CHAPTER 12

MEMORY DEVICES

OUTLINE

- 12-1 Memory Terminology
- 12-2 General Memory Operation
- 12-3 CPU–Memory Connections
- 12-4 Read-Only Memories
- 12-5 ROM Architecture
- 12-6 ROM Timing
- 12-7 Types of ROMs
- 12-8 Flash Memory
- 12-9 ROM Applications
- 12-10 Semiconductor RAM
- 12-11 RAM Architecture
- 12-12 Static RAM (SRAM)

- 12-13 Dynamic RAM (DRAM)
- 12-14 Dynamic RAM Structure and Operation
- 12-15 DRAM Read/Write Cycles
- 12-16 DRAM Refreshing
- 12-17 DRAM Technology
- 12-18 Expanding Word Size and Capacity
- 12-19 Special Memory Functions
- 12-20 Troubleshooting RAM Systems
- 12-21 Testing ROM

OBJECTIVES

Upon completion of this chapter, you will be able to:

- Understand and correctly use the terminology associated with memory systems.
- Describe the difference between read/write memory and read-only memory.
- Discuss the difference between volatile and nonvolatile memory.
- Determine the capacity of a memory device from its inputs and outputs.
- Outline the steps that occur when the CPU reads from or writes to memory.
- Distinguish among the various types of ROMs and cite some common applications.
- Understand and describe the organization and operation of static and dynamic RAMs.
- Compare the relative advantages and disadvantages of EPROM, EEPROM, and flash memory.
- Combine memory ICs to form memory modules with larger word size and/or capacity.
- Use the test results on a RAM or ROM system to determine possible faults in the memory system.

INTRODUCTION

A major advantage of digital over analog systems is the ability to store easily large quantities of digital information and data for short or long periods. This memory capability is what makes digital systems so versatile and adaptable to many situations. For example, in a digital computer, the internal main memory stores instructions that tell the computer what to do under *all* possible circumstances so that the computer will do its job with a minimum amount of human intervention.

This chapter is devoted to a study of the most commonly used types of memory devices and systems. We have already become very familiar with the flip-flop, which is an electronic memory device. We have also seen how groups of FFs called *registers* can be used to store information and how this information can be transferred to other locations. FF registers are high-speed memory elements that are used extensively in the internal operations of a digital computer, where digital information is continually being moved from one location to another. Advances in LSI and VLSI technology have made it possible to obtain large numbers of FFs on a single chip arranged in various memory-array formats. These bipolar and MOS semiconductor memories are the fastest memory devices available, and their cost has been continuously decreasing as LSI technology improves.

Digital data can also be stored as charges on capacitors, and a very important type of semiconductor memory uses this principle to obtain highdensity storage at low power-requirement levels.

Semiconductor memories are used as the **main memory** of a computer (Figure 12-1), where fast operation is important. A computer's main memory—also called its *working memory*—is in constant communication with the central processing unit (CPU) as a program of instructions is being executed. A program and any data used by the program reside in the main memory while the computer is working on that program. RAM and ROM (to be defined shortly) make up main memory.

Another form of storage in a computer is performed by **auxiliary memory** (Figure 12-1), which is separate from the main working memory. Auxiliary memory—also called *mass storage*—has the capacity to store massive amounts of data without the need for electrical power. Auxiliary memory operates at a much slower speed than main memory, and it stores programs and data that are not currently being used by the CPU. This information is transferred to the main memory when the computer needs it. Common auxiliary memory devices are magnetic disk and compact disk (CD).

We will take a detailed look at the characteristics of the most common memory devices used as the internal memory of a computer. First, we define some of the common terms used in memory systems.

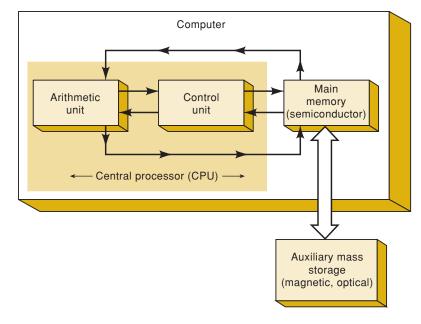


FIGURE 12-1 A computer system normally uses high-speed main memory and slower external auxiliary memory.

12-1 MEMORY TERMINOLOGY

The study of memory devices and systems is filled with terminology that can sometimes be overwhelming to a student. Before we get into any comprehensive discussion of memories, it would be helpful if you had the meaning of some of the more basic terms under your belt. Other new terms will be defined as they appear in the chapter.

- Memory Cell. A device or an electrical circuit used to store a single bit (0 or 1). Examples of memory cells include a flip-flop, a charged capacitor, and a single spot on magnetic tape or disk.
- Memory Word. A group of bits (cells) in a memory that represents instructions or data of some type. For example, a register consisting of eight FFs can be considered to be a memory that is storing an eight-bit word. Word sizes in modern computers typically range from 8 to 64 bits, depending on the size of the computer.
- Byte. A special term used for a group of eight bits. A byte always consists of eight bits. Word sizes can be expressed in bytes as well as in bits. For example, a word size of eight bits is also a word size of one byte, a word size of 16 bits is two bytes, and so on.
- **Capacity.** A way of specifying how many bits can be stored in a particular memory device or complete memory system. To illustrate, suppose that we have a memory that can store 4096 20-bit words. This represents a total capacity of 81,920 bits. We could also express this memory's capacity as 4096 \times 20. When expressed this way, the first number (4096) is the number of words, and the second number (20) is the number of bits per word (word size). The number of words in a memory is often a multiple of 1024. It is common to use the designation "1K" to represent 1024 = 2^{10} when referring to memory capacity. Thus, a memory that has a storage capacity of $4K \times 20$ is actually a 4096 \times 20 memory. The development of larger memories has brought about the designation "1M" or "1 meg" to represent $2^{20} = 1,048,576$. Thus, a memory that has a capacity of $2M \times 8$ is actually one with a capacity of $2,097,152 \times 8$. The designation "giga" refers to $2^{30} = 1,073,741,824$.

EXAMPLE 12-1A

A certain semiconductor memory chip is specified as $2 \text{ K} \times 8$. How many words can be stored on this chip? What is the word size? How many total bits can this chip store?

Solution

 $2K = 2 \times 1024 = 2048$ words

Each word is eight bits (one byte). The total number of bits is therefore

 $2048 \times 8 = 16,384$ bits

EXAMPLE 12-1B

Which memory stores the most bits: a $5M \times 8$ memory or a memory that stores 1M words at a word size of 16 bits?

Solution

 $5M \times 8 = 5 \times 1,048,576 \times 8 = 41,943,040$ bits $1M \times 16 = 1,048,576 \times 16 = 16,777,216$ bits

The 5M \times 8 memory stores more bits.

- Density. Another term for *capacity*. When we say that one memory device has a greater density than another, we mean that it can store more bits in the same amount of space. It is more dense.
- Address. A number that identifies the location of a word in memory. Each word stored in a memory device or system has a unique address. Addresses always exist in a digital system as a binary number, although octal, hexadecimal, and decimal numbers are often used to represent the address for convenience. Figure 12-2 illustrates a small memory consisting of eight words. Each of these eight words has a specific address represented as a three-bit number ranging from 000 to 111. Whenever we refer to a specific word location in memory, we use its address code to identify it.
- Read Operation. The operation whereby the binary word stored in a specific memory location (address) is sensed and then transferred to another device. For example, if we want to use word 4 of the memory of Figure 12-2 for some purpose, we must perform a read operation on address 100. The read operation is often called a *fetch* operation because a word is being fetched from memory. We will use both terms interchangeably.
- Write Operation. The operation whereby a new word is placed into a particular memory location. It is also referred to as a *store* operation. Whenever a new word is written into a memory location, it replaces the word that was previously stored there.
- Access Time. A measure of a memory device's operating speed. It is the amount of time required to perform a read operation. More specifically, it is the time between the memory receiving a new address input and the data becoming available at the memory output. The symbol t_{ACC} is used for access time.
- Volatile Memory. Any type of memory that requires the application of electrical power in order to store information. If the electrical power is removed, all information stored in the memory will be lost. Many semiconductor memories are volatile, while all magnetic memories are *non-volatile*, which means that they can store information without electrical power.
- Random-Access Memory (RAM). Memory in which the actual physical location of a memory word has no effect on how long it takes to read

FIGURE 12-2 Each word location has a specific binary address.

4	Addresse	s
	000	Word 0
	001	Word 1
	010	Word 2
	011	Word 3
	100	Word 4
	101	Word 5
	110	Word 6
	111	Word 7

from or write into that location. In other words, the access time is the same for any address in memory. Most semiconductor memories are RAMs.

- Sequential-Access Memory (SAM). A type of memory in which the access time is not constant but varies depending on the address location. A particular stored word is found by sequencing through all address locations until the desired address is reached. This produces access times that are much longer than those of random-access memories. An example of a sequential-access memory device is a magnetic tape backup. To illustrate the difference between SAM and RAM, consider the situation where you have recorded 60 minutes of songs on an audio tape cassette. When you want to get to a particular song, you have to rewind or fastforward the tape until you find it. The process is relatively slow, and the amount of time required depends on where on the tape the desired song is recorded. This is SAM because you have to sequence through all intervening information until you find what you are looking for. The RAM counterpart to this would be an audio CD, where you can quickly select any song by punching in the appropriate code, and it takes approximately the same time no matter what song you select. Sequential-access memories are used where the data to be accessed will always come in a long sequence of successive words. Video memory, for example, must output its contents in the same order over and over again to keep the image refreshed on the CRT screen.
- **Read/Write Memory (RWM).** Any memory that can be read from or written into with equal ease.
- Read-Only Memory (ROM). A broad class of semiconductor memories designed for applications where the ratio of read operations to write operations is very high. Technically, a ROM can be written into (programmed) only once, and this operation is normally performed at the factory. Thereafter, information can only be read from the memory. Other types of ROM are actually read-mostly memories (RMM), which can be written into more than once; but the write operation is more complicated than the read operation, and it is not performed very often. The various types of ROM will be discussed later. *All ROM is nonvolatile* and will store data when electrical power is removed.
- Static Memory Devices. Semiconductor memory devices in which the stored data will remain permanently stored as long as power is applied, without the need for periodically rewriting the data into memory.
- Dynamic Memory Devices. Semiconductor memory devices in which the stored data will *not* remain permanently stored, even with power applied, unless the data are periodically rewritten into memory. The latter operation is called a *refresh* operation.
- Main Memory. Also referred to as the computer's working memory. It stores instructions and data the CPU is currently working on. It is the highest-speed memory in the computer and is always a semiconductor memory.
- Auxiliary Memory. Also referred to as mass storage because it stores massive amounts of information external to the main memory. It is slower in speed than main memory and is always nonvolatile. Magnetic disks and CDs are common auxiliary memory devices.

 Define the following terms. (a) Memory cell (b) Memory word (c) Address (d) Byte
(b) Memory word(c) Address
(c) Address
(d) Byte
(e) Access time
2. A certain memory has a capacity of $8K \times 16$. How many bits are in each word? How many words are being stored? How many memory cells does this memory contain?
3. Explain the difference between the read (fetch) and write (store) opera- tions.
4. <i>True or false:</i> A volatile memory will lose its stored data when electrical power is interrupted.
5. Explain the difference between SAM and RAM.
6. Explain the difference between RWM and ROM.
7. <i>True or false:</i> A dynamic memory will hold its data as long as electrical power is applied.

Although each type of memory is different in its internal operation, certain basic operating principles are the same for all memory systems. An understanding of these basic ideas will help in our study of individual memory devices.

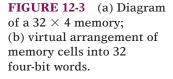
Every memory system requires several different types of input and output lines to perform the following functions:

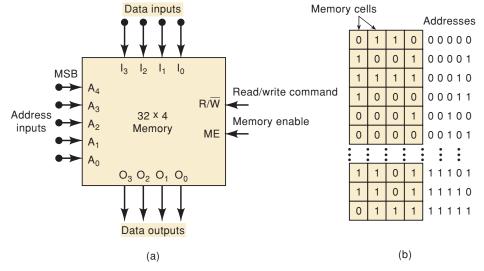
- 1. Select the address in memory that is being accessed for a read or write operation.
- 2. Select either a read or a write operation to be performed.
- 3. Supply the input data to be stored in memory during a write operation.
- 4. Hold the output data coming from memory during a read operation.
- 5. Enable (or disable) the memory so that it will (or will not) respond to the address inputs and read/write command.

Figure 12-3(a) illustrates these basic functions in a simplified diagram of a 32 \times 4 memory that stores 32 four-bit words. Because the word size is four bits, there are four data input lines I_0 to I_3 and four data output lines O_0 to O_3 . During a write operation, the data to be stored into memory must be applied to the data input lines. During a read operation, the word being read from memory appears at the data output lines.

Address Inputs

Because this memory stores 32 words, it has 32 different storage locations and therefore 32 different binary addresses ranging from 00000 to 11111 (0 to 31 in decimal). Thus, there are five address inputs, A_0 to A_4 . To access one





of the memory locations for a read or a write operation, the five-bit address code for that particular location is applied to the address inputs. In general, N address inputs are required for a memory that has a capacity of 2^N words.

We can visualize the memory of Figure 12-3(a) as an arrangement of 32 registers, with each register holding a four-bit word, as illustrated in Figure 12-3(b). Each address location is shown containing four memory cells that hold 1s and 0s that make up the data word stored at that location. For example, the data word 0110 is stored at address 00000, the data word 1001 is stored at address 00001, and so on.

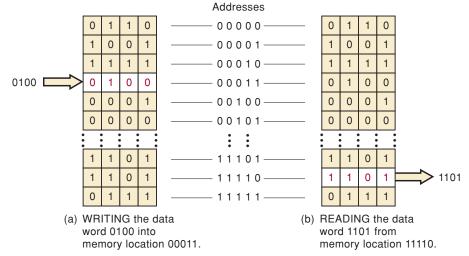
The R/\overline{W} Input

This input controls which memory operation is to take place: read (R) or write (W). The input is labeled R/\overline{W} ; there is no bar over the R, which indicates that the read operation occurs when $R/\overline{W} = 1$. The bar over the W indicates that the write operation takes place when $R/\overline{W} = 0$. Other labels are often used for this input. Two of the more common ones are \overline{W} (write) and \overline{WE} (write enable). Again, the bar indicates that the write operation occurs when the input is LOW. It is understood that the read operation occurs for a HIGH.

A simplified illustration of the read and write operations is shown in Figure 12-4. Figure 12-4(a) shows the data word 0100 being written into the memory register at address location 00011. This data word would have been applied to the memory's data input lines, and it replaces the data previously stored at address 00011. Figure 12-4(b) shows the data word 1101 being read from address 11110. This data word would appear at the memory's data output lines. After the read operation, the data word 1101 is still stored in address 11110. In other words, the read operation does not change the stored data.

Memory Enable

Many memory systems have some means for completely disabling all or part of the memory so that it will not respond to the other inputs. This is represented in Figure 12-3 as the MEMORY ENABLE input, although it can have different names in the various memory systems, such as chip enable (*CE*) or chip select (*CS*). Here, it is shown as an active-HIGH input that enables the memory to operate normally when it is kept HIGH. A LOW on this input disables the memory so that it will not respond to the address and R/\overline{W} inputs. **FIGURE 12-4** Simplified illustration of the read and write operations on the 32×4 memory: (a) writing the data word 0100 into memory location 00011; (b) reading the data word 1101 from memory location 11110.



This type of input is useful when several memory modules are combined to form a larger memory. We will examine this idea later.

EXAMPLE 12-2 Describe the conditions at each input and output when the contents of address location 00100 are to be read. **Solution** Address inputs: 00100 Data inputs: xxxx (not used) R/\overline{W} : HIGH MEMORY ENABLE: HIGH Data outputs: 0001 **EXAMPLE 12-3** Describe the conditions at each input and output when the data word 1110 is to be written into address location 01101. **Solution** Address inputs: 01101 Data inputs: 1110 R/\overline{W} : LOW MEMORY ENABLE: HIGH Data outputs: xxxx (not used; usually Hi-Z) **EXAMPLE 12-4** A certain memory has a capacity of $4K \times 8$. (a) How many data input and data output lines does it have? (b) How many address lines does it have? (c) What is its capacity in bytes?

Solution

- (a) Eight of each because the word size is eight.
- (b) The memory stores $4K = 4 \times 1024 = 4096$ words. Thus, there are 4096 memory addresses. Because $4096 = 2^{12}$, it requires a 12-bit address code to specify one of 4096 addresses.
- (c) A byte is eight bits. This memory has a capacity of 4096 bytes.

The example memory in Figure 12-3 illustrates the important input and output functions common to most memory systems. Of course, each type of memory may have other input and output lines that are peculiar to that memory. These will be described as we discuss the individual memory types.

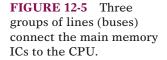
REVIEW QUESTIONS

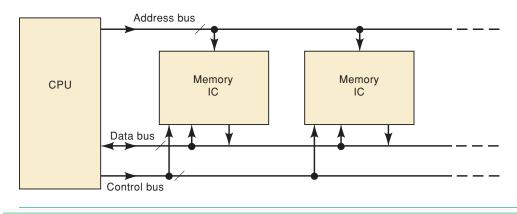
- 1. How many address inputs, data inputs, and data outputs are required for a 16K \times 12 memory?
- 2. What is the function of the R/\overline{W} input?
- 3. What is the function of the MEMORY ENABLE input?

12-3 CPU–MEMORY CONNECTIONS

A major part of this chapter is devoted to semiconductor memory, which, as pointed out earlier, makes up the main memory of most modern computers. Remember, this main memory is in constant communication with the central processing unit (CPU). It is not necessary to be familiar with the detailed operation of a CPU at this point, and so the following simplified treatment of the CPU-memory interface will provide the perspective needed to make our study of memory devices more meaningful.

A computer's main memory is made up of RAM and ROM ICs that are interfaced to the CPU over three groups of signal lines or buses. These are shown in Figure 12-5 as the address lines or address bus, the data lines or data bus, and the control lines or control bus. Each of these buses consists of several lines (note that they are represented by a single line with a slash), and the number of lines in each bus will vary from one computer to the next. The three buses play a necessary part in allowing the CPU to write data into memory and to read data from memory.





When a computer is executing a program of instructions, the CPU continually fetches (reads) information from those locations in memory that contain (1) the program codes representing the operations to be performed and (2) the data to be operated upon. The CPU will also store (write) data into memory locations as dictated by the program instructions. Whenever the CPU wants to write data to a particular memory location, the following steps must occur:

Write Operation

- 1. The CPU supplies the binary address of the memory location where the data are to be stored. It places this address on the address bus lines.
- 2. The CPU places the data to be stored on the data bus lines.
- 3. The CPU activates the appropriate control signal lines for the memory write operation.
- 4. The memory ICs decode the binary address to determine which location is being selected for the store operation.
- 5. The data on the data bus are transferred to the selected memory location.

Whenever the CPU wants to read data from a specific memory location, the following steps must occur:

Read Operation

- 1. The CPU supplies the binary address of the memory location from which data are to be retrieved. It places this address on the address bus lines.
- 2. The CPU activates the appropriate control signal lines for the memory read operation.
- 3. The memory ICs decode the binary address to determine which location is being selected for the read operation.
- 4. The memory ICs place data from the selected memory location onto the data bus, from which they are transferred to the CPU.

The steps above should make clear the function of each of the system buses:

- Address Bus. This *unidirectional* bus carries the binary address outputs from the CPU to the memory ICs to select one memory location.
- Data Bus. This *bidirectional* bus carries data between the CPU and the memory ICs.
- **Control Bus.** This bus carries control signals (such as the R/\overline{W} signal) from the CPU to the memory ICs.

As we get into discussions of actual memory ICs, we will examine the signal activity that appears on these buses for the read and write operations.

REVIEW QUESTIONS

- 1. Name the three groups of lines that connect the CPU and the internal memory.
- 2. Outline the steps that take place when the CPU reads from memory.
- 3. Outline the steps that occur when the CPU writes to memory.

12-4 READ-ONLY MEMORIES

The read-only memory is a type of semiconductor memory designed to hold data that either are permanent or will not change frequently. During normal operation, no new data can be written into a ROM, but data can be read from ROM. For some ROMs, the data that are stored must be built-in during the manufacturing process; for other ROMs, the data can be entered electrically. The process of entering data is called **programming** or *burning-in* the ROM. Some ROMs cannot have their data changed once they have been programmed; others can be *erased* and reprogrammed as often as desired. We will take a detailed look later at these various types of ROMs. For now, we will assume that the ROMs have been programmed and are holding data.

