MEMORY ARCHITECTURE

Besides using memory to retain states, a digital system uses memory to store instructions and data. Today the most commonly known digital system is a digital computer. All digital computers being sold commercially are based on the same model: the von Neumann architecture. In this model a computer has three main parts: the central processing unit (CPU), the memory, and the input/output (I/O) unit. There are many ways to design and organize these parts in a computer. We use the term *computer architecture* to describe the art and science of building a computer. We view the *memory architecture* from four different perspectives: (1) memory access interface, (2) memory hierarchy, (3) memory organization, and (4) memory device technology.

First let us examine memory access interface. Logically, computer memory is a collection of sequential entries, each with a unique address as its label. Supplying the address of the desired entry to the memory results in accessing of data and programs. If the operation is to read, after a certain time delay, the data residing in the entry corresponding to the address is obtained. If the operation is to write, data are supplied after the address and are entered into the memory replacing the original content of that entry. Reading and writing can be done asynchronously and synchronously with a reference clock. Other control signals supply the necessary information to direct the transfer of memory contents. Some special memory structures do not follow this general accessing method of using an address. Two of the most frequently used are content addressable memory (CAM) and first-in first-out (FIFO) memory. Another type of memory device, which accepts multiple addresses and produces several results at different ports, is called multiported memory. One of the most common multiported memories, which is written in parallel but is read serially, is called video random access memory (VRAM or VDRAM). It gets its name because it is used primarily in computer graphic display applications.

The second perspective of the memory architecture is memory hierarchy. The speed of memory devices has been lagging behind the speed of processing units. As technology advances, processors become faster and more capable and larger memory spaces are required to keep up with the every increasing program complexity. Due to the nature of increasing memory size, more time is needed to decode wider and wider addresses and to sense the information stored in the ever-shrinking physical storage element. The speed gap between CPU and memory devices will continue to grow wider. The traditional strategy used to remedy this problem is called memory hierarchy. Memory hierarchy works because of the locality property of memory references. Program instructions are usually fetched sequentially, and data used in a program are related and tend to conjugate. Thus, a smaller but fast memory is allocated and brought right next to the processor to bridge the speed gap of the CPU and memory. There can be many levels in the hierarchy. As the distance grows greater between the CPU and memory levels, the performance requirement for the memory is relaxed. At the same time, the size of the memory grows larger to accommodate the overall memory size requirement.

Third we look at memory organization. Most of the time, a memory device is internally organized as a two-dimensional array of cells internally. Usually a cell can store one bit of information. A cell in this array is identified and accessed with row and column numbers. A memory device accepts an address and breaks it down into row and column numbers and uses them to identify the location of the cell being accessed. Sometimes, more than one cell can be accessed at a given time. The size of content that a memory transfers is called the width of the memory device. There are many ways to organize the array in a memory device. By organizing it differently, we can have different widths.

The last aspect of memory architecture is memory technology. Physically, memory can be implemented with different technology. Memory devices can be categorized according to their functionality and fall into two major categories: read-only memory (ROM) and write-and-read memory, more commonly known as random access memory (RAM). There is also another subcategory of ROM, mostly-read-but-sometimes-write memory or flash ROM memory. Within the RAM category there are two types of memory devices differentiated by storage characteristics, static and dynamic RAM or SRAM and DRAM, respectively. DRAM devices represent the stored information with charge. Therefore it needs to be refreshed periodically to prevent the corruption of its contents due to charge leakage. On the other hand, SRAM uses a bistable element to represent the stored information, and thus it does not need to be refreshed. Both of SRAM and DRAM are volatile memory devices, which means that their contents are lost if the power supply is removed from these devices. Nonvolatile memory retains its contents even when the power supply is turned off. All current ROM devices, including mostly-read-sometimes-write devices, are nonvolatile memories.

MEMORY ACCESS INTERFACE

Technology is not the only factor that contributes to the performance of a memory device. Architectural methods also affect the speed of memory. Some of the architectural features are time multiplexing, pipelining, burst mode, clocking methodology, and separated input and output ports. Many times we need to trade off cost with performance when deciding what method to use. We will first discuss several common features used in memory devices.

Asynchronous Versus Synchronous Access

Memory can be accessed asynchronously or synchronously. It is more natural to follow the asynchronous interface. In this mode an address is presented to the memory by a processor. After a certain delay, data are made available at the pin for access. We call the delay between address made available to data ready the *memory access time.* Sometimes the access time is measured from a particular control signal. For example, the time between read control line ready and

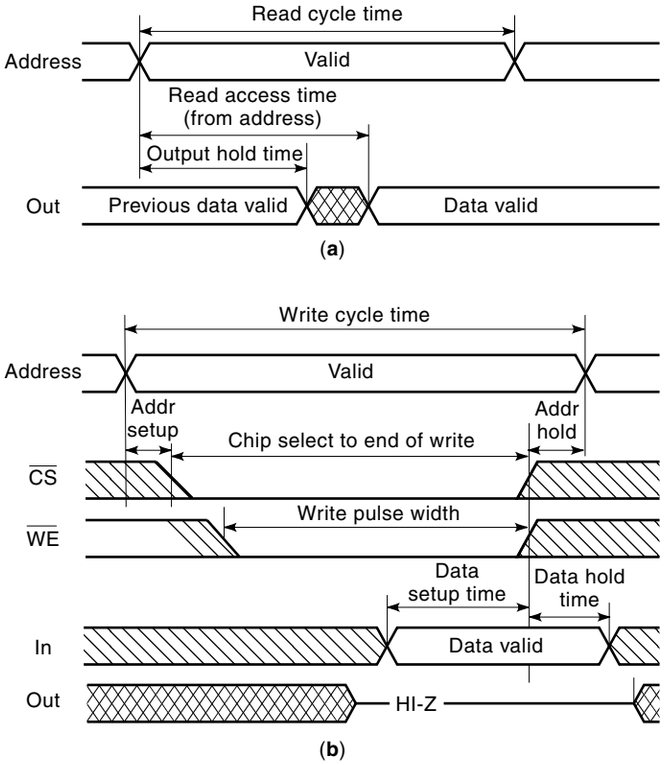

Figure 1. Asynchronous memory access. (a) Asynchronous read cycle. (b) Asynchronous write cycle.

data ready is called *read access time.* Figure 1 shows the timing diagrams of asynchronous memory access scheme. In the first diagram we assume that both the chip select and read enable signals are enabled. The write cycle diagram shown is a write cycle controlled by the write enable control signal. It is important to note that memory access time is different from memory cycle time. The *memory cycle time* is the minimum time between two consecutive memory accesses. The *memory writes command time* is measured from the write control ready to data stored in the memory. The *memory latency time* is the interval between CPU issuing an address and data available for processing. The *memory bandwidth* is the maximum amount of memory capacity being transferred in a given time.

