CHAPTER

7

MEMORY BASICS

Memory is a major component of a digital computer and is present in a large proportion of all digital systems. Random-access memory (RAM) stores data temporarily, and read-only memory (ROM) stores data permanently. ROM is one form of a variety of components called programmable logic devices (PLDs) that use stored information to define logic circuits.

Our study of RAM begins by looking at it in terms of a model with inputs, outputs, and signal timing. We then use equivalent logical models to understand the internal workings of RAM chips. Both static RAM and dynamic RAM are considered. The various types of dynamic RAM used for movement of data at high speeds between the CPU and memory are surveyed. Finally, we put RAM chips together to build simple RAM systems.

In many of the previous chapters, the concepts presented were broad, pertaining to much of the generic computer at the beginning of Chapter 1. In this chapter, for the first time, we can be more precise and point to specific uses of memory and related components. Beginning with the processor, the internal cache is very fast static RAM. Outside the CPU, the external cache is fast static RAM. The RAM subsystem, by its very name, is a type of memory. In the I/O area, we find substantial memory for storing information about the screen image in the video adapter. RAM appears in disk cache in the disk controller, to speed up disk access. Aside from the highly central role of the RAM subsystem in storing data and programs, we find memory in various forms applied in most subsystems of the generic computer.

7-1 MEMORY DEFINITIONS

In digital systems, memory is a collection of cells capable of storing binary information. In addition to these cells, memory contains electronic circuits for storing and retrieving the information. As indicated in the discussion of the generic computer, memory is used in many different parts of a modern computer, providing temporary or permanent storage for substantial amounts of binary information. In order for

this information to be processed, it is sent from the memory to processing hardware consisting of registers and combinational logic. The processed information is then returned to the same or to a different memory. Input and output devices also interact with memory. Information from an input device is placed in memory so that it can be used in processing. Output information from processing is placed in memory, and from there it is sent to an output device.

Two types of memories are used in various parts of a computer: *random-access memory* (RAM) and *read-only memory* (ROM). RAM accepts new information for storage to be available later for use. The process of storing new information in memory is referred to as a *memory write* operation. The process of transferring the stored information out of memory is referred to as a *memory read* operation. RAM can perform both the write and the read operations, whereas ROM, as introduced in Section 6-8, performs only read operations. RAM sizes may range from hundreds to billions of bits.

7-2 Random-Access Memory

Memory is a collection of binary storage cells together with associated circuits needed to transfer information into and out of the cells. Memory cells can be accessed to transfer information to or from any desired location, with the access taking the same time regardless of the location, hence the name *random-access memory.* In contrast, *serial memory*, such as is exhibited by a hard drive, takes different lengths of time to access information, depending on where the desired location is relative to the current physical position of the disk.

Binary information is stored in memory in groups of bits, each group of which is called a *word.* A word is an entity of bits that moves in and out of memory as a unit—a group of 1s and 0s that represents a number, an instruction, one or more alphanumeric characters, or other binary-coded information. A group of eight bits is called a *byte.* Most computer memories use words that are multiples of eight bits in length. Thus, a 16-bit word contains two bytes, and a 32-bit word is made up of four bytes. The capacity of a memory unit is usually stated as the total number of bytes that it can store. Communication between a memory and its environment is achieved through data input and output lines, address selection lines, and control lines that specify the direction of transfer of information. A block diagram of a memory is shown in Figure 7-1. The n data input lines provide the information to be stored in memory, and the n data output lines supply the information coming out of memory. The k address lines specify the particular word chosen among the many available. The two control inputs specify the direction of transfer desired: the Write input causes binary data to be transferred into memory, and the Read input causes binary data to be transferred out of memory.

The memory unit is specified by the number of words it contains and the number of bits in each word. The address lines select one particular word. Each word in memory is assigned an identification number called an *address.* Addresses range from 0 to $2^k - 1$, where k is the number of address lines. The selection of a specific word inside memory is done by applying the k-bit binary address to the address lines. A decoder accepts this address and opens the paths needed to select the word

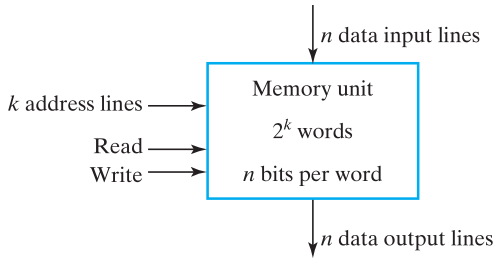


FIGURE 7-1
Block Diagram of Memory

specified. Computer memory varies greatly in size. It is customary to refer to the number of words (or bytes) in memory with one of the letters K (kilo), M (mega), or G (giga). K is equal to 2^{10}, M to 2^{20}, and G to 2^{30}. Thus, $64K = 2^{16}$, $2M = 2^{21}$, and $4G = 2^{32}$.

Consider, for example, a memory with a capacity of 1K words of 16 bits each. Since $1K = 1024 = 2^{10}$, and 16 bits constitute two bytes, we can say that the memory can accommodate 2048, or 2K, bytes. Figure 7-2 shows the possible contents of the first three and the last three words of this size of memory. Each word contains 16 bits that can be divided into two bytes. The words are recognized by their decimal addresses from 0 to 1023. An equivalent binary address consists of 10 bits. The first address is specified using ten 0s, and the last address is specified with ten 1s. This is because 1023 in binary is equal to 1111111111. A word in memory is selected by its binary address. When a word is read or written, the memory operates on all 16 bits as a single unit.

The 1K × 16 memory of the figure has 10 bits in the address and 16 bits in each word. The number of address bits needed in memory is dependent on the total

Memory Address		
Binary	Decimal	Memory Contents
0000000000	0	10110101 01011100
0000000001	1	10101011 10001001
0000000010	2	00001101 01000110
	⋮	⋮
1111111101	1021	10011101 00010101
1111111110	1022	00001101 00011110
1111111111	1023	11011110 00100100

FIGURE 7-2
Contents of a 1024 × 16 Memory

number of words that can be stored and is independent of the number of bits in each word. The number of bits in the address for a word is determined from the relationship $2^k \geq m$, where m is the total number of words and k is the minimum number of address bits satisfying the relationship.

Write and Read Operations

The two operations that a random-access memory can perform are write and read. A *write* is a transfer into memory of a new word to be stored. A *read* is a transfer of a copy of a stored word out of memory. A Write signal specifies the transfer-in operation, and a Read signal specifies the transfer-out operation. On accepting one of these control signals, the internal circuits inside memory provide the desired function.

The steps that must be taken for a write are as follows:

1. Apply the binary address of the desired word to the address lines.
2. Apply the data bits that must be stored in memory to the data input lines.
3. Activate the Write input.

The memory unit will then take the bits from the data input lines and store them in the word specified by the address lines.

The steps that must be taken for a read are as follows:

1. Apply the binary address of the desired word to the address lines.
2. Activate the Read input.

The memory will then take the bits from the word that has been selected by the address and apply them to the data output lines. The contents of the selected word are not changed by reading them.

Memory is made up of RAM integrated circuits (chips), plus additional logic circuits. RAM chips usually provide the two control inputs for the read and write operations in a somewhat different configuration from that just described. Instead of having separate Read and Write inputs to control the two operations, most integrated circuits provide at least a Chip Select that selects the chip to be read from or written to, and a Read/$\overline{\text{Write}}$ that determines the particular operation. The memory operations that result from these control inputs are shown in Table 7-1.

☐ TABLE 7-1
Control Inputs to a Memory Chip

Chip Select CS	Read/$\overline{\text{Write}}$ R/$\overline{\text{W}}$	Memory Operation
0	×	None
1	0	Write to selected word
1	1	Read from selected word

The Chip Select is used to enable the particular RAM chip or chips containing the word to be accessed. When Chip Select is inactive, the memory chip or chips are not selected, and no operation is performed. When Chip Select is active, the Read/$\overline{\text{Write}}$ input determines the operation to be performed. While Chip Select accesses chips, a signal is also provided that accesses the entire memory. We will call this signal the Memory Enable.

Timing Waveforms

The operation of the memory unit is controlled by an external device, such as a CPU. The CPU is synchronized by its own clock pulses. The memory, however, does not employ the CPU clock. Instead, its read and write operations are timed by changes in values on the control inputs. The *access time* of a memory read operation is the maximum time from the application of the address to the appearance of the data at the Data Output. Similarly, the *write cycle time* is the maximum time from the application of the address to the completion of all internal memory operations required to store a word. Memory writes may be performed one after the other at the intervals of the cycle time. The CPU must provide the memory control signals in such a way as to synchronize its own internal clocked operations with the read and write operations of memory. This means that the access time and the write cycle time of the memory must be related within the CPU to a period equal to a fixed number of CPU clock periods.

Assume, as an example, that a CPU operates with a clock frequency of 50 MHz, giving a period of 20 ns (1 ns = 10^{-9} s) for one clock pulse. Suppose now that the CPU communicates with a memory with an access time of 65 ns and a write cycle time of 75 ns. The number of clock pulses required for a memory request is the integer value greater than or equal to the larger of the access time and the write cycle time, divided by the clock period. Since the period of the CPU clock is 20 ns, and the larger of the access time and write cycle time is 75 ns, it will be necessary to devote at least four clock pulses to each memory request.

