions to a low if the D input transitions to a high level. Therefore, the input must be kept stable till the value on node X before the rising edge of the clock propagates to Y. This represents the *hold time* of the register (note that the *hold time* less than 1 inverter delay since it takes 1 delay for the input to affect node X). The *propagation delay* of the register is essentially three inverters since the value on node X must propagate to the output Q. Finally, the *set-up time* is the time for node X to be valid, which is one inverter delay.

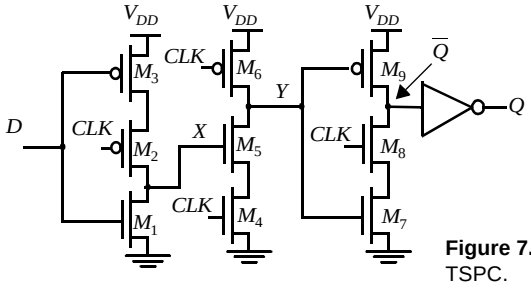
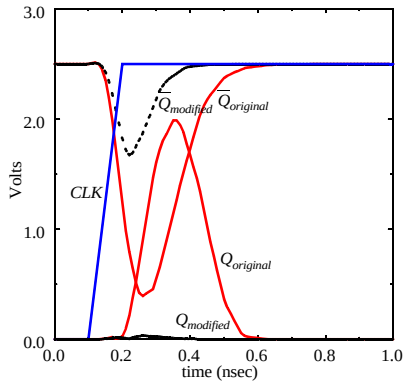


Figure 7.33 *Positive edge-triggered* register TSPC.

WARNING: Similar to the C^2MOS latch, the TSPC latch malfunctions when the *slope of the clock* is not sufficiently steep. Slow clocks cause both the NMOS and PMOS clocked transistors to be on simultaneously, resulting in undefined values of the states and race conditions. The clock slopes should therefore be carefully controlled. If necessary, local buffers must be introduced to ensure the quality of the clock signals.

Example 7.5 TSPC Edge-Triggered Register

Transistor sizing is critical for achieving correct functionality in the TSPC register. With improper sizing, glitches may occur at the output due to a *race condition* when the clock transitions from low to high. Consider the case where D is low and $\overline{Q}$=1 (Q=0). While CLK is low, Y is pre-charged high turning on M_7. When CLK transitions from low to high, nodes Y and Q start to discharge simultaneously (through M_4-M_5 and M_7-M_8, respectively). Once Y is sufficiently low, the trend on $\overline{Q}$ is reversed and the node is pulled high anew through M_9. In a sense, this chain of events is comparable to what would happen if we chain dynamic logic gates. Figure 7.34 shows the transient response of the circuit of Figure 7.33 for different sizes of devices in the final two stages.



	M_4, M_5	M_7, M_8
Original Width	0.5µm	2µm
Modified Width	1µm	1µm

Figure 7.34 Transistor sizing issues in TSPC (for the register of Figure 7.33).

This glitch may be the cause of fatal errors, as it may create unwanted events (for instance, when the output of the latch is used as a clock signal input to another register). It also reduces the *contamination delay* of the register. The problem can be corrected by resizing the relative strengths of the pull-down paths through M_4-M_5 and M_7-M_8, so that Y discharges much faster than $\overline{Q}$. This is accomplished by reducing the strength of the M_7-M_8 pulldown path, and by speeding up the M_4 -M_5 pulldown path.

7.4 Alternative Register Styles

7.4.1 Pulse Registers

Until now, we have used the *master-slave* configuration to create an *edge-triggered* register. A fundamentally different approach for constructing a register uses pulse signals. The idea is to construct a short pulse around the rising (or falling) edge of the clock. This pulse acts as the clock input to a latch (e.g., a TSPC flavor is shown in Figure 7.35a), sampling the input only in a short window. Race conditions are thus avoided by keeping the opening time (i.e, the *transparent* period) of the latch very short. The combination of the glitch-generation circuitry and the latch results in a *positive edge-triggered* register.

Figure 7.35b shows an example circuit for constructing a short intentional glitch on each rising edge of the clock [Kozu96]. When CLK = 0, node X is charged up to V_{DD} (M_N is *off* since $CLKG$ is low). On the rising edge of the clock, there is a short period of time when both inputs of the AND gate are high, causing $CLKG$ to go high. This in turn acti-

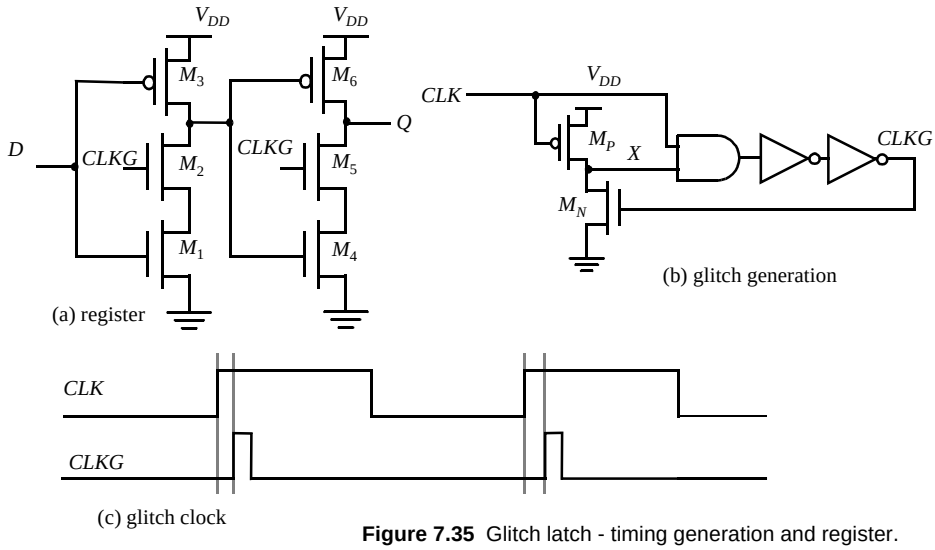


Figure 7.35 Glitch latch - timing generation and register.

vates M_N, pulling X and eventually *CLKG* low (Figure 7.35c). The length of the pulse is controlled by the delay of the AND gate and the two inverters. Note that there exists also a delay between the rising edges of the input clock (*CLK*) and the glitch clock (*CLKG*) — also equal to the delay of the AND gate and the two inverters. If every register on the chip uses the same clock generation mechanism, this sampling delay does not matter. However, process variations and load variations may cause the delays through the glitch clock circuitry to be different. This must be taken into account when performing timing verification and clock skew analysis (which is the topic of a later Chapter).

If *set-up time* and *hold time* are measured in reference to the rising edge of the glitch clock, the *set-up time* is essentially zero, the *hold time* is equal to the length of the pulse (if the *contamination delay* is zero for the gates), and the *propagation delay* ($t_{c\text{-}q}$) equals two gate delays. The advantage of the approach is the reduced clock load and the small number of transistors required. The glitch-generation circuitry can be amortized over multiple register bits. The disadvantage is a substantial increase in verification complexity. This has prevented a wide-spread use. They do however provide an alternate approach to conventional schemes, and have been adopted in some high performance processors (e.g., [Kozu96]).

Another version of the pulsed register is shown in Figure 7.36 (as used in the AMD-K6 processor [Partovi96]). When the clock is low, M_3 and M_6 are *off*, and device P_1 is turned on. Node X is precharged to V_{DD}, the output node (Q) is decoupled from X and is held at its previous state. $\overline{CLKD}$ is a delay-inverted version of *CLK*. On the rising edge of the clock, M_3 and M_6 turn *on* while devices M_1 and M_4 stay on for a short period, determined by the delay of the three inverters. During this interval, the circuit is *transparent* and the input data D is sampled by the latch. Once $\overline{CLKD}$ goes low, node X is decoupled from the D input and is either held or starts to precharge to V_{DD} by PMOS device P_2. On the falling edge of the clock, node X is held at V_{DD} and the output is held stable by the cross-coupled inverters.

Note that this circuit also uses a one-shot, but the one-shot is integrated into the reg-

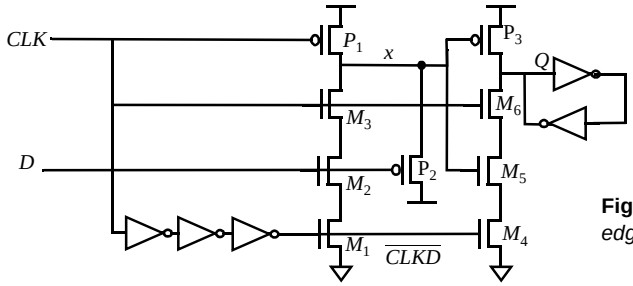


Figure 7.36 Flow-through *positive edge-triggered* register.

ister. The *transparency* period also determines the *hold time* of the register. The window must be wide enough for the input data to propagate to the *Q* output. In this particular circuit, the *set-up time* can be negative. This is the case if the transparency window is longer than the delay from input to output. This is attractive, as data can arrive at the register even after the clock goes high, which means that time is borrowed from the previous cycle.

Example 7.6 *Set-up time* of glitch register

The glitch register of Figure 7.36 is transparent during the (1-1) overlap of *CLK* and $\overline{CLKD}$. As a result, the input data can actually change after the rising edge of the clock, resulting in a negative *set-up time* (Figure 7.37). The *D*-input transitions to low after the rising edge of the clock, and transitions high before the falling edge of $\overline{CLKD}$ (this is, during the transparency period). Observe how the output follows the input. The output *Q* does go to the correct value of V_{DD} as long as the input *D* is set up correctly some time before the falling edge of $\overline{CLKD}$. When the negative *set-up time* is exploited, there can be no guarantees on the monotonic behavior of the output. That is, the output can have multiple transitions around the rising edge, and therefore, the output of the register should not be used as a clock to other registers.

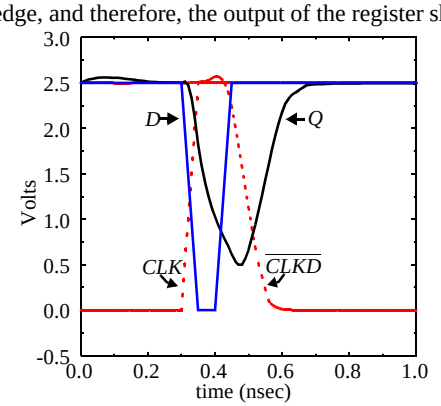


Figure 7.37 Simulation showing a negative *set-up time* for the glitch register.

Problem 7.7 Converting a glitch register to a conditional glitch register

Modify the circuit in Figure 7.36 so that it takes an additional *Enable* input. The goal is to convert the register to a conditional register which latches only when the enable signal is asserted.

7.4.2 Sense-Amplifier Based Registers

So far, we have presented two fundamental approaches towards building *edge-triggered* registers: the *master-slave* concept and the glitch technique. Figure 7.38 introduces another technique that uses a sense amplifier structure to implement an *edge-triggered* register [Montanaro96]. *Sense-amplifier* circuits accept small input signals and amplify them to generate rail-to-rail swings. As we will see, sense amplifier circuits are used extensively in memory cores and in low swing bus drivers to amplify small voltage swings present in heavily loaded wires. There are many techniques to construct these amplifiers, with the use of feedback (e.g., cross-coupled inverters) being one common approach. The circuit shown in Figure 7.38 uses a precharged front-end amplifier that samples the differential input signal on the rising edge of the clock signal. The outputs of front-end are fed into a NAND cross-coupled *SR FF* that holds the data and guarantees that the differential outputs switch only once per clock cycle. The differential inputs in this implementation don't have to have rail-to-rail swing and hence this register can be used as a receiver for a reduced swing differential bus.

The core of the front-end consists of a cross-coupled inverter (M_5-M_8), whose outputs (L_1 and L_2) are precharged using devices M_9 and M_{10} during the low phase of the clock. As a result, PMOS transistors M_7 and M_8 to be turned *off* and the NAND *FF* is holding its previous state. Transistor M_1 is similar to an evaluate switch in dynamic circuits and is turned *off* ensuring that the differential inputs don't affect the output during the low phase of the clock. On the rising edge of the clock, the evaluate transistor turns *on* and the differential input pair (M_2 and M_3) is enabled, and the difference between the input signals is amplified on the output nodes on L_1 and L_2. The cross-coupled inverter pair flips to one of its the stable states based on the value of the inputs. For example, if *IN* is 1, L_1 is pulled to 0, and L_2 remains at V_{DD}. Due to the amplifying properties of the input stage, it is *not* necessary for the input to swing all the way up to V_{DD} and enables the use of low-swing signaling on the input wires.

The shorting transistor, M_4, is used to provide a DC leakage path from either node L_3, or L_4, to ground. This is necessary to accommodate the case where the inputs change

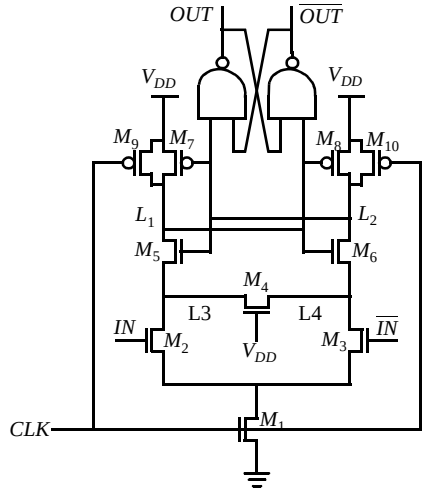


Figure 7.38 *Positive edge-triggered* register based on sense-amplifier.

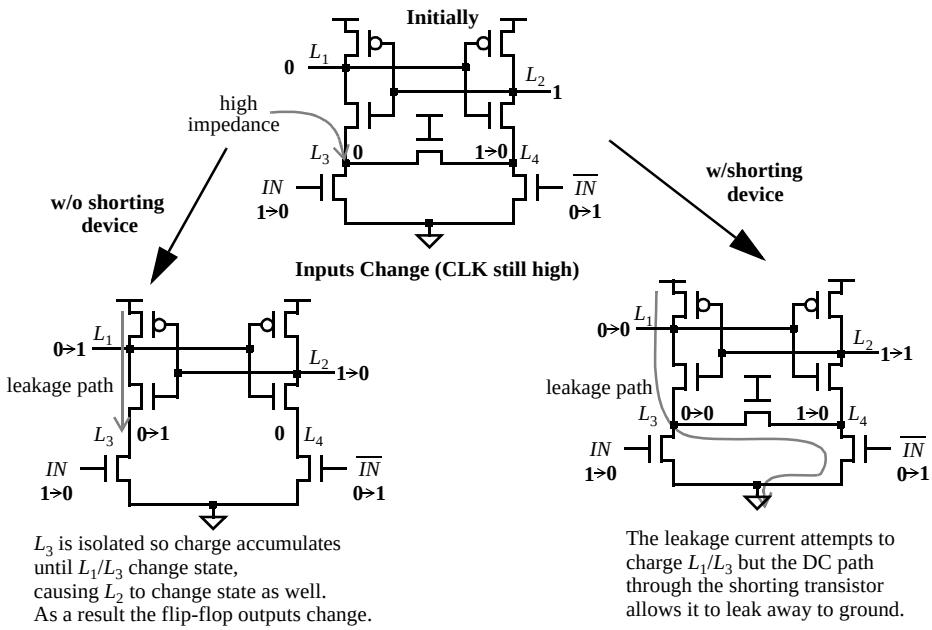


Figure 7.39 The need for the shorting transistor M_4.

their value after the positive edge of *CLK* has occurred, resulting in either L_3 or L_4 being left in a high-impedance state with a logical low voltage level stored on the node. Without the leakage path that node would be susceptible to charging by leakage currents. The latch could then actually change state prior to the next rising edge of *CLK*! This is best illustrated graphically, as shown in Figure 7.39.

7.5 Pipelining: An approach to optimize sequential circuits

Pipelining is a popular design technique often used to accelerate the operation of the datapaths in digital processors. The idea is easily explained with the example of Figure 7.40a. The goal of the presented circuit is to compute $\log(|a - b|)$, where both a and b represent streams of numbers, that is, the computation must be performed on a large set of input values. The minimal clock period T_{min} necessary to ensure correct evaluation is given as:

$$T_{min} = t_{c\text{-}q} + t_{pd,\text{logic}} + t_{su} \tag{7.6}$$

where $t_{c\text{-}q}$ and t_{su} are the *propagation delay* and the *set-up time* of the register, respectively. We assume that the registers are edge-triggered D registers. The term $t_{pd,logic}$ stands for the worst-case delay path through the combinational network, which consists of the adder, absolute value, and logarithm functions. In conventional systems (that don't push the edge of technology), the latter delay is generally much larger than the delays associated with the registers and dominates the circuit performance. Assume that each logic module has an equal propagation delay. We note that each logic module is then active for only 1/3 of the clock period (if the delay of the register is ignored). For example, the adder

unit is active during the first third of the period and remains idle—this is, it does no useful computation— during the other 2/3 of the period. Pipelining is a technique to improve the resource utilization, and increase the functional throughput. Assume that we introduce registers between the logic blocks, as shown in Figure 7.40b. This causes the computation for one set of input data to spread over a number of clock periods, as shown in Table 7.1. The result for the data set (a_1, b_1) only appears at the output after three clock-periods. At that time, the circuit has already performed parts of the computations for the next data sets, (a_2, b_2) and (a_3,b_3). The computation is performed in an assembly-line fashion, hence the name pipeline.

Table 7.1 Example of pipelined computations.

Clock Period	Adder	Absolute Value	Logarithm
1	$a_1 + b_1$		
2	$a_2 + b_2$	$\|a_1 + b_1\|$	
3	$a_3 + b_3$	$\|a_2 + b_2\|$	$\log(\|a_1 + b_1\|)$
4	$a_4 + b_4$	$\|a_3 + b_3\|$	$\log(\|a_2 + b_2\|)$
5	$a_5 + b_5$	$\|a_4 + b_4\|$	$\log(\|a_3 + b_3\|)$

The advantage of pipelined operation becomes apparent when examining the minimum clock period of the modified circuit. The combinational circuit block has been partitioned into three sections, each of which has a smaller *propagation delay* than the original function. This effectively reduces the value of the minimum allowable clock period:

$$T_{min,\text{pipe}} = t_{c\text{-}q} + \max(t_{pd,\text{add}}, t_{pd,\text{abs}}, t_{pd,\log}) \tag{7.7}$$

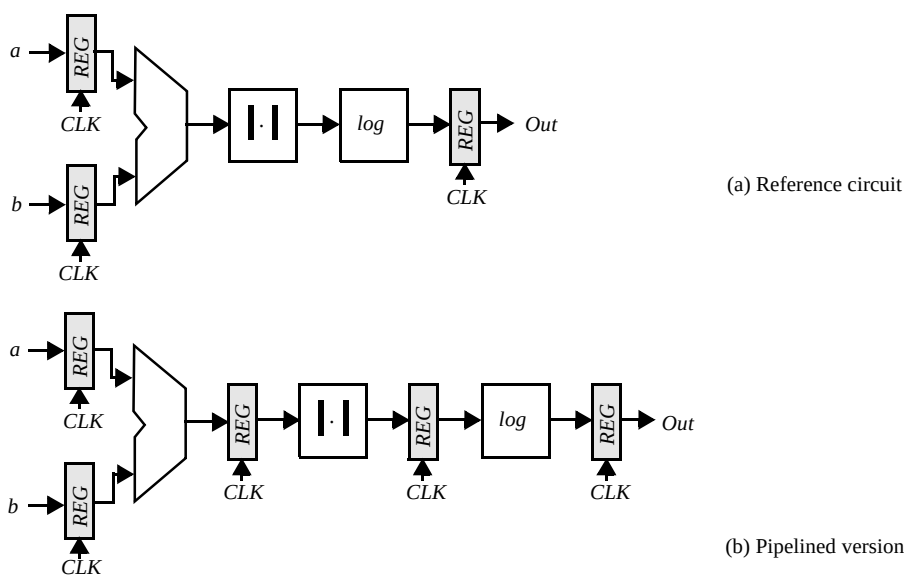


Figure 7.40 Datapath for the computation of log(|*a* + *b*|).

Suppose that all logic blocks have approximately the same *propagation delay*, and that the register overhead is small with respect to the logic delays. The pipelined network outperforms the original circuit by a factor of three under these assumptions, or $T_{min,pipe} = T_{min}/3$. The increased performance comes at the relatively small cost of two additional registers, and an increased latency.[2] This explains why pipelining is popular in the implementation of very high-performance datapaths.

7.5.1 Latch- vs. Register-Based Pipelines

Pipelined circuits can be constructed using *level-sensitive* latches instead of *edge-triggered* registers. Consider the pipelined circuit of Figure 7.41. The pipeline system is implemented based on pass-transistor-based *positive and negative* latches instead of *edge-triggered* registers. That is, logic is introduced between the master and slave latches of a *master-slave* system. In the following discussion, we use without loss of generality the CLK-$\overline{CLK}$ notation to denote a two-phase clock system. Latch-based systems give significantly more flexibility in implementing a pipelined system, and often offers higher performance. When the clocks CLK and $\overline{CLK}$ are non-overlapping, correct pipeline operation is obtained. Input data is sampled on C_1 at the negative edge of CLK and the computation of

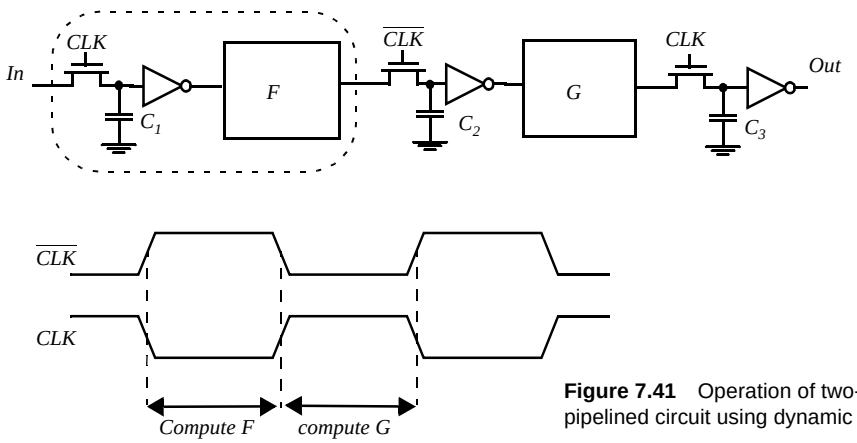


Figure 7.41 Operation of two-phase pipelined circuit using dynamic registers.

logic block F starts; the result of the logic block F is stored on C_2 on the falling edge of $\overline{CLK}$, and the computation of logic block G starts. The non-overlapping of the clocks ensures correct operation. The value stored on C_2 at the end of the CLK low phase is the result of passing the previous input (stored on the falling edge of CLK on C_1) through the logic function F. When overlap exists between CLK and $\overline{CLK}$, the next input is already being applied to F, and its effect might propagate to C_2 before $\overline{CLK}$ goes low (assuming that the *contamination delay* of F is small). In other words, a *race* develops between the previous input and the current one. Which value wins depends upon the logic function F, the overlap time, and the value of the inputs since the *propagation delay* is often a func-

[2] Latency is defined here as the number of clock cycles it takes for the data to propagate from the input to the output. For the example at hand, pipelining increases the latency from 1 to 3. An increased latency is in general acceptable, but can cause a global performance degradation if not treated with care.

tion of the applied inputs. The latter factor makes the detection and elimination of race conditions non-trivial.

7.5.2 NORA-CMOS—A Logic Style for Pipelined Structures

The latch-based pipeline circuit can also be implemented using C^2MOS latches, as shown in Figure 7.42. The operation is similar to the one discussed above. This topology has one additional, important property:

> A C^2MOS-based pipelined circuit is race-free as long as all the logic functions *F* (implemented using static logic) between the latches are non-inverting.

The reasoning for the above argument is similar to the argument made in the construction of a C^2MOS register. During a (0-0) overlap between *CLK* and $\overline{CLK}$, all C^2MOS latches, simplify to pure pull-up networks (see Figure 7.27). The only way a signal can

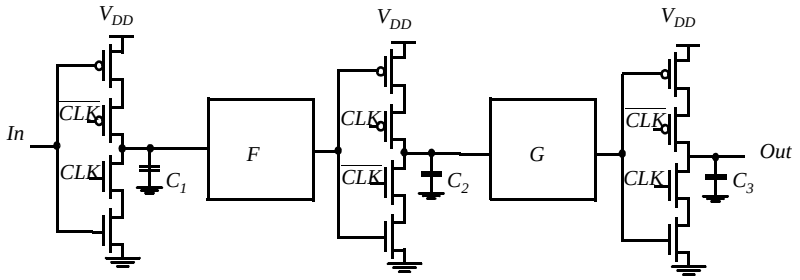


Figure 7.42 Pipelined datapath using C^2MOS latches.

race from stage to stage under this condition is when the logic function *F* is inverting, as illustrated in Figure 7.43, where *F* is replaced by a single, static CMOS inverter. Similar considerations are valid for the (1-1) overlap.

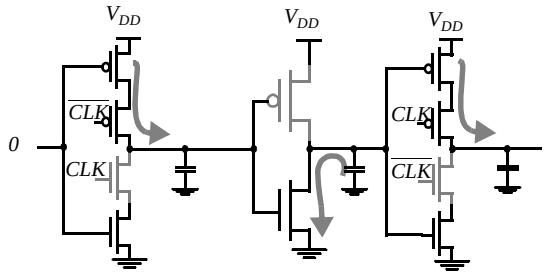
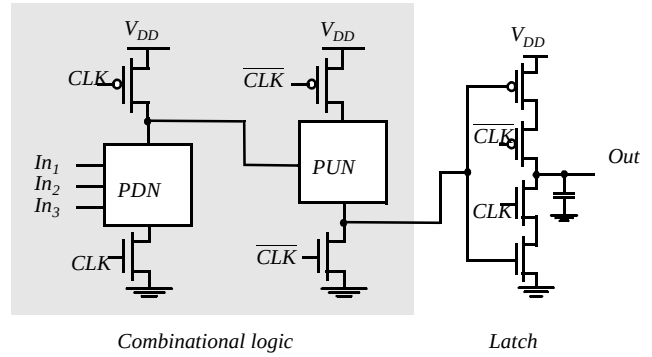


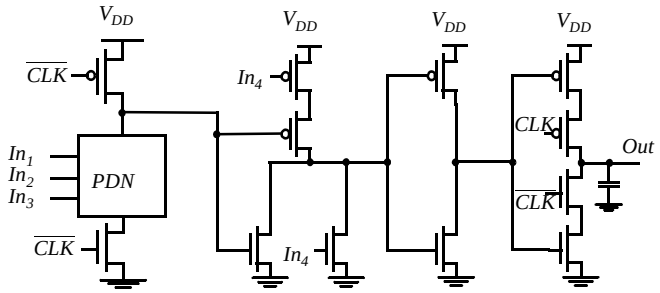
Figure 7.43 Potential race condition during (0-0) overlap in C^2MOS-based design.

Based on this concept, a logic circuit style called *NORA-CMOS* was conceived [Goncalves83]. It combines C^2MOS pipeline registers and *NORA* dynamic logic function blocks. Each module consists of a block of combinational logic that can be a mixture of static and dynamic logic, followed by a C^2MOS latch. Logic and latch are clocked in such a way that both are simultaneously in either evaluation, or hold (precharge) mode. A block that is in evaluation during *CLK* = 1 is called a *CLK-module*, while the inverse is called a

$\overline{CLK}$*-module*. Examples of both classes are shown in Figure 7.44 a and b, respectively. The operation modes of the modules are summarized in Table 7.2.



(a) CLK-module



(b) $\overline{CLK}$-module

Figure 7.44 Examples of NORA CMOS Modules.

Table 7.2 Operation modes for NORA logic modules.

	CLK **block**		*CLK* **block**	
	Logic	**Latch**	**Logic**	**Latch**
CLK = 0	Precharge	Hold	Evaluate	Evaluate
CLK = 1	Evaluate	Evaluate	Precharge	Hold

A NORA datapath consists of a chain of alternating *CLK* and $\overline{CLK}$ modules. While one class of modules is precharging with its output latch in hold mode, preserving the previous output value, the other class is evaluating. Data is passed in a pipelined fashion from module to module.

NORA offers designers a wide range of design choices. Dynamic and static logic can be mixed freely, and both CLK_p and CLK_n dynamic blocks can be used in cascaded or in pipelined form. With this freedom of design, extra inverter stages, as required in DOMINO-CMOS, are most often avoided.

Design Rules

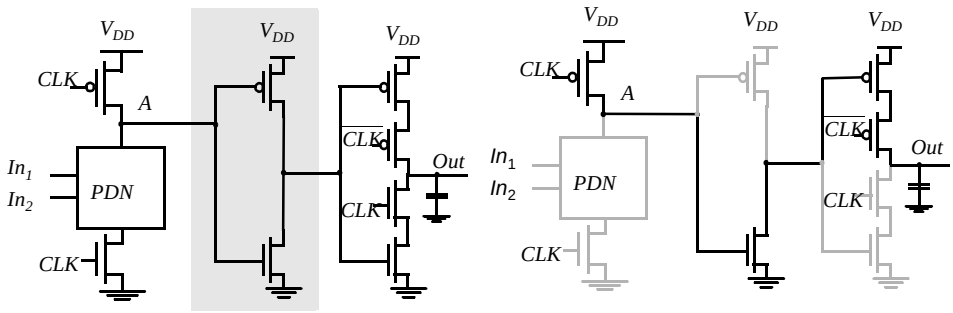
In order to ensure correct operation, two important rules should always be followed:

- **The dynamic-logic rule:** Inputs to a dynamic CLK_n (CLK_p) block are only allowed to make a single $0 \rightarrow 1$ ($1 \rightarrow 0$) transition during the evaluation period (Chapter 6).
- **The C²MOS rule:** In order to avoid races, the number of static inversions between C²MOS latches should be even.

The presence of dynamic logic circuits requires the introduction of some extensions to the latter rule. Consider the situation pictured in Figure 7.45a. During precharge (CLK = 0), the output register of the module has to be in hold mode, isolating the output node from the internal events in the module. Assume now that a (0-0) overlap occurs. Node A gets precharged to V_{DD}, while the latch simplifies to a pull-up network (Figure 7.45b). It can be observed that under those circumstances the output node charges to V_{DD}, and the stored value is erased! This malfunctioning is caused by the fact that the number of static inversions between the last dynamic node in the module and the latch is odd, which creates an active path between the precharged node and the output. This translates into the following rule: The number of static inversions between the last dynamic block in a logic function and the C²MOS latch should be *even.* This and similar considerations lead to a reformulated C²MOS rule [Goncalvez83].

Revised C²MOS Rule

- The number of static inversions between C²MOS latches should be even (in the absence of dynamic nodes); if dynamic nodes are present, the number of static inverters between



(a) Circuit with odd number of static inversions between dynamic logic stage and register

(b) Same circuit during (0-0) clock overlap.

Figure 7.45 Extended C²MOS rules.

a latch and a dynamic gate in the logic block should be even. The number of static inversions between the last dynamic gate in a logic block and the latch should be even as well.

Adhering to the above rules is not always trivial and requires a careful analysis of the logic equations to be implemented. This often makes the design of an operational NORA-CMOS structure cumbersome. Its use should only be considered when maximum circuit performance is a must.

7.6 Non-Bistable Sequential Circuits

In the preceding sections, we have focused on one single type of sequential element, this is the latch (and its sibling the register). The most important property of such a circuit is that it has two stable states, and is hence called *bistable*. The bistable element is not the only sequential circuit of interest. Other regenerative circuits can be catalogued as *astable* and *monostable*. The former act as oscillators and can, for instance, be used for on-chip clock generation. The latter serve as pulse generators, also called *one-shot circuits.* Another interesting regenerative circuit is the Schmitt trigger. This component has the useful property of showing hysteresis in its dc characteristics—its switching threshold is variable and depends upon the direction of the transition (low-to-high or high-to-low). This peculiar feature can come in handy in noisy environments.

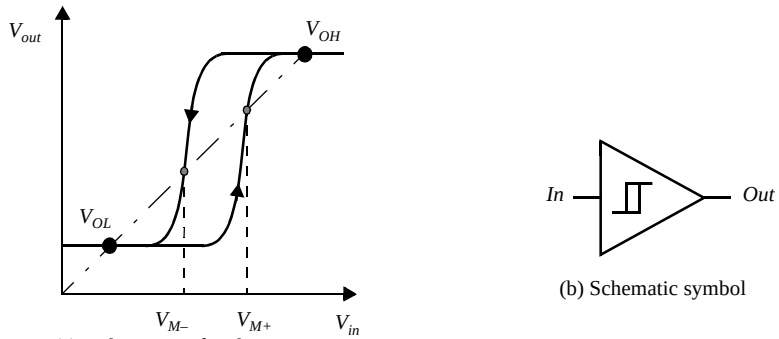
7.6.1 The Schmitt Trigger

Definition

A Schmitt trigger [Schmitt38] is a device with two important properties:

1. It responds to a slowly changing input waveform with a *fast transition time at the output*.
2. The voltage-transfer characteristic of the device displays *different switching thresholds for positive- and negative-going input signals*. This is demonstrated in Figure 7.46, where a typical voltage-transfer characteristic of the Schmitt trigger is shown (and its schematics symbol). The switching thresholds for the low-to-high and high-to-low transitions are called V_{M+} and V_{M-}, respectively. The *hysteresis voltage* is defined as the difference between the two.

One of the main uses of the Schmitt trigger is to turn a noisy or slowly varying input signal into a clean digital output signal. This is illustrated in Figure 7.47. Notice how the hysteresis suppresses the ringing on the signal. At the same time, the fast low-to-high (and high-to-low) transitions of the output signal should be observed. For instance, steep signal slopes are beneficial in reducing power consumption by suppressing direct-path currents. The “secret” behind the Schmitt trigger concept is the use of positive feedback.



(a) Voltage-transfer characteristic

(b) Schematic symbol

Figure 7.46 Non-inverting Schmitt trigger.

CMOS Implementation

One possible CMOS implementation of the Schmitt trigger is shown in Figure 7.48. The idea behind this circuit is that the switching threshold of a CMOS inverter is determined by the (k_n/k_p) ratio between the NMOS and PMOS transistors. Increasing the ratio results in a reduction of the threshold, while decreasing it results in an increase in $V_{M.}$ Adapting the ratio depending upon the direction of the transition results in a shift in the switching threshold and a hysteresis effect. This adaptation is achieved with the aid of feedback.

Suppose that V_{in} is initially equal to 0, so that V_{out} = 0 as well. The feedback loop biases the PMOS transistor M_4 in the conductive mode while M_3 is off. The input signal effectively connects to an inverter consisting of two PMOS transistors in parallel (M_2 and M_4) as a pull-up network, and a single NMOS transistor (M_1) in the pull-down chain. This modifies the effective transistor ratio of the inverter to $k_{M1}/(k_{M2}+k_{M4})$, which moves the switching threshold upwards.

Once the inverter switches, the feedback loop turns off M_4, and the NMOS device M_3 is activated. This extra pull-down device speeds up the transition and produces a clean output signal with steep slopes.

A similar behavior can be observed for the high-to-low transition. In this case, the pull-down network originally consists of M_1 and M_3 in parallel, while the pull-up network is formed by M_2. This reduces the value of the switching threshold to V_{M-}.

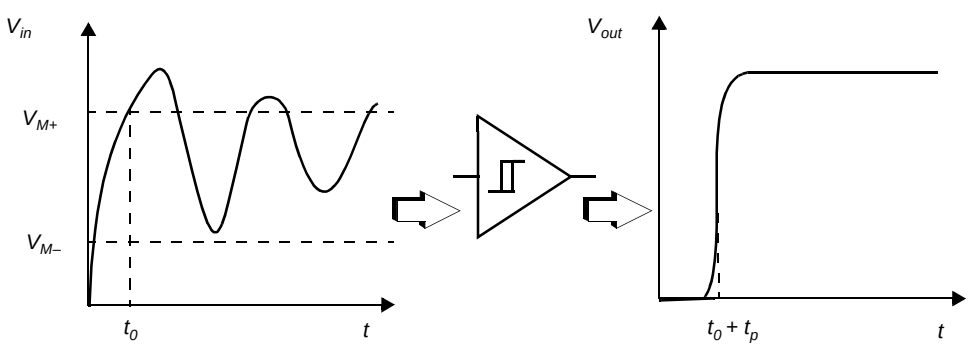


Figure 7.47 Noise suppression using a Schmitt trigger.

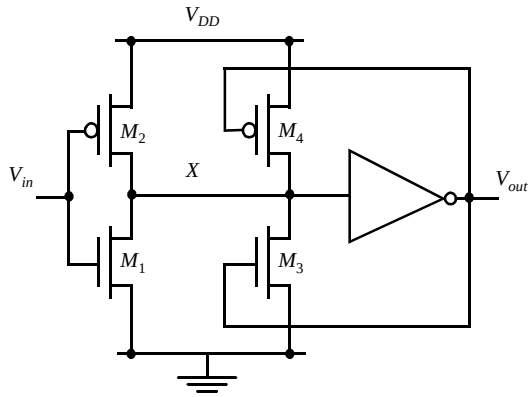
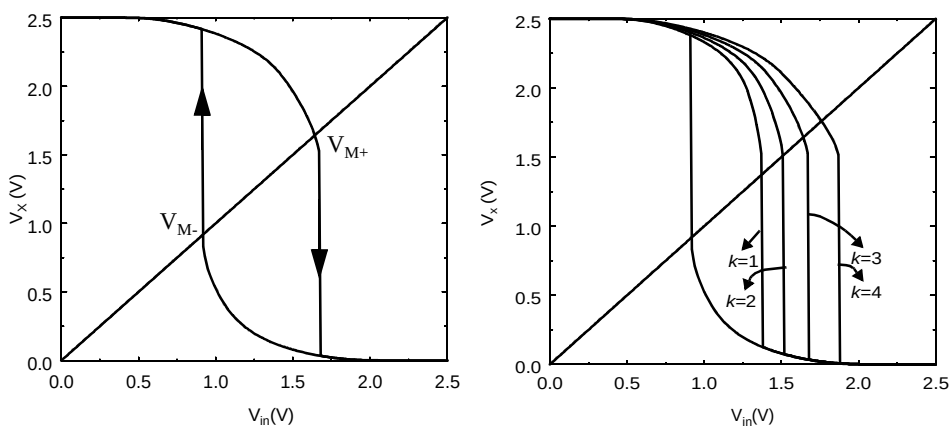


Figure 7.48 CMOS Schmitt trigger.

Example 7.7 CMOS Schmitt Trigger

Consider the schmitt trigger with the following device sizes. Devices M_1 and M_2 are 1μm/0.25μm, and 3μm/0.25μm, respectively. The inverter is sized such that the switching threshold is around $V_{DD}/2$ (= 1.25 V). Figure 7.49a shows the simulation of the Schmitt trigger assuming that devices M_3 and M_4 are 0.5μm/0.25μm and 1.5μm/0.25μm, respectively. As apparent from the plot, the circuit exhibits hysteresis. The high-to-low switching point (V_{M-} = 0.9 V) is lower than $V_{DD}/2$, while the low-to-high switching threshold (V_{M+} = 1.6 V) is larger than $V_{DD}/2$.



(a) Voltage-transfer characteristics with hysteresis.

(b) The effect of varying the ratio of the PMOS device M_4. The width is k * 0.5μm.

Figure 7.49 Schmitt trigger simulations.

It is possible to shift the switching point by changing the sizes of M_3 and M_4. For example, to modify the low-to-high transition, we need to vary the PMOS device. The high-to-low threshold is kept constant by keeping the device width of M_3 at 0.5 μm. The device width of M_4 is varied as k * 0.5μm. Figure 7.49b demonstrates how the switching threshold increases with raising values of k.

Problem 7.8 An Alternative CMOS Schmitt Trigger

Another CMOS Schmitt trigger is shown in Figure 7.50. Discuss the operation of the gate, and derive expressions for V_{M-} and V_{M+}.

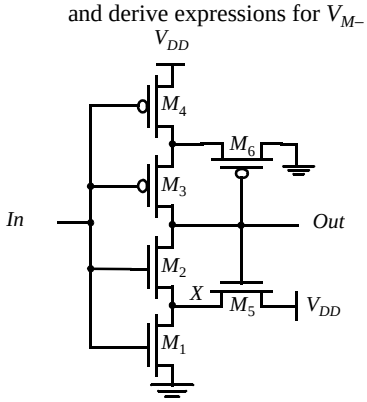


Figure 7.50 Alternate CMOS Schmitt trigger.

7.6.2 Monostable Sequential Circuits

A monostable element is a circuit that generates *a pulse of a predetermined width* every time the quiescent circuit is triggered by a pulse or transition event. It is called *monostable* because it has only one stable state (the quiescent one). A trigger event, which is either a signal transition or a pulse, causes the circuit to go temporarily into another quasi-stable state. This means that it eventually returns to its original state after a time period determined by the circuit parameters. This circuit, also called a *one-shot*, is useful in generating pulses of a known length. This functionality is required in a wide range of applications. We have already seen the use of a one-shot in the construction of glitch registers. Another notorious example is the *address transition detection* (ATD) circuit, used for the timing generation in static memories. This circuit detects a change in a signal, or group of signals, such as the address or data bus, and produces a pulse to initialize the subsequent circuitry.

The most common approach to the implementation of one-shots is the use of a simple delay element to control the duration of the pulse. The concept is illustrated in Figure 7.51. In the quiescent state, both inputs to the XOR are identical, and the output is low. A transition on the input causes the XOR inputs to differ temporarily and the output to go high. After a delay t_d (of the delay element), this disruption is removed, and the output goes low again. A pulse of length t_d is created. The delay circuit can be realized in many different ways, such as an *RC*-network or a chain of basic gates.

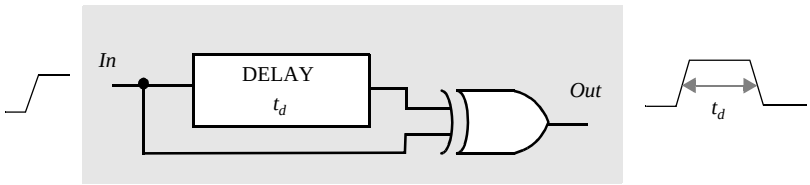


Figure 7.51 Transition-triggered one-shot.

7.6.3 Astable Circuits

An astable circuit has no stable states. The output oscillates back and forth between two quasi-stable states with a period determined by the circuit topology and parameters (delay, power supply, etc.). One of the main applications of oscillators is the on-chip generation of clock signals. This application is discussed in detail in a later chapter (on timing).

The ring oscillator is a simple, example of an astable circuit. It consists of an odd number of inverters connected in a circular chain. Due to the odd number of inversions, no stable operation point exists, and the circuit oscillates with a period equal to $2 \times t_p \times N$, with N the number of inverters in the chain and t_p the *propagation delay* of each inverter.

Example 7.8 Ring oscillator

The simulated response of a ring oscillator with five stages is shown in Figure 7.52 (all gates use minimum-size devices). The observed oscillation period approximately equals 0.5 nsec, which corresponds to a gate *propagation delay* of 50 psec. By tapping the chain at various points, different phases of the oscillating waveform are obtained (phases 1, 3, and 5 are displayed in the plot). A wide range of clock signals with different duty-cycles and phases can be derived from those elementary signals using simple logic operations.

The ring oscillator composed of cascaded inverters produces a waveform with a fixed oscillating frequency determined by the delay of an inverter in the CMOS process. In many applications, it is necessary to control the frequency of the oscillator. An example of such a circuit is the *voltage-controlled oscillator (VCO)*, whose oscillation frequency is a function (typically non-linear) of a control voltage. The standard ring oscillator can be modified into a *VCO* by replacing the standard inverter with a *current-starved* inverter as shown in Figure 7.53 [Jeong87]. The mechanism for controlling the delay of each inverter is to limit the current available to discharge the load capacitance of the gate.

In this modified inverter circuit, the maximal discharge current of the inverter is limited by adding an extra series device. Note that the low-to-high transition on the inverter can also be controlled by adding a PMOS device in series with M_2. The added NMOS transistor M_3, is controlled by an analog control voltage V_{cntl}, which determines the available discharge current. Lowering V_{cntl} reduces the discharge current and, hence, increases t_{pHL}. The ability to alter the *propagation delay* per stage allows us to control the frequency of the ring structure. The control voltage is generally set using feedback techniques. Under low operating current levels, the current-starved inverter suffers from slow fall times at its

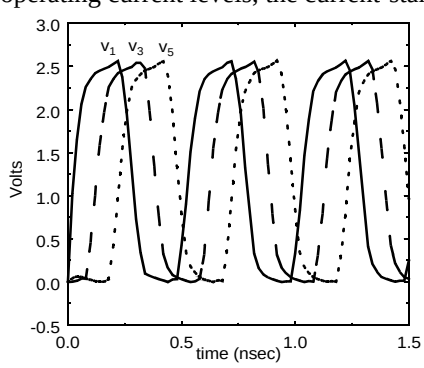


Figure 7.52 Simulated waveforms of five-stage ring oscillator. The outputs of stages 1, 3, and 5 are shown.

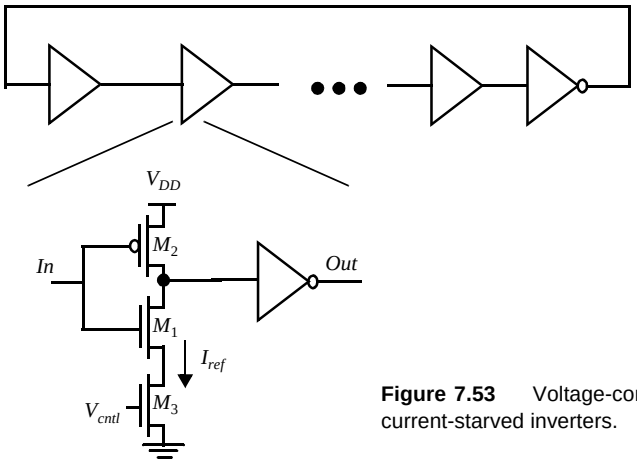


Figure 7.53 Voltage-controlled oscillator based on current-starved inverters.

output. This can result in significant short-circuit current. This is resolved by feeding its output into a CMOS inverter or better yet a Schmitt trigger. An extra inverter is needed at the end to ensure that the structure oscillates.

Example 7.9 Current-Starved Inverter Simulation

Figure 7.54 show the simulated delay of the current-starved inverter as a function of the control voltage V_{cntl}. The delay of the inverter can be varied over a large range. When the control voltage is smaller than the threshold, the device enters the sub-threshold region. This results in large variations of the *propagation delay*, as the drive current is exponentially dependent on the drive voltage. When operating in this region, the delay is very sensitive to variations in the control voltage, and, hence, to noise.

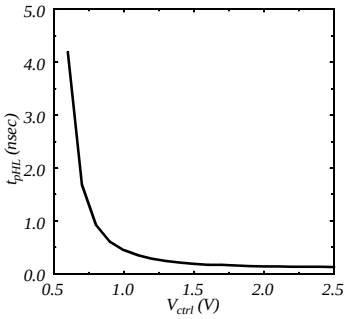
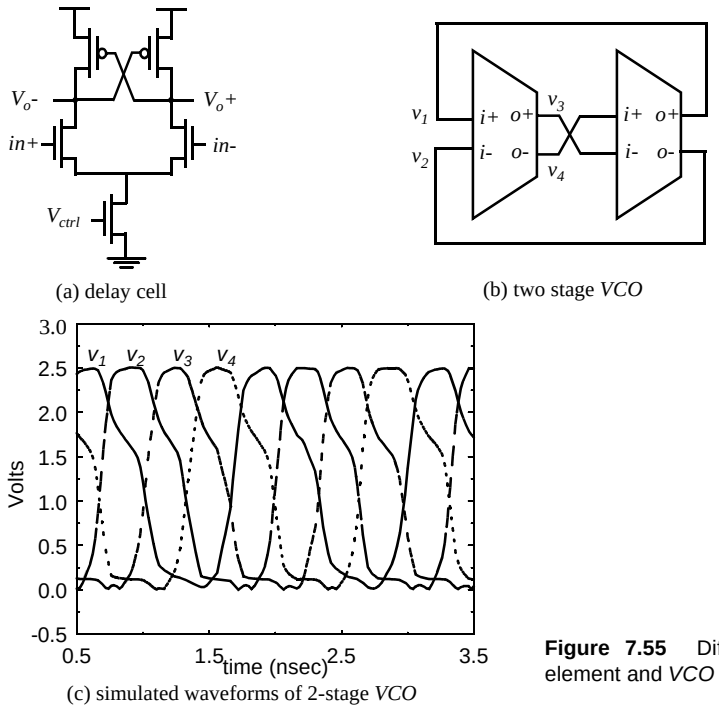


Figure 7.54 t_{pHL} of current-starved inverter as a function of the control voltage.