Synchronous access implies a clock signal. Both address and control signals are latched into registers upon the arrival of the clock signal freeing the processor from holding the input to the memory for the entire access time. Instead the processor can initiate the access and continue to perform other important tasks. Figure 2 illustrates generic synchronous access cycles. In this figure we say that the read access has a two-cycle latency, since the data are made available after two clock cycles. Similarly we say that the write operation has zero-cycle latency.

Time Multiplexing

In order to reduce the cost of packaging, many different memory devices use time multiplexing to communicate information to and from other devices. One of the most common time-multiplexing examples is shared input/output (I/O). A memory chip can be configured with either separated or shared data inputs and outputs. The advantage of having a smaller package when shared inputs and outputs are used is more evident when the width of the data is large. However, the drawback is the possibility of having a slower interface due to contention. For a shared I/O device, either the write enable or the chip select control signal must be off during address transition when writing. Setting one of the control signals off disables the read operation. When the device is not being read, the I/O bus is set to high impedance, thus allowing the data input to be loaded onto the I/O pins. Other common examples of time multiplexing are most of the dynamic random access memory (DRAM) devices. DRAM differs from a static random access memory (SRAM) in that its row and column addresses are time-multiplexed. Again the main advantage is to reduce the pins of the chip package. Due to time multiplexing there are two address strobe lines for the DRAM address: row address strobe (RAS) line and column address strobe (CAS) line. These control signals are used to latch the row and column addresses, respectively. There are many ways to access the DRAM.

When reading, a row address is given first, followed by the row address strobe signal RAS. RAS is used to latch the row address on chip. After RAS, a column address is given followed by the column address strobe CAS. After a certain delay (read access time), valid data appear on the data lines. Memory write is done similarly to memory read, with only the read/write control signal reversed. There are three cycles available to write a DRAM. They are early write, read-modify-write, and late write cycles. Figure 3 shows only the early write cycle of a DRAM chip. Other write cycles can be found in most of the DRAM data books. We list a few of them here: (1) page mode, (2) extended data output (EDO) mode or hyper page mode, (3) nibble mode, and (4) static column mode.

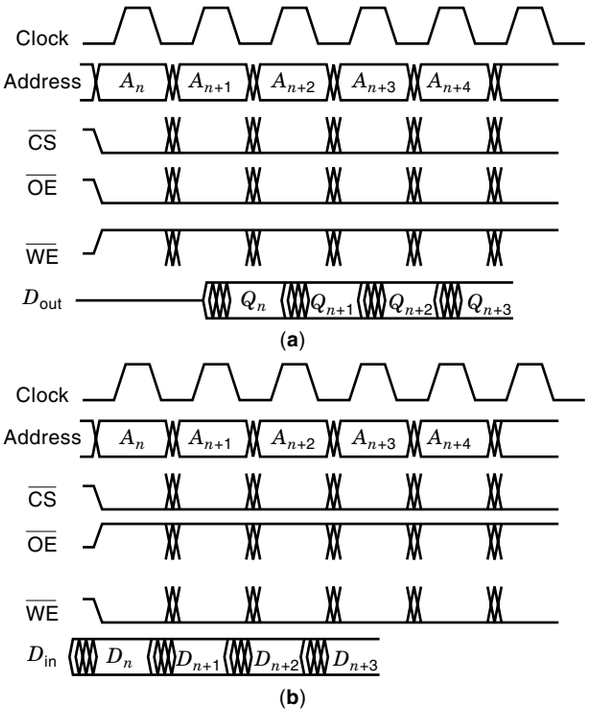

Figure 2. Synchronous memory access. (a) Synchronous (pipelined) read cycle. (b) Synchronous (pipelined) write cycle.

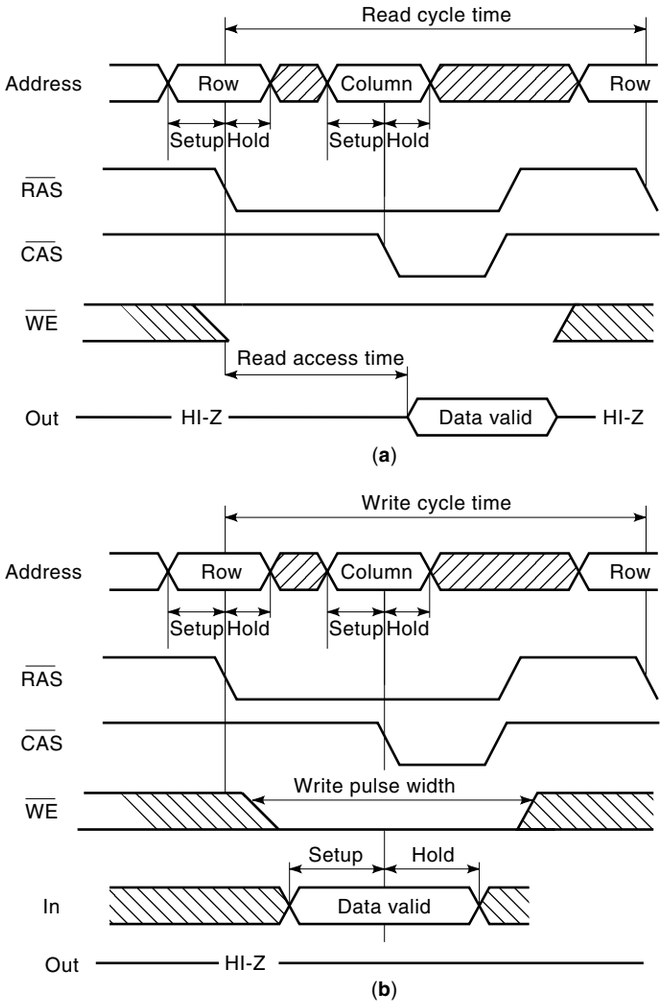

Figure 3. DRAM read and write cycles. (a) DRAM read cycle. (b) DRAM (Early) write cycle.

In page mode (or fast page mode), a read is done by lowering the RAS when the row address is ready. Then, repeatedly give the column address and CAS whenever a new one is ready without cycling the RAS line. In this way a whole row of the two-dimensional array (matrix) can be accessed with only one RAS and the same row address. This is called page mode, since we can arrange the memory device so that the upper part of the memory address specifies a page and the lower portion of the address is used as a column address to specify the offsets within a page. Due to locality, access local to the page does not need to change the row address, allowing faster access. Figure 4 illustrates the read timing cycle of a page mode DRAM chip. Static column is almost the same as page mode except the CAS signal is not cycled when a new column address is given—thus the static column name. In page mode, CAS must stay low until valid data reach the output. Once the CAS assertion is removed, data are disabled and the output pin goes to the open circuit. With EDO DRAM, an extra latch following the sense amplifier allows the CAS line to return to high much sooner, permitting the memory to start precharging earlier to prepare for the next access. Moreover, data are not disabled after CAS goes high. With burst EDO DRAM, not only does the CAS line return to high, it can also be toggled to step though the sequence in burst counter mode, providing even faster data transfer between memory and the host. IBM originated the EDO mode and called it the hyper page mode (HPM). In the nibble mode after one CAS with a given column, three more accesses are performed automatically without giving another column address (the address is assumed to be increased from the given address).