ROMs are used to store data and information that are not to change during the normal operation of a system. A major use for ROMs is in the storage of programs in microcomputers. Because all ROMs are *nonvolatile*, these programs are not lost when electrical power is turned off. When the microcomputer is turned on, it can immediately begin executing the program stored in ROM. ROMs are also used for program and data storage in microprocessor-controlled equipment such as electronic cash registers, appliances, and security systems.

ROM Block Diagram

A typical block diagram for a ROM is shown in Figure 12-6(a). It has three sets of signals: address inputs, control input(s), and data outputs. From our previous discussions, we can determine that this ROM is storing 16 words because it has $2^4 = 16$ possible addresses, and each word contains eight bits because there are eight data outputs. Thus, this is a 16×8 ROM. Another way to describe this ROM's capacity is to say that it stores 16 bytes of data.

The data outputs of most ROM ICs are tristate outputs, to permit the connection of many ROM chips to the same data bus for memory expansion. The most common numbers of data outputs for ROMs are four, eight, and 16 bits, with eight-bit words being the most common.

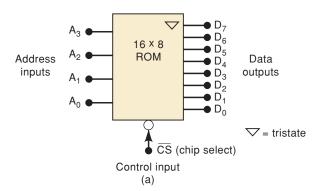
The control input CS stands for **chip select**. This is essentially an enable input that enables or disables the ROM outputs. Some manufacturers use different labels for the control input, such as CE (chip enable) or OE (output enable). Many ROMs have two or more control inputs that must be active in order to enable the data outputs so that data can be read from the selected address. In some ROM ICs, one of the control inputs (usually the CE) is used to place the ROM in a low-power standby mode when it is not being used. This reduces the current drain from the system power supply.

The *CS* input shown in Figure 12-6(a) is active-LOW; therefore, it must be in the LOW state to enable the ROM data to appear at the data outputs. Notice that there is no R/\overline{W} (read/write) input because the ROM cannot be written into during normal operation.

The Read Operation

Let's assume that the ROM has been programmed with the data shown in the table of Figure 12-6(b). Sixteen different data words are stored at the 16 different address locations. For example, the data word stored at location 0011 is 10101111. Of course, the data are stored in binary inside the ROM, but very often we use hexadecimal notation to show the programmed data efficiently. This is done in Figure 12-6(c).

In order to read a data word from ROM, we need to do two things: (1) apply the appropriate address inputs and then (2) activate the control inputs. For



		Add	ress					Da	ta								Address	Data
Word	A_3	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				Word	$A_3 A_2 A_1 A_0$	D ₇ -D ₀
0	0	0	0	0	1	1	0	1	1	1	1	0				0	0	DE
1	0	0	0	1	0	0	1	1	1	0	1	0				1	1	3A
2	0	0	1	0	1	0	0	0	0	1	0	1				2	2	85
3	0	0	1	1	1	0	1	0	1	1	1	1				3	3	AF
4	0	1	0	0	0	0	0	1	1	0	0	1				4	4	19
5	0	1	0	1	0	1	1	1	1	0	1	1				5	5	7B
6	0	1	1	0	0	0	0	0	0	0	0	0				6	6	00
7	0	1	1	1	1	1	1	0	1	1	0	1				7	7	ED
		_	_	-	-	-					-	_	-			8	8	3C
8	1	0	0	0	0	0	1	1	1	1	0	0				9	9	FF
9	1	0	0	1		1	1	1	1	1	1	1				10	Ă	B8
10 11	1	0 0	1	0		0	0	0	0	0	0	0				11	В	C7
12	1	1	0	0	0	0	1	0	0	1	1	1				12	C	27
13	1	1	0	1	0	1	1	0	1	0	1	0				13	D	6A
14	1	1	1	0	1	1	0	1	0	0	1	0				14	E	D2
15	1	1	1	1	0	1	0	1	1	0	1	1				15	F	5B
(b)							-		-		(c)	1						

FIGURE 12-6 (a) Typical ROM block symbol; (b) table showing binary data at each address location; (c) the same table in hex.

example, if we want to read the data stored at location 0111 of the ROM in Figure 12-6, we must apply $A_3A_2A_1A_0 = 0111$ to the address inputs and then apply a LOW to \overline{CS} . The address inputs will be decoded inside the ROM to select the correct data word, 11101101, that will appear at outputs D_7 to D_0 . If \overline{CS} is kept HIGH, the ROM outputs will be disabled and will be in the Hi-Z state.

REVIEW QUESTIONS 1. *True or false:* All ROMs are nonvolatile.

- 2. Describe the procedure for reading from ROM.
- 3. What is programming or burning-in a ROM?

12-5 ROM ARCHITECTURE

The internal architecture (structure) of a ROM IC is very complex, and we need not be familiar with all of its detail. It is instructive, however, to look at a simplified diagram of the internal architecture, such as that shown in Figure 12-7, for the 16×8 ROM. There are four basic parts: *register array, row decoder, column decoder,* and *output buffers.*

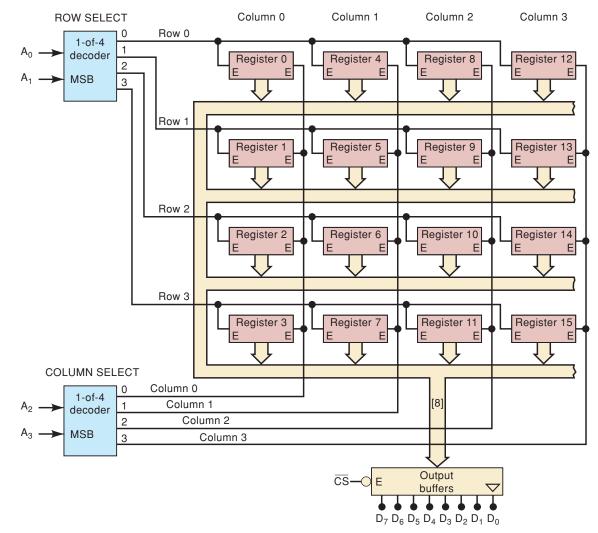


FIGURE 12-7 Architecture of a 16 × 8 ROM. Each register stores one eight-bit word.

Register Array

The register array stores the data that have been programmed into the ROM. Each register contains several memory cells equal to the word size. In this case, each register stores an eight-bit word. The registers are arranged in a square matrix array that is common to many semiconductor memory chips. We can specify the position of each register as being in a specific row and a specific column. For example, register 0 is in row 0, column 0, and register 9 is in row 1, column 2.

The eight data outputs of each register are connected to an internal data bus that runs through the entire circuit. Each register has two enable inputs (E); both must be HIGH in order for the register's data to be placed on the bus.

Address Decoders

The applied address code $A_3A_2A_1A_0$ determines which register in the array will be enabled to place its eight-bit data word onto the bus. Address bits A_1A_0 are fed to a 1-of-4 decoder that activates one row-select line, and address bits A_3A_2 are fed to a second 1-of-4 decoder that activates one column-select line.

Only one register will be in both the row and the column selected by the address inputs, and this one will be enabled.

EXAMPLE 12-5

Which register will be enabled by input address 1101?

Solution

 $A_3A_2 = 11$ will cause the column decoder to activate the column 3 select line, and $A_1A_0 = 01$ will cause the row decoder to activate the row 1 select line. This will place HIGHs at both enable inputs of register 13, thereby causing its data outputs to be placed on the bus. Note that the other registers in column 3 will have only one enable input activated; the same is true for the other row 1 registers.

EXAMPLE 12-6

What input address will enable register 7?

Solution

The enable inputs of this register are connected to the row 3 and column 1 select lines, respectively. To select row 3, the A_1A_0 inputs must be at 11, and to select column 1, the A_3A_2 inputs must be at 01. Thus, the required address will be $A_3A_2A_1A_0 = 0111$.

Output Buffers

The register that is enabled by the address inputs will place its data on the data bus. These data feed into the output buffers, which will pass the data to the external data outputs, provided that \overline{CS} is LOW. If \overline{CS} is HIGH, the output buffers are in the Hi-Z state, and D_7 through D_0 will be floating.

The architecture shown in Figure 12-7 is similar to that of many IC ROMs. Depending on the number of stored data words, the registers in some ROMs will not be arranged in a square array. For example, the Intel 27C64 is a CMOS ROM that stores 8192 eight-bit words. Its 8192 registers are arranged in an array of 256 rows \times 32 registers. ROM capacities range from 256 \times 4 to 8M \times 8.

EXAMPLE 12-7

Describe the internal architecture of a ROM that stores 4K bytes and uses a square register array.

Solution

4K is actually $4 \times 1024 = 4096$, and so this ROM holds 4096 eight-bit words. Each word can be thought of as being stored in an eight-bit register, and there are 4096 registers connected to a common data bus internal to the chip. Because $4096 = 64^2$, the registers are arranged in a 64×64 array; that is, there are 64 rows and 64 columns. This requires a 1-of-64 decoder to decode six address inputs for the row select, and a second 1-of-64 decoder to decode six other address inputs for the column select. Thus, a total of 12 address inputs is required. This makes sense because $2^{12} = 4096$, and there are 4096 different addresses.

REVIEW QUESTIONS

- 1. What input address code is required if we want to read the data from register 9 in Figure 12-7?
- 2. Describe the function of the row-select decoder, the column-select decoder, and the output buffers in the ROM architecture.

12-6 ROM TIMING

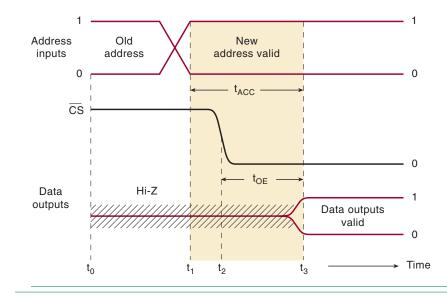
There will be a propagation delay between the application of a ROM's inputs and the appearance of the data outputs during a read operation. This time delay, called access time (t_{ACC}) is a measure of the ROM's operating speed. Access time is described graphically by the waveforms in Figure 12-8.

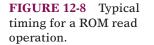
The top waveform represents the address inputs; the middle waveform is an active-LOW chip select, \overline{CS} ; and the bottom waveform represents the data outputs. At time t_0 the address inputs are all at some specific level, some HIGH and some LOW. \overline{CS} is HIGH, so that the ROM data outputs are in their Hi-Z state (represented by the hatched line).

Just prior to t_1 , the address inputs are changing to a new address for a new read operation. At t_1 , the new address is valid; that is, each address input is at a valid logic level. At this point, the internal ROM circuitry begins to decode the new address inputs to select the register that is to send its data to the output buffers. At t_2 , the \overline{CS} input is activated to enable the output buffers. Finally, at t_3 , the outputs change from the Hi-Z state to the valid data that represent the data stored at the specified address.

The time delay between t_1 , when the new address becomes valid, and t_3 , when the data outputs become valid, is the access time t_{ACC} . Typical bipolar ROMs will have access times in the range from 30 to 90 ns; access times of NMOS devices will range from 35 to 500 ns. Improvements to CMOS technology have brought access times into the 20-to-60-ns range. Consequently, bipolar and NMOS devices are rarely produced in newer (larger) ROMs.

Another important timing parameter is the *output enable time* (t_{OE}), which is the delay between the \overline{CS} input and the valid data output. Typical values for t_{OE} are 10 to 20 ns for bipolar, 25 to 100 ns for NMOS, and 12 to 50 ns for CMOS ROMs. This timing parameter is important in situations where





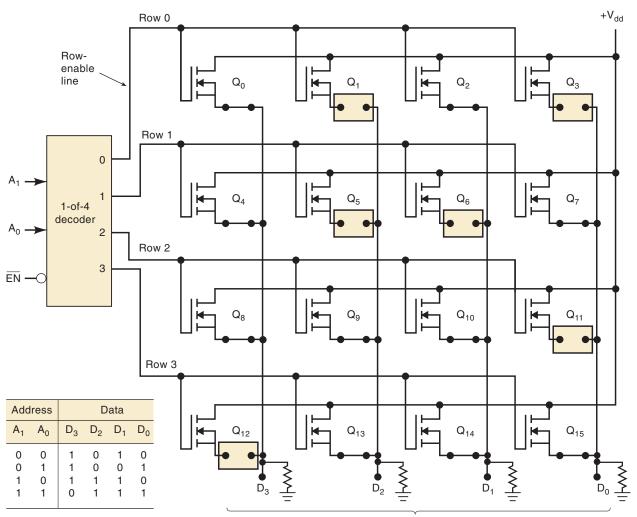
the address inputs are already set to their new values, but the ROM outputs have not yet been enabled. When \overline{CS} goes LOW to enable the outputs, the delay will be t_{OE} .

12-7 TYPES OF ROMs

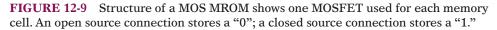
Now that we have a general understanding of the internal architecture and external operation of ROM devices, we will look at the various types of ROMs to see how they differ in the way they are programmed, erased, and reprogrammed.

Mask-Programmed ROM

The mask-programmed ROM (MROM) has its information stored at the time the integrated circuit is manufactured. As you can see from Figure 12-9, ROMs are made up of a rectangular array of transistors. Information is stored by either connecting or disconnecting the source of a transistor to the output



Data outputs



column. The last step in the manufacturing process is to form all these conducting paths or connections. The process uses a "mask" to deposit metals on the silicon that determine where the connections form in a way similar to using stencils and spray paint but on a much smaller scale. The mask is very precise and expensive and must be made specifically for the customer, with the correct binary information. Consequently, this type of ROM is economical only when many ROMs are being made with exactly the same information.

Mask-programmed ROMs are commonly referred to as just ROMs, but this can be confusing because the term ROM actually represents the broad category of devices that, during normal operation, are only read from. We will use the abbreviation MROM whenever we refer to mask-programmed ROMs.

Figure 12-9 shows the structure of a small MOS MROM. It consists of 16 memory cells arranged in four rows of four cells. Each cell is an N-channel MOSFET transistor connected in the common-drain configuration (input at gate, output at source). The top row of cells (ROW 0) constitutes a four-bit register. Note that some of the transistors in this row (Q_0 and Q_2) have their source connected to the output column line, while others (Q_1 and Q_3) do not. The same is true of the cells in each of the other rows. The presence or absence of these source connections determines whether a cell is storing a 1 or a 0, respectively. The condition of each source connection is controlled during production by the photographic mask based on the customer-supplied data.

Notice that the data outputs are connected to column lines. Referring to output D_3 , for instance, any transistor that has a connection from the source (such as Q_0 , Q_4 , and Q_8) to the output column can switch V_{dd} onto the column, making it a HIGH logic level. If V_{dd} is not connected to the column line, the output will be held at a LOW logic level by the pull-down resistor. At any given time, a maximum of one transistor in a column will ever be turned on due to the row decoder.

The 1-of-4 decoder is used to decode the address inputs A_1A_0 to select which row (register) is to have its data read. The decoder's active-HIGH outputs provide the ROW enable lines that are the gate inputs for the various rows of cells. If the decoder's enable input, \overline{EN} , is held HIGH, all of the decoder outputs will be in their inactive LOW state, and all of the transistors in the array will be off because of the absence of any gate voltage. For this situation, the data outputs will all be in the LOW state.

When \overline{EN} is in its active-LOW state, the conditions at the address inputs determine which row (register) will be enabled so that its data can be read at the data outputs. For example, to read ROW 0, the A_1A_0 inputs are set to 00. This places HIGH at the ROW 0 line; all other row lines are at 0 V. This HIGH at ROW 0 turns on transistors Q_0 , Q_1 , Q_2 , and Q_3 . With all of the transistors in the row conducting, V_{dd} will be switched on to each transistor's source lead. Outputs D_3 and D_1 will go HIGH because Q_0 and Q_2 are connected to their respective columns. D_2 and D_0 will remain LOW because there is no path from the Q_1 and Q_3 source leads to their columns. In a similar manner, application of the other address codes will produce data outputs from the corresponding register. The table in Figure 12-9 shows the data for each address. You should verify how this correlates with the source connections to the various cells.

EXAMPLE 12-8

MROMs can be used to store tables of mathematical functions. Show how the MROM in Figure 12-9 can be used to store the function $y = x^2 + 3$, where the input address supplies the value for *x*, and the value of the output data is *y*.

TABLE 12-1

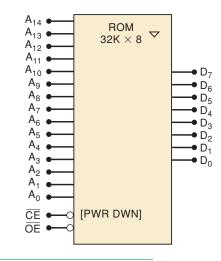
ż	x	$y = x^2 + 3$							
A ₁	Â ₀	$\overline{D_3}$	<i>D</i> ₂	<i>D</i> ₁	\overline{D}_0				
0	0	0	0	1	1				
0	1	0	1	0	0				
1	0	0	1	1	1				
1	1	1	1	0	0				

Solution

The first step is to set up a table showing the desired output for each set of inputs. The input binary number, x, is represented by the address A_1A_0 . The output binary number is the desired value of y. For example, when $x = A_1A_0 = 10_2 = 2_{10}$, the output should be $2^2 + 3 = 7_{10} = 0111_2$. The complete table is shown in Table 12-1. This table is supplied to the MROM manufacturer for developing the mask that will make the appropriate connections within the memory cells during the fabrication process. For instance, the first row in the table indicates that the connections to the source of Q_0 and Q_1 will be left unconnected, while the connections to Q_2 and Q_3 will be made.

MROMs typically have tristate outputs that allow them to be used in a bus system, as we discussed in Chapter 9. Consequently, there must be a control input to enable and disable the tristate outputs. This control input is usually labeled *OE* (for output enable). In order to distinguish this tristate enable input from the address decoder enable input, the latter is usually referred to as a chip enable (*CE*). The chip enable performs more than just enabling the address decoder. When *CE* is disabled, all functions of the chip are disabled, including the tristate outputs, and the entire circuit is placed in a **power-down** mode that draws much less current from the power supply. Figure 12-10 shows a $32K \times 8$ MROM. The 15 address lines (A0–A14) can identify 2^{15} memory locations (32, 767, or 32K). Each memory location holds an eight-bit data value that can be placed on the data lines D7–D0 when the chip is enabled and the outputs are enabled.



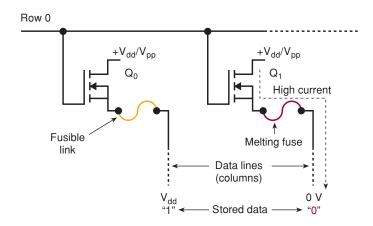


Programmable ROMs (PROMs)

A mask-programmable ROM is very expensive and would not be used except in high-volume applications, where the cost would be spread out over many units. For lower-volume applications, manufacturers have developed **fusiblelink** PROMs that are user-programmable; that is, they are not programmed during the manufacturing process but are custom-programmed by the user. Once programmed, however, a PROM is like an MROM because it cannot be erased and reprogrammed. Thus, if the program in the PROM is faulty or must be changed, the PROM must be thrown away. For this reason, these devices are often referred to as "one-time programmable" (OTP) ROMs.

The fusible-link PROM structure is very similar to the MROM structure because certain connections either are left intact or are opened in order to program a memory cell as a 1 or a 0, respectively. A PROM comes from the manufacturer with a thin, fuse link connection in the source leg of every transistor. In this condition, every transistor stores a 1. The user can then "blow" the fuse for any transistor that needs to store a 0. Typically, data can be programmed or "burned into" a PROM by selecting a row by applying the desired address to the address inputs, placing the desired data on the data pins, and then applying a pulse to a special programming pin on the IC. Figure 12-11 shows the inner workings of how this is done.

FIGURE 12-11 PROMs use fusible links that can be selectively blown open by the user to program a logic 0 into a cell.



All of the transistors in the selected row (row 0) are turned on, and $V_{\rm pp}$ is applied to their drain leads. Those columns (data lines) that have a logic 0 on them (e.g., Q_1) will provide a high-current path through the fusible link, burning it open and permanently storing a logic 0. Those columns that have a logic 1 (e.g., Q_0) have $V_{\rm pp}$ on one side of the fuse and $V_{\rm dd}$ on the other side, drawing much less current and leaving the fuse intact. Once all address locations have been programmed in this manner, the data are permanently stored in the PROM and can be read over and over again by accessing the appropriate address. The data will not change when power is removed from the PROM chip because nothing will cause an open fuse link to become closed again.

A PROM is programmed using the same equipment and process described in Chapter 4 for programming a PLD. The TMS27PC256 is a very popular CMOS PROM with a capacity of $32K \times 8$ and a standby power dissipation of only 1.4 mW. It is available with maximum access times ranging from 100 to 250 ns.

Erasable Programmable ROM (EPROM)

An EPROM can be programmed by the user, and it can also be *erased* and reprogrammed as often as desired. Once programmed, the EPROM is a *nonvolatile* memory that will hold its stored data indefinitely. The process for programming an EPROM is the same as that for a PROM.