Another approach to implement the delay element is to use a differential element as shown in Figure 7.55a. Since the delay cell provides both inverting and non-inverting outputs, an oscillator with an even number of stages can be implemented. Figure 7.55b shows a two-stage differential *VCO*, where the feedback loop provides 180° phase shift through two gate delays, one non-inverting and the other inverting, therefore forming an oscillation. The simulated waveforms of this two stage *VCO* are shown in Figure 7.55c. The in-phase and quadrature phase outputs are available simultaneously. The differential type

VCO has better immunity to common mode noise (for example, supply noise) compared to the common ring oscillator. However, it consumes more power due to the increased complexity, and the static current.



(a) delay cell

(b) two stage *VCO*

(c) simulated waveforms of 2-stage *VCO*

Figure 7.55 Differential delay element and *VCO* topology.

7.7 Perspective: Choosing a Clocking Strategy

A crucial decision that must be made in the earliest phases of a chip design is to select the appropriate clocking methodology. The reliable synchronization of the various operations occurring in a complex circuit is one of the most intriguing challenges facing the digital designer of the next decade. Choosing the right clocking scheme affects the functionality, speed and power of a circuit.

A number of widely-used clocking schemes were introduced in this chapter. The most robust and conceptually simple scheme is the two-phase *master-slave* design. The predominant approach is use the multiplexer-based register, and to generate the two clock phases locally by simply inverting the clock. More exotic schemes such as the glitch register are also used in practice. However, these schemes require significant hand tuning and must only be used in specific situations. An example of such is the need for a negative *set-up time* to cope with clock skew.

The general trend in high-performance CMOS VLSI design is therefore to *use simple clocking schemes*, even at the expense of performance. Most automated design methodologies such as standard cell employ a single-phase, *edge-triggered* approach, based on static flip-flops. But the tendency towards simpler clocking approaches is also apparent in

high-performance designs such as microprocessors. The use of latches between logic is also very common to improve circuit performance.

7.8 Summary

This chapter has explored the subject of sequential digital circuits. The following topics were discussed:

- The cross-coupling of two inverters creates a *bistable* circuit, called a *flip-flop*. A third potential operation point turns out to be metastable; that is, any diversion from this bias point causes the flip-flop to converge to one of the stable states.
- A latch is a *level-sensitive* memory element that samples data on one phase and holds data on the other phase. A register (sometime also called a *flip-flop*) on the other hand samples the data on the rising or falling edge. A register has three important parameter *the set-up time, the hold time, and the propagation delay*. These parameters must be carefully optimized since they may account for a significant portion of the clock period.
- Registers can be *static* or *dynamic*. A static register holds state as long as the power supply is turned on. It is ideal for memory that is accessed infrequently (e.g., reconfiguration registers or control information). Dynamic memory is based on temporary charge store on capacitors. The primary advantage is the reduced complexity and higher performance/lower power. However, charge on a dynamic node leaks away with time, and hence dynamic circuits have a minimum clock frequency.
- There are several fundamentally different approaches towards building a register. The most common and widely used approach is the *master-slave configuration* which involves cascading a positive latch and negative latch (or vice-versa).
- Registers can also be constructed using the *pulse or glitch concept*. An intentional pulse (using a one shot circuit) is used to sample the input around an edge. Generally, the design of such circuits requires careful timing analysis across all process corners. Sense-amplifier based schemes are also used to construct registers and are to be used when high performance or low signal swing signalling is required.
- Choice of clocking style is an important consideration. Two phase design can result in race problems. Circuit techniques such as C^2MOS can be used to eliminate race conditions in two-phase clocking. Another option is to use true single phase clocking. However, the rise time of clocks must be carefully optimized to eliminate races.
- The combination of dynamic logic with dynamic latches can produce extremely fast computational structures. An example of such an approach, the NORA logic style, is very effective in pipelined datapaths.
- Monostable structures have only one stable state. They are useful as pulse generators.
- Astable multivibrators, or oscillators, possess no stable state. The ring oscillator is the best-known example of a circuit of this class.

- Schmitt triggers display hysteresis in their dc characteristic and fast transitions in their transient response. They are mainly used to suppress noise.

7.9 To Probe Further

The basic concepts of sequential gates can be found in many logic design textbooks (e.g., [Mano82] and [Hill74]). The design of sequential circuits is amply documented in most of the traditional digital circuit handbooks. [Partovi01] and [Bernstein98] provide in-depth overviews of the issues and solutions in the design of high-performance sequential elements.

References

[Bernstein98] K. Bernstein et al., *High-Speed CMOS Design Styles*, Kluwer Academic Publishers, 1998.

[Dopperpuhl92] D. Dopperpuhl et al., "A 200 MHz 64-b Dual Issue CMOS Microprocessor," *IEEE JSSC*, vol. 27, no. 11, Nov. 1992, pp. 1555–1567.

[Gieseke97] B. Bieseke et al., "A 600MHz Superscalar RISC Microprocessor with Out-Of-Order Execution," *IEEE ISSCC*, pp. 176-177, Feb. 1997.

[Goncalves83] N. Goncalves and H. De Man, "NORA: a racefree dynamic CMOS technique for pipelined logic structures," *IEEE JSSC*, vol. SC-18, no. 3, June 1983, pp. 261–266.

[Hill74] F. Hill and G. Peterson, *Introduction to Switching Theory and Logical Design*, Wiley, 1974.

[Jeong87] D. Jeong et al., "Design of PLL-based clock generation circuits," *IEEE JSSC*, vol. SC-22, no. 2, April 1987, pp. 255–261.

[Kozu96] S. Kozu et al., "A 100MHz 0.4W RISC Processor with 200MHz Multiply-Adder, using Pulse-Register Technique," *IEEE ISSCC*, pp. 140-141, February 1996.

[Mano82] M. Mano, *Computer System Architecture*, Prentice Hall, 1982.

[Montanaro96] J. Montanaro et al., "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor," *IEEE JSSC*, pp. 1703-1714, November 1996.

[Mutoh95] S. Mutoh et al., "1-V Power Supply High-Speed Digital Circuit Technology with Multi-threshold-Voltage CMOS," *IEEE Journal of Solid State Circuits*, pp. 847-854, August 1995.

[Partovi96] H. Partovi, "Flow-Through Latch and *Edge-Triggered* Flip Flop Hybrid Elements," *IEEE ISSCC*, pp. 138-139, February 1996.

[Partovi01] H. Partovi, "Clocked Storage Elements," in *Design of High-Performance Microprocessor Circuits*, Chandakasan et al, Ed., Chapter 11, pp. 207-233, 2001.

[Schmitt38] O. H. Schmitt, "A Thermionic Trigger," *Journal of Scientific Instruments*, vol. 15, January 1938, pp. 24–26.

[Suzuki73] Y. Suzuki, K. Odagawa, and T. Abe, "Clocked CMOS calculator circuitry," *IEEE Journal of Solid State Circuits*, vol. SC-8, December 1973, pp. 462–469.

[Yuan89] J. Yuan and Svensson C., "High-Speed CMOS Circuit Technique," *IEEE JSSC*, vol. 24, no. 1, February 1989, pp. 62–70.

CHAPTER 9

COPING WITH INTERCONNECT

Driving large capacitors

n

Dealing with transmision line effects in wires

n

Signal integrity in the presence of interconnect parasitics

n

Noise in supply networks

n

9.1 Introduction

9.2 Capacitive Parasitics

- 9.2.1 Capacitance and Reliability—Cross Talk
- 9.2.2 Capacitance and Performance in CMOS

9.3 Resistive Parasitics

- 9.3.1 Resistance and Reliability—Ohmic Voltage Drop
- 9.3.2 Electromigration
- 9.3.3 Resistance and Performance—RC Delay

9.4 Inductive Parasitics

- 9.4.1 Inductance and Reliability— Voltage Drop
- 9.4.2 Inductance and Performance—Transmission Line Effects

9.5 Advanced Interconnect Techniques

- 9.5.1 Reduced-Swing Circuits
- 9.5.2 Current-Mode Transmission Techniques

9.6 Perspective: Networks-on-a-Chip

9.7 Chapter Summary

9.1 Introduction

In previous Chapters, we have pointed out the growing impact of interconnect parasitics on all design metrics of digital integrated circuits. As mentioned, interconnect introduces three types of parasitic effects—capacitive, resistive, and inductive—, all of which influence the signal integrity and degrade the performance of the circuit. While so far we have concentrated on the modeling aspects of the wire, we now analyze how interconnect affects the circuit operation, and we present a collection of design techniques to cope with these effects. The discussion addresses each parasitic in sequence.

9.2 Capacitive Parasitics

9.2.1 Capacitance and Reliability—Cross Talk

An unwanted coupling from a neighboring signal wire to a network node introduces an interference that is generally called *cross talk*. The resulting disturbance acts as a noise source and can lead to hard-to-trace intermittent errors, since the injected noise depends upon the transient value of the other signals routed in the neighborhood. In integrated circuits, this intersignal coupling can be both capacitive and inductive, as shown earlier in Figure 1.10. Capacitive cross talk is the dominant effect at current switching speeds, although inductive coupling forms a major concern in the design of the input-output circuitry of mixed-signal circuits.

The potential impact of capacitive crosstalk is influenced by the impedance of the line under examination. If the line is floating, the disturbance caused by the coupling persists and may be worsened by subsequent switching on adjacent wires. If the wire is driven, on the other hand, the signal returns to its original level.

Floating Lines

Let us consider the circuit configuration of Figure 9.1. Line X is coupled to wire Y by a parasitic capacitance C_{XY}. Line Y sees a total capacitance to ground equal to C_Y. Assume that the voltage at node X experiences a step change equal to ΔV_X. This step appears on node Y attenuated by the capacitive voltage divider.

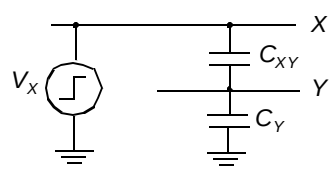


Figure 9.1 Capacitive coupling to a floating line.

$$\Delta V_Y = \frac{C_{XY}}{C_Y + C_{XY}} \Delta V_X \tag{9.1}$$

Circuits that are particularly susceptive to capacitive cross talk are networks with low-swing precharged nodes, located in adjacency to full-swing wires (with $\Delta V_X = V_{DD}$). Examples are dynamic memories, low-swing on-chip busses, and some dynamic logic families.

Example 9.1 Interwire Capacitance and Cross Talk

Consider the dynamic logic circuit of Figure 9.2. The storage capacitance C_Y of the dynamic node Y is composed of the diffusion capacitances of the pre- and discharge transistors, the gate capacitance of the connecting inverter, and the wire capacitance. A nonrelated signal Y is routed as a Metal1 wire over the polysilicon gate of one of the transistors in the inverter. This creates a parasitic capacitance C_{XY} with respect to node Y. Suppose now that node Y is precharged to 2.5 V and that signal X undergoes a transition from 2.5 V to 0 V. The charge redistribution causes a voltage drop ΔV_Y on node Y, as given by Eq. (9.1).

Assume that C_X equals 6 fF. An overlap of 3×1 μm^2 between metal1 and polysilicon results in a parasitic coupling capacitance of 0.5 fF ($3 \times 1 \times 0.057 + 2 \times 3 \times 0.054$), as obtained from Table 4.2. The computation of the fringing effect in the case of overlapping wires is complex and depends upon the relative orientation of the wires. We assumed in this analysis that two sides of the cross section contribute. A 2.5 V transition on Y thus causes a voltage disturbance of 0.19 V on the dynamic node (or 7.5%). Combined with other parasitic effects, such as charge redistribution and clock feedthrough, this might lead to circuit malfunctioning.

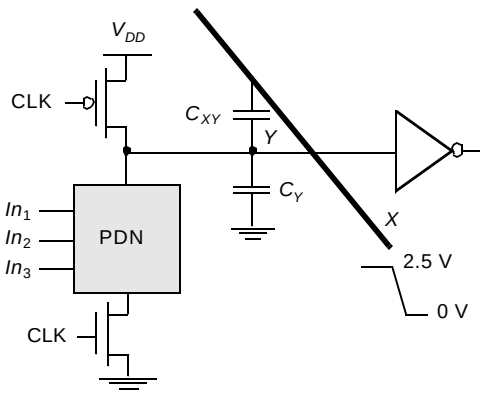
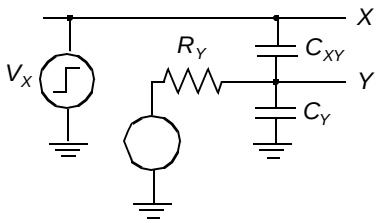


Figure 9.2 Cross talk in dynamic circuits.

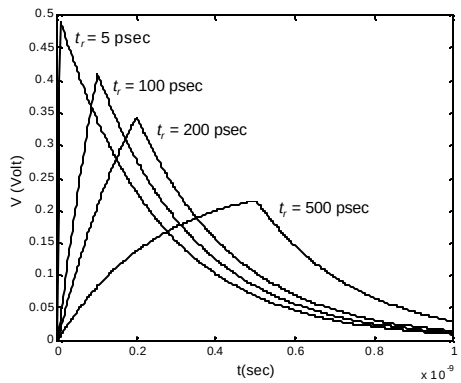
Driven Lines

If the line Y is driven with a resistance R_Y, a step on line X results in a transient on line Y (Figure 9.3a). The transient decays with a time constant $\tau_{XY} = R_Y(C_{XY}+C_Y)$. The actual impact on the "victim line" is a strong function of the rise (fall) time of the interfering signal. If the rise time is comparable or larger than the time constant, the peak value of disturbance is diminished. This is illustrated by the simulated waveforms of b. Obviously, keeping the driving impedance of a wire—and hence τ_{XY}—low goes a long way towards reducing the impact of capacitive cross talk. The keeper transistor, added to a dynamic gate or precharged wire, is an excellent example of how impedance reduction helps to control noise.

In summary, the impact of cross talk on the signal integrity of driven nodes is rather limited. The resulting glitches may cause malfunctioning of connecting sequential ele-



(a) Driven line *Y* with interferer *X*



(b) Voltage response for different rise times of V_x(0 t0 2.5 V) (R_x = 10 kΩ, C_y = 20 fF, C_{xy} = 5 fF)

Figure 9.3 Capacitive coupling to a driven line.

ments, and should therefore be carefully monitored. However, the most important effect is an increase in delay as is discussed in a subsequent paragraph.

Design Techniques—Dealing with Capacitive Cross Talk

Cross talk is a proportional noise source. This means that scaling the signal levels to increase noise margins does not help since the noise sources scale in a similar way. The only options in addressing the problem is to control the circuit geometry, or to adopt signaling conventions that are less sensitive to coupled energy. A number of ground rules can be established (as advocated in [Dally98]).

1. It is obviously not a good idea to allow the capacitance between two signal wires to grow too large if one wants to keep cross talk at a minimum. It is, for instance, bad practice to have two wires on the same layer run parallel for a long distance. One is often tempted to do just that when distributing the two clocks in a two-phase system or when routing a bus. Wires on adjacent layers should be run perpindicular.
2. Avoid floating nodes if at all possible. Nodes sensitive to cross talk problems, such as precharged busses, should be equiped with keeper devices to reduce the impedance.
3. Sensitive nodes should be well-separated from full-swing signals.
4. Make the rise (fall) time as large as possible, subject to timing constraints.
5. Use differential signaling in sensitive low-swing wiring networks. This turns the cross talk signal into a common-mode noise source that does not impact the operation of the circuit.
6. If necessary, provide a *shielding* wire—*GND* or V_{DD}—between the two signals (Figure 9.4). This effectively turns the interwire capacitance into a capacitance-to-ground and eliminates interference.

7. The interwire capacitance between signals on different layers can be further reduced by the addition of extra routing layers. When four or more routing layers are available, we can fall back to an approach often used in printed circuit board design. Every signal layer is interleaved with a *GND* or V_{DD} metal plane (Figure 9.4).

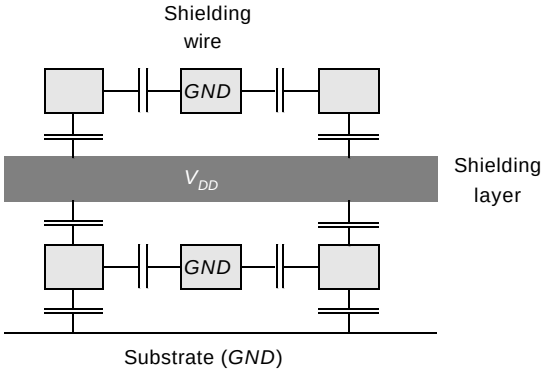


Figure 9.4 Cross section of routing layers, illustrating the use of shielding to reduce capacitive cross talk.

9.2.2 Capacitance and Performance in CMOS

Cross Talk and Performance

The previous section discussed the impact of capacitive crosstalk on the signal integrity. Even when cross talk does not result in fatal breakdown of the circuit, it still should be monitored carefully as it also impacts the performance of the gate. The circuit schematic of Figure 9.5 is illustrative of how capacitive cross talk may result in a data-dependent variation of the propagation delay. Assume that the inputs to the three parallel wires X, Y,

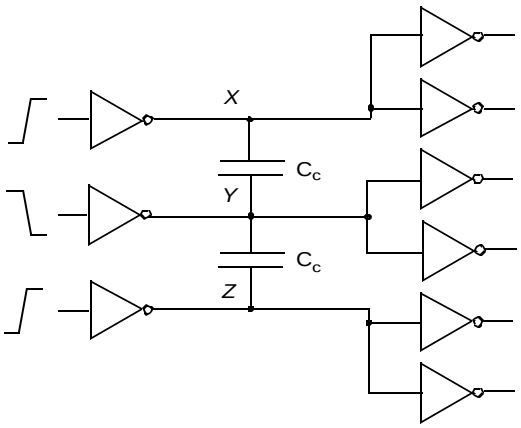


Figure 9.5 Impact of cross talk on propagation delay.

and Z experience simultaneous transitions. Wire *Y* (called the victim wire) switches in a direction that is opposite to the transitions of its neighboring signals *X* and *Z*. The coupling

capacitances experience a voltage swing that is double the signal swing, and hence represent an effective capacitive load that is twice as large as C_c—the by now well-known *Miller effect*. Since the coupling capacitance represents a large fraction of the overall capacitance in the deep-submicron dense wire structures, this increase in capacitance is substantial, and has a major impact on the propagation delay of the circuit. Observe that this is a worst-case scenario. If all inputs experience a simultaneous transition in the same direction, the voltage over the coupling capacitances remains constant, resulting in a zero-contribution to the effective load capacitance. The total load capacitance C_L of gate Y hence depends upon the data activities on the neighboring signal,s and varies between the following bounds:

$$C_{GND} \leq C_L \leq C_{GND} + 4C_c \tag{9.2}$$

with C_{GND} the capacitance of node Y to ground, including the diffusion and fanout capacitances. In [Sylvester98], it was established that for a 0.25 μm technology the wire delay with noise is potentially 80% larger than the wire delay without noise (scenario of Figure 9.5; wire length = 100 μm, fanout = 2).

Complicating the analysis of this problem is the fact that the capacitance not only depends upon the values of the surrounding wires, but also upon the exact timing of the transitions. The simultaneity of transitions can only be detected from detailed timing simulations, making the timing verification process substantially more complex. The ensuing explosion in verification cost caused by the unpredictably of the actual delay is hence a source of major concern. Just assuming worst-case conditions for the capacitances—this is, always assuming the Miller effect—leads to an overly pessimistic estimation, and overkill in the circuit design.

Design Techniques—Circuit Fabrics with Predictable Wire Delay

With cross talk making wire-delay more and more unpredictable, a designer can choose between a number of different methodology options to address the issue, some of which are enumerated below.

1. Evaluate and improve—After detailed extraction and simulation, the bottlenecks in delay are identified, and the circuit is appropriately modified.
2. Constructive layout generation—Wire routing programs take into account the effects of the adjacent wires, ensuring that the performance requirements are met.
3. Predictable structures—By using predefined, known, or conservative wiring structures, the designer is ensured that the circuit will meet his specifications and that cross talk will not be a show stopper.

The first approach is the one most often used. It has the disadvantage of requiring many iterations through the complete design generation process, and hence being very slow. The second technique is appealing. The complexity of the required tool set makes it however doubtful how much of this ambitious goal is actually achievable in a reliable (and affordable) way in the foreseeable future. although major inroads have been made already. The last approach may be the only one that is really workable in the short haul. Just as regular structures helped to tame

the complexity of transistor layout in the early 1980's, regular and predictable wiring topologies may be the way to address the capacitive cross talk problem. Field-programmable gate arrays (FPGA's), with their well-characterized interconnect grid, are an example in case. The availability of multiple layers of metal interconnect makes this approach viable for semi-custom (and custom) design as well. The *dense wire fabric* [Khatri99], shown in Figure 9.6, pre-

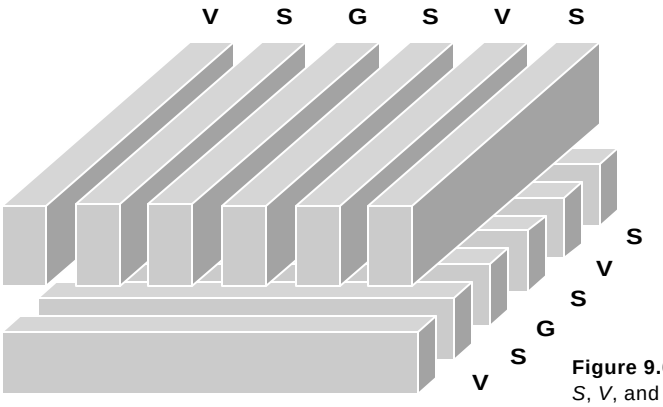


Figure 9.6 Dense wire fabric (DWF) [Khatri99]. *S*, *V*, and *G* stand for Signal, V_{DD}, and GND, respectively.

sents such a solution. Minimum-width wires are prewired on a minimum pitch distance. Wires on adjacent layers are routed orthogonally, mimimizing the cross talk. Signals on the same layer are separated by V_{DD} or GND shields. The wiring structure is personalized by providing via's at the appropriate places. The advantage of the approach is that cross talk is virtually eliminated, and that delay variation is reduced to maximally 2%. This comes at a cost in area, and a 5% capacitance—and hence performance and power—increase. For most designs, the reduction in design and verification time easily compensates for this performance penalty.

Capacitive Load and Performance

The increasing values of the interconnect capacitances, especially those of the global wires, emphasize the need for effective driver circuits that can (dis)charge capacitances with sufficient speed. This need is further highlighted by the fact that in complex designs a single gate often has to drive a large fan-out and hence has a large capacitive load. Typical examples of large on-chip loads are busses, clock networks, and control wires. The latter include, for instance, reset and set signals. These signals control the operation of a large number of gates, so fan-out is normally high. Other examples of large fan-outs are encountered in memories where a large number of storage cells is connected to a small set of control and data wires. The capacitance of these nodes is easily in the multi-picofarad range. The worst case occurs when signals go off-chip. In this case, the load consists of the package wiring, the printed circuit board wiring, and the input capacitance of the connected ICs or components. Typical off-chip loads range from 20 to 50 pF, which is multi-

ple thousand times larger than a standard on-chip load. Driving those nodes with sufficient speed becomes one of the most crucial design problems.

The main secrets to the efficient driving of large capacitive loads was already unveiled in Chapter 5.They boil down to two dominant concepts:

- Adequate transistor sizing is instrumental when dealing with large loads,
- Partitioning drivers into chains of gradually-increasing buffers helps to deal with large fanout factors.

Some other important conclusions from that analysis are worth repeating for the sake of clarity:

- When optimizing for performance, the delay of a multi-stage driver should be partitioned equally over all stages.
- If the number of stages can be freely chosen, a fanout (sizing) factor of approximately 4 per stage leads to the minimum delay for contemporary semiconductor processes. Choosing factors that are somewhat larger do not impact the performance that much, while yielding substantial area benefits.

In the following sections, we elaborate further on this topic. We focus especially on the driving of very large capacitances, such as encountered when going off-chip. We also introduce some special, but useful driver circuits.

Driving Off-Chip Capacitances—A Case Study. As established in Chapter 2, the increase in complexity of today's integrated circuits translates in an explosive need for input/outputs pins. Packages with over 1000 pins have become a necessity. This puts tough requirements on the bonding-pad design in terms of noise immunity. The simultaneous switching of a lot of pads, each driving a large capacitor, causes large transient currents and creates voltage fluctuations on the power and ground lines. This reduces the noise margins and affects the signal integrity, as will become apparent later in this chapter.

At the same time, technology scaling reduces the internal capacitances on the chip, while off-chip capacitances remain approximately constant—typically 10-20 pF. The overall effective fanout factor F of an output-pin driver hence experiences a net increase when scaling technologies. In a consistent system-design approach, we expect the off-chip propagation delays to scale in the same way as the on-chip delays. This puts an even tougher burden on the bonding-pad buffer design. The challenges associated with a bonding-pad driver design are illustrated with a simple case study.

Example 9.2 Output Buffer-Design

Consider the case where an on-chip minimum-size inverter has to drive an off-chip capacitor C_L of 20 pF. C_i equals approximately 2.5 fF for a standard gate in a 0.25 μm CMOS process. This corresponds to a t_{p0} of approximately 30 psec. The overall effective fanout F (the ratio between C_L and C_i) equals 8000, which definitely calls for a multistage buffer design. From Eq. (5.36) and assuming that $\gamma = 1$, we learn that seven stages result in a near-optimal design with a scaling factor f of 3.6, and an overall propagation delay of 0.76 nsec. With a PMOS / NMOS ratio of 1.9— the optimum ratio derived for our standard process parameters in Example 5.6—and a minimum dimension of 0.25 μm, we can derive the widths for the NMOS and PMOS transistors in the consecutive inverter stages, as shown in Table 9.1.

Table 9.1 Transistor sizes for optimally-sized cascaded buffers.

Stage	1	2	3	4	5	6	7
W_n (μm)	0.375	1.35	4.86	17.5	63	226.8	816.5
W_p (μm)	0.71	2.56	9.2	33.1	119.2	429.3	1545.5

This solution obviously requires some extremely large transistors with gate widths of up to 1.5 mm! The overall size of this buffer equals several thousand minimum-size inverters, which is quite prohibitive as a complex chip requires a large number of those drivers. This argument clearly illustrates the enormous cost associated with attempting to achieve optimum delay.

Trading off Performance for Area and Energy Reduction. Fortunately, it is not necessary to achieve the optimal buffer delay in most cases. Off-chip communications can often be performed at a fraction of the on-chip clocking speeds. Relaxing the delay requirements still allows for off-chip clock speeds in excess of 100 Mhz, while substantially reducing the buffering requirements. This redefines the buffer design problem.

Given a maximum propagation delay time $t_{p,max}$, determine the number of buffer stages N and the required scaling factor f, such that the overall area is minimized. This is equivalent to finding a solution that sets t_p as close as possible to $t_{p,max}$.

The optimization problem is now reformulated into the finding of the minimum integer value of N that obeys the constraints of Eq. (9.3).

$$\frac{t_{p,max}}{t_{p0}} \geq \ln(F)\frac{f}{\ln(f)} = N \times F^{1/N} \tag{9.3}$$

This transcendental optimization problem can be solved using a small computer program or using mathematical packages such as MATLAB [Etter93]. Figure 9.7 plots the right-hand side of this equation as a function of N for some values of F. For a given value of the overall effective fanout F, and a maximum value of the delay (t_{pmax}/t_{po}), the minimum number of buffers N is derived from inspection of the appropriate curve.

Using the total width of the transitors as a measure, we can also project the area savings of the larger scaling factor. Assuming that the area of the minimum inverter equals A_{min}, and that scaling the transistors with a factor f results in a similar area increase, we can derive the area of the driver as a function of f.

$$\begin{aligned} A_{\text{driver}} &= (1 + f + f^2 + \ldots + f^{N-1})A_{min} \\ &= \left(\frac{f^N - 1}{f - 1}\right)A_{min} = \frac{F - 1}{f - 1}A_{min} \end{aligned} \tag{9.4}$$

In short, the area of the driver is approaximately inversely proportional to the tapering factor f. Choosing larger values of f can help to substantially reduce the area cost.

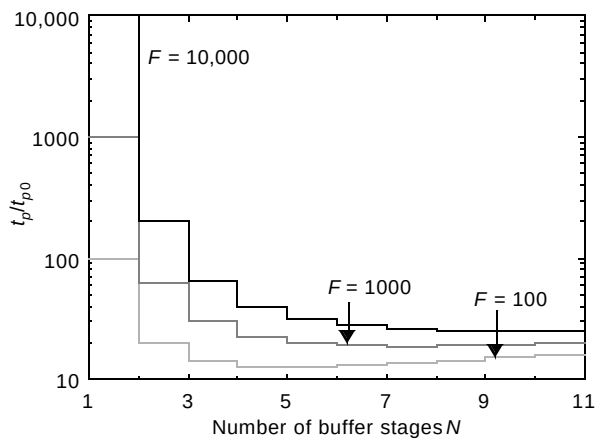


Figure 9.7 t_p/t_{p0} as a function of the number of buffer stages for various values of the overall fanout F.

It is also worth reflecting briefly on the energy dissipation of the exponential buffer. While it takes an amount of energy equal to $C_L V_{DD}{}^2$ to just switch just the load capacitance, the driver itself consumes energy as well to switch its internal capacitances (ignoring short-circuit energy). This overhead energy is approximated by the expression below,

$$\begin{aligned} E_{\text{driver}} &= (1 + f + f^2 + \ldots + f^{N-1}) C_i V_{DD}^2 \\ &= \left(\frac{F-1}{f-1}\right) C_i V_{DD}^2 \approx \frac{C_L}{f-1} V_{DD}^2 \end{aligned} \tag{9.5}$$

This means that driving a large capacitance fast (suing the optimum taperaing factor of 3.6) takes an extra 40% of energy. For large load capacitances, this is a substantial overhead. Backing off somewhat from the optimum tapering factor can help to reduce the extra dissipation.

Example 9.3 Output Driver Design—Revisited

Applying these results to the bonding-pad driver problem and setting $t_{p,max}$ to 2 nsec results in the following solution: $N = 3$, $f = 20$, and $t_p = 1.8$ nsec. The required transistor sizes are summarized in Table 9.2.

Table 9.2 Transistor sizes of redesigned cascaded buffer.

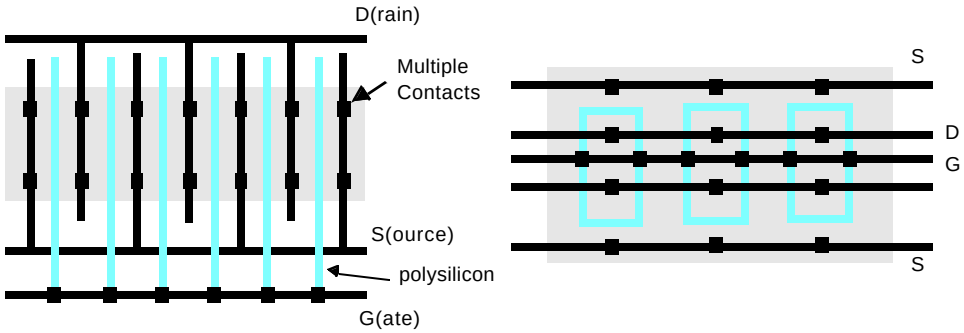
Stage	1	2	3
W_n (μm)	0.375	7.5	150
W_p (μm)	0.71	14.4	284

The overall area of this solution is approximately 7.5 times smaller than the optimum solution, while its speed is reduced by less than a factor of 2.5. It also reduces the extra energy dissipation per switching transition due to the buffer intrinsic capacitances to an almost negligible amount. The overall dissipation of the combined buffer and load is reduced by 24%,

which corresponds to a hefty amount given the size of the load capacitor. This clearly shows that **designing a circuit for the right speed, not for the maximum speed**, pays off!

Design Consideration—Implementing Wide Transistors

Even this redesigned buffer requires wide transistors. One must be careful when designing those devices, since the large value of *W* translates into very long gate connections. Long poly-



(a) Small transistors in parallel (b) Circular transistors

Figure 9.8 Approaches to implementing very wide transistors.

silicon wires tend to be highly resistive, which degrades the switching performance. This problem can be addressed in a number of ways. For instance, a wide transistor can be constructed by connecting many smaller transistors in parallel (Figure 9.8a) or by using ring- or even spiral-shaped devices (Figure 9.8b). In each approach, the resistance of the gate is reduced with the aid of a low-resistance metal bypass connecting the shorter polysilicon sections. An example of a pad driver designed using these techniques is shown in Figure 9.9.

Design Challenge—Designing Reliable Output and Input Pads

The design of bonding pad-drivers is obviously a critical and nontrivial task that is further complicated by noise and reliability considerations. For instance, the large transient currents resulting from the switching of the huge output capacitance can cause latchup to occur. Multiple well and substrate contacts supplemented with guard rings help avoid the onset of this destructive effect.

Guard rings are grounded p+ diffusions in a p-well and supply-connected n+ diffusions in an n-well that are used to collect injected minority carriers before they reach the base of the paraistic bipolar transistors. These rings should surround the NMOS and PMOS transistors of the final stage of the output pad driver.

The designer of an input pad faces some different challenges. The input of the first stage of the input buffer is directly connected to external circuitry, and is hence sensitive

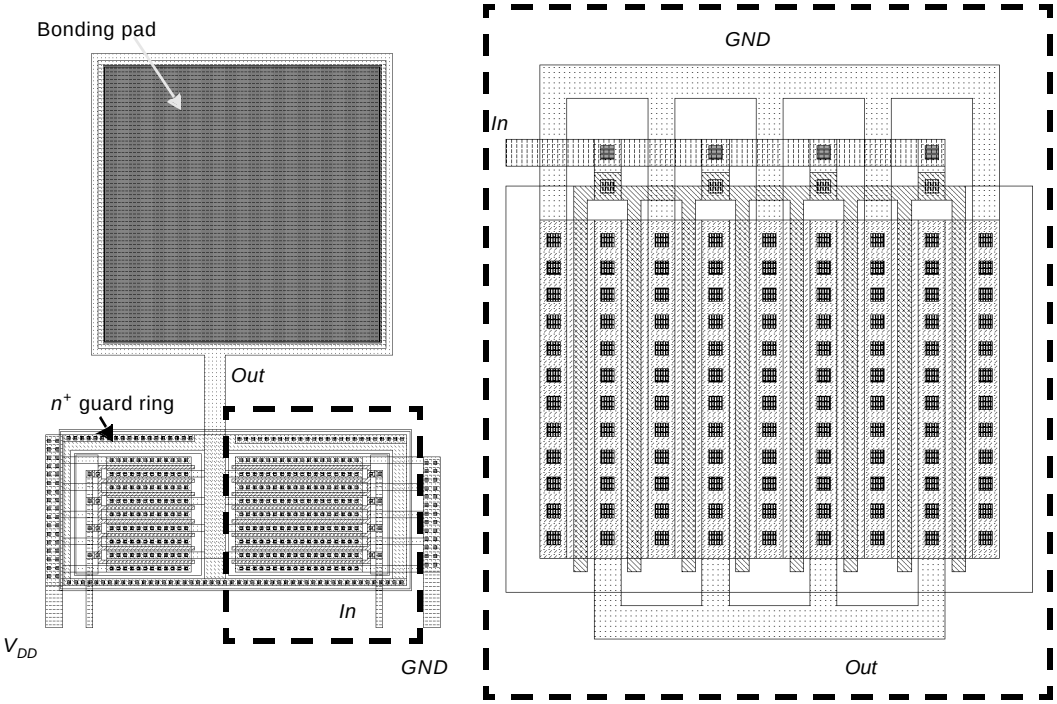


Figure 9.9 Layout of final stage of bonding-pad driver. The plot on the right side is a magnification of the NMOS transistor connected between *GND* and *Out*. See also Colorplate 13.

to any voltage excursions on the connected input pins. A human walking over a synthetic carpet over in 80% relative humidity can accumulate a voltage potential of 1.5 kV—you have probably experienced the sparks that jump from your hand when touching a metal object under those circumstances. The same is true for the assembly machinery. The gate connection of an MOS transistor has a very high input resistance (10^{12} to 10^{13} Ω). The voltage at which the gate oxide punctures and breaks down is about 40-100 V, and is getting smaller with reducing oxide thicknesses. A human or machine, charged up to a high static potential, can hence easily cause a fatal breakdown of the input transistors to happen when brought in contact with the input pin. This phenomenum called *Electrostatic Discharge* (ESD) has proven to be fatal to many circuits duirng manufacturing and assembly.

A combination of resistance and diode clamps are used to defray and limit this potentially destructive voltage. A typical *electrostatic protection circuit* is shown in Figure 9.10a. The protection diodes D_1 and D_2 turn on when the voltage at node X rises below V_{DD} or goes below ground. The resistor R is used to limit the peak current that flows in the diodes in the event of an unusual voltage excursion. Current process tend to use tub resistors (*p*-diffusion in an *n*-well, and n-diffusion for a p-well) to implement the resistors, and values can be anywhere from 200Ω to 3 kΩ. The designer should be aware that the resulting *RC* time-constant can be a performance limiting factor in high-speed circuits. Figure 9.10b shows the layout of a typical input pad.

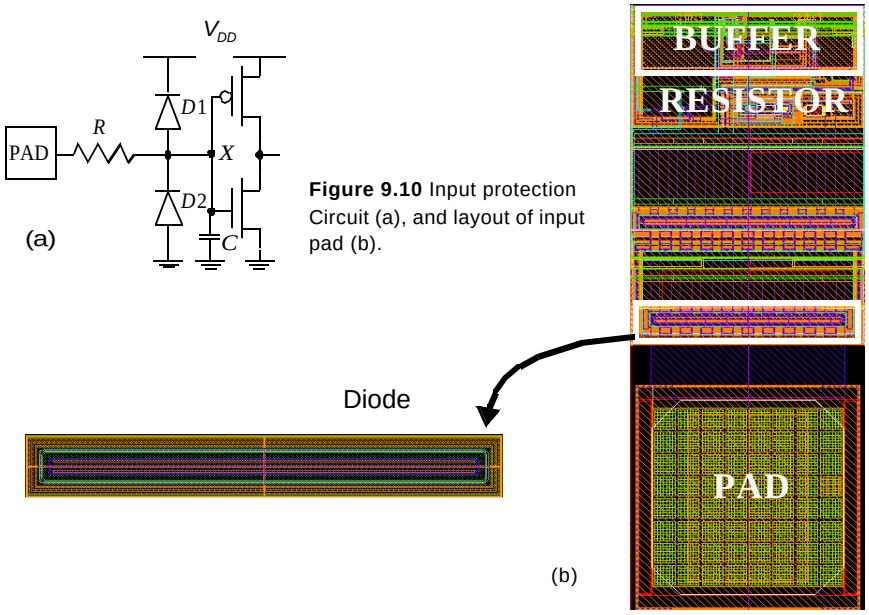


Figure 9.10 Input protection Circuit (a), and layout of input pad (b).

The task of the pad designer seems to become harder for every new technology generation. Fortunately, a number of novel packaging technologies that reduce off-chip capacitance, are currently emerging. For instance, advanced technqiues such as *ball grid arrays* and chip-on-board go a long way in helping to reducing the off-chip driving requirements (see also Chapter 2).

Some Interesting Driver Circuits. Sofar, most of our driver circuits have been simple inverters. Sometimes, some variants are necessary. The *tri-state* buffer is an example of such. Busses are essential in most digital systems. A bus is a bundle of wires that connect a set of sender and receiver devices such as processors, memories, disks, and input/output devices.When one device is sending on the bus, all other sending devices should be disconnected. This can be achieved by putting the output buffers of those devices in a high-impedance state *Z* that effectively disconnects the gate from the output wire. Such a buffer has three possible states—0, 1, and *Z*—and is therefore called a *tri-state* buffer.

Implementing a tri-state inverter is straightforward in CMOS. Simultaneously turning off the NMOS and PMOS transistor produces a floating output node that is disconnected from its input. Two possible implementations of a tri-state buffer are shown in Figure 9.11. While the first one is simple, the second is more appropriate for driving large capacitances. Having transistors in series in the output stage is to be avoided due to the huge area overhead.

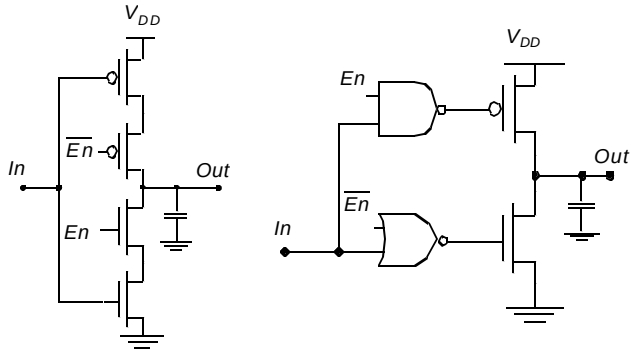


Figure 9.11 Two possible implementations of a tri-state buffer. *En* = 1 enables the buffer.

9.3 Resistive Parasitics

In this section, we discuss the impact of wiring resistance on reliability and performance. Before doing so, we first analyze the effects of scaling on the wiring resistance and introduce a number of approximative models for resistive wires.

9.3.1 Resistance and Reliability—Ohmic Voltage Drop

Current flowing through a resistive wire results in an ohmic voltage drop that degrades the signal levels. This is especially important in the power distribution network, where current levels can easily reach amperes, as illustrated in Figure 9.12 for the Compaq Alpha processor family. Consider now a 2 cm long V_{DD} or *GND* wire with a current of 1mA per µm

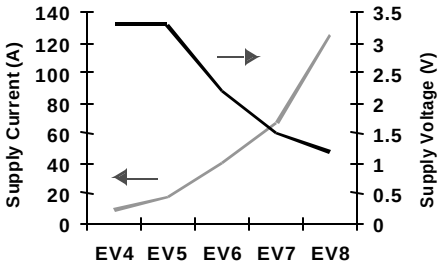


Figure 9.12 Evolution of power-supply current and supply voltage for different generations of the high-performance alpha microprocessor family from Compaq. Even with the reductions in supply voltage, a total supply current of over 100 A has to be supported for the newer generations.

width. This current is about the maximum that can be sustained by an aluminum wire due to *electromigration*, which is discussed in the subsequent section. Assuming a sheet resistance of 0.05 Ω/q, the resistance of this wire (per µm width) equals 1 kΩ. A current of 1 mA/µm would result in a voltage drop of 1 V. The altered value of the voltage supply reduces noise margins and changes the logic levels as a function of the distance from the supply terminals. This is demonstrated by the circuit in Figure 9.13, where an inverter placed far from the power and ground pins connects to a device closer to the supply. The difference in logic levels caused by the *IR* voltage drop over the supply rails might partially turn on transistor M_1. This can result in an accidental discharging of the precharged, dynamic node *X*, or cause static power consumption if the connecting gate is static. In

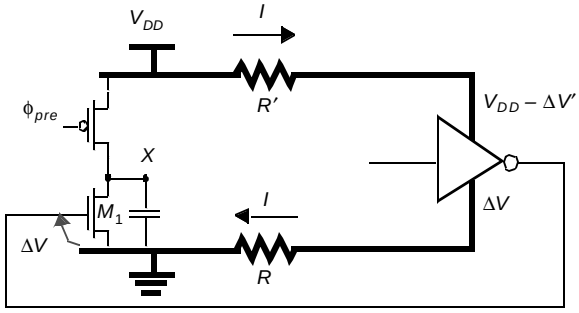


Figure 9.13 Ohmic voltage drop on the supply rails reduces the noise margins.

short, the current pulses from the on-chip logic, memories and I/O pins cause voltage drops over the power-distribution network, and are the major source for on-chip power-supply noise. Beyond causing a reliability risk, IR drops on the supply network also impact the performance of the system. A small drop in the supplu voltage may cause a significant increase in delay.

The most obvious solution to this problem is to reduce the maximum distance between the supply pins and the circuit supply connections. This is most easily accomplished through a structured layout of the power distribution network. A number of on-chip power-distribution networks with peripheral bonding are shown in Figure 9.14. In all

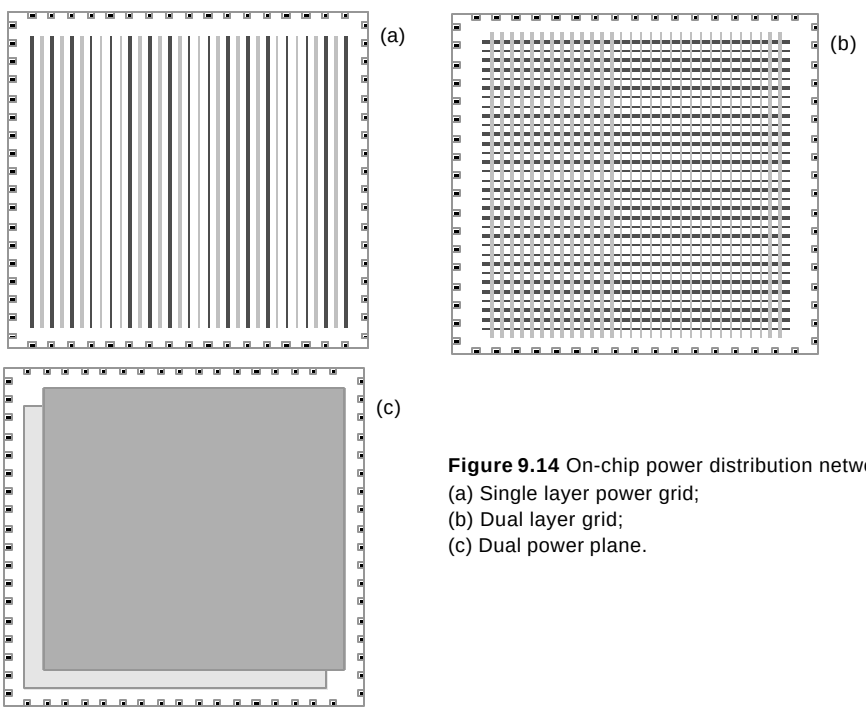


Figure 9.14 On-chip power distribution networks.
(a) Single layer power grid;
(b) Dual layer grid;
(c) Dual power plane.

cases, power and ground are brought onto the chip via bonding pads located on the four sides of the chip. Which approach to use depends upon the number of coarse metal layers (this is, thick, high pitch topmost metal layers) one wants to allocate to power distribution.

In the first approach (a), power and ground are routed vertically (or horizontally) on the same layer. Power is brought in from two sides of the chip. Local power strips are strapped to this upper grid, and then further routed on the lower metal levels. Method (b) uses two coarse metal layers for the power distribution, and the power is brought in from the four sides of the die. This approach was used in the EV5 generation of the Compaq Alpha processor [Herrick00]. The combination of power and clock distribution occupied more than 90% of the 3rd and 4th Al layers. One further and more aggressive step to take is to use two solid metal planes for the distribrution of V_{dd} and GND (c). This approach has the advantage of drastically reducing the resistance of the network. The metal planes also act as shields between data signalling layers, hence reducing cross-talk. They also help to reduce the on-chip inductance. Obviously, this approach is only feasible when sufficient metal layers are available.

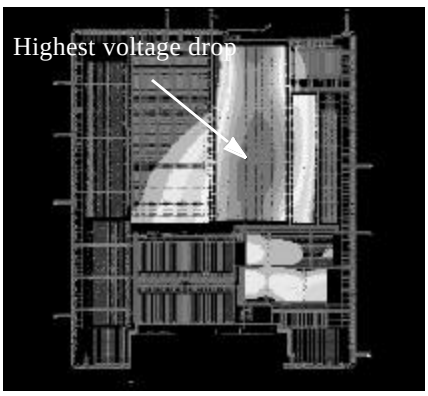
The sizing of the power network is a non-trivial task. The power grid can be modeled as a network of resistors (wires) and current sources (logic), containing hundreds of millions of elements. Very often, many paths exist between the power pins and the supply connections of a chip module or gate. While in general the current follows the path of the lowest resistance, the exact flow depends upon factors such as the current draw from neighboring modules that share the same network. The analysis is complicated by the fact that peak currents drawn by the connected modules are distributed over time. IR drop is a dynamic phenomenon due primarily to simultaneous switching events such as clocks, and bus drivers. As large drivers begin to switch, the simultaneous demand from current from the power network stresses the grid. At the same time, a worst-case analysis adding all the peak currents may lead to a gross overdimensioning of the wires.