Special Memory Structures

The current trend in memory devices is toward larger, faster, better-performance products. There is a complementary trend toward the development of special purpose memory devices. Several types of special-purpose memory are offered for particular applications such as content addressable memory for cache memory, line buffers (FIFO or queue) for office automation machines, frame buffers for TV and broadcast equipment or queue, and graphics buffers for computers.

A special type of memory called content addressable memory (CAM) or associative memory is used in many applications such as cache memory and associative processor. CAM is also used in many structures within the processor such as scheduling circuitry and branch prediction circuitry. A CAM stores a data item consisting of a tag and a value. Instead of giving an address, a data pattern is given to the tag section of the CAM. This data pattern is matched with the content of the tag section. If an item in the tag section of the CAM matches the supplied data pattern, the CAM will output the value associated with the matched tag. CAM cells must be both readable and writable just like the RAM cell. Most of the time the matching circuit is built within the memory cell to reduce the circuit complexity. Figure 5 shows a circuit diagram for a basic CAM cell with a "match" output signal. This output signal may be used as input for other logic such as scheduling or used as an enable signal to retrieve the information contained in the other portion of the matched entry.

A FIFO/queue is used to hold data while waiting. It is often called a "buffer" because it serves as the buffering region for two systems, which may have different rates of consuming and producing data. A very popular application of FIFO is in office automation equipment. These machines require high-performance serial access of large amounts of data in each horizontal line such as digital facsimile machines, copiers and image scanners. FIFO can be implemented using shift registers or RAM with pointers.

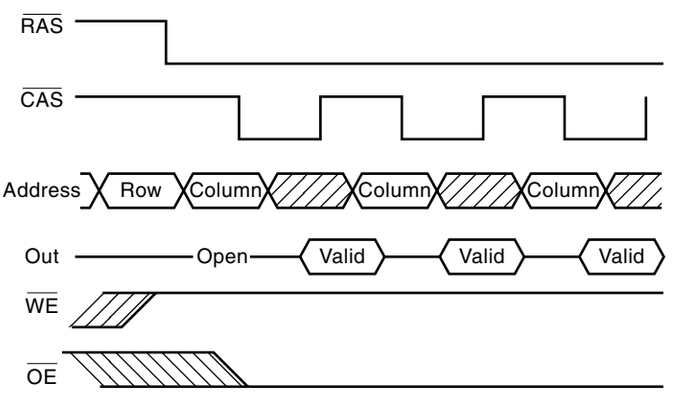

Figure 4. Page mode read cycle.

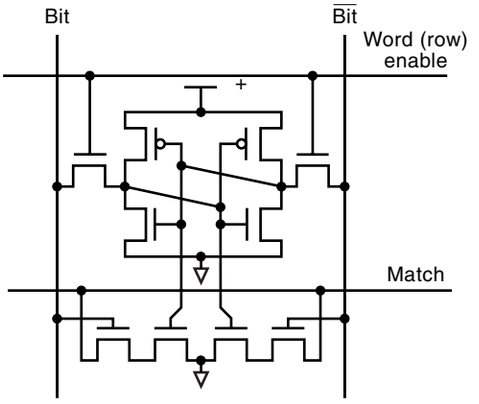

Figure 5. Static CMOS CAM cell.

There is rapid growth in computer graphic applications. The technology which is most successful, is termed raster scanning. In a raster scanning display system, an image is constructed with a series of horizontal lines. Each of these lines is connected to pixels of the picture image. Each pixel is represented with bits controlling the intensity. Usually there are three planes corresponding to each primary color: red, green, and blue. These three planes of bit maps are called frame buffer or image memory. Frame buffer architecture affects the performance of a raster scanning graphic system greatly. Since these frame buffers need to be read out serially to display the image line by line, a special type of DRAM memory called video memory or VDRAM is used. Usually this memory is dual ported with a parallel random access port for writing and a serial port for reading. Although synchronous DRAMs are still popular for current PCs, VDRAM is used commonly in high-end graphic systems because of the memory access bandwidth required. We can calculate the memory bus speed as follows. Assume we have a screen size of x by y pixels. Each pixel is made of three colors of z bytes. We further assume that the refresh cycle of the screen is r Hz. Then the total data rate required is the product of all four terms $xyzr$. Now depending on the memory we use, only a certain percentage of the memory access time can be allocated for refresh. Other times we need the interface channel to store new image information. That is, only a portion of the bandwidth is available for reading, since we need to write and refresh the memory. Let's assume that the portion used for refresh (refresh efficiency) is e. We further assume that the width of the memory system is w, and then the memory bus speed required to provide the refresh rate for this graphic screen is $xyzr/we$. For example, in order to refresh a screen size of 1280 $\times$ 1024 pixels with 3 bytes (1 byte for each primary color) at 75 Hz and a 30% refresh efficiency, we need a bus speed of 245 MHz if the bus width is 32 bits. Figure 6 illustrates two designs of a multiple-ported SRAM cell.

New Memory Interface Technique

Until recently, memory interface has progressed with evolution instead of revolution. However, since the memory bandwidth requirement continues to grow, revolutionary techniques are necessary. A new general method uses a packet-type of memory interface. One such interface is proposed by Rambus called Direct RDRAM. Another is termed SLDRM. Both technologies use a narrow bus topology with matched termination operating at high clock frequency to provide the needed bandwidth. In addition, they utilize heavily banked memory blocks to allow parallel access to the memory arrays providing the needed average access time (see paragraph on memory interleaving in the "Memory Organization" section to learn more about memory banks).