The storage element of an EPROM is a MOS transistor with a silicon gate that has no electrical connection (i.e., a floating gate) but is very close to an electrode. In its normal state there is no charge stored on the floating gate and the transistor will produce a logic 1 whenever it is selected by the address decoder. To program a 0, a high-voltage pulse is used to leave a net charge on the floating gate. This charge causes the transistor to output a logic 0 when it is selected. Since the charge is trapped on the floating gate and has no discharge path, the 0 will be stored until it is erased. The data are erased by restoring all cells to a logic 1. To do this, the charge on the floating electrode is neutralized by exposing the silicon to high-intensity ultraviolet (UV) light for several minutes.

The 27C64 is an example of a small $8K \times 8K$ memory IC that is available as a "one-time-programmable" (OTP) PROM or as an erasable UV EPROM. The obvious difference in the two ICs is the EPROM's clear quartz "window," shown in Figure 12-12(b), which allows the UV light to shine on the silicon. Both versions operate from a single +5-V power source during normal operation.

Figure 12-12(a) is the logic symbol for the 27C64. Note that it shows 13 address inputs (because $2^{13} = 8192$) and eight data outputs. It has four control inputs. \overline{CE} is the chip enable input that is used to place the device in a standby mode where its power consumption is reduced. \overline{OE} is the output enable and is used to control the device's data output tristate buffers so that the device can be connected to a microprocessor data bus without bus contention. V_{PP} is the special programming voltage required during the programming process. \overline{PGM} is the program enable input that is activated to store data at the selected address.

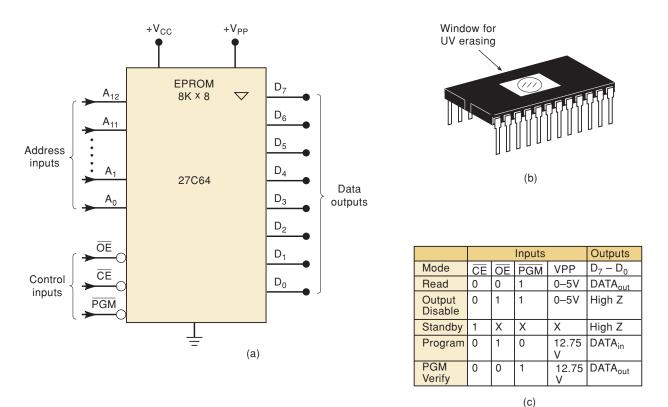


FIGURE 12-12 (a) Logic symbol for 27C64 EPROM; (b) typical EPROM package showing ultraviolet window; (c) 27C64 operating modes.

The 27C64 has several operating modes that are controlled by the \overline{CE} , \overline{OE} , V_{PP} , and PGM pins, as presented in Figure 12-12(c). The program mode is used to write new data into the EPROM cells. This is most often done on a "clean" EPROM, one that has previously been erased with UV light so that all cells are 1s. The programming process writes one eight-bit word into one address location at one time as follows: (1) the address is applied to the address pins; (2) the desired data are placed at the data pins, which function as inputs during the programming process; (3) a higher programming voltage of 12.75V is applied to V_{PP} ; (4) \overline{CE} is held LOW; (5) \overline{PGM} is pulsed LOW for 100 μ s and the data are read back. If the data were not successfully stored, another pulse is applied to \overline{PGM} . This is repeated at the same address until the data are successfully stored.

A clean EPROM can be programmed in less than a minute once the desired data have been entered, transferred, or downloaded into the EPROM programmer. The 27C512 is a common $64K \times 8$ EPROM that operates very much like the 27C64 but offers more storage capacity.

The major disadvantages of UVEPROMs are that they must be removed from the circuit to be programmed and erased, the erase operation erases the entire chip, and the erase operation takes up to 20 minutes.

Electrically Erasable PROM (EEPROM)

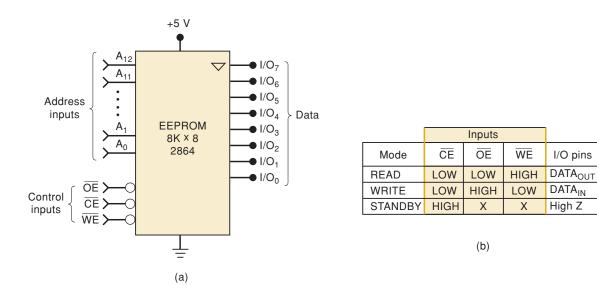
The disadvantages of the EPROM were overcome by the development of the **electrically erasable PROM (EEPROM)** as an improvement over the EPROM. The EEPROM retains the same floating-gate structure as the EPROM, but with the addition of a very thin oxide region above the drain of the MOSFET memory cell. This modification produces the EEPROM's major characteristic its electrical erasability. By applying a high voltage (21 V) between the MOSFET's gate and drain, a charge can be induced onto the floating gate, where it will remain even when power is removed; reversal of the same voltage causes a removal of the trapped charges from the floating gate and erases the cell. Because this charge-transport mechanism requires very low currents, the erasing and programming of an EEPROM can be done *in circuit* (i.e., without a UV light source and a special PROM programmer unit).

Another advantage of the EEPROM over the EPROM is the ability to erase and rewrite *individual* bytes (eight-bit words) in the memory array electrically. During a write operation, internal circuitry automatically erases all of the cells at an address location prior to writing in the new data. This byte erasability makes it much easier to make changes in the data stored in an EEPROM.

The early EEPROMs, such as Intel's 2816, required appropriate support circuitry external to the memory chips. This support circuitry included the 21-V programming voltage (V_{PP}), usually generated from a +5 V supply through a dc-to-dc converter, and it included circuitry to control the timing and sequencing of the erase and programming operations. The newer devices, such as the Intel 2864, have integrated this support circuitry onto the same chip with the memory array, so that it requires only a single 5-V power pin. This makes the EEPROM as easy to use as the read/write memory we will be discussing shortly.

The byte erasability of the EEPROM and its high level of integration come with two penalties: density and cost. The memory cell complexity and the on-chip support circuitry place EEPROMs far behind an EPROM in bit capacity per square millimeter of silicon; a 1-Mbit EEPROM requires about twice as much silicon as a 1-Mbit EPROM. So despite its operational superiority, the EEPROM's shortcomings in density and cost-effectiveness have kept it from replacing the EPROM in applications where density and cost are paramount factors.

The logic symbol for the Intel 2864 is shown in Figure 12-13(a). It is organized as an $8K \times 8$ array with 13 address inputs ($2^{13} = 8192$) and eight data I/O pins. Three control inputs determine the operating mode according to the



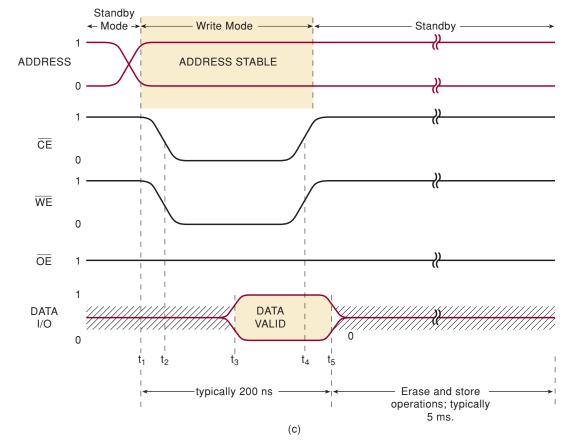


FIGURE 12-13 (a) Symbol for the 2864 EEPROM; (b) operating modes; (c) timing for the write operation.

table given in Figure 12-13(b). With \overline{CE} = HIGH, the chip is in its low-power standby mode, in which no operations are being performed on any memory location and the data pins are in the Hi-Z state.

To read the contents of a memory location, the desired address is applied to the address pins; \overline{CE} is driven LOW; and the output enable pin, \overline{OE} , is driven LOW to enable the chip's output data buffers. The write enable pin, \overline{WE} , is held HIGH during a read operation.

To write into (program) a memory location, the output buffers are disabled so that the data to be written can be applied as inputs to the I/O pins. The timing for the write operation is diagrammed in Figure 12-13(c). Prior to t_1 , the device is in the standby mode. A new address is applied at that time. At t_2 , the \overline{CE} and \overline{WE} inputs are driven LOW to begin the write operation; \overline{OE} is HIGH so that the data pins will remain in the Hi-Z state. Data are applied to the I/O pins at t_3 and are written into the address location on the rising edge of \overline{WE} at t_4 . The data are removed at t_5 . Actually, the data are first latched (on the rising edge of \overline{WE}) into a FF buffer memory that is part of the 2864 circuitry. The data are held there while other circuitry on the chip performs an erase operation on the selected address location in the EEPROM array, after which the data byte is transferred from the buffer to the EEPROM array and stored at that location. This erase and store operation typically takes 5 ms. With \overline{CE} returned HIGH at t_4 , the chip is back in the standby mode while the internal erase and store operations are completed.

The 2864 has an enhanced write mode that allows the user to write up to 16 bytes of data into the FF buffer memory, where it is held while the EEPROM circuitry erases the selected address locations. The 16 bytes of data are then transferred to the EEPROM array for storage at these locations. This process also takes about 5 ms.

Because the internal process of storing a data value in an EEPROM is quite slow, the speed of the data transfer operation can also be slower. Consequently, many manufacturers offer EEPROM devices in eight-pin packages that are interfaced to a two- or three-wire *serial* bus. This saves physical space on the system board as opposed to using a 2864 in a 28-pin, wide-DIP package. It also simplifies the hardware interface between the CPU and the EEPROM.

CD-ROM

A very prominent type of read-only storage used today in computer systems is the compact disk (CD). The disk technology and the hardware necessary to retrieve the information are the same as those used in audio systems. Only the format of the data is different. The disks are manufactured with a highly reflective surface. To store data on the disks, a very intense laser beam is focused on a *very* small point on the disk. This beam burns a light-diffracting pit at that point on the disk surface. Digital data (1s and 0s) are stored on the disk one bit at a time by burning or not burning a pit into the reflective coating. The digital information is arranged on the disk as a continuous spiral of data points. The precision of the laser beam allows very large quantities of data (over 550 Mbytes) to be stored on a small, 120-mm disk.

In order to read the data, a much less powerful laser beam is focused onto the surface of the disk. At any point, the reflected light is sensed as either a 1 or a 0. This optical system is mounted on a mechanical carriage that moves back and forth along the radius of the disk, following the spiral of data as the disk rotates. The data retrieved from the optical system come one bit at a time in a serial data stream. The angular rotation of the disk is controlled to maintain a constant rate of incoming data points. If the disk is being used for audio recording, this stream of data is converted into an analog waveform. If the disk is being used as ROM, the data are decoded into parallel bytes that the computer can use. The CD player technology, although very sophisticated, is relatively inexpensive and is becoming a standard way of loading large amounts of data into a personal computer. The major improvements that are occurring now in CD-ROM technology involve quicker access time in retrieving data.

REVIEW QUESTIONS

- 1. True or false: An MROM can be programmed by the user.
- 2. How does a PROM differ from an MROM? Can it be erased and reprogrammed?
- 3. True or false: A PROM stores a logic 1 when its fusible link is intact.
- 4. How is an EPROM erased?
- 5. True or false: There is no way to erase only a portion of an EPROM's memory.
- 6. What function is performed by PROM and EPROM programmers?
- 7. What EPROM shortcomings are overcome by EEPROMs?
- 8. What are the major drawbacks of EEPROM?
- 9. What type of ROM can erase one byte at a time?
- 10. How many bits are read from a CD-ROM disk at any point in time?

12-8 FLASH MEMORY

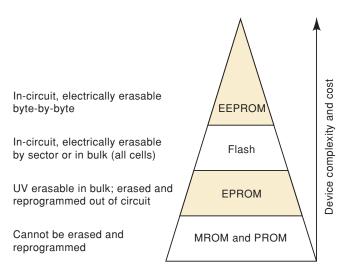
EPROMs are nonvolatile, offer fast read access times (typically 120 ns), and have high density and low cost per bit. They do, however, require removal from their circuit/system to be erased and reprogrammed. EEPROMs are nonvolatile, offer fast read access, and allow rapid in-circuit erasure and reprogramming of individual bytes. They suffer from lower density and much higher cost than EPROMs.

The challenge for semiconductor engineers was to fabricate a nonvolatile memory with the EEPROM's in-circuit electrical erasability, but with densities and costs much closer to those of EPROMs, while retaining the high-speed read access of both. The response to this challenge was the **flash memory**.

Structurally, a flash memory cell is like the simple single-transistor EPROM cell (and unlike the more complex two-transistor EEPROM cell), being only slightly larger. It has a thinner gate-oxide layer that allows electrical erasability but can be built with much higher densities than EEPROMs. The cost of flash memory is considerably less than for EEPROM. Figure 12-14 illustrates the trade-offs for the various semiconductor nonvolatile memories. As erase/programming flexibility increases (from base to apex of the triangle), so do device complexity and cost. MROM and PROM are the simplest and cheapest devices, but they cannot be erased and reprogrammed. EEPROM is the most complex and expensive because it can be erased and reprogrammed in circuit on a byte-by-byte basis.

Flash memories are so called because of their rapid erase and write times. Most flash chips use a *bulk erase* operation in which all cells on the chip are erased simultaneously; this bulk erase process typically requires hundreds of milliseconds compared to 20 minutes for UV EPROMs. Some newer flash memories offer a *sector erase* mode, where specific sectors of the memory array (e.g., 512 bytes) can be erased at one time. This prevents having to erase and reprogram all cells when only a portion of the memory needs to be updated. A typical flash memory has a write time of 10 μ s per byte compared

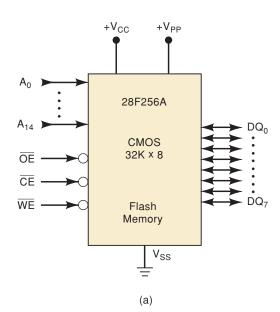
FIGURE 12-14 Trade-offs for semiconductor nonvolatile memories show that complexity and cost increase as erase and programming flexibility increases.



to 100 μ s for the most advanced EPROM and 5 ms for EEPROM (which includes automatic byte erase time).

The 28F256A CMOS Flash Memory IC

Figure 12-15(a) shows the logic symbol for Intel Corporation's 28F256A CMOS flash memory chip, which has a capacity of $32K \times 8$. The diagram shows 15 address inputs (A_0 - A_{14}) needed to select the different memory addresses; that is, $2^{15} = 32K = 32,768$. The eight data input/output pins (DQ_0 - DQ_7) are used as inputs during memory write operations and as outputs during memory read operations. These data pins float to the Hi-Z state when the chip is deselected ($\overline{CE} = \text{HIGH}$) or when the outputs are disabled ($\overline{OE} = \text{HIGH}$) The write enable input (\overline{WE}) is used to control memory write operations. Note that the chip requires two power-supply voltages: V_{CC} is the standard +5 V used for the logic circuitry; V_{PP} is the erase/programming power-supply voltage, nominally +12 V, which is needed for the erase and programming (write) operations. Newer



		Inputs		
Mode	CE	ŌĒ	WE	Data pins
READ	LOW	LOW	HIGH	DATA _{OUT}
STANDBY	HIGH	Х	Х	High Z
WRITE*	LOW	HIGH	LOW	DATA _{IN}

*Note: If $V_{PP} \le 6.5$ V, a write operation cannot be performed

(b)

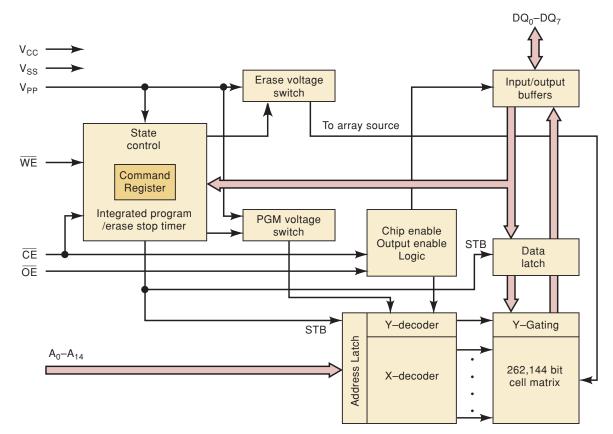
FIGURE 12-15 (a) Logic symbol for the 28F256A flash memory chip; (b) control inputs \overline{CE} , \overline{WE} , and \overline{OE} .

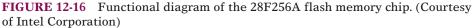
flash chips generate V_{PP} internally and require only a single supply. The latest low-voltage devices operate on only 1.8 V.

The control inputs (\overline{CE} , \overline{OE} , and \overline{WE}) control what happens at the data pins in much the same way as for the 2864 EEPROM, as the table in Figure 12-15(b) shows. These data pins are normally connected to a data bus. During a write operation, data are transferred over the bus—usually from the microprocessor—and into the chip. During a read operation, data from inside the chip are transferred over the data bus—usually to the microprocessor.

The operation of this flash memory chip can be better understood by looking at its internal structure. Figure 12-16 is a diagram of the 28F256A showing its major functional blocks. You should refer to this diagram as needed during the following discussion. The unique feature of this structure is the *command register*, which is used to manage all of the chip functions. Command codes are written into this register to control which operations take place inside the chip (e.g., erase, erase-verify, program, program-verify). These command codes usually come over the data bus from the microprocessor. State control logic examines the contents of the command register and generates logic and control signals to the rest of the chip's circuits to carry out the steps in the operation. Some examples of the types of commands that can be sent to the flash are shown here to give you an idea of why they are necessary. Each command is stored in the command register by using the same write cycle as described for the EEPROM in Figure 12-13(c).

Read Command. Writing a code of 00 hex into the command register prepares the memory IC for the read operation. After this, a normal read cycle can be used to access data stored at any address.





Set-Erase/Erase Command. The code of 20 hex must be written to the command register twice in a row to begin the internal erase sequence.

Erase Verify Command. This command (FF Hex) causes the memory IC to check all of its memory locations to verify that all bits are HIGH.

Set-Up Program/Program Command. This command (40 hex) puts the memory IC in a mode that allows subsequent write cycles to store data at a specified address, one byte at a time.

Program-Verify Command. This command (C0 hex) is used to verify that the correct data have been stored in the flash ROM. After this code is written to the command register, the next read operation will produce the contents of the last location that was written to, and these data can be compared with the intended value.

Improved Flash Memory

The core architecture of flash memory today and the basic set of command codes are very similar to those of the first-generation devices. The newest flash devices have new features, and new command codes to control these features, in addition to those common to earlier devices. Of course, the latest flash devices have much more capacity, run on much less power (and at lower voltages), come in smaller packages, and cost much less per bit than their predecessors. They also offer features such as the ability to read/write data while a block of memory is being erased. The V_{pp} programming voltage is generated internally, allowing it to use a single supply. The speed of operation can be enhanced by using a burst mode. This simply means that several addresses in a row can be accessed very rapidly, providing a burst of data transfer. A synchronous clock input is provided to control the burst operation. A base address is latched into the memory and then the contents of this location are transferred on the clock edge, which also increments the address to the next location. In this way, several sequential memory locations are accessed as fast as the system clock can oscillate, without the overhead of generating each address. All of these features have made flash memory the predominant solid-state nonvolatile memory technology in use today.

REVIEW QUESTIONS

- 1. What is the main advantage of flash memory over EPROMs?
- 2. What is the main advantage of flash memory over EEPROMs?
- 3. Where does the word *flash* come from?
- 4. What is V_{PP} needed for?
- 5. What is the function of the 28F256A's command register?
- 6. What is the purpose of an erase-verify command?
- 7. What is the purpose of the program-verify command?

12-9 ROM APPLICATIONS

With the exception of MROM and PROM, most ROM devices can be reprogrammed, so technically they are not *read-only* memories. However, the term *ROM* can still be used to include EPROMs, EEPROMs, and flash memory because, during normal operation, the stored contents of these devices is not changed nearly as often as it is read. So ROMs are taken to include all semiconductor, nonvolatile memory devices, and they are used in applications where nonvolatile storage of information, data, or program codes is needed and where the stored data rarely or never change. Here are some of the most common application areas.

Embedded Microcontroller Program Memory

Microcontrollers are prevalent in most consumer electronic products on the market today. Your car's automatic braking system and engine controller, your cell phone, your digital camcorder, your microwave oven, and many other products have a microcontroller for a brain. These little computers have their program instructions stored in nonvolatile memory—in other words, in a ROM. Most embedded microcontrollers today have flash ROM integrated into the same IC as the CPU. Many also have an area of EEPROM that offers the features of byte erasure and nonvolatile storage.