The above makes it clear that computer-aided design tools are a necessary companion of the power-distribution network designer. Given the complexity of todays integrated circuits, transistor-level analysis of the complete power-network requirements is just not feasible. At the same time, partitioning of the problem over sub-sections might not result in an accurate picture either. Changes to the power grid in one section tend to have a global impact. This is illustrated in Figure 9.15, which shows the simulated IR voltage drop of a complex digital circuit. A first implementation (a) suffers from a more than acceptable drop in the upper right module of the design, because only the top portion of the power grid feeds the large drivers at the top. The lower portions of the module are not directly connected to the grid. Adding just one single strap to the network largely resolves this problem, as shown in (b).

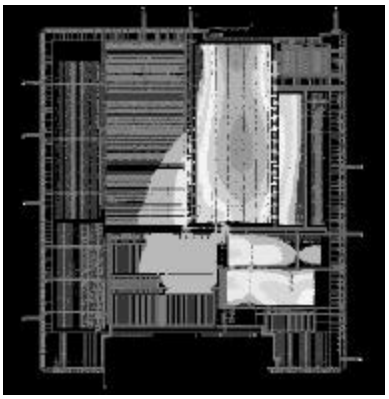
Therefore, an accurate picture of the IR risk cannot be obtained unless the entire chip is verified as a single entity. Any tool used for this purpose must have the capacity to analyze multi-million resistor grids. Fortunately, a number of efficient and quite accurate power-grid analysis tools are currently available [Simplex,]. They combine a dynamic analyis of the current requirements of the circuit modules with a detailed modeling of the popwer network. Tools as such are quite indispensable in todays design process.

9.3.2 Electromigration

The current density (current per unit area) in a metal wire is limited due to an effect called *electromigration*. A *direct* current in a metal wire running over a substantial time period,



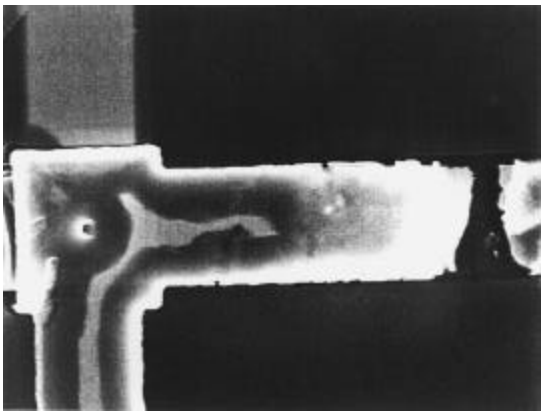
(a)



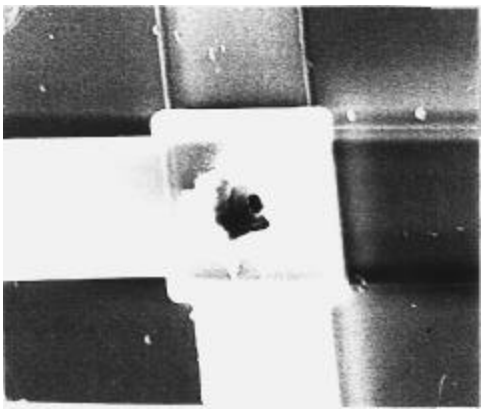
(b)

Figure 9.15 Simulated IR voltage drop in power-distribution network of two versions of a complex digital integrated circuit. The gray-scale is used to indicate the severeness of the voltage drop. Adding an extra strap to the upper layers of the power network (b) helps to address the large voltage trop in the upper right module in the initial design (a).

causes a transport of the metal ions. Eventually, this causes the wire to break or to short-circuit to another wire. This type of failure will only occur after the device has been in use for some time. Some examples of failure caused by migration are shown in Figure 9.16. Notice how the first photo clearly shows hillock formation in the direction of the electron current flow.



(a) Line-open failure.



(b) Open failure in contact plug.

Figure 9.16 Electromigration-related failure modes (*Courtesy of N. Cheung and A. Tao, U.C. Berkeley*).

The rate of the electromigration depends upon the temperature, the crystal structure, and the average current density. The latter is the only factor that can be effectively controlled by the circuit designer. Keeping the current below 0.5 to 1 mA/µm normally prevents migration. This parameter can be used to determine the minimal wire width of the power and ground network. Signal wires normally carry an ac-current and are less susceptible to migration. The bidirectional flow of the electrons tends to anneal any damage done

to the crystal structure. Most companies impose a number of strict wire-sizing guidelines on their designers, based on measurements and past experience. Research results have shown that many of these rules tend to be overly conservative [Tao94].

Design Rule

Electromigration effects are proportional to the average current flow through the wire, while IR voltage drops are a function of the peak current.

At the technology level, a number of precautions can be taken to reduce the migration risk. One option is to add alloying elements (such as Cu or Tu) to the aluminum to prevent the movement of the Al ions. Another approach is to control the granularity of the ions. The introduction of new interconnect materials is a big help as well. For instance, the use of Copper interconnect increases the expected lifetime of a wire with a factor of 100 over Al.

9.3.3 Resistance and Performance—RC Delay

In Chapter 4, we established that the delay of a wire grows quadratically with its length. Doubling the length of a wire increases its delay by a factor of four! The signal delay of long wires therefore tends to be dominated by the *RC* effect. This is becoming an ever larger problem in modern technologies, which feature an increasing average length of the global wires (Figure 4.27)—at the same time that the average delay of the individual gates is going down. This leads to the rather bizar situation that it may take multiple clock cycles to get a signal from one side of a chip to its opposite end [Dally-DAC]. Providing accurate synchronization and correct operation becomes a major challenge under these circumstances. In this section, we discuss a number of design techniques that may help to cope with the delay imposed by the resistance of a wire.

Better Interconnect Materials

A first option for reducing *RC* delays is to use better interconnect materials when they are available and appropriate. The introduction of silicides and Copper have helped to reduce the resistance of polysilicon (and diffused) and metal wires, respectively, while the adoption of dielectric materials with a lower permittivity lowers the capacitance. Both Copper and low-permittivity dielectrics have become common in advanced CMOS technologies (starting from the 0.18 μm technology generation). Yet, the designer should be aware that these new materials only provide a temporary respite of one or two generations, and do not solve the fundamental problem of the delay of long wires. Innovative design techniques are often the only way of coping with the latter.

Example 9.4 Impact of Advanced Interconnect Materials

Copper offers a resistivity that is 1.6 times lower than that of Aluminum (Table 4.4) from a pure material perspective. Cladding and other manufacturing artifacts may increase the effective resistivity of on-chip Cu wires to approximately 2.2 10^{-8} Ω-m.

Sometimes, it is hard to avoid the use of long polysilicon wires. A good example of such circumstance are the address lines in memories, which must connect to a large number of transistor gates. Keeping the wires in polysilicon increases the memory density substantially by avoiding the overhead of the extra metal contacts. The polysilicon-only option unfortunately leads to an excessive propagation delay. One possible solution is to drive the word line from both ends, as shown in Figure 9.17a. This effectively reduces the worst-case delay by a factor of four. Another option is to provide an extra metal wire, called a *bypass*, which runs parallel to the polysilicon one, and connects to it every *k* cells (Figure 9.17b). The delay is now dominated by the much shorter polysilicon segments between the contacts. Providing contacts only every *k* cells helps to preserve the implementation density.

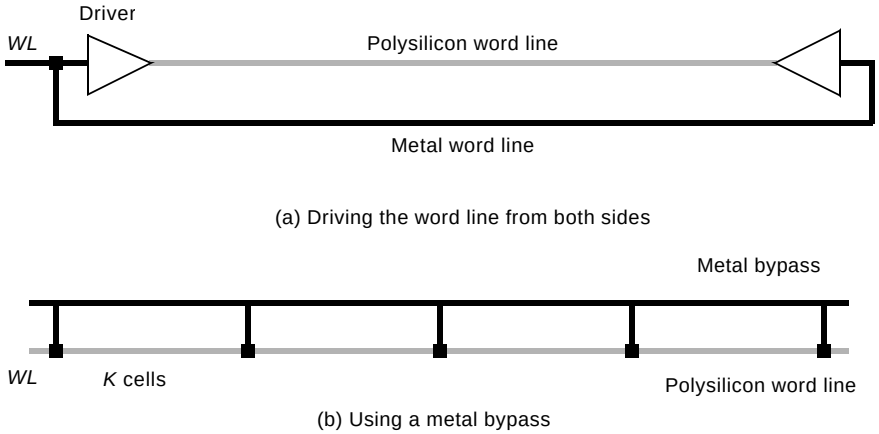
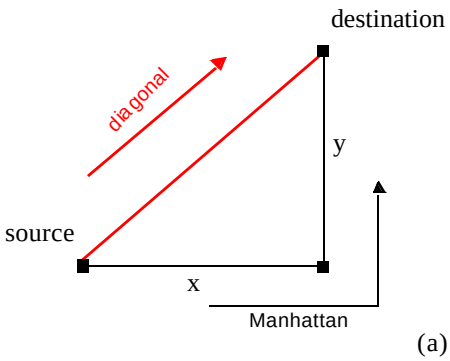


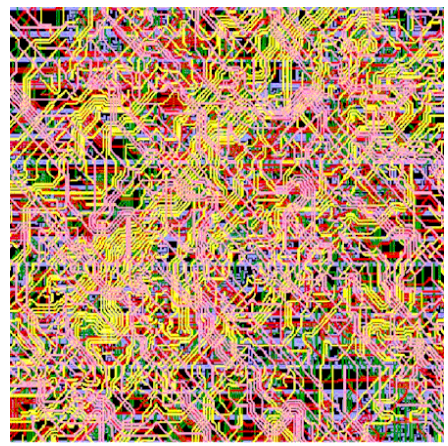
Figure 9.17 Approaches to reduce the word-line delay.

Better Interconnect Strategies

With the length of the wire being a prime factor in both the delay and the energy consumption of an interconnect wire, any approach that helps to reduce the wire length is bound to have an essential impact. It was pointed out earlier that the addition of interconnect layers tends to reduce the average wire length, as routing congestion is reduced and interconnections can pretty much follow a direct path between source and destination. Yet, the "Mahattan-style" wiring approach that is typical in todays routing tools brings with it a substantial amount of overhead that is often overlooked. In the Manhattan-style routing, interconnections are routed first along one of two preferred directions, followed by a connection in the other direction (Figure 9.18a). It seems obvious that routing along the diagonal direction would yield a sizable reduction in wirelength—of up to 29% in the best case. Ironically, 45° lines were very popolar in the early days of integrated circuits

designs, but got of out of vogue because of complexity issues, impact on tools, and mask-making concerns. Recently, it has been demonstrated that these concerns can be addressed adequately, and that 45° lines are perfectly feasible [Simplex01]. The impact on wiring is quite tangible: a reduction of 20% in wirelength! This in turn results in higher performance, lower power dissipation, and smaller chip area. The density of wiring structures that go beyond pure Manhattan directions, is demonstrated clearly in a sample layout, shown in Figure 9.18b.





(b)

Figure 9.18 Manhattan versus diagonal routing. (a) Manhattan routing uses preferential routing along orthogonal axis, while diagonal routing allows for 45° lines. (b) example of layout using 45° lines.

Introducing Repeaters

The most popular design approach to reducing the propagation delay of long wires is to introduce intermediate buffers, also called *repeaters*, in the interconnect line (Figure 9.19). Making an interconnect line m times shorter reduces its propagation delay quadratically, and is sufficient to offset the extra delay of the repeaters when the wire is sufficiently long. Assuming that the repeaters have a fixed delay t_{pbuf}, we can derive the delay of the partitioned wire.

$$t_p = 0.38\,rc\left(\frac{L}{m}\right)^2 m + m t_{pbuf} \tag{9.6}$$

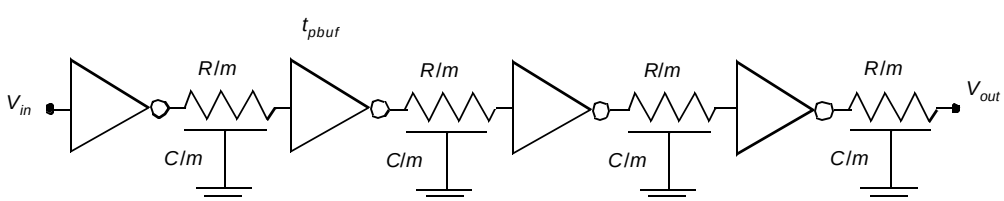


Figure 9.19 Reducing *RC* interconnect delay by introducing repeaters.

The optimal number of buffers that minimes the overall delay can be found by setting $\partial t_p/\partial m = 0$.

$$m_{opt} = L\sqrt{\frac{0.38rc}{t_{pbuf}}} = \sqrt{\frac{t_{pwire(unbuffered)}}{t_{pbuf}}} \tag{9.7}$$

yielding a minimum delay of

$$t_{p,\,opt} = 2\sqrt{t_{pwire(unbuffered)}t_{pbuf}} \tag{9.8}$$

and is obtained when the delay of the individual wire segments is made equal to that of a repeater. Observe also the equivalence between this design problem and the chain of transmission gates, as discussed in Example 6.11.

Example 9.5 Reducing the Wire Delay through Repeaters

In Example 4.8, we derived the propagation delay of a 10 cm long, 1 μm wide Al1 wire to be 31.4 nsec. Eq. (9.7) indicates that a partitioning of the wire into 18 sections would minimize its delay, assuming a fixed t_{pbuf} of 0.1 nsec. This results in an overall delay time of 3.5 nsec, an important improvement. Similarly, the delays of equal-length Polysilicon and Al5 wires would be reduced to 212 nsec (from 112 μsec) and 1.3 nsec (from 4.2 nsec) by introducing 1058 and 6 stages, respectively.

The above analysis is simplified and optimistic in the sense that t_{pbuf} is a function of the load capacitance. Sizing the repeaters is needed to reduce the delay. A more precise expression of the delay of the interconnect chain is obtained by modeling the repeater as an RC network, and by using the Ellmore delay approach. Assuming that R_d and C_d are the resistance and capacitance of a minimum-sized repeater, and s is the sizing factor, this leads to the following expression:

$$t_p = m\left(0.69\frac{R_d}{s}\left(\frac{cL}{m} + sC_d\right) + 0.69\left(\frac{rL}{m}\right)(sC_d) + 0.38rc\left(\frac{L}{m}\right)^2\right) \tag{9.9}$$

By setting $\partial t_p/\partial m$ and $\partial t_p/\partial s$ to 0, optimal values for m, s and t_{pmin} are obtained.

$$\begin{aligned} m_{opt} &= L\sqrt{\frac{0.38rc}{0.69R_dC_d}} = \sqrt{\frac{t_{pwire(unbuffered)}}{t_{pbuf,min}}} \\ s_{opt} &= \sqrt{\frac{R_dc}{rC_d}} \\ t_{p,min} &= 2.4L\sqrt{R_dC_drc} \end{aligned} \tag{9.10}$$

Eq. (9.10) clearly demonstrates how the insertion of repeaters linearizes the delay of a wire. Also, observe that for a given technology and a given interconnect layer, there exists aan optimal length of the wire segments between repeaters. We call this the critical length L_{crit}.

$$L_{crit} = \frac{L}{m_{opt}} = \sqrt{\frac{0.69 R_d C_d}{0.38 rc}} \tag{9.11}$$

It can be derived that the delay of a segment of critical length always equals $3.23 R_d C_d$, and is independent of the routing layer. Inserting repeaters to reduce the delay of a wire only makes sense when the wire is at least two times longer than the critical length.

Example 9.6 Minimizing the Wire Delay (Revised)

In Chapter 5 (Example 5.5), we determined the intinsic delay of a minimum-sized inverter in our 0.25 µm CMOS process to be 32.5 psec with (average) values of R_d and C_d of 7.8 kΩ and 6 fF, respectively. For an Al1 wire (c = 110 aF/µm; r = 0.075 Ω/µm), this yields an optimum repeater sizing factor s_{opt} = 43. Inserting 31 of these sized-up inverters, sets the minimum delay of the 10 cm wire of Example 9.5 to a more realistic 4.7 nsec. This is in contrast to just inserting 31 minimum-sized repeaters, which would actually increase the delay to 61 nsec due to the poor driving capability of the repeater.

The critical length for Al1 evaluates to 3.2 mm, while it equals 54 µm and 8.8 mm for Polysilicon and Al5, respectively.

Repeater insertion has proven such an efficient and essential tool in combatting long wire delays that modern design automation tools perform this task automatically [REFERENCE].

Optimizing the Interconnect Architecture

Even with buffer insertion, the delay of a resistive wire cannot be reduced below the minimum dictated by Eq. (9.8). Long wires hence often exhibit a delay that is longer than the clock period of the design. For instance, the 10 cm long Al1 wire of Example 9.6 comes with a minimum delay of 4.7 nsec, even after optimal buffer insertion and sizing, while the 0.25 µm CMOS process featured in this text can sustain clock speeds in excess of 1 GHz (this is, clock periods below 1 nsec). The wire delay all-by-itself hence becomes the limiting factor on the performance achievable by the integrated circuit. The only way to address this bottleneck is to tackle it at the system architecture-level.

Wire pipelining is a popular performance-improvement technique in this category. The concept of pipelining was introduced in Chapter 7 as a means of improving the throughput performance of logic modules with long critical paths. A similar approach can be used to increase the throughput of a wire, as is illustrated in Figure 9.20. The wire is partitioned in k segments by inserting registers or latches. While this does not reduce the delay through the wire segment—it takes k clock cycles for a signal to proceed through the wire—, it helps to increase its throughput, as the wire is handling k signals simultaneously at any point in time. The delay of the individual wire segments can further be optimized by repeater insertion, and should be below a single clock period.

This is only one example of the many techniques that the chip architect has at her disposition to deal with the wire delay problem. The most important message that emerges from this discussion is that wires have to be considered early on in the design process, and can no longer be treated as an afterthought as was most often the case in the past.

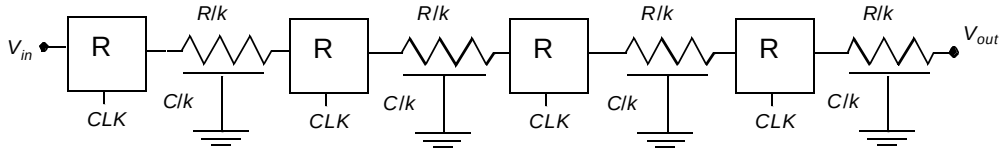


Figure 9.20 Wire pipelining improves the throughput of a wire.

9.4 Inductive Parasitics

Besides having a parasitic resistance and capacitance, interconnect wires also exhibit an inductive parasitic. An important source of parasitic inductance is introduced by the bonding wires and chip packages. Even for intermediate-speed CMOS designs, the current through the input-output connections can experience fast transitions that cause voltage drops as well as ringing and overshooting, phenomena not found in *RC* circuits. At higher switching speeds, wave propagation and transmission line effects can come into the picture. Both effects are analyzed in this section. First, we discuss the source of these inductive parasitics and their quantitative values.

9.4.1 Inductance and Reliability—$L\frac{di}{dt}$ Voltage Drop

During each switching action, a transient current is sourced from (or sunk into) the supply rails to charge (or discharge) the circuit capacitances, as modeled in Figure 9.21. Both V_{DD} and V_{SS} connections are routed to the external supplies through bonding wires and package pins and possess a nonignorable series inductance. Hence, a change in the transient current creates a voltage difference between the external and internal (V_{DD}', GND') supply voltages. This situation is especially severe at the output pads, where the driving of the large external capacitances generates large current surges. The deviations on the internal supply voltages affect the logic levels and result in reduced noise margins.

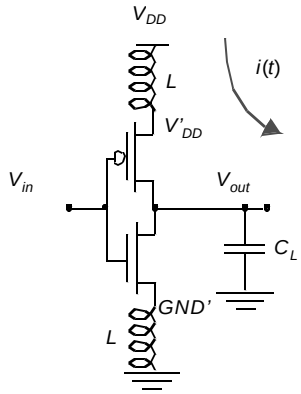


Figure 9.21 Inductive coupling between external and internal supply voltages.

Example 9.7 Noise Induced by Inductive Bonding Wires and Package Pins

Assume that the circuit in Figure 9.21 is the last stage of an output pad driver, driving a load capacitance of 10 pF over a voltage swing of 2.5 V. The inverter has been dimensioned so that the 10–90% rise and fall times of the output signal (t_r, t_f) equal 1 nsec. Since the power and ground connections are connected to the external supplies through the supply pins, both connections have a series inductance L. For a traditional throughole packaging approach, an inductance of around 2.5 nH is typical. To simplify the analysis, assume first that the inverter acts as a current source with a constant current (dis)charging the load capacitance. A (average) current of 20 mA is required to achieve the 1 nsec output rise and fall times.

$$I_{avg} = (10\text{ pF} \times (0.9 - 0.1) \times 2.5\text{ V}) / 1\text{ nsec} = 20\text{ mA}$$

This scenario occurs where the the buffer is driven by a steep step-function at its input. The left side of Figure 9.22 plots the simulated evolution of output voltage, inductor current, and inductor voltage over time for t_f = 50 psec. The abrupt current change causes a steep voltage spike of up to 0.95 V over the inductor. In fact, the voltage drop would have been larger, if the drop itself did not slow down the transients and reduce the demand for current through positive feedback. Nevertheless, a supply voltage with that much variation is unacceptable.

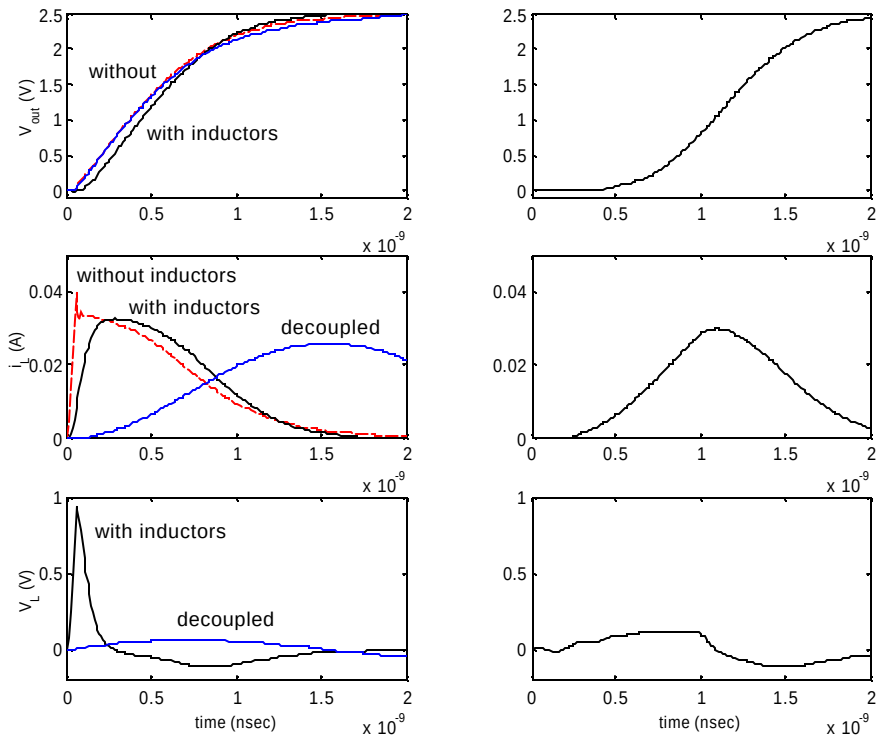
In reality, the charging current is rarely a constant. A better approximation assumes that the current rises linearly to a maximum after which it drops back to zero, again in a linear fashion. The current distribution over time models as a triangle. This situation occurs when the input signal to the buffer displays slow rise/fall times. Under this model, the (estimated) voltage drop over the inductor equals

$$v_L = L\, di_L/dt = (2.5\text{ nH} \times 40\text{ mA}) / ((1.25/2) \times 1\text{ nsec}) = 133\text{ mV}.$$

The peak current of (approximately) 40 mA results from the fact that the total delivered charge—or the integrated area under I_L—is fixed, which means that the peak of the triangular current distribution is double the value of the rectangular one. The denominator factor estimates the time it takes to go from 0 to peak current. Simulated results of this scenario are shown on the right side of Figure 9.22 for t_f = 1 nsec. It can be observed that the current distribution indeed becomes triangular for this slow input slope. The simulated voltage drop over the inductor is reduced to xxx mV. This indicates that avoiding fast-changing signals at the inputs of the drivers for large capacitirs goes a long way in reducing the *Ldi/dt* effects.

In an actual circuit, a single supply pin serves a large number of gates or output drivers. A simultaneous switching of those drivers causes even worse current transients and voltage drops. As a result, the internal supply voltages deviate in a substantial way from the external ones. For instance, the simultaneous switching of the 16 output drivers of an output bus would cause a voltage drop of at least 1.1 V if the supply connections of the buffers were connected to the same pin on the package.

Improvements in packaging technologies are leading to ever-increasing numbers of pins per package. Packages with up to 1000 pins are currently available. Simultaneous switching of a substantial number of those pins results in huge spikes on the supply rails that are bound to disturb the operation of the internal circuits as well as other external components connected to the same supplies.



(a) input rise/fall time: 50 psec (b) input rise/fall time: 800 psec

Figure 9.22 Simulated signal waveforms for output driver connected to bonding pads for input rise/fall times of 50 psec (a) and 800 psec (b), respectively. The waveforms in (a) show the results when (1) no inductors are present in the power distribution network (ideal case); (b) in the presence of inductors; and (c) with added decoupling capcitance of 200 pF.

Design Techniques

A number of approaches are available to the designer to address the $L(di/dt)$ problem.

1. **Separate power pins for I/O pads and chip core—**Since the I/O drivers require the largest switching currents, they also cause the largest current changes. It is wise to isolate the center of the chip, where most of the logic action occurs, from the drivers by providing different power and ground pins.

2. **Multiple power and ground pins**—In order to reduce the *di/dt* per supply pin, we can restrict the number of I/O drivers connected to a single supply pin. Typical numbers are five to ten drivers per supply pin. Be aware that this number depends heavily upon the switching characteristics of the drivers such as the number of simultaneously switching gates and the rise and fall times.

3. **Careful selection of the positions of the power and ground pins on the package**—The inductance of pins located at the corners of the package is substantially higher (Figure 9.23).

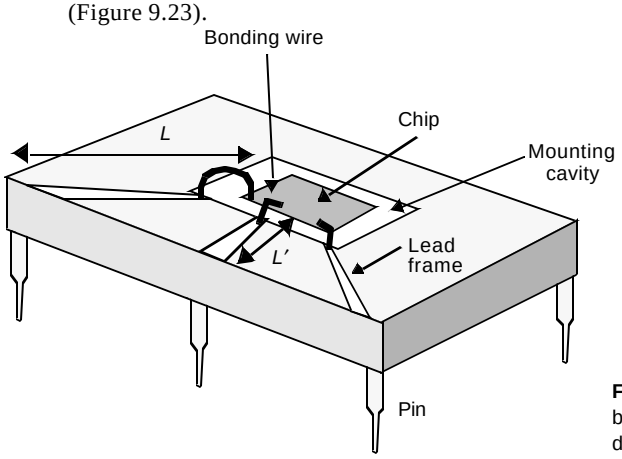


Figure 9.23 The inductance of a bonding-wire/pin combination depends upon the pin position.

4. **Increase the rise and fall times** of the off-chip signals to the maximum extent allowable. and distributed all over the chip, especially under the data busses. The example above demonstrated that output drivers with over-designed rise and fall times are not only expensive in terms of area, but also might affect the circuit operation and reliability. We concluded in Section 9.2.2 that the better driver in terms of area is the one that achieves a specified delay, not the one with the fastest delay. When noise is considered, the best driver is the one that achieves a specified delay with the maximum allowable rise and fall times at the output.

Problem 9.1 Design of Output Driver with Reduced Rise/Fall Times

Given a cascaded output driver designed for a given delay that produces excessive noise on the supply lines, define the best approach to address the noise problem: (a) Scale down all stages of the buffer; (b) Scale down the last stage only; (c) Scale down all stages except the last one.

---

5. **Use advanced packaging technologies** such as surface-mount or hybrids that come with a substantially reduced capacitance and inductance per pin. For instance, we can see from Table 2.2 that the bonding inductance of a chip mounted in flip-chip style on a substrate using the solder-bump techniques is reduced to 0.1nH, which is 50 to 100 times smaller than for standard packages.

6. **Adding decoupling capacitances on the board—**These capacitances, which should be added for every supply pin, act as local supplies and stabilize the supply voltage seen by the chip. They separate the bonding-wire inductance from the inductance of the board interconnect (Figure 9.24). The bypass capacitor, combined with the inductance, actually acts as a low-pass network that filters away the high-frequency components of the transient voltage spikes on the supply lines.[1]

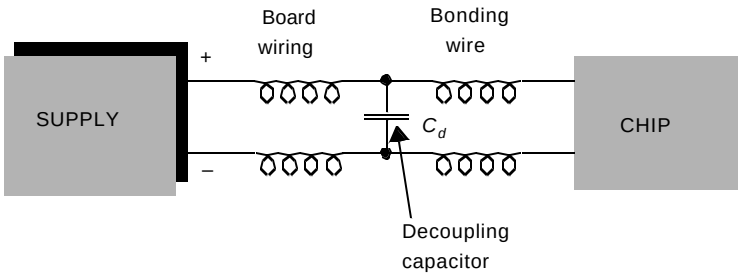


Figure 9.24 Decoupling capacitors isolate the board inductance from the bonding wire and pin inductance.

Example 9.8 Impact of Decoupling Capacitances

A decoupling capacitance of 200 pF was added between the supply connections of the buffer circuit examined in Example 9.7. Its impact is shown in the simulations of Figure 9.22. For an input signal rise time of 50 psec, the spike on the supply rail is reduced from 0.95 — without decoupling capacitance — to approximately 70 mV. The waveforms also demonstrate how the current for charging the output capacitance is drawn from the decoupling capacitor initially. Later in the transition, current is gradually being drawn from the supply network through the bonding-wire inductors.

7. **Adding decoupling capacitances on the chip—**In high-performance circuits with high switching speeds and steep signal transitions, it is becoming common practice to integrate decoupling capacitances on the chip, which ensures cleaner supply voltages. On-chip bypass capacitors reduce the peak current demand to the average value. To limit the voltage ripple to 0.25 V, a capacitance of around 12.5 nF must be provided for every 50 Kgate module in a 0.25 μm CMOS process [Dally98]. This capacitance is typically implemented using the thin gate oxide. A thin-oxide capacitor is essentially a MOS transistor with its drain and source tied together. Diffusion capacitors can be used for that purpose as well.

Example 9.9 On-Chip Decoupling Capacitances in Compaq's Alpha Processor Family [Herrick00]

The Alpha processor family, mentioned earlier in this chapter, has been at the leading edge in high-performance microprocessors for the last decade. Pushing the clock speed to the extreme seriously challenges the power-distribution network. The Alpha processors are hence at fore-

[1] A word of caution. The combination of inductor-capacitance might act as an underdamped resonator, making the noise problem actually worse. A careful insertion of damping resistance is often required.

front when it comes to on-chip decoupling capacitance. Table 9.3 enumerates some characteristics of consecutive processors in the family.

Table 9.3 On-chip decoupling capacitances in Alpha Processor family.

Processor	Technology	Clock Frequency	Total Switching Capacitance	On-chip Decoupling Capacitance
EV4	0.75 μm CMOS	200 MHz	12.5 nF	128 nF
EV5	0.5 μm CMOS	350 MHz	13.9 nF	160 nF
EV6	0.35 μm CMOS	575 MHz	34 nF	320 nF

Given a gate capacitance of 4 fF/μm^2 (t_{ox} = 9.0 nm), the 320 nF decoupling capacitance of the EV6 requires 80 mm^2 of die area, which is 20% of the total chip. To minimize the impact, the capacitance was placed under the major busses. Even with this large amount of real estate dedicated to decoupling, the designers were runing out of capacitance, and had to resort to some innovative techniques to provide more near-chip decoupling. The problem was solved by wirebonding a 2 μF, 2 cm^2 capacitor to the chip, and connecting the power grid to the capacitance through 160 Vdd/GND wirebond pairs. A diagram of the *Wirebond Attached Chip Capacitor* (WACC) approach is shown in Figure 9.25. This example serves as a clear illustration of the problems that power-grid designers are facing in the high-performance arena.

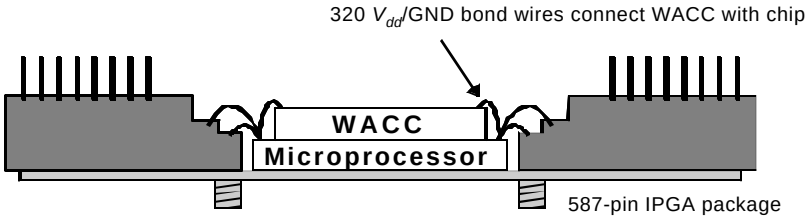


Figure 9.25 The Wirebond-attached Chip capacitor (WACC) approach provides extra decoupling capaciatnce close to the chip for the Compaq EV6 microprocessor [Herrick00]. Thr microprocessor is connected to the outside world via 320 signal pins and 198 supply/ground pins.

Finally, be aware that the mutual inductance between neighboring wires also introduces cross talk. This effect is not yet a major concern in CMOS but definitely is emerging as an issue at the highest switching speeds ([Johnson93]).

9.4.2 Inductance and Performance—Transmission Line Effects

When an interconnection wire becomes sufficiently long or when the circuits become sufficiently fast, the inductance of the wire starts to dominate the delay behavior, and transmission line effects must be considered. This is more precisely the case when the rise and fall times of the signal become comparable to the time of flight of the signal waveform

across the line as determined by the speed of light. Until recently, this was only the case for the fastest digital designs implemented on a board level, or in exotic technologies such as GaAs and SiGe. As advancing technology increases line lenghts and switching speeds, this situation is gradually becoming common in fastest CMOS circuits as well, and transmission-line effects are bound to become a concern of the CMOS designer as well.

In this section, we discuss some techniques to minimize or mitigate the impact of the transmission line behavior. A first and foremost technique is to use approriate termination. Termination is however only effective if the current-return path is well-behaved. Shielding of wires that are prone to transmission line effects will prove to be a necessity.

Termination

In our discussion of transmission lines in Chapter 5, it became apparent that appropriate termination is the most effective way of minimizing the delay. Matching the load impedance to the characteristic impedance of the line results in the fastest response. This leads to the following design rule:

> **To avoid the negative effects of transmission-line behavior such as ringing or slow propagation delays, the line should be terminated, either at the source (series termination), or at the destination (parallel termination) with a resistance matched to its characteristic impedance Z_0.**

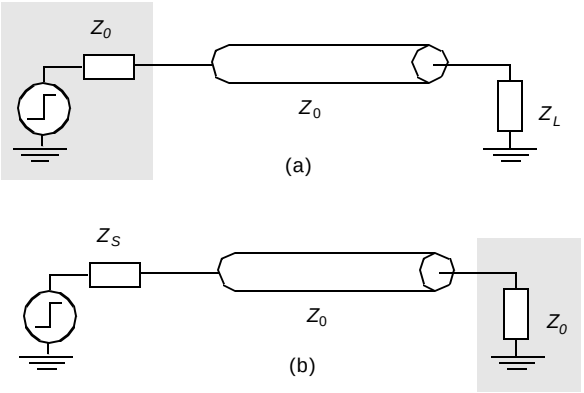


Figure 9.26 Matched termination scenarios for wires behaving as transmission lines: (a) series termination at the source; (b) parallel termination at the destimation.

The two scenarios — series and parallel termination — are depicted in Figure 9.26. Series termination requires that the impedance of the signal source is matched to the connecting wire. This approach is appropriate for many CMOS designs, where the destination load is purely capacitive. The impedance of the driver inverter can be matched to the line by careful transistor sizing. To drive a 50-Ω line, for example, requires a 53-μm-long NFET and a 135-μm-long PFET (in our 0.25 μm CMOS technology) to give a nominal output impedance of 50 Ω.

It is important that the impedance of the driver is closely matched to the line, typically to within 10% or better, if excessive reflections of travelling waves are to be avoided. Unfortunately, the on-resistance of a FET may vary by 100% across process, voltage and temperature variations. This can be compensated for by making the resistance

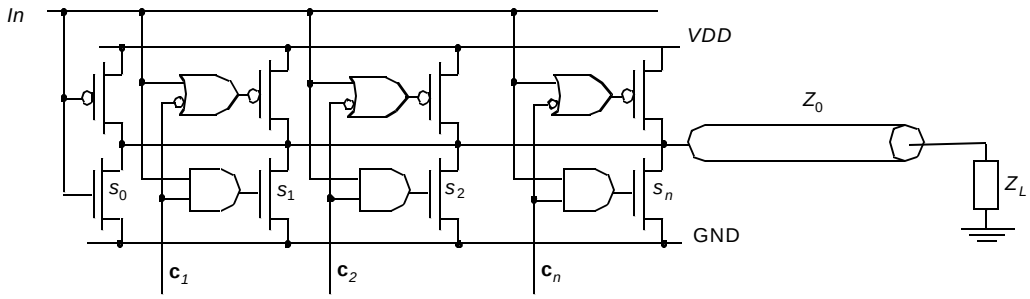


Figure 9.27 Tunable segmented driver providing matched series-termination to a transmission line load.

of the driver transistors electrically tunable (Figure 9.27). Each of the driver transistors is replaced by a segmented driver, with the segments being switched in and out by control lines c_1 to c_n, to match the driver impedance as close as possible to the line impedance. Each of the segments has a different resistance, shaped by the shape factors s_i (typically ratioed with factors of 2). A fixed element (s_0) is added in parallel with the adjustable ones, so that the range of adjustment is limited and a more precise adjustment can be made with less bits. The control lines are usually driven by a feedback control circuit that compares an on-chip reference driver transistor to a fixed external reference transistor [Dally98].

Similar considerations are valid when the termination is provided at the destination end, called *parallel termination*. Be aware that this approach results in a standby current, and that keeping this current flowing continuously might result in unacceptable power dissipation. Parallel termination is very popular for high-speed inter-chip, board-level interconnections, where the termination is often implemented by inserting a (grounded) resistor next to the input pin (of the package) at the end of the wire. This off-chip termination does not take into account the package parasitics and the internal circuitry, and may introduce unacceptably large reflections on the signal line. A more effective approach is to include the terminating resistor inside the package after the package parasitics.

Generally, CMOS fabrication does not provide us with means to make precise, temperature-insensitive resistors. Usually, it is necessary to use interconnect materials or FETs as resistors. Because these resistors are temperature, supply-voltage, and process-dependent, tuning is necessary (either statically or continuously). Digital trimming such as proposed earlier in this section can be used to accomplish that goal. Another issue when using MOSFETs as resistors is ensuring that the device exhibits a linear behavior over the operation region of interest. Since PMOS transistors typically display a larger region of linear operation than their NMOS counterparts, they are the preferred way of implementing a terminating resistance. Assume that the triode-connected PMOS transistor of Figure 9.28a is used as a 50-Ω matched termination, connected to V_{DD}. From simulations (Figure 9.28d), we observe that the resistance is fairly constant for small values of VR, but increases rapidly once the transistor staurates. We can extend the linear region by increasing the bias voltage of the PMOS transistor (Figure 9.28b). This however requires an extra supply voltage, which is not practical in most situations. A better approach is to add a diode-connected NMOS transistor in parallel with the PMOS device (Figure 9.28c). The

combination of the two devices gives a near-constant resistance over the complete voltage range (as we have already observed in Chapter 6 when discussing transmission gates). Many other clever schemes have been devised, but they are out the scope of this textbook. We refer the reader to [Dally98] for a more detailed perspective.

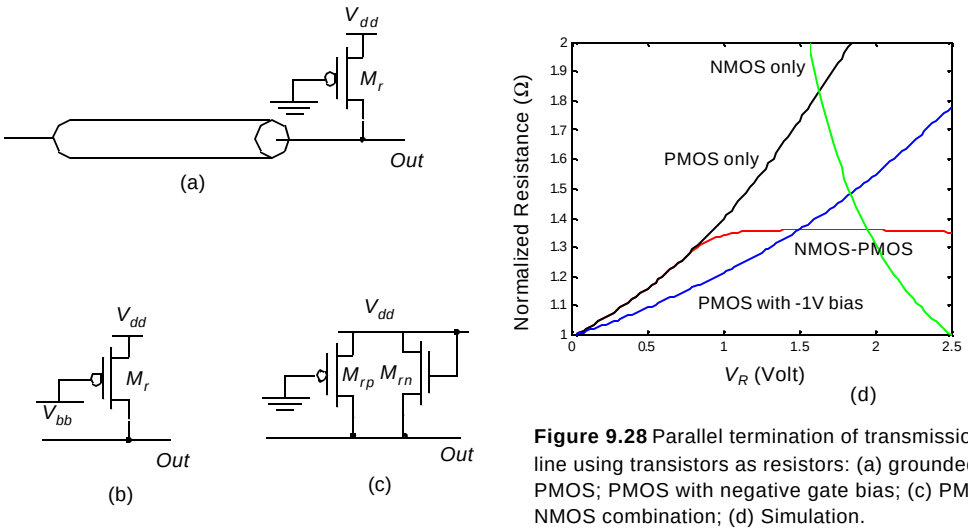


Figure 9.28 Parallel termination of transmission line using transistors as resistors: (a) grounded PMOS; PMOS with negative gate bias; (c) PMOS-NMOS combination; (d) Simulation.

Shielding

A transmission-line is in essence a four-port network. While we focus mostly on the signal path, the *signal-return path* should not be ignored. Kirchoff's law tells us that, when we inject a current i into a signal conductor, a net current of $-i$ must flow in the return. The characteristics of the transmission line (such as its characteristic impedance) are a strong function of the signal return. Hence, if we want to control the behavior of a wire behaving as a transmission line, we should carefully plan and manage how the return current flows. A good example of a well-defined transmission line is the coaxial cable, where the signal wire is surrounded by a cylindrical ground plane. To accomplish similar effects on a board or on a chip, designers often surround the signal wire with ground (supply) planes and shielding wires. While expensive in real-estate, adding shielding makes the behavior and the delay of an interconnection a lot more predictable. Yet even with these precautions, powerful extraction and simulation tools will be needed in the future for the high-performance circuit designer.

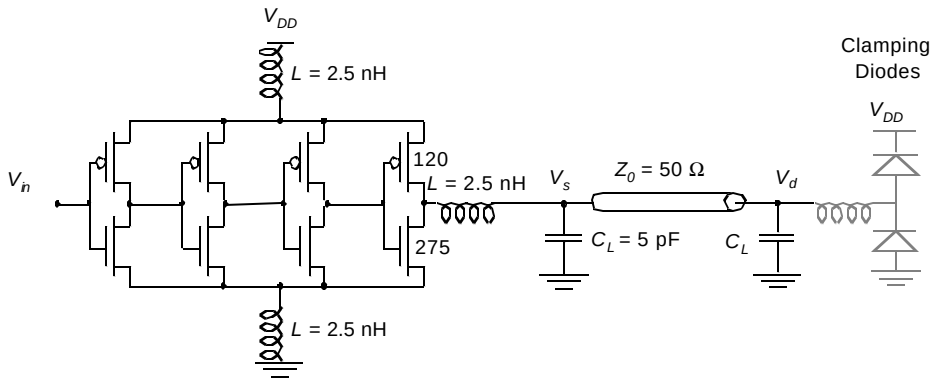
Example 9.10 Design of an Output Driver—Revisited

As a conclusion to this section, we simulate an output driver that includes the majority of the parasitic effects introduced in this chapter. The schematics of the driver are shown in Figure 9.29a. The inductance of the power, ground, and signal pins (estimated at 2.5 nH), as well as the transmission line behavior of the board have been included (assuming a wire length of 15cm, which is a typical for board traces). The driver is initially designed to produce rise and fall times of 0.33 nsec for a total load capacitance of 10pF. Producing the required average current of 60 mA requires transistor widths of 120 μm and 275 μm for the NMOS and PMOS

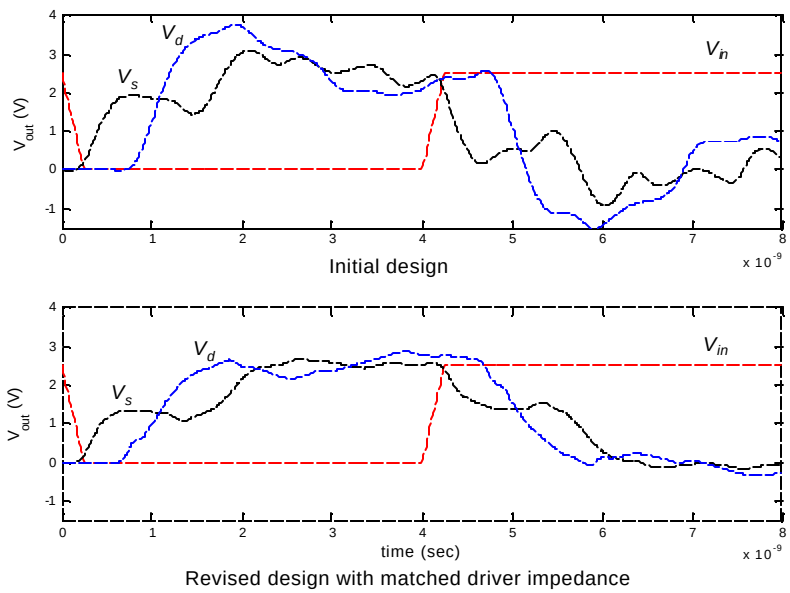
transistors in the final stage of the driver. The first graph of Figure 9.29b shows the simulated waveforms for this configuration. Severe ringing is observed. It is quite obvious that the driver resistance is underdamped, hence the large overshoots and the large settling time.

The circuit was redesigned to eliminate some of these effects. First of all, the sizes of the driver transistsors were reduced so that their impedances match the characteristic impedance of the transmission line (to 65 μm and 155 μm for NMOS and PMOS, respectively). A decoupling capacitance of 200 pF was added to the supplies of the drivers. Finally, the input-protection diodes of the fan-out device were added to the model to create a more realistic per-

spective. The simulation shows that these modifications are quite effective, giving a circuit that is both better behaved and faster..



(a) Output driver schematic.



(b) Transient simulations

Figure 9.29 Simulation of output driver for various terminations.

9.5 Advanced Interconnect Techniques

In previous sections, we have discussed a number of techniques on how to cope with the capacitive, resistive and inductive parasitics that come with interconnect wires. In this sec-

tion, we discuss a number of more-advanced circuits that have emerged in recent years. More specifically, we discuss how reducing the signal swing can help to reduce the delay and the power dissipation when driving wires with large capacitances.

9.5.1 Reduced-Swing Circuits

Increasing the size of the driver transistor, and thus increasing the average current I_{av} during switching, is only one way of coping with the delay caused by a large load capacitance. Another approach could be the reduction of the signal swing at the output of the driver and over the load capacitance. Intuitively, we can understand that reducing the charge that has to be displaced may be beneficial to the performance of the gate. This is best understood by revisiting the propagation-delay equation Eq. (5.16)

$$t_p = \int_{v_1}^{v_2} \frac{C_L(v)}{i(v)} dv \tag{9.12}$$

Assuming tthat the load capacitance is a constant, we can derived a simplified expression for the delay,

$$t_p = \frac{C_L V_{swing}}{I_{av}} \tag{9.13}$$

where $V_{swing} = v_2 - v_1$ equals the signal swing at the output, and I_{av} is the average (dis)charge current. This expression clearly demonstrates that the delay reduces linearly with the signal swing, under the condition that the (dis)charge current is not affected by the reduction in voltage swing. Just lowering the overall supply voltage does not work: while reducing the swing, it also lowers the current by a similar ratio. This was demonstrated earlier in Figure 3.28, which showed that the equivalent resistance of a (dis)charge transistor is approximately constant over a wide range of supply voltages. While potentially offering an increased performance, lower signal swings further carry the major benefit of lowering the dynamic power consumption, which can be substantial when the load capacitance is large.