MEMORY HIERARCHY

Modern computer systems have ever growing applications. As a result, the application programs running on these computer systems grow in size and require large memories with quick access time. However, the speed of memory devices has been lagging behind the speed of processors. As CPU's speed continues to grow with the advancement of technology and design technique (in particular pipelining), due to the nature of increasing memory size, more time is needed to decode wider and wider addresses and to sense the information stored in the ever-shrinking storage element. The speed gap between processor and memory will continue to grow wider in the future. Cost is another important reason why memory hierarchy is important. Memory hierarchy works because of the locality property of memory references due to the sequentially fetched program instructions and the conjugation of related data. It works also because we perform memory reads much more than memory writes. In a hierarchical memory system there are many levels of memory. A small amount of very fast memory is usually allocated and brought right next to the central processing unit to help match up the speed of the CPU and memory. As the distance becomes greater between the CPU and memory, the performance requirement for the memory is relaxed. At the same time, the size of the memory grows larger to accommodate the overall memory size requirement. Some of the memory hierarchies are registers, cache, main memory, and secondary memory (or disk). When a memory reference is made, the processor accesses the memory at the top of the hierarchy. If the desired

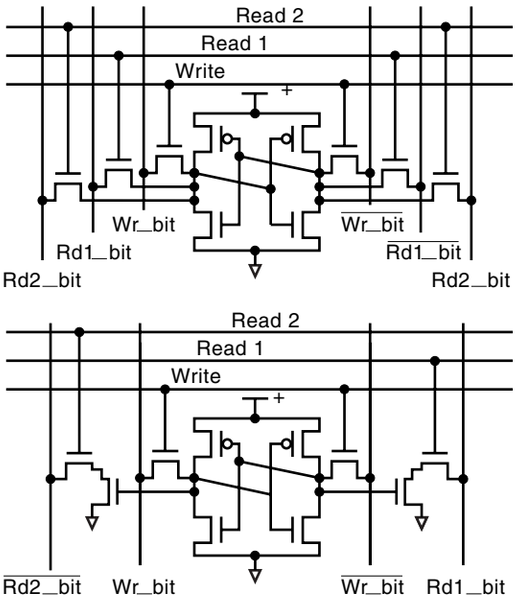

Figure 6. Two designs of Multiported CMOS SRAM cell (shown with 2-read and 1-write ports).

data are in the higher hierarchy, it wins because information is obtained quickly. Otherwise a *miss* is encountered. The requested information must be brought up from a lower level in the hierarchy. We will discuss cache memory and virtual memory in more detail.

Cache

Cache memory provides a fast and effective access time to main memory. A memory reference *hits* if the data are found in the cache. It *misses* if the data are not in the cache and had to be brought in. The amount of misses over the total reference is called the *miss rate.* We may categorize the cache misses in three ways—*compulsory miss, capacity miss,* and *conflict miss.* Compulsory miss rate is independent of the cache organization. It is incurred when a new memory is referenced or after a cache flush. Capacity miss occurs mainly due to the fact that caches are smaller in size compared with main memory. Depending on the cache mapping strategy, there also may be conflict miss even when the cache is not filled. Conflict miss happens because two memory references are mapped into the same cache location. When a miss occurs, a whole block of memory containing the requested missing information is brought in from the lower hierarchy. This block of memory is called a *cache line* or simply a *cache block.* Cache line is the basic unit used in cache. Access to only a part of the line brings the entire line into the cache. Since data and instructions process spatial locality, an entire line acts like pre-fetching, since the nearby addresses are likely to be used soon. Large lines pre-fetch more. However, too large a line may bring unused memory into the cache and pollute the cache unnecessarily and cause the cache to have greater capacity miss. It also wastes memory bandwidth.

Each cache line coexists with a tag that identifies the data held in the line by the data's address. The line hits if the tag matches the requested address. *Sets* comprise lines and do not distinguish among these lines. That is, any lines within a set can be mapped into the same cache location. A cache access takes two steps. The first step is a selection step where the set is indexed. The second step is the tag check step where the tags from the lines are checked and compared against the address. The size of the set gives the *associativity* of the cache. A cache with set size of one is called a *direct mapped* cache. A set size of two is called a *two-way set-associative* cache. A cache with all lines in one set is called *fully associative.* There are several ways to map the cache line into the cache from the main memory. We illustrate these mapping methods with an example. Assume that there are 8 blocks in a cache. An address 11 will map to location 3 in a direct mapped cache. The same address will be mapped to either location 6 or 7 if the cache is two-way set associative. If the cache is a four-way associative cache, then the address 11 may be mapped to locations 4 to 7 of the cache. In a fully associative cache, the address 11 may be mapped into any location of the cache. Figure 7 shows this example in detail. With higher associativity, conflict misses can be reduced. However, such cashes are more complex to build too. In general, associativity trades latency for miss rate. A fully associative cache is a CAM; since each address may be mapped to any location of the cache, a reference to see if an entry is in the cache needs to check every tag of the entire cache. When a memory location needs to be updated with a new result, we must update both the cache and the main memory. The *write-through* cache updates both the cache and the memory simultaneously at the time a write is issued. The *copy-back* (or *write back*) cache does not update immediately the main memory at writing until a block is replaced from the cache. This technique requires an extra bit for each cache block signaling whether the block is *dirty* (has changed the content since reading into the cache) or not. With the dirty bit, we don't have to write the memory every time a cache block is replaced. Only the block with the dirty bit set needs to be written into the main memory while others are simply thrown away. However, in a multi-processor system we need to prevent a processor from reading a stalled cache line, when that cache line has been written by another processor with the copy-back write policy. That is, we need to enforce the coherency of the cache. A popular method is called snooping cache. In this method all caches monitor the memory bus activity. When a cache write occurs, it updates the cache and also issues a memory write cycle for the first word of the cache line. All other caches snooping on the memory bus cycle will detect this write and invalidate the cache line in their cache. Write-through cache requires a larger memory bandwidth and has a longer average write access time.

If the current memory hierarchy level is full when a miss occurs, some existing blocks must be removed and sometimes written back to a lower level to allow the new one(s) to be brought in. There are several different replacement algorithms. One of the commonly used methods is the *least recently used* (LRU) replacement algorithm. Other algorithms are first-in first-out (FIFO) and random. In modern computing systems, there may be several sublevels of cache within the hierarchy of cache. For example, the Intel Pentium PRO system has on-chip cache (on the CPU chip) which is called Level 1 (L1) cache. There is another level of cache which resides in the same package (multichip module) with the CPU chip which is called Level 2 (L2) cache. There could also be a Level 3 (L3) cache on the motherboard (system board) between the CPU chip(s) and main memory chips (DRAMs). Moreover, there are also newer memory devices such as synchronous RAM, which provides enough bandwidth and speed to be interfaced with a processor directly through pipelining. We can express the average memory access time with the following equation:

$$T_{\text{avg}} = \sum_{i=j}^{p_i} \left[p_i \prod_{j=j}^{i-j} (1 - p + j) t_{t_i} \right] + \prod_{i=j}^{p_i} (1 - p_i) t_{\text{m}}$$

For example, a particular computer system has two levels of cache between the processor and the main memory. L1 cache has the same access time as the processor (t). L2 cache an access time 5 times the processor cycle time. Main memory has an access time 50 times the processor cycle time. If we assume that a particular program running on this system has an L1 cache hit rate of 95% and an L2 hit rate of 70%, the average memory access time will be $1.875t$. If we use some kind of cleaver design and increase the hit rate of L2 by 5%, the average access time will reduce to $1.7625t$. On the other hand, if we introduce another level of hierarchy between the main memory and L2 cache, which has a hit rate of 60% and an access time of $20t$, the average access time will reduce further to $1.605t$ instead. By making the cache smarter and having more levels of cache, we can reduce the average memory access time, assuming that the memory