Data Transfer and Portability

The need to store and transfer large sets of binary information is a requirement of many low-power battery-operated systems today. Cell phones store photos and video clips. Digital cameras store many pictures on removable memory media. Flash drives connect to a computer's USB port and store gigabytes of information. Your MP-3 player is loaded up with music and runs all day on batteries. A PDA (personal digital assistant) stores appointment information, email, addresses, and even entire books. All of these common personal electronic gadgets require the low-power, low-cost, high-density, nonvolatile storage with in-circuit write capability that is available in flash memory.

Bootstrap Memory

Many microcomputers and most larger computers do not have their operating system programs stored in ROM. Instead, these programs are stored in external mass memory, usually magnetic disk. How, then, do these computers know what to do when they are powered on? A relatively small program, called a **bootstrap program**, is stored in ROM. (The term *bootstrap* comes from the idea of pulling oneself up by one's own bootstraps.) When the computer is powered on, it will execute the instructions that are in this bootstrap program. These instructions typically cause the CPU to initialize the system hardware. The bootstrap program then loads the operating system programs from mass storage (disk) into its main internal memory. At that point, the computer begins executing the operating system program and is ready to respond to the user commands. This startup process is often called "booting up the system."

Many of the digital signal processing chips load their internal program memory from an external bootstrap ROM when they are powered on. Some of the more advanced PLDs also load the programming information that configures their logic circuits from an external ROM into a RAM area inside the PLD. This is also done when power is applied. In this way, the PLD is reprogrammed by changing the bootstrap ROM, rather than changing the PLD chip itself.

Data Tables

ROMs are often used to store tables of data that do not change. Some examples are the trigonometric tables (i.e., sine, cosine, etc.) and code-conversion tables. The digital system can use these data tables to "look up" the correct

value. For example, a ROM can be used to store the sine function for angles from 0° to 90°. It could be organized as a 128×8 with seven address inputs and eight data outputs. The address inputs represent the angle in increments of approximately 0.7°. For example, address 0000000 is 0°, address 0000001 is 0.7°, address 0000010 is 1.41°, and so on, up to address 1111111, which is 89.3°. When an address is applied to the ROM, the data outputs will represent the approximate sine of the angle. For example, with input address 1000000 (representing approximately 45°) the data outputs will be 10110101. Because the sine is less than or equal to 1, these data are interpreted as a fraction, that is, 0.10110101, which when converted to decimal equals 0.707 (the sine of 45°). It is vital that the user of this ROM understands the format in which the data are stored.

Standard look-up-table ROMs for functions such as these were at one time readily available TTL chips. Only a few are still in production. Today, most systems that need to look up equivalent values involve a microprocessor, and the "look-up" table data are stored in the same ROM that holds the program instructions.

Data Converter

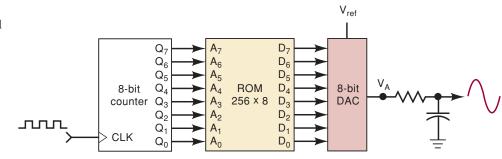
The data-converter circuit takes data expressed in one type of code and produces an output expressed in another type. Code conversion is needed, for example, when a computer is outputting data in straight binary code and we want to convert it to BCD in order to display it on 7-segment LED readouts.

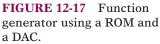
One of the easiest methods of code conversion uses a ROM programmed so that the application of a particular address (the old code) produces a data output that represents the equivalent in the new code. The 74185 is a TTL ROM that stores the binary-to-BCD code conversion for a six-bit binary input. To illustrate, a binary address input of 100110 (decimal 38) will produce a data output of 00111000, which is the BCD code for decimal 38.

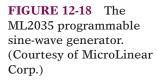
Function Generator

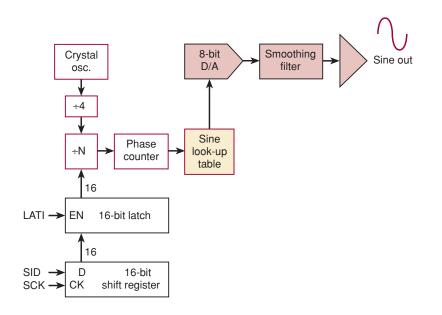
The function generator is a circuit that produces waveforms such as sine waves, sawtooth waves, triangle waves, and square waves. Figure 12-17 shows how a ROM look-up table and a DAC are used to generate a sine-wave output signal.

The ROM stores 256 different eight-bit values, each one corresponding to a different waveform value (i.e., a different voltage point on the sine wave). The eight-bit counter is continuously pulsed by a clock signal to provide sequential address inputs to the ROM. As the counter cycles through the 256 different addresses, the ROM outputs the 256 data points to the DAC. The DAC output will be a waveform that steps through the 256 different analog voltage values corresponding to the data points. The low-pass filter smooths out the steps in the DAC output to produce a smooth waveform.









Circuits such as this are used in some commercial function generators. The same idea is employed in some speech synthesizers, where the digitized speech waveform values are stored in the ROM. The ML2035, illustrated in Figure 12-18, is a programmable sine-wave generator chip that incorporates this basic strategy to generate a sine wave of fixed amplitude and a frequency that can be selected from dc to 50 kHz. The number that is shifted into the 16-bit shift register is used to determine the clocking frequency for the counter that drives the address inputs on the ROM look-up table. The ML2035 is intended for telecommunications applications that require precise tones of various frequencies to be generated.

Auxiliary Storage

Because of their nonvolatility, high speed, low power requirements, and lack of moving parts, flash memory modules have become feasible alternatives to magnetic disk storage. This is especially true for lower capacities (5 Mbytes or less), where flash is cost-competitive with magnetic disk. The low power consumption of flash memory makes it particularly attractive for laptop and notebook computers that use battery power.

REVIEW QUESTIONS	1. Describe how a computer uses a bootstrap program.
	2. What is a code converter?
	3. What are the main elements of a function generator?
	4. Why are flash memory modules a feasible alternative to auxiliary disk storage?

12-10 SEMICONDUCTOR RAM

Recall that the term *RAM* stands for *random-access memory*, meaning that any memory address location is as easily accessible as any other. Many types of memory can be classified as having random access, but when the term RAM is used with semiconductor memories, it is usually taken to mean read/write memory (RWM) as opposed to ROM. Because it is common practice to use RAM to mean semiconductor RWM, we will do so throughout the following discussions.

RAM is used in computers for the temporary storage of programs and data. The contents of many RAM address locations will be read from and written to as the computer executes a program. This requires fast read and write cycle times for the RAM so as not to slow down the computer operation.

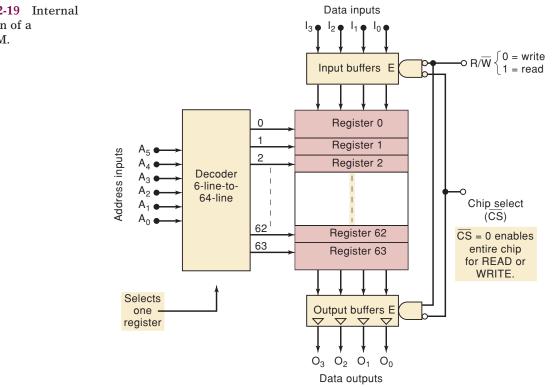
A major disadvantage of RAM is that it is volatile and will lose all stored information if power is interrupted or turned off. Some CMOS RAMs, however, use such small amounts of power in the standby mode (no read or write operations taking place) that they can be powered from batteries whenever the main power is interrupted. Of course, the main advantage of RAM is that it can be written into and read from rapidly with equal ease.

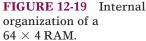
The following discussion of RAM will draw on some of the material covered in our treatment of ROM because many of the basic concepts are common to both types of memory.

RAM ARCHITECTURE 12-11

As with the ROM, it is helpful to think of the RAM as consisting of a number of registers, each storing a single data word, and each having a unique address. RAMs typically come with word capacities of 1K, 4K, 8K, 16K, 64K, 128K, 256K, and 1024K, and with word sizes of one, four, or eight bits. As we will see later, the word capacity and the word size can be expanded by combining memory chips.

Figure 12-19 shows the simplified architecture of a RAM that stores 64 words of four bits each (i.e., a 64×4 memory). These words have addresses





ranging from 0 to 63_{10} . In order to select one of the 64 address locations for reading or writing, a binary address code is applied to a decoder circuit. Because $64 = 2^6$, the decoder requires a six-bit input code. Each address code activates one particular decoder output, which in turn enables its corresponding register. For example, assume an applied address code of

$$A_5A_4A_3A_2A_1A_0 = 011010$$

Because $011010_2 = 26_{10}$, decoder output 26 will go high, selecting register 26 for either a read or a write operation.

Read Operation

The address code picks out one register in the memory chip for reading or writing. In order to *read* the contents of the selected register, the READ/WRITE $(R/\overline{W})^*$ input must be a 1. In addition, the CHIP SELECT (\overline{CS}) input must be activated (a 0 in this case). The combination of $R/\overline{W} = 1$ and $\overline{CS} = 0$ enables the output buffers so that the contents of the selected register will appear at the four data outputs. $R/\overline{W} = 1$ also *disables* the input buffers so that the data inputs do not affect the memory during a read operation.

Write Operation

To write a new four-bit word into the selected register requires $R/\overline{W} = 0$ and $\overline{CS} = 0$. This combination *enables* the input buffers so that the four-bit word applied to the data inputs will be loaded into the selected register. The $R/\overline{W} = 0$ also *disables* the output buffers, which are tristate, so that the data outputs are in their Hi-Z state during a write operation. The write operation, of course, destroys the word that was previously stored at that address.

Chip Select

Most memory chips have one or more *CS* inputs that are used to enable the entire chip or disable it completely. In the disabled mode, all data inputs and data outputs are disabled (Hi-Z) so that neither a read nor a write operation can take place. In this mode, the contents of the memory are unaffected. The reason for having *CS* inputs will become clear when we combine memory chips to obtain larger memories. Note that many manufacturers call these inputs CHIP ENABLE (*CE*). When the *CS* or *CE* inputs are in their active state, the memory chip is said to be *selected*; otherwise, it is said to be *deselected*. Many memory ICs are designed to consume much less power when they are deselected. In large memory systems, for a given memory operation, one or more memory chips will be selected while all others are deselected. More will be said on this topic later.

Common Input/Output Pins

In order to conserve pins on an IC package, manufacturers often combine the data input and data output functions using common input/output pins. The R/\overline{W} input controls the function of these I/O pins. During a read operation, the I/O pins act as data outputs that reproduce the contents of the selected address location. During a write operation, the I/O pins act as data inputs to which the data to be written are applied.

^{*}Some manufacturers use the symbol \overline{WE} (write enable) or \overline{W} instead of R/\overline{W} . In any case, the operation is the same.

We can see why this is done by considering the chip in Figure 12-19. With separate input and output pins, a total of 18 pins is required (including ground and power supply). With four common I/O pins, only 14 pins are required. The pin saving becomes even more significant for chips with larger word size.

EXAMPLE 12-9

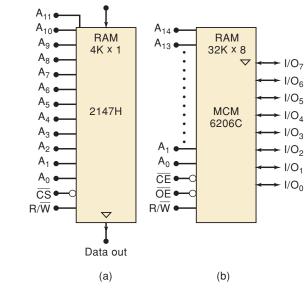
The 2147H is an NMOS RAM that is organized as a $4K \times 1$ with separate data input and output and a single active-LOW chip select input. Draw the logic symbol for this chip, showing all pin functions.

Data in

Solution

The logic symbol is shown in Figure 12-20(a).

FIGURE 12-20 Logic symbols for (a) the 2147H RAM chip; (b) the MCM6206C RAM.



EXAMPLE 12-10

The MCM6206C is a CMOS RAM with $32K \times 8$ capacity, common I/O pins, an active-LOW chip enable, and an active-LOW output enable. Draw the logic symbol.

Solution

The logic symbol is shown in Figure 12-20(b).

In most applications, memory devices are used with a bidirectional data bus like we studied in Chapter 9. For this type of system, even if the memory chip had separate input and output pins, they would be connected together on the same data bus. A RAM having separate input and output pins is referred to as dual-port RAM. These are used in applications where speed is very important and the data in comes from a different device than the data out is going to. A good example is the video RAM on your PC. The RAM must be read repeatedly by the video card to refresh the screen and constantly filled with new updated information from the system bus.

REVIEW QUESTIONS

- 1. Describe the input conditions needed to read a word from a specific RAM address location.
- 2. Why do some RAM chips have common input/output pins?
- 3. How many pins are required for the MCM6208C 64K \times 4 RAM with one CS input and common I/O?

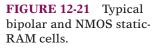
12-12 STATIC RAM (SRAM)

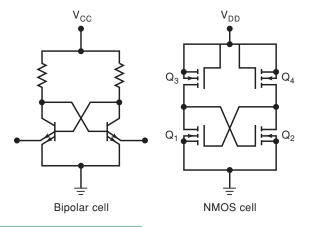
The RAM operation that we have been discussing up to this point applies to a static RAM—one that can store data as long as power is applied to the chip. Static-RAM memory cells are essentially flip-flops that will stay in a given state (store a bit) indefinitely, provided that power to the circuit is not interrupted. In Section 12-13, we will describe **dynamic RAM**, which stores data as charges on capacitors. With dynamic RAMs, the stored data will gradually disappear because of capacitor discharge, so it is necessary to **refresh** the data periodically (i.e., recharge the capacitors).

Static RAMs (SRAMs) are available in bipolar, MOS, and BiCMOS technologies; the majority of applications use NMOS or CMOS RAMs. As stated earlier, the bipolars have the advantage in speed (although CMOS is gradually closing the gap), and MOS devices have much greater capacities and lower power consumption. Figure 12-21 shows for comparison a typical bipolar static memory cell and a typical NMOS static memory cell. The bipolar cell contains two bipolar transistors and two resistors, while the NMOS cell contains four N-channel MOSFETs. The bipolar cell requires more chip area than the MOS cell because a bipolar transistor is more complex than a MOSFET, and because the bipolar cell requires separate resistors while the MOS cell uses MOSFETs as resistors (Q_3 and Q_4). A CMOS memory cell would be similar to the NMOS cell except that it would use P-channel MOSFETs in place of Q_3 and Q_4 . This results in the lowest power consumption but increases the chip complexity.

Static-RAM Timing

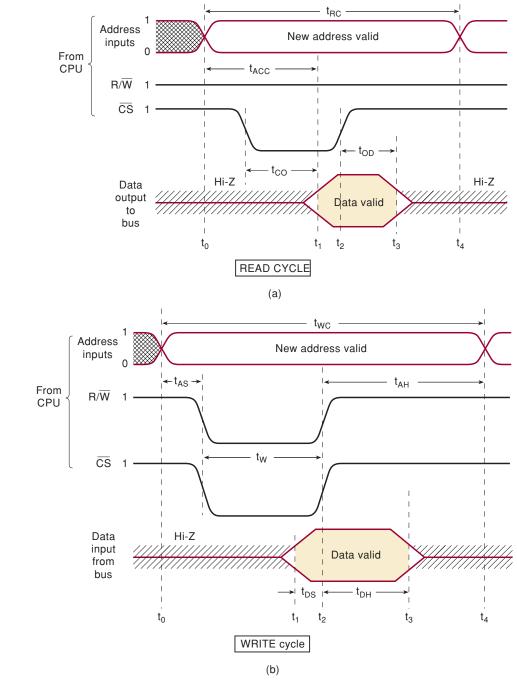
RAM ICs are most often used as the internal memory of a computer. The CPU (central processing unit) continually performs read and write operations on this memory at a very fast rate that is determined by the limitations

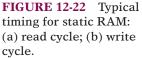




of the CPU. The memory chips that are interfaced to the CPU must be fast enough to respond to the CPU read and write commands, and a computer designer must be concerned with the RAM's various timing characteristics.

Not all RAMs have the same timing characteristics, but most of them are similar, and so we will use a typical set of characteristics for illustrative purposes. The nomenclature for the different timing parameters will vary from one manufacturer to another, but the meaning of each parameter is usually easy to determine from the memory timing diagrams on the RAM data sheets. Figure 12-22 shows the timing diagrams for a complete read cycle and a complete write cycle for a typical RAM chip.





Read Cycle

The waveforms in Figure 12-22(a) show how the address, R/W, and chip select inputs behave during a memory read cycle. As noted, the CPU supplies these input signals to the RAM when it wants to read data from a specific RAM address location. Although a RAM may have many address inputs coming from the CPU's address bus, for clarity the diagram shows only two. The RAM's data output is also shown; we will assume that this particular RAM has one data output. Recall that the RAM's data output is connected to the CPU data bus (Figure 12-5).

The read cycle begins at time t_0 . Prior to that time, the address inputs will be whatever address is on the address bus from the preceding operation. Because the RAM's chip select is not active, it will not respond to its "old" address. Note that the R/\overline{W} line is HIGH prior to t_0 and stays HIGH throughout the read cycle. In most memory systems, R/\overline{W} is normally kept in the HIGH state except when it is driven LOW during a write cycle. The RAM's data output is in its Hi-Z state because $\overline{CS} = 1$.

At t_0 , the CPU applies a new address to the RAM inputs; this is the address of the location to be read. After allowing time for the address signals to stabilize, the \overline{CS} line is activated. The RAM responds by placing the data from the addressed location onto the data output line at t_1 . The time between t_0 and t_1 is the RAM's access time, t_{ACC} , and is the time between the application of the new address and the appearance of valid output data. The timing parameter, t_{CO} , is the time it takes for the RAM output to go from Hi-Z to a valid data level once \overline{CS} is activated.

At time t_2 , the *CS* is returned HIGH, and the RAM output returns to its Hi-*Z* state after a time interval, t_{OD} . Thus, the RAM data will be on the data bus between t_1 and t_3 . The CPU can take the data from the data bus at any point during this interval. In most computers, the CPU will use the PGT of the \overline{CS} signal at t_2 to latch these data into one of its internal registers.

The complete read cycle time, t_{RC} , extends from t_0 to t_4 , when the CPU changes the address inputs to a different address for the next read or write cycle.

Write Cycle

Figure 12-22(b) shows the signal activity for a write cycle that begins when the CPU supplies a new address to the RAM at a time t_0 . The CPU drives the R/\overline{W} and \overline{CS} lines LOW after waiting for a time interval t_{AS} , called the *address setup time*. This gives the RAM's address decoders time to respond to the new address. R/\overline{W} and \overline{CS} are held LOW for a time interval t_W , called the write time interval.

During this write time interval, at time t_1 , the CPU applies valid data to the data bus to be written into the RAM. These data must be held at the RAM input for at least a time interval t_{DS} prior to, and for at least a time interval t_{DH} after, the deactivation of R/\overline{W} and \overline{CS} at t_2 . The t_{DS} interval is called the *data setup time*, and t_{DH} is called the *data hold time*. Similarly, the address inputs must remain stable for the address hold time interval, t_{AH} , after t_2 . If any of these setup time or hold time requirements are not met, the write operation will not take place reliably.

The complete write-cycle time, t_{WC} , extends from t_0 to t_4 , when the CPU changes the address lines to a new address for the next read or write cycle.

The read-cycle time, t_{RC} , and write-cycle time, t_{WC} , are what essentially determine how fast a memory chip can operate. For example, in an actual application, a CPU will often be reading successive data words from memory

one right after the other. If the memory has a $t_{\rm RC}$ of 50 ns, the CPU can read one word every 50 ns, or 20 million words per second; with $t_{\rm RC} = 10$ ns, the CPU can read 100 million words per second. Table 12-2 shows the minimum read-cycle and write-cycle times for some representative static-RAM chips.

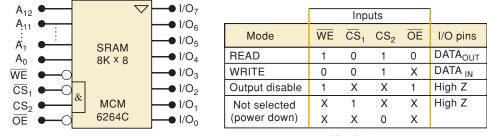
TABLE 12-2

Device	<i>t</i> _{RC} (min) (ns)	t _{WC} (min) (ns)
CMOS MCM6206C, 32K $ imes$ 8	15	15
NMOS 2147H, 4K $ imes$ 1	35	35
BiCMOS MCM6708A, 64K $ imes$ 4	8	8

Actual SRAM Chip

An example of an actual SRAM IC is the MCM6264C CMOS $8K \times 8$ RAM with read-cycle and write-cycle times of 12 ns and a standby power consumption of only 100 mW. The logic symbol for this IC is shown in Figure 12-23. Notice that it has 13 address inputs, because $2^{13} = 8192 = 8K$, and eight data I/O lines. The four control inputs determine the device's operating mode according to the accompanying mode table.

The *WE* input is the same as the R/W input that we have been using. A LOW at \overline{WE} will write data into the RAM, provided that the device is selected—both chip select inputs are active. Note that the "&" symbol is used to denote that *both* must be active. A HIGH at \overline{WE} will produce the read operation, provided that the device is selected and the output buffers are enabled by \overline{OE} = LOW. When deselected, the device is in its low-power mode, and none of the other inputs have any effect.