On the negative side, reduced signal swings result in smaller noise margins, and hence impact the signal integrity and reliability. Furthermore, CMOS gates are not particularly effective in detecting and reacting to small signal changes, because of the relatively small transconductance of the MOS device. In order to work properly and to achieve high performance, reduced-swing circuits normally require amplifier circuits, whose task it is to restore the signal to its full swing in a minimum amount of time and with a minimum amount of extra energy consumption. The overhead of adding extra amplifiers is only justifiable for network nodes with a large fan-in, where the circuit can be shared over many input gates. Typical examples of such nodes are the data or address buses of a microprocessor, or the data lines in a memory array. In the former case, the amplifier is most often called a *receiver*, while in a memory it is named a *sense amplifier*.[2]

[2] More details on sense amplifier circuits can be found in the later chapter on semiconductor memories.

A typical diagram of a reduced-swing network, consisting of a driver, a large capacitance/resistance interconnect wire, and a receiver circuit, is shown in Figure 9.30. Many different designs for both drivers and receivers have been devised [Zhang00], of which we only discuss a few. In general, the reduced-swing circuits fall into two major categories, *static* and *dynamic* (or precharged). Another differentiating factor between circuits is the signalling technique. Most receivers circuits use a *single-ended* appraoch, where the receiver detects an absolute change in voltage on a single wire. Other circuits use *differential* or *double-ended* signalling techniques, where both the signal and its complement are transmitted, and the receiver detects a relative change in voltage between the two wires. While the latter approach requires substantial more wiring real-estate, it has the advantage of being more robust in the presence of noise.

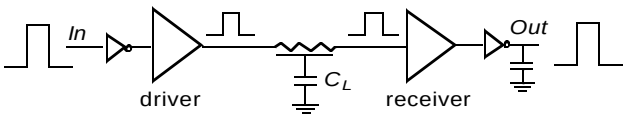


Figure 9.30 Reduced-swing interconnect circuit. The driver circuit reduces the normal voltage swing to a reduced value, while the receiver detects the signal and restores it to the normal swing value.

Static Reduced-Swing Networks

The task of designing a reduced-swing network simplifies substantially when a second lower supply rail VDD_L is available. A simple and robust single-ended driver circuit using that second supply is shown in Figure 9.31. The challenge is in the receiver design. Just using an inverter does not work very well: the small swing at the input of the NMOS transistor results in a small pull-down current, and hence a slow high-to-low transition at the output; furthermore, the low value of VDD_L is not sufficient to turn off the PMOS transistor, which detoriates the performance even more, and causes static power dissipation to occur in addition. A more refined receiver circuit, inspired by the DCVSL gate discussed eralier, is shown in Figure 9.31 Using a low-voltage inverter, the complement of the input

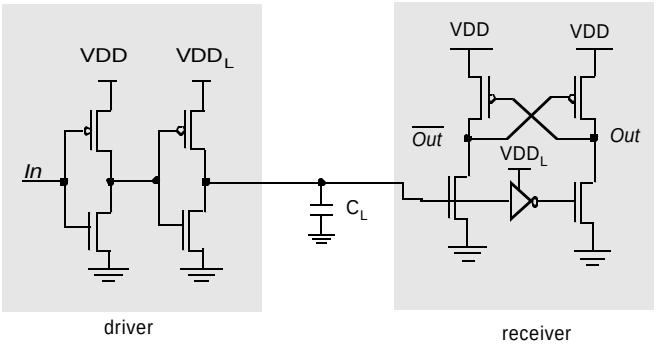
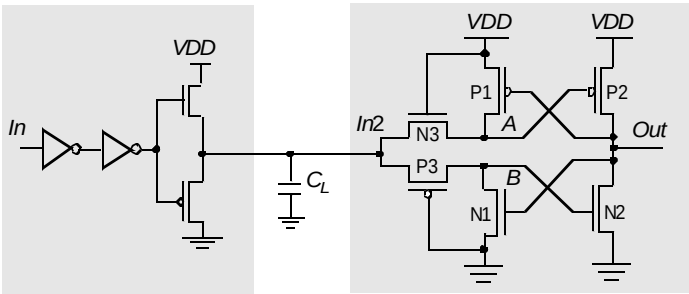
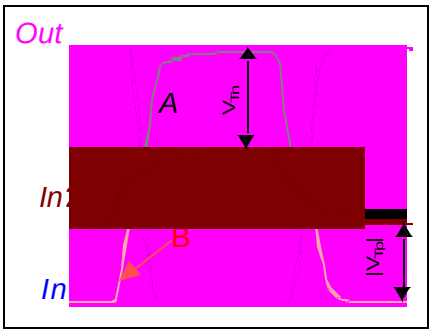


Figure 9.31 Single-ended, static reduced-swing driver and receiver.

signal is generated. The receiver now acts as a differential amplifier. The cross-coupled load transistors ensure that the output is restored to VDD, and that no static power is con-



(a)



(b)

Figure 9.32 (a) Symmetric source-follower driver with level converter; (b) Simulated waveforms;

sumed in steady-state mode. The positive feedback helps to accelerate the transitions. The disadvantage of the circuit is that it becomes unacceptably slow for low swing values.

A circuit that avoids the use of a second supply rail is shown in Figure 9.32. By reversing the postions of the NMOS and the PMOS transistors in the final stage of the driver, we have limited the signal swing on the interconnect from $|V_{Tp}|$ to V_{DD}-V_{Tn}, or approximately 2 threshold values below VDD. The symmetrical receiver/level converter consists of two cross-coupled transistor pairs (P1-P2 and N1-N2), and two diode-connected transistors, which isolate the reduced-swing interconnect wire from the full-swing output signal. To understand the operation of the receiver, assume that node *In2* goes from low to high, or from $|V_{Tp}|$ to VDD-V_{Tn}. Initially, node A and B are at $|V_{Tp}|$ and GND, respectively. During the transition period, with both N3 and P3 conducting, *A* and *B* rise to VDD-V_{Tn} as shown in Figure 9.32b. Consequently, N2 turns on, and *Out* goes to low. The feedback transistor P1 pulls *A* further up to *VDD* to turn P2 completely off. *In2* and *B* stay at VDD-V_{Tn}. Note that there is no standby current path from *VDD* to GND through N3 although the gate-source voltage of N3 almost V_{Tn}. Since the circuit is symmetric, a similar explanation holds for the high-to-low transition. Since the main function of transistors P1 and N1 is to provide positive feedback and to completely cut off P2 or N2, they can be very weak, which minimize their fight against the driver. The sensing delay of the receiver is as small as two inverter delays.

Problem 9.2 Energy Consumption of Reduced-Swing Receiver

Assuming that the capacitance is dominated by the load capacitance of the wire, derive the energy-reduction per signal transition that the circuit of Figure 9.32 offers over the full-swing version.

The interconnect systems shown so far are all single-ended. Figure 9.33 shows a differential scheme. The driver generates two complimentary reduced-wing signals using a second supply rail. The receiver is nothing less than a clocked sense-amplifier-based differential register, introduced earlier in Section 7.7. The differential approach offers a very high rejection of common-mode noise signals such as supply-rail noise and crosstalk interference. The signal-swing can hence be reduced to very low levels — operation with swings as low as 200 mV has been demonstrated. The driver uses NMOS transistors for both pull-up and pull-down. The receiver is a clocked unbalanced current-latch sense amplifier, which is discharged and charged at every clock cycle. The main disadvantage of the differential approach is the doubling of the number of wires, which certainly presents a major concern in many designs. The extra clock signal adds further to the overhead.

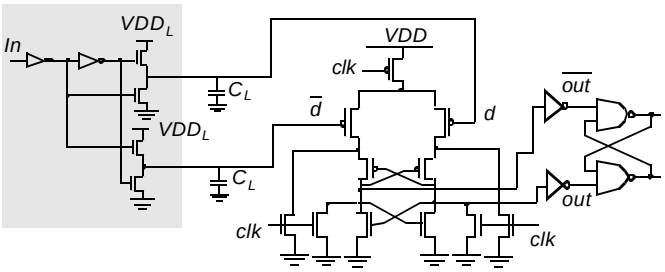


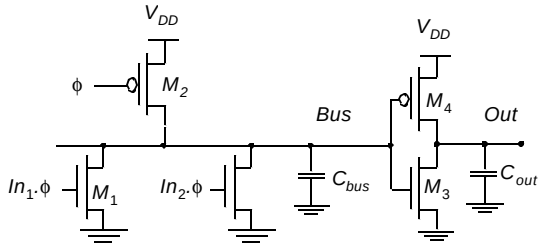
Figure 9.33 Differential reduced-swing interconnect system using an dditional supply rail at the driver. The receiver is a clocked differential flip-flop [Burd00].

Dynamic Reduced-Swing Networks

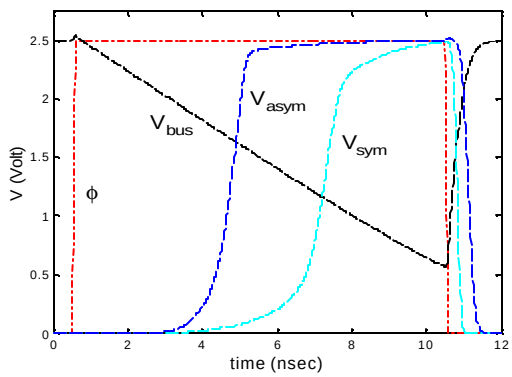
Another approach to speeding up the response of large fan-in circuits such as buses is to make use of precharging, an example of which is shown in Figure 9.34. During $\phi = 0$, the bus wire is precharged to V_{DD} through transistor M_2. Because this device is shared by all input gates, it can be made large enough to ensure a fast precharging time. During $\phi = 1$, the bus capacitance is conditionally discharged by one of the pull-down transistors. This operation is slow because the large capacitance C_{bus} must be discharged through the small pull-down device M_1.

A speed-up at the expense of noise margin can be obtained by observing that all transitions on the bus are from high-to-low during evaluation. A faster response can be obtained by moving the switching threshold of the subsequent inverter upwards. This results in an asymmetrical gate. In a traditional inverter design, M_3 and M_4 are sized so that t_{pHL} and t_{pLH} are identical, and the switching threshold (V_M) of the inverter is situated around 0.5 V_{DD}. This means that the bus voltage V_{bus} has to drop over $V_{DD}/2$ before the output inverter switches. This can be avoided by making the PMOS device larger, moving V_M upwards. This causes the output buffer to start switching earlier.

The precharged approach can result in a substantial speed-up for the driving of large capacitive lines. However, it also suffers from all the disadvantages of dynamic circuit techniques—charge-sharing, leakage, and inadvertent charge loss as a result of cross talk and noise. Cross talk between neighboring wires is especially an issue in densely wired bus networks. The reduced noise margin NM_H of the asymmetrical read-out inverter makes this circuit particularly sensitive to these parasitic effects. Extreme caution and



(a)Circuit diagram.



(b)Simulated response for both symmetrical and asymmetrical read-out inverters. M_4 is made fifteen times wider in the asymmetrical case. The load capacitance equals 1 pF.

Figure 9.34 Precharged bus.

extensive simulation is required when designing large precharged networks. However, the speed benefit is sometimes worth the effort.

The simulated response of a precharged bus network is shown in Figure 9.34b. The output signal is plotted for both symmetrical and asymmetrical output inverters. Skewing the switching threshold upwards reduces the propragation delay with more than 2.5 nsec. This allows for a further reduction in the evaluation period, which reduces the voltage swing of the bus with 0.6 V and the energy consumption with 18%. Careful timing of the precharge/evaluate signals is required to reap the maximum benefits of this scheme.

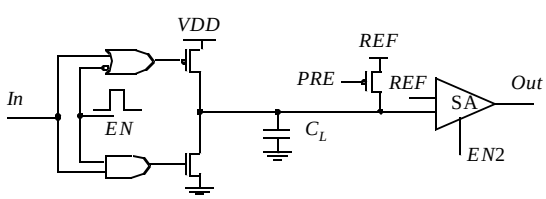


Figure 9.35 Pulse-controlled driver with sense amplifier

A variant on the dynamic theme is offered by the pulse-controlled driver scheme of Figure 9.35. The idea is to control the (dis)charging time of the drivers so that a desired swing is obtained on the interconnect. The interconnect wire is precharged to a reference voltage *REF*, typically situated at *VDD*/2. The receiver consists of a differential sense amplifier, which compares the voltage on the interconnect wire to REF. Since the amplifier consumes static power, it should only be enable for a short while. The advantage of this circuit is that the pulse width can be fine-tuned to realize a very-low swing, while no extra voltage supply is needed. This concept has been widely applied in memory designs. However, it only works well in the cases

when the capacitive loads are well-known beforehand. Furthermore, the wire is floating when the driver is disabled, making it susceptible to noise.

The circuits detailed above present only a small sub-group of the many reduced-swing circuits that have been devised. A more complete overview can be found in [Zhang00]. When venturing into the area of reduced-swing interconnect circuits, you should always be aware of the trade-off's involved: power and performance versus signal integrity and reliability.

9.5.2 Current-Mode Transmission Techniques

All the approaches presented above assume that the data to be transmitted over the wire is represented by a set of voltage levels, referenced to the supply rails. While this approach conforms with the logical levels we typically use in digital logic, it does not necessarily represent the best solution from a performance, power, and reliabilty perspective [Dally98]. Consider the problem of transmitting a bit over a lengthy transmission line with a characteristic impedance of 50 Ω. The traditional voltage-mode approach is depicted in a. The driver switches the line between the two supply lines, that represent a **1** and a **0**, respectively, and has an impedance of R_{out}. The receiver is a CMOS inverter that compares the incoming voltage with a power-supply referenced threshold voltage, typically centered in the middle of the two supply rails. The signal swing is lower-bounded by noise considerations. More specifically, power supply noise has a large impact as it impact both the signal levels and the switching threshold of the receiver. The latter is also a strong function of manufacturing process variations.

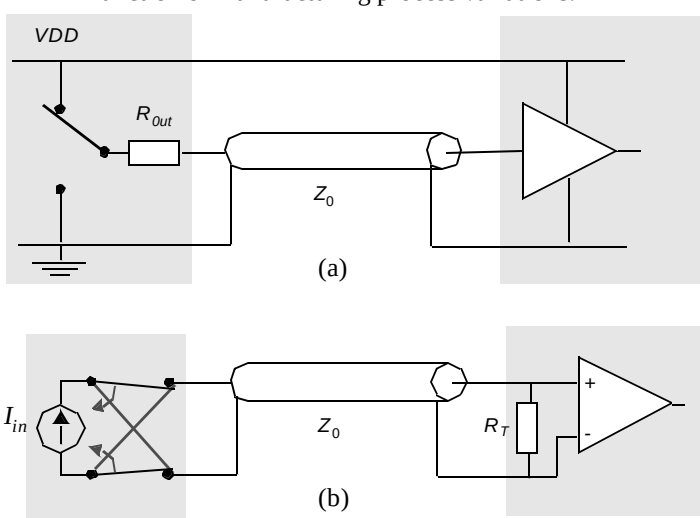


Figure 9.36 Voltage- versus current-mode transmission systems. The latter has the advantage of higher noise immunity with respect to supply noise.

Another option is to use a low-swing current-mode transmission system as is shown in b. The driver injects a current I_{in} into the line for a **1** and a reserve current of $-I_{in}$ for a **0**. This induces a voltage wave of $2 \times I_{in} \times Z_0$ into the transmission line, which ultimately gets absorbed into the parallel termination resistance R_T. The convergence time depends on how well the termination resistance is matched to the characteristic impedance of the

transmission line. A differential amplifier is used to the detect the voltage changes over R_T. Observe that the signal and its return path are both isolated from the supply rails and the associated noise, making all supply-noise common-mode to the differential receiver. Analog designers will testify that this type of noise is easily suppressed in any decent differential amplifier design.

While both approaches can be tuned to accomplish approximately the same performance levels, the current-mode approach holds a definite edge in terms of (dynamic) power dissipation. Due to its immunity to power-supply noise, it can operate at a much lower noise margin than the voltage-mode network, and hence also at a much lower swing. The value of the voltage wave propagating over the transmission line can be as low as 100 mV. The main challenge in designing an efficient current mode-circuit is the static power consumption. This is not an issue in ultra high-speed networks, where dynamic power tends to play the major role. Based on these observations, current-mode CMOS transmission systems have become popular in the area of high-speed off-chip interconnections. We would not be surprised to see them emerge on-chip as well in the foreseeable future.

9.6 Perspective: Networks-on-a-Chip

With chip sizes and clock rates continuing to increase, and feature sizes getting smaller at a constant rate, it is quite simple to conjecture that interconnect problems will be with us for some time to come. Physics constraints as such as the speed of light and thermal noise determinate how fast and how reliably a communication over a “long” distance can be established. As communication bottlenecks have been the major damper on hyper-supercomputer performance, similar constraints are starting to hamper integrated systems (“systems-on-a-chip”). New solutions conceived at the technology and circuit level only help to temporarily postpone the problems. The only and ultimate solution is to address interconnections on a chip as a communication problem, and to apply techniques and approaches that have made large-world communications and networking systems work reliably and correctly for quite some time. For example, the fact that the Internet is working correctly given it spanning scope and the numbers of connection points is quite amazing. The secret to its success is a well-thought-out protocol stack that isolates and orthoganizes the various functionality, performance, and reliability concenrns. So, rather than considering on-chip interconnections as point-to-point wires, they should be abstracted as communication channels over which we want to transmit data within a “quality-of-service” setting, putting constraints on throughtput, latency, and correctness [Sgroi01].

As an example, today’s on-chip interconnect signalling techniques are designed with noise margins large enough to ensure that a bit will always be transmitted correctly. It is probably safe to conjecture that future designs might take a more extreme tack: give up on integrity all together for the sake of energy/performance, and allow errors to occur on the transmitted signals. These errors can be taken care off by other circuitry, that provides error-correcting capabilities and/or retransmit corrupted data.

On a higher level of abstraction, we already are witnessing the emergence of complete “networks” on a chip. Rather than being statically wired from source-to-destination,

data is injected as packets into a complete network of wires, switches, and routers, and it is the network that dynamically decides how and when to route these packets through its segments [Dally01]. Ultimately, this is the only approach that can reliably work when the discrepancy between device sizes and on-chip distances becomes macroscopic.

9.7 Chapter Summary

This chapter has introduced a number of techniques to cope with the impact of interconnect on the performance and reliability of digital integrated circuits. The parasitics introduced by the interconnect have a dual effect on circuit operation: (1) they introduce noise and (2) they increase the propagation delay and power dissipation.

- Capacitive cross-talk in dense wire networks affects the reliability of the system, and impacts the performance. Careful design using well-behaved regular structures or using advanced design-automation tools is a necessity. Providing the necessary shielding is important for wires such as busses or clock signals.
- Driving large capacitances rapidly in CMOS requires the introduction of a *cascade of buffer stages* that must be carefully sized. More advanced techniques include the lowering of the signal-swing on long wires, and the use of current-mode signalling.
- Resistivity affects the reliability of the circuit by introducing *RI* drops. This is especially important for the supply network, where wire sizing is important.
- The extra delay introduced by the *rc* effects can be minimized by partitioning of the wire and by using a better interconnect technology.
- The inductance of the interconnect becomes important at higher switching speeds. The chip package is currently one of the most important contributors of inductance. Novel packaging techniques are gaining importance with faster technologies.
- Ground bounce introduced by the $L\ di/dt$ voltage drop over the supply wires is one of the most important sources of noise in current integrated circuits. Ground bounce can be reduced by providing sufficient supply pins and by controlling the slopes of the off-chip signals.
- Transmission line effects are rapifly becoming an issue in super-GHz designs. Providing the correct termination is the only means of dealing with the transmission line delay.

9.8 To Probe Further

Excellent overviews of the issues involved in dealing with interconnect in digital designs can be found in [Dally98] and [Bakoglu90]. Chandrakasan book - others? journals?

REFERENCES

[Abo92] A. Abo and A. Behzad, "Interconnect Driver Design," EE141 Project Report, Univ. of California—Berkeley, 1992.

[Bakoglu90] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990.

[Brews89] J. Brews, "Electrical Modeling of Interconnections," in *SubMicron Integrated Circuits*, ed. R. Watts, John Wiley & Sons, pp. 269–331, 1989.

[Doane93] D. Doane, ed., *Multichip Module Technologies and Alternatives*, Van Nostrand-Reinhold, 1993.

[Dopperpuhl92] D. Dopperpuhl et al., "A 200 MHz 64-b Dual Issue CMOS Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 27, no. 11, pp. 1555–1567, November 1992.

[Elmore48] E. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, pp. 55–63, January 1948.

[Etter93] D. Etter, "Engineering Problem Solving with Matlab," Prentice Hall, 1993.

[Franzon93] P. Franzon, "Electrical Design of Digital Multichip Modules," in [Doane93], pp 525–568, 1993.

[Heller75] L. Heller et al., "High-Sensitivity Charge-Transfer Sense Amplifier," *Proceedings ISSCC Conf.*, pp. 112-113, 1975.

[Hedenstierna87] N. Hedenstierna et al., "CMOS Circuit Speed and Buffer Optimization," *IEEE Trans. on Computer-Aided Design*, vol CAD-6, no. 2, pp. 270–281, March 1987.

[Horowitz83] M. Horowitz, "Timing Models for MOS Circuits," Ph.D. diss., Stanford University, 1983.

[Johnson93] H. Johnson and M. Graham, *High-Speed Digital Design—A Handbook of Black Magic*, Prentice Hall, 1993.

[Kang87] S. Kang, "Metal-Metal Matrix (M^3) for High-Speed MOS VLSI Layout," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 886–891, September 1987.

[Landman71] B. Landman and R. Russo, "On a Pin versus Block Relationship for Partitions of Logic Graphs," *IEEE Trans. on Computers*, vol. C-20, pp. 1469–1479, December 1971.

[Masaki92] A. Masaki, "Deep-Submicron CMOS Warms Up to High-Speed Logic," *Circuits and Devices Magazine*, Nov. 1992.

[Mead80] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[Nagata92] M. Nagata, "Limitations, Innovations, and Challenges of Circuits and Devices into a Half Micrometer and Beyond," *IEEE Journal of Solid State Circuits*, vol. 27, no. 4, pp. 465–472, April 1992.

[Rubinstein83] J. Rubinstein, P. Penfield, and M. Horowitz, "Signal Delay in *RC* Networks," *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.

[Schaper83] L. Schaper and D. Amey, "Improved Electrical Performance Required for Future MOS Packaging," *IEEE Trans. on Components, Hybrids and Manufacturing Technology*, vol. CHMT-6, pp. 282–289, September 1983.

[Solomon82] P. Solomon, "A Comparison of Semiconductor Devices for High-Speed Logic," *Proc. of the IEEE*, vol. 70, no. 5, May 1982.

[Sorkin87] G. Sorkin, "Asymptotically Perfect Trivial Global Routing: A Stochastic Analysis," *IEEE Trans. on Computer-Aided Design*, vol. CAD-6, p. 820, 1987.

[Steidel83] C. Steidel, "Assembly Techniques and Packaging," in [Sze83], pp. 551–598, 1983.

[Sze81] S. Sze, *Physics of Semi-Conductor Devices*, 2nd ed., Wiley, 1981.

[Sze83] S. Sze, ed., *VLSI Technology*, McGraw-Hill, 1983.

[Tao94] J. Tao, N. Cheung, and C. Hu, "An Electromigration Failure Model for Interconnects under Pulsed and Bidirectional Current Stressing," *IEEE Trans. on Devices*, vol. 41, no. 4, pp. 539–545, April 1994.

[Tewksbury94] S. Tewksbury, ed., *Microelectronics System Interconnections—Performance and Modeling*, IEEE Press, 1994.

[Vaidya80] S. Vaidya, D. Fraser, and A. Sinha, "Electromigration Resistance of Fine-Line Al," *Proc. 18th Reliability Physics Symposium*, IEEE, p. 165, 1980.

[Vdmeijs84] N. Van De Meijs and J. Fokkema, "VLSI Circuit Reconstruction from Mask Topology," *Integration*, vol. 2, no. 2, pp. 85–119, 1984.

[Wada92] O. Wada, T. Kamijoh, and M. Nakamura, "Integrated Optical Connections," *IEEE Circuits and Devices Magazine*, vol. 8, no. 6, pp. 37–42, 1992.

[Weste93] N. Weste and K.Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective*, 2nd ed., Addison-Wesley, 1993.

[Yoshino90] T. Yoshino et al., "A 100 MHz 64-tap FIR Digital Filter in a 0.8 μm BiCMOS Gate Array," *IEEE Journal of Solid State Circuits*, vol. 25, no. 6, pp. 1494–1501, December 1990.

CHAPTER

10

TIMING ISSUES IN DIGITAL CIRCUITS

Impact of clock skew and jitter on performance and functionality

n

Alternative timing methodologies

n

Synchronization issues in digital IC and board design

n

Clock generation

10.1 Introduction

10.2 Classification of Digital Systems

10.3 Synchronous Design — An In-depth Perspective

10.3.1 Synchronous Timing Basics

10.3.2 Sources of Skew and Jitter

10.3.3 Clock-Distribution Techniques

10.3.4 Latch-Based Clocking

10.4 Self-Timed Circuit Design*

10.5 Synchronizers and Arbiters*

10.6 Clock Synthesis and Synchronization Using a Phase-Locked Loop

10.7 Future Directions

10.8 Perspective: Synchronous versus Asynchronous Design

10.9 Summary

10.10 To Probe Further

10.1 Introduction

All sequential circuits have one property in common—a well-defined ordering of the switching events must be imposed if the circuit is to operate correctly. If this were not the case, wrong data might be written into the memory elements, resulting in a functional failure. The *synchronous* system approach, in which all memory elements in the system are simultaneously updated using a globally distributed periodic synchronization signal (that is, a global clock signal), represents an effective and popular way to enforce this ordering. Functionality is ensured by imposing some strict contraints on the generation of the clock signals and their distribution to the memory elements distributed over the chip; non-compliance often leads to malfunction.

This Chapter starts with an overview of the different timing methodologies. The majority of the text is devoted to the popular synchronous approach. We analyze the impact of spatial variations of the clock signal, called *clock skew*, and temporal variations of the clock signal, called *clock jitter*, and introduce techniques to cope with it. These variations fundamentally limit the performance that can be achieved using a conventional design methodology.

At the other end of the design spectrum is an approach called *asynchronous design*, which avoids the problem of clock uncertainty all-together by eliminating the need for globally-distributed clocks. After discussing the basics of *asynchronous design* approach, we analyze the associated overhead and identify some practical applications. The important issue of *synchronization*, which is required when interfacing different clock domains or when sampling an asynchronous signal, also deserves some in-depth treatment. Finally, the fundamentals of on-chip clock generation using feedback is introduced along with trends in timing.

10.2 Classification of Digital Systems

In digital systems, signals can be classified depending on how they are related to a local clock [Messerschmitt90][Dally98]. Signals that transition only at predetermined periods in time can be classified as synchronous, mesochronous, or plesiochronous with respect to a system clock. A signal that can transition at arbitrary times is considered asynchronous.

10.2.1 Synchronous Interconnect

A synchronous signal is one that has the exact same frequency, and a known fixed phase offset with respect to the local clock. In such a timing methodology, the signal is “synchronized” with the clock, and the data can be sampled directly without any uncertainty. In digital logic design, synchronous systems are the most straightforward type of interconnect, where the flow of data in a circuit proceeds in lockstep with the system clock as shown below.

Here, the input data signal *In* is sampled with register R_1 to give signal C_{in}, which is synchronous with the system clock and then passed along to the combinational logic block. After a suitable setting period, the output C_{out} becomes valid and can be sampled by

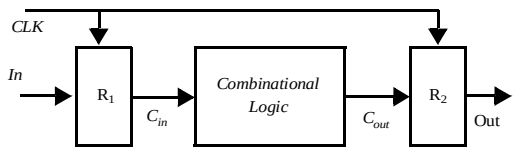


Figure 10.1 Synchronous interconnect methodology.

R_2 which synchronizes the output with the clock. In a sense, the "certainty period" of signal C_{out}, or the period where data is valid is synchronized with the system clock, which allows register R_2 to sample the data with complete confidence. The length of the "uncertainty period," or the period where data is not valid, *places an upper bound on how fast a synchronous interconnect system can be clocked*.

10.2.2 Mesochronous interconnect

A mesochronous signal is one that has the same frequency but an unknown phase offset with respect to the local clock ("meso" from Greek is middle). For example, if data is being passed between two different clock domains, then the data signal transmitted from the first module can have an unknown phase relationship to the clock of the receiving module. In such a system, it is not possible to directly sample the output at the receiving module because of the uncertainty in the phase offset. A (mesochronous) synchronizer can be used to synchronize the data signal with the receiving clock as shown below. The synchronizer serves to adjust the phase of the received signal to ensure proper sampling.

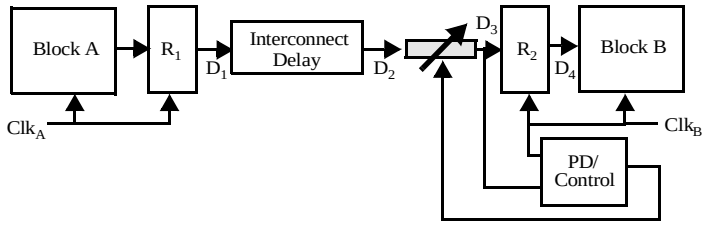


Figure 10.2 Mesochronous communication approach using variable delay line.

In Figure 10.2, signal D_1 is synchronous with respect to Clk_A. However, D_1 and D_2 are mesochronous with Clk_B because of the unknown phase difference between Clk_A and Clk_B and the unknown interconnect delay in the path between Block A and Block B. The role of the synchronizer is to adjust the variable delay line such that the data signal D_3 (a delayed version of D_2) is aligned properly with the system clock of block B. In this example, the variable delay element is adjusted by measuring the phase difference between the received signal and the local clock. After register R_2 samples the incoming data during the certainty period, then signal D_4 becomes synchronous with Clk_B.

10.2.3 Plesiochronous Interconnect

A plesiochronous signal is one that has nominally the same, but slightly different frequency as the local clock ("plesio" from Greek is near). In effect, the phase difference

drifts in time. This scenario can easily arise when two interacting modules have independent clocks generated from separate crystal oscillators. Since the transmitted signal can arrive at the receiving module at a different rate than the local clock, one needs to utilize a buffering scheme to ensure all data is received. Typically, plesiochronous interconnect only occurs in distributed systems like long distance communications, since chip or even board level circuits typically utilize a common oscillator to derive local clocks. A possible framework for plesiochronous interconnect is shown in Figure 10.3.

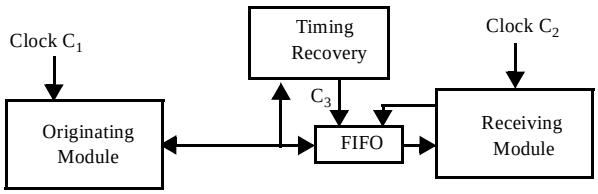


Figure 10.3 Plesiochronous communications using FIFO.

In this digital communications framework, the originating module issues data at some unknown rate characterized by C_1, which is plesiochronous with respect to C_2. The timing recovery unit is responsible for deriving clock C_3 from the data sequence, and buffering the data in a FIFO. As a result, C_3 will be synchronous with the data at the input of the FIFO and will be mesochronous with C_1. Since the clock frequencies from the originating and receiving modules are mismatched, data might have to be dropped if the transmit frequency is faster, and data can be duplicated if the transmit frequency is slower than the receive frequency. However, by making the FIFO large enough, and periodically resetting the system whenever an overflow condition occurs, robust communication can be achieved.

10.2.4 Asynchronous Interconnect

Asynchronous signals can transition at any arbitrary time, and are not slaved to any local clock. As a result, it is not straightforward to map these arbitrary transitions into a synchronized data stream. Although it is possible to synchronize asynchronous signals by detecting events and introducing latencies into a data stream synchronized to a local clock, a more natural way to handle asynchronous signals is to simply eliminate the use of local clocks and utilize a self-timed asynchronous design approach. In such an approach, communication between modules is controlled through a handshaking protocol to perform the proper ordering of commands.

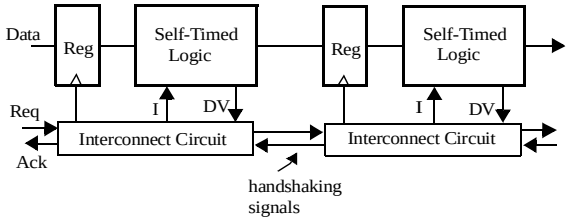


Figure 10.4 Asynchronous design methodology for simple pipeline interconnect.

When a logic block completes an operation, it will generate a completion signal *DV* to indicate that output data is valid. The handshaking signals then initiate a data transfer to the next block, which latches in the new data and begins a new computation by asserting the initialization signal *I*. Asynchronous designs are advantageous because computations are performed at the native speed of the logic, where block computations occur whenever data becomes available. There is no need to manage clock *skew*, and the design methodology leads to a very modular approach where interaction between blocks simply occur through a handshaking procedure. However, these handshaking protocols result in increased complexity and overhead in communication that can reduce performance.

10.3 Synchronous Design — An In-depth Perspective

10.3.1 Synchronous Timing Basics

Virtually all systems designed today use a periodic *synchronization* signal or clock. The generation and distribution of a clock has a significant impact on performance and power dissipation. For a positive *edge-triggered* system, the rising edge of the clock is used to denote the beginning and completion of a clock cycle. In the ideal world, assuming the clock paths from a central distribution point to each register are perfectly balanced, the phase of the clock (i.e., the position of the clock edge relative to a reference) at various points in the system is going to be exactly equal. However, the clock is neither perfectly periodic nor perfectly simultaneous. This results in performance degradation and/or circuit malfunction. Figure 10.5 shows the basic structure of a synchronous pipelined datapath. In

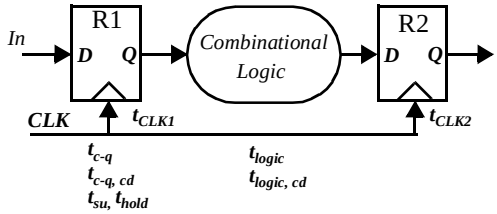


Figure 10.5 Pipelined Datapath Circuit and timing parameters.

the ideal scenario, the clock at registers 1 and 2 have the same clock period and transition at the exact same time. The following timing parameters characterize the timing of the sequential circuit.

- The contamination (minimum) delay $t_{c\text{-}q,cd}$, and maximum propagation delay of the register $t_{c\text{-}q}$.
- The set-up (t_{su}) and hold time (t_{hold}) for the registers.
- The contamination delay $t_{logic,cd}$ and maximum delay t_{logic} of the combinational logic.

- t_{clk1} and t_{clk2}, corresponding to the position of the rising edge of the clock relative to a global reference.

Under ideal conditions ($t_{clk1} = t_{clk2}$), the worst case propagation delays determine the minimum clock period required for this sequential circuit. The period must be long enough for the data to propagate through the registers and logic and be set-up at the destination register before the next rising edge of the clock. This constraint is given by (as derived in Chapter 7):

$$T > t_{c-q} + t_{logic} + t_{su} \tag{10.1}$$

At the same time, the hold time of the destination register must be shorter than the minimum propagation delay through the logic network,

$$t_{hold} < t_{c-q,cd} + t_{logic,cd} \tag{10.2}$$

The above analysis is simplistic since the clock is never ideal. As a result of process and environmental variations, the clock signal can have *spatial* and *temporal* variations.

Clock Skew

The spatial variation in arrival time of a clock transition on an integrated circuit is commonly referred to as *clock skew*. The *clock skew* between two points *i* and *j* on a IC is given by $\delta(i,j) = t_i - t_j$, where t_i and t_j are the position of the rising edge of the clock with respect to a reference. Consider the transfer of data between registers *R*1 and *R2* in Figure 10.5. The clock skew can be positive or negative depending upon the routing direction and position of the clock source. The timing diagram for the case with positive skew is shown in Figure 10.6. As the figure illustrates, the rising clock edge is delayed by a positive δ at the second register.

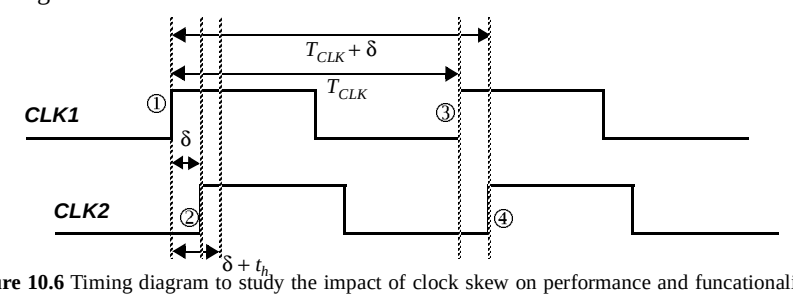


Figure 10.6 Timing diagram to study the impact of clock skew on performance and funcationality. In this sample timing diagram, $\delta > 0$.

Clock skew is caused by static path-length mismatches in the clock load and by definition skew is constant from cycle to cycle. That is, if in one cycle *CLK2* lagged *CLK1* by δ, then on the next cycle it will lag it by the same amount. It is important to note that clock skew does not result in clock period variation, but rather phase shift.

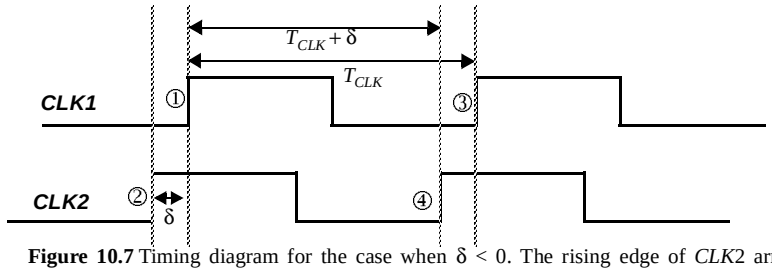


Figure 10.7 Timing diagram for the case when $\delta < 0$. The rising edge of *CLK*2 arrives earlier than the edge of *CLK*1.

Skew has strong implications on performance and functionality of a sequential system. First consider the impact of clock skew on performance. From Figure 10.6, a new input In sampled by *R1* at edge ① will propagate through the combinational logic and be sampled by *R2* on edge ④. If the clock skew is positive, the time available for signal to propagate from *R1* to *R2* is increased by the skew δ. The output of the combinational logic must be valid one set-up time before the rising edge of *CLK2* (point ④). The constraint on the minimum clock period can then be derived as:

$$T + \delta \geq t_{c-q} + t_{logic} + t_{su} \qquad \text{or} \qquad T \geq t_{c-q} + t_{logic} + t_{su} - \delta \tag{10.3}$$

The above equation suggests that clock skew actually has the potential to improve the performance of the circuit. That is, the minimum clock period required to operate the circuit reliably reduces with increasing clock skew! This is indeed correct, but unfortunately, increasing skew makes the circuit more susceptible to race conditions may and harm the correct operation of sequential systems.

As above, assume that input *In* is sampled on the rising edge of *CLK*1 at edge ① into *R*1. The new values at the output of *R*1 propagates through the combinational logic and should be valid before edge ④ at *CLK*2. However, if the minimum delay of the combinational logic block is *small*, the inputs to *R*2 may change before the clock edge ②, resulting in incorrect evaluation. To avoid races, we must ensure that the minimum propagation delay through the register and logic must be long enough such that the inputs to *R*2 are valid for a hold time after edge ②. The constraint can be formally stated as

$$\delta + t_{hold} < t_{(c-q,\,cd)} + t_{(\log ic,\,cd)}$$
$$\text{or}$$
$$\delta < t_{(c-q,\,cd)} + t_{(\log ic,\,cd)} - t_{hold} \tag{10.4}$$

Figure 10.7 shows the timing diagram for the case when $\delta < 0$. For this case, the rising edge of *CLK*2 happens before the rising edge of *CLK*1. On the rising edge of *CLK*1, a new input is sampled by *R*1. The new sampled data propagates through the combinational logic and is sampled by *R*2 on the rising edge of *CLK*2, which corresponds to edge ④. As can be seen from Figure 10.7 and Eq. (10.3), a negative skew directly impacts the performance of sequential system. However, a negative skew implies that the system never fails, since edge ② happens before edge ①! This can also be seen from Eq. (10.4), which is always satisfied since $\delta < 0$.

Example scenarios for positive and negative clock skew are shown in Figure 10.8.

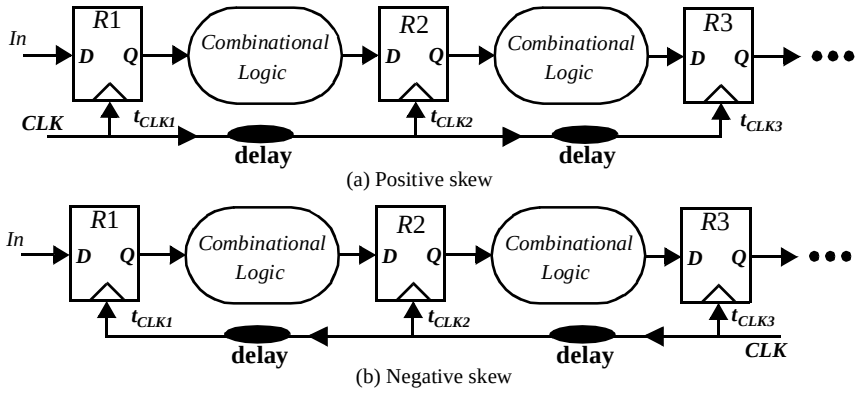


Figure 10.8 Positive and negative clock skew.

- $\delta > \mathbf{0}$—This corresponds to a clock routed in the same direction as the flow of the data through the pipeline (Figure 10.8a). In this case, the skew has to be strictly controlled and satisfy Eq. (10.4). If this constraint is not met, the circuit does malfunction **independent of the clock period**. Reducing the clock frequency of an edge-triggered circuit does not help get around skew problems! On the other hand, positive skew increases the throughput of the circuit as expressed by Eq. (10.3), because the clock period can be shortened by δ. The extent of this improvement is limited as large values of δ soon provoke violations of Eq. (10.4).

- $\delta < \mathbf{0}$—When the clock is routed in the opposite direction of the data (Figure 10.8b), the skew is negative and condition (10.4) is unconditionally met. The circuit operates correctly independent of the skew. The skew reduces the time available for actual computation so that the clock period has to be increased by $|\delta|$. In summary, routing the clock in the opposite direction of the data avoids disasters but hampers the circuit performance .

Unfortunately, since a general logic circuit can have data flowing in both directions (for example, circuits with feedback), this solution to eliminate races will not always work (Figure 10.9). The skew can assume both positive and negative values depending on the direction of

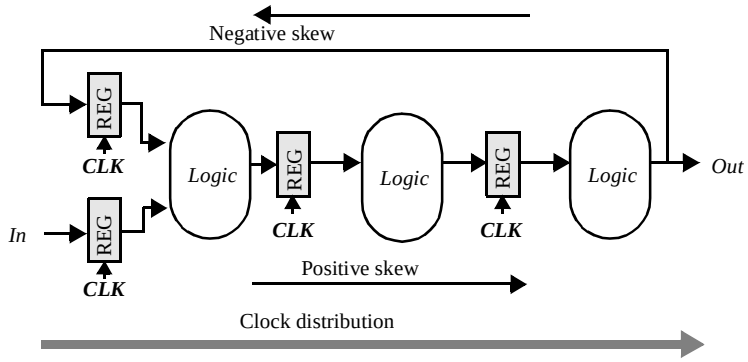


Figure 10.9 Datapath structure with feedback.

the data transfer. Under these circumstances, the designer has to account for the worst-case skew condition. In general, routing the clock so that only negative skew occurs is not feasible. Therefore, the design of a low-skew clock network is essential.

Example 10.1 Propagation and Contamination Delay Estimation

Consider the logic network shown in Figure 10.10. Determine the propagation and contamination delay of the network, assuming that the worst case gate delay is t_{gate}. The maximum and minimum delays of the gates is made, as they are assumed to be identical.

The contamination delay is given by 2 t_{gate} (the delay through OR_1 and OR_2). On the other hand, computation of the worst case propagation delay is not as simple as it appears. At first glance, it would appear that the worst case corresponds to path ① and the delay is $5t_{gate}$. However, when analyzing the data dependencies, it becomes obvious that path ① is never exercised. Path ① is called a *false path*. If $A = 1$, the critical path goes through OR_1 and OR_2. If $A = 0$ and $B = 0$, the critical path is through I_1,OR_1 and OR_2 (corresponding to a delay of 3 t_{gate}). For the case when $A= 0$ and $B =1$, the critical path is through I_1,OR_1, AND_3 and OR_2. In other words, for this simple (but contrived) logic circuit, the output does not even depend on inputs C and D (that is, there is redundancy). Therefore, the propagation delay is 4 t_{gate}. Given the propagation and contamination delay, the minimum and maximum allowable skew can be easily computed.

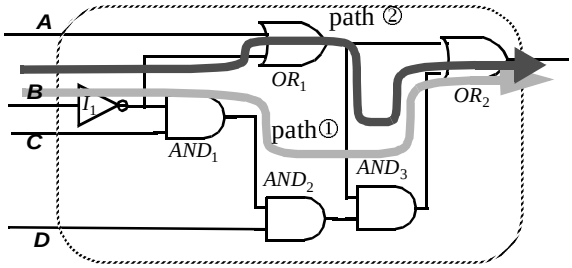


Figure 10.10 Logic network for computation of performance.

WARNING: The computation of the worst-case propagation delay for combinational logic, due to the existence of *false paths*, cannot be obtained by simply adding the propagation delay of individual logic gates. The critical path is strongly dependent on circuit topology and data dependencies.

Clock Jitter

Clock jitter refers to the temporal variation of the clock period at a given point — that is, the clock period can reduce or expand on a cycle-by-cycle basis. It is strictly a temporal uncertainty measure and is often specified at a given point on the chip. Jitter can be measured and cited in one of many ways. *Cycle-to-cycle jitter* refers to time varying deviation

of a single clock period and for a given spatial location *i* is given as $T_{jitter,i}(n) = T_{i,\,n+1} - T_{i,n} - T_{CLK}$, where $T_{i,n}$ is the clock period for period *n*, $T_{i,\,n+1}$ is clock period for period *n*+1, and T_{CLK} is the nominal clock period.

Jitter directly impacts the performance of a sequential system. Figure 10.11 shows the nominal clock period as well as variation in period. Ideally the clock period starts at edge ②and ends at edge ⑤ and with a nominal clock period of T_{CLK}. However, as a result of *jitter*, the worst case scenario happens when the leading edge of the current clock period is delayed (edge ③), and the leading edge of the next clock period occurs early (edge ④). As a result, the total time available to complete the operation is reduced by 2 t_{jiiter} in the worst case and is given by

$$T_{CLK} - 2t_{jitter} \geq t_{c-q} + t_{logic} + t_{su} \text{ or } \quad T \geq t_{c-q} + t_{logic} + t_{su} + 2t_{jitter} \tag{10.5}$$

The above equation illustrates that *jitter* directly reduces the performance of a sequential circuit. Care must be taken to reduce jitter in the clock network to maximize performance.

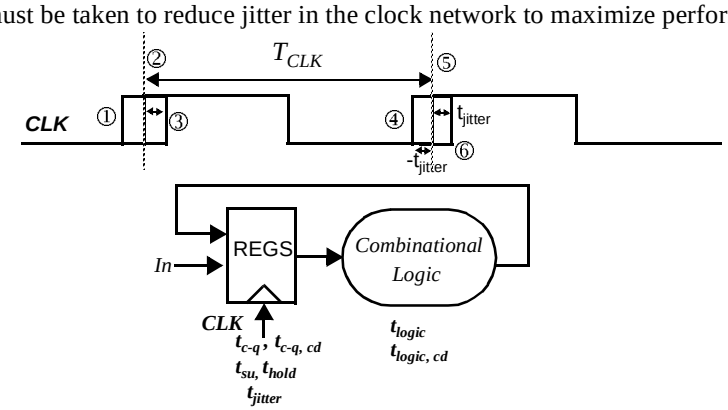


Figure 10.11 Circuit for studying the impact of *jitter* on performance.

Impact of Skew and Jitter on Performance

In this section, the combined impact of *skew* and *jitter* is studied with respect to conventional *edge-triggered* clocking. Consider the sequential circuit show in Figure 10.12.