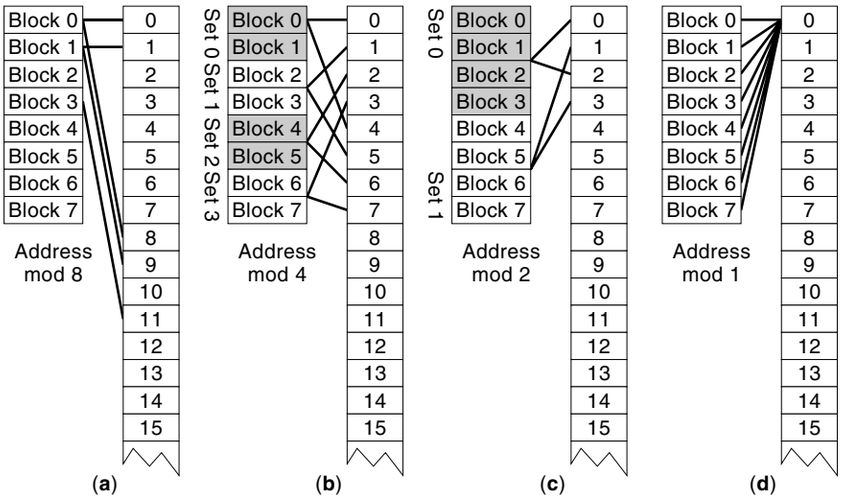

Figure 7. Mapping methods. (a) Direct mapped. (b) Two-way set associative. (c) Four-way set associative. (d) Fully associative.

access time keeps up with the processor cycle time. Unfortunately the trend says otherwise. The speed gap between DRAM and CPU continues to grow. The following scenario explains the effect of this gap. In most programs, 20% to 40% of the instructions reference memory; a particular program that references the memory with 25% of its instruction means that, on average, during execution every fourth instruction references memory. The previous memory system, with three levels of cache, will reach this barrier when the average memory cycle time (in multiples of processor cycle time) reaches $450t$. That is, at the speed ratio the computer system performance running this program is totally determined by memory speed. Making the processor faster will not affect the wall clock to complete the program. We call this the "memory wall."

Virtual Memory

A virtual memory system provides a memory space that is larger than the actual physical memory size of application program being executed. In a computer system the size of the total memory space is usually defined by the instruction set and memory management architecture. The size of the total memory space is typically governed by the width of the computer data path, since a computer uses the arithmetic unit of the CPU to calculate addresses. For example, a 32-bit processor usually has a memory space of size 4 GB (2 to the power of 32). We refer to this type of memory space as linear address space. A clear exception to this rule is the Intel Architecture (or $\times$86 architecture). The 32-bit Intel Architecture (IA-32) uses segmentation to manage its memory and gives a larger space than 4 GB. Nevertheless, all modern processors divide the entire memory space into chunks which are called *pages*. The size of a memory chunk is called page size. A typical page size is about a few kilobytes. A special program called operation system (OS) manages the pages by setting up a page table. A page table keeps track of pages that are actually in the physical memory. When a process makes a memory reference by issuing a virtual address, this is translated into (1) an index in the page table in order to locate the page this address is in and (2) an offset within the located page. If the page that it looks up is not in the physical memory, a page fault occurs. *Demand paging* brings that page in from the secondary memory (usually a disk). Since the physical memory is smaller than the total memory space, eventually all space in the physical memory will be filled. After the physical memory is filled and a new page needs to be brought in, we must replace the existing page with a new page. This process of replacing an existing page is called *swapping*. If the total memory space required by a program is much larger than the physical memory space, we may thrash the computer by swapping back and forth pages which have been used recently. There also might be another level of indirection when the number of pages is too many. We call it the *directory table*. In this case a virtual address must be translated and used to look up the directory table to find the page table fist. Then a page entry is located within the page table where it has been located. Then an offset into the page table is used to locate the entry of the physical memory, which is being accessed. The looking up of tables required for every memory access can consume a significant amount of time since each is a memory reference too, not to mention the addition operation it sometimes needs. To speed up the translation time, a *translation lookaside buffer* (TLB) stores frequently used completed translations for reuse.

MEMORY ORGANIZATION

System Level Organization

So far, we have not specified the exact size of a memory entry. A commonly used memory entry size is one byte. For historical reasons, memory is organized in bytes. A byte is usually the smallest unit of information transferred with each memory access. Wider memory entry is becoming more popular as the CPU continues to grow in speed and complexity. There are many modern systems which have a data width wider than a byte. A common size is a double word (32-bit), for example, in current desktop computers. As a result, memory in bytes is organized in sections of multibytes. However, due to need for backward compatibility, these wide datapath systems are also organized to be byte addressable. The maximum width of the memory transfer is usually called *memory word length,* and the size of the memory in bytes is called *memory capacity*. Since there are different memory device sizes, the

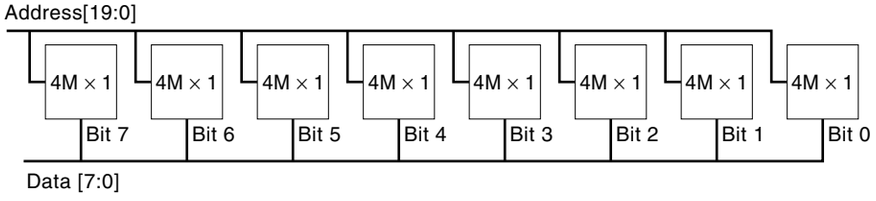

Figure 8. Eight 4 M × 1 chips used to construct a 4 Mbyte memory.

memory system can be populated with different-sized memory devices. For example, a 4 Mbyte of main memory (physical memory) can be put together with eight 4 Mbit × 1 chips as depicted in Fig. 8. It can also be designed with eight 512 Kbyte × 8-memory devices. Moreover, it can also be organized with a mixture of different-sized devices. These memory chips are grouped together to form memory modules. SIMM is a commonly used memory module which is widely used in current desktop computers. Similarly, a memory space can also be populated by different types of memory devices. For example, out of the 4MB space, some may be SRAM, some may be PROM, and some may be DRAM. They are used in the system for different purposes. We will discuss the differences of these different types of memory devices later.

There are two performance parameters in a memory system, namely, *memory bandwidth* and *memory latency*. In many cases the important factor in a high-performance computer system is the bandwidth because if we can access more data per access, then the average access time per data is shorter. However, a wider memory system is less flexible. It must increase by a larger chunk when upgraded.