X = don't care

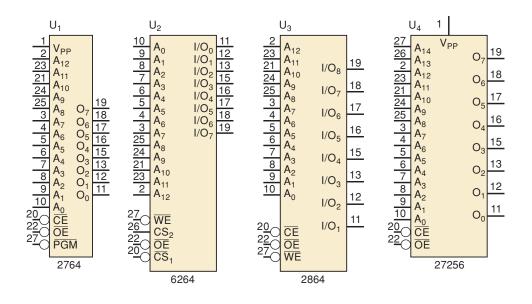
Most of the devices that have been discussed in this chapter are available from several different manufacturers. Each manufacturer may offer different devices of the same dimension (e.g., $32K \times 8$) but with different specifications or features. There are also various types of packaging available such as DIP, PLCC, and various forms of gull-wing and surface-mount.

As you look at the various memory devices that have been described in this chapter, you will notice some similarities. For example, look at the chips in Figure 12-24 and take note of the pin assignments. The fact that the same function is assigned to the same pins on all of these diverse devices, manufactured

FIGURE 12-23 Symbol and mode table for the CMOS MCM6264C.



packaging.



by different companies, is no coincidence. Industry standards created by the Joint Electronic Device Engineering Council (**JEDEC**) have led to memory devices that are interchangeable.

EXAMPLE 12-11

A system is wired for an $8K \times 8$ ROM chip (2764) and two $8K \times 8$ SRAM chips (6264). The entire 8K of ROM space is being used for storage of the microprocessor's instructions. You want to upgrade the system to have some nonvolatile read/write storage. Can the existing circuit be modified to accommodate the new revisions?

Solution

A 2864 EEPROM chip can simply be substituted into one of the RAM sockets. The only functional difference is the much longer write-cycle time requirements of the EEPROM. This can usually be handled by changing the program of the microcomputer that is using the memory device. Because there is no room left in the ROM for these changes, we need a larger ROM. A $32K \times 8 \text{ ROM} (27C256)$ has basically the same pin-out as a 2764. We simply need to connect two more address lines (A_{13} and A_{14}) to the ROM socket and replace the old chip with a 27C256 chip.

Many memory systems take advantage of the versatility that the JEDEC standards provide. The pins that are common for all of the devices are hardwired to the system buses. The few pins that are different among the various devices are connected to circuitry that can easily be modified to configure the system for the proper size and type of memory device. This allows the user to reconfigure the hardware without needing to cut or solder on the board. The configuration circuitry can be as simple as movable jumpers or DIP switches that the user sets up and as complicated as an in-circuit programmable logic device that the computer can set up or modify to meet the system requirements.

REVIEW QUESTIONS	1. How does a static-RAM cell differ from a dynamic-RAM cell?
	2. Which memory technology generally uses the least power?
	3. What device places data on the data bus during a read cycle?
	4. What device places data on the data bus during a write cycle?
	5. What RAM timing parameters determine its operating speed?
	6. <i>True or false:</i> A LOW at \overline{OE} will enable the output buffers of an MCM6264C provided that both chip select inputs are active.

7. What must be done with pin 26 and pin 27 if a 27256 is replaced with a 2764?

12-13 DYNAMIC RAM (DRAM)

Dynamic RAMs are fabricated using MOS technology and are noted for their high capacity, low power requirement, and moderate operating speed. As we stated earlier, unlike static RAMs, which store information in FFs, dynamic RAMs store 1s and 0s as charges on a small MOS capacitor (typically a few picofarads). Because of the tendency for these charges to leak off after a period of time, dynamic RAMs require periodic recharging of the memory cells; this is called refreshing the dynamic RAM. In modern DRAM chips, each memory cell must be refreshed typically every 2, 4, or 8 ms, or its data will be lost.

The need for refreshing is a drawback of dynamic RAM compared to static RAM because it may require external support circuitry. Some DRAM chips have built-in refresh control circuitry that does not require extra external hardware but does require special timing of the chip's input control signals. Additionally, as we shall see, the address inputs to a DRAM must be handled in a less straightforward way than SRAM. So, all in all, designing with and using DRAM in a system is more complex than with SRAM. However, their much larger capacities and much lower power consumption make DRAMs the memory of choice in systems where the most important design considerations are keeping down size, cost, and power.

For applications where speed and reduced complexity are more critical than cost, space, and power considerations, static RAMs are still the best. They are generally faster than dynamic RAMs and require no refresh operation. They are simpler to design with, but they cannot compete with the higher capacity and lower power requirement of dynamic RAMs.

Because of their simple cell structure, DRAMs typically have four times the density of SRAMs. This increased density allows four times as much memory capacity to be placed on a single board; alternatively, it requires onefourth as much board space for the same amount of memory. The cost per bit of dynamic RAM storage is typically one-fifth to one-fourth that of static RAMs. An additional cost saving is realized because the lower power requirements of a dynamic RAM, typically one-sixth to one-half those of a static RAM, allow the use of smaller, less expensive power supplies.

The main applications of SRAMs are in areas where only small amounts of memory are needed or where high speed is required. Many microprocessorcontrolled instruments and appliances have very small memory capacity requirements. Some instruments, such as digital storage oscilloscopes and logic analyzers, require very high-speed memory. For applications such as these, SRAM is normally used. The main internal memory of most personal microcomputers (e.g., Windowsbased PCs or Macs) uses DRAM because of its high capacity and low power consumption. These computers, however, sometimes use some small amounts of SRAM for functions requiring maximum speed, such as video graphics, look-up tables, and cache memory.

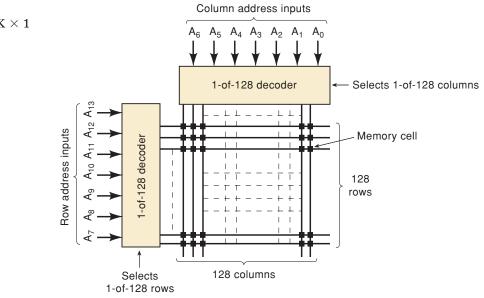
REVIEW QUESTIONS

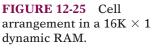
- 1. What are the main drawbacks of dynamic RAM compared with static?
- 2. List the advantages of dynamic RAM compared with static RAM.
- 3. Which type of RAM would you expect to find on the main memory modules of your PC?

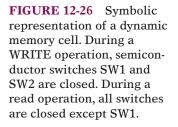
12-14 DYNAMIC RAM STRUCTURE AND OPERATION

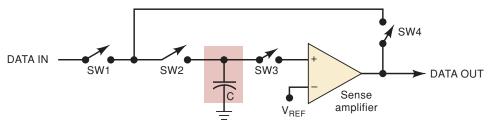
The dynamic RAM's internal architecture can be visualized as an array of single-bit cells, as illustrated in Figure 12-25. Here, 16,384 cells are arranged in a 128 × 128 array. Each cell occupies a unique row and column position within the array. Fourteen address inputs are needed to select one of the cells ($2^{14} = 16,384$); the lower address bits, A_0 to A_6 , select the column, and the higher-order bits, A_7 to A_{13} , select the row. Each 14-bit address selects a unique cell to be written into or read from. The structure in Figure 12-25 is a 16K × 1 DRAM chip. DRAM chips are currently available in various configurations. DRAMs with a four-bit (or greater) word size have a cell arrangement similar to that of Figure 12-25 except that each position in the array contains four cells, and each applied address selects a group of four cells for a read or a write operation. As we will see later, larger word sizes can also be attained by combining several chips in the appropriate arrangement.

Figure 12-26 is a symbolic representation of a dynamic memory cell and its associated circuitry. Many of the circuit details are not shown, but this simplified diagram can be used to describe the essential ideas involved in









writing to and reading from a DRAM. The switches SW1 through SW4 are actually MOSFETs that are controlled by various address decoder outputs and the R/\overline{W} signal. The capacitor, of course, is the actual storage cell. One sense amplifier would serve an entire column of memory cells, but operate only on the bit in the selected row.

To write data to the cell, signals from the address decoding and read/write logic will close switches SW1 and SW2, while keeping SW3 and SW4 open. This connects the input data to C. A logic 1 at the data input charges C, and a logic 0 discharges it. Then the switches are open so that C is disconnected from the rest of the circuit. Ideally, C would retain its charge indefinitely, but there is always some leakage path through the off switches, so that C will gradually lose its charge.

To read data from the cell, switches SW2, SW3, and SW4 are closed, and SW1 is kept open. This connects the stored capacitor voltage to the *sense amplifier*. The sense amplifier compares the voltage with some reference value to determine if it is a logic 0 or 1, and it produces a solid 0 V or 5 V for the data output. This data output is also connected to C (SW2 and SW4 are closed) and refreshes the capacitor voltage by recharging or discharging. In other words, the data bit in a memory cell is refreshed each time it is read.

Address Multiplexing

The $16K \times 1$ DRAM array depicted in Figure 12-25 is obsolete and nearly unavailable. It has 14 address inputs; a $64K \times 1$ DRAM array would have 16 address inputs. A $1M \times 4$ DRAM needs 20 address inputs; a $4M \times 1$ needs 22 address inputs. High-capacity memory chips such as these would require many pins if each address input required a separate pin. In order to reduce the number of pins on their high-capacity DRAM chips, manufacturers utilize **address multiplexing** whereby each address input pin can accommodate two different address bits. The saving in pin count translates to a significant decrease in the size of the IC packages. This is very important in large-capacity memory boards, where you want to maximize the amount of memory that can fit on one board.

In the discussions that follow, we will be describing the order in which the address is multiplexed into the DRAM chips. It should be noted that in older, small-capacity DRAMs, the convention was to present the low-order address first specifying the row, followed by the high-order address specifying the column. The newer DRAMs and the controllers that perform the multiplexing use the opposite convention of applying the high-order bits as the row address and then the low-order bits as the column address. We will describe the more recent convention, but you should be aware of this change as you investigate older systems.

We will use the TMS44100 $4M \times 1$ DRAM from Texas Instruments to illustrate the operation of DRAM chips today. The functional block diagram of this chip's internal architecture (shown in Figure 12-27) is typical of

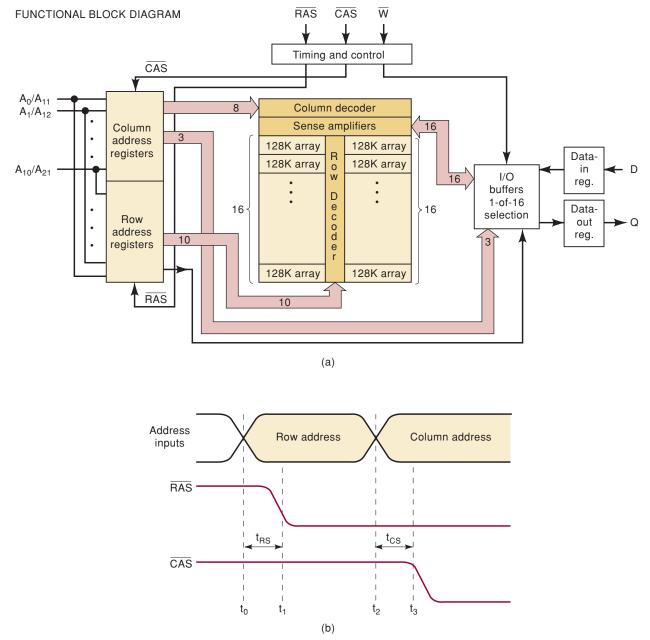


FIGURE 12-27 (a) Simplified architecture of the TMS44100 4M \times 1 DRAM; (b) $\overline{RAS}/\overline{CAS}$ timing. (Reprinted by permission of Texas Instruments)

diagrams you will find in data books. The layout of the memory array in this diagram may appear complicated at first glance, but it can be thought of as just a bigger version of the $16K \times 1$ DRAM in Figure 12-25. Functionally, it is an array of cells arranged as 2048 rows by 2048 columns. A single row is selected by address decoder circuitry that can be thought of as a 1-of-2048 decoder. Likewise, a single column is selected by what is effectively a 1-of-2048 decoder. Because the address lines are multiplexed, the entire 22-bit address cannot be presented simultaneously. Notice that there are only 11 address lines and that they go to both the row and the column address registers. Each of the two address registers stores half of the 22-bit address. The row register stores the upper half, and the column register stores the

lower half. Two very important strobe inputs control when the address information is latched. The row address strobe (\overline{RAS}) clocks the 11-bit row address register. The column address strobe (\overline{CAS}) clocks the 11-bit column address register.

A 22-bit address is applied to this DRAM in two steps using \overline{RAS} and \overline{CAS} . The timing is shown in Figure 12-27(b). Initially, \overline{RAS} and \overline{CAS} are both HIGH. At time t_0 , the 11-bit row address (A_{11} to A_{21}) is applied to the address inputs. After allowing time for the setup time requirement (t_{RS}) of the row address register, the \overline{RAS} input is driven LOW at t_1 . This NGT loads the row address into the row address register so that A_{11} to A_{21} now appear at the row decoder inputs. The LOW at \overline{RAS} also enables this decoder so that it can decode the row address and select one row of the array.

At time t_2 , the 11-bit column address (A_0 to A_{10}) is applied to the address inputs. At t_3 , the \overline{CAS} input is driven LOW to load the column address into the column address register. \overline{CAS} also enables the column decoder so that it can decode the column address and select one column of the array.

At this point the two parts of the address are in their respective registers, the decoders have decoded them to select the one cell corresponding to the row and column address, and a read or a write operation can be performed on that cell just as in a static RAM.

You may have noticed that this DRAM does not have a chip select (*CS*) input. The \overline{RAS} and \overline{CAS} signals perform the chip select function because they must both be LOW for the decoders to select a cell for reading or writing.

As you can see, there are several operations that must be performed before the data that is stored in the DRAM can actually appear on the outputs. The term **latency** is often used to describe the time required to perform these operations. Each operation takes a certain amount of time, and this amount of time determines the maximum rate at which we can access data in the memory.

EXAMPLE 12-12

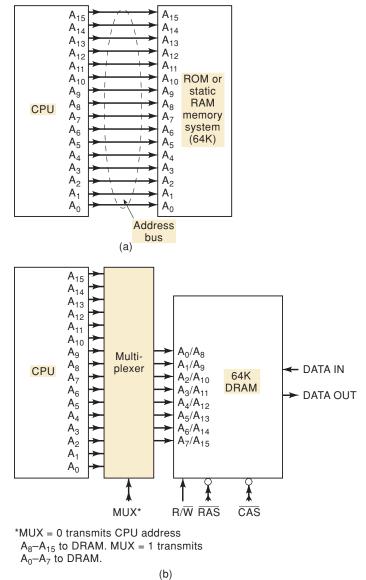
How many pins are saved by using address multiplexing for a $16M \times 1$ DRAM?

Solution

Twelve address inputs are used instead of 24; \overline{RAS} and \overline{CAS} are added; no \overline{CS} is required. Thus, there is a net saving of *eleven* pins.

In a simple computer system, the address inputs to the memory system come from the central processing unit (CPU). When the CPU wants to access a particular memory location, it generates the complete address and places it on address lines that make up an address bus. Figure 12-28(a) shows this for a small computer memory that has a capacity of 64K words and therefore requires a 16-line address bus going directly from the CPU to the memory.

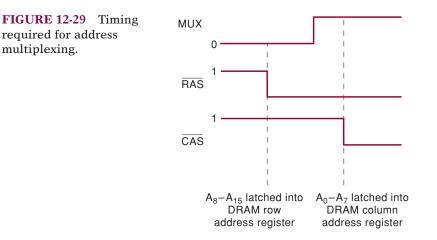
This arrangement works for ROM or for static RAM, but it must be modified for DRAM that uses multiplexed addressing. If all 64K of the memory is DRAM, it will have only eight address inputs. This means that the 16 address lines from the CPU address bus must be fed into a FIGURE 12-28 (a) CPU address bus driving ROM or static-RAM memory; (b) CPU addresses driving a multiplexer that is used to multiplex the CPU address lines into the DRAM.



multiplexer circuit that will transmit eight address bits at a time to the memory address inputs. This is shown symbolically in Figure 12-28(b). The multiplexer select input, labeled MUX, controls whether CPU address lines A_0 to A_7 or address lines A_8 to A_{15} will be present at the DRAM address inputs.

The timing of the *MUX* signal must be synchronized with the *RAS* and \overline{CAS} signals that clock the addresses into the DRAM. This is shown in Figure 12-29. *MUX* must be LOW when \overline{RAS} is pulsed LOW so that address lines A_8 to A_{15} from the CPU will reach the DRAM address inputs to be loaded on the NGT of \overline{RAS} . Likewise, *MUX* must be HIGH when \overline{CAS} is pulsed LOW so that A_0 to A_7 from the CPU will be present at the DRAM inputs to be loaded on the NGT of \overline{CAS} .

The actual multiplexing and timing circuitry will not be shown here but will be left to the end-of-chapter problems (Problems 12-26 and 12-27).



REVIEW QUESTIONS

- 1. Describe the array structure of a 64K \times 1 DRAM.
- 2. What is the benefit of address multiplexing?
- 3. How many address inputs would there be on a 1M imes 1 DRAM chip?
- 4. What are the functions of the RAS and CAS signals?
- 5. What is the function of the *MUX* signal?

12-15 DRAM READ/WRITE CYCLES

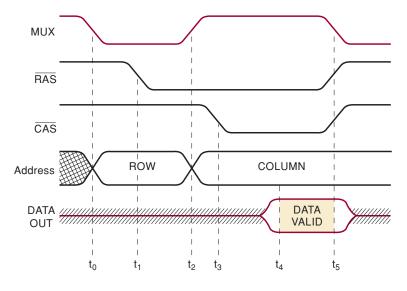
The timing of the read and write operations of a DRAM is much more complex than for a static RAM, and there are many critical timing requirements that the DRAM memory designer must consider. At this point, a detailed discussion of these requirements would probably cause more confusion than enlightenment. We will concentrate on the basic timing sequence for the read and write operations for a small DRAM system like that of Figure 12-28(b).

Dram Read Cycle

Figure 12-30 shows typical signal activity during the read operation. It is assumed that R/\overline{W} is in its HIGH state throughout the operation. The following is a step-by-step description of the events that occur at the times indicated on the diagram.

- t₀: MUX is driven LOW to apply the row address bits (A₈ to A₁₅) to the DRAM address inputs.
- t_1 : *RAS* is driven LOW to load the row address into the DRAM.
- t_2 : *MUX* goes HIGH to place the column address (A_0 to A_7) at the DRAM address inputs.
- t_3 : *CAS* goes LOW to load the column address into the DRAM.
- t₄: The DRAM responds by placing valid data from the selected memory cell onto the DATA OUT line.
- t_5 : MUX, \overline{RAS} , \overline{CAS} , and DATA OUT return to their initial states.

FIGURE 12-30 Signal activity for a read operation on a dynamic RAM. The R/\overline{W} input (not shown) is assumed to be HIGH.



Dram Write Cycle

Figure 12-31 shows typical signal activity during a DRAM write operation. Here is a description of the sequence of events.

- t_0 : The LOW at *MUX* places the row address at the DRAM inputs.
- t_1 : The NGT at \overline{RAS} loads the row address into the DRAM.
- t_2 : *MUX* goes HIGH to place the column address at the DRAM inputs.
- t_3 : The NGT at \overline{CAS} loads the column address into the DRAM.
- t_4 : Data to be written are placed on the DATA IN line.
- t_5 : R/\overline{W} is pulsed LOW to write the data into the selected cell.
- t_6 : Input data are removed from DATA IN.
- t_7 : MUX, \overline{RAS} , \overline{CAS} , and R/\overline{W} are returned to their initial states.

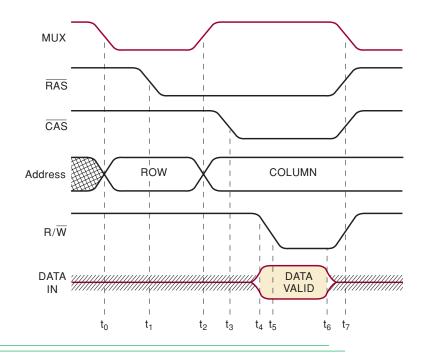


FIGURE 12-31 Signal activity for a write operation on a dynamic RAM.

REVIEW QUESTIONS	1. True or false:
	(a) During a read cycle, the \overline{RAS} signal is activated before the \overline{CAS} signal.
	(b) During a write operation, \overline{CAS} is activated before \overline{RAS} .
	(c) R/\overline{W} is held LOW for the entire write operation.
	(d) The address inputs to a DRAM will change twice during a read or a write operation.
	2. Which signal in Figure 12-28(b) makes sure that the correct portion of the complete address appears at the DRAM inputs?