Assume that nominally ideal clocks are distributed to both registers (the clock period is identical every cycle and the *skew* is 0). In reality, there is static *skew* δ between the two clock signals (assume that δ > 0). Assume that *CLK1* has a jitter of $t_{jitter1}$ and CLK2 has a jitter of $t_{jitter2}$. To determine the constraint on the minimum clock period, we must look at the minimum available time to perform the required computation. The worst case happen when the leading edge of the current clock period on *CLK1* happens late (edge ③) and the leading edge of the next cycle of *CLK2* happens early (edge ⑩). This results in the following constraint

$$T_{CLK} + \delta - t_{jitter1} - t_{jitter2} \geq t_{c-q} + t_{logic} + t_{su}$$
$$\text{or} \quad T \geq t_{c-q} + t_{logic} + t_{su} - \delta + t_{jitter1} + t_{jitter2} \tag{10.6}$$

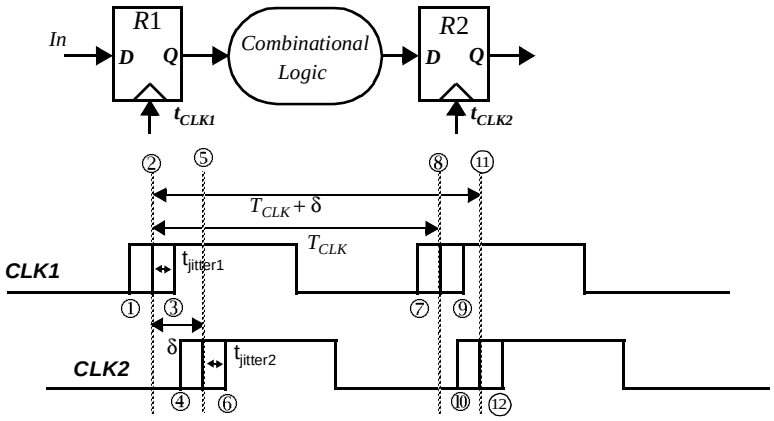


Figure 10.12 Sequential circuit to study the impact of skew and jitter on *edge-triggered* systems. In this example, a positive *skew* (δ) is assumed.

As the above equation illustrates, while positive *skew* can provide potential performance advantage, *jitter* has a negative impact on the minimum clock period. To formulate the minimum delay constraint, consider the case when the leading edge of the *CLK1* cycle arrives early (edge ①) and the leading edge the current cycle of *CLK2* arrives late (edge ⑥). The separation between edge ① and ⑥ should be smaller than the minimum delay through the network. This results in

$$\delta + t_{hold} + t_{jitter1} + t_{jitter2} < t_{(c-q,\,cd)} + t_{(\log ic,\,cd)}$$

or

$$\delta < t_{(c-q,\,cd)} + t_{(\log ic,\,cd)} - t_{hold} - t_{jitter1} - t_{jitter2} \tag{10.7}$$

The above relation indicates that the acceptable skew is reduced by the jitter of the two signals.

Now consider the case when the skew is negative (δ <0) as shown in Figure 10.13. For the timing shown, $|\delta| > t_{jitter2}$. It can be easily verified that the worst case timing is exactly the same as the previous analysis, with δ taking a negative value. That is, negative skew reduces performance.

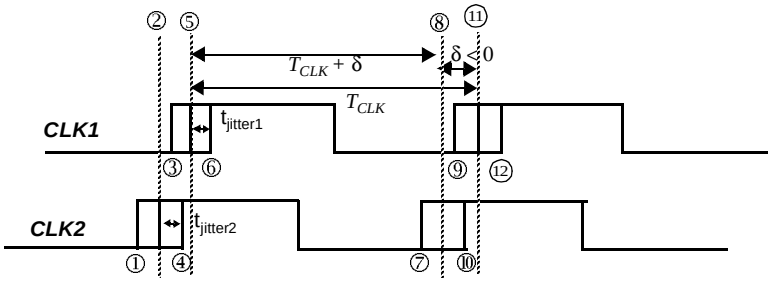


Figure 10.13 Consider a negative clock *skew* (δ) and the *skew* is assumed to be larger than the *jitter.*

10.3.2 Sources of Skew and Jitter

A perfect *clock* is defined as perfectly periodic signal that is simultaneous triggered at various memory elements on the chip. However, due to a variety of process and environmental variations, clocks are not ideal. To illustrate the sources of skew and jitter, consider the simplistic view of clock generation and distribution as shown in Figure 10.14. Typically, a high frequency clock is either provided from off chip or generated on-chip. From a central point, the clock is distributed using multiple *matched* paths to low-level memory element.s registers. In this picture, two paths are shown. The clock paths include wiring and the associated distributed buffers required to drive interconnects and loads. A key point to realize in clock distribution is that the **absolute delay through a clock distribution path is not important**; what matters is the relative arrival time between the output of each path at the register points (i.e., it is pefectly acceptable for the clock signal to take multiple cycles to get from a central distribution point to a low-level register as long as all clocks arrive at the same time to different registers on the chip).

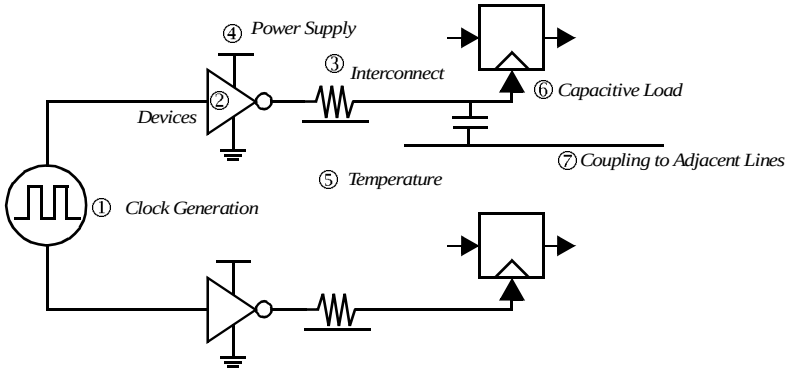


Figure 10.14 Skew and jitter sources in synchronous clock distribution.

The are many reasons why the two parallel paths don't result in exactly the same delay. The sources of clock uncertainty can be classified in several ways. First, errors can be divided into *systematic* or *random*. *Systematic* errors are nominally identical from chip to chip, and are typically predictable (e.g., variation in total load capacitance of each clock path). In principle, such errors can be modeled and corrected at design time given sufficiently good models and simulators. Failing that, systematic errors can be deduced from measurements over a set of chips, and the design adjusted to compensate. *Random* errors are due to manufacturing variations (e.g., dopant fluctuations that result in threshold variations) that are difficult to model and eliminate. Mismatch may also be characterized as *static* or *time-varying*. In practice, there is a continuum between changes that are slower than the time constant of interest, and those that are faster. For example, temperature variations on a chip vary on a millisecond time scale. A clock network tuned by a one-time calibration or trimming would be vulnerable to time-varying mismatch due to varying thermal gradients. On the other hand, to a feedback network with a bandwidth of several megahertz, thermal changes appear essentially static. For example, the clock net is usually by far the largest single net on the chip, and simultaneous transitions on the clock drivers induces noise on the power supply. However, this high speed effect does not contribute to

time-varying mismatch because it is the same on every clock cycle, affecting each rising clock edge the same way. Of course, this power supply glitch may still cause static mismatch if it is not the same throughout the chip. Below, the various sources of *skew* and *jitter*, introduced in Figure 10.14, are described in detail.

Clock-Signal Generation (1)

The generation of the clock signal itself causes *jitter.* A typical on-chip clock generator, as described at the end of this chapter, takes a low-frequency reference clock signal, and produces a high-frequency global reference for the processor. The core of such a generator is a Voltage-Controlled Oscillator (VCO). This is an analog circuit, sensitive to intrinsic device noise and power supply variations. A major problem is the coupling from the surrounding noisy digital circuitry through the substrate. This is particularly a problem in modern fabrication processes that combine a lightly-doped epitaxial layer and a heavily-doped substrate (to combat latch-up). This causes substrate noise to travel over large distances on the chip. These noise source cause temporal variations of the clock signal that propagate unfiltered through the clock drivers to the flip-flops, and result in *cycle-to-cycle* clock-period variations. This *jitter* causes performance degradation.

Manufacturing Device Variations (2)

Distributed buffers are integral components of the clock distribution networks, as they are required to drive both the register loads as well as the global and local interconnects. The matching of devices in the buffers along multiple clock paths is critical to minimizing timing uncertainty. Unfortunately, as a result of process variations, devices parameters in the buffers vary along different paths, resulting in static *skew.* There are many sources of variations including oxide variations (that affects the gain and threshold), dopant variations, and lateral dimension (width and length) variations. The doping variations can affect the depth of junction and dopant profiles and cause variations in electrical parameters such as device threshold and parasitic capacitances. The orientation of polysilicon can also have a big impact on the device parameters. Keeping the orientation the same across the chip for the clock drivers is critical.

Variation in the polysilicon critical dimension, is particularly important as it translates directly into MOS transistor channel length variation and resulting variations in the drive current and switching characteristics. Spatial variation usually consists of wafer-level (or within-wafer) variation and die-level (or within-die) variation. At least part of the variation is systematic and can be modeled and compensated for. The random variations however, ultimately limits the matching and *skew* that can be achieved.

Interconnect Variations (3)

Vertical and lateral dimension variations cause the interconnect capacitance and resistance to vary across a chip. Since this variation is static, it causes *skew* between different paths. One important source of interconnect variation is the *Inter-level Dielectric (ILD)* thickness variations. In the formation of aluminum interconnect, layers of silicon dioxide are interposed between layers of patterned metallization. The oxide layer is deposited over a layer of patterned metal features, generally resulting in some remaining step height or surface topography. *Chemical-mechanical polishing* (CMP) is used to "planarize" the surface and remove topography resulting from deposition and etch (as shown in Figure 10.15a). While at the feature scale (over an individual metal line), CMP can achieve excellent planarity, there are limitations on the planarization that can be achieved over a global range. This is primarily caused due to variations in polish rate that is a function of the circuit layout density and pattern effects. Figure 10.15b shows this effect where the polish rate is higher for the lower spatial density region, resulting in smaller dielectric thickness and higher capacitance.

The assessment and control of variation is of critical importance in semiconductor process development and manufacturing. Significant advances have been made to develop analytical models for estimating the ILD thickness variations based on spatial density. Since this component is often predictable from the layout, it is possible to actually correct for the systematic component at design time (e.g., by adding appropriate delays or making the density uniform by adding "dummy fills"). Figure 10.16 shows the spatial pattern density and ILD thickness for a high performance microprocessor. The graphs show that there is clear correlation between the density and the thickness of the dielectric. So clock distribution networks must exploit such information to reduce clock *skew.*

Other interconnect variations include deviation in the width of the wires and line spacing. This results from photolithography and etch dependencies. At the lower levels of metallization, lithographic effects are important while at higher levels etch effects are important that depend on width and layout. The width is a critical parameter as it directly impacts the resistance of the line and the wire spacing affects the wire-to-wire capaci-

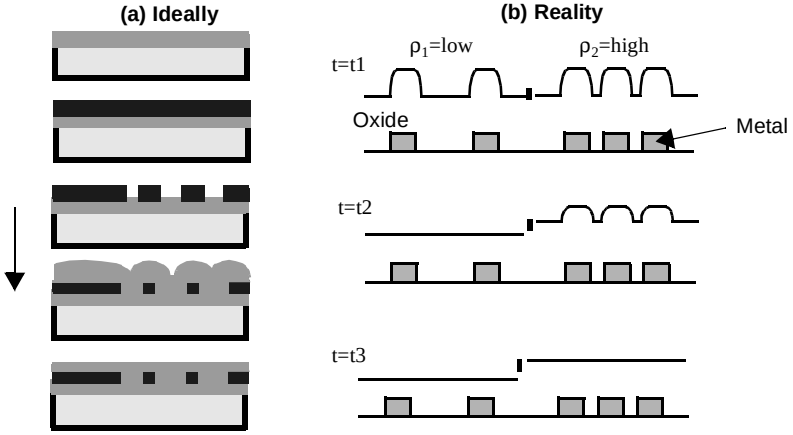
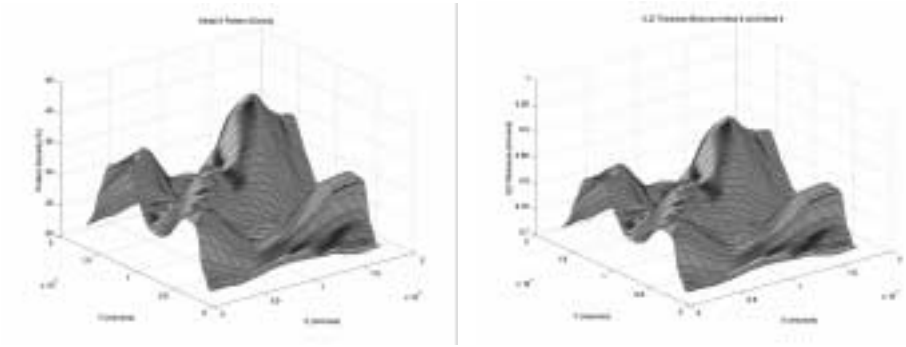


Figure 10.15 Inter-level Dielectric (ILD) thickness variation due to density (coutersy of Duane Boning).

tance, which the dominant component of capacitance. A detailed review of device and interconnect variations is presented in [Boning00].



(a) Pattern Density (b) ILD thickness variation

Figure 10.16 Pattern density and ILD thickness variation for a high performance microprocessor.

Environmental Variations (4 and 5)

Environmental variations are probably the most significant and primarily contribute to *skew* and *jitter*. The two major sources of environmental variations are temperature and power supply. Temperature gradients across the chip is a result of variations in power dissipation across the die. This has particularly become an issue with clock gating where some parts of the chip maybe idle while other parts of the chip might be fully active. This results in large temperature variations. Since the device parameters (such as threshold, mobility, etc.) depend strongly on temperature, buffer delay for a clock distribution network along one path can vary drastically for another path. More importantly, this component is time-varying since the temperature changes as the logic activity of the circuit varies. As a result, it is not sufficient to simulate the clock networks at worst case corners of temperature; instead, the worst-case variation in temperature must be simulated. An interesting question is does temperature variation contribute to *skew* or to *jitter*? Clearly the variation in temperature is time varying but the changes are relatively slow (typical time constants for temperature on the order of milliseconds). Therefore it usually considered as a skew component and the worst-case conditions are used. Fortunately, using feedback, it is possible to calibrate the temperature and compensate.

Power supply variations on the hand is the major source of *jitter* in clock distribution networks. The delay through buffers is a very strong function of power supply as it directly affects the drive of the transistors. As with temperature, the power supply voltage is a strong function of the switching activity. Therefore, the buffer delay along one path is very different than the buffer delay along another path. Power supply variations can be classified into static (or slow) and high frequency variations. Static power supply variations may result from fixed currents drawn from various modules, while high-frequency variations result from instantaneous IR drops along the power grid due to fluctuations in switching activity. Inductive issues on the power supply are also a major concern since they cause voltage fluctuations. This has particularly become a concern with clock gating as the load current can vary dramatically as the logic transitions back and forth between the idle and active states. Since the power supply can change rapidly, the period of the

clock signal is modulated on a cycle-by-cycle basis, resulting in *jitter.* The *jitter* on two different clock points maybe correlated or uncorrelated depending on how the power network is configured and the profile of switching patterns. Unfortunately, high-frequency power supply changes are difficult to compensate even with feedback techniques. As a result, power supply noise fundamentally limits the performance of clock networks.

Capacitive Coupling (✗ and ✗)

The variation in capacitive load also contributes to timing uncertainty. There are two major sources of capacitive load variations: coupling between the clock lines and adjacent signal wires and variation in gate capacitance. The clock network includes both the interconnect and the gate capacitance of latches and registers. Any coupling between the clock wire and adjacent signal results in timing uncertainty. Since the adjacent signal can transition in arbitrary directions and at arbitrary times, the exactly coupling to the clock network is not fixed from cycle-to-cycle. This results in clock *jitter.* Another major source of clock uncertainty is variation in the gate capacitance related to the sequential elements. The load capacitance is highly non-linear and depends on the applied voltage. In many latches and registers this translates to the clock load being a function of the current state of the latch/register (this is, the values stored on the internal nodes of the circuit), as well as the next state. This causes the delay through the clock buffers to vary from cycle-to-cycle, causing *jitter.*

Example 10.2 Data-dependent Clock Jitter

Consider the circuit shown in Figure 10.17, where a minimum-sized local clock buffer drives a register (actually, four registers are driven, though only one is shown here). The simulation shows *CKb*, the output of the first inverter for four possible transitions (0→0, 0→1, 1→0 and 1→1). The jitter on the clock based on data-dependent capacitance is illustrated. In general, the only way to deal with this problem is to use registers that don't exhibit a large variation in load as a function of data — for example, the differential sense-amplifier register shown in Chapter 7.

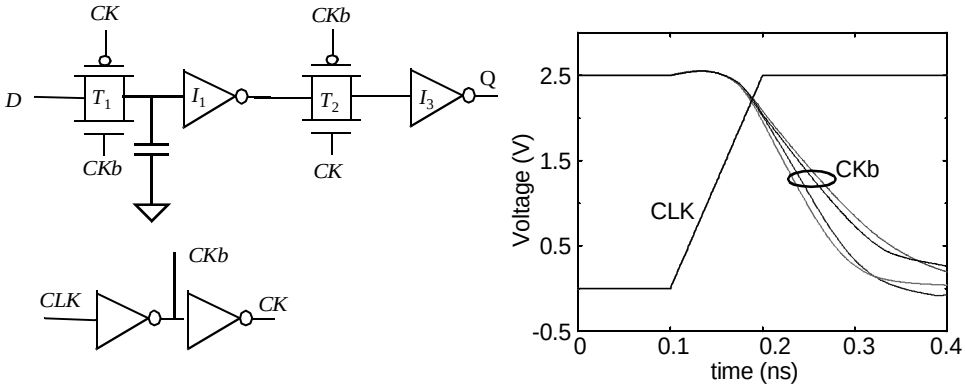


Figure 10.17 Impact of data-dependent clock load on clock jitter for pass-transistor register.

10.3.3 Clock-Distribution Techniques

It is clear from the previous discussion that clock *skew* and *jitter* are major issues in digital circuits, and can fundamentally limit the performance of a digital system. It is necessary to design a clock network that minimizes *skew* and *jitter*. Another important consideration in clock distribution is the power dissipation. In most high-speed digital processors, a majority of the power is dissipated in the clock network. To reduce power dissipation, clock networks must support clock conditioning — this is, the ability to shut down parts of the clock network. Unfortunately, clock gating results in additional clock uncertainty.

In this section, an overview of basic constructs in high-performance clock distribution techniques is presented along with a case study of clock distribution in the Alpha microprocessor. There are many degrees of freedom in the design of a clock network including the type of material used for wires, the basic topology and hierarchy, the sizing of wires and buffers, the rise and fall times, and the partitioning of load capacitances.

Fabrics for clocking

Clock networks typically include a network that is used to distribute a global reference to various parts of the chip, and a final stage that is responsible for local distribution of the clock while considering the local load variations. Most clock distribution schemes exploit the fact that the absolute delay from a central clock source to the clocking elements is irrelevant — only the relative phase between two clocking points is important. Therefore one common approach to distributing a clock is to use balanced paths (or called trees).

The most common type of clock primitive is the *H-tree network* (named for the physical structure of the network) as illustrated in Figure 10.18, where a 4x4 array is shown. In this scheme, the clock is routed to a central point on the chip and balanced paths, that include both matched interconnect as well as buffers, are used to distribute the reference to various leaf nodes. Ideally, if each path is balanced, the clock skew is zero. That is, though it might take multiple clock cycles for a signal to propagate from the central point to each leaf node, the arrival times are equal at every leaf node. However, in reality, as discussed in the previous section, process and environmental variations cause clock *skew* and *jitter* to occur.

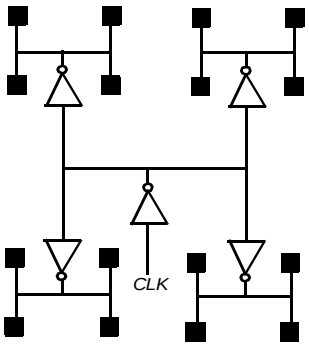


Figure 10.18 Example of an *H*-tree clock-distribution network for 16 leaf nodes.

The *H*-tree configuration is particularly useful for regular-array networks in which all elements are identical and the clock can be distributed as a binary tree (for example, arrays of identical tiled processors). The concept can be generalized to a more generic set-

Figure 10.19 An example RC-matched distribution for an IBM Microprocessor [Restle98].

ting. The more general approach, referred to as routed RC trees, represents a floorplan that distributes the clock signal so that the interconnections carrying the clock signals to the functional sub-blocks are of equal length. That is, the general approach does not rely on a regular physical structure. An example of a matched RC is shown in Figure 10.19. The chip is partitioned into ten balanced load segments (tiles). The global clock driver distributes the clock to tile drivers located at the dots in the figure. A lower level RC-matched tree is used to drive 580 additional drivers inside each tile.

Another important clock distribution template is the grid structure as shown in Figure 10.20 [Bailey00]. Grids are typically used in the final stage of clock network to distribute the clock to the clocking element loads. This approach is fundamentally different from the balanced RC approach. The main difference is that the delay from the final driver to each load is not matched. Rather, the absolute delay is minimized assuming that the grid size is small. A major advantage of such a grid structure is that it allows for late design changes since the clock is easily accessible at various points on the die. Unfortunately, the penalty is the power dissipation since the structure has a lot of unnecessary interconnect.

It is essential to consider clock distribution in the earlier phases of the design of a complex circuit, since it might influence the shape and form of the chip floorplan. Clock distribution is often only considered in the last phases of the design process, when most of the chip layout is already frozen. This results in unwieldy clock networks and multiple

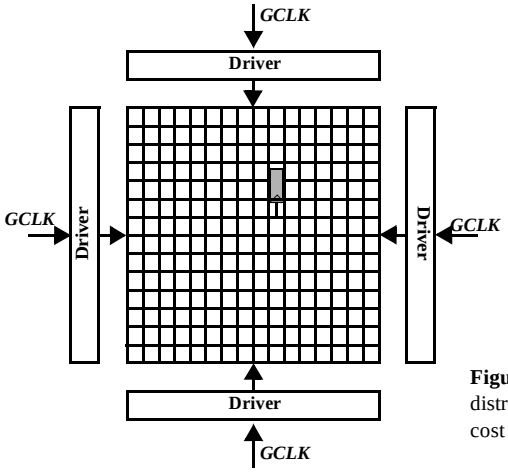


Figure 10.20 Grid structures allow a low skew distribution and physical design flexibility at the cost of power dissipation [Bailey00].

timing constraints that hamper the performance and operation of the final circuit. With careful planning, a designer can avoid many of these problems, and clock distribution becomes a manageable operation.

Case Study—The Digital Alpha Microprocessors

In this section, the clock distribution strategy for three generations of the Alpha microprocessor is discussed in detail. These processors have always been at the cutting edge of the technology, and therefore represent an interesting perspective on the evolution of clock distribution.

The Alpha 21064 Processor. The first generation Alpha microprocessor (21064 or EV4) from Digital Equipment Corporation used a single global clock driver [Dobberpuhl92]. The distribution of clock load capacitance among various functional block shown in Figure 10.20, resulting in a total clock load of 3.25nF! The processor uses a single-phase clock methodology and the 200Mhz clock is fed to a binary fanning tree with five levels of buffering. The inputs to the clock drivers are shorted out to smooth out the asymmetry in the incoming signals. The final output stage, residing in the middle of the chip, drives the clock net. The clock driver and the associated pre-drivers account for 40% of the effective switched capacitance (12.5nF), resulting in significant power dissipation. The overall width of the clock driver was on the order of 35cm in a 0.75µm technology. A detail clock skew simulation with process variations indicates that a clock uncertainty of less than 200psec (< 10%) was achieved.

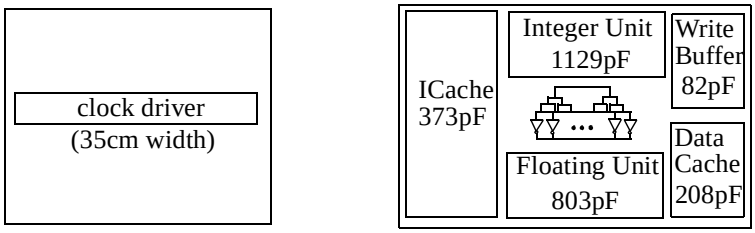
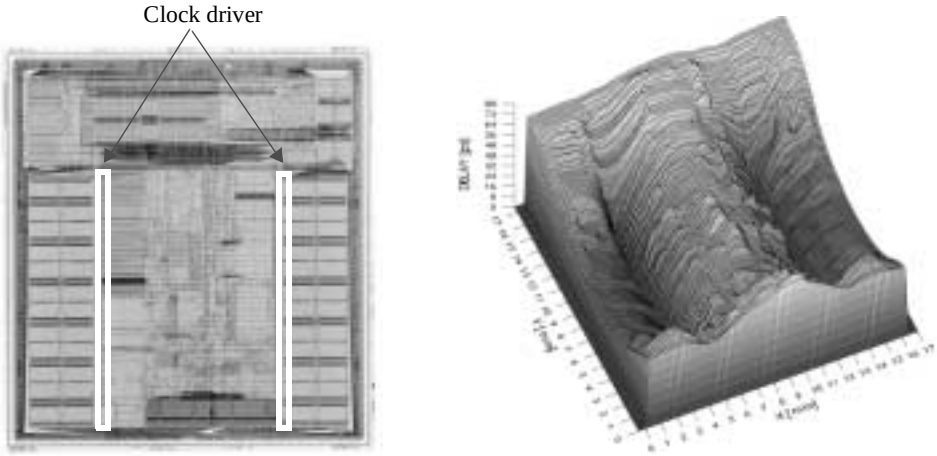


Figure 10.21 Clock load for the .

The Alpha 21164 Processor . The Alpha 21164 microprocessor (EV5) operates at a clock frequency of 300 Mhz while using 9.3 million transistors on a 16.5 mm × 18.1 mm die in a 0.5 µm CMOS technology [Bowhill95]. A single-phase clocking methodology was selected and the design made extensive use of dynamic logic, resulting in a substantial clock load of 3.75 nF. The clock-distribution system consumes 20 W, which is 40% of the total dissipation of the processor.

The incoming clock signal is first routed through a single six-stage buffer placed at the center of the chip. The resulting signal is distributed in metal-3 to the left and right banks of final clock drivers, positioned between the secondary cache memory and the outside edge of the execution unit (Figure 10.22a). The produced clock signal is driven onto a grid of metal-3 and metal-4 wires. The equivalent transistor width of the final driver inverter equals 58 cm! To ensure the integrity of the clock grid across the chip, the grid was extracted from the layout, and the resulting *RC*-network was simulated. A three-dimensional representation of the simulation results is plotted in Figure 10.22b. As evi-



(a) Chip microphotograph, showing positioning of clock drivers.

(b) Clock skew simulation.

Figure 10.22 Clock distribution and skew in 300 MHz microprocessor (*Courtesy of Digital Equipment Corporation*).

dent from the plot, the skew is zero at the output of the left and right drivers. The maximum value of the absolute skew is smaller than 90 psec. The critical instruction and execution units all see the clock within 65 psec.

Clock skew and race problems were addressed using a mix-and-match approach. The clock skew problems were eliminated by either routing the clock in the opposite direction of the data at a small expense of performance or by ensuring that the data could not overtake the clock. A standardized library of level-sensitive transmission-gate latches was used for the complete chip. To avoid race-through conditions, a number of design guidelines were followed including:

- Careful sizing of the local clock buffers so that their skew was minimal.
- At least one gate had to be inserted between connecting latches. This gate, which can be part of the logic function or just a simple inverter, ensures that the signal cannot overtake the clock. Special design verification tools were developed to guarantee that this rule was obeyed over the complete chip.

To improve the inter-layer dielectric uniformity, filler polygons were inserted between widely spaced lines (Figure 10.23). Though this may increase the capacitance to nearby signal lines, the improved uniformity results in lower variation and clock uncertainty. The dummy fills are automatically inserted, and tied to one of the power rails (V_{DD} or *GND*). This technique is used in many processors for controlling the clock skew.

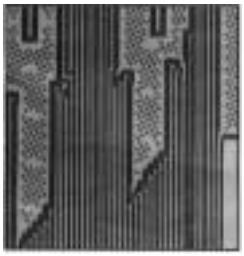


Figure 10.23 Dummy fills reduce the ILD variation and improve clock skew.

This example demonstrates that managing clock skew and clock distribution for large, high-performance synchro-

nous designs is a feasible task. However, making such a circuit work in a reliable way requires careful planning and intensive analysis.

The Alpha 21264 Processor. A hierarchical clocking scheme is used in the 600Mhz Alpha 21264 (EV6) processor (in 0.35µm CMOS) as shown in Figure 10.24. The choice of a hierarchical clocking scheme for this processor is significantly different than their previous approaches, which did not have a hierarchy of clocks beyond the global clock grid. Using a hierarchical clocking approach makes trade-off's between power and skew management. Power is reduced because the clocking networks for individual blocks can be gated. As seen in previous generation microprocessors, the clock power contributes to a large fraction of overall power consumption. Also the flexibility of having local clocks provides the designers with more freedom with circuit styles at the module level. The drawback of using a hierarchical clock network is that skew reduction becomes more difficult because clocks to various local registers may go through very different paths, which may contribute to the skew. However, using timing verification tools, the skew can be managed by tweaking of clock drivers.

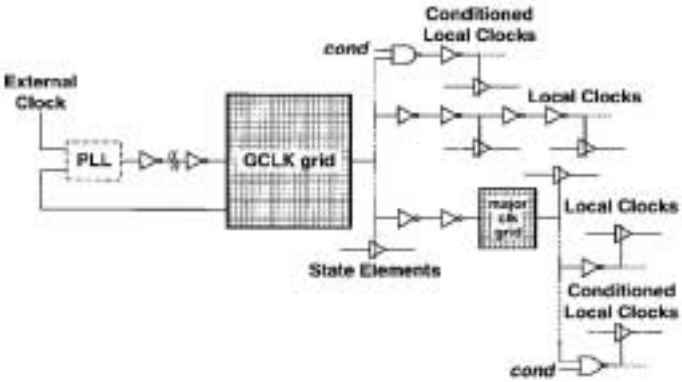


Figure 10.24 Clock hierarchy for the Alpha 21264 Processor.

The clock hierarchy consists of a global clock grid, called *GCLK*, that covers the entire die. State elements and clocking points exist from 0 to 8 levels past *GCLK*. The on-chip generated clock is routed to the center of the die and distributed using tree structures to 16 distributed clock drivers (Figure 10.25). The global clock distribution network utilizes a windowpane configuration, that achieves low skew by dividing up the clock into 4 regions -- this reduces the distance from the drivers to the loads. Each grid pane is driven from four sides, reducing the dependence on process variations. This also helps the power supply and thermal problems as the drivers are distributed through the chip.

Use of a grid-ded clock has the advantage of reducing the clock skew and provides universal availability of clock signals. The drawback clearly is the increased capacitance of the Global Clock grid when compared to a tree distribution approach. Beyond the *GCLK* is a major clock grid. The major clock grid are used to drive different large execution blocks within the chip including 1) Bus interface unit 2) integer issue and execution units 3) floating point issue and execution units 4) instruction fetch and branch prediction unit 5) load/store unit and 6) pad ring. The reason the major clocks were introduced was to reduce power because the major clocks have localized loads, and can be sized appropriately to meet the skew and edge requirements for the local loading conditions.

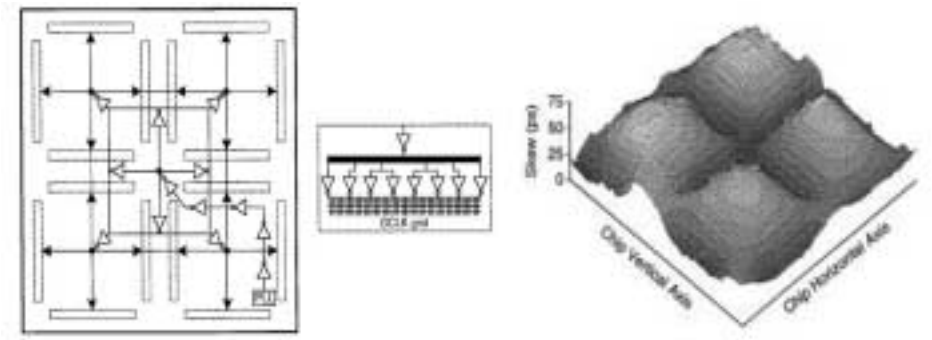


Figure 10.25 Global clock-distribution network in a window-pane structure.

The last hierarchy of clocks in the Alpha processor are the local clocks, which are generated as needed from any other clock, and typically can be customized to meet local timing constraints. The local clocks provide great flexibility in the design of the local logic blocks, but at the same time, makes it significantly more difficult to manage skew. For example, local clocks can be gated in order to reduce power, can be driven with wildly different loads, and can even be tweaked to enable time borrowing between different blocks. Furthermore, the local clocks are susceptible to coupling from data lines as well because they are local and not shielded like the global grid-ded clocks. As a result, the local clock distribution is highly dependent on it's local interconnection, and has to be designed very carefully to manage local clock skew.

Design Techniques—Dealing with Clock Skew and Jitter

To fully exploit the improved performance of logic gates with technology scaling, clock *skew* and *jitter* must be carefully addressed. *Skew* and *jitter* can fundamentally limit the peformance of a digital circuits. Some guidelines for reducing of clock *skew* and *jitter* are presented below.

1. To minimize *skew*, balance clock paths from a central distribution source to individual clocking elements using H-tree structures or more generally routed tree structures. When using routed clock trees, the effective clock load of each path that includes wiring as well as transistor loads must be equalized.

2. The use of local clock grids (instead of routed trees) can reduce *skew* at the cost of increased capacitive load and power dissipation.

3. If data dependent clock load variations causes significant *jitter*, differential registers that have a data independent clock load should be used. The use of gated clocks to save also results in data dependent clock load and increased *jitter*. In clock networks where the fixed load is large (e.g., using clock grids), the data dependent variation might not be significant.

4. If data flows in one direction, route data and clock in opposite directions. This elminates races at the cost of perfomance.

5. Avoid data dependent noise by shielding clock wires from adjacent signal wires. By placing power lines (V_{DD} or *GND*) next to the clock wires, coupling from neighboring signal nets can be minimized or avoided.

6. Variations in interconnect capacitance due to inter-layer dielectric thickness variation can be greatly reduced through the use of dummy fills. Dummy fills are very common and reduce *skew* by increasing uniformity. Systematic variations should be modeled and compensated for.

7. Variation in chip temperature across the die causes variations in clock buffer delay. The use of feedback circuits based on delay locked loops as discussed later in this chapter can easily compensate for temperature variations.

8. Power supply variation is a significant component of *jitter* as it impacts the cycle to cycle delay through clock buffers. High frequency power supply variation can be reduced by addition of on-chip decoupling capacitors. Unfortunately, decoupling capacitors require a significant amount of area and efficient packaging solutions must be leveraged to reduce chip area.

10.3.4 Latch-Based Clocking

While the use of registers in a sequential circuits enables a robust design methodology, there are significant performance advantages to using a latch based design in which combinational logic is separated by transparent latches. In an edge-triggered system, the worst case logic path between two registers determines the minimum clock period for the entire system. If a logic block finishes before the clock period, it has to idle till the next input is latched in on the next system clock edge. The use of a latch based methodology (as illustrated in Figure 10.26) enables more flexible timing, allowing one stage to pass slack to or steal time from following stages. This flexibility, allows an overall performance increase. Note that the latch based methodology is nothing more than adding logic between latches of a master-slave flip-flop.

For the latch-based system in Figure 10.26, assume a two-phase clocking scheme. Assume furthermore that the clock are ideal, and that the two clocks are inverted versions of each other (for sake of simplicity). In this configuration, a stable input is available to the combinational logic block A (*CLB_A*) on the falling edge of CLK1 (at edge ②) and it has a maximum time equal to the $T_{CLK}/2$ to evaluate (that is, the entire low phase of CLK1). On the falling edge of CLK2 (at edge ③), the output *CLB_A* is latched and the computation of *CLK_B* is launched. *CLB_B* computes on the low phase of CLK2 and the output is available on the falling edge of CLK1 (at edge ④). This timing appears equivalent to having an edge-triggered system where *CLB_A* and *CLB_B* are cascaded and between two edge-triggered registers (Figure 10.27). In both cases, it appears that the time available to perform the combination of *CLB_A* and *CLB_B* is T_{CLK}.

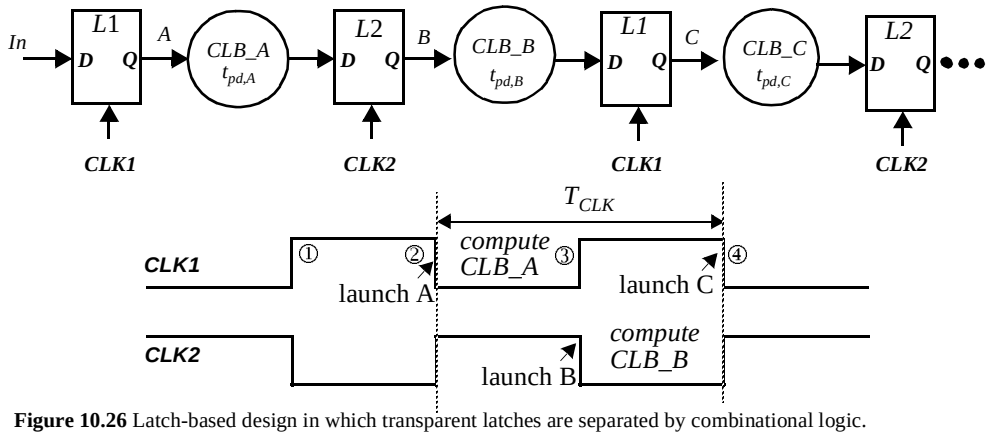


Figure 10.26 Latch-based design in which transparent latches are separated by combinational logic.

However, there is an important performance related difference. In a latch based system, since the logic is separated by level sensitive latches, it possible for a logic block to utilize time that is left over from the previous logic block and this is referred to as *slack borrowing* [Bernstein00]. This approach requires no explicit design changes, as the passing of slack from one block to the next is automatic. The key advantage of slack borrowing is that it allows logic between cycle boundaries to use more than one clock cycle while satisfying the cycle time constraint. Stated in another way, if the sequential system works at a particular clock rate and the total logic delay for a complete cycle is larger than the clock period, then unused time or *slack* has been implicitly borrowed from preceding stages. This implies that the clock rate can be higher than the worst case critical path!

Slack passing happens due to the level sensitive nature of latches. In Figure 10.26, the input to *CLB_A* should be valid by the falling edge of CLK1 (edge ②). What happens if the combinational logic block of the previous stage finishes early and has a valid input data for *CLB_A* before edge ②? Since the a latch is transparent during the entire high phase of the clock, as soon as the previous stage has finished computing, the input data for *CLB_A* is valid. This implies that the maximum time available for *CLB_A* is its phase time (i.e., the low phase of CLK1) and any left over time from the previous computation. Formally state, slack passing has taken place if $T_{CLK} < t_{pd,A} + t_{pd,B}$ and the logic functions correctly (for simplicity, the delay associated with latches are ignored). An excellent quantitative analysis of slack-borrowing is presented in [Bernstein00].

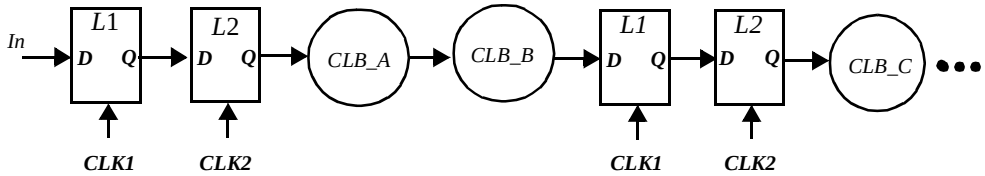


Figure 10.27 Edge-triggered pipeline (back-to-back latches for edge-triggered registers) of the logic in Figure 10.26.

Consider the latch based system in Figure 10.28. In this example, point *a* (input to *CLB_A*) is valid before the edge ②. This implies that the previous block did not use up the entire high phase of CLK1, which results in slack time (denoted by the shaded area). By

construction, *CLB_A* can start computing as soon as point *a* becomes valid and uses the slack time to finish well before its allocated time (edge ③). Since L2 is a transparent latch, *c* becomes valid on the high phase of CLK2 and *CLB_B* starts to compute by using the slack provided by *CLB_A*. *CLB_B* completes before its allocated time (edge ④) and passes a small slack to the next cycle. As this picture indicates, the total cycle delay, that is the sum of the delay for *CLB_A* and *CLB_B*, is larger than the clock period. Since the pipeline behaves correctly, slack passing has taken place and a higher throughput has been achieved.

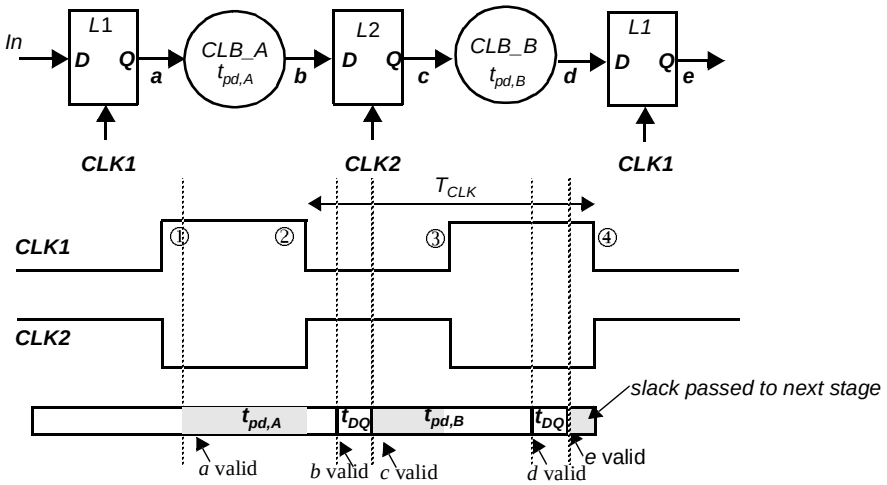


Figure 10.28 Example of slack-borrowing.

An important question related to slack passing relates to the maximum possible slack that can be passed across cycle boundaries. In Figure 10.28, it is easy that see that the earliest time that *CLB_A* can start computing is ①. This happens if the previous logic block did not use any of its allocated time (CLK1 high phase) or if it finished by using slack from previous stages. Therefore, the maximum time that can be borrowed from the previous stage is 1/2 cycle or $T_{CLK}/2$. Similarly, *CLB_B* must finish its operation by edge ④. This implies that the maximum logic cycle delay is equal to $1.5 * T_{CLK}$. However, note that for an *n*-stage pipeline, the overall logic delay cannot exceed the time available of $n * T_{CLK}$.

Example 10.3 Slack-passing example

First consider an edge-triggered pipeline of Figure 10.29. Assume that the primary input *In* is valid slightly after the rising edge of the clock. For this circuit, it is easy to see that the minimum clock period required is 125ns. The latency is two clock cycles (actually, the output is valid 2.5 cycles after the input settles). Note that for the first pipeline stage, 1/2

cycle is wasted as the input data can only be latched in on the falling edge of the clock. This time can be exploited in a latch based system.

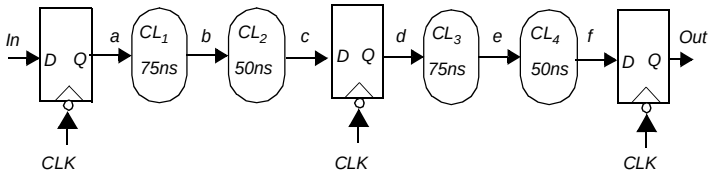


Figure 10.29 Conventional edge-triggered pipeline.

Figure 10.30 shows a latch based pipeline of the same sequential circuit. As the timing indicates, exactly the same timing can be achieve with a clock period of 100ns. This is enabled by slack borrowing between logical partitions.

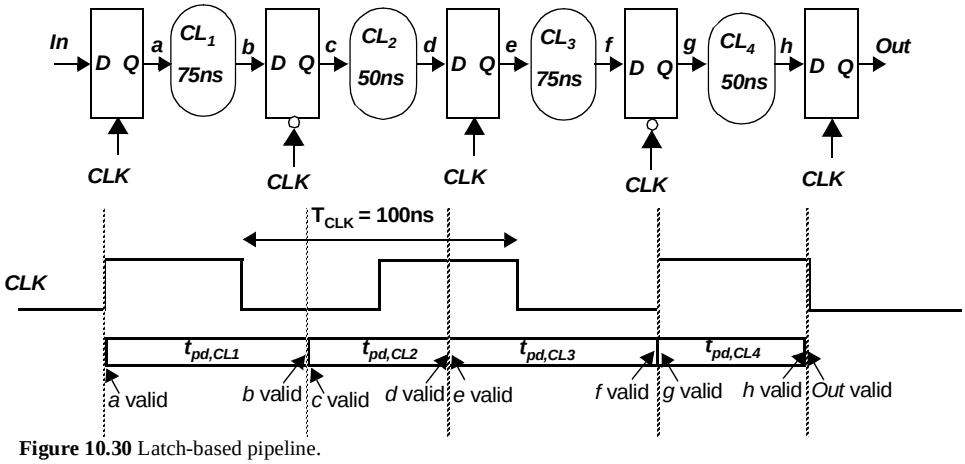


Figure 10.30 Latch-based pipeline.

10.4 Self-Timed Circuit Design*

10.4.1 Self-Timed Logic - An Asynchronous Technique

The synchronous design approach advocated in the previous sections assumes that all circuit events are orchestrated by a central clock. Those clocks have a dual function.

- They insure that the *physical timing constraints* are met. The next clock cycle can only start when all logic transitions have settled and the system has come to a steady state. This ensures that only legal logical values are applied in the next round of computation. In short, clocks account for the worst case delays of logic gates, sequential logic elements and the wiring.
- Clock events serve as a *logical ordering mechanism* for the global system events. A clock provides a time base that determines what will happen and when. On every

clock transition, a number of operations are initiated that change the state of the sequential network.

Consider the pipelined datapath of Figure 10.31. In this circuit, the data transitions through logic stages under the command of the clock. The important point to note under this methodology is that the clock period is chosen to be larger than the worst-case delay of each pipeline stage, or $T > \max(t_{pd1}, t_{pd2}, t_{pd3}) + t_{pd,reg}$. This will ensure satisfying the physical constraint. At each clock transition, a new set of inputs is sampled and computation is started anew. The throughput of the system—which is equivalent to the number of data samples processed per second—is equivalent to the clock rate. When to sample a new input or when an output is available depends upon the *logical ordering* of the system events and is clearly orchestrated by the clock in this example.

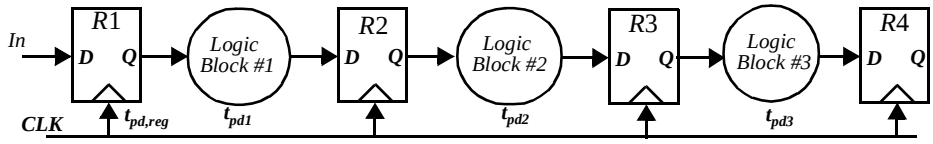


Figure 10.31 Pipelined, synchronous datapath.

The synchronous design methodology has some clear advantages. It presents a structured, deterministic approach to the problem of choreographing the myriad of events that take place in digital designs. The approach taken is to equalize the delays of all operations by making them as bad as the worst of the set. The approach is robust and easy to adhere to, which explains its enormous popularity; however it does have some pitfalls.

- It assumes that all clock events or timing references happen simultaneously over the complete circuit. This is not the case in reality, because of effects such as clock *skew* and *jitter*.
- As all the clocks in a circuit transitions at the same time, significant current flows over a very short period of time (due to the large capacitance load). This causes significant noise problems due to package inductance and power supply grid resistance.
- The linking of physical and logical constraints has some obvious effects on the performance. For instance, the throughput rate of the pipelined system of Figure 10.31 is directly linked to the worst-case delay of the slowest element in the pipeline. On the average, the delay of each pipeline stage is smaller. The same pipeline could support an average throughput rate that is substantially higher than the synchronous one. For example, the propagation delay of a 16-bit adder is highly data dependent (e.g., adding two 4-bit numbers requires a much shorter time compared to adding two 16-bit numbers).