Memory Device Organization

Physically, within a memory device, cells are arranged in a two-dimensional array, with each of the cells capable of storing one bit of information. Specifying the desired row and column addresses will access this matrix of cells. The individual row enable line is generated using an address decoder while the column is selected through a multiplexer. There is usually a sense amplifier between the column bit line and the multiplexer input to detect the content of the memory cell it is accessing. Figure 9 illustrates this general memory cell array described by an r-bit of row address and a c-bit of column address. With the total number of $r + c$ address bits, this memory structure contains a 2^{r+c} number of bits. As the size of the memory array increases, the row enable lines as well as the column bit lines become longer. In order to reduce the capacitive load of a long row enable line, the row decoders, sense amplifiers, and column multiplexers are often placed in the middle of divided matrices of cells as illustrated in Fig. 10. By designing the multiplexer differently we are able to construct memory with different output width—for example, ×1, ×8, ×16, and so on. In fact, memory designers make great effort to design the column multiplexers so that most of the fabrication masks may be shared for memory devices which have the same capacity but with different configurations. In large memory systems, with tens or hundreds of integrated circuit (IC) chips, it is more efficient to use 1-bit-wide (×1) memory IC chips. This tends to minimize the number of data pins for each chip, thereby reducing the total board area. One-bit-wide memory chips are a disadvantage in small systems, since a minimum of eight chips is needed to implement the desired memory for a memory system with one byte width. Due to the limit of board size, often several memory chips are connected to form a memory module on a specialized package. We called these memory modules. Some examples are SIMM, ZIF, and so on.

Memory Interleaving

Interleaving is a technique for organizing memory into leaves (memory banks) that increases the sustainable memory bandwidth. Each leaf can process a memory request for a processor independently. The latency of DRAM access, which is long

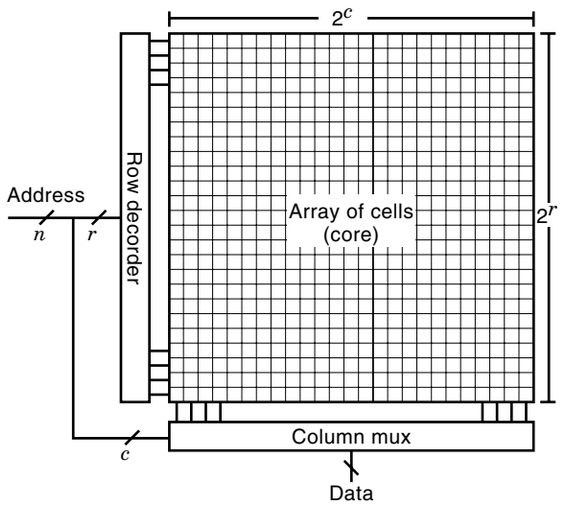

Figure 9. Generic 2-D memory structure.

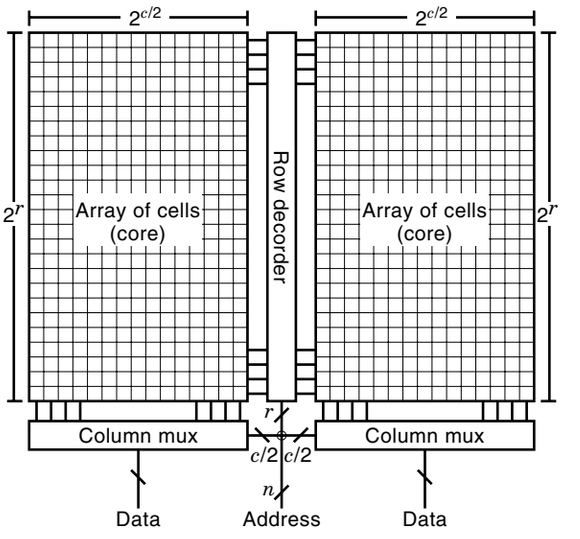

Figure 10. Divided memory structure.

compared with the CPU clock rate, is hidden from the processor when overlapped memory access is initiated in multiple memory leaves.

MEMORY DEVICE TYPES

As mentioned before, according to the functionality and characteristics of memory, we may divide memory devices into two major categories: ROM and RAM. We will describe these different types of devices in the following sections.

Read-Only Memory

In many systems, it is desirable to have the system level software (e.g., BIOS) stored in a read-only format, because these type of programs are seldom changed. Many embedded systems also use ROM to store their software routines because these programs are also never changed during their lifetime in general. Information stored in this ROM is permanent. It is retained even if the power supply is turned off. This memory can be read out reliably by a simple current-sensing circuit without worrying about destroying the stored data. The effective switch position at the intersection of word-line/bit-line determines the stored value. This switch could be implemented using many different technologies resulting in different types of ROM. The most basic type of this ROM is called masked ROM or simply ROM. It is programmed at the manufacturing time using fabrication processing masks. ROM can be produced using many different technologies: bipolar, CMOS, nMOS, pMOS, and so on. Once they are programmed, there is no means to change their contents. Moreover, the programming process is performed at the factory. Some ROM is also one time programmable, but it is programmable by the user at the user's own site. These are called programmable read-only memory (PROM). It is also often referred to as write-once memory (WOM). PROMs are based mostly on bipolar technology, since this technology supports it very nicely. Each of the single transistors in a cell has a fuse connected to its emitter. This transistor and fuse make up the memory cell. When a fuse is blown, no connection can be established when the cell is selected using the ROW line, and thus a zero is stored. Otherwise, with the fuse intact, logic one is represented. The programming is done through a programmer called PROM programmer or PROM burner. It is sometimes inconvenient to program the ROM only once. Thus the erasable PROM is designed. This type of erasable PROM is called EPROM. The programming of a cell is achieved by avalanche injection of high-energy electrons from the substrate through the oxide. This is accomplished by applying a high drain voltage, causing the electrons to gain enough energy to jump over the 3.2 eV barrier between the substrate and silicon dioxide, thus collecting charge at the floating gate. Once the applied voltage is removed, this charge is trapped on the floating gate. Erasing is done using an ultraviolet (UV) light eraser. Incoming UV light increases the energy of electrons trapped on the floating gate. Once the energy is increased above the 3.2 eV barrier, it leaves the floating gate and moves toward the substrate and the selected gate. Therefore these EPROM chips all have a window on their package where erasing UV light can reach inside the package to erase the content of cells. The erase time is usually in minutes. The presence of a charge on the floating gate will cause the metal oxide semiconductor (MOS) transistor to have a high threshold voltage. Thus even with a positive select gate voltage applied at the second level of poly-silicon, the MOS remains to be turned off. The absence of a charge on the floating gate causes the MOS to have a lower threshold voltage. When the gate is selected, the transistor will turn on and give the opposite data bit. EPROM technologies that migrate toward smaller geometry make floating-gate discharge (erase) via UV light exposure increasingly difficult. One problem is that the width of metal bit-lines cannot reduce proportionally with advancing process technologies. EPROM metal width requirements limit bit-lines spacing, thus reducing the amount of high-energy photons that reach charged cells. Therefore, EPROM products built on submicron technologies will face longer and longer UV exposure time.