The memory cycle timing shown in Figure 7-3 is for a CPU with a 50 MHz clock and memory with a 75 ns write cycle time and a 65 ns access time. The write cycle in part (a) shows four pulses $T1$, $T2$, $T3$, and $T4$ with a cycle of 20 ns. For a write operation, the CPU must provide the address and input data to the memory. The address is applied, and Memory Enable is set to the high level at the positive edge of the $T1$ pulse. The data, needed somewhat later in the write cycle, is applied at the positive edge of $T2$. The two lines that cross each other in the address and data waveforms designate a possible change in value of the multiple lines. The shaded areas represent unspecified values. A change of the Read/$\overline{\text{Write}}$ signal to 0 to designate the write operation is also at the positive edge of $T2$. To avoid destroying data in other memory words, it is important that this change occur after the signals on the address lines have become fixed at the desired values. Otherwise, one or more other words might be momentarily addressed and accidentally written over with different data. The Read/$\overline{\text{Write}}$ signal must stay at 0 long enough after application of the address and Memory Enable to allow the write operation to complete. Finally, the address and data signals must remain

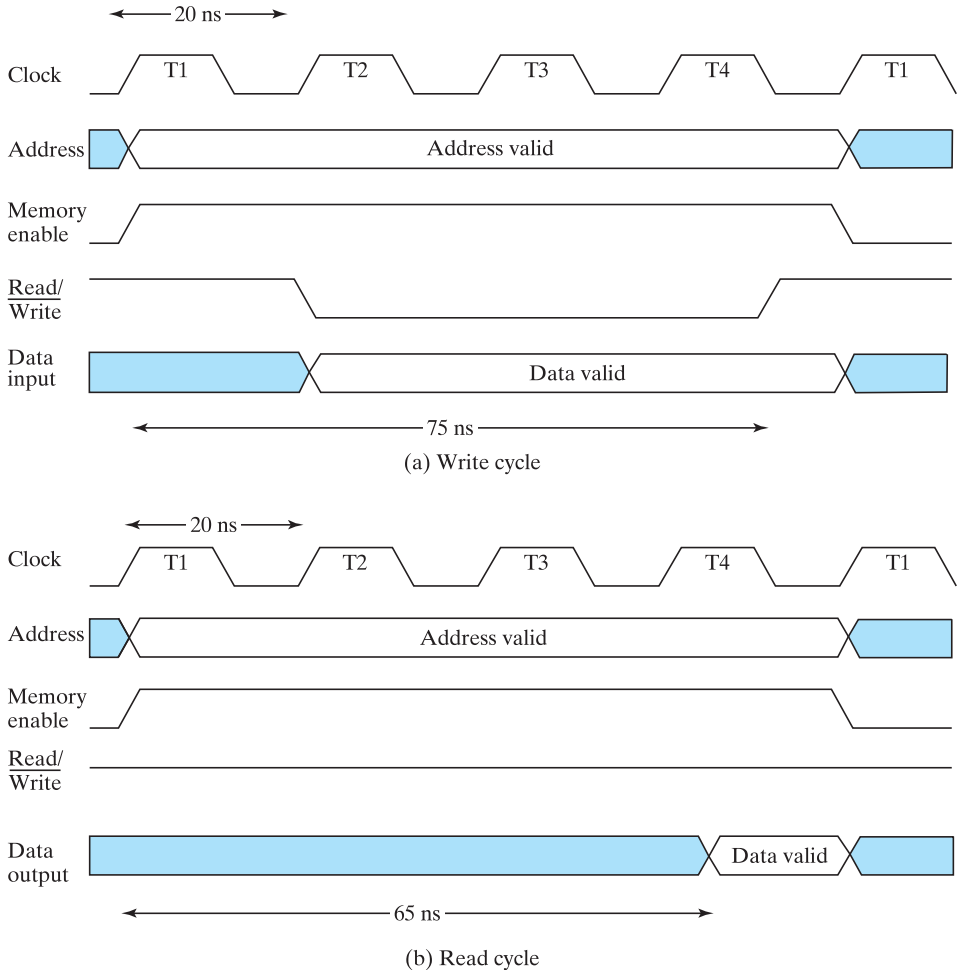


FIGURE 7-3
Memory Cycle Timing Waveforms

stable for a short time after the Read/$\overline{\text{Write}}$ goes to 1, again to avoid destroying data in other memory words. At the completion of the fourth clock pulse, the memory write operation has ended with 5 ns to spare, and the CPU can apply the address and control signals for another memory request with the next *T*1 pulse.

The read cycle shown in Figure 7-3(b) has an address for the memory that is provided by the CPU. The CPU applies the address, sets the Memory Enable to 1, and sets Read/$\overline{\text{Write}}$ to 1 to designate a read operation, all at the positive edge of *T*1. The memory places the data of the word selected by the address onto the data output lines within 65 ns from the time that the address is applied and the memory enable is activated. Then, the CPU transfers the data into one of its internal registers during the positive transition of the next *T*1 pulse, which can also change the address and controls for the next memory request.

Properties of Memory

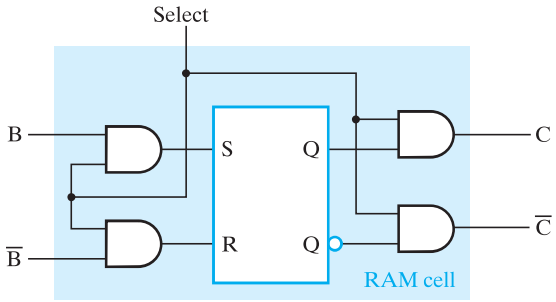
Integrated-circuit RAM may be either static or dynamic. *Static* RAM (SRAM) consists of internal latches that store the binary information. The stored information remains valid as long as power is applied to the RAM. *Dynamic* RAM (DRAM) stores the binary information in the form of electric charges on capacitors. The capacitors are accessed inside the chip by *n*-channel MOS transistors. The stored charge on the capacitors tends to discharge with time, and the capacitors must be periodically recharged by *refreshing* the DRAM. This is done by cycling through the words every few milliseconds, reading and rewriting them to restore the decaying charge. DRAM offers reduced power consumption and larger storage capacity in a single memory chip, but SRAM is easier to use and has shorter read and write cycles. Also, no refresh is required for SRAM.

Memory units that lose stored information when power is turned off are said to be *volatile*. Integrated-circuit RAMs, both static and dynamic, are of this category, since the binary cells need external power to maintain the stored information. In contrast, a *nonvolatile memory*, such as magnetic disk, retains its stored information after the removal of power. This is because the data stored on magnetic components is represented by the direction of magnetization, which is retained after power is turned off. Another nonvolatile memory is ROM, discussed in Section 5-2.

7-3 SRAM Integrated Circuits

As indicated earlier, memory consists of RAM chips plus additional logic. We will consider the internal structure of the RAM chip first. Then we will study combinations of RAM chips and additional logic used to construct memory. The internal structure of a RAM chip of m words with n bits per word consists of an array of mn binary storage cells and associated circuitry. The circuity is made up of decoders to select the word to be read or written, read circuits, write circuits, and output logic. The *RAM cell* is the basic binary storage cell used in the RAM chip, which is typically designed as an electronic circuit rather than a logic circuit. Nevertheless, it is possible and convenient to model the RAM chip using a logic model.

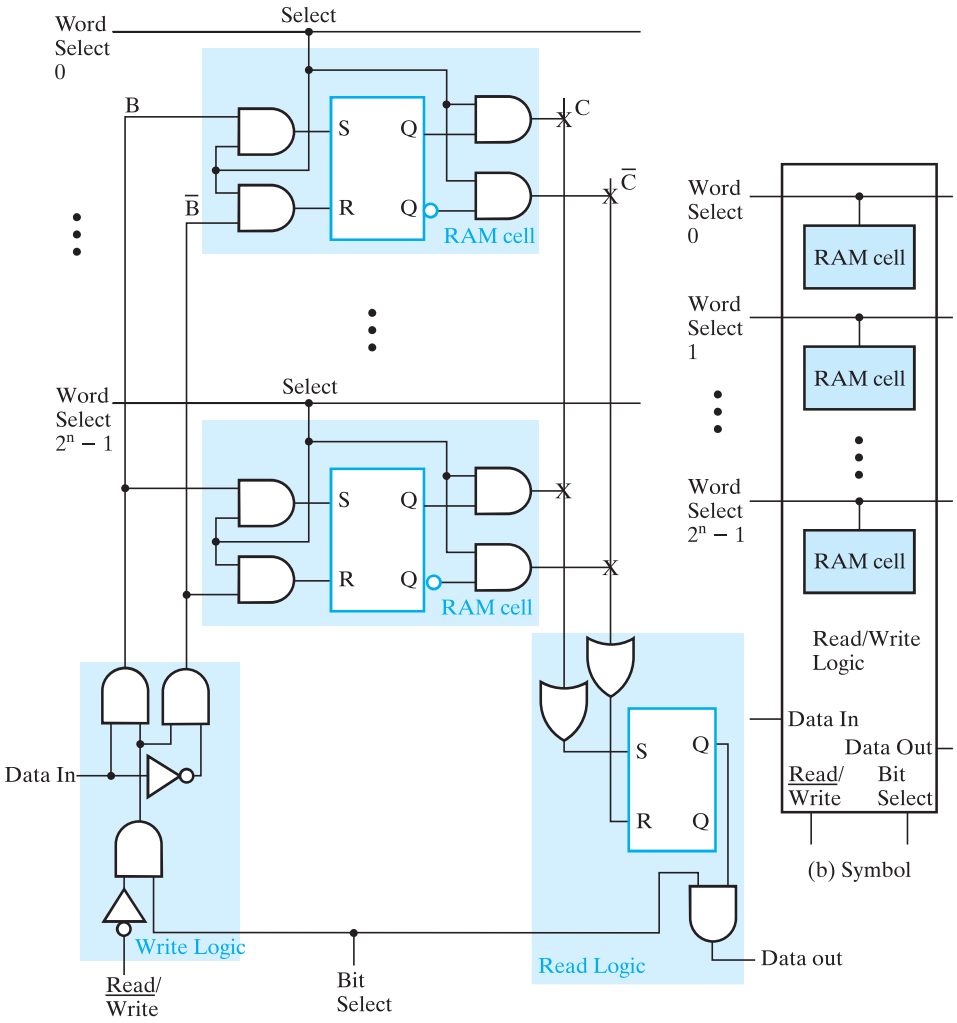
A static RAM chip serves as the basis for our discussion. We first present RAM cell logic for storing a single bit and then use the cell in a hierarchy to describe the RAM chip. Figure 7-4 shows the logic model of the RAM cell. The storage part of the



□ **FIGURE 7-4**
Static RAM Cell

cell is modeled by an *SR* latch. The inputs to the latch are enabled by a Select signal. For Select equal to 0, the stored content is held. For Select equal to 1, the stored content is determined by the values on B and $\overline{B}$. The outputs from the latch are gated by Select to produce cell outputs C and $\overline{C}$. For Select equal to 0, both C and $\overline{C}$ are 0, and for Select equal to 1, C is the stored value and $\overline{C}$ is its complement.

To obtain simplified static RAM diagrams, we interconnect a set of RAM cells and read and write circuits to form a *RAM bit slice* that contains all of the circuitry associated with a single bit position of a set of RAM words. The logic diagram for a RAM bit slice is shown in Figure 7-5(a). The portion of the model representing each RAM cell



□ **FIGURE 7-5**
RAM Bit Slice Model

is highlighted in blue. The loading of a cell latch is now controlled by a Word Select input. If this is 0, then both S and R are 0, and the cell latch contents remain unchanged. If the Word Select input is 1, then the value to be loaded into the latch is controlled by two signals B and $\overline{B}$ from the Write Logic. In order for either of these signals to be 1 and potentially change the stored value, Read/$\overline{\text{Write}}$ must be 0 and Bit Select must be 1. Then the Data In value and its complement are applied to B and $\overline{B}$, respectively, to set or reset the latch in the RAM cell selected. If Data In is 1, the latch is set to 1, and if Data In is 0, the latch is reset to 0, completing the write operation.