One way to avoid these problems is to opt for an *asynchronous* design approach and to eliminate all the clocks. Designing a purely asynchronous circuit is a nontrivial and potentially hazardous task. Ensuring a correct circuit operation that avoids all potential race conditions under any operation condition and input sequence requires a careful timing analysis of the network. In fact, the logical ordering of the events is dictated by the structure of the transistor network and the relative delays of the signals. Enforcing timing

constraints by manipulating the logic structure and the lengths of the signal paths requires an extensive use of CAD tools, and is only recommended when strictly necessary.

A more reliable and robust technique is the self-timed approach, which presents a local solution to the timing problem [Seitz80]. Figure 10.32 uses a pipelined datapath to illustrate how this can be accomplished. This approach assumes that each combinational function has a means of indicating that it has completed a computation for a particular piece of data. The computation of a logic block is initiated by asserting a *Start* signal. The combinational logic block computes on the input data and in a data-dependent fashion (taking the physical constraints into account) generates a *Done* flag once the computation is finished. Additionally, the operators must signal each other that they are either ready to receive a next input word or that they have a legal data word at their outputs that is ready for consumption. This signaling ensures the logical ordering of the events and can be achieved with the aid of an extra *Ack*(*nowledge*) and *Req*(*uest*) signal. In the case of the pipelined datapath, the scenario could proceed as follows.

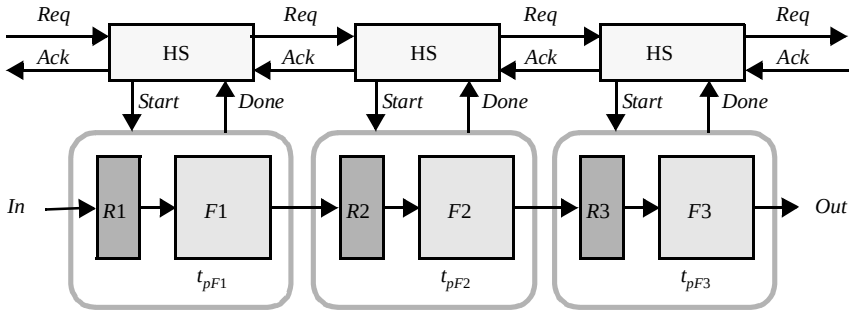


Figure 10.32 Self-timed, pipelined datapath.

1. An input word arrives, and a *Req(uest)* to the block *F*1 is raised. If *F*1 is inactive at that time, it transfers the data and acknowledges this fact to the input buffer, which can go ahead and fetch the next word.

2. *F*1 is enabled by raising the *Start* signal. After a certain amount of time, dependent upon the data values, the *Done* signal goes high indicating the completion of the computation.

3. A *Re(quest)* is issued to the *F*2 module. If this function is free, an *Ack(nowledge)* is raised, the output value is transferred, and *F*1 can go ahead with its next computation.

The self-timed approach effectively separates the physical and logical ordering functions implied in circuit timing. The completion signal *Done* ensures that the physical timing constraints are met and that the circuit is in steady state before accepting a new input. The logical ordering of the operations is ensured by the acknowledge-request scheme, often called *a handshaking protocol*. Both interested parties synchronize with each other by mutual agreement or, if you want, by shaking hands. The ordering protocol described above and implemented in the module HS is only one of many that are possible. The choice of protocol is important, since it has a profound effect on the circuit performance and robustness.

When compared to the synchronous approach, self-timed circuits display some alluring properties.

- In contrast to the global centralized approach of the synchronous methodology, timing signals are generated *locally*. This avoids all problems and overheads associated with distributing high-speed clocks.
- Separating the physical and logical ordering mechanisms results in a potential increase in performance. In synchronous systems, the period of the clock has to be stretched to accommodate the slowest path over all possible input sequences. In self-timed systems, a completed data-word does not have to wait for the arrival of the next clock edge to proceed to the subsequent processing stages. Since circuit delays are often dependent upon the actual data value, a self-timed circuit proceeds at the *average speed* of the hardware in contrast to the *worst-case* model of synchronous logic. For a ripple carry adder, the average length of carry-propagation is O(log (N)). This is a fact that can be exploited in self-timed circuits, while in a synchronous methodology, a worst case performance that varies linearly with the number of bits O(N)) must be assumed.
- The automatic shut-down of blocks that are not in use can result in power savings. Additionally, the power consumption overhead of generating and distributing high-speed clocks can be partially avoided. As discussed earlier, this overhead can be substantial. It is also possible to reduce the average clock load in synchronous processors through the use of gated clocks in which portions of the circuits that don't compute in a cycle are shut off.
- Self-timed circuits are by nature robust to variations in manufacturing and operating conditions such as temperature. Synchronous systems are limited by their performance at the extremes of the operating conditions. The performance of a self-timed system is determined by the actual operating conditions.

Unfortunately, these nice properties are not for free; they come at the expense of a substantial circuit-level overhead, which is caused by the need to generate completion signals and the need for handshaking logic that acts as a local traffic agent to order the circuit events (see block HS in Figure 10.32). Both of these topics are treated in more detail in subsequent sections.

10.4.2 Completion-Signal Generation

A necessary component of self-timed logic is the circuitry to indicate when a particular piece of circuitry has completed its operation for the current piece of data. There are two common and reliable ways to generate the completion signal.

Dual-Rail Coding

One common approach to completion signal generation is the use of dual rail coding. It actually requires the introduction of redundancy in the data representation to signal that a particular bit is either in a transition or steady-state mode. Consider the redundant data

model presented in Table 10.1. Two bits ($B0$ and $B1$) are used to represent a single data bit B. For the data to be valid or the computation to be completed, the circuit must be in a legal 0 ($B0 = 0$, $B1 = 1$) or 1 ($B0 = 1$, $B1 = 0$) state. The ($B0 = 0$, $B1 = 0$) condition signals that the data is non-valid and the circuit is in either a reset or transition mode. The ($B0 = 1$, $B1 = 1$) state is illegal and should never occur in an actual circuit.

Table 10.1 Redundant signal representation to include transition state.

B	***B*0**	***B*1**
in transition (or reset)	0	0
0	0	1
1	1	0
illegal	1	1

A circuit that actually implements such a redundant representation is shown in Figure 10.33, which is a dynamic version of the DCVSL logic style where the clock is replaced by the *Start* signal [Heller84]. DCVSL uses a redundant data representation by nature of its differential dual-rail structure. When the *Start* signal is low, the circuit is precharged by the PMOS transistors, and the output ($B0$, $B1$) goes in the *Reset-Transition* state (0, 0). When the *Start* signal goes high, signaling the initiation of a computation, the NMOS pull-down network evaluates, and one of the precharged nodes is lowered. Either $B0$ or $B1$—but never both—goes high, which raises *Done* and signals the completion of the computation.

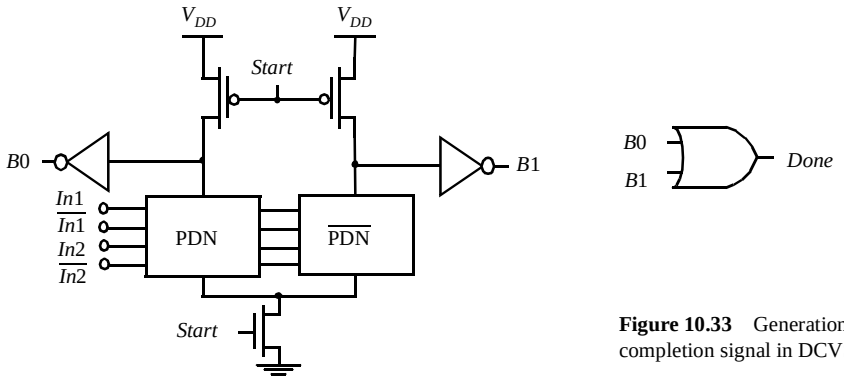
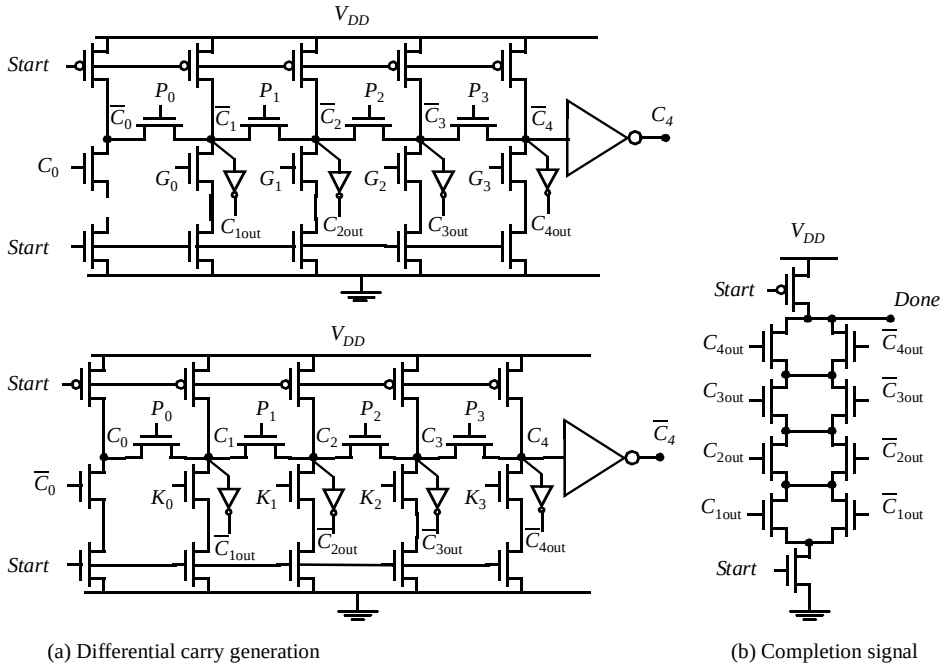


Figure 10.33 Generation of a completion signal in DCVSL.

DCVSL is more expensive in terms of area than a non-redundant circuit due to its dual nature. The completion generation is performed in series with the logic evaluation, and its delay adds directly to the total delay of the logic block. The completion signals of all the individual bits must be combined to generate the completion for an N-bit data word. Completion generation thus comes at the expense of both area and speed. The benefits of the dynamic timing generation often justify this overhead. Redundant signal presentations other than the one presented in Table 10.1 can also be envisioned. One essential element is the presence of a *transition state* denoting that the circuit is in evaluation mode and the output data is not valid.

Example 10.4 Self-Timed Adder Circuit

An efficient implementation of a self-timed adder circuit is shown in Figure 10.34 [Abnous93]. A Manchester-carry scheme is used to boost the circuit performance. The proposed approach is based on the observation that the circuit delay of an adder is dominated by the carry-propagation path. It is consequentially sufficient to use the differential signaling in the carry path only (Figure 10.34a). The completion signal is efficiently derived by combining the carry signals of the different stages (Figure 10.34b). This safely assumes that the sum generation, which depends upon the arrival of the carry signal, is faster than the completion generation. The benefit of this approach is that the completion generation starts earlier and proceeds in parallel with sum generation, which reduces the critical timing path. All other signals such as *P(ropagate)*, *G(enerate)*, *K(ill)*, and *S(um)* do not require completion generation and can be implemented in single-ended logic. As shown in the circuit schematics, the differential carry paths are virtually identical. The only difference is that the *G(enerate)* signal is replaced by a *K(ill)*. A simple logic analysis demonstrates that this indeed results in an inverted carry signal and, hence, a differential signaling.



(a) Differential carry generation (b) Completion signal

Figure 10.34 Manchester-carry scheme with differential signal representation.

Replica Delay

While the dual-rail coding above allows tracking of the signal statistics, it comes at the cost of power dissipation. Every single gate must transition for every new input vector, regardless of the value of the data vector. An attempt to reduce the overhead of completion detection is to use a critical-path replica configured as a delay element, as shown in Figure

10.35. To start a computation, the *Start* signal is raised and the computation of the logic network is initiated. At the same time, the start signal is fed into the replica delay line, which tracks the critical path of the logic network. It is important that the replica is structured such that no glitching transitions occur. When the output of the delay line makes a transition, it indicates that the logic is complete—as the delay line mimics the critical path. In general, it is important to add extra padding in the delay line to compensate for possible random process variations. The advantage of this approach is that the logic can be implemented using a standard non-redundant circuit style such as complementary CMOS. Also, if multiple logic units are computing in parallel, it is possible to amortize the overhead of the delay line over multiple blocks. Note that this approach generates the completion signal after a time equal to the worst case delay through the network. As a result, it does not exploit the statistical properties of the incoming data. However, it can track the local effects of process variations and environmental variations (e.g., temperature or power supply variations). This approach is widely used to generate the internal timing of semiconductor memories where self-timing is a commonly used technique.

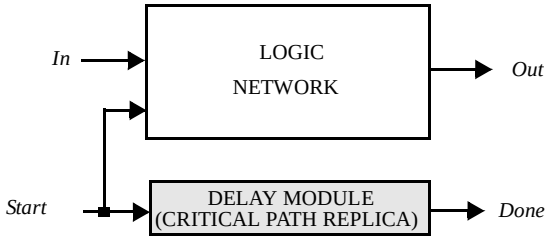


Figure 10.35 Completion-signal gener using delay module.

Example 10.5An alternate completion detection circuit using current sensing

Ideally, logic should be implemented using non-redundant CMOS (e.g., static CMOS) and the completion signal circuitry should track data dependencies. Figure 10.36 shown an example technique that attempts to realize the above [REF]. A current sensor is inserted in series with the combinational logic, and monitors the current flowing through the logic. The current sensor outputs a low value when no current flows through the logic (this is, the logic is idle), and is high when the combinational logic is switching performing computation. This signal effectively determines when the logic has completed its cycle. Note that this approach tracks data-dependent computation times — if only the lower order bits of an adder switch, current will stop flowing once the lower order bits switch to the final value.

If the input data vector does not change from one cycle to the next, no current is drawn from the supply for static CMOS logic. In this case, a delay element that tracks the minimum delay through the logic and current sensor is used for signal completion. The outputs of the current sensor and minimum delay element are then combined. This approach is interesting, but requires careful analog design. Ensuring reliability while keeping the overhead circuitry small is the main challenge. These concerns have kept the applicability of the approach very limited, notwithstanding its great potential.

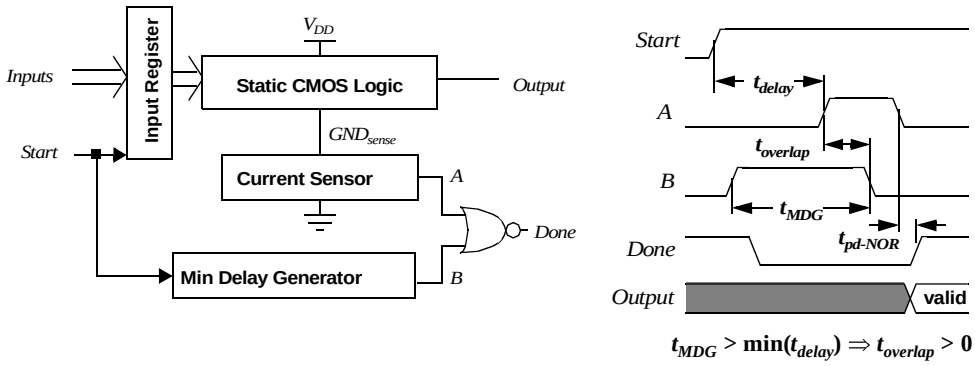
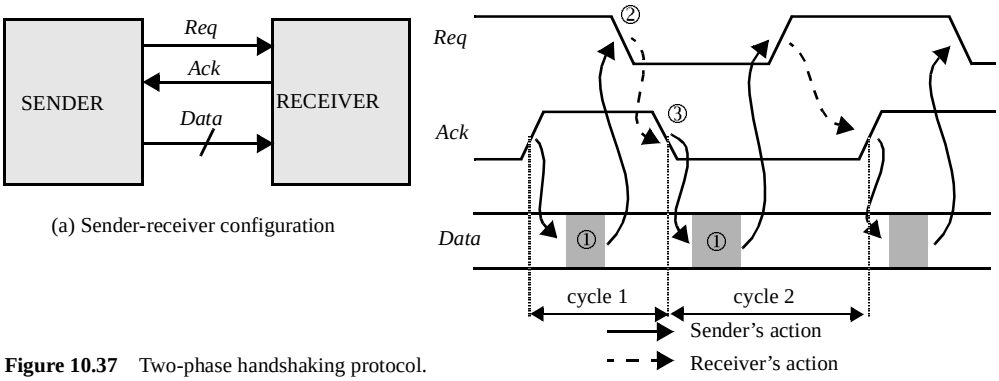


Figure 10.36 Completion-signal generation using current sensing.

10.4.3 Self-Timed Signaling

Besides the generation of the completion signals, a self-timed approach also requires a handshaking protocol to logically order the circuit events avoiding races and hazards. The functionality of the signaling (or handshaking) logic is illustrated by the example of Figure 10.37, which shows a *sender module* transmitting data to a *receiver* ([Sutherland89]).

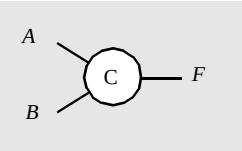


(a) Sender-receiver configuration

(b) Timing diagram

Figure 10.37 Two-phase handshaking protocol.

The sender places the data value on the data bus ① and produces an event on the *Req* control signal by changing the polarity of the signal ②. In some cases the request event is a rising transition; at other times it is a falling one—the protocol described here does not distinguish between them. Upon receiving the request, the receiver accepts the data when possible and produces an event on the *Ack* signal to indicate that the data has been accepted ③. If the receiver is busy or its input buffer is full, no *Ack* event is generated, and the transmitter is stalled until the receiver becomes available by, for instance, freeing space in the input buffer. Once the *Ack* event is produced, the transmitter goes ahead and produces the next data word ①. The four events, *data change, request, data acceptance,*



(a) Schematic

A	B	F_{n+1}
0	0	0
0	1	F_n
1	0	F_n
1	1	1

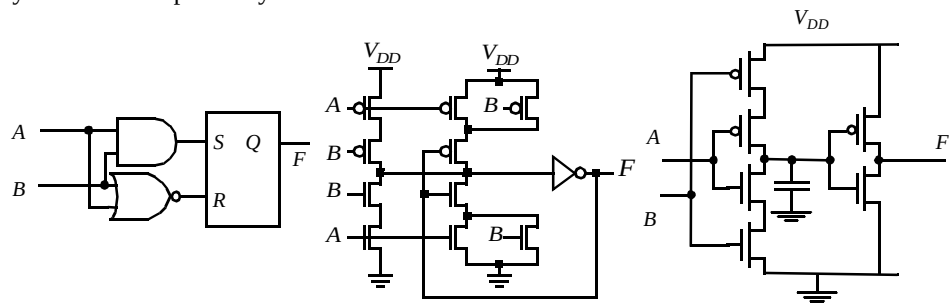
(b) Truth table

Figure 10.38 Muller C-element.

and *acknowledge*, proceed in a cyclic order. Successive cycles may take different amounts of time depending upon the time it takes to produce or consume the data.

This protocol is called *two-phase*, since only two phases of operation can be distinguished for each data transmission—the active cycle of the sender and the active cycle of the receiver. Both phases are terminated by certain events. The *Req* event terminates the active cycle of the sender, while the receiver's cycle is completed by the *Ack* event. The sender is free to change the data during its active cycle. Once the *Req* event is generated, it has to keep the data constant as long as the receiver is active. The receiver can only accept data during its active cycle.

The correct operation of the sender-receiver system requires a strict ordering of the signaling events, as indicated by the arrows in Figure 10.37. Imposing this order is the task of the handshaking logic which, in a sense, performs logic manipulations on events. An essential component of virtually any handshaking module is the *Muller C-element*. This gate, whose schematic symbol and truth table are given in Figure 10.38, performs an AND-operation on events. The output of the C-element is a copy of its inputs when both inputs are identical. When the inputs differ, the output retains its previous value. Phrased in a different way—events must occur at both inputs of a Muller C-element for its output to change state and to create an output event. As long as this does not happen, the output remains unchanged and no output event is generated. The implementation of a C-element is centered around a flip-flop, which should be of no surprise, given the similarities in their truth tables. Figure 10.39 displays two potential circuit realizations—a static and a dynamic one respectively.



(a) Logic (b) Majority Function (c) Dynamic

Figure 10.39 Implementations of a Muller C-element.

Figure 10.40 shows how to use this component to enforce the two-phase handshaking protocol for the example of the sender-receiver. Assume that *Req*, *Ack*, and *Data Ready* are initially 0. When the sender wants to transmit the next word, the *Data Ready* signal is set to 1**,** which triggers the C-element because both its inputs are at 1. *Req* goes

high—this is commonly denoted as *Req*↑. The sender now resides in the wait mode, and control is passed to the receiver. The C-element is blocked, and no new data is sent to the data bus (*Req* stays high) as long as the transmitted data is not processed by the receiver. Once this happens, the *Data accepted* signal is raised. This can be the result of many different actions, possibly involving other C-elements communicating with subsequent blocks. An *Ack*↑ ensues, which unblocks the C-element and passes the control back to the sender. A *Data ready*↓ event, which might already have happened before *Ack*↑, produces a *Req*↓, and the cycle is repeated.

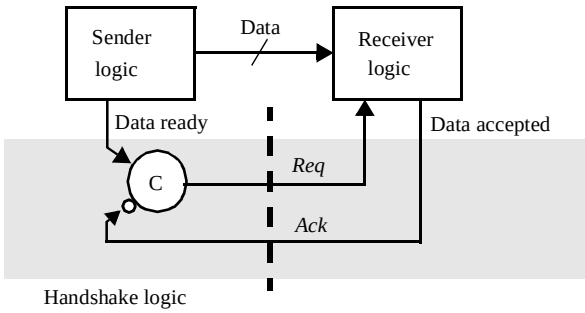


Figure 10.40 A Muller C-element implements a two-phase handshake protocol. The circle at the lower input of the Muller C-element stands for inversion.

Problem 10.1 Two-Phase Self-Timed FIFO

Figure 10.41 shows a two-phase, self-timed implementation of a FIFO (first-in first-out) buffer with three registers. Assuming that the registers accept a data word on both positive- and negative-going transitions of the *En* signals and that the *Done* signal is simply a delayed version of *En*, examine the operation of the FIFO by plotting the timing behavior of all signals of interest. How can you observe that the FIFO is completely empty (full)? (Hint: Determine the necessary conditions on the *Ack* and *Req* signals.)

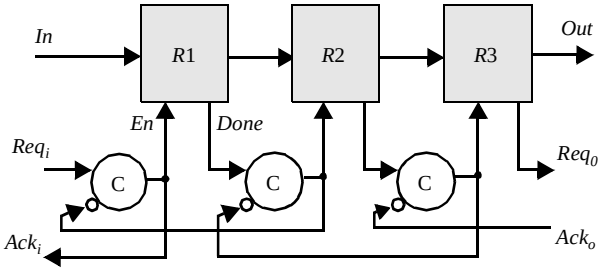


Figure 10.41 Three-stage self-timed FIFO, using a two-phase signaling protocol.

The two-phase protocol has the advantage of being simple and fast. There is some bad news, however. This protocol requires the detection of transitions that may occur in either direction. Most logic devices in the MOS technology tend to be sensitive to levels or to transitions in one particular direction. Event-triggered logic, as required in the two-phase protocol, requires extra logic as well as state information in the registers and the computational elements. Since the transition direction is important, initializing all the Muller C-elements in the appropriate state is essential. If this is not done, the circuit might dead-lock, which means that all elements are permanently blocked and nothing will ever

happen. A detailed study on how to implement event-triggered logic can be found in the text by Sutherland on micropipelines [Sutherland89].

The only alternative is to adopt a different signaling approach, called four-phase signaling, or *return-to-zero* (*RTZ*). This class of signaling requires that all controlling signals be brought back to their initial values before the next cycle can be initiated. Once again, this is illustrated with the example of the sender-receiver. The four-phase protocol for this example is shown in Figure 10.42.

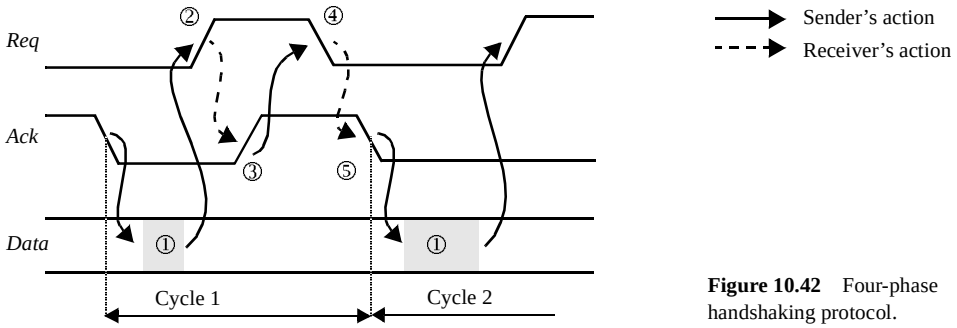


Figure 10.42 Four-phase handshaking protocol.

The protocol presented is initially the same as the two-phase one. Both the *Req* and the *Ack* are initially in the zero-state, however. Once a new data word is put on the bus ①, the *Req* is raised (*Req*↑ or ②) and control is passed to the receiver. When ready, the receiver accepts the data and raises *Ack* (*Ack*↑ or ③). So far, nothing new. The protocol proceeds, now by bringing both *Req* (*Req*↓ or ④) and *Ack* (*Ack*↓ or ⑤) back to their initial state in sequence. Only when that state is reached is the sender allowed to put new data on the bus ①. This protocol is called four-phase because four distinct time-zones can be recognized per cycle: two for the sender; two for the receiver. The first two phases are identical to the two-phase protocol; the last two are devoted to resetting of the state. An implementation of the protocol, based on Muller C-elements, is shown in Figure 10.43. It is interesting to notice that the four-phase protocol requires two C-elements in series (since four states must be represented). The *Data ready* and *Data accepted* signals must be pulses instead of single transitions.

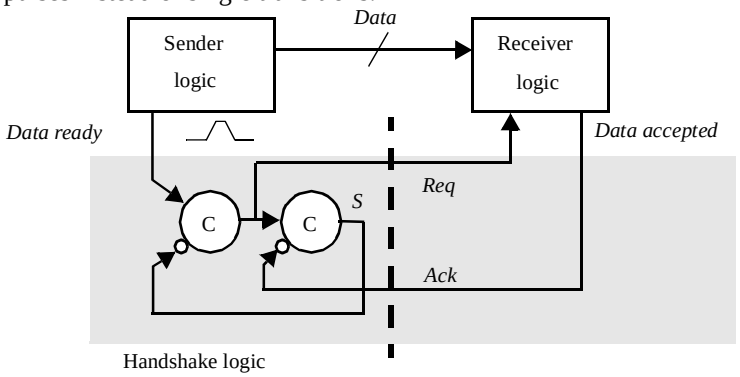


Figure 10.43 Implementation of 4-phase handshake protocol using Muller C-elements.

Problem 10.2 Four-Phase Protocol

Derive the timing diagram for the signals shown in Figure 10.43. Assume that the *Data Ready* signal is a pulse and that the *Data Accepted* signal is a delayed version of *Req*.

The four-phase protocol has the disadvantage of being more complex and slower, since two events on *Req* and *Ack* are needed per transmission. On the other hand, it has the advantage of being robust. The logic in the sender and receiver modules does not have to deal with transitions, which can go either way, but only has to consider rising (or falling) transition events or signal levels. This is readily accomplished with traditional logic circuits. For this reason, four-phase handshakes are the preferred implementation approach for most of the current self-timed circuits. The two-phase protocol is mostly selected when the sender and receiver are far apart and the delays on the control wires (*Ack*, *Req*) are substantial.

Example 10.6 The Pipelined Datapath—Revisited

We have introduced both the signaling conventions and the concepts of the completion-signal generation. Now it is time to bring them all together. We do this with the example of the pipelined data path, which was presented earlier. A view of the self-timed data path, including the timing control, is offered in Figure 10.44. The logic functions *F*1 and *F*2 are implemented using dual-rail, differential logic.

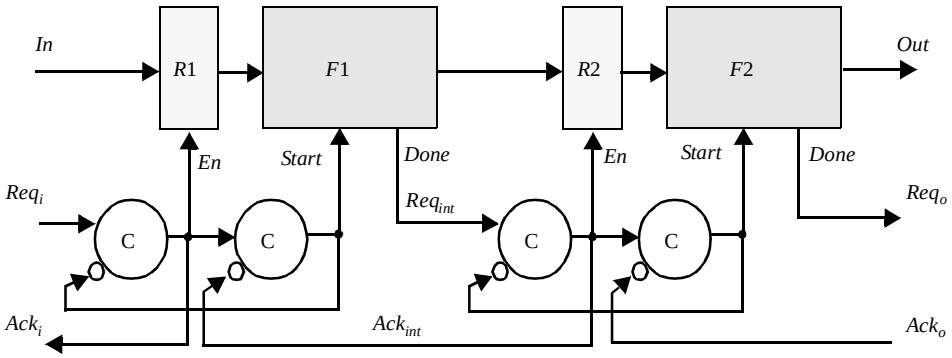


Figure 10.44 Self-timed pipelined datapath—complete composition.

To understand the operation of this circuit, assume that all *Req* and *Ack* signals, including the internal ones, are set to 0, what means there is no activity in the data path. All *Start* signals are low so that all logic circuits are in precharge condition. An input request ($Req_i\uparrow$) triggers the first C-element. The enable signal *En* of *R*1 is raised, effectively latching the input data into the register, assuming a positive edge-triggered or a level-sensitive implementation. $Ack_i\uparrow$ acknowledges the acceptance of the data. The second C-element is triggered as well, since Ack_{int} is low. This raises the *Start* signal and starts the evaluation of *F*1. At its completion, the output data is placed on the bus, and a request is initiated to the second stage ($Req_{int}\uparrow$), which acknowledges its acceptance by raising Ack_{int}.

At this point, stage 1 is still blocked for further computations. However, the input buffer can respond to the $Ack_i\uparrow$ event by resetting Req_i to its zero state ($Req_i\downarrow$). In turn, this lowers *En* and Ack_i. Upon receival of $Ack_{int}\uparrow$, *Start* goes low, the pre-charge phase starts, and

*F*1 is ready for new data. Note that this sequence corresponds to the four-phase handshake mechanism described earlier. The dependencies among the events are presented in a more pictorial fashion in the *state transition diagram* (*STG*) shown in Figure 10.45. These STGs can become very complex. Computer tools are often used to derive STGs that ensure proper operation and optimize the performance.

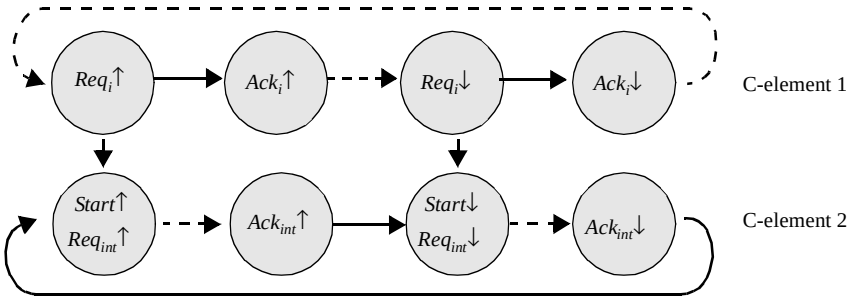


Figure 10.45 State transition diagram for pipeline stage 1. The nodes represent the signaling events, while the arrows express dependencies. Arrows in dashed lines express actions in either the preceding or following stage.

10.4.4 Practical Examples of Self-Timed Logic

As described in the previous section, self-timed circuits can provide significant performance advantage. Unfortunately, the overhead circuits preclude widespread application in general purpose digital computing. However, several applications exist that exploit the key concepts of self-timed circuits. A few examples that illustrate the use of self-timed concepts for either power savings or performance enhancement are presented below.

Glitch Reduction Using Self-timing

A major source of unnecessary switched capacitance in a large datapath such as bit-sliced adders and multipliers, is due to spurious transitions caused by glitch propagation. Imbalances in a logic network cause inputs of a logic gate or block to arrive at different times, resulting in glitching transitions. In large combinational blocks such as multipliers, transitions happens in waves as the primary input changes ripple through the logic. If a logic block can be enabled after all of the inputs settle, then the number of glitching transitions can be reduced. One approach for minimizing spurious transitions is the use of a self-timed gating approach that involves partitioning each computational logic block into smaller blocks and distinct phases. Tri-state buffers are inserted between each of these phases to prevent glitches from propagating further in the datapath (Figure 10.46). Assuming an arbitrary logic network in Figure 10.46, the outputs of logic block 1 will not be synchronized. When the tri-state buffers at the output of logic block 1 are enabled, the the computation of logic block 2 is allowed to proceed. To reduce glitching transitions, the tri-state buffer should be enabled only when the outputs of logic block 1 are ensured to be stable and valid. The control of the tri-state buffer can be performed through the use of a self-timed enable signal which is generated by passing the system clock through a delay chain

that models the critical path of the processor. The delay chain is then tapped at the points corresponding to the locations of the buffers, and the resulting signals are distributed throughout the chip. This technique succeeds in reducing the switched capacitance combinational logic blocks such as multipliers, even including the overhead of the delay chain, gating signal distribution and buffers [Goodman98].

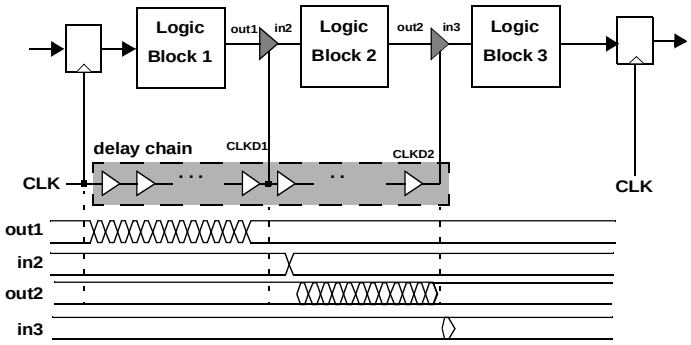


Figure 10.46 Application of self-timing for glitch reduction.

Post-Charge Logic

An interesting form of self-timed logic is self-resetting CMOS. This structure uses a different control structure than the conventional self-timed pipeline described earlier. The idea is that instead of checking if all of the logic blocks have completed their operation before transitioning to the reset stage, the idea is to precharge a logic block as soon as it completes it operation. Since the precharge happens after operation instead of before evaluation, it is often termed *post-charge logic*. A block diagram of post-charge logic is shown in Figure 10.47 [Williams00]. As seen from this block diagram, the precharging of L1

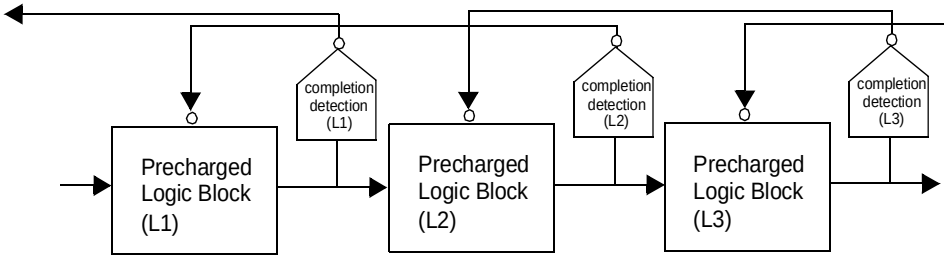


Figure 10.47 Self-resetting logic.

happen when the successor stage has finished evaluating. It is possible to precharge a block based on the completion of its own output, but care must be taken to ensure that the following stage has properly evaluated based on the output before it is precharged.

It should be noted that unlike other logic styles, the signals are represented as pulses, and are valid only for a limited duration. The pulse width must be shorter than the reset delay or else, there would be a contention between the precharge device and the evaluate switches. While this logic style offers potential speed advantages, special care must be taken to ensure correct timing. Also, circuitry that converts level signals to pulses are required. An example of self-resetting logic is shown in Figure 10.48 [Bernstein98]. Assume that all inputs are low, and *int* is initially precharged. If *A* goes high, *int* will fall, causing out to go high. This causes, the gate to precharge. When the PMOS precharge device is active, the inputs must be in a reset state to avoid contention.

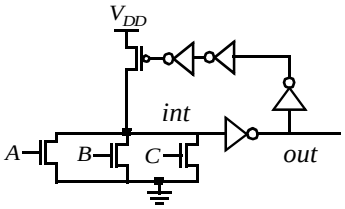


Figure 10.48 Self-resetting 3-input OR.

Clock-Delayed Domino

One interesting application of self-timed circuits using the delay-matching concept is Clock-Delayed Domino. This is a style of dynamic logic, where there is no global clock signal. Instead, the clock for one stage is derived from the previous stage. A block diagram of this is shown in Figure 10.49. The two inverter delays along the clock path emulate the worst case delay through the logic path (i.e., the Pulldown Network). Sometimes, there is a transmission gate inserted between the two inverters to accomplish this (the transmission gate is on with the gate terminal of NMOS tied to V_{DD} and the PMOS to *GND*). Clock-delayed Domino was used in the IBM's 1GHz Microprocessor and is used widely in high speed Domino logic. There are several advantages of using such a self-clocked timing abstraction. First, CD domino can provide both inverting and non-inverting function. This alleviates a major limitation of conventional Domino that is only capable of non-inverting logic. The inverter after the pulldown network is not essential as the clock arrives to the next stage only after the current stage has evaluated. Also, notice that is possible to eliminate the "foot-switch" in the later stages, as the clock-evaluation edge arrives only when the input is stable. This provides significant speed up of the logic critical path. A careful analysis of the timing shows that the short circuit power can be eliminated.

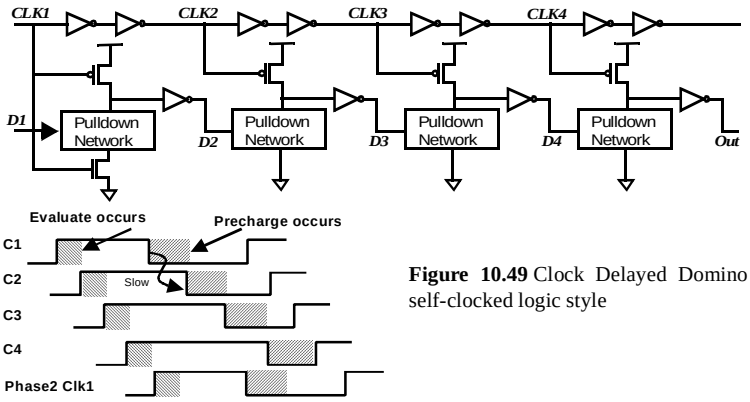


Figure 10.49 Clock Delayed Domino Logic. A self-clocked logic style

10.5 Synchronizers and Arbiters*

10.5.1 Synchronizers—Concept and Implementation

Even though a complete system may be designed in a synchronous fashion, it must still communicate with the outside world, which is generally asynchronous. An asynchronous input can change value at any time related to the clock edges of the synchronous system, as is illustrated in Figure 10.50.

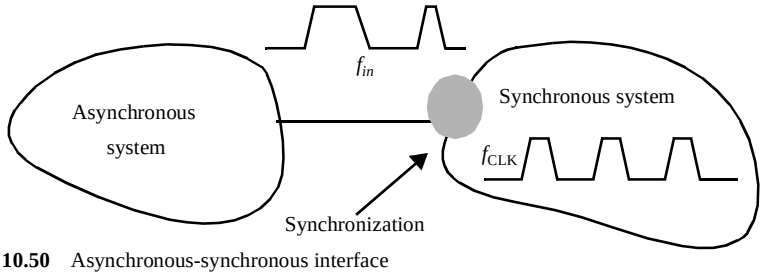


Figure 10.50 Asynchronous-synchronous interface

Consider a typical personal computer. All operations within the system are strictly orchestrated by a central clock that provides a time reference. This reference determines what happens within the computer system at any point in time. This synchronous computer has to communicate with a human through the mouse or the keyboard, who has no knowledge of this time reference and might decide to press a keyboard key at any point in time. The way a synchronous system deals with such an asynchronous signal is to sample or poll it at regular intervals and to check its value. If the sampling rate is high enough, no transitions will be missed—this is known as the Nyquist criterion in the communication community. However, it might happen that the signal is polled in the middle of a transition. The resulting value is neither low or high but undefined. At that point, it is not clear if the key was pressed or not. Feeding the undefined signal into the computer could be the source of all types of trouble, especially when it is routed to different functions or gates that might interpret it differently. For instance, one function might decide that the key is pushed and start a certain action, while another function might lean the other way and issue a competing command. This results in a conflict and a potential crash. Therefore, the undefined state must be resolved in one way or another before it is interpreted further. It does not really matter what decision is made, as long as a unique result is available. For instance, it is either decided that the key is not yet pressed, which will be corrected in the next poll of the keyboard, or it is concluded that the key is already pressed.

Thus, an asynchronous signal must be resolved to be either in the high or low state before it is fed into the synchronous environment. A circuit that implements such a decision-making function is called a *synchronizer*. Now comes the bad news—**building a perfect synchronizer that always delivers a legal answer is impossible!** [Chaney73, Glasser85] A synchronizer needs some time to come to a decision, and in certain cases this time might be arbitrarily long. An asynchronous/synchronous interface is thus always prone to errors called *synchronization failures*. The designer's task is to ensure that the probability of such a failure is small enough that it is not likely to disturb the normal system behavior. Typically, this probability can be reduced in an exponential fashion by waiting longer before

making a decision. This is not too troublesome in the keyboard example, but in general, waiting affects system performance and should therefore be avoided to a maximal extent.

To illustrate why waiting helps reduce the failure rate of a synchronizer, consider a synchronizer as shown in Figure 10.51. This circuit is a latch that is transparent during the low phase of the clock and samples the input on the rising edge of the clock *CLK*. However, since the sampled signal is not synchronized to the clock signal, there is a finite probability that the set-up time or hold time of the latch is violated (the probability is a strong function of the transition frequencies of the input and the clock). As a result, one the clock goes high, there is a the chance that the output of the latch resides somewhere in the undefined transition zone. The sampled signal eventually evolves into a legal 0 or 1 even in the latter case, as the latch has only two stable states.

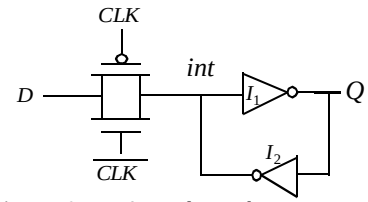


Figure 10.51 A simple synchronizer.

Example 10.7 Flip-Flop Trajectories

Figure 10.52 shows the simulated trajectories of the output nodes of a cross-coupled static CMOS inverter pair for an initial state close to the metastable point. The inverters are composed of minimum-size devices.

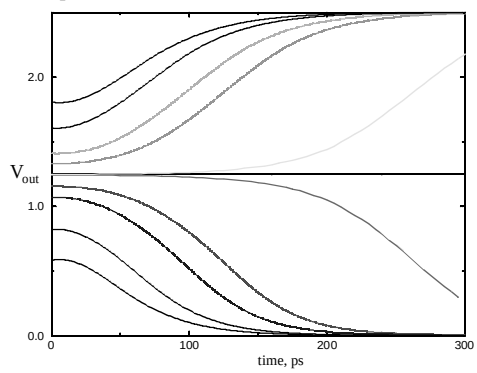


Figure 10.52 Simulated trajectory for a simple *flip-flop* synchronizer.

If the input is sampled such that cross-coupled inverter starts at the metastable point, the voltage will remain at the metastable state forever in the absense of noise. If the data is sampled with a small offset (positive or negative), the differential voltage will evolve in an exponential form, with a time constant that is dependent on the strength of the transistor as well as the parasitic capacitances. The time it takes to reach the acceptable signal zones depends upon the initial distance of the sampled signal from the metastable point.

In order to determine the required waiting period, let us build a mathematical model of the behavior of the bistable element and use the results to determine the probability of synchronization failure as a function of the waiting period. This model is used to compute the range of values for $v(0)$ that still cause an error, or a voltage in the undefined range, after a waiting period T. A signal is called undefined if its value is situated between V_{IH} and V_{IL}.

$$V_{MS} - (V_{MS} - V_{IL})e^{-T/\tau} \le v(0) \le V_{MS} + (V_{IH} - V_{MS})e^{-T/\tau} \tag{10.8}$$

Eq. (10.8) conveys an important message: The range of input voltages that cause a synchronization error decreases exponentially with the waiting period *T*. Increasing the waiting period from 2τ to 4τ decreases the interval and the chances of an error by a factor of 7.4.

Some information about the asynchronous signal is required in order to compute the probability of an error. Assume that V_{in} is a periodical waveform with an average period T_{signal} between transitions and with identical rise and fall times t_r. Assume also that the slopes of the waveform in the undefined region can be approximated by a linear function (Figure 10.53). Using this model, we can estimate the probability P_{init} that $v(0)$, the value of V_{in} at the sampling time, resides in the undefined region.

$$P_{init} = \frac{\left(\frac{V_{IH} - V_{IL}}{V_{swing}}\right) t_r}{T_{signal}} \tag{10.9}$$

The chances for a synchronization error to occur depend upon the frequency of the synchronizing clock ϕ. The more sampling events, the higher the chance of running into an error. This means that the average number of synchronization errors per second $N_{sync}(0)$ equals Eq. (10.10) if no synchronizer is used.

$$N_{sync}(0) = \frac{P_{init}}{T_\phi} \tag{10.10}$$

where T_ϕ *is* the sampling period.

From Eq. (10.8) we learned that waiting a time *T* before observing the output reduces exponentially the probability that the signal is still undefined.

$$N_{sync}(T) = \frac{P_{init}e^{-T/\tau}}{T_\phi} = \frac{(V_{IH} - V_{IL})e^{-T/\tau}}{V_{swing}} \frac{t_r}{T_{signal}T_\phi} \tag{10.11}$$

The robustness of an asynchronous-synchronous interface is determined by the following parameters: signal switching rate and rise time, sampling frequency, and waiting time *T*.

Example 10.8 Synchronizers and Mean Time-to-Failure

Consider the following design example. T_ϕ = 5 nsec, which corresponds to a 200 Mhz clock), $T = T_\phi$ = 5 nsec, T_{signal} = 50 nsec, t_r =0.5 nsec, and τ = 150 psec (as obtained in Example 10.7).

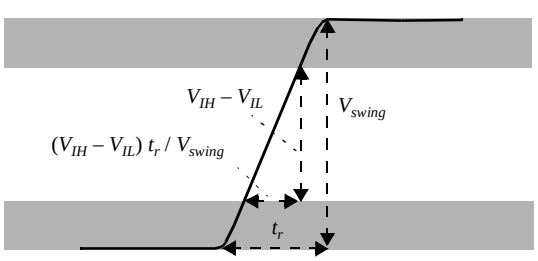


Figure 10.53 Linear approximation of signal slope.

From the VTC of a typical CMOS inverter, it can be derived that $V_{IH} - V_{IL}$ approximately equals 0.5 V for a voltage swing of 2.5 V. Evaluation of Eq. (10.11) yields an error probability of 1.38x 10^{-9} errors/sec. The inverse of N_{sync} is called the *mean time-to-failure*, or the MTF, and equals 7×10^8sec, or 23 years. If no synchronizer was used, the MTF would only have been 2.5 μsec!

Design Consideration

When designing a synchronous/asynchronous interface, we must keep in mind the following observations:

- The acceptable failure rate of a system depends upon many economic and social factors and is a strong function of its application area.
- The exponential relation in Eq. (10.11) makes the failure rate extremely sensitive to the value of τ. Defining a precise value of τ is not easy in the first place. τ varies from chip to chip and is a function of temperature as well. The probability of an error occurring can thus fluctuate over large ranges even for the same design. A worst-case design scenario is definitely advocated here. If the worst-case failure rate exceeds a certain criterion, it can be reduced by increasing the value of *T*. A problem occurs when *T* exceeds the sampling period T_ϕ. This can be avoided by cascading (or pipelining) a number of synchronizers, as shown in Figure 10.54. Each of those synchronizers has a waiting period equal to T_ϕ. Notice that this arrangement requires the ϕ-pulse to be short enough to avoid race conditions. The global waiting period equals the sum of the *T*s of all the individual synchronizers. The increase in MTF comes at the expense of an increased latency.