Reprogrammability is a very desirable property. However, it is very inconvenient to use a separate light-source eraser for altering the contents of the memory. Furthermore, even a few minutes of erase time is intolerable. For this reason, a new type of erasable PROM is then designed, called EEPROM. EEPROM stands for electrical erasable PROM. EEPROM provides new applications where erase is done without removing the device from the system in which it resides. There are a few basic technologies used in the processing of EEPROMs or electrical reprogrammable ROMs. All of them use the Fowler–Nordheim tunneling effect to some extent. In this tunneling effect, cold electrons jump through the energy barrier at a silicon–silicon dioxide interface and into the oxide conduction band through the application of high field. This can only happen when the oxide thickness is of 100 Å or less depending on the technology. This tunneling effect is reversible, allowing the reprogrammable ROMs to be used over and over again.

A new alternative has been introduced recently, namely, flash EEPROM. This type of erasable PROMs lacks the circuitry to erase individual locations. When you erase them, they are erased completely. By doing so, many transistors may be saved, and larger memory capacities are possible. One needs to note that sometimes one does not need to erase before writing. One can also write to an erased, yet unwritten, location, which results in an average write time comparable to an EEPROM. Another important thing to know is that writing zeros into a location charges each of the flash EEPROM's memory cells to the same electric potential so that subsequent erasure will drain an equal amount of free charge (electrons) from each cell. Failure to equalize the charge in each cell prior to erasure can result in the overerasure of some cells by dislodging bound electrons in the floating gate and driving them out. When a floating gate is depleted in this way, the corresponding transistor can never be turned off again, thus destroying the flash EEPROM.

Random Access Memory

RAM stands for random access memory. It is really read-and-write memory because ROM is also random access in the sense that given an address randomly, the corresponding entry is read. RAM can be categorized by the duration its content can last. Static RAM's contents will always be retained as long as power is applied. On the other hand, a DRAM needs to the refreshed every few milliseconds. However, most RAMs by themselves are volatile, which means that without the power supply their content will be lost. All of the ROMs

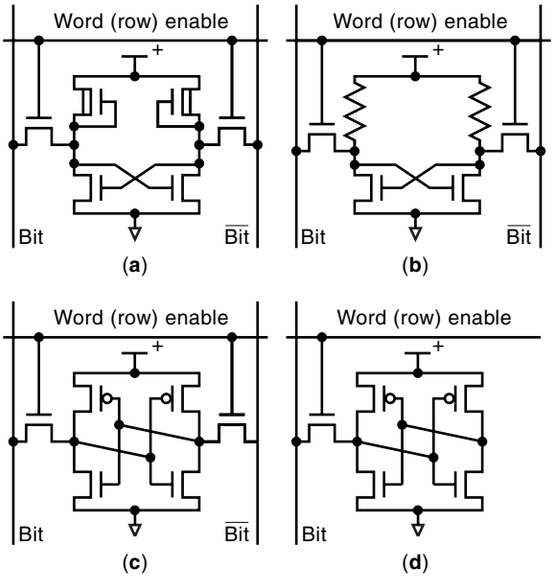

Figure 11. Different SRAM cell circuits. (a) Six-transistor SRAM cell with depletion transistor load. (b) Four-transistor SRAM cell with Poly-resistor load. (c) CMOS Six-transistor SRAM cell. (d) Five-transistor SRAM cell.

mentioned in the previous section are nonvolatile. RAM can be made nonvolatile by using a backup battery.

Figure 11 shows various SRAM memory cells (6T, 5T, and 4T). The six-transistor (6T) SRAM cell is the most commonly used SRAM. The crossed-coupled inverters in a SRAM cell retain the information indefinitely as long as the power supply is on, since one of the pull-up transistors supplies current to compensate for the leakage current. During a read, the bit and bitbar lines are pre-charged while the word enable line is held low. Depending on the content of the cell, one of the lines is discharged a little bit, causing the precharged voltage to drop, when the word enable line is strobed. This difference in voltage between the bit and bitbar lines is sensed by the sense amplifier, which produces the read result. During a write process, one of the bit/bitbar lines is discharged, and by strobing the word enable line the desired data are forced into the cell before the word line goes away.

The main disadvantage of SRAM is in its size since it takes six transistors (or at least four transistors and two resistors) to construct a single memory cell. Thus the DRAM is used to improve the capacity. Figure 12 shows the corresponding circuits for different DRAM cells. There is the four-transistor DRAM cell, the three-transistor DRAM cell, and the one-transistor DRAM cell. In a three-transistor cell DRAM, writing to the cell is accomplished by keeping the Read line low [refer to Fig. 12(b)] while strobing the Write line, and the desired data to be written are kept on the bus. If a one is desired to be stored. The gate of T2 is charged turning on T2. This charge will remain on the gate of T2 for a while before the leakage current discharge it to a point where it cannot be used to turn on T2. When the charge is still there, precharging the bus and strobing the Read line can perform a read. If a one is stored, then both T2 and T3 are on during a read, causing the charge on bus to be discharged. The sense amplifier can pick up the lowering of voltage. If a zero is stored, then there is no direct path from bus to GND; thus the charge on bus will remain. To further reduce the area of a memory cell, a single transistor cell is often used and is most common in today's commercial DRAM cell. Figure 12(c) shows the one-transistor cell with a capacitor. Usually two columns of cells are the mirror image of each other to reduce the layout area. The sense amplifier is shared. In this one-transistor DRAM cell, there is a capacitor used to store the charge, which determines the content of the memory. The amount of the charge in the capacitor also determines the overall performance of the memory. Putting either a 0 or 1 (the desired data to store) does the writing on the read/writing line. Then the row select line is strobed. A zero or one is stored in the capacitor as charge. A read is performed by precharging the read/write line and then strobing the row select. If a zero is stored due to charge sharing, the voltage on the read/write line will decrease. Otherwise the voltage will remain. A sense amplifier is placed at the end to pick up if there is a voltage change or not. DRAM differs from SRAM in another aspect. As the density of DRAM increases, the amount of charge stored in a cell also reduces. It becomes more subjective to noise. One type of noise is caused by radiation called alpha particles. These particles are helium nuclei, which are present in the environment naturally or are emitted from the package that houses the DRAM die. If an alpha particle hits a storage cell, it may change the state of the memory. Since alpha particles can be reduced but not eliminated, some DRAMs institute error detection and correction techniques to increase their reliability.

Since DRAM loses the charge with time, it needs to be refreshed periodically. Reading the information stored and writing it back does refresh. There are several methods to perform refresh. The first is *RAS-only refresh.* This type of refresh is done row by row. As a row is selected by providing the row address and strobing RAS, all memory cells in the row are refreshed in parallel. It will take as many cycles as the number of rows in the memory to refresh the entire device. For example, a 1M×1 DRAM which is built with 1024 rows and columns will take 1024 cycles to refresh the device. In order to reduce the number of refresh cycles, memory arrays are sometimes arranged to have fewer rows and more columns.