12-16 DRAM REFRESHING

A DRAM cell is refreshed each time a read operation is performed on that cell. Each memory cell must be refreshed periodically (typically, every 4 to 16 ms, depending on the device) or its data will be lost. This requirement would appear to be extremely difficult, if not impossible, to meet for large-capacity DRAMs. For example, a $1M \times 1$ DRAM has $10^{20} = 1,048,576$ cells. To ensure that each cell is refreshed within 4 ms, it would require that read operations be performed on successive addresses at the rate of one every 4 ns (4 ms/1,048,576 \approx 4 ns). This is much too fast for any DRAM chip. Fortunately, manufacturers have designed DRAM chips so that

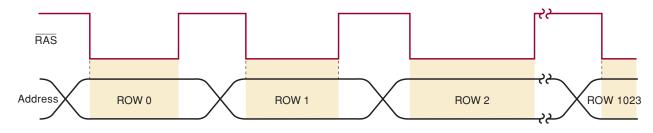
whenever a read operation is performed on a cell, all of the cells in that row will be refreshed.

Thus, it is necessary to do a read operation only on each *row* of a DRAM array once every 4 ms to guarantee that each *cell* of the array is refreshed. Referring to the $4M \times 1$ DRAM of Figure 12-27(a), if any address is strobed into the row address register, all 2048 cells in that row are automatically refreshed.

Clearly, this row-refreshing feature makes it easier to keep all DRAM cells refreshed. However, during the normal operation of the system in which a DRAM is functioning, it is unlikely that a read operation will be performed on each row of the DRAM within the required refresh time limit. Therefore, some kind of refresh control logic is needed either external to the DRAM chip or as part of its internal circuitry. In either case, there are two refresh modes: a *burst* refresh and a *distributed* refresh.

In a burst refresh mode, the normal memory operation is suspended, and each row of the DRAM is refreshed in succession until all rows have been refreshed. In a distributed refresh mode, the row refreshing is interspersed with the normal operations of the memory.

The most universal method for refreshing a DRAM is the \overline{RAS} -only refresh. It is performed by strobing in a row address with \overline{RAS} while \overline{CAS} and R/\overline{W} remain HIGH. Figure 12-32 illustrates how \overline{RAS} -only refresh is used for a burst refresh of the TMS44100. Some of the complexity of the memory array in this chip is there to make refresh operations simpler. Because two banks are lined up in the same row, both banks can be refreshed at the same time, effectively making it the same as if there were only 1024 rows. A **refresh counter** is used to supply 10-bit row addresses to the DRAM address inputs starting at 000000000 (row 0). \overline{RAS} is pulsed LOW to load this address into the DRAM, and this refreshes row 0 in both banks. The counter is incremented and the process is repeated up to address 111111111 (row 1023). For the TMS44100, a burst refresh can be completed in just over 113 μ s and must be repeated every 16 ms or less.



* R/\overline{W} and \overline{CAS} lines held HIGH

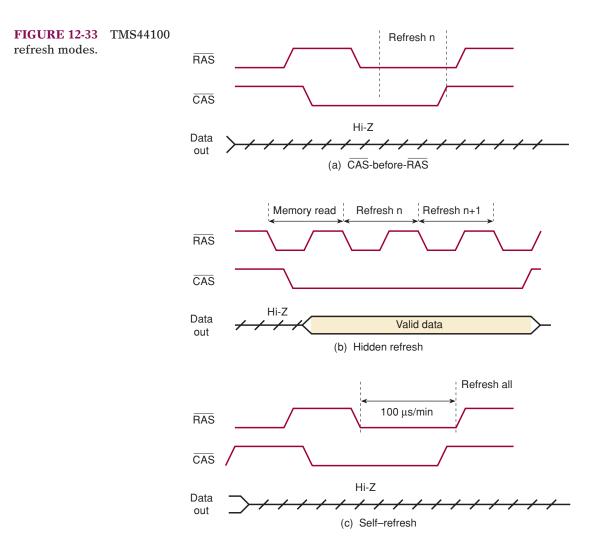
FIGURE 12-32 The \overline{RAS} -only refresh method uses only the \overline{RAS} signal to load the row address into the DRAM to refresh all cells in that row. The \overline{RAS} -only refresh can be used to perform a burst refresh as shown. A refresh counter supplies the sequential row addresses from row 0 to row 1023 (for a $4M \times 1$ DRAM).

While the refresh counter idea seems easy enough, we must realize that the row addresses from the refresh counter cannot interfere with the addresses coming from the CPU during normal read/write operations. For this reason, the refresh counter addresses must be multiplexed with the CPU addresses, so that the proper source of DRAM addresses is activated at the proper times.

In order to relieve the computer's CPU from some of these burdens, a special chip called a dynamic RAM (DRAM) controller is often used. At a minimum, this chip will perform address multiplexing and refresh count sequence generation, leaving the generation of the timing for RAS, CAS, and MUX signals up to some other logic circuitry and the person who programs the computer. Other DRAM controllers are fully automatic. Their inputs look very much like a static RAM or ROM. They automatically generate the refresh sequence often enough to maintain the memory, multiplex the address bus, generate the RAS and CAS signals, and arbitrate control of the DRAM between the CPU read/write cycles and local refresh operations. In current personal computers, the DRAM controller and other high-level controller circuits are integrated into a set of VLSI circuits that are referred to as a "chip set." As newer DRAM technologies are developed, new chip sets are designed to take advantage of the latest advances. In many cases, the number of existing (or anticipated) chip sets supporting a certain technology in the market determines the DRAM technology in which manufacturers invest.

Most of the DRAM chips in production today have on-chip refreshing capability that eliminates the need to supply external refresh addresses. One of these methods, shown in Figure 12-33(a), is called \overline{CAS} -before- \overline{RAS} refresh. In this method, the \overline{CAS} signal is driven LOW first and is held LOW until after \overline{RAS} goes LOW. This sequence will refresh one row of the memory array and increment an internal counter that generates the row addresses. To perform a burst refresh using this feature, \overline{CAS} can be held LOW while \overline{RAS} is pulsed once for each row until all are refreshed. During this refresh cycle, all external addresses are ignored. The TMS44100 also offers "hidden refresh," which allows a row to be refreshed while holding data on the output. This is done by holding \overline{CAS} LOW after a read cycle and then pulsing \overline{RAS} as in Figure 12-33(b).

The self-refresh mode of Figure 12-33(c) fully automates the process. By forcing \overline{CAS} LOW before \overline{RAS} and then holding them both LOW for at least 100 μ s, an internal oscillator clocks the row address counter until all cells are refreshed. The mode that a system designer chooses depends on how busy the computer's CPU is. If it can spare 100 μ s without accessing its memory, and if it can do this every 16 ms, then the self-refresh is the way to go.



However, if this will slow the program execution down too much, it may require some distributed refreshing using \overline{CAS} -before- \overline{RAS} or hidden refresh cycles. In any case, all cells must be refreshed within the allotted time or data will be lost.

REVIEW QUESTIONS	1. True or false:
	(a) In most DRAMs, it is necessary to read only from one cell in each row in order to refresh all cells in that row.
	(b) In the burst refresh mode, the entire array is refreshed by one \overline{RAS} pulse.
	2. What is the function of a refresh counter?
	3. What functions does a DRAM controller perform?
	4. True or false:
	(a) In the \overline{RAS} -only refresh method, the \overline{CAS} signal is held LOW.
	(b) \overline{CAS} -before- \overline{RAS} refresh can be used only by DRAMs with on-chip refresh control circuitry.

12-17 DRAM TECHNOLOGY*

In selecting a particular type of RAM device for a system, a designer has some difficult decisions. The capacity (as large as possible), the speed (as fast as possible), the power needed (as little as possible), the cost (as low as possible), and the convenience (as easy to change as possible) must all be kept in a reasonable balance because no single type of RAM can maximize all of these desired features. The semiconductor RAM market is constantly trying to produce the ideal mix of these characteristics in its products for various applications. This section explains some of the current terms used regarding RAM technology. This is a very dynamic topic, and perhaps some of the art today.

Memory Modules

With many companies manufacturing motherboards for personal computer systems, standard memory interface connectors have been adopted. These connectors receive a small printed circuit card with contact points on both sides of the edge of the card. These modular cards allow for easy installation or replacement of memory components in a computer. The single-in-line memory module (SIMM) is a circuit card with 72 functionally equivalent contacts on both sides of the card. A redundant contact point on each side of the board offers some assurance that a good, reliable contact is made. These modules use 5-V-only DRAM chips that vary in capacity from 1 to 16 Mbits in surface-mount gull-wing or J-lead packages. The memory modules vary in capacity from 1 to 32 Mbytes.

The newer, 168-pin, dual-in-line memory module (DIMM) has 84 functionally unique contacts on each side of the card. The extra pins are necessary because DIMMs are connected to 64-bit data buses such as those found in modern PCs. Both 3.3-V and 5-V versions are available. They also come in buffered and unbuffered versions. The capacity of the module depends on the DRAM chips that are mounted on it; and as DRAM capacity increases, the capacity of the DIMMs will increase. The chip set and motherboard design that is used in any given system determines which type of DIMM can be used. For compact applications, such as laptop computers, a small-outline, dual-in-line memory module is available (SODIMM).

The primary problem in the personal computer industry is providing a memory system that is fast enough to keep up with the ever-increasing clock speeds of the microprocessors while keeping the cost at an affordable level. Special features are being added to the basic DRAM devices to enhance their total bandwidth. A new type of package called the RIMM has entered the market. RIMM stands for Rambus In-line Memory Module. Rambus is a company that has invented some revolutionary new approaches to memory technology. The RIMM is their proprietary package that holds their proprietary memory chips called Direct Rambus DRAM (DRDRAM) chips. Although these methods of improving performance are constantly changing, the technologies described in the following sections are currently being referred to extensively in memory-related literature.

FPM DRAM

Fast page mode (FPM) allows quicker access to random memory locations within the current "page." A page is simply a range of memory addresses

^{*}This topic may be omitted without affecting the continuity of the remainder of the book.

that have identical upper address bit values. In order to access data on the current page, only the lower address lines must be changed.

EDO DRAM

Extended data output (EDO) DRAMs offer a minor improvement to FPM DRAMs. For accesses on a given page, the data value at the current memory location is sensed and latched onto the output pins. In the FPM DRAMs, the sense amplifier drives the output without a latch, requiring \overline{CAS} to remain low until data values become valid. With EDO, while these data are present on the outputs, \overline{CAS} can complete its cycle, a new address on the current page can be decoded, and the data path circuitry can be reset for the next access. This allows the memory controller to be outputting the next address at the same time that the current word is being read.

SDRAM

Synchronous DRAM is designed to transfer data in rapid-fire *bursts* of several sequential memory locations. The first location to be accessed is the slowest due to the overhead (latency) of latching the row and column address. Thereafter, the data values are clocked out by the bus system clock (instead of the \overline{CAS} control line) in bursts of memory locations within the same page. Internally, SDRAMs are organized in two banks. This allows data to be read out at a very fast rate by alternately accessing each of the two banks. In order to provide all of the features and the flexibility needed for this type of DRAM to work with a wide variety of system requirements, the circuitry within the SDRAM has become more complex. A command sequence is necessary to tell the SDRAM which options are needed, such as burst length, sequential or interleaved data, and \overline{CAS} -before- \overline{RAS} or self-refresh mode allows the memory device to perform all of the necessary functions to keep its cells refreshed.

DDRSDRAM

Double Data Rate SDRAM offers an improvement of SDRAM. In order to speed up the operation of SDRAM, while operating from a synchronous system clock, this technology transfers data on the rising and falling edges of the system clock, effectively doubling the potential rate of data transfer.

SLDRAM

Synchronous-Link DRAM is an evolutionary improvement over DDRS-DRAM. It can operate at bus speeds up to 200 MHz and clocks data synchronously on the rising and falling edges of the system clock. A consortium of several DRAM manufacturers is developing it as an open standard. If chip sets are developed that can take advantage of these memory devices and enough system designers adopt this technology, it is likely to become a widely used form of DRAM.

DRDRAM

Direct Rambus DRAM is a proprietary device developed and marketed by Rambus, Inc. It uses a revolutionary new approach to DRAM system architecture with much more control integrated into the memory device. This technology is still battling with the other standards to find its niche in the market.

REVIEW QUESTIONS	1. Are SIMMs and DIMMs interchangeable?
	2. What is a "page" of memory?
	3. Why is "page mode" faster?
	4. What does <i>EDO</i> stand for?
	5. What term is used for accessing several consecutive memory locations?
	6. What is an SDRAM synchronized to?

12-18 EXPANDING WORD SIZE AND CAPACITY

In many memory applications, the required RAM or ROM memory capacity or word size cannot be satisfied by one memory chip. Instead, several memory chips must be combined to provide the capacity and/or the word size. We will see how this is done through several examples that illustrate the important ideas that are used when memory chips are interfaced to a microprocessor. The examples that follow are intended to be instructive, and the memory chip sizes that are used were chosen to conserve space. The techniques that are presented can be extended to larger memory chips.

Expanding Word Size

Suppose that we need a memory that can store 16 eight-bit words and all we have are RAM chips that are arranged as 16×4 memories with common I/O lines. We can combine two of these 16×4 chips to produce the desired memory. The configuration for doing so is shown in Figure 12-34. Examine this diagram carefully and see what you can find out from it before reading on.

Because each chip can store 16 four-bit words and we want to store 16 eight-bit words, we are using each chip to store *half* of each word. In other words, RAM-0 stores the four *higher*-order bits of each of the 16 words, and RAM-1 stores the four *lower*-order bits of each of the 16 words. A full eight-bit word is available at the RAM outputs connected to the data bus.

Any one of the 16 words is selected by applying the appropriate address code to the four-line *address bus* (A_3, A_2, A_1, A_0) . The address lines typically originate at the CPU. Note that each address bus line is connected to the corresponding address input of each chip. This means that once an address code is placed on the address bus, this same address code is applied to both chips so that the same location in each chip is accessed at the same time.

Once the address is selected, we can read or write at this address under control of the common R/\overline{W} and \overline{CS} line. To read, R/\overline{W} must be high and \overline{CS} must be low. This causes the RAM I/O lines to act as *outputs*. RAM-0 places its selected four-bit word on the upper four data bus lines, and RAM-1 places its selected four-bit word on the lower four data bus lines. The data bus then contains the full selected eight-bit word, which can now be transmitted to some other device (usually a register in the CPU).

To write, $R/\overline{W} = 0$ and $\overline{CS} = 0$ causes the RAM I/O lines to act as *inputs*. The eight-bit word to be written is placed on the data bus (usually by the CPU). The higher four bits will be written into the selected location of RAM-0, and the lower four bits will be written into RAM-1.

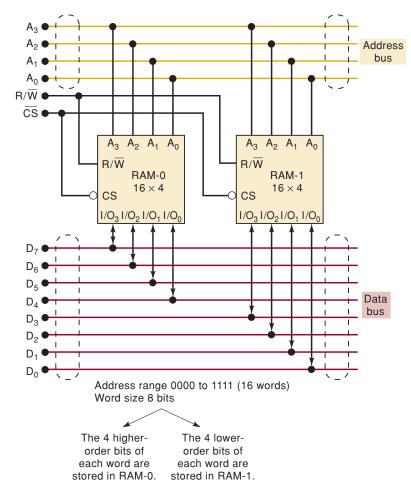


FIGURE 12-34 Combining two 16×4 RAMs for a 16×8 module.

In essence, the combination of the two RAM chips acts like a single 16 \times 8 memory chip. We refer to this combination as a 16 \times 8 memory module.

The same basic idea for expanding word size will work for many different situations. Read the following example and draw a rough diagram for what the system will look like before looking at the solution.

EXAMPLE 12-13

The 2125A is a static-RAM IC that has a capacity of $1K \times 1$, one active-LOW chip select input, and separate data input and output. Show how to combine several 2125A ICs to form a $1K \times 8$ module.

Solution

The arrangement is shown in Figure 12-35, where eight 2125A chips are used for a $1K \times 8$ module. Each chip stores one of the bits of each of the 1024 eight-bit words. Note that all of the R/\overline{W} and \overline{CS} inputs are wired together, and the 10-line address bus is connected to the address inputs of each chip. Also note that because the 2125A has separate data in and data out pins, both of these pins of each chip are tied to the same data bus line.

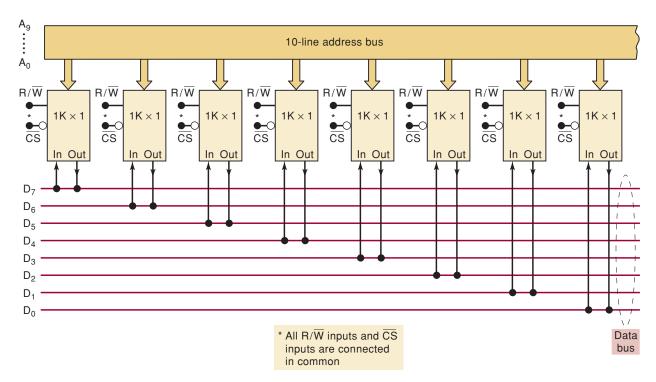


FIGURE 12-35 Eight 2125A 1K \times 1 chips arranged as a 1K \times 8 memory.

Expanding Capacity

Suppose that we need a memory that can store 32 four-bit words and all we have are the 16 \times 4 chips. By combining two 16 \times 4 chips as shown in Figure 12-36, we can produce the desired memory. Once again, examine this diagram and see what you can determine from it before reading on.

Each RAM is used to store 16 four-bit words. The four data I/O pins of each RAM are connected to a common four-line data bus. Only one of the RAM chips can be selected (enabled) at one time so that there will be no buscontention problems. This is ensured by driving the respective \overline{CS} inputs from different logic signals.

The total capacity of this memory module is 32×4 , so there must be 32 different addresses. This requires *five* address bus lines. The upper address line A_4 is used to select one RAM or the other (via the \overline{CS} inputs) as the one that will be read from or written into. The other four address lines A_0 to A_3 are used to select the one memory location out of 16 from the selected RAM chip.

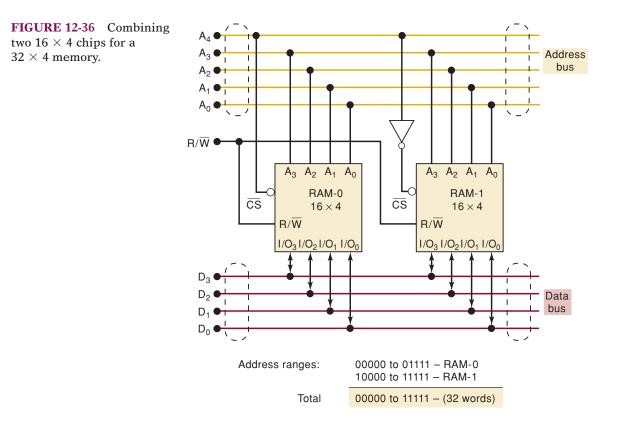
To illustrate, when $A_4 = 0$, the \overline{CS} of RAM-0 enables this chip for read or write. Then any address location in RAM-0 can be accessed by A_3 through A_0 . The latter four address lines can range from 0000 to 1111 to select the desired location. Thus, the range of addresses representing locations in RAM-0 is

$$A_4 A_3 A_2 A_1 A_0 = 00000$$
 to 01111

Note that when $A_4 = 0$, the \overline{CS} of RAM-1 is high, so that its I/O lines are disabled (Hi-Z) and cannot communicate with (give data to or take data from) the data bus.

It should be clear that when $A_4 = 1$, the roles of RAM-0 and RAM-1 are reversed. RAM-1 is now enabled, and the lines A_3 to A_0 select one of its locations. Thus, the range of addresses located in RAM-1 is

$$A_4 A_3 A_2 A_1 A_0 = 10000$$
 to 11111



EXAMPLE 12-14

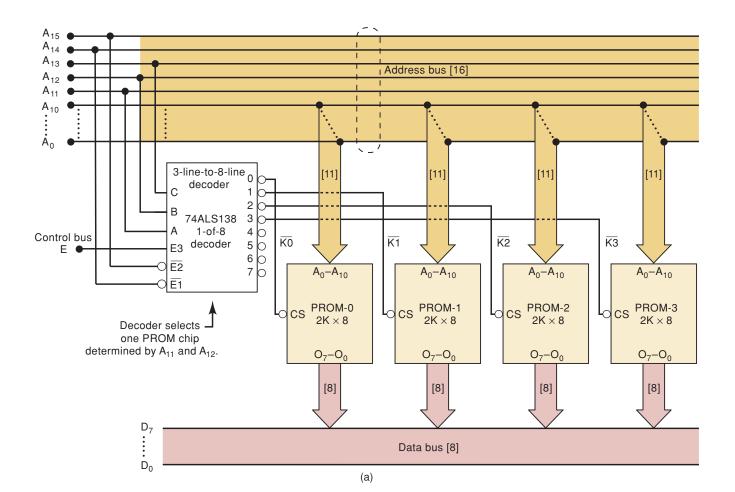
We want to combine several $2K \times 8$ PROMs to produce a total capacity of $8K \times 8$. How many PROM chips are needed? How many address bus lines are required?