Only one word is written at a time. That is, only one Word Select line is 1, and all other Word Select lines are 0. Thus, only one RAM cell attached to B and $\overline{B}$ is written. The Word Select also controls the reading of the RAM cells, using shared Read Logic. If Word Select is 0, then the stored value in the SR latch is prevented by the AND gates from reaching the pair of OR gates in the Read Logic. But if Word Select is 1, the stored value passes through to the OR gates and is captured in the Read Logic SR latch. If Bit Select is also 1, the captured value appears on the Data Out line of the RAM bit slice. Note that for this particular Read Logic design, the read occurs regardless of the value of Read/$\overline{\text{Write}}$.

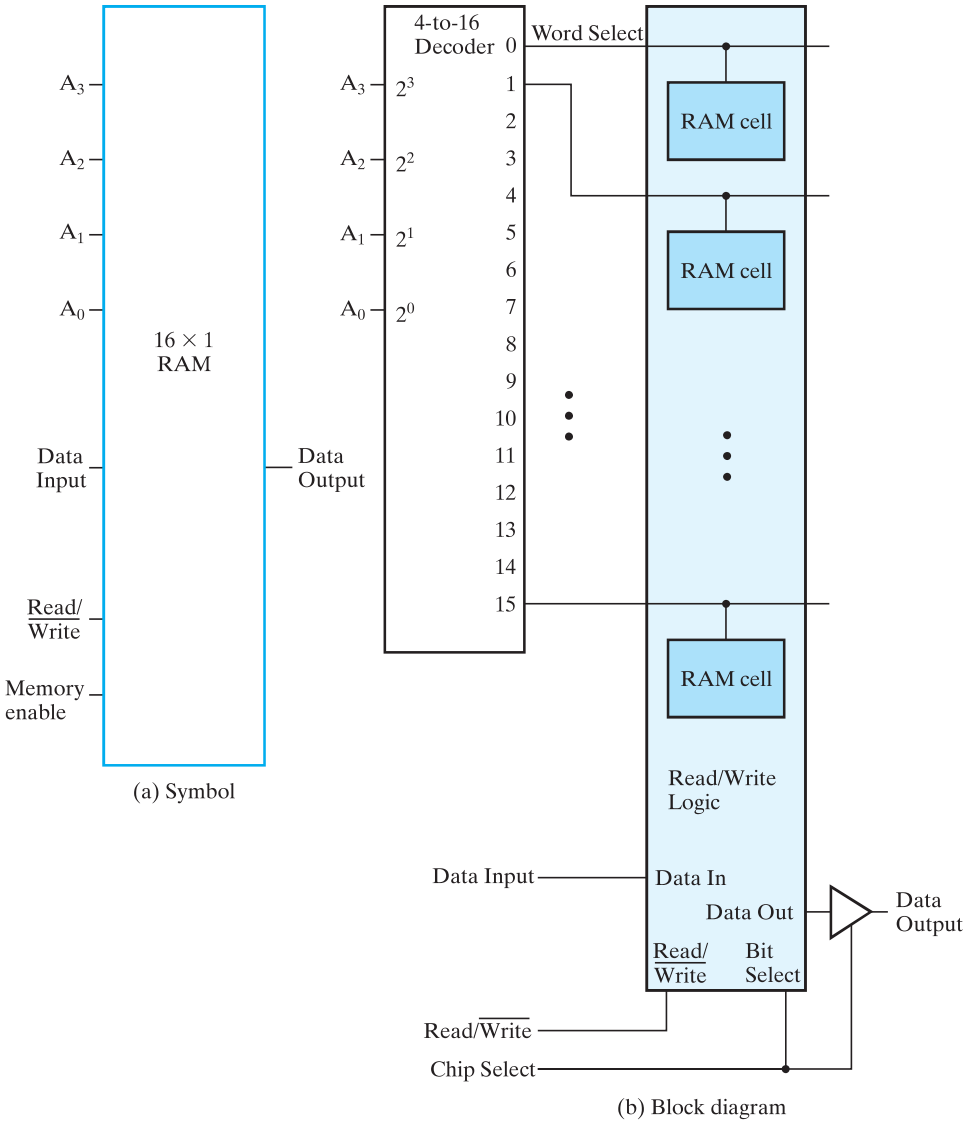
The symbol for the RAM bit slice given in Figure 7-5(b) is used to represent the internal structure of RAM chips. Each Word Select line extends beyond the bit slice, so that when multiple RAM bit slices are placed side by side, corresponding Word Select lines connect. The other signals in the lower portion of the symbol may be connected in various ways, depending on the structure of the RAM chip.

The symbol and block diagram for a 16 $\times$ 1 RAM chip are shown in Figure 7-6. Both have four address inputs for the 16 one-bit words stored in RAM. There are also Data Input, Data Output, and Read/$\overline{\text{Write}}$ signals. The Chip Select at the chip level corresponds to the Memory Enable at the level of a RAM consisting of multiple chips. The internal structure of the RAM chip consists of a RAM bit slice having 16 RAM cells. Since there are 16 Word Select lines to be controlled such that one and only one has the value logic 1 at a given time, a 4–to–16-line decoder is used to decode the four address bits into 16 Word Select bits.

The only additional logic in the figure is a triangular symbol with one normal input, one normal output, and a second input on the bottom of the symbol. This symbol is a three-state buffer that allows construction of a multiplexer with an arbitrary number of inputs. Three-state outputs are connected together and properly controlled using the Chip Select inputs. By using three-state buffers on the outputs of RAM chips, these outputs can be connected together to provide the word from the chip being read on the bit lines attached to the RAM outputs. The enable signals in the preceding discussion correspond to the Chip Select inputs on the RAM chips. To read a word from a particular RAM chip, the Chip Select value for that chip must be 1, and for all other chips attached to the same output bit lines, the Chip Select must be 0. These combinations containing a single 1 can be obtained from a decoder.

Coincident Selection

Inside a RAM chip, the decoder with k inputs and 2^k outputs requires 2^k AND gates with k inputs per gate if a straightforward design approach is used. In addition, if the



□ **FIGURE 7-6**
16-Word by 1-Bit RAM Chip

number of words is large, and all bits for one bit position in the word are contained in a single RAM bit slice, the number of RAM cells sharing the read and write circuits is also large. The electrical properties resulting from both of these situations cause the access and write cycle times of the RAM to become long, which is undesirable.

The total number of decoder gates, the number of inputs per gate, and the number of RAM cells per bit slice can all be reduced by employing two decoders with a *coincident selection* scheme. In one possible configuration, two $k/2$-input decoders are used

instead of one *k*-input decoder. One decoder controls the word select lines and the other controls the bit select lines. The result is a two-dimensional matrix selection scheme. If the RAM chip has *m* words with 1 bit per word, then the scheme selects the RAM cell at the intersection of the Word Select row and the Bit Select column. Since the Word Select is no longer strictly selecting words, its name is changed to *Row Select.* An output from the added decoder that selects one or more bit slices is referred to as a *Column Select.*

Coincident selection is illustrated for the 16 × 1 RAM chip with the structure shown in Figure 7-7. The chip consists of four RAM bit slices of four bits each and has a total of 16 RAM cells in a two-dimensional array. The two most significant address inputs go through the 2–to–4-line row decoder to select one of the four rows of the array. The two least significant address inputs go through the

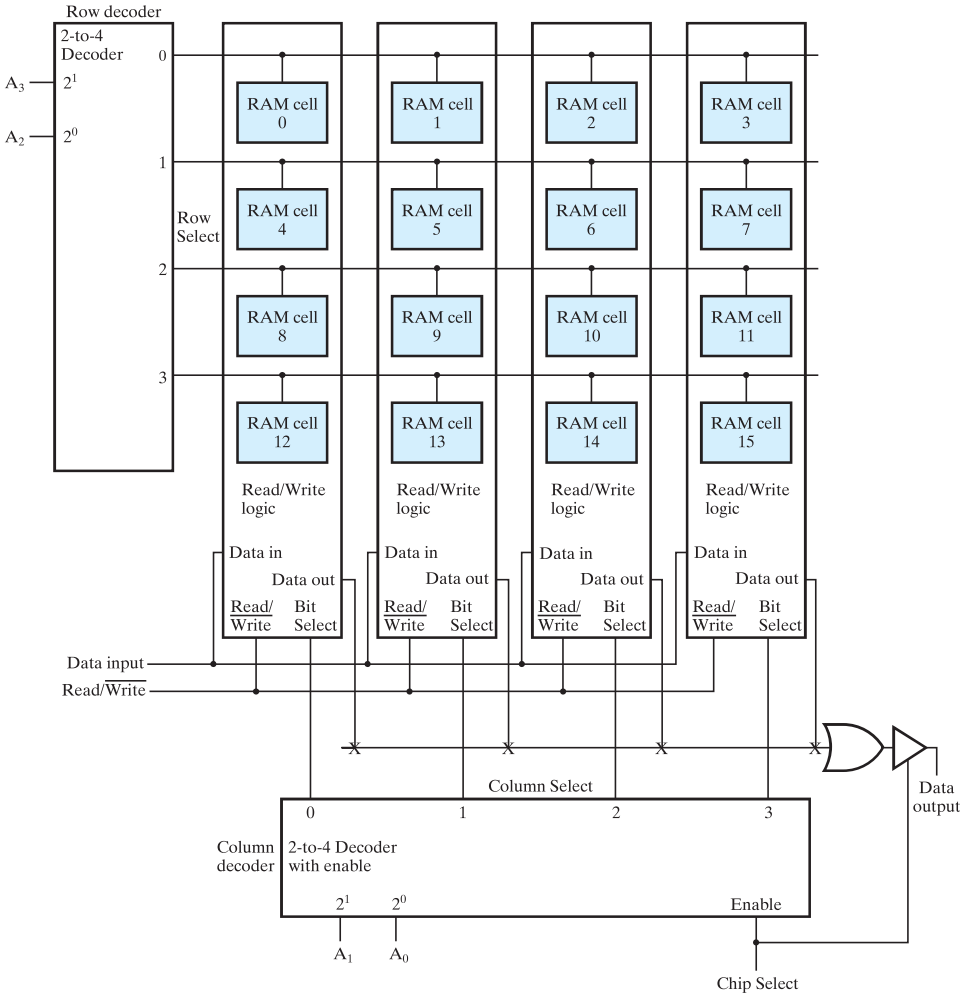


FIGURE 7-7
Diagram of a 16 × 1 RAM Using a 4 × 4 RAM Cell Array

2–to–4-line column decoder to select one of the four columns (RAM bit slices) of the array. The column decoder is enabled with the Chip Select input. When the Chip Select is 0, all outputs of the decoder are 0 and none of the cells is selected. This prevents writing into any RAM cell in the array. With Chip Select at 1, a single bit in the RAM is accessed. For example, for the address 1001, the first two address bits are decoded to select row 10 of the RAM cell array. The second two address bits are decoded to select column 01 of the array. The RAM cell accessed, in row 2 and column 1 of the array, is cell 9 ($10_2 01_2$). With a row and column selected, the Read/$\overline{\text{Write}}$ input determines the operation. During the read operation (Read/$\overline{\text{Write}}$ = 1), the selected bit of the selected row goes through the OR gate to the three-state buffer. Note that the gate is drawn according to the array logic established in Figure 5-5. Since the buffer is enabled by Chip Select, the value read appears at the Data Output. During the write operation (Read/$\overline{\text{Write}}$ = 0), the bit available on the Data Input line is transferred into the selected RAM cell. Those RAM cells not selected are disabled, and their previous binary values remain unchanged.