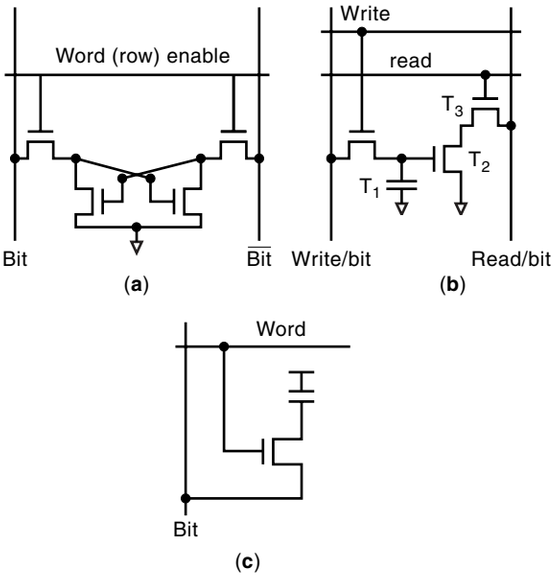

Figure 12. Different DRAM cells.

The address, however, is nevertheless multiplexed as two evenly divided words (in the case of 1M×1 DRAM the address word width is 10 bits each for rows and columns). The higher-order bits of address lines are used internally as column address lines, and they are ignored during the refresh cycle. No CAS signal is necessary to perform the RAS-only refresh. Since the DRAM output buffer is enabled only when CAS is asserted, the data bus is not affected during the RAS-only refresh cycles. Another method is called *hidden refresh.* During a normal read cycle, RAS and CAS are strobed after the respective row and column addresses are supplied. Instead of restoring the CAS signal to high after the read, several RAS may be asserted with the corresponding refresh row address. This refresh style is called the hidden refresh cycles. Again since the CAS is strobed and not restored, the output data are not affected by the refresh cycles. The number of refresh cycles performed is limited by the maximum time that CAS signal may be held asserted. One more method is named *CAS-before-RAS refresh* (self-refresh). In order to simplify and speed up the refresh process, an on-chip refresh counter may be used to generate the refresh address to the array. In such a case, a separate control pin is needed to signal to the DRAM to initiate the refresh cycles. However, since in normal operating RAS is always asserted before CAS for read and write, the opposite condition can be used to signal the start of a refresh cycles. Thus, in modern self-refresh DRAMs, if the control signal CAS is asserted before the RAS, it signals the start of refresh cycles. We called this CAS-before-RAS refresh, and it is the most commonly used refresh mode in 1 Mbit DRAMs. One discrepancy needs to be noted. In this refresh cycle the WE~ pin is a "don't care" for the 1 Mbit chips. However, the 4 Mbit specifies the CAS-before-RAS refresh mode with WE~ pin held at high voltage. A CAS-before-RAS cycle with WE~ low will put the 4 Meg into the JEDEC-specified test mode (WCBR). In contrast, applying a high to the test pin enters the 1 Meg test mode. All of the above-mentioned three refresh cycles can be implemented on the device in two ways. One method utilizes a distributed method, and the second method uses a wait-and-burst method. Devices using the first method refresh the row at a regular rate utilizing the CBR refresh counter to turn on rows one at a time. In this type of system, when it is not being refreshed, the DRAM can be accessed and the access can begin as soon as the self-refresh is done. The first CBR pulse should occur within the time of the external refresh rate prior to active use of the DRAM to ensure maximum data integrity and must be executed within three external refresh rate periods. Since CBR refresh is commonly implemented as the standard refresh, this ability to access the DRAM right after exiting the self-refresh is a desirable advantage over the second method. The second method is to use an internal burst refresh scheme. Instead of turning on rows at a regular interval, a sensing circuit is used to detect the voltage of the storage cells to see if they need to be refreshed. The refresh is done with a serial of refresh cycles one after another until all rows are completed. During the refresh, other access to the DRAM is not allowed.

CONCLUSION

Memory is becoming the determining factor in the performance of a computer. In this section we discussed four aspects of the memory architecture. These four aspects are (1) memory interface access, (2) memory hierarchy, (3) memory organization, and (4) memory devices. As projected, memory device size will continue to shrink and its capacity will continue to increase. Two newly merged memory architecture techniques to speed up computing systems are: (1) synchronous linked high-speed point-to-point connection and; (2) merged DRAM/logic.

GLOSSARY

Cache. A smaller and faster memory that is used to speed up the average memory access time.

CAM. Content addressable memory. This special memory is accessed not by an address but by a key, which matches to the content of the memory.

DRAM. Acronym for dynamic random access memory. This memory is dynamic because it needs to be refreshed periodically. It is random access because it can be read and written randomly.

Interleaved memory. Dividing a memory into multiple banks so that access to different banks can be in parallel.

Memory access time. The time between a valid address supplied to a memory device and data becoming ready at output of the device.

Memory bandwidth. Amount of memory access per unit time.

Memory cycle time. The time between subsequent address issues to a memory device.

Memory hierarchy. Organize memory in levels to make the speed of memory comparable to the processor.

Memory latency. The delay between address issue and data valid.

Memory read. The process of retrieving information from memory.

Memory write. The process of storing information into memory.

ROM. Acronym for read-only memory.

SRAM. Acronym for static random access memory. This memory is static because it does not need to be refreshed. It is random access because it can be read and written.

Virtual memory. A method to use a smaller physical memory to support a larger logical memory space.

SHIH-LIEN L. LU
Oregon State University

MEMORY, CACHE PROTOCOLS. See CACHE MEMORY PROTOCOLS.

MEMORY CIRCUITS. See BICMOS MEMORY CIRCUITS.

MEMORY CIRCUITS, BIPOLAR. See BIPOLAR MEMORY CIRCUITS.

MEMORY, MAGNETIC BUBBLE. See MAGNETIC BUBBLE MEMORY.

MEMORY-MAPPED FILES. See APPLICATION PROGRAM INTERFACES.

MEMORY, QUANTUM STORAGE. See QUANTUM STORAGE DEVICES.

MESSAGE PASSING. See DISTRIBUTED MEMORY PARALLEL SYSTEMS.

METACOMPUTING. See HETEROGENEOUS DISTRIBUTED COMPUTING.

METAL-INSULATOR-SEMICONDUCTOR (MIS) TRANSMISSION LINES. See SLOW WAVE STRUCTURES.

METALLURGY OF BETA TUNGSTEN SUPERCONDUCTORS. See SUPERCONDUCTORS, METALLURGY OF BETA TUNGSTEN.

METAL-METAL INTERFACES. See BIMETALS.

METAL-SEMICONDUCTOR BOUNDARIES. See OHMIC CONTACTS.