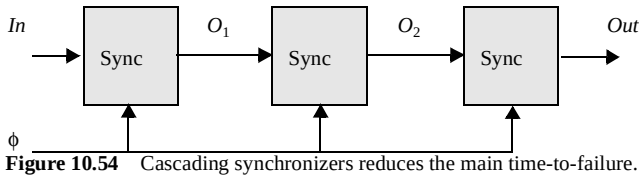


Figure 10.54 Cascading synchronizers reduces the main time-to-failure.

- Synchronization errors are very hard to trace due to their probabilistic nature. Making the mean time-to-failure very large does not preclude errors. The number of synchronizers in a system should therefore be severely restricted. A maximum of one or two per system is advocated.

10.5.2 Arbiters

Finally, a sibling of the synchronizer called the *arbiter*, interlock element, or mutual-exclusion circuit, should be mentioned. An arbiter is an element that decides which of two events has occurred first. Such components for instance allow multiple processors to access a single resource, such as a large shared memory. A synchronizer is actually a

special case of an arbiter, since it determines if a signal transition happened before or after a clock event. A synchronizer is thus an arbiter with one of its inputs tied to the clock.

An example of a mutual-exclusion circuit is shown in Figure 10.55. It operates on two input-request signals, that operate on a four-phase signaling protocol; that is, the *Req(uest)* signal has to go back to the reset state before a new *Req(uest)* can be issued. The output consists of two *Ack(nowledge)* signals that should be mutually exclusive. While *Requests* may occur concurrently, only one of the *Acknowledges* is allowed to go high. The operation is most easily visualized starting with both inputs low—neither device issuing a request—nodes *A* and *B* high, and both *Acknowledges* low. An event on one of the inputs (e.g., *Req1*↑) causes the flip-flop to switch, node *A* goes low, and *Ack1*↑. Concurrent events on both inputs force the flip-flop into the metastable state, and the signals *A* and *B* might be undefined for a while. The cross-coupled output structure keeps the output values low until one of the NAND outputs differs from the other by more than a threshold value V_T. This approach eliminates glitches at the output.

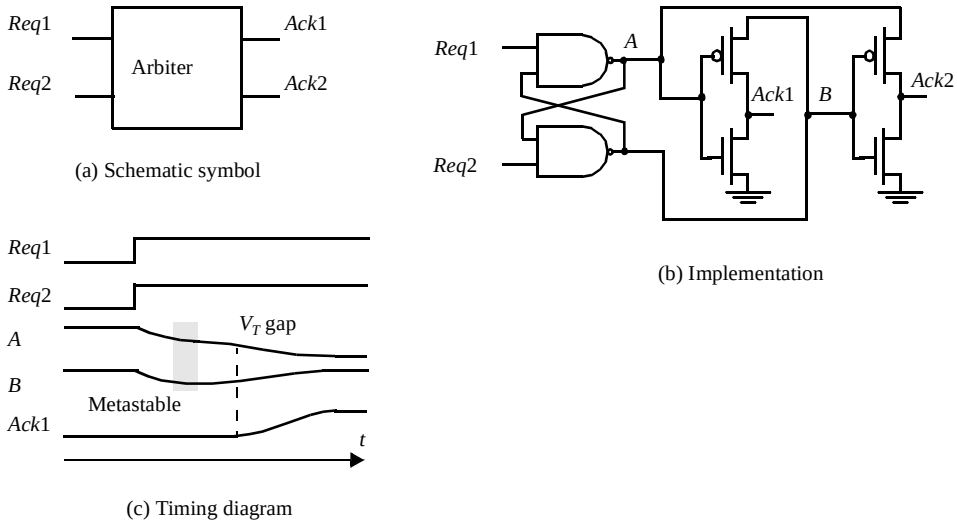


Figure 10.55 Mutual-exclusion element (or arbiter).

10.6 Clock Synthesis and Synchronization Using a Phase-Locked Loop

There are numerous digital applications that require the on-chip generation of a periodic signal. Synchronous circuits need a global periodic clock reference to drive sequential elements. Current microprocessors and high performance digital circuits require clock frequencies in the gigahertz range. Crystal oscillators generate accurate, low-jitter clocks with a frequency range from 10's of Megahertz to approximately 200MHz. To generate a higher frequency required by digital circuits, a *phase-locked loop* (PLL) structure is typically used. A PLL takes an external low-frequency reference crystal frequency signal and multiplies its frequency by a rational number *N* (see the left side of Figure 10.56).

PLLs are also used to perform synchronization of communication between chips. Typically as shown in Figure 10.56, a reference clock is sent along with the parallel data being communicated (in this example only the transmit path from chip 1 to chip 2 is shown). Since chip-to-chip communication occurs at a lower rate than the on-chip clock rate, the reference clock is a divided but in-phase version of the system clock. The reference clock synchronizes all input flip-flops on chip 2; this can present a significant clock load for wide data busses. Introducing clock buffers to deal with this problem unfortunately introduces skew between the data and sample clock. A PLL, using feedback, can be align (i.e., de-skew) the output of the clock buffer with respect to the data. In addition, for the configuration shown in Figure 10.56, the PLL can multiply the frequency of the incoming reference clock, allowing the reference clock to be a fraction of the data rate.

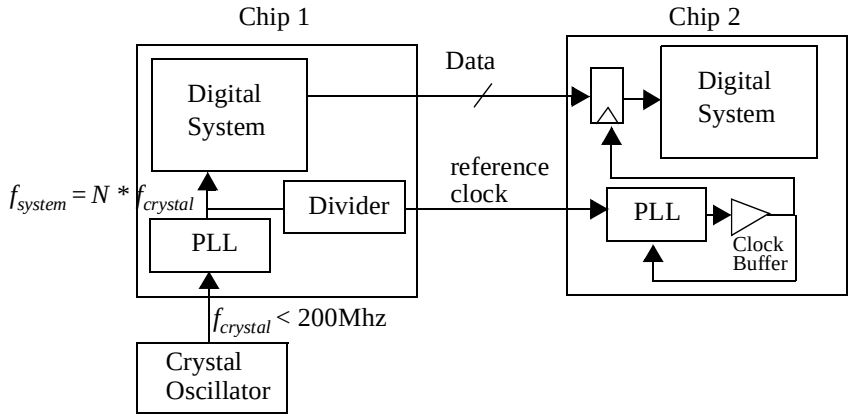


Figure 10.56 Applications of Phase Locked Loops (PLL).

10.6.1 Basic Concept

Periodic signals of known frequency can be discribed exactly by only one parameter, their phase. More accurately a set of two or more periodic signals of the same frequency can be well defined if we know one of them and its phase with respect to the other signals, as in Figure 10.57.Here $\phi_1$and ϕ_2 represent the phase of the two signals. The relative phase is defined as the difference between the two phases.

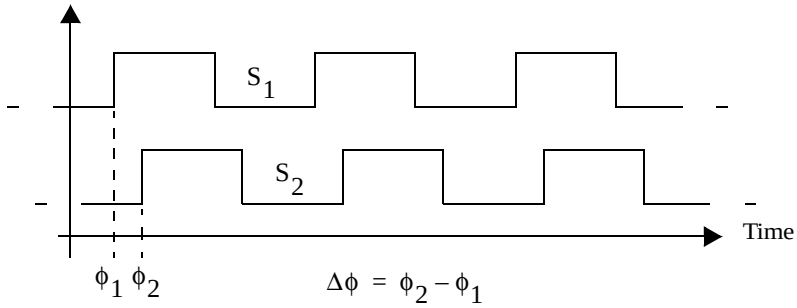


Figure 10.57 Relative and absolute phase of two periodic signals

A PLL is a complex, nonlinear feedback circuit, and its basic operation is understood with the aid of Figure 10.58 [Jeong87]. The *voltage-controlled oscillator* (VCO) takes an analog control input and generates a clock signal of the desired frequency. In general, there is a non-linear relationship between the control voltage (v_{cont}) and the output frequency. To synthesize a system clock of a particular frequency, it necessary to set the control voltage to the appropriate value. This is function of the rest of the blocks (i.e., feedback loop) in the PLL. The feedback loop is critical to tracking process and environmental variations. The feedback also allows frequency multiplication.

The reference clock is typically generated off-chip from an accurate crystal reference. The reference clock is compared to a divided version of the system clock (i.e., the local clock). The local clock and reference clock are compared using a phase detector that compares the phase difference between the signals and produces an *Up* or *Down* signal when the local clock lags or leads the reference signal. It detects which of the two input signals arrives earlier and produces an appropriate output signal. Next, the *Up* and *Down* signals are fed into a charge pump, which translates the digital encoded control information into an analog voltage [Gardner80]. An *Up* signal increases the value of the control voltage and speeds up the VCO, which causes the local signal to catch up with the reference clock. A *Down* signal, on the other hand, slows down the oscillator and eliminates the phase lead of the local clock.

Passing the output of the charge pump directly into the VCO creates a jittery clock signal. The edge of the local clock jumps back and forth instantaneously and oscillates around the targeted position. As discussed earlier, *clock jitter*, is highly undesirable, since it reduces the time available for logic computation and therefore should be kept within a given percentage of the clock period. This is partially accomplished by the introduction of the *loop filter*. This low-pass filter removes the high-frequency components from the VCO control voltage and smooths out its response, which results in a reduction of the jitter. Note that the PLL structure is a feedback structure and the addition of extra phase shifts, as is done by a high-order filter, may result in instability. Important properties of a PLL are its *lock range*—the range of input frequencies over which the loop can maintain functionality; the *lock time*—the time it takes for the PLL to lock onto a given input signal; and the jitter. When in lock, the system clock is N-times the reference clock frequency.

A PLL is an analog circuit and is inherently sensitive to noise and interference. This is especially true for the loop filter and VCO, where induced noise has a direct effect on the resulting clock jitter. A major source of interference is the noise coupling through the supply

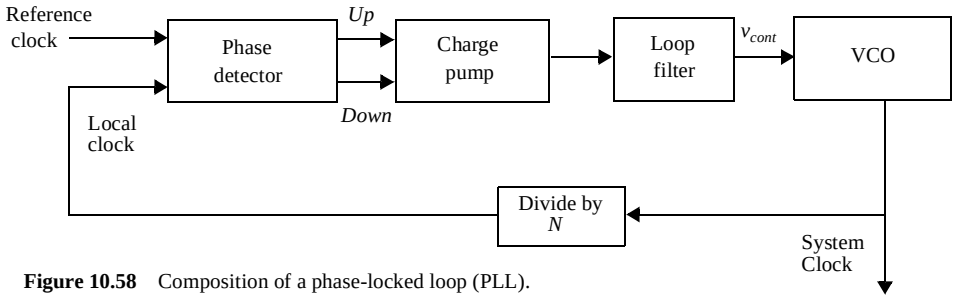


Figure 10.58 Composition of a phase-locked loop (PLL).

rails and the substrate. This is particularly a concern in digital environments, where noise is introduced due to a variety of sources. Analog circuits with a high supply rejection, such as differential VCOs, are therefore desirable [Kim90]. In summary, integrating a highly sensitive component into a hostile digital environment is nontrivial and requires expert analog design. Below a more detailed description of various components of a PLL is given.

10.6.2 Building Blocks of a PLL

Voltage Controlled Oscillator (VCO)

A VCO generates a periodic signal, whose frequency is a linear function of the input control voltage v_{cont}. In other words a VCO is characterized by, $\omega = \omega_0 + K_{vco} \cdot v_{cont}$. As the phase is the time integral of the frequency, the output phase of the VCO block is given by,

$$\phi_{out} = \omega_0 t + K_{vco} \cdot \int_{-\infty}^{t} v_{cont} dt \tag{10.12}$$

where K_{vco} is the gain of the VCO given in rad/s/V, and ω_0 is a fixed frequency offset. v_{cont} controls a frequency centered around ω_0. The output signal has the form

$$x(t) = A \cdot \cos\left(\omega_0 t + K_{vco} \cdot \int_{-\infty}^{t} v_{cont} dt\right) \tag{10.13}$$

Phase Detectors

The phase detector determines the relative phase difference between two incoming signals and outputs a signal that is proportional to this phase difference. This output signal is then used to adjust the output of the VCO and thus align the two inputs via a feedback network. One of the inputs to the phase detector is a reference clock that is typically generated off-chip while the other clock input is a divided version of the VCO. Two basic types of phase detectors are commonly used. These include the XOR gate and the phase frequency detector (PFD).

XOR Phase Detector. The XOR phase detector is the simplest phase detector. The XOR is useful as a phase detector since the time when the two inputs are different (or same) represents the relative phase. Figure 10.59 shows the XOR of two waveforms. The output of

the XOR is low pass filtered and acts as a control voltage to the VCO. The output (low pass filtered) as a function of the phase error is also shown.

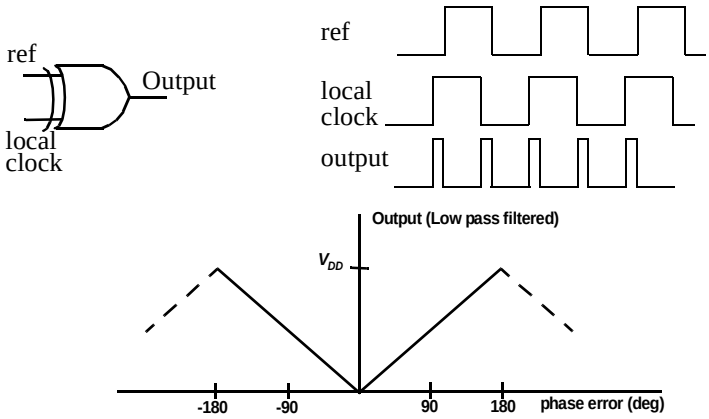


Figure 10.59 The XOR as a phase detector.

For this detector any deviation in a positive or negative direction from the the perfect in-phase condition (i.e., phase error of zero) produces the same change in duty factor resulting in the same average voltage. Thus the linear phase range is only 180 degrees. Using such a phase detector, a PLL will lock to a quadrature phase relationship (i.e., 1/4 cycle offset) between the two inputs. A drawback of the XOR phase detector is that it may lock to a multiple of the clock frequency. If the local clock is a multiple of the reference clock frequency, the output of the phase detector will still be a square wave of 50% duty cycle, albeit at a different frequency. The filtered version of this signal will be identical to that of the truly locked state and thus the VCO will operate at the nominal frequency.

Phase-Frequency Detector. The phase-frequency detector (PFD) is the most commonly used form of phase detector, and it solves several of the shortcomings of the detectors discussed above. As the name implies, the output of the PFD is dependent both on the phase and frequency difference of the applied signals. Accordingly, it cannot lock to an incorrect

multiple of the frequency. The PFD takes two clock inputs and produces two outputs, UP and DOWN as shown in Figure 10.60.

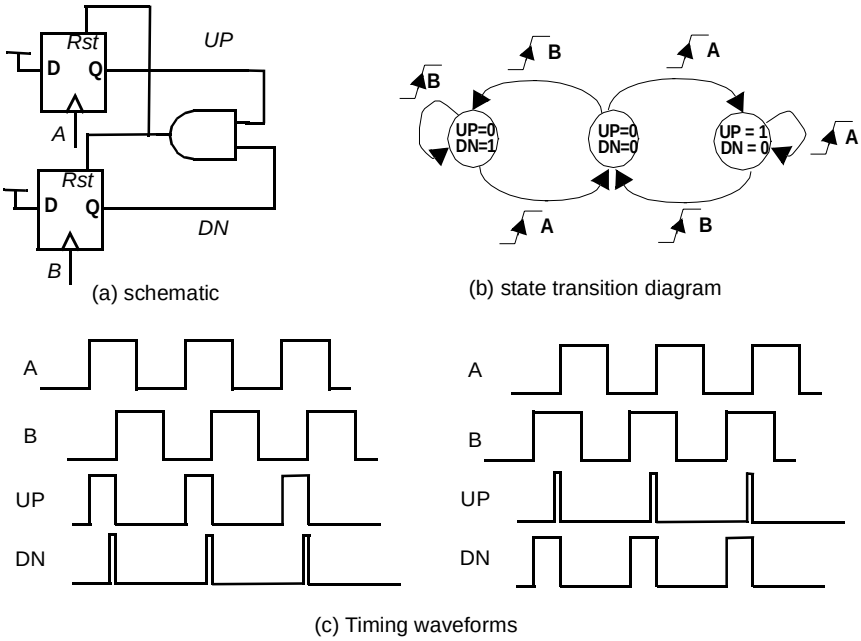


Figure 10.60 Phase-Frequency Detector. (a) Schematic (b) State Transition Diagram (c) Timing.

The PFD is a state machine with 3 states. Assume that both UP and DN outputs are initially low. When input A leads B, the UP output is asserted on the rising edge of input A. The UP signal remain in this state until a low-to-high transition occurs on input B. At that time, the DN output is asserted, causing both flip-flops to reset through the asynchronous reset signal. Notice that a short pulse proportional to the phase error is generated on the DN signal, and that there is a small pulse on the DN output, whose duration is is equal to the delay throught he AND gate and register reset delay. The pulse width of the UP pulse is equal to the phase error between the two signal. The roles are reversed for the case when input B lags A, and a pulse proportional to the phase error is generated on the DN output. If the loop is in lock, short pulses will be generated on the UP and DN outputs.

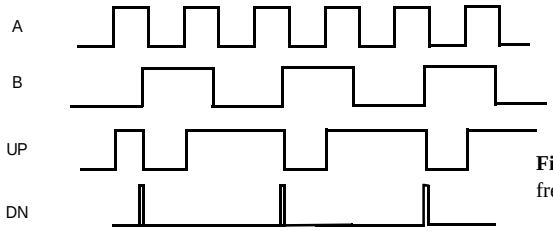


Figure 10.61 Timing of the PFD measuring frequency error.

The circuit also acts as a frequency detector, providing a measure of the frequency error (Figure 10.61). For the case when A is at a higher frequency than B, the PFD generates a lot more UP pulses (with the average proportional to the frequency difference),

while the DN pulses average close to zero. The exactly the opposite is true for the case when B has a frequency larger than A — many more pulses are generated on the DN output than the UP output.

The phase characterisctics of the phase detector is shown in Figure 10.62. Notice that the linear range has been expanded to 4π.

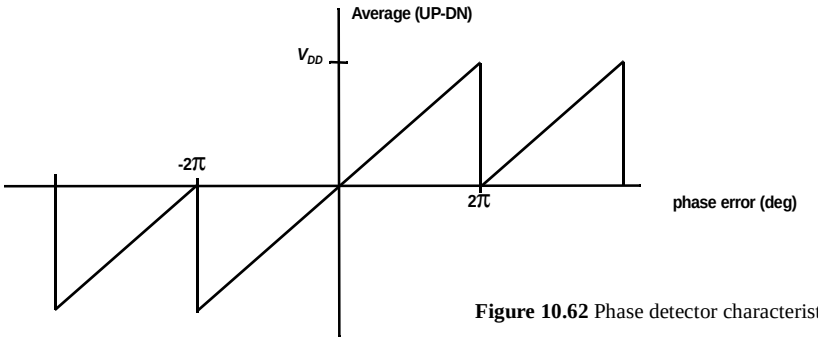


Figure 10.62 Phase detector characteristic.

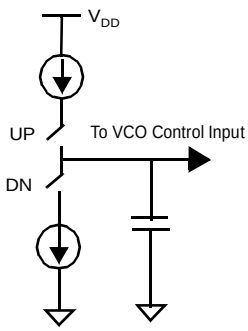


Figure 10.63 Charge Pump.

Charge Pump

The UP/DN pulses must be converted to an analog voltage that controls the VCO. One possible implementation is shown in Figure 10.63. A pulse on the UP signal adds a charge packet proportional to the size of the UP pulse, and a pulse on the DN signal removes a charge packet proportional to the DN pulse. If the width of the UP pulse is large than the DN pulse, then there is a net increase in the control voltage. This effectively increases the frequency of the VCO.

Simulation

The settling time of a PLL is the time it takes it reach steady state behavior. The period of the startup transient is strongly dependent on the bandwidth of the loop filter. Figure 10.64 shows a spice level simulation of a PLL (an ideal VCO is used to speed up the simulation) implemented in 0.25µm CMOS. In this example a reference frequency of 100Mhz is chosen and the PLL multiplies this frequency by 8 to 800Mhz. The figure illustrates the transient response and settling process of the control voltage to the VCO. Once the control voltage reaches its final value, the output frequency (the clock to the digital system) settles to its final value. The simulation on the right shows the reference frequency, the output of the divider and the output frequency. As illustrated in this plot, the top graph show lock in which $f_{out} = 8 * f_{ref}$. The lower graph shows the waveforms during the locking process where the output is not in phase with the input and the frequencies are not related through the divide ratio

.

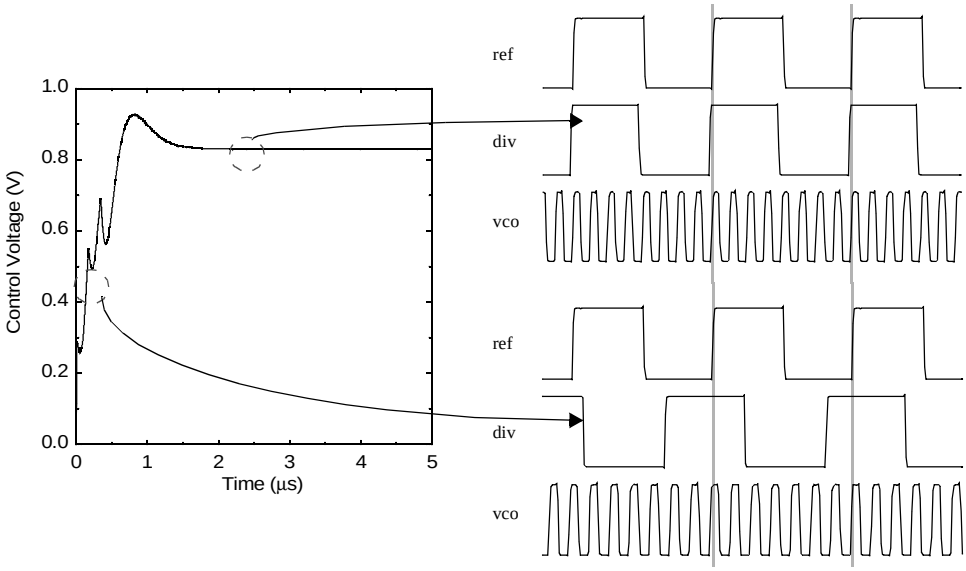


Figure 10.64 Spice simulation of a PLL. The control voltage of the VCO is shown along with waveforms before lock and after lock.

Summary

In a short span of time, phased-locked loops have become an essential component of any high-performance digital design. Their design requires considerable skill, integrating analog circuitry into a hostile digital environment. Yet, experience has demonstrated that this combination is perfectly feasible, and leads to new and better solutions.

10.7 Future Directions

This section highlights some of the trends in high-performance and low-power timing optimization.

10.7.1 Distributed Clocking Using DLLs

A recent trend in high-performance clocking is the use of *Delay-Locked Loop* (DLL) structures, a small variation of the PLL structure. A schematic of a DLL is shown in Figure 10.65 [Maneatis00]. The key component of a DLL is a *voltage-controlled delay line* (VCDL). It consists of a cascade of adjustable delay elements (for instance, a current-starved inverter). The idea is to delay the output clock such that it perfectly lines up with the reference. Unlike a VCO, there is no clock generator. The reference frequency is fed into the input of the VCDL. Similar to a PLL structure, a phase-detector compares the reference frequency to the output of the delay line (F0), and generates an UP/DN error signal. Note that only a phase detection is required instead of a PFD. When in lock there is no

error between the two clocks. The function of the feedback is to adjust the delay through the VCDL such that the rising edge of the input reference clock (f_{REF}), and the output clock f_O are aligned .

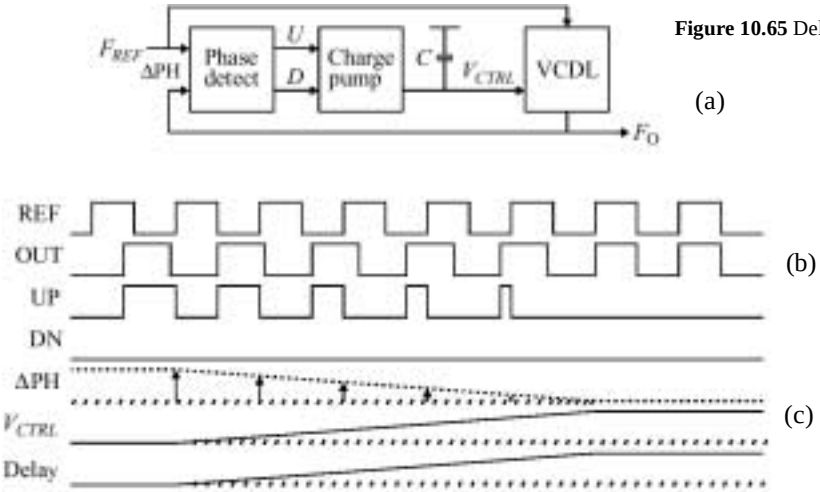


Figure 10.65 Delay-Locked Loop.

A qualitative sketch of the signals in the DLL is shown in Figure 10.66c. Initially, the DLL is out of lock. Since the first edge of the output arrives before the reference edge, an UP pulse of width equal to the error between the two signals. The role of the charge pump is to generate a charge packet proportional to the error, increasing the voltage of the VCDL control voltage. This causes the edge of the output signal to be delayed in the next cycle (this implementation of the VCDL assumes that a large voltage results in larger delay). After many cycles, the phase error is corrected, and the two signals are in lock.

Figure 10.67 shows the application of a DLL strcuture to clock distribution. The chip is partitioned into many small regions (or tiles). A global clock is distributed to each tile in a low-skew manner (e.g., this could be done through the package or using low skew on-chip routing schemes). For purpose of simplicity, the Figure shows a two-tile chip, but this is easily extended to many regions. Inside each tile, the global clock is buffered before driving the digital load. In front of each buffer is a VCDL. The goal of the clock network is to deliver a signal to the digital circuit with close-to-zero skew and jitter. Unfortunately, the static and dynamic variations of the buffers cause the phase error between the buffered clocks to be non-zero and time-varying. This is especially a concern since the delay through the buffer chain can equal multiple clock cycles. The feedback inside each tile adjusts the control voltage of VCDL, such that the buffered output is locked in phase to the global input clock. The feedback loop compensates for both static process variations and for slow dynamic variations (e.g., temperature). Such configurations have become common in high performance digital microprocessors.

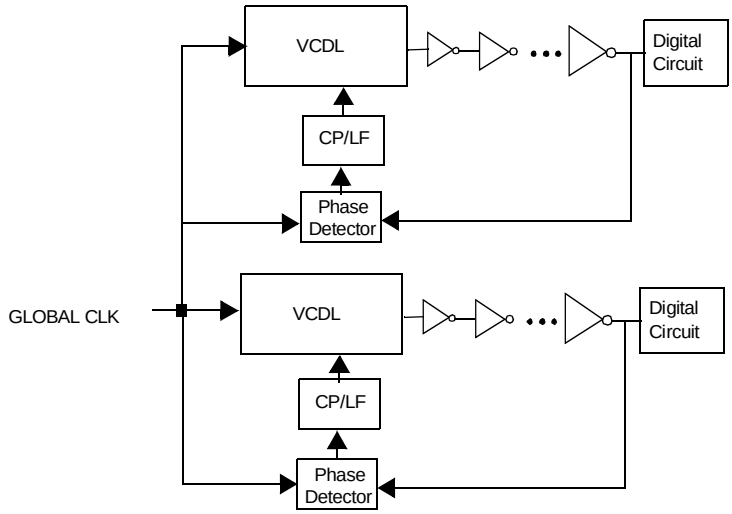


Figure 10.66 DLL Approach to Clock Distribution.

10.7.2 Optical Clock Distribution

It is clear that there are some fundamental problems associated with electrical synchronization techniques for future high-performance multi-GHz systems. The performance of a digital is fundametally limited by process and environmental variations. Even with aggressive active clock management schemes such as the use of DLLs and PLLs, the variations in power supply and clock load result in unacceptable clock uncertainity. As a result, there has been a major recent push towards the use of optics for system wide synchronization. An excellent review of the rationale and trade-offs in optical interconnects vs. electrical interconnect is given in [Miller00].

The potential advantage of optical technology for clock distribution is due to the fact that the delay is not sensitive to temperature and the clock edges don't degrade over long distances (e.g., 10's meters). In fact, it is possible to deliver an optical clock signal with 10-100ps of uncertainty for 10's meters. Of course, the performance of an optical system is limited by the speed of light. Optical clocks can distribution on-chip via waveguides or freespace. Figure 10.67 shows the plot of an optical clock architecutre using waveguides. The off-chip optical source is brought to the chip, distributed through waveguides, and converted through receiver circuitry to a local electrical clock distribution network. The chip is divided into small sections (or tiles) and each the global clock is distributed from the photon source (the most popular choice being a laser for its ease of use) through waveguides with splitters and bends to each of the sections. Notice that an H-treee is used in distributing the optical clock. At the end of the waveguide is a photodetector (which can be implemented using silicon or germanium).Once reaching the detector in each section, the global clock optical pulses are converted into current pulses. These current pulses have a very small magnitude (10's μA) and they are fed into a transimpedance amplifier of the

optoelectronic circuitry that amplifes the signal into voltage levels appropriate for digital processing. into voltage signals. The electrical clock is then distributed through conventional techniques to the local load. Optics has the additional advantage that many of the difficulties with electromagnetic wave phenomena are avoided (e.g., crosstalk or inductive coupling). A common problem in electrical signals is reflection from the loads, often requiring termination, hence increasing power dissipation. Optical clocks avoid this probelm and don't require termination [Miller00].

In such an approach, the skew of the global clock to the photodectors is virtually zero. There are some variations in the arrival time of the optical signal (e.g., due to variations at bends cause different energy loss along different paths). However, the optical receiver is typically composed of multiple stages to amplify the small current pulses and is susceptible to process and environmental variations. The sources of variations are very similar to a conventional electrical approach including threshold and device variations, power supply and temperature variations, and variations in the local drivers.

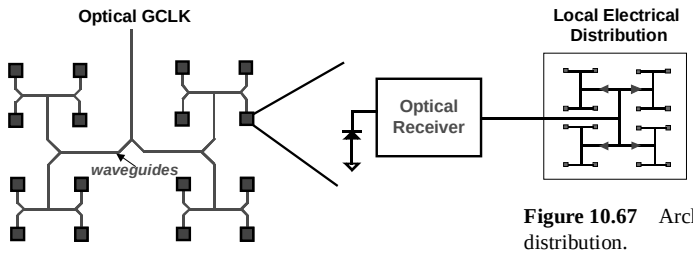


Figure 10.67 Architecture for optical clock distribution.

Optical clocking may an important future in high-performance systems. However, the challenges of dealing with process variations in the opto-electronic circuitry must be addressed for this to become a reality.

10.8 Perspective: Synchronous versus Asynchronous Design

The self-timed approach offers a potential solution to the growing clock-distribution problem. It translates the global clock signal into a number of local synchronization problems. Independence from physical timing constraints is achieved with the aid of completion signals. Handshaking logic is needed to ensure the logical ordering of the circuit events and to avoid race conditions. This requires adherence to a certain protocol, which normally consists of either two or four phases.

Despite all its advantages, self-timing has only been used in a number of isolated cases. Examples of self-timed circuits can be found in signal processing [Jacobs90], fast arithmetic units (such as self-timed dividers) [Williams87], simple microprocessors [Martin89] and memory (static RAM, FIFOs). In general, synchronous logic is both faster and simpler since the overhead of completion-signal generation and handshaking logic is avoided. The design a fool-proof network of handshaking units, that is robust with respect to races, live-lock, and dead-lock, is nontrivial and requires the availability of dedicated design-automation tools.

On the other hand, distributing a clock at high speed becomes exceedingly difficult. This was amply illustrated by the example of the 21164 Alpha microprocessor. Skew man-

agement requires extensive modeling and analysis, as well as careful design. It will not be easy to extend this methodology into the next generation of designs. This observation is already reflected in the fact that the routing network for the latest generation of massively parallel supercomputers is completely implemented using self-timing [Seitz92]. For self-timing to become a mainstream design technique however (if it ever will), further innovations in circuit and signaling techniques and design methodologies are needed. Other alternative timing approaches might emerge as well. Possible candidates are fully asynchronous designs or islands of synchronous units connected by an asynchronous network.

10.9 Summary

This chapter has explored the timing of sequential digital circuits.

- An in-depth analysis of the synchronous digital circuits and clocking approaches was presented. Clock *skew* and *jitter* has a major impact on the functionality and performance of a system. Important parameters are the clocking scheme used and the nature of the clock-generation and distribution network.
- Alternative timing approaches, such as self-timed design, are becoming attractive to deal with clock distribution problems. Self-timed design uses completion signals and handshaking logic to isolate physical timing constraints from event ordering.
- The connection of synchronous and asynchronous components introduces the risk of synchronization failure. The introduction of synchronizers helps to reduce that risk, but can never eliminate it.
- Phase-locked loops are becoming an important element of the digital-designer's tool box. They are used to generate high speed clock signals on a chip. The analog nature of the PLL makes its design a real challenge.
- Important trends for clock distribution include the use of delay-locked loops to actively adjust delays on a chip.
- The key message of this chapter is that synchronization and timing are among the most intriguing challenges facing the digital designer of the next decade.

10.10 To Probe Further

While system timing is an important topic, no congruent reference work is available in this area. One of the best discussions so far is the chapter by Chuck Seitz in [Mead80, Chapter 7]. Other in-depth overviews are given in [Bakoglu90, Chapter 8], [Johnson93, Chapter 11], [Hatamian88], and [Chandrakasan00]. A collection of papers on clock distribution networks is presented in [Friedman95]. Numerous other publications are available on this topic in the leading journals, some of which are mentioned below.

REFERENCES

[Abnous93] A. Abnous and A. Behzad, “A High-Performance Self-Timed CMOS Adder,” in *EE241 Final Class Project Reports*, by J. Rabaey, Univ. of California—Berkeley, May 1993.

[Bailey00] D. Bailey, “Clock Distribution,” in [Chandrakasan00].

[Bakoglu90] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, pp. 338–393, 1980.

[Boning00] D. Boning, S. Nassif, “Models of Process Variations in Device and Interconnect” in [Chandrakasan00].

[Bowhill95] W. Bowhill et al., “A 300 MHz Quad-Issue CMOS RISC Microprocessor,” *Technical Digest of the 1995 ISSCC Conference*, San Fransisco, February 1995.

[Bernstein98] K. Bernstein et. al, *High Speed CMOS Design Styles*, Kluwer Academic Publishers, 1998.

[Bernstein00] K. Bernstein, “Basic Logic Families”, in [Chandrakasan00].

[Chandrakasan00] A. Chandrakasan, W. Bowhill, and F. Fox, *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2000.

[Chaney73] T. Chaney and F. Rosenberger, “Anomalous Behavior of Synchronizer and Arbiter Circuits,” *IEEE Trans. on Computers*, vol. C-22, April 1973, pp. 421–422.

[Dally 98] W. Dally and J. Poulton, *Digital Systems Engineering*, Cambridge University Press, 1998.

[Dopperpuhl92] D. Dopperpuhl et al., “A 200 MHz 64-b Dual Issue CMOS Microprocessor,” *IEEE Journal on Solid State Circuits*, vol. 27, no. 11, Nov. 1992, pp. 1555–1567.

[Friedman95] E. Friedman, ed., *Clock Distribution Networks in VLSI Circuits and Systems*, IEEE Press, 1995.

[Gardner80] F. Gardner, “Charge-Pump Phase-Locked Loops,” I*EEE Trans. on Communications*, vol. COM-28, November 1980, pp. 1849–1858.

[Glasser85] L. Glasser and D. Dopperpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, 1985, pp. 360–365.

[Goodman98] J. Goodman, A. P. Dancy, A. P. Chandrakasan, "An Energy/Security Scalable Encryption Processor Using an Embedded Variable Voltage DC/DC Converter," IEEE Journal of Solid-state Circuits, pp. 1799-1809, 1998.

[Gray93] P. Gray and R. Meyer, *Analysis and Design of Analog Integrated Circuits*, 3rd ed., Wiley, 1993.

[Hatamian88] M. Hatamian, “Understanding Clock Skew in Synchronous Systems,” in *Concurrent Computations*, ed. S. Tewksbury et al., Plenum Publishing, pp. 86–96, 1988.

[Heller84] L. Heller et al., “Cascade Voltage Switch Logic: A Differential CMOS Logic Family,” *IEEE International Solid State Conference Digest*, Feb. 1984, San Francisco, pp. 16–17.

[Jacobs90] G. Jacobs and R. Brodersen, “A Fully Asynchronous Digital Signal Processor,” *IEEE Journal on Solid State Circuits*, vol. 25, No 6, December 1990, pp. 1526–1537.

[Jeong87] D. Jeong et al., “Design of PLL-Based Clock Generation Circuits”, *IEEE Journal on Solid State Circuits*, vol. SC-22, no 2, April 1987, pp 255–261.

[Johnson93] H. Johnson and M. Graham, *High-Speed Digital Design—A Handbook of Black Magic*, Prentice-Hall, N.J, 1993.

[Kim90] B. Kim, D. Helman, and P. Gray, “A 30 MHz Hybrid Analog/Digital Clock Recovery Circuit in 2 µm CMOS,” *IEEE Journal on Solid State Circuits*, vol. SC-25, no. 6, December 1990, pp. 1385–1394.

[Martin89] A. Matrin et. al, "The First Asynchronous Microprocessor: Test Results," *Computer Architecture News*, vol. 17 No 4, June 1989, pp. 95–110.

[Mead80] C. Mead and L. Conway, *Introduction to VLSI Design*, Addison-Wesley, 1980.

[Messerschmitt90] D. Messerschmidtt, "Synchronization in Digital Systems," IEEE Journal on Selected Areas in Communications, vol. 8, No. 8, pp. 1404-1419, October 1990.

[Miller2000] D. Miller, "Rationale and Challenges for Optical Interconnects to Electronic Chips," Proceedings of the IEEE, pp. 728-749, June 2000.

[Nielsen94] L Nielsen, C. Niessen, J. Sparso, K. van Berkel, "Low-Power Operation Using Self-Timed Circuits and Adaptive Scaling of Supply Voltage," IEEE Transactions on VLSI systems, December 1994, pp 391-397.

[Restle98] P. Restle, K. Jenkins, A. Deutsch, P. Cook, "Measurement and Modeling of On-chip Transmission Line Effects in a 400MHz Microprocessor," IEEE Journal of Solid-state Circuits, pp. 662-665, April 1998.

[Seitz80] C. Seitz, "System Timing," in [Mead80], pp. 218–262, 1980.

[Seitz92] C. Seitz, "Mosaic C: An Experimental Fine-Grain Multicomputer," in *Future Tendencies in Computer Science, Control and Applied Mathematics*, Proceedings International Conference on the 25th Anniversary of INRIA, Springer-Verlag, Germany, pp. 69–85, 1992.

[Shoji88] M. Shoji, *CMOS Digital Circuit Technology*, Prentice-Hall, 1988.

[Sutherland89] I. Sutherland, "Micropipelines," *Communications of the ACM*, pp. 720–738, June 1989.

[Veendrick80] H. Veendrick, "The Behavior of Flip Flops Used as Synchronizers and Prediction of Their Failure Rates", *IEEE Journal of Solid State Circuits*, vol. SC-15, no 2, April 1980, pp. 169–176.

[Williams87] T. Williams et al., "A Self-Timed Chip for Division," in *Proc. of Advanced Research in VLSI 1987*, Stanford Conf., pp. 75–96, March 1987.

EXERCISES AND DESIGN PROBLEM

Please refer to the book web-page (http://bwrc.eecs.berkeley.edu/IcBook) for insightful and challenging exercises and design problems.

Chapter 10
PROBLEMS

1. [C, None, 9.2] For the circuit in Figure 0.1, assume a unit delay through the Register and Logic blocks (i.e., $t_R = t_L = 1$). Assume that the registers, which are positive edge-triggered, have a set-up time t_S of 1. The delay through the multiplexer t_M equals 2 t_R.

a. Determine the minimum clock period. Disregard clock skew.

b. Repeat part a, factoring in a nonzero clock skew: $\delta = t'_\theta - t_\theta = 1$.

c. Repeat part a, factoring in a non-zero clock skew: $\delta = t'_\theta - t_\theta = 4$.

d. Derive the maximum positive clock skew that can be tolerated before the circuit fails.

e. Derive the maximum negative clock skew that can be tolerated before the circuit fails.

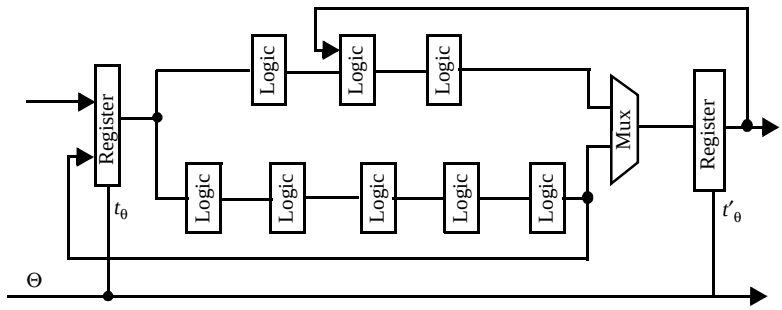


Figure 0.1 Sequential circuit.

2. This problem examines sources of skew and jitter.

a. A balanced clock distribution scheme is shown in Figure 0.2. For each source of variation, identify if it contributes to skew or jitter. Circle your answer in Table 0.1

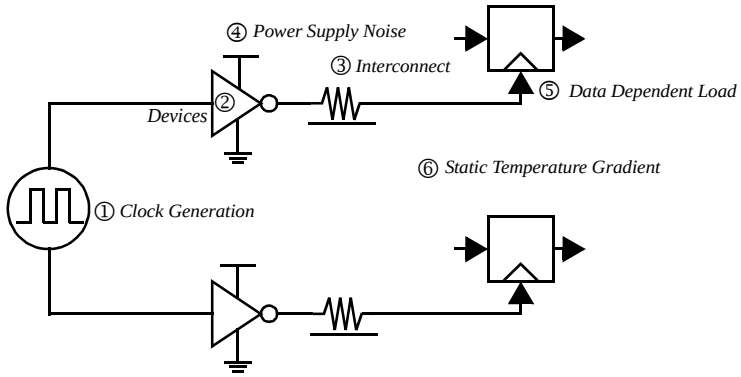


Figure 0.2 Sources of Skew and Jitter in Clock Distribution.

1) Uncertainty in the clock generation circuit	Skew	Jitter
2) Process variation in devices	Skew	Jitter
3) Interconnect variation	Skew	Jitter
4) Power Supply Noise	Skew	Jitter
5) Data Dependent Load Capacitance	Skew	Jitter
6) Static Temperature Gradient	Skew	Jitter

Table 0.1 Sources os Skew and Jitter

b. Consider a Gated Clock implementation where the clock to various logical modules can be individually turned off as shown in Figure 0.3. (i.e., $Enable_1$,..., $Enable_N$ can take on different values on a cycle by cycle basis). Which approach (*A* or *B*) results in lower *jitter* at the output of the input clock driver? (hint: consider gate capacitance) Explain.

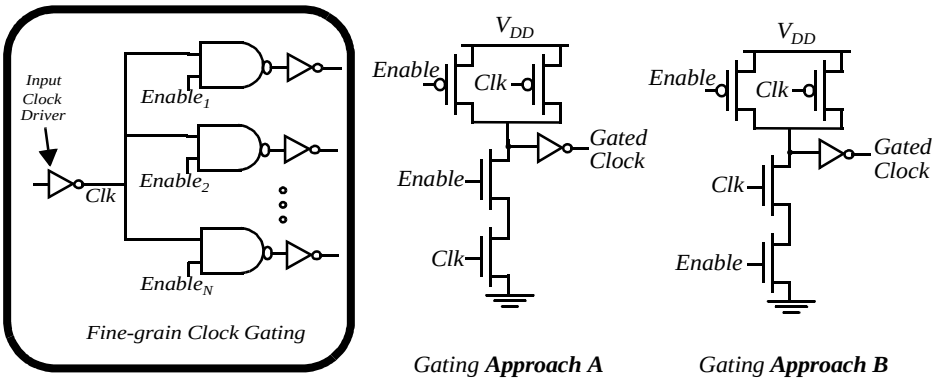


Figure 0.3 Jitter in clock gating

3. Figure 0.4 shows a latch based pipeline with two combinational logic units.

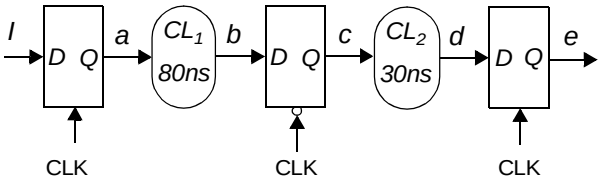


Figure 0.4 Latch Based Pipeline

Recall that the timing diagram of a combinational logic block and a latch can be drawn as follows, where the shaded region represents that the data is not ready yet.

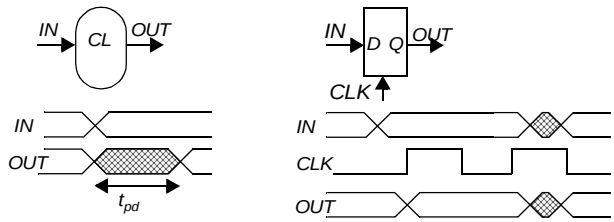


Figure 0.5 Timing diagrams of combinational logic and latch

Assume that the contamination delay t_{cd} of the combinational logic block is zero, and the $t_{clk\text{-}q}$ of the latch is zero too.

a. Assume the following timing for the input *I*. Draw the timing diagram for the signals *a*, *b*, *c*, *d* and *e*. Include the clock in your drawing.

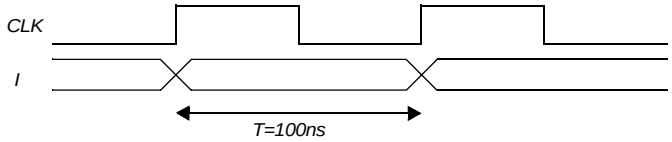


Figure 0.6 Input timing

b. State the deadline for the computation of the signal *b* and *d*, i.e. when is the latest time they can be computed, relative to the clock edges. In your diagram for (a), label with a "< >" the "slack time" that the signals *b* and *d* are ready before the latest time they must be ready.

c. Hence deduce how much the clock period can be reduced for this shortened pipeline. Draw the modified timing diagram for the signals *a*, *b*, *c*, *d*, and *e*. Include the clock in your drawing.

4. Consider the circuit shown in Figure 0.7.

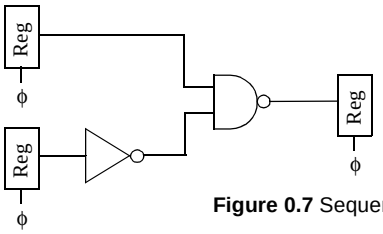


Figure 0.7 Sequential Circuit

a. Use SPICE to measure t_{max} and t_{min}. Use a minimum-size NAND gate and inverter. Assume no skew and a zero rise/fall time. For the registers, use the following:

- A TSPC Register.
- A C^2MOS Register.

b. Introduce clock skew, both positive and negative. How much skew can the circuit tolerate and still function correctly?

c. Introduce finite rise and fall time to the clocks. Show what can occur and describe why.

5. Consider the following latch based pipeline circuit shown in Figure 0.8.

Assume that the input, *IN*, is valid (i.e., set up) 2ns before the falling edge of *CLK* and is held till the falling edge of *CLK* (there is no guarantee on the value of *IN* at other times). Determine the maximum *positive* and *negative* skew on *CLK'* for correct functionality.