The same RAM cell array is used in Figure 7-8 to produce an 8 × 2 RAM chip (eight words of two bits each). The row decoding is unchanged from that in Figure 7-7; the only changes are in the column and output logic. Since there are just three address bits, and two are handled by the row decoder, the column decoder has only one address bit and Chip Select as inputs and produces just two Column Select lines. Since two bits at a time are to be written or read, the Column Select lines go to adjacent pairs of RAM bit slices. Two input lines, Data Input 0 and Data Input 1, each go to a different bit in all of the pairs. Finally, corresponding bits of the pairs share output OR gates and three-state buffers, giving output lines Data Output 0 and Data Output 1. The operation of this structure can be illustrated by the application of the address 3 (011_2). The first two bits of the address, 01, access row 1 of the array. The final bit, 1, accesses column 1, which consists of bit slices 2 (10_2) and 3 (11_2). So the word to be written or read lies in RAM cells 6 and 7 ($011\ 0_2$ and $011\ 1_2$), which contain bits 0 and 1, respectively, of word 3.

We can demonstrate the savings of the coincident selection scheme by considering a more realistic static RAM size, 32K × 8. This RAM chip contains a total of 256K bits. To make the number of rows and columns in the array equal, we take the square root of 256K, giving $512 = 2^9$. So the first nine bits of the address are fed to the row decoder and the remaining six bits to the column decoder. Without coincident selection, the single decoder would have 15 inputs and 32,768 outputs. With coincident selection, there is one 9–to–512-line decoder and one 6–to–64-line decoder. The number of gates for a straightforward design of the single decoder would be 32,800. For the two coincident decoders, the number of gates is 608, reducing the gate count by a factor of more than 50. In addition, although it appears that there are 64 times as many Read/Write circuits, the column selection can be done between the RAM cells and the Read/Write circuits, so that only the original eight circuits are required. Because of the reduced number of RAM cells attached to each Read/Write circuit at any time, the access time of the chip is also improved.

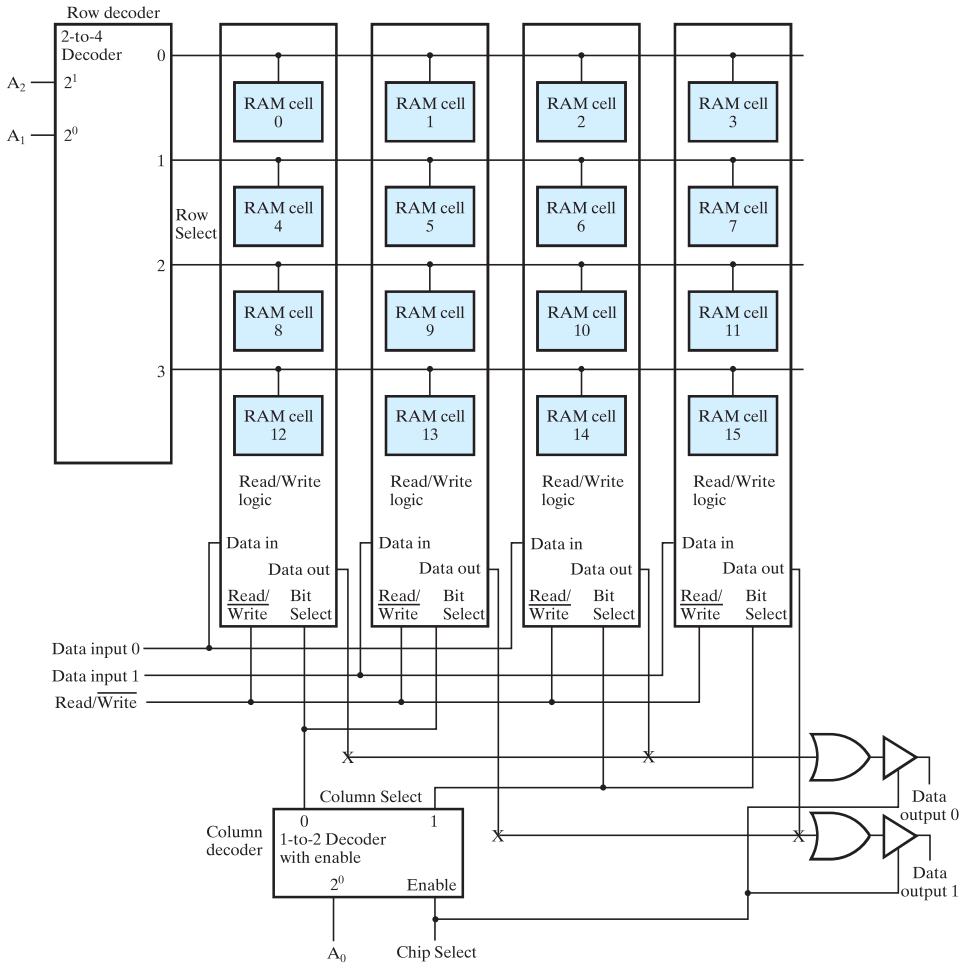
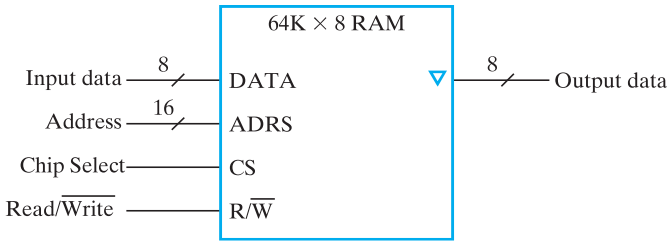


FIGURE 7-8
Block Diagram of an 8 × 2 RAM Using a 4 × 4 RAM Cell Array

7-4 Array of SRAM ICs

Integrated-circuit RAM chips are available in a variety of sizes. If the memory unit needed for an application is larger than the capacity of one chip, it is necessary to combine a number of chips in an array to form the required size of memory. The capacity of the memory depends on two parameters: the number of words and the number of bits per word. An increase in the number of words requires that we increase the address length. Every bit added to the length of the address doubles the number of words in memory. An increase in the number of bits per word requires that we increase the number of data input and output lines, but the address length remains the same.



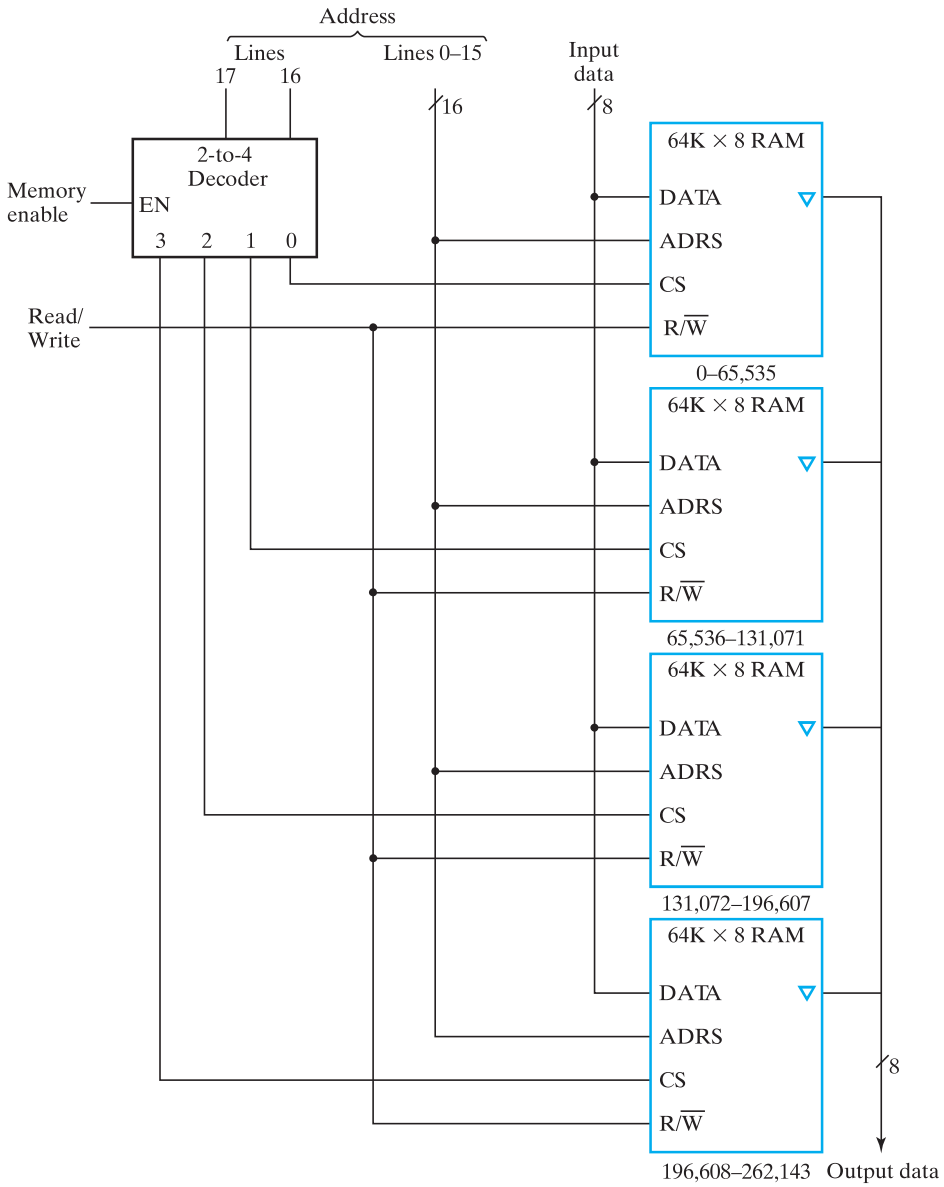
□ **FIGURE 7-9**
Symbol for a 64K × 8 RAM Chip

To illustrate an array of RAM ICs, let us first introduce a RAM chip using the condensed representation for inputs and outputs shown in Figure 7-9. The capacity of this chip is 64K words of 8 bits each. The chip requires a 16-bit address and 8 input and output lines. Instead of 16 lines for the address and 8 lines each for data input and data output, each is shown in the block diagram by a single line. Each line has a slash across it with a number indicating the number of lines represented. The *CS* (Chip Select) input selects the particular RAM chip, and the $R/\overline{W}$ (Read/$\overline{\text{Write}}$) input specifies the read or write operation when the chip is selected. The small triangle shown at the outputs is the standard graphics symbol for three-state outputs. The *CS* input of the RAM controls the behavior of the data output lines. When $CS = 0$, the chip is not selected, and all its data outputs are in the high-impedance state. With $CS = 1$, the data output lines carry the eight bits of the selected word.

Suppose that we want to increase the number of words in the memory by using two or more RAM chips. Since every bit added to the address doubles the binary number that can be formed, it is natural to increase the number of words in factors of two. For example, two RAM chips will double the number of words and add one bit to the composite address. Four RAM chips multiply the number of words by four and add two bits to the composite address.

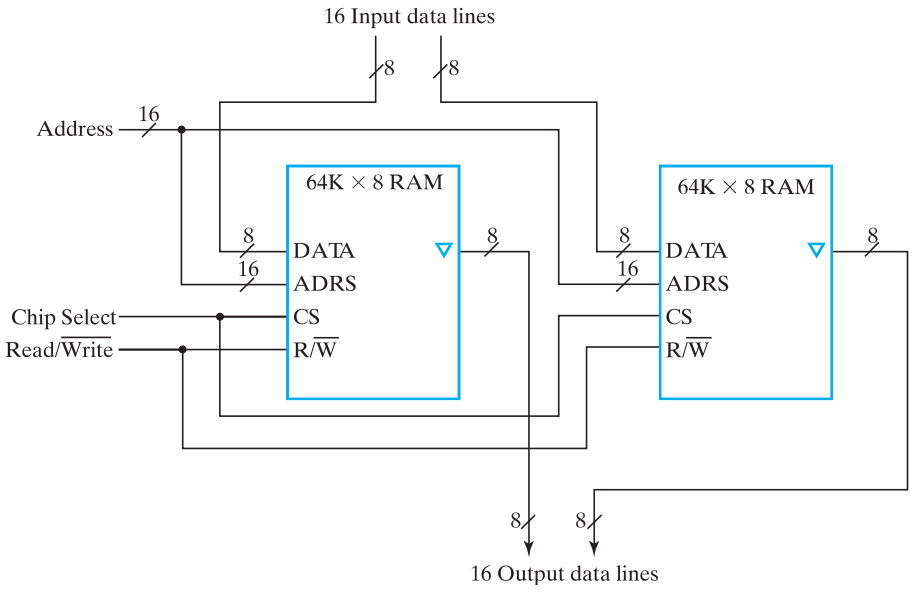
Consider the possibility of constructing a 256K × 8 RAM with four 64K × 8 RAM chips, as shown in Figure 7-10. The eight data input lines go to all the chips. The three-state outputs can be connected together to form the eight common data output lines. This type of output connection is possible only with three-state outputs. Just one Chip Select input will be active at any time, while the other three chips will be disabled. The eight outputs of the selected chip will contain 1s and 0s, and the other three will be in a high-impedance state, presenting only open circuits to the binary output signals of the selected chip.

The 256K-word memory requires an 18-bit address. The 16 least significant bits of the address are applied to the address inputs of all four chips. The two most significant bits are applied to a 2 × 4 decoder. The four outputs of the decoder are applied to the *CS* inputs of the four chips. The memory is disabled when the *EN* input of the decoder, Memory Enable, is equal to 0. All four outputs of the decoder are then 0, and none of the chips is selected. When the decoder is enabled, address bits 17 and 16 determine the particular chip that is selected. If these bits are equal to 00, the first



□ **FIGURE 7-10**
Block Diagram of a 256K × 8 RAM

RAM chip is selected. The remaining 16 address bits then select a word within the chip in the range from 0 to 65,535. The next 65,536 words are selected from the second RAM chip with an 18-bit address that starts with 01 followed by the 16 bits from the common address lines. The address range for each chip is listed in decimal under its symbol in the figure.



□ **FIGURE 7-11**
Block Diagram of a 64K × 16 RAM

It is also possible to combine two chips to form a composite memory containing the same number of words, but with twice as many bits in each word. Figure 7-11 shows the interconnection of two 64K × 8 chips to form a 64K × 16 memory. The 16 data input and data output lines are split between the two chips. Both receive the same 16-bit address and the common *CS* and $R/\overline{W}$ control inputs.

The two techniques just described may be combined to assemble an array of identical chips into a large-capacity memory. The composite memory will have a number of bits per word that is a multiple of that for one chip. The total number of words will increase in factors of two times the word capacity of one chip. An external decoder is needed to select the individual chips based on the additional address bits of the composite memory.

To reduce the number of pins on the chip package, many RAM ICs provide common terminals for the data input and data output. The common terminals are said to be *bidirectional*, which means that for the read operation they act as outputs, and for the write operation they act as inputs. Bidirectional lines are constructed with three-state buffers and are discussed further in Section 6-8. The use of bidirectional signals requires control of the three-state buffers by both Chip Select and Read/$\overline{\text{Write}}$.

7-5 DRAM ICs

Because of its ability to provide high storage capacity at low cost, dynamic RAM (DRAM) dominates the high-capacity memory applications, including the primary RAM in computers. Logically, DRAM in many ways is similar to SRAM. However,

because of the electronic circuit used to implement the storage cell, its electronic design is considerably more challenging. Further, as the name "dynamic" implies, the storage of information is inherently only temporary. As a consequence, the information must be periodically "refreshed" to mimic the behavior of static storage. This need for refresh is the primary logical difference in the behavior of DRAM compared to SRAM. We explore this logical difference by examining the dynamic RAM cell, the logic required to perform the refresh operation, and the impact of the need for refresh on memory system operation.

DRAM Cell

The dynamic RAM cell circuit is shown in Figure 7-12(a). It consists of a capacitor C and a transistor T. The capacitor is used to store electrical charge. If sufficient charge is stored on the capacitor, it can be viewed as storing a logical 1. If insufficient charge is stored on the capacitor, it can be viewed as storing a logical 0. The transistor acts much like a switch, as described in Section 5-1. When the switch is "open," the charge on the capacitor roughly remains fixed—in other words, is stored. But when the switch is "closed," charge can flow into and out of the capacitor from the external Bit (B) line. This charge flow allows the cell to be written with a 1 or 0 and to be read.

In order to understand the read and write operations for the cell, we will use a hydraulic analogy with charge replaced by water, the capacitor by a small storage tank, and the transistor by a valve. Since the bit line has a large capacitance, it is represented by a large tank, and pumps which can fill and empty this tank rapidly. This analogy is given in Figures 7-12(b) and (c) with the valve closed. Note that in one case the small storage tank is full, representing a stored 1, and in the other case it is empty, representing a stored 0. Suppose that a 1 is to be written into the cell. The valve is opened and the pumps fill up the large tank. Water flows through the valve, filling the

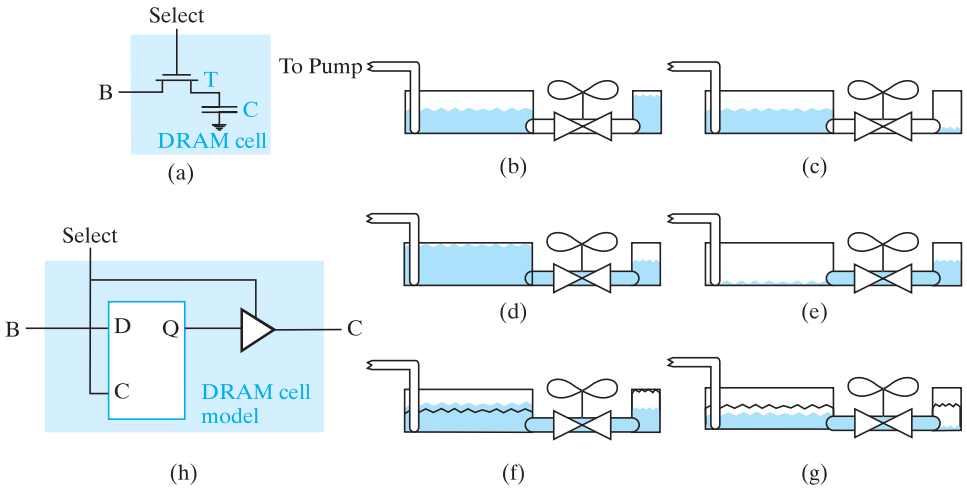


FIGURE 7-12
Dynamic RAM cell, hydraulic analogy of cell operation, and cell model

small storage tank, as shown in Figure 7-12(d). Then the valve is closed, leaving the small tank full, which represents a 1. A 0 can be written using the same sort of operations, except that the pumps empty the large tank as shown in Figure 7-12(e).

Now, suppose we want to read a stored value and that the value is a 1 corresponding to a full storage tank. With the large tank at a known intermediate level, the valve is opened. Since the small storage tank is full, water flows from the small tank to the large tank, increasing the level of the water surface in the large tank slightly as shown in Figure 7-12(f). This increase in level is observed as the reading of 1 from the storage tank. Correspondingly, if the storage tank is initially empty, there will be a slight decrease in the level in the large tank in Figure 7-12(g), which is observed as the reading of a 0 from the storage tank.

In the read operation just described, Figures 7-12(f) and (g) show that, regardless of the initial stored value in the storage tank, it now contains an intermediate value which will not cause enough change in the level of the external tank to permit a 0 or 1 to be observed. So the read operation has destroyed the stored value; this is referred to as a *destructive read*. To be able to read the original stored value in the future, we must *restore* it (i.e., return the storage tank to its original level). To perform the restore for a stored 1 observed, the large tank is filled by the pumps and the small tank fills through the open valve. To perform the restore for a stored 0 observed, the large tank is emptied by the pumps, and the small tank drains through the open valve.

In the actual storage cell, there are other paths present for charge flow. These paths are analogous to small leaks in the storage tank. Due to these leaks, a full small storage tank will eventually drain to a point at which the increase in the level of the large tank on a read cannot be observed as an increase. In fact, if the small tank is less than half full when read, it is possible that a decrease in the level of the large tank may be observed. To compensate for these leaks, the small storage tank storing a 1 must be periodically refilled. This is referred to as a refresh of the cell contents. Every storage cell must be refreshed before its level has declined to a point at which the stored value can no longer be properly observed.

Through the hydraulic analogy, the DRAM operation has been explained. Just as for the SRAM, we employ a logic model for the cell. The model shown in Figure 7-12(h) is a *D* latch. The *C* input to the *D* latch is Select and the *D* input to the *D* latch is *B*. In order to model the output of the DRAM cell, we use a three-state buffer with Select as its control input and *C* as its output. In the original electronic circuit for the DRAM cell in Figure 7-12(a), *B* and *C* are the same signal, but in the logical model they are separate. This is necessary in the modeling process to avoid connecting gate outputs together.

DRAM Bit Slice

Using the logic model for the DRAM cell, we can construct the DRAM bit-slice model shown in Figure 7-13. This model is similar to that for the SRAM bit slice in Figure 7-5. It is apparent that, aside from the cell structure, the two RAM bit slices are logically similar. However, from the standpoint of cost per bit, they are quite different. The DRAM cell consists of a capacitor plus one transistor. The SRAM cell typically contains six transistors, giving a cell complexity roughly three times that of the DRAM. Therefore, the number of SRAM cells in a chip of a given size is less

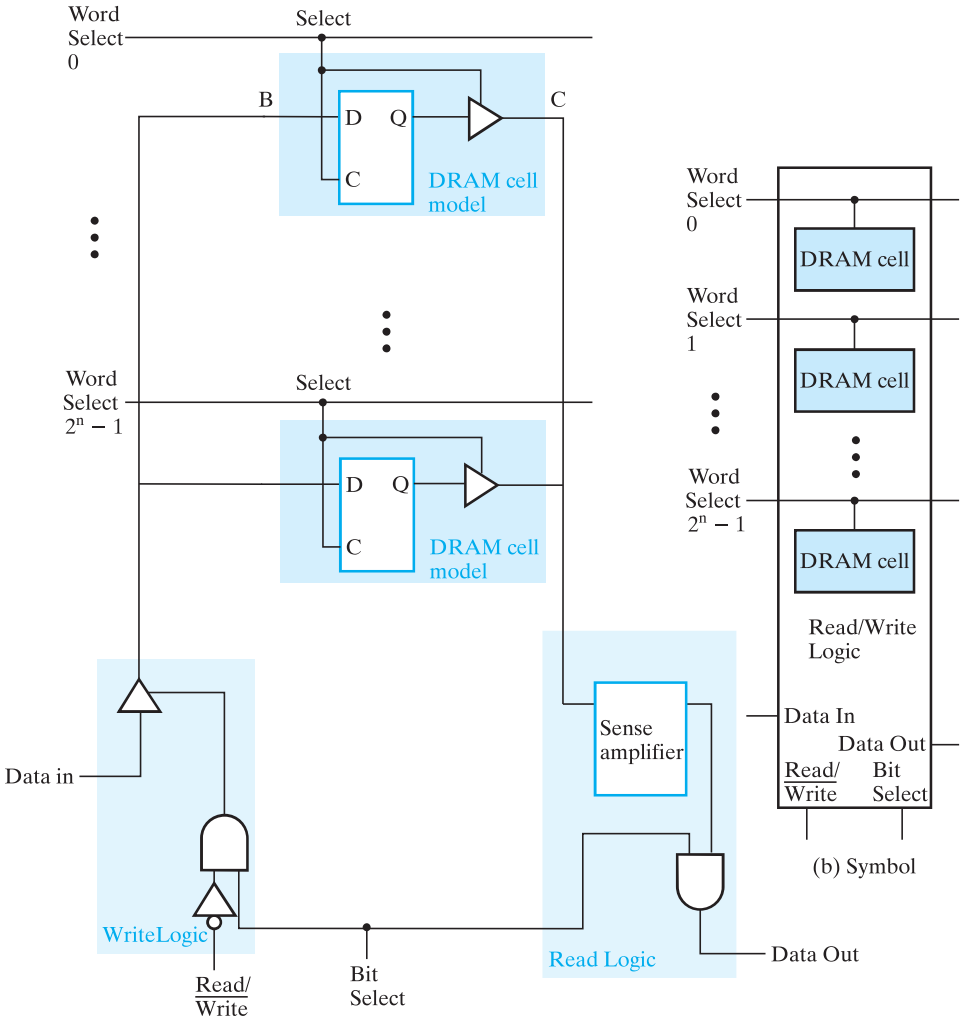


FIGURE 7-13
DRAM Bit-Slice Model

than one-third of those in the DRAM. The DRAM cost per bit is less than one-third the SRAM cost per bit, which justifies the use of DRAM in large memories.

Refresh of the DRAM contents remains to be discussed. But first, we need to develop the typical structure used to handle addressing in DRAMs. Since many DRAM chips are used in a DRAM, we want to reduce the physical size of the DRAM chips. Large DRAMs require 20 or more address bits, which would require 20 address pins on each DRAM chip. To reduce the number of pins, the DRAM address is applied serially in two parts with the row address first and the column address second. This can be done

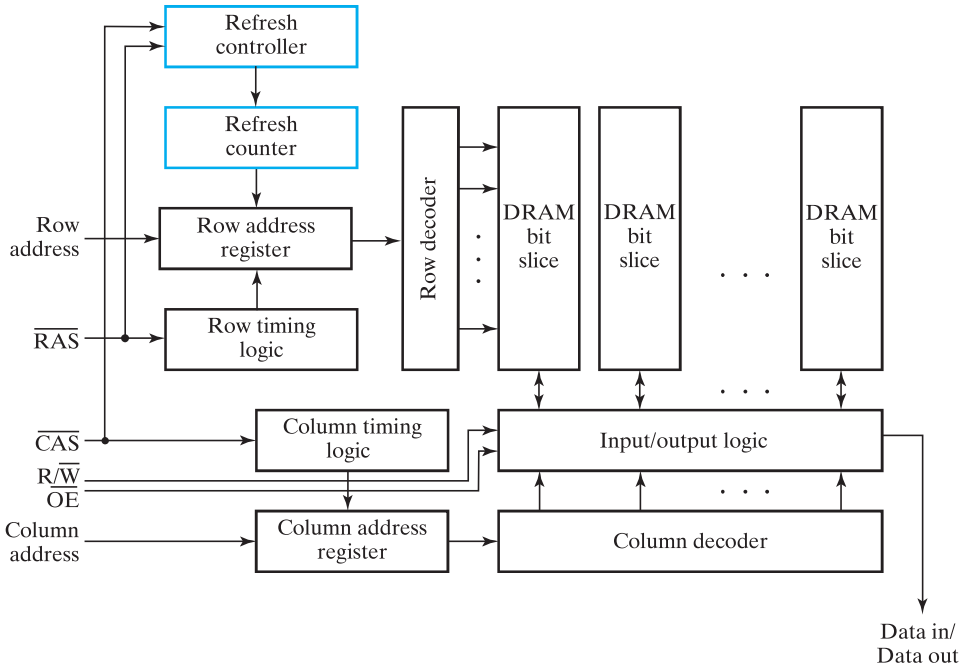


FIGURE 7-14
Block Diagram of a DRAM Including Refresh Logic

since the row address, which performs the row selection, is actually needed before the column address, which reads out the data from the row selected. In order to hold the row address throughout the read or write cycle, it is stored in a register, as shown in Figure 7-14. The column address is also stored in a register. The load signal for the row address register is $\overline{\text{RAS}}$ ($\overline{\text{Row Adress Strobe}}$) and for the column addresses is $\overline{\text{CAS}}$ ($\overline{\text{Column Address Strobe}}$). Note that in addition to $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$, control signals for the DRAM chip include R/$\overline{\text{W}}$ (Read/$\overline{\text{Write}}$) and $\overline{\text{OE}}$ ($\overline{\text{Output enable}}$). Note that this design uses signals active at the LOW (0) level.

The timing for DRAM write and read operation appears in Figure 7-15(a). The row address is applied to the address inputs, and then RAS changes from 1 to 0, loading the row address into the row address register. This address is applied to the row address decoder and selects a row of DRAM cells. Meanwhile, the column address is applied, and then CAS changes from 1 to 0, loading the column address into the column address register. This address is applied to the column address decoder, which selects a set of columns of the RAM array of size equal to the number of RAM data bits. The input data with Read/$\overline{\text{Write}}$ = 0 is applied over a time interval similar to that for the column address. The data bits are applied to the set of bit lines selected by the column address decoder, which in turn apply the values to the DRAM cells in the selected row, writing the new data into the cells. When CAS and RAS return to 1, the write cycle is complete and the DRAM cells store the newly written data. Note that the stored data in all of the other cells in the addressed row has been restored.

The read operation timing shown in Figure 7-15(b) is similar. Timing of the address operations is the same. However, no data is applied and Read/$\overline{\text{Write}}$ is 1 instead of 0. Data values in the DRAM cells in the selected row are applied to the bit

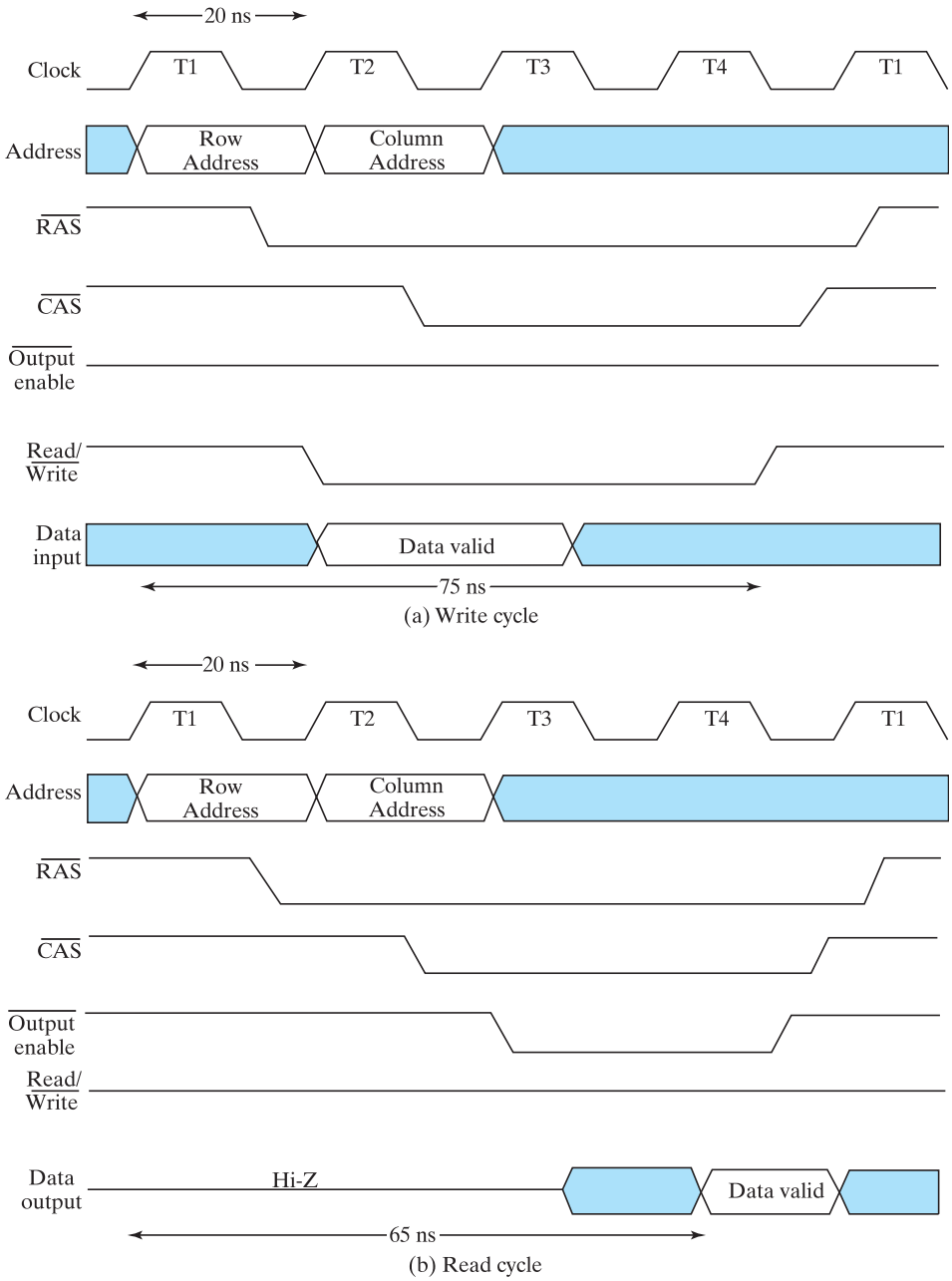


FIGURE 7-15
Timing for DRAM Write and Read Operations

lines and sensed by the sense amplifiers. The column address decoder selects the values to be sent to the Data output, which is enabled by $\overline{\text{Output enable}}$. During the read operation, all values in the addressed row are restored.

To support refresh, additional logic shown in color is present in the block diagram in Figure 7-14. There is a Refresh counter and a Refresh controller. The Refresh counter is used to provide the address of the row of DRAM cells to be refreshed. It is essential for the refresh modes that require the address to be provided from within the DRAM chip. The refresh counter advances on each refresh cycle. Due to the number of bits in the counter, when it reaches $2^n - 1$, where n is the number of rows in the DRAM array, it advances to 0 on the next refresh. The standard ways in which a refresh cycle can be triggered and the corresponding refresh types are as follows:

1. **RAS-only refresh.** A row address is placed on the address lines and RAS is changed to 0. In this case, the refresh addresses must be applied from outside the DRAM chip, typically by an IC called a DRAM controller.
2. **CAS-before-RAS refresh.** The CAS is changed from 1 to 0 followed by a change from 1 to 0 on RAS. Additional refresh cycles can be performed by changing RAS without changing CAS. The refresh addresses for this case come from the refresh counter, which is incremented after the refresh for each cycle.
3. **Hidden refresh.** Following a normal read or write, CAS is left at 0 and RAS is cycled, effectively performing a CAS-before-RAS refresh. During a hidden refresh, the output data from the prior read remains valid. Thus, the refresh is hidden. Unfortunately, the time taken by the hidden refresh is significant, so a subsequent read or write operation is delayed.

In all cases, note that the initiation of a refresh is controlled externally by using the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ signals. Each row of a DRAM chip requires refreshing within a specified maximum refresh time, typically ranging from 16 to 64 milliseconds (ms). Refreshes may be performed at evenly spaced points in the refresh time, an approach called distributed refresh. Alternatively, all refreshes may be performed one after the other, an approach called burst refresh. For example, a 4M $\times$ 4 DRAM has a refresh time of 64 ms and has 4096 rows to be refreshed. The length of time to perform a single refresh is 60 ns, and the refresh interval for distributed refresh is 64 ms/4096 = 15.6 microseconds (μs). A total time out for refresh of 0.25 ms is used out of the 64 ms refresh interval. For the same DRAM, a burst refresh also takes 0.25 ms. The DRAM controller must initiate a refresh every 15.6 μs for distributed refresh and must initiate 4096 refreshes sequentially every 64 ms for burst refresh. During any refresh cycle, no DRAM reads or writes can occur. Since use of burst refresh would halt computer operation for a fairly long period, distributed refresh is more commonly used.

7-6 DRAM Types

Over the last two decades, both capacity and speed of DRAM have increased significantly. The quest for speed has resulted in the evolution of many types of DRAM. Several are listed with brief descriptions in Table 7-2. Of the memory types listed, the first two have largely been replaced in the marketplace by the more advanced SDRAM and RDRAM approaches. Since we have chosen to provide a discussion of

TABLE 7-2
DRAM Types

Type	Abbreviation	Description
Fast page mode DRAM	FPM DRAM	Takes advantage of the fact that, when a row is accessed, all of the row values are available to be read out. By changing the column address, data from different addresses can be read out without reapplying the row address and waiting for the delay associated with reading out the row cells to pass if the row portions of the addresses match.
Extended data output DRAM	EDO DRAM	Extends the length of time that the DRAM holds the data values on its output, permitting the CPU to perform other tasks during the access, since it knows the data will still be available.
Synchronous DRAM	SDRAM	Operates with a clock rather than being asynchronous. This permits a tighter interaction between memory and CPU, since the CPU knows exactly when the data will be available. SDRAM also takes advantage of the row value availability and divides memory into distinct banks, permitting overlapped accesses.
Double-data-rate synchronous DRAM	DDR SDRAM	The same as SDRAM except that data output is provided on both the negative and the positive clock edges.
Rambus® DRAM	RDRAM	A proprietary technology that provides very high memory access rates using a relatively narrow bus.
Error-correcting code	ECC	May be applied to most of the DRAM types above to correct single-bit data errors and often detect double errors.

error-correcting codes (ECC) for memories on the text website, our discussion of memory types here will omit the ECC feature and focus on synchronous DRAM, double-data-rate synchronous DRAM, and Rambus® DRAM. Before considering these, we briefly cover some underlying concepts.

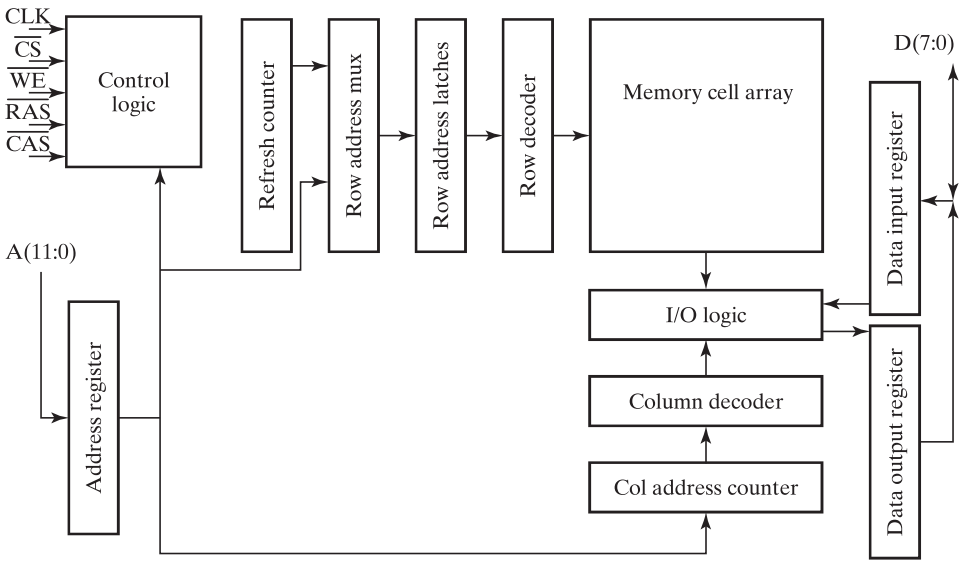
First, all three of these DRAM types work well because of the particular environment in which they operate. In modern high-speed computer systems, the processor

interacts with the DRAM within a memory hierarchy. Most of the instructions and data for the processor are fetched from two lower levels of the hierarchy, the L1 and L2 caches. These are comparatively smaller SRAM-based memory structures that are covered in detail in Chapter 12. For our purposes, the key issue is that most of the reads from the DRAM are not directly from the CPU, but instead are initiated to bring data and instructions into these caches. The reads are in the form of a *line* (i.e., some number of bytes in contiguous addresses in memory) that is brought into the cache. For example, in a given read, the 16 bytes in hexadecimal addresses 000000 through 00000F would be read. This is referred to as a *burst read*. For burst reads, the effective *rate* of reading bytes, which is dependent upon reading bursts from contiguous addresses, rather than the access time is the important measure. With this measure, the three DRAM types we are discussing provide very fast performance.

Second, the effectiveness of these three DRAM types depends upon a very fundamental principle involved in DRAM operation, the reading out of all of the bits in a row for each read operation. The implication of this principle is that all of the bits in a row are available after a read using that row if only they can be accessed. With these two concepts in mind, the synchronous DRAM can be introduced.

Synchronous DRAM (SDRAM)

The use of clocked transfers differentiates SDRAM from conventional DRAM. A block diagram of a 16-megabyte SDRAM IC appears in Figure 7-16. The inputs and outputs differ little from those for the DRAM block diagram in Figure 7-14 with the exception of the presence of the clock for synchronous operation. Internally, there are a number of differences. Since the SDRAM appears synchronous from the outside,



□ **FIGURE 7-16**
Block Diagram of a 16 MB SDRAM

there are synchronous registers on the address inputs and the data inputs and outputs. In addition, a column address counter has been added, which is key to the operation of the SDRAM. While the control logic may appear to be similar, the control in this case is much more complex, since the SDRAM has a mode control word that can be loaded from the address bus. Considering a 16 MB memory, the memory array contains 134,217,728 bits and is almost square, with 8192 rows and 16,384 columns. There are 13 row address bits. Since there are 8 bits per byte, the number of column addresses is 16,384 divided by 8, which equals 2048. This requires 11 column address bits. Note that 13 plus 11 equals 24, giving the correct number of bits to address 16 MB.

As with the regular DRAM, the SDRAM applies the row address first, followed by the column address. The timing, however, is somewhat different, and some new terminology is used. Before performing an actual read operation from a specified column address, the entire row of 2048 bytes specified by the applied row address is read out internally and stored in the I/O logic. Internally, this step takes a few clock cycles. Next, the actual read step is performed with the column address applied. After an additional delay of a few clock cycles, the data bytes begin appearing on the output, one per clock period. The number of bytes that appear, the burst length, has been set by loading a mode control word into the control logic from the address input.

The timing of a burst read cycle with burst length equal to four is shown in Figure 7-17. The read begins with the application of the row address and the row address strobe (RAS), which causes the row address to be captured in the address register and the reading of the row to be initiated. During the next two clock periods,

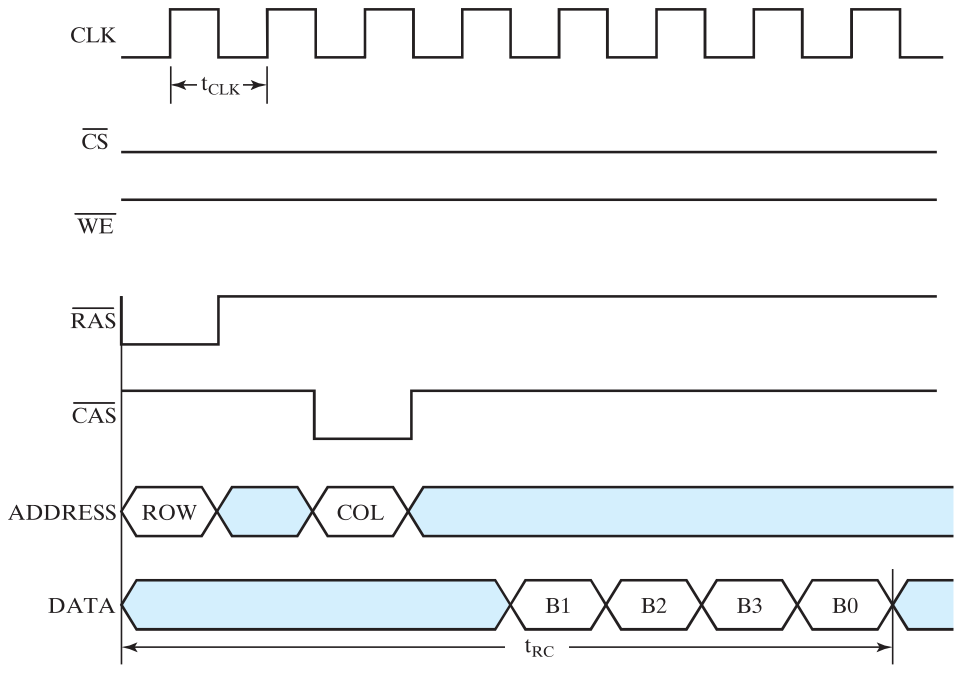


FIGURE 7-17
Timing Diagram for an SDRAM

the reading of the row is taking place. During the third clock period, the column address and the column address strobe are applied, with the column address captured in the address register and the reading of the first data byte initiated. The data byte is then available to be read from the SDRAM at the positive clock edge occurring two cycles later. The second, third, and fourth bytes are available for reading on subsequent clock edges. In Figure 7-17, note that the bytes are presented in the order 1, 2, 3, 0. This is because, in the column address identifying the byte immediately needed by the CPU, the last two bits are 01. The subsequent bytes appear in the order of these two bits counted up modulo (burst length) by the column address counter, giving addresses ending in 01, 10, 11, and 00, with all other address bits fixed.

It is interesting to compare the byte rate for reading bytes from SDRAM to that of the basic DRAM. We assume that the read cycle time t_{RC} for the basic DRAM is 60 ns and that the clock period t_{CLK} for the SDRAM is 7.5 ns. The byte rate for the basic DRAM is one byte per 60 ns, or 16.67 MB/s. For the SDRAM, from Figure 7-17, it requires 8.0 clock cycles, or 60 ns, to read four bytes, giving a byte rate of 66.67 MB/s. If the burst is eight instead of four bytes, a read cycle time of 90 ns is required, giving a byte rate of 88.89 MB/s. Finally, if the burst is the entire 2048-byte row of the SDRAM, the read cycle time becomes $60 + (2048 - 4) \times 7.5 = 15{,}390$ ns, giving a byte rate of 133.07 MB, which approaches the limit of one byte per 7.5 ns clock period.

Double-Data-Rate SDRAM (DDR SDRAM)

The second DRAM type, double-data-rate SDRAM (DDR SDRAM) overcomes the preceding limit without decreasing the clock period. Instead, it provides two bytes of data per clock period by using both the positive and negative clock edges. In Figure 7-17, four bytes are read, one per positive clock edge. By using both clock edges, eight bytes can be transferred in the same read cycle time t_{RC}. For a 7.5 ns clock period, the byte rate limit doubles in the example to 266.14 MB/s.

Additional basic techniques can be applied to further increase the byte rate. For example, instead of having single byte data, an SDRAM IC can have the data I/O length of four bytes (32 bits). This gives a byte rate limit of 1.065 GB/s with a 7.5 ns clock period. Eight bytes give a byte rate limit of 2.130 GB/s.

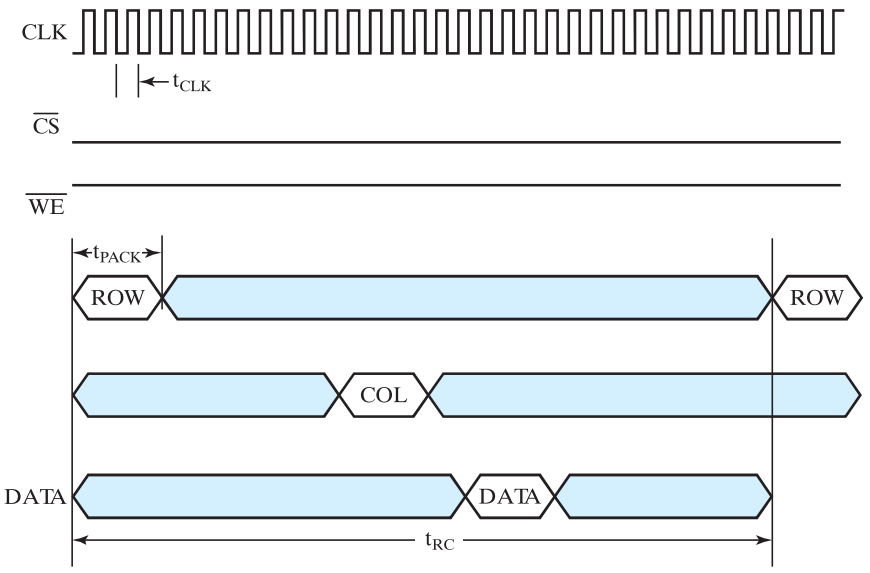
The byte rates achieved in the examples are upper limits. If the actual accesses needed are to different rows of the RAM, the delay from the application of the RAS pulse to read out the first byte of data is significant and leads to performance well below the limit. This can be partially offset by breaking up the memory into multiple banks, where each bank performs the row read independently. Provided that the row and bank addresses are available early enough, row reads can be performed on one or more banks while data is still being transferred from the currently active row. When the column reads from the currently active row are complete, data can potentially be available immediately from other banks, permitting an uninterrupted flow of data from the memory. This permits the actual read rate to more closely approach the limit. Nevertheless, due to the fact that multiple row accesses to the same bank may occur in sequence, the maximum rate is not reached.

More recent versions of DDR memory include DDR2 and DDR3, with DDR4 expected to become available in 2014. While none of the versions of DDR are compatible with each other due to differences in electrical properties and timing, all of

the versions depend on the same principle of transferring data on both the rising and the falling edges of the memory bus clock. Each succeeding version has increased the number of data transfers per memory bus clock cycle.

RAMBUS® DRAM (RDRAM)

The final DRAM type to be discussed is RAMBUS DRAM (RDRAM). Although no longer widely available, we include a description of RDRAM to illustrate its alternative design for the memory interface. RDRAM ICs are designed to be integrated into a memory system that uses a packet-based bus for the interaction between the RDRAM ICs and the memory bus to the processor. The primary components of the bus are a 3-bit path for the row address, a 5-bit path for the column address, and a 16-bit or 18-bit path for data. The bus is synchronous and performs transfers on both clock edges, the same property possessed by the DDR SDRAM. Information on the three paths mentioned above is transferred in packets that are four clock cycles long, which means that there are eight transfers/packet. The number of bits per packet for each of the paths is 24 bits for the row address packet, 40 bits for the column address packet, and 128 bits or 144 bits for the data packet. The larger data packet includes 16 parity bits for implementing an error-correcting code. The RDRAM IC employs the concept of multiple memory banks mentioned earlier to provide capability for concurrent memory accesses with different row addresses. RDRAM uses the usual row-activate technique in which the addressed row data of the memory is read. From this row data, the column address is used to select byte pairs in the order in which they are to be transmitted in the packet. A typical timing picture for an RDRAM read access is shown in Figure 7-18. Due to the sophisticated electronic design of the RAMBUS



□ **FIGURE 7-18**
Timing of a 16 MB RDRAM

system, we can consider a clock period of 1.875 ns. Thus, the time for transmission of a packet is $t_{PACK} = 4 \times 1.875 = 7.5$ ns. The cycle time for accessing a single data packet of 8 byte pairs or 16 bytes is 32 clock cycles or 60 ns, as shown in Figure 7-18. The corresponding byte rate is 266.67 MB/s. If four of the byte packets are accessed from the same row, the rate increases to 1.067 GB/s. By reading an entire RDRAM row of 2048 bytes, the cycle time increases to $60 + (2048 - 64) \times 1.875/4 = 990$ ns or a byte rate limit of $2048/(990 \times 10^{-9}) = 2.069$ MB/s, approaching the ideal limit of 4/1.875 ns or 2.133 GB/s.

7-7 Arrays of Dynamic RAM ICs

Many of the same design principles used for SRAM arrays in Section 7-4 apply to DRAM arrays. There are, however, a number of different requirements for the control and addressing of DRAM arrays. These requirements are typically handled by a *DRAM controller*, which performs the following functions:

1. controlling separation of the address into a row address and a column address, and providing these addresses at the required times,
2. providing the $\overline{RAS}$ and $\overline{CAS}$ signals at the required times for read, write, and refresh operations,
3. performing refresh operations at the necessary intervals, and
4. providing status signals to the rest of the system (e.g., indicating whether the memory is busy performing refresh).

The DRAM controller is a complex synchronous sequential circuit with the external CPU clock providing synchronization of its operation.

7-8 Chapter Summary

Memory is of two types: random-access memory (RAM) and read-only memory (ROM). For both types, we apply an address to read from or write into a data word. Read and write operations have specific steps and associated timing parameters, including access time and write cycle time. Memory can be static or dynamic and volatile or nonvolatile. Internally, a RAM chip consists of an array of RAM cells, decoders, write circuits, read circuits, and output circuits. A combination of a write circuit, read circuit, and the associated RAM cells can be logically modeled as a RAM bit slice. RAM bit slices, in turn, can be combined to form two-dimensional RAM cell arrays, which, with decoders and output circuits added, form the basis for a RAM chip. Output circuits use three-state buffers in order to facilitate connecting together an array of RAM chips without significant additional logic. Due to the need for refresh, additional circuitry is required within DRAMs, as well as in arrays of DRAM chips. In a quest for faster memory access, a number of new DRAM types have been developed. The most recent forms of these high-speed DRAMs employ a synchronous interface that uses a clock to control memory accesses.