Solution

Four PROM chips are required, with each one storing 2K of the 8K words. Because $8K = 8 \times 1024 = 8192 = 2^{13}$, *thirteen* address lines are needed.

The configuration for the memory of Example 12-14 is similar to the 32×4 memory of Figure 12-36. It is slightly more complex, however, because it requires a decoder circuit for generating the \overline{CS} input signals. The complete diagram for this 8192×8 memory is shown in Figure 12-37(a).

The total capacity of the block of ROM is 8192 bytes. This system containing the block of memory has an address bus of 16 bits, which is typical of a small microcontroller-based system. The decoder in this system can only be enabled when A_{15} and A_{14} are LOW and *E* is HIGH. This means that it can only decode addresses less than 4000 hex. It is easier to understand this by looking at the memory map of Figure 12-37(b). You can see that the top two MSBs (in red) are always LOW for addresses under 4000 hex. Address lines A_{13} – A_{11} (blue font) are connected to decoder inputs *C*–*A*, respectively. These three bits are decoded and used to select one of the memory ICs. Notice in the bit map of Figure 12-37(b) that all the addresses that are contained in PROM-0 have A_{13} , A_{12} , $A_{11} = 0$, 0, 0; PROM-1 is selected when these bits have a value of 0, 0, 1; PROM-2 when 0, 1, 0; and PROM-3 when 0, 1, 1. When any PROM is selected, the address lines A_{10} – A_0 can range from all 0s to all 1s. To summarize the address scheme of this system, the top two bits are used to select this decoder,



A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address	Syst	em Map	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000]
																		PROM-0	2K
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FF			
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0800			
																		PROM-1	2K
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF			
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000			
																		PROM-2	2K
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	17FF			
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1800			
																		PROM-3	2K
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF			
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000	O ₄		
		1	0	1													O ₅	Decoded	8K
		1	1	0													O ₆	Expansion	
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF	07		
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000			1
																		Available	48K
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF			

(b)

FIGURE 12-37 (a) Four $2K \times 8$ PROMs arranged to form a total capacity of $8K \times 8$. (b) Memory map of the full system.

the next three bits $(A_{13}-A_{11})$ are used to select one out of four PROM chips, and the lower 11 address lines are used to select one out of 2048 byte-sized memory locations in the enabled PROM.

When the system address of 4000 or more is on the address bus, none of the PROMs will be enabled. However, decoder outputs 4–7 can be used to enable more memory chips if we wish to expand the capacity of the memory system. The memory map on the right side of Figure 12-37(b) shows a 48K area of the system's space that is not occupied by this memory block. In order to expand into this area of the memory map, more decoding logic would be needed.

EXAMPLE 12-15

What would be needed to expand the memory of Figure 12-37 to $32K \times 8$? Describe what address lines are used.

Solution

A 32K capacity will require 16 of the 2K PROM chips. Four are already shown and four more can be connected to the O_4-O_7 of the existing decoder outputs. This accounts for half of the system. The other eight PROM chips can be selected by adding another 74ALS138 decoder and enabling it only when $A_{15} = 0$ and $A_{14} = 1$. This can be accomplished by connecting an inverter between A_{14} and $\overline{E_1}$ while connecting A_{15} directly to $\overline{E_2}$. The other connections are the same as in the existing decoder.

Incomplete Address Decoding

In many instances, it is necessary to use various memory devices in the same memory system. For example, consider the requirements of a digital dashboard system on an automobile. Such a system is typically implemented using a microprocessor. Consequently, we need some nonvolatile ROM to store the program instructions. We need some read/write memory to store the digits that represent the speed, RPM, gallons of fuel, and so on. Other digitized values must be stored to represent oil pressure, engine temperature, battery voltage, and so on. We also need some nonvolatile read/write storage (EE-PROM) for the odometer readout because it would not be good to have this number reset to 0 or assume a random value whenever the car battery is disconnected.

Figure 12-38 shows a memory system that could be used in a microcomputer system. Notice that the ROM portion is made up of two $8K \times 8$ devices (PROM-0 and PROM-1). The RAM section requires a single $8K \times 8$ device. The EEPROM available is only a $2K \times 8$ device. The memory system requires a decoder to select only one device at a time. This decoder divides the entire memory space (assuming 16 address bits) into 8K address blocks. In other words, each decoder output is activated by 8192 (8K) different addresses. Notice that the upper three address lines control the decoder. The 13 lowerorder address lines are tied directly to the address inputs on the memory chips. The only exception to this is the EEPROM, which has only 11 address lines for its 2-Kbyte capacity. If the address (in hex) of this EEPROM is intended to range from 6000 to 67FF, it will respond to these addresses as intended. However, the two address lines, A_{11} and A_{12} , are not involved in the decoding scheme for this chip. The decoder output (K3) is active for 8K addresses, but the chip that it is connected to contains only 2K locations. As a result, the EEPROM will also respond to the other 6K of addresses in this

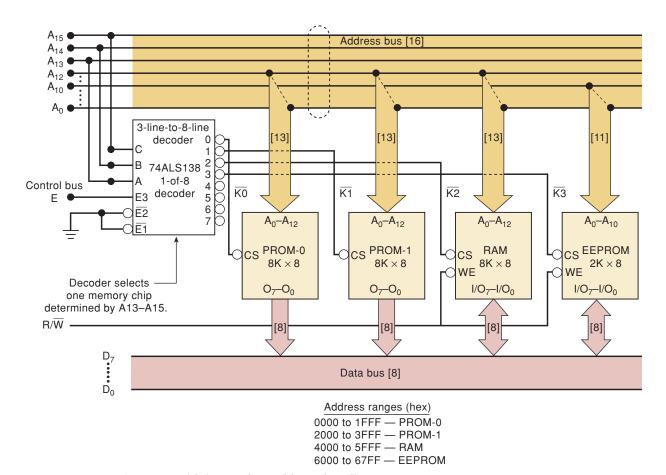
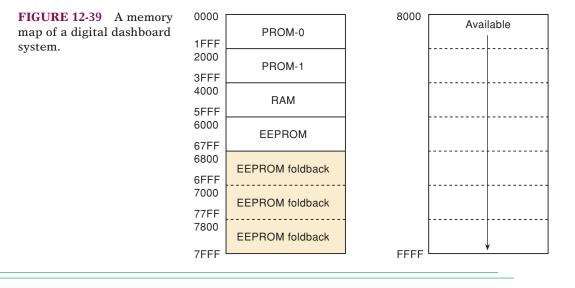


FIGURE 12-38 A system with incomplete address decoding.

decoded block of memory. The same contents of the EEPROM will also appear at addresses 6800–6FFF, 7000–77FF, and 7800–7FFF. These areas of memory that are redundantly occupied by a device due to incomplete address decoding are referred to as **memory foldback** areas. This occurs frequently in systems where there is an abundance of address space and a need to minimize decoding logic. A **memory map** of this system (see Figure 12-39)



clearly shows the addresses that each device is assigned to as well as the memory space that is available for expansion.

Combining DRAM Chips

DRAM ICs often come with word sizes of one or four bits, so it is necessary to combine several of them to form larger word size modules. Figure 12-40 shows how to combine eight TMS44100 DRAM chips to form a $4M \times 8$ module. Each chip has a $4M \times 1$ capacity.

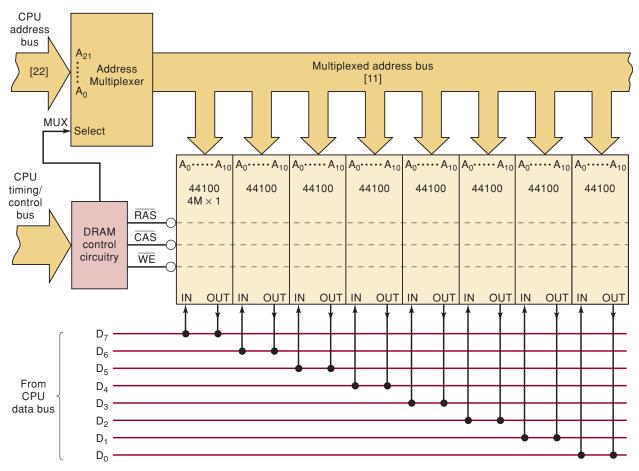


FIGURE 12-40 Eight $4M \times 1$ DRAM chips combined to form a $4M \times 8$ memory module.

There are several important points to note. First, because $4M = 2^{22}$, the TMS44100 chip has *eleven* address inputs; remember, DRAMs use multiplexed address inputs. The address multiplexer takes the 22-line CPU address bus and changes it to an 11-line address bus for the DRAM chips. Second, the \overline{RAS} , \overline{CAS} , and \overline{WE} inputs of all eight chips are connected together so that all chips are activated simultaneously for each memory operation. Finally, recall that the TMS44100 has on-chip refresh control circuitry, so there is no need for an external refresh counter.

REVIEW QUESTIONS

- 1. The MCM6209C is a 64K \times 4 static-RAM chip. How many of these chips are needed to form a 1M \times 4 module?
- 2. How many are needed for a $64K \times 16$ module?
- 3. *True or false:* When memory chips are combined to form a module with a larger word size or capacity, the *CS* inputs of each chip are always connected together.
- 4. *True or false:* When memory chips are combined for a larger capacity, each chip is connected to the same data bus lines.

12-19 SPECIAL MEMORY FUNCTIONS

We have seen that RAM and ROM devices are used as a computer's highspeed internal memory that communicates directly with the CPU (e.g., microprocessor). In this section, we briefly describe some of the special functions that semiconductor memory devices perform in computers and other digital equipment and systems. The discussion is not intended to provide details of how these functions are implemented but to introduce the basic ideas.

Power-Down Storage

In many applications, the volatility of semiconductor RAM can mean the loss of critical data when system power is shut down—either purposely or as the result of an unplanned power interruption. Just two of many examples of this are:

- 1. Critical operating parameters for graphics terminals, intelligent terminals, and printers. These changeable parameters determine the operating modes and attributes that will be in effect upon power-up.
- 2. Industrial process control systems that must never "lose their place" in the middle of a task when power unexpectedly fails.

There are several approaches to providing storage of critical data in power-down situations. In one method, all critical data during normal system operation are stored in RAM that can be powered from backup batteries whenever power is interrupted. Some CMOS RAM chips have very low standby power requirements (as low as 0.5 mW) and are particularly well suited for this task. Some CMOS SRAMs actually include a small lithium battery right on the chip. Of course, even with their low power consumption, these CMOS RAMs will eventually drain the batteries if power is out for prolonged periods, and data will be lost.

Another approach stores all critical system data in nonvolatile flash memory. This approach has the advantage of not requiring backup battery power, and so it presents no risk of data loss even for long power outages. Flash memory, however, cannot have its data changed as easily as static RAM. Recall that with a flash chip, we cannot erase and write to one or two bytes; it must be erased a sector at a time. This requires the CPU to have to rewrite a large block of data even when only a few bytes need to be changed.

In a third approach, the CPU stores all data in high-speed, volatile RAM during normal system operation. On power-down, the CPU executes a short power-down program (from ROM) that transfers critical data from the system

RAM into either battery-backup CMOS RAM or nonvolatile flash memory. This requires a special circuit that senses the onset of a power interruption and sends a signal to the CPU to tell it to begin executing the power-down sequence.

In any case, when power is turned back on, the CPU executes a power-up program (from ROM) that transfers the critical data from the backup storage memory to the system RAM so that the system can resume operation where it left off when power was interrupted.

Cache Memory

Computers and other digital systems may have thousands or millions of bytes of internal memory (RAM and ROM) that store programs and data that the CPU needs during normal operation. Normally, this would require that all of the internal memory have an operating speed comparable to that of the CPU in order to achieve maximum system operation. In many systems, it is not economical to use high-speed memory devices for all of the internal memory. Instead, system designers use a block of high-speed **cache** memory. This cache memory block is the only block that communicates directly with the CPU at high speed; program instructions and data are transferred from the slower, cheaper internal memory to the cache memory when they are needed by the CPU. The success of cache memory depends on many complex factors, and some systems will not benefit from using cache memory.

Modern PC CPUs have a small (8–64 Kbytes) internal memory cache. This is referred to as a level 1 or L1 cache. The chip set of most computer systems also controls an external bank of static RAM (SRAM) that implements a level 2 or L2 cache (64 Kbytes to 2 Mbytes). The cache memory is filled with a sequence of instruction words from the system memory. The CPU (many operating at over 2 GHZ clock rates) can access the cache contents at very high speed. However, when the CPU needs a piece of information that is not currently in either the L1 or the L2 cache (i.e., a cache miss), it must go out to the slow system DRAM to get it. This transfer must occur at the much slower *bus clock rate*, which may be from 66 MHz to 800 MHz depending on your system. In addition to the slower clock rate, the DRAM access time (latency) is much greater.

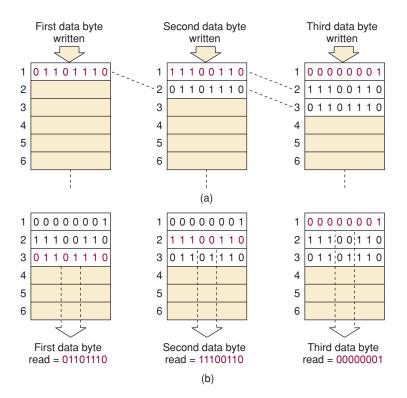
The specification of 7-2-2-2 or 5-1-1-1 for a memory system refers to the number of *bus* clock cycles necessary to transfer a burst of four 64-bit words from DRAM to the L2 cache. The first access takes the most time due to latency associated with RAS/CAS cycles. Subsequent data are clocked out in a burst that takes much less time. For example, the 7-2-2-2 system would require 7 clocks to obtain the first 64-bit word, and each of the next three 64-bit words would require 2 clock cycles each. A total of 13 clock cycles are necessary to get the four words out of memory.

First-In, First-Out Memory (FIFO)

In **FIFO** memory systems, data that are written into the RAM storage area are read out in the same order that they were written in. In other words, the first word written into the memory block is the first word that is read out of the memory block: hence the name FIFO. This idea is illustrated in Figure 12-41.

Figure 12-41(a) shows the sequence of writing three data bytes into the memory block. Note that as each new byte is written into location 1, the other bytes move to the next location. Figure 12-41(b) shows the sequence of reading the data out of the FIFO block. The first byte read is the same as the first byte that was written, and so on. The FIFO operation is controlled by special *address pointer registers* that keep track of where data are to be written and the location from which they are to be read.

FIGURE 12-41 In FIFO, data values are read out of memory (b) in the same order that they were written into memory (a).



A FIFO is useful as a **data-rate buffer** between systems that transfer data at widely different rates. One example is the transfer of data from a computer to a printer. The computer sends character data to the printer at a very high rate, say, one byte every $10 \ \mu$ s. These data fill up a FIFO memory in the printer. The printer then reads out the data from the FIFO at a much slower rate, say, one byte every 5 ms, and prints out the corresponding characters in the same order as sent by the computer.

A FIFO can also be used as a data-rate buffer between a slow device, such as a keyboard, and a high-speed computer. Here, the FIFO accepts keyboard data at the slow and asynchronous rate of human fingers and stores them. The computer can then read all of the recently stored keystrokes very quickly at a convenient point in its program. In this way, the computer can perform other tasks while the FIFO is slowly being filled with data.

Circular Buffers

Data rate buffers (FIFOs) are often referred to as **linear buffers.** As soon as all the locations in the buffer are full, no more entries are made until the buffer is emptied. This way, none of the "old" information is lost. A similar memory system is called a **circular buffer.** These memory systems are used to store the last *n* values entered, where *n* is the number of memory locations in the buffer. Each time a new value is written to a circular buffer, it overwrites (replaces) the oldest value. Circular buffers are addressed by a MOD-*n* address counter. Consequently, when the highest address is reached, the address. As you recall from Chapter 11, digital filtering and other DSP operations perform calculations using a group of recent samples. Special hardware included in a DSP allows easy implementation of circular buffers in memory.

	REV	IEW	QU	ESTI	ONS
--	-----	-----	----	------	-----

- 1. What are the various ways to handle the possible loss of critical data when power is interrupted?
- 2. What is the principal reason for using a cache memory?
- 3. What does *FIFO* mean?
- 4. What is a data-rate buffer?
- 5. How does a circular buffer differ from a linear buffer?

12-20 TROUBLESHOOTING RAM SYSTEMS

All computers use RAM. Many general-purpose computers and most specialpurpose computers (such as microprocessor-based controllers and processcontrol computers) also use some form of ROM. Each RAM and ROM IC that is part of a computer's internal memory typically contains thousands of memory cells. A single faulty memory cell can cause a complete system failure (commonly referred to as a "system crash") or, at the least, unreliable system operation. The testing and troubleshooting of memory systems involves the use of techniques that are not often used on other parts of the digital system. Because memory consists of thousands of identical circuits acting as storage locations, any tests of its operation must involve checking to see exactly which locations are working and which are not. Then, by looking at the pattern of good and bad locations along with the organization of the memory circuitry, one can determine the possible causes of the memory malfunction. The problem typically can be traced to a bad memory IC; a bad decoder IC, logic gate, or signal buffer; or a problem in the circuit connections (i.e., shorts or open connections).

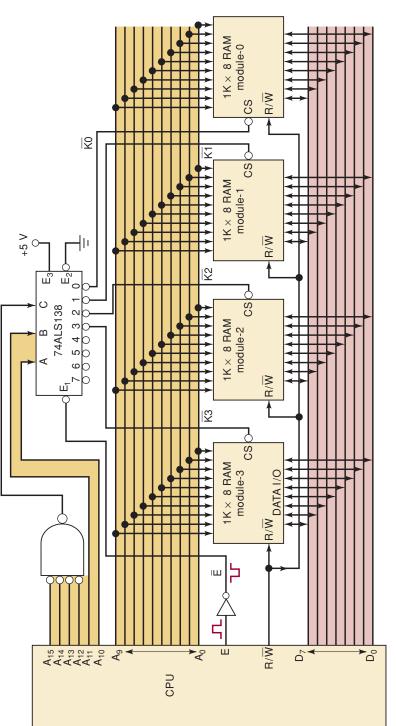
Because RAM must be written to and read from, testing RAM is generally more complex than testing ROM. In this section, we will look at some common procedures for testing the RAM portion of memory and interpreting the test results. We will examine ROM testing in the next section.

Know the Operation

The RAM memory system shown in Figure 12-42 will be used in our examples. As we emphasized in earlier discussions, successful troubleshooting of a relatively complex circuit or system begins with a thorough knowledge of its operation. Before we can discuss the testing of this RAM system, we should first analyze it carefully so that we fully understand its operation.

The total capacity is $4K \times 8$ and is made up of four $1K \times 8$ RAM modules. A module may be just a single IC, or it may consist of several ICs (e.g., two $1K \times 4$ chips). Each module is connected to the CPU through the address and data buses and through the R/W control line. The modules have common I/O data lines. During a read operation, these lines become data output lines through which the selected module places its data on the bus for the CPU to read. During a write operation, these lines act as input lines for the memory to accept CPU-generated data from the data bus for writing into the selected location.

The 74ALS138 decoder and the four-input OR gate combine to decode the six high-order address lines to generate the chip select signals $\overline{K0}$, $\overline{K1}$, $\overline{K2}$, and $\overline{K3}$. These signals enable a specific RAM module for a read or a write operation. The INVERTER is used to invert the CPU-generated Enable signal (*E*)





so that the decoder is enabled only while *E* is HIGH. The *E* pulse occurs only after allowing enough time for the address lines to stabilize following the application of a new address on the address bus. *E* will be LOW while the address and R/\overline{W} lines are changing; this prevents the occurrence of decoder output glitches that could erroneously activate a memory chip and possibly cause invalid data to be stored.

Each RAM module has its address inputs connected to the CPU address bus lines A_0 through A_9 . The high-order address lines A_{10} through A_{15} select one of the RAM modules. The selected module decodes address lines A_0 through A_9 to find the word location that is being addressed. The following examples will show how to determine the addresses that correspond to each module.

EXAMPLE 12-16 Assume that the CPU is performing a read operation from address 06A3 (hex). Which RAM module, if any, is being read from?

Solution

First write out the address in binary.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	1	0	1	0	1	0	0	0	1	1

You should be able to verify that levels A_{15} to A_{10} will activate decoder output $\overline{K1}$ to select RAM module-1. This module internally decodes the address lines A_9 to A_0 to select the location whose data are to be placed on the data bus.

EXAMPLE 12-17

Which RAM module will have data written into it when the CPU executes a write operation to address 1C65?

Solution

Writing out the address in binary, we can see that $A_{12} = 1$. This produces a HIGH out of the OR gate and at the *C* input of the decoder. With $A_{11} = A_{10} = 1$, the decoder inputs are 111, which activates output 7. Outputs $\overline{K0}$ to $\overline{K3}$ will be inactive, and so none of the RAM modules will be enabled. In other words, the data placed on the data bus by the CPU will not be accepted by any of the RAMs.

EXAMPLE 12-18

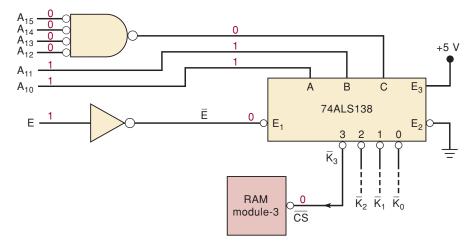
Determine the range of addresses for each module in Figure 12-42.

Solution

Each module stores 1024 eight-bit words. To determine the addresses of the words stored in any module, we start by determining the address bus conditions that activate that module's chip select input. For example, module-3 will be selected when decoder input $\overline{K3}$ is LOW (Figure 12-43). $\overline{K3}$ will be LOW for *CBA* = 011. Working back to the CPU address lines A_{15} to A_{10} , we see that module-3 will be enabled when the following address is placed on the address bus:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	1	1	x	x	x	x	x	x	x	x	x	x

FIGURE 12-43 Example 12-18, showing address bus conditions needed to select RAM module-3.



The x's under A_9 through A_0 indicate don't care because these address lines are not used by the decoder to select module-3. A_9 to A_0 can be any combination ranging from 000000000 to 1111111111, depending on which word in module-3 is being accessed. Thus, the complete range of addresses for module-3 is determined by using all 0s, and then all 1s for the x's.

A ₁₅	<i>A</i> ₁₄	<i>A</i> ₁₃	<i>A</i> ₁₂	<i>A</i> ₁₁	<i>A</i> ₁₀	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	\rightarrow	0C00 ₁₆
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	\rightarrow	$0 \mathrm{FFF}_{16}$

Finally, this gives us 0C00 to 0FFF as the range of hex addresses stored in module-3. When the CPU places any address in this range onto the address bus, only module-3 will be enabled for either a read or a write, depending on the state of R/\overline{W} .

A similar analysis can be used to determine the address ranges for each of the other RAM modules. The results are as follows:

- Module-0: 0000–03FF
- Module-1: 0400–07FF
- Module-2: 0800–0BFF
- Module-3: 0C00–0FFF

Note that the four modules combine for a total address range of 0000 to 0FFF.

Testing the Decoding Logic

In some situations, the decoding logic portion of the RAM circuit (Figure 12-43) can be tested using the various techniques that we have applied to combinatorial circuits. It can be tested by applying signals to the six most significant address lines and E and by monitoring the decoder outputs. To do this, it must be possible to disconnect the CPU easily from these signal lines. If the CPU is a microprocessor chip in a socket, it can simply be removed from its socket.

Once the CPU is disconnected, you can supply the A_{10} - A_{15} and E signals from an external test circuit to perform a static test, using manually operated switches for each signal, or a dynamic test, using some type of counter to cycle through the various address codes. With these test signals applied, the decoder output lines can be checked for the proper response. Standard signal-tracing techniques can be used to isolate any faults in the decoding logic. If you do not have access to the system address lines or do not have a convenient way of generating the static logic signals, it is often possible to force the system to generate a sequence of addresses. Most computer systems used for development have a program stored in a ROM that allows the user to display and change the contents of any memory location. Whenever the computer accesses a memory location, the proper address must be placed on the bus, which should cause the decoder output to go low, even if it is for a short time. Enter the following command to the computer:

Display from 0400 to 07FF

Then place the logic probe on the $\overline{K1}$ output. The logic probe should show pulses during the time the data values are being displayed.

EXAMPLE 12-19

A dynamic test is performed on the decoding logic of Figure 12-43 by keeping E = 1 and connecting the outputs of a six-bit counter to the address inputs A_{10} through A_{15} . The decoder outputs are monitored as the counter repetitively cycles through all six-bit codes. A logic probe check on the decoder outputs shows pulses at $\overline{K1}$ and $\overline{K3}$, but shows $\overline{K0}$ and $\overline{K2}$ remaining HIGH. What are the most probable faults?

Solution

It is possible, but highly unlikely, that $\overline{K0}$ and $\overline{K2}$ could both be stuck HIGH due to either an internal or an external short to V_{CC} . A more likely fault is an open between A_{10} and the A input of the decoder because this would act as a logic HIGH and prevent any even-numbered decoder output from being activated. It is also possible that the decoder's A input is shorted to V_{CC} , but this is also unlikely because this short would have probably affected the operation of the counter that is supplying the address inputs.

Testing the Complete RAM System

Testing and troubleshooting the decoding logic will not reveal problems with the memory chips and their connections to the CPU buses. The most common methods for testing the operation of the *complete* RAM system involve writing known patterns of 1s and 0s to each memory location and reading them back to verify that the location has stored the pattern properly. While many different patterns can be used, one of the most widely used is the "checkerboard pattern." In this pattern, 1s and 0s are alternated as in 01010101. Once all locations have been tested using this pattern, the pattern is reversed (i.e., 10101010), and each location is tested again. Note that this sequence of tests will check each cell for the ability to store and read both a 1 and a 0. Because it alternates 1s and 0s, the checkerboard pattern will also detect any interactions or shorts between adjacent cells. Many other patterns can be used to detect various failure modes within RAM chips.

No memory test can catch all possible RAM faults with 100 percent accuracy, even though it may show that each cell can store and read a 0 or a 1. Some faulty RAMs can be pattern-sensitive. For instance, a RAM may be able to store and read 01010101 and 10101010, but it may fail when 11100011 is stored. Even for a small RAM system, it would take a prohibitively long time to try storing and reading every possible pattern in each location. For this reason, if a RAM system passes the checkerboard test, you can conclude that it is *probably* good; if it fails the test, then it *definitely* contains a fault.

Manually testing thousands of RAM locations by storing and reading checkerboard patterns would take hundreds of hours and is obviously not feasible. RAM pattern testing is usually done automatically either by having the CPU run a memory test program or by connecting a special test instrument to the RAM system buses in place of the CPU. In fact, in many computers and microprocessor-based equipment, the CPU will automatically run a memory test program every time it is powered up; this is called a **power-up self-test**. The self-test routine (we will call it SELF-TEST) is stored in ROM, and it is executed whenever the system is turned on or when the operator requests it from the keyboard. As the CPU executes SELF-TEST, it will write test patterns to and read test patterns from each RAM location and will display some type of message to the user. It may be something as simple as an LED to indicate faulty memory, or it may be a descriptive message printed on the screen or printer. Typical messages might be:

RAM module-3 test OK ALL RAM working properly Location 027F faulty in bit positions 6 and 7

With messages like these and a knowledge of the RAM system operation, the troubleshooter can determine what additional action is needed to isolate the fault.

REVIEW QUESTIONS

- 1. What is *E*'s function in the RAM circuit of Figure 12-42?
- 2. What is the checkerboard test? Why is it used?
- 3. What is a power-up self-test?

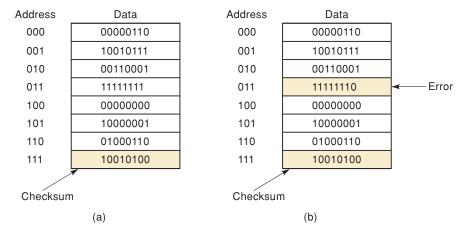
12-21 TESTING ROM

The ROM circuitry in a computer is very similar to the RAM circuitry (compare Figures 12-37 and 12-42). The ROM decoding logic can be tested in the same manner described in the preceding section for the RAM system. The ROM chips, however, must be tested differently from RAM chips because we cannot write patterns into ROM and read them back as we can for RAM. Several methods are used to check the contents of a ROM IC.

In one approach, the ROM is placed in a socket in a special test instrument that is typically microprocessor-controlled. The special test instrument can be programmed to read every location in the test ROM and print out a listing of the contents of each location. The listing can then be compared with what the ROM is supposed to contain. Except for low-capacity ROM chips, this process can be very time-consuming.

In a more efficient approach, the test instrument has the correct data stored in its own *reference* ROM chip. The test instrument is then programmed to read the contents of each location of the test ROM and compare it with the contents of the reference ROM. This approach, of course, requires the availability of a preprogrammed reference ROM.

A third approach uses a **checksum**, a special code placed in the last one or two locations of the ROM chip when it is programmed. This code is derived by adding up the data words to be stored in all of the ROM locations (excluding FIGURE 12-44 Checksum method for an 8×8 ROM: (a) ROM with correct data; (b) ROM with error in its data.



those containing the checksum). As the test instrument reads the data from each test ROM location, it will add them up and develop its own checksum. It then compares its calculated checksum with that stored in the last ROM locations, and they should agree. If so, there is a high probability that the ROM is good (there is a very small chance that a combination of errors in the test ROM data could still produce the same checksum value). If they do not agree, then there is a definite problem in the test ROM.

The checksum idea is illustrated in Figure 12-44(a) for a very small ROM. The data word stored in the last address is the eight-bit sum of the other seven data words (ignoring carries from the MSB). When this ROM is programmed, the checksum is placed in the last location. Figure 12-44(b) shows the data that might actually be read from a faulty ROM that was originally programmed with the data in Figure 12-44(a). Note the error in the word at address 011. As the test instrument reads the data from each location of the faulty ROM, it calculates its own checksum from those data. Because of the error, the calculated checksum will be 10010011. When the test instrument compares this with the checksum value stored at ROM location 111, it sees that they disagree, and a ROM error is indicated. Of course, the exact location of the error cannot be determined.

The checksum method can also be used by a computer or microprocessorbased equipment during an automatic power-up self-test to check out the contents of the system ROMs. Again, as in the self-test used for RAM, the CPU would execute a program on power-up that would do a checksum test on each ROM chip and would print out some type of status message. The selftest program itself will be located in a ROM, and so any error in that ROM could prevent the successful execution of the checksum tests.

REVIEW QUESTIONS

1. What is a checksum? What is its purpose?

SUMMARY

1. All memory devices store binary logic levels (1s and 0s) in an array structure. The size of each binary word (number of bits) that is stored varies depending on the memory device. These binary values are referred to as *data*.

- 2. The place (location) in the memory device where any data value is stored is identified by another binary number referred to as an *address*. Each memory location has a unique address.
- 3. All memory devices operate in the same general way. To write data in memory, the address to be accessed is placed on the address input, the data value to be stored is applied to the data inputs, and the control signals are manipulated to store the data. To read data from memory, the address is applied, the control signals are manipulated, and the data value appears on the output pins.
- 4. Memory devices are often used along with a microprocessor CPU that generates the addresses and control signals and either provides the data to be stored or uses the data from the memory. Reading and writing are *always* done from the CPU's perspective. Writing puts data into the memory, and reading gets data out of the memory.
- 5. Most read-only memories (ROMs) have data entered one time, and from then on their contents do not change. This storage process is called *programming*. They do not lose their data when power is removed from the device. MROMs are programmed during the manufacturing process. PROMs are programmed one time by the user. EPROMs are just like PROMs but can be erased using UV light. EEPROMs and flash memory devices are electrically erasable and can have their contents altered after programming. CD ROMs are used for mass storage of information that does not need to change.
- 6. Random access memory (RAM) is a generic term given to devices that can have data easily stored and retrieved. Data are retained in a RAM device only as long as power is applied.
- 7. Static RAM (SRAM) uses storage elements that are basically latch circuits. Once the data are stored, they will remain unchanged as long as power is applied to the chip. Static RAM is easier to use but more expensive per bit and consumes more power than dynamic RAM.
- 8. Dynamic RAM (DRAM) uses capacitors to store data by charging or discharging them. The simplicity of the storage cell allows DRAMs to store a great deal of data. Because the charge on the capacitors must be refreshed regularly, DRAMs are more complicated to use than SRAMs. Extra circuitry is often added to DRAM systems to control the reading, writing, and refreshing cycles. On many new devices, these features are being integrated into the DRAM chip itself. The goal of DRAM technology is to put more bits on a smaller piece of silicon so that it consumes less power and responds faster.
- 9. Memory systems require a wide variety of different configurations. Memory chips can be combined to implement any desired configuration, whether your system needs more bits per location or more total word capacity. All of the various types of ROM and RAM can be combined within the same memory system.

IMPORTANT TERMS

- main memory auxiliary memory memory cell memory word byte
- capacity density address read operation write operation
- access time volatile memory random-access memory (RAM)

sequential-access memory (SAM) read/write memory (RWM) read-only memory (ROM) static RAM (SRAM) dynamic RAM address bus data bus control bus programming chip select power-down fusible link electrically erasable PROM (EEPROM) flash memory bootstrap program refresh JEDEC address multiplexing strobing row address strobe (RAS) column address strobe (CAS) latency RAS-only refresh refresh counter DRAM controller memory foldback memory map cache FIFO data-rate buffer linear buffer circular buffer power-up self-test checksum

PROBLEMS

SECTIONS 12-1 TO 12-3

- B 12-1.*A certain memory has a capacity of 16K × 32. How many words does it store? What is the number of bits per word? How many memory cells does it contain?
- **B** 12-2. How many different addresses are required by the memory of Problem 12-1?
- **B** 12-3.* What is the capacity of a memory that has 16 address inputs, four data inputs, and four data outputs?
- B 12-4. A certain memory stores 8K 16-bit words. How many data input and data output lines does it have? How many address lines does it have? What is its capacity in bytes?

DRILL QUESTIONS

- 12-5. Define each of the following terms.
 - (a) RAM

В

В

- (b) RWM
- (c) ROM
- (d) Internal memory
- (e) Auxiliary memory
- (f) Capacity
- (g) Volatile
- (h) Density
- (i) Read
- (j) Write

12-6. (a) What are the three buses in a computer memory system?

(b) Which bus is used by the CPU to select the memory location?

- (c) Which bus is used to carry data from memory to the CPU during a read operation?
- (d) What is the source of data on the data bus during a write operation?

^{*}Answers to problems marked with an asterisk can be found in the back of the text.

SECTIONS 12-4 AND 12-5

B

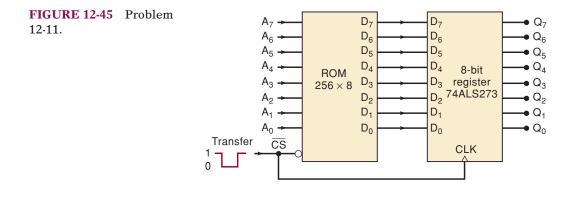
- 12-7.*Refer to Figure 12-6. Determine the data outputs for each of the following input conditions.
- (a) [A] = 1011; CS = 1
 - (b) [*A*] = 0111; *CS* = 0
- **B** 12-8. Refer to Figure 12-7.
 - (a) What register is enabled by input address 1011?
 - (b) What input address code selects register 4?
- **B** 12-9.*A certain ROM has a capacity of $16K \times 4$ and an internal structure like that shown in Figure 12-7.
 - (a) How many registers are in the array?
 - (b) How many bits are there per register?
 - (c) What size decoders does it require?

DRILL QUESTION

- **B** 12-10. (a) *True or false:* ROMs cannot be erased.
 - (b) What is meant by *programming* or *burning* a ROM?
 - (c) Define a ROM's access time.
 - (d) How many data inputs, data outputs, and address inputs are needed for a 1024 $\,\times\,$ 4 ROM?
 - (e) What is the function of the decoders on a ROM chip?

SECTION 12-6

C, **D** 12-11.* Figure 12-45 shows how data from a ROM can be transferred to an external register. The ROM has the following timing parameters: $t_{ACC} = 250$ ns and $t_{OE} = 120$ ns. Assume that the new address inputs have been applied to the ROM 500 ns before the occurrence of the TRANS-FER pulse. Determine the minimum duration of the TRANSFER pulse for reliable transfer of data.



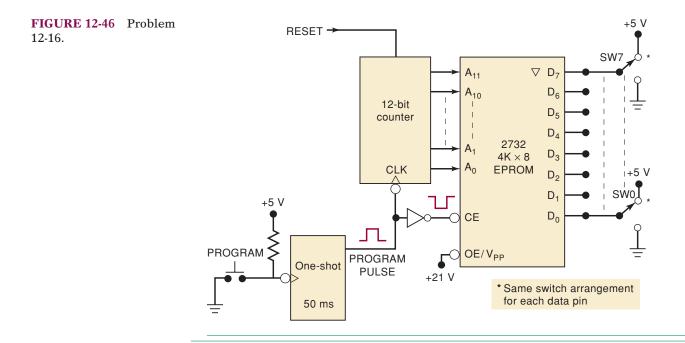
C, D 12-12. Repeat Problem 12-11 if the address inputs are changed 70 ns prior to the TRANSFER pulse.

SECTIONS 12-7 AND 12-8

B 12-13. DRILL QUESTION

For each item below, indicate the type of memory being described: MROM, PROM, EPROM, EEPROM, flash. Some items will correspond to more than one memory type.

- (a) Can be programmed by the user but cannot be erased.
- (b) Is programmed by the manufacturer.
- (c) Is volatile.
- (d) Can be erased and reprogrammed over and over.
- (e) Individual words can be erased and rewritten.
- (f) Is erased with UV light.
- (g) Is erased electrically.
- (h) Uses fusible links.
- (i) Can be erased in bulk or in sectors of 512 bytes.
- (j) Does not have to be removed from the system to be erased and reprogrammed.
- (k) Requires a special supply voltage for reprogramming.
- (l) Erase time is about 15 to 20 min.
- **B** 12-14. Which transistors in Figure 12-9 will be conducting when $A_1 = A_0 = 1$ and $\overline{EN} = 0$?
 - 12-15.* Change the MROM connections in Figure 12-9 so that the MROM stores the function y = 3x + 5.
- D 12-16. Figure 12-46 shows a simple circuit for manually programming a 2732 EPROM. Each EPROM data pin is connected to a switch that can be set at a 1 or a 0 level. The address inputs are driven by a 12-bit counter. The 50-ms programming pulse comes from a one-shot each time the PROGRAM push button is actuated.
 - (a) Explain how this circuit can be used to program the EPROM memory locations sequentially with the desired data.



- (b) Show how 74293s and a 74121 can be used to implement this circuit.
- (c) Should switch bounce have any effect on the circuit operation?
- N 12-17.* Figure 12-47 shows a 28F256A flash memory chip connected to a CPU over a data bus and an address bus. The CPU can write to or read from the flash memory array by sending the desired memory address and generating the appropriate control signals to the chip [Figure 12-15(b)]. The CPU can also write to the chip's command register (Figure 12-16) by generating the appropriate control signals and sending the desired command code over the data bus. For this latter operation, the CPU does not have to send a specific memory address to the chip; in other words, the address lines are don't-cares.
 - (a) Consider the following sequence of CPU operations. Determine what will have happened to the flash memory at the completion of the sequence. Assume that the command register is holding 00_{16} .
 - 1. The CPU places 20_{16} on the data bus and pulses \overline{CE} and \overline{WE} LOW while holding \overline{OE} HIGH. The address bus is at 0000_{16} .
 - 2. The CPU repeats step 1.
 - (b) After the sequence above has been executed, the CPU executes the following sequence. Determine what this does to the flash memory chip.
 - 1. The CPU places 40_{16} on the data bus and pulses \overline{CE} and \overline{WE} LOW while holding \overline{OE} HIGH. The address bus is at 0000_{16} .
 - 2. The CPU places $3C_{16}$ on the data bus and 2300_{16} onto the address bus, and it pulses \overline{CE} and \overline{WE} LOW while holding \overline{OE} HIGH.

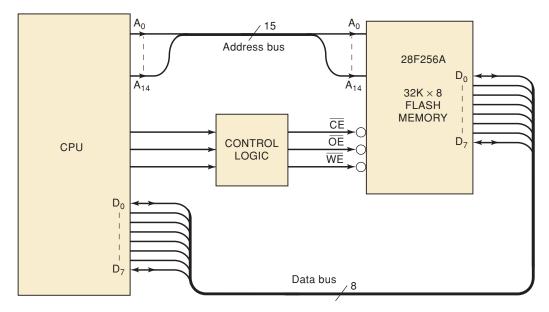