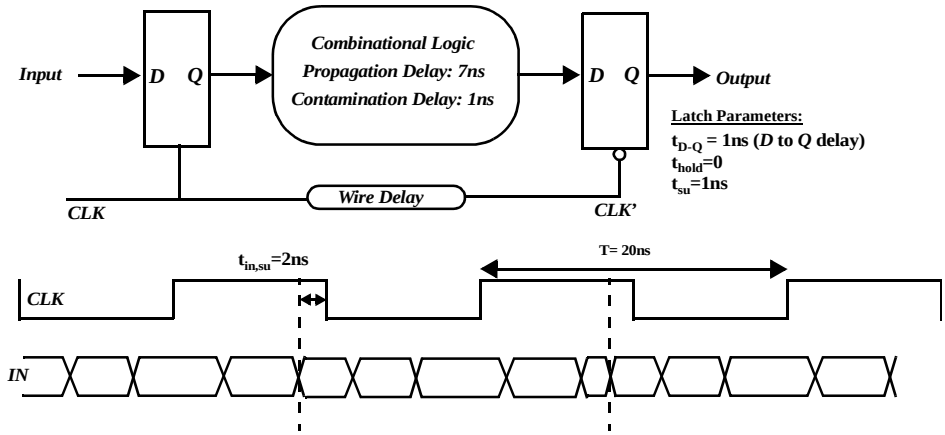


Figure 0.8 Latch based pipeline

6. For the L1-L2 latch based system from Figure 0.9**,** with two overlapping clocks derive all the necessary constraints for proper operation of the logic. The latches have setup times T_{SU1} and T_{SU2}, data-to-output delays $T_{D\text{-}Q1}$ and $T_{D\text{-}Q2}$, clock-to-output delays $T_{Clk\text{-}Q1}$ and $T_{Clk\text{-}Q2}$, and hold times T_{H1} and T_{H2}, respectively. Relevant clock parameters are also illustrated in Figure 0.9. The constraints should relate the logic delays, clock period, overlap time T_{OV}, pulse widths *PW*1 and *PW*2 to latch parameters and skews.

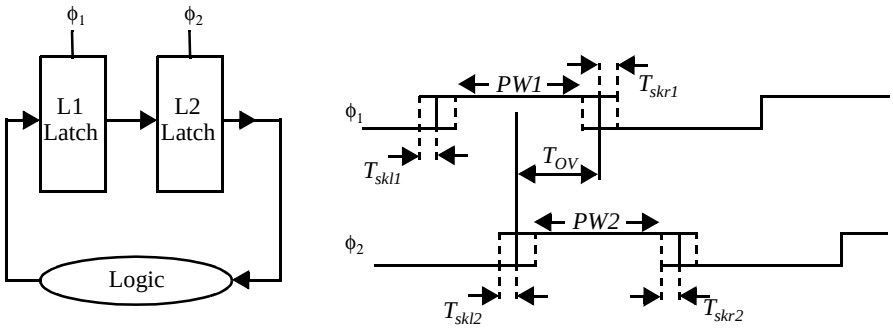


Figure 0.9 Timing constraints

7. For the self-timed circuit shown in Figure 0.10, make the following assumptions. The propagation through the NAND gate can be 5 nsec, 10 nsec, or 20 nsec with equal probability. The logic in the succeeding stages is such that the second stage is always ready for data from the first.

a. Calculate the average propagation delay with t_{hs} = 6 nsec.

b. Calculate the average propagation delay with t_{hs} =12 nsec.

c. If the handshaking circuitry is replaced by a synchronous clock, what is the smallest possible clock frequency?

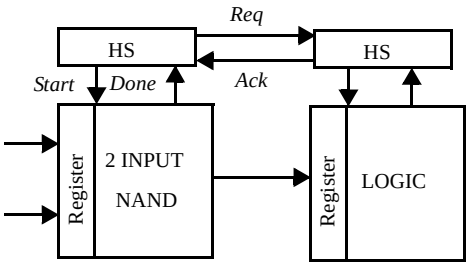


Figure 0.10 Self-timed circuit.

8. Lisa and Marcus Allen have a luxurious symphony hall date. After pulling out of their driveway, they pull up to a four-way stop sign. They pulled up to the sign at the same time as a car on the cross-street. The other car, being on the right, had the right-of-way and proceeded first. On the way they also have to stop at traffic signals. There is so much traffic on the freeway, the metering lights are on. Metering lights regulate the flow of merging traffic by allowing only one lane of traffic to proceed at a time. With all the traffic, they arrive late for the symphony and miss the beginning. The usher does not allow them to enter until after the first movement.

On this trip, Lisa and Marcus proceeded through both synchronizers and arbiters. Please list all and explain your answer.

9. Design a self-timed FIFO. It should be six stages deep and have a two phase handshakin with the outside world. The black-box view of the FIFO is given in Figure 0.11.

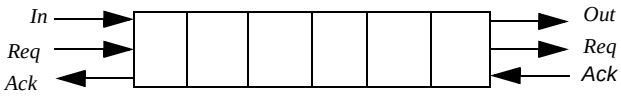


Figure 0.11 Overall structure of FIFO.

10. System Design issues in self-timed logic

One of the benefits of using self-timed logic is that it delivers average-case of performance rather than the worst-case performance that must be assumed when designing synchronous circuits. In some applications where the average and worst cases differ significantly you can have significant improvements in terms of performance. Here we consider the case of ripple carry addition. In a synchronous design the ripple carry adder is assumed to have a worst case performance which means a carry-propagation chain of length N for an N-bit adder. However, as we will prove during the course of this problem the average length of the carry-propagation chain assuming uniformly distributed input values is in fact O(log N)!

a. Given that $p_n(v)$ = Pr(carry-chain of an n-bit addition is ≥ v bits), what is the probability that the carry chain is of length k for an n-bit addition?

b. Given your answer to part (a), what is the average length of the carry chain (i.e., a_n)? Simplify your answer as much as possible.

Now $p_n(v)$ can be decomposed into two mutually-exclusive events, A and B. Where A represents that a carry chain of length ≥ v occurs in the first n-1 bits, and B represents that a carry chain of length v ends on the nth bit.

c. Derive an expression for Pr(A).

d. Derive an expression for Pr(B). (HINT: a carry bit i is propagated only if $a_i \neq b_i$, and a carry chain begins only if $a_i = b_i = 1$).

e. Combine your results from (c) and (d) to derive an expression for $p_n(v) - p_{n-1}(v)$ and then bound this result from above to yield an expression in terms of only the length of the carry chain (i.e., v).

f. Using what you've shown thus far, derive an upper bound for the expression:

$$\sum_{i=v}^{n} (p_i(v) - p_{i-1}(v))$$

Use this result, coupled with the fact that $p_n(v)$ is a probability (i.e., it's bounded from above by 1), to determine a two-part upper bound for $p_n(v)$.

g. (The magic step!) Bound n by a clever choice of k such that $2^k \leq n \leq 2^{k+1}$ and exploit the fact that $log_2\, x$ is concave down on $(0, \infty)$ to ultimately derive that $a_n \leq log_2\, n$, which concludes your proof!

h. Theoretically speaking, how much faster would a self-timed 64-bit ripple carry adder be than its synchronous counterpart? (You may assume that the overhead costs of using self-timed logic are negligible).

11. Figure 0.12 shows a simple synchronizer. Assume that the asynchronous input switches at a rate of approximately 10 MHz and that t_r = 2 nsec, f_ϕ = 50 MHz, $V_{IH} - V_{IL}$=0.5 V, and V_{DD} = 2.5 V.

a. If all NMOS devices are minimum-size, find $(W/L)p$ required to achieve V_{MS} = 1.25 V. Verify with SPICE.

b. Use SPICE to find τ for the resulting circuit.

c. What waiting time T is required to achieve a MTF of 10 years?

d. Is it possible to achieve an MTF of 1000 years (where $T > T_\phi$)? If so, how?

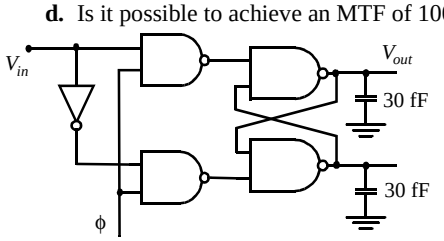


Figure 0.12 Simple synchronizer

12. Explain how the phase-frequency comparator shown in Figure 0.13 works.

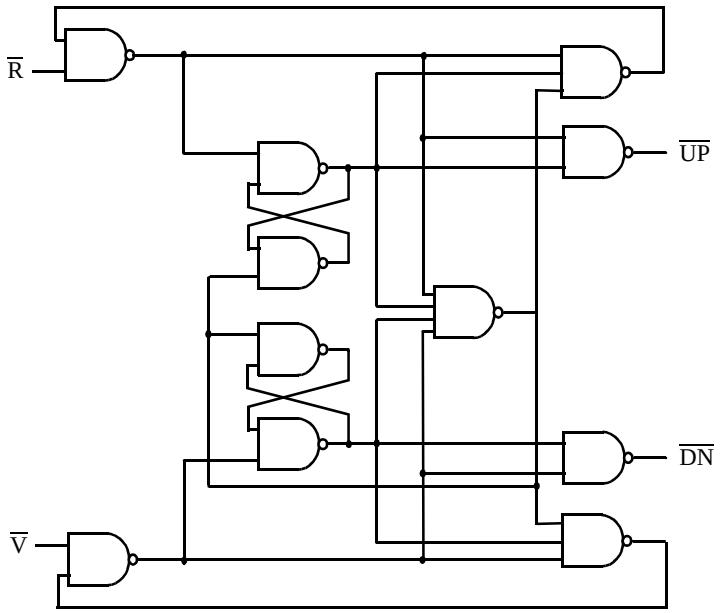


Figure 0.13 Phase-frequency comparator

13. The heart of any static latch is the cross-coupled structure shown in Figure 0.14 (part a).

a. Assuming identical inverters with *Wp/Wn =kn'/kp'*, what is the metastable point of this circuit? Give an expression for the time trajectory of V_Q, assuming a small initial *Vd0* centered around the metastable point of the circuit, V_M.

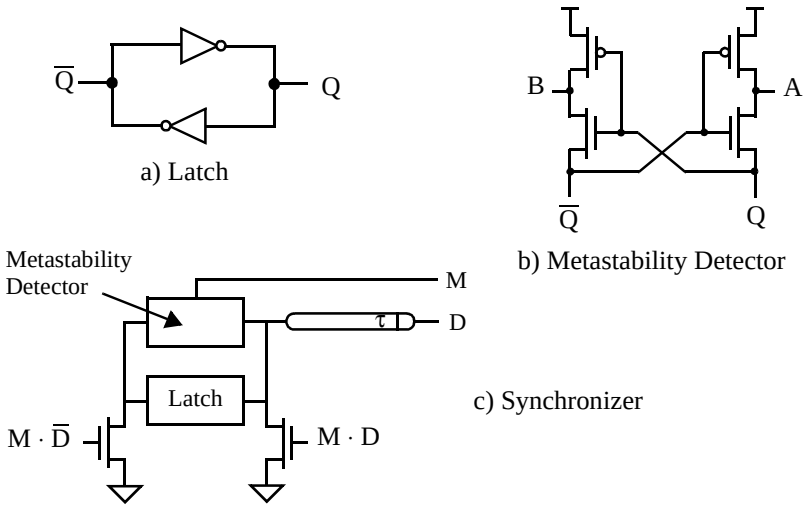


Figure 0.14 Simple synchronizer

b. The circuit in part b has been proposed to detect metastability. How does it work? How would you generate a signal M that is high when the latch is metastable?

c. Consider the circuit of part c. This circuit was designed in an attempt to defeat metastability in a synchronizer. Explain how the circuit works? What ís the function of the delay element?

14. An adjustable duty-cycle clock generator is shown in Figure 0.15. Assume the delay through the delay element matches the delay of the multiplexer.

a. Describe the operation of this circuit

b. What is the range of duty-cycles that can be achieved with this circuit.

c. Using an inverter and an additional multiplexer, show how to make this circuit cover the full range of duty cycles.

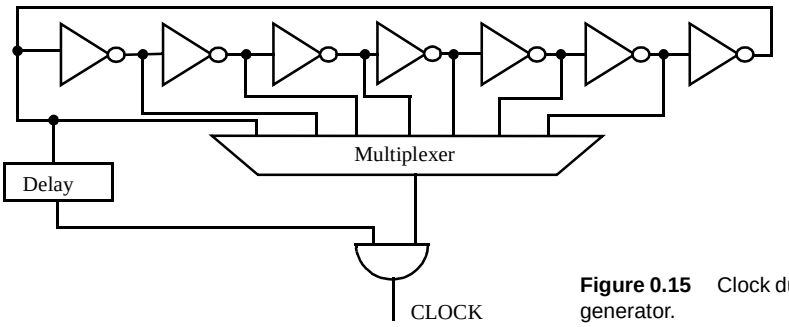


Figure 0.15 Clock duty-cycle generator.

15. The circuit style shown in Figure 0.17.a has been proposed by Acosta et. al. as a new self-timed logic style. This structure is known as a Switched Output Differential Structure[1].

a. Describe the operation of the SODS gate in terms of its behavior during the pre-charge phase, and how a valid completion signal can be generated from its outputs.

b. What are the advantages of using this logic style in comparison to the DCVSL logic style given in the notes?

c. What are the disadvantages of using this style in comparison to DCVSL?

d. Figure 0.16.b shows a 2-input AND gate implemented using a SODS style. Simulate the given circuit using Hspice. Do you notice any problems? Explain the cause of any problems that you may observe and propose a fix. Re-simulate your corrected circuit and verify that you have in fact fixed the problem(s).

[1] A.J. Acosta, M. Valencia, M.J. Bellido, J.L. Huertas, "SODS: A New CMOS Differential-type Structure," *IEEE Journal of Solid State Circuits*, vol. 30, no. 7, July 1995, pp. 835-838

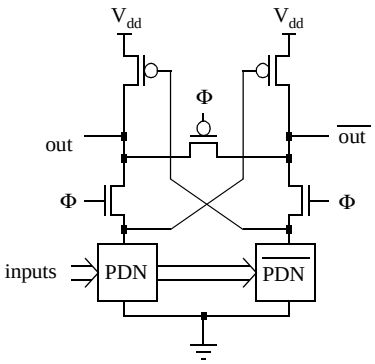


Figure 0.17 a - SODS Logic Style

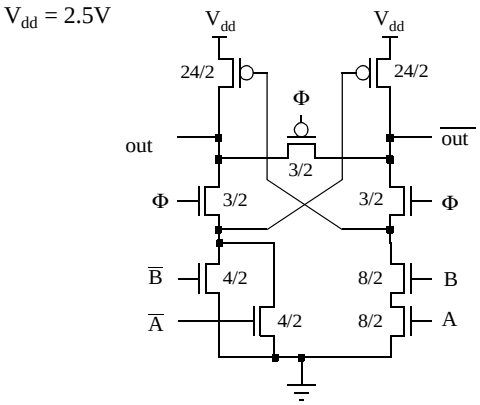


Figure 0.16 b - 2-input And Gate in SODS Style

16. Voltage Control Ring Oscillator.

In this problem, we will explore a voltage controlled-oscillator that is based upon John G. Maneatis' paper in Nov. 1996, entitled "Low Jitter Process-Independent DLL and PLL Based on Self-Biased Techniques," appeared in the Journal of Solid-State Circuits. We will focus on a critical component of the PLL design: the voltage-controlled ring oscillator. Figure 0.17 shows the block diagram of a voltage controlled ring oscillator:

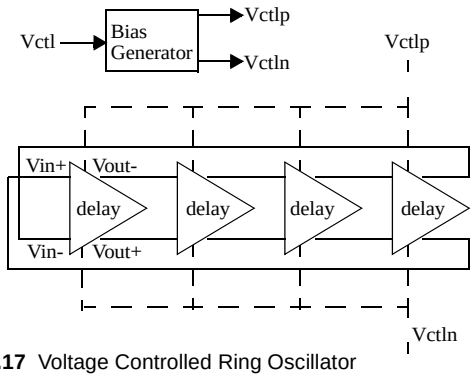


Figure 0.17 Voltage Controlled Ring Oscillator

The control voltage, Vctl, is sent to a bias generator that generates two voltages used to properly bias each delay cell equally, so that equal delay (assuming no process variations) appear across each delay cell. The delay cells are simple, "low-gain" fully differential input and output operational amplifiers that are connected in such a way that oscillations will occur at any one of the outputs with a frequency of 1/(4*delay). Each delay is modeled as an RC time constant; C comes from parasitic capacitances at the output nodes of the delay element,

and R comes from the variable resistor that is the load for the delay cell. Below is a circuit schematic of a typical delay cell.

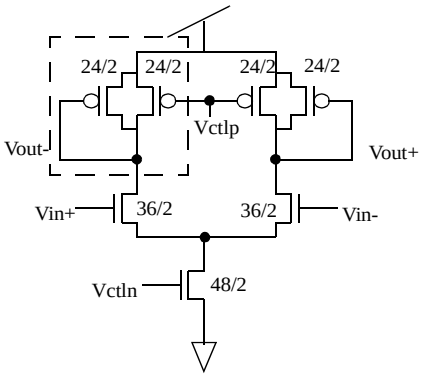


Figure 0.18 One delay Cell

As mentioned before, the value of R is set by a variable resistor. How can one make a variable resistor? The object in the delay cell that is surrounded by a dotted line is called a "symmetric load," and provides the answer to a voltage-controlled variable resistor. R should be linear so that the differential structure cancels power supply noise. We will begin our analysis with the symmetric load.

a. In Hspice, input the circuit below and plot Vres on the X axis and Ires on the Y axis, for the following values of Vctlp: 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, and 2.0 volts, by varying Vtest from Vctlp to Vdd, all on the same graph. For each curve, plot Vres from 0 volts to Vdd-Vctlp. When specifying the Hspice file, be sure to estimate area and perimeter of drains/sources.

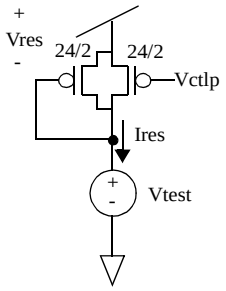


Figure 0.19 :Symmetric Load Test Circuit

After you have plotted the data and printed it out, use a straight edge to connect the end points for each curve. What do you notice about intersection points between the line you drew over each curve, and the curves themselves? Describe any symmetries you see.

b. For each Vctlp curve that you obtained in a), extract the points of symmetries (Vres, Ires), and find the slope of the line around these points of symmetry. These are the effective resistances of the resistors. Also, for each Vctl curve, state the maximum amplitude the output swing can be, without running into asymmetries. Put all of this data in an worksheet format.

c. Using the estimations you made for area and perimeter of drain ad source that you put in your Hspice file, calculate the effective capacitance. (Just multiply area and perimeter by CJ and CJSW from the spice deck). Since we are placing these delay elements in a cas-

caded fashion, remember to INCLUDE THE GATE CAPACITANCE of the following stage. Each delay element is identical to one another. Now, calculate the delay in each cell, according to each setting of Vctlp that you found in a): delay=0.69*R*C. Then, write a general equation, in terms of R and C, for the frequency value that will appear at each delay output. Why is it necessary to cross the feedback lines for the ring oscillator in the first figure? Finally, draw a timing/transient analysis of each output node of the delay lines. How many phases of the base frequency are there?

d. Now, we will look at the bias generator. The circuit for the bias generator is as follows:

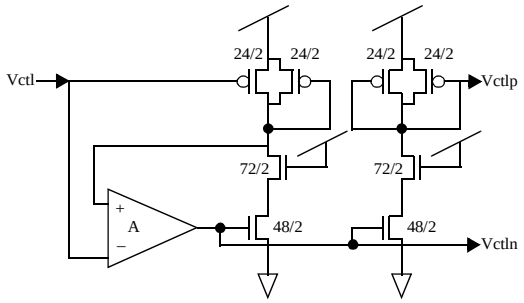


Figure 0.20 :Bias Generator

Implement this circuit in Hspice, and use the ideal voltage controlled voltage source for your amplifier. Use a value of 20 for A. This circuit automatically sets the Vctln and Vctlp voltages to the buffer delays to set the DC operating points of the delay cells such that the symmetric load is swinging reflected around its point of symmetry for a given Vctl voltage. Also, it is important to note that Vctl is the same as Vctlp. It must go through this business to obtain Vctln (which sets the bias current to the correct value, which sets the DC operating point of the buffer). Do a transient run in Hspice to verify that Vctlp is indeed very close to Vctl over a range of inputs for Vctl. Show a Spice transient simulation that goes for 1uS, and switches Vctl in a pwl waveform across a range of inputs between 0.5V and 2.0V. For extra points, explain how this circuit works.

e. Now, hook up the bias generator you just built with 4 delay cells, as shown in the first figure. For each control voltageVctlp from part c), verify your hand calculations with spice simulations. Show a spreadsheet of obtained frequencies vs. hand-calculation predictions, and in a separate column, calculate % error. Give a brief analysis of what you see. Print out all of the phases (4) of the clock, for a Vctl value of your choice.

Tips for using Hspice

I. Getting Started

- Setup the environment:
  source /usr/eesww/HSPICE/98.2/bin/cshrc.meta
- Run the simulator on your input file:
  hspice filename.sp >! filename.lis
- Use the waveform viewer to see the output
  awaves
  Input files must have the extension *.sp* for the waveform viewer to work.
  Also, the input file must have ".OPTION POST=2" specified.
  Waveforms can be printed by choosing Tools -> Print...
- View the online documentation
  acroread /usr/eesww/HSPICE/98.2/docs/hspice.pdf &
  The page numbers in the rest of this document refer to this manual.
  This file is 11 MB (1714 pages). Do not print it out!

II. Netlist Format

- The input files are case insensitive.
- The first line is always a comment. Other lines are commented with a leading * or $
- All nonlinear devices must have a .MODEL statement.

Names:
- can contain letters, numbers, and the characters ! # $ % * + - / < > [ ] _ (see p. 3-18)
- can be 1024 characters long
- Node names can begin with letters, numbers, or the characters # _ ! % (see p. 3-19)
- Trailing alphabetic characters are ignored in net names. For example a node named 1A is considered to be equivalent to node 1 (see p. 3-19)
- Nodes named 0, GND, GND!, and GROUND all refer to the global ground node.

III. Values

Expressing Values:
- Scientific notation: *e.g.* 1.1e-17
- Use a suffix: *e.g.* 2.3u (x=mega, k=kilo, m=mili, u=micro, n=nano, p=pico, f=femto)
- Use a parameter: *e.g.* minlen (parameters must be declared with a .PARAM statement)
- Evaluated expressions: *e.g.* '500m*minlen'

Output variables:

- Voltage between two nodes: v(n1,n2)
- Voltage of a node relative to ground: v(n1)
- Current through an independent source: i(vin)

IV. Analysis

- .OP Operating point, DC circuit solution (see p. 6-8)
- .DC Sweep of DC operating points (capacitances are ignored) (see p. 8-1)
  .DC var startval stopval incr - performs a DC sweep on the independent source or parameter **var**, varying its value from **startval** to **stopval** using the increment **incr**.
- .TRAN Perform a transient analysis (differential equation solver) (see p. 7-1)
  .TRAN tincr tstop – finds the operating point (.OP) and then performs a transient analysis of duration **tstop** seconds with a maximum time step of **tincr**.

V. Control

- .INCLUDE – includes a file (see p. 3-60)
- .OPTION – sets simulation options (see p. 3-45)
- .END – marks the end of an input file
- .ALTER – treated as .END, but allows another simulation to be performed with the changes that follow the .ALTER statement (see p. 3-34)

VI. Measurements

- **.MEASURE TRAN t_delay TRIG v(in) VAL=2.5 CROSS=1 TARG v(out) VAL=2.5 CROSS=1**
  Measures the propagation delay between the nodes in and out, where the signals first cross 2.5 volts.
- **.MEAS t_rise TRIG v(out) VAL=0.5 RISE=1 TARG v(out) VAL=4.5 RISE=1**
  Measures the first 10%-90% rise time of a 5V signal
- **.MEAS TRAN max_current MAX I(Vdd)**
  Measures the maximum current through the independent source Vdd
- **.MEAS peak_power PARAM=’max_current*5’**
  Calculates the peak power, assuming that max_current has been measured

Measured values are placed in a file called ***filename*.mt#.** See p. 4-19 for more details.

VII. Troubleshooting

- **Failure to converge** (OP and DC) – The DC solver uses an iterative method to find the operating point, but some circuits exist which have no or multiple operating points. The best solution is to perform hand analysis of your circuit to make sure that you haven’t done this.

- **No DC Path to ground** (OP and DC) – This often happens with floating MOSFET gates. Just add a resistor between the node and ground, or use the .IC or .NODESET commands to create an initial condition. Be warned, however, that .NODESET and .IC can cause convergence problems.
- **Stability Problems** (TRAN) – Sometimes a transient analysis shows a "ringing" or oscillation that shouldn't be there. This can often be solved by reducing the maximum time step or using slower rise and fall times for independent sources.

Send questions to **wrdavis@eecs.berkeley.edu** ...

DESIGN METHODOLOGY INSERT

A

IC LAYOUT

Creating a manufacturable layout

n

Verifying the layout

The increasing complexity of the integrated circuit has made the role of design-automation tools indispensable, and raises the abstractions the designer is working with to ever higher levels. Yet, when performance or design density is of primary importance, the designer has no other choice than to return to handcrafting the circuit topology and physical design. The labor-intensive nature of this approach, called *custom design*, translates into a high cost and a long *time-to-market*. Therefore, it can only be justified economically under the following conditions:

- The custom block can be reused many times, for instance as a library cell
- The cost can be amortized over a large volume. Microprocessors and semiconductor memories are examples of applications in this class.
- Cost is not the prime design criterion. This is becoming increasingly rare. Examples are space-applications and scientific instrumentation.

With continuous progress in the design-automation arena, the share of custom design reduces from year to year. Even in high-performance microprocessors, large portions are designed automatically using semicustom design approaches. Only the most performance-critical modules such as the integer and floating-point execution units are handcrafted.

Even though the amount of design automation in the custom design process is minimal, some design tools have proven to be indispensable. Together with circuit simulators, these programs form the core of every design-automation environment, and are the first tools an aspirant circuit designer will encounter.

Layout Editor

The layout editor is the premier working tool of the designer and exists primarily for the generation of a physical representation of a design, given a circuit topology. Virtually every design-automation vendor offers an entry in this field. Most well-known is the MAGIC tool developed at the University of California at Berkeley [Ousterhout84], which has been widely distributed. Even though MAGIC did not withstand the evolution of software technology and user interface, some of its offspring did. Throughout this textbook, we will be using a layout tool called **max**, a MAGIC descendant developed by a company called MicroMagic [mmi00]. A typical **max** display is shown in Figure A.1 and illustrates the basic function of the layout editor—placing polygons on different mask layers so that a functional physical design is obtained (scathingly called *polygon pushing*).

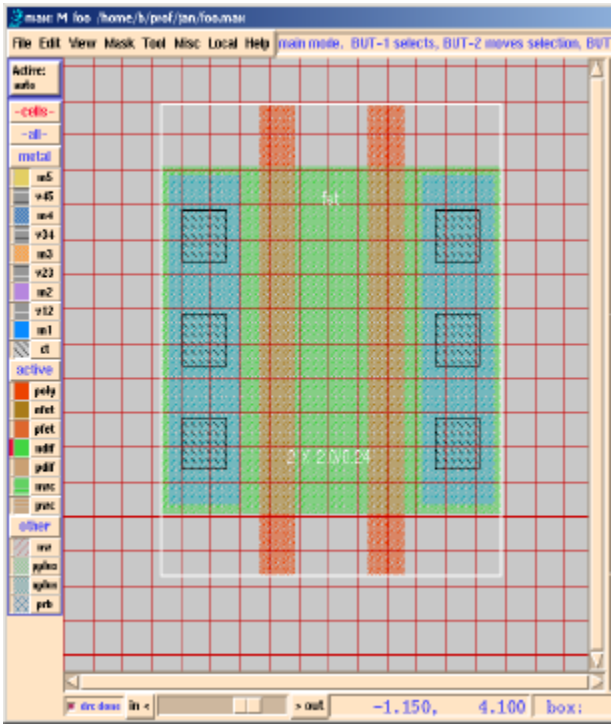


Figure A.1 View of a **max** display window. It plots the layout of two stacked NMOS transistor. The menu on the left side allows for the selection of the layer a particular polygon will be placed on.

Since physical design occupies a major fraction of the design time for a new cell or component, techniques to expedite this process have been in continual demand. The *symbolic-layout* approach has gained popularity over the years. In this design methodology, the designer only draws a shorthand notation for the layout structure. This notation indicates only the *relative* positioning of the various design components (transistors, contacts,

wires). The *absolute* coordinates of these elements are determined automatically by the editor using a *compactor* [Hsueh79, Weste93]. The compactor translates the design rules into a set of constraints on the component positions, and solves a constrained optimization problem that attempts to minimize the area or another cost function.

An example of a symbolic notation for a circuit topology, called a *sticks diagram*, is shown in Figure A.2. The different layout entities are dimensionless, since only position-

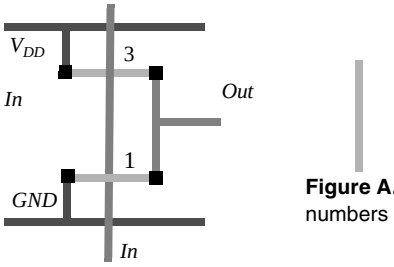


Figure A.2 Sticks representation of CMOS inverter. The numbers represent the (*Width*/*Length*)-ratios of the transistors.

ing is important. The advantage of this approach is that the designer does not have to worry about design rules, because the compactor ensures that the final layout is physically correct. Thus, she can avoid cumbersome polygon manipulations. Another plus of the symbolic approach is that cells can adjust themselves automatically to the environment. For example, automatic pitch-matching of cells is an attractive feature in module generators. Consider the case of Figure A.3 (from [Croes88]), in which the original cells have

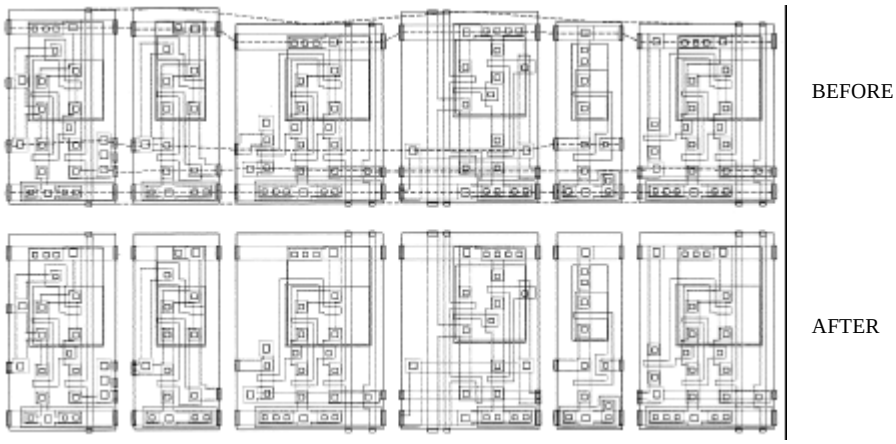


Figure A.3 Automatic pitch matching of data path cells based on symbolic layout.

different heights, and the terminal positions do not match. Connecting the cells would require extra wiring. The symbolic approach allows the cells to adjust themselves and connect without any overhead.

The disadvantage of the symbolic approach is that the outcome of the compaction phase is often unpredictable. The resulting layout can be less dense than what is obtained with the manual approach. Notwithstanding, symbolic layout tools have improved considerably over the years and are currently a part of the mainstream design process.

Design-Rule Checking

Design rules were introduced in Chapter 2 as a set of layout restrictions that ensure the manufactured design will operate as desired with no short or open circuits. A prime requirement of the physical layout of a design is that it adhere to these rules. This can be verified with the aid of a *design-rule checker (DRC)*, which uses as inputs the physical layout of a design and a description of the design rules presented in the form of a *technology file*. Since a complex circuit can contain millions of polygons that must be checked against each other, efficiency is the most important property of a good DRC tool. The verification of a large chip can take hours or days of computation time. One way of expediting the process is to preserve the design hierarchy at the physical level. For instance, if a cell is used multiple times in a design, it should be checked only once. Besides speeding up the process, the use of hierarchy can make error messages more informative by retaining knowledge of the circuit structure.

DRC tools come in two formats: (1) The *on-line DRC* runs concurrent with the layout editor and flags design violations during the cell layout. For instance, **max** has a built-in design-rule checking facility. An example of on-line DRC is shown in Figure A.4. (2) *Batch DRC* is used as a post-design verifier, and is run on a complete chip prior to shipping the mask descriptions to the manufacturer.

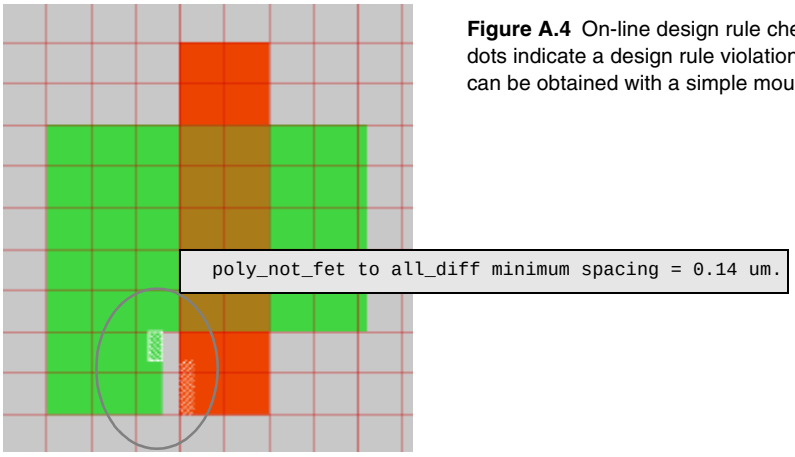


Figure A.4 On-line design rule checking. The white dots indicate a design rule violation. The violated rule can be obtained with a simple mouse click.

Circuit Extraction

Another important tool in the custom-design methodology is the circuit extractor, which derives a circuit schematic from a physical layout. By scanning the various layers and their interactions, the extractor reconstructs the transistor network, including the sizes of the devices and the interconnections. The schematic produced can be used to verify that the artwork implements the intended function. Furthermore, the resulting circuit diagram contains precise information on the parasitics, such as the diffusion and wiring capacitances and resistances. This allows for a more accurate simulation and analysis. The complexity of the extraction depends greatly upon the desired information. Most extractors extract the transistor network and the capacitances of the interconnect with respect to *GND* or other network nodes. Extraction of the wiring resistances already comes at a

greater cost, yet has become a necessity for virtually all high-performance circuits. Clever algorithms have helped to reduce the complexity of the resulting circuit diagrams. For very high speed circuits, extraction of the inductance would be desirable as well. Unfortunately, this requires a three-dimensional analysis and is only feasible for small-sized circuits at present.

To Probe Further

More detailed information regarding the MAGIC and **max** layout editors can be found on the web-site of this book. In-depth textbooks on layout generation and verification have been published, and can be of great help to the novice designer. Just to mention a number of them, [Clein00], [Uyemura95], [Wolf94], and [Weste93] offer some comprehensive and well-illustrated treatment and discussion.

REFERENCES

[Clein00] D. Clein, *CMOS IC Layout—Concepts, Methodologies, and Tools*, Newnes, 2000.

[Croes88] K. Croes, H. De Man, and P. Six, “CAMELEON: A Process-Tolerant Symbolic Layout System,” *Journal of Solid State Circuits*, vol. 23 no. 3, pp. 705–713, June 1988.

[Hsueh79] M. Hsueh and D. Pederson, “Computer-Aided Layout of LSI Building Blocks,” *Proceedings ISCAS Conf.*, pp. 474–477, Tokyo, 1979.

[mmi00] MicroMagic, Inc, *http://www.micromagic.com.*

[Ousterhout84] J. Ousterhout, G. Hamachi, R. Mayo, W. Scott, and G. Taylor, “Magic: A VLSI Layout System,” *Proc. 21st Design Automation Conference*, pp. 152–159, 1984.

[Uyemura95] J. Uyemura, *Physical Design of CMOS Integrated Circuits Using L-EDIT*, PWS Publishing Company, 1995.

[Weste93] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design—A Systems Perspective*, Addison-Wesley, 1993.

[Wolf94] W. Wolf, *Modern VLSI Design—A Systems Approach*, Prentice Hall, 1994.

DESIGN METHODOLOGY INSERT

E

CHARACTERIZING LOGIC AND SEQUENTIAL CELLS

The challenge of library characterization

n

Characterization methods for logic cells and registers

n

Cell parameters

The Importance and Challenge of Library Characterization

The quality of the results produced by a logic synthesis tool is a strong function of the level of detail and accuracy with which the individual cells were characterized. To estimate the delay of a complex module, a logic synthesis program must rely on higher-level delay models of the individual cells—falling back to a full circuit- or switch level timing model for each delay estimation is simply not possible from a perspective of compute time. Hence, an important component in the development process of a standard-cell library is the generation of the delay models. In previous chapters, we learned that the delay of a complex gate is a function of the fanout (consisting of connected gates and wires), and the rise-and fall times of the input signals. Furthermore, the delay of a call can vary between manufacturing runs as a result of process variations.

In this insert, we discuss first the models and characterization methods that are commonly used for logic cells. Sequential registers require some extra timing parameters, and deserve a separate discussion.

Characterization of Logic Cells

Unfortunately, no common delay model for standard cells has been adopted. Every vendor has his own favored methods of cell characterization. Even within a single tool, various delay models can often be used, trading off acuracy for performance. The basic concepts are however quite similar, and are closely related to the ones we introduced in Chapters 5 and 6. We therefore opt to concentrate on a single set of models in this section; more pre cisley, those used in the Synopsys Design Compiler [DesignCompiler01], one of the most popular synthesis tools. Once a model has been adopted, it has to be adopted for all the cells in the block; in other words, it can't be changed from cell to cell.

The total delay consists of four components, as illustrated in Figure E.1:

$$D_{total} = D_I + D_T + D_S + D_C. \tag{E.1}$$

D_I represents the intrinsic delay, this is the delay with no output loading. D_T is the 'transition' component, or the part of the delay caused by the output load. D_S is the fraction of the delay due to the input slope. Finally, D_C is the delay of the wire following the gate. All delays are characterized for both rising and falling transitions.

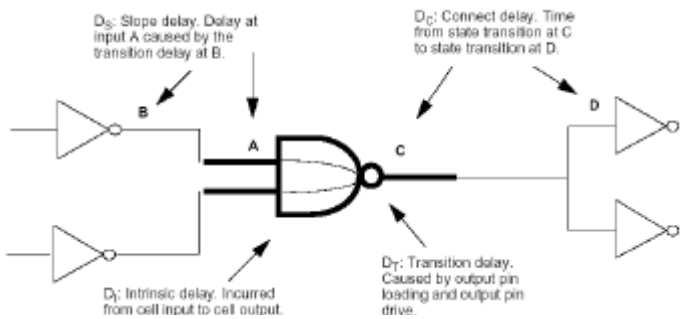


Figure E.1 Delay components.

The simplest model for the transition delay is the linear delay model of Chapter 5,

$$D_T = R_{driver}(\Sigma C_{gate} + C_{wire}), \tag{5.2}$$

where ΣC_{gate} is the sum of all input pin capacitances of gates connected to the output of this gate, and C_{wire} is the estimated wire capacitance. Similarly, the slope delay D_S can be approximated as a linear function of the transition delay D_T of the previous gate,

$$D_S = S_S\, D_{Tprev} \tag{5.3}$$

with S_S is the *slope-sensitivity factor*, and D_{Tprev} the transition delay of the previous stage.

The characterization of a library cell therefore must provide the following components, each of them for both rising and falling transitions, and with respect to each of the input pins:

- intrinsic delay;
- input pin capacitance;
- equivalent output driving resistance;

- and slope sensitivity.

In addition to the cell models, the synthesis tools also must have access to a wire model. Since the length of the wires is unknown before the placement of the cells, estimates of C_{wire} and R_{wire} are made based on the size of the block, and the fanout of the gate. The length of a wire is most often proportional the number of destinations it has to connect.

Example E.1 3-Input NAND Gate Cell

To characterization of the 3-input NAND standard cell gate, presented earlier in Example 8.1, is given in Table E.1. The table characterizes the performance of the cell as a function of the load capacitance and the input-rise(fall) time for two different supply voltages and operating temperatures. The cell is designed in a 0.18 µm CMOS technology.

Table E.1 Delay characterization of 3-input NAND gate (in nsec) as a function of the input node for two operation corners (supply voltage-temperature of 1.2V - 125°C, and 1.6V - 40°C) . The parameters are the load capacitance *C* and the input rise(fall) time *T*.

Path	1.2V - 125°C	1.6V - 40°C
$In1—t_{pLH}$	$0.073+7.98C+0.317T$	$0.020+2.73C+0.253T$
$In1—t_{pHL}$	$0.069+8.43C+0.364T$	$0.018+2.14C+0.292T$
$In2—t_{pLH}$	$0.101+7.97C+0.318T$	$0.026+2.38C+0.255T$
$In2—t_{pHL}$	$0.097+8.42C+0.325T$	$0.023+2.14C+0.269T$
$In3—t_{pLH}$	$0.120+8.00C+0.318T$	$0.031+2.37C+0.258T$
$In3—t_{pHL}$	$0.110+8.41C+0.280T$	$0.027+2.15C+0.223T$

While linear delay models offer good first-order estimates, more precise models are often used in synthesis, especially when the real wire lengths are back-annotated onto the design. Under those circumstances, non-linear models have to be adopted. The most common approach is to capture the non-linear relations as lookup tables for each of these parameters. To increase computational efficiency and minimize storage and characterization requirements, only a limited set of loads and slopes are captured, and linear interpolation is used to determine the values in-between.

Example E.2 Delay models using lookup-tables

A (partial) characterization of a 2-input AND cell (AND2), designed in a 0.25 µm CMOS technology [STMicro01], is given below. The delays are captured for output capacitances of 7fF, 35fF, 70fF and 140fF, and input slopes of 40ps, 200ps, 800ps and 1.6ns, respectively.

```
cell(AND) {
  area : 36 ;
  pin(Z) {
    direction : output ;
    function : "A*B";
    max_capacitance : 0.14000 ;
```

```
 timing() {
   related_pin : "A" ; /* delay between input pin A and output pin Z */
   cell_rise {
   values( "0.10810, 0.17304, 0.24763, 0.39554", \
           "0.14881, 0.21326, 0.28778, 0.43607", \
           "0.25149, 0.31643, 0.39060, 0.53805", \
           "0.35255, 0.42044, 0.49596, 0.64469" ); }
  rise_transition {
   values( "0.08068, 0.23844, 0.43925, 0.84497", \
           "0.08447, 0.24008, 0.43926, 0.84814", \
          "0.10291, 0.25230, 0.44753, 0.85182", \
           "0.12614, 0.27258, 0.46551, 0.86338" );}
  cell_fall(table_1) {
   values( "0.11655, 0.18476, 0.26212, 0.41496", \
           "0.15270, 0.22015, 0.29735, 0.45039", \
           "0.25893, 0.32845, 0.40535, 0.55701", \
           "0.36788, 0.44198, 0.52075, 0.67283" );}
  fall_transition(table_1) {
   values( "0.06850, 0.18148, 0.32692, 0.62442", \
           "0.07183, 0.18247, 0.32693, 0.62443", \
           "0.09608, 0.19935, 0.33744, 0.62677", \
           "0.12424, 0.22408, 0.35705, 0.63818" );}
  intrinsic_rise : 0.13305 ; /* unloaded delays */
  intrinsic_fall : 0.13536 ;
 }
timing() {
   related_pin : "B" ; /* delay between input pin A and output pin Z */
  ...
  intrinsic_rise : 0.12426 ;
  intrinsic_fall : 0.14802 ;
 }
}
pin(A) {
 direction : input ;
 capacitance : 0.00485 ; /* gate capacitance */
}
pin(B) {
 direction : input ;
 capacitance : 0.00519 ;
}
}
```

Characterization of Registers

In Chapter 7, we identified the three important timing parameters of a register. The *set-up time* (t_{su}) is the time that the data inputs (*D* input) must be valid before the clock transition (this is, the 0 to 1 transition for a *positive edge-triggered* register). The *hold time* (t_{hold}) is the time the data input must remain valid after the clock edge. Finally, the propagation delay ($t_{c\text{-}q}$) equals the time it takes for the data to be copied to the *Q* output after a clock event. The latter parameter is illustrated in Figure E.2a.

Latches are a bit more complex, and require an extra timing parameter. While $t_{C\text{-}Q}$, corresponds to the delay of re-launching of data that arrived to a closed latch, $t_{D\text{-}Q}$ equals the delay between *D* and *Q* terminals when the latch is in transparent mode (Figure E.2b).

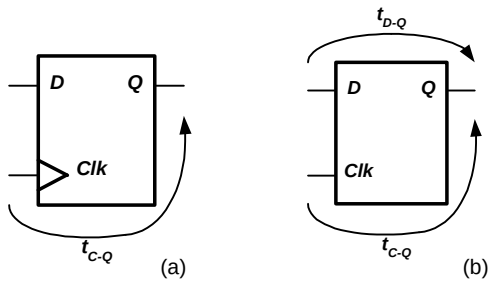


Figure E.2 Propagation delay definitions for sequential components:
(a) register;
(b) latch.

The characterization of the $t_{C\text{-}Q}$ ($t_{D\text{-}Q}$) delay is fairly straightforward. It consists of a delay measurement between the 50% transitions of *Clk* (*D*) and *Q*, for different values of the input slopes and the output loads.

The characterization of setup and hold times is more elaborate, and depends upon what is defined as "valid" in the definitions of both setup and hold times. Consider the case of the set-up time. Narrowing the time interval between the arrival of the data at the D input and the Clk event does not lead into instantaneous failure, but rather in a gradual degradation in the delay of the register. This is documented in Figure E.3a, which illustrates the behavior of a register when the data arrives close to the setup time. If *D* changes long before the clock edge, the $t_{C\text{-}Q}$ delay has a constant value. Moving the data transition closer to the clock edge causes $t_{C\text{-}Q}$ to increase. Finally, if the data changes too close to the clock edge, the register fails to register the transition altogether.

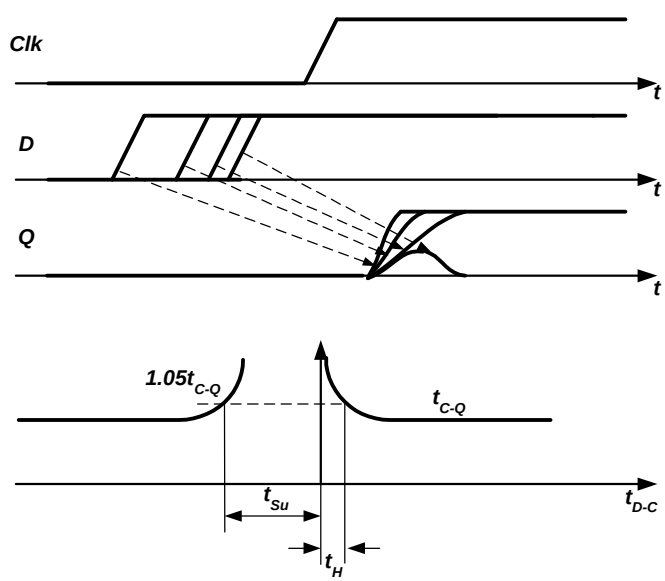


Figure E.3 Characterization of sequential elemnts: (a) determining the setup time of a register; (b) definition of setup and hold times.

Clearly, a more precise definition of the "setup time" concept is necessary. A unambiguous specification can be obtained by plotting the $t_{C\text{-}Q}$ delay against the data-to-clock offset, as shown in Figure E.3b. The degradation of the delay for smaller values of the off-

set can be observed. The actual definition of the setup time is rather precarious. If it were defined as the minimum *D-Clk* offset that causes the flip-flop to fail, the logic following the register would suffer from excessive delay if the offset is close to, but larger than, that point of failure. Another option is to place it at the operation point of the register that minimizes the sum of the data-clock offset and the $t_{C\text{-}Q}$ delay. This point, which minimizes the overall flip-flop overhead, is reached when the slope of the delay curve in Figure E.3b equals 45 degrees.

While custom design can take advantage of driving flip-flops that close to their point of failure—and take all the risk that comes with it—, semi-custom design must take a more conservative approach. For the characterization of registers in a standard cell library, both setup and hold times are commonly defined as data-clock offsets that correspond to **some fixed percentage increase in $t_{C\text{-}Q}$**, typically set **at 5%**, as indicated in Figure E.3b. Note that these curves are different for 0-1 and 1-0 transitions, resulting in different setup (an hold) times for 0 and 1 values. Identical definitions hold for latches.

Example E.3 Register setup and hold times

REFERENCES

[STMicro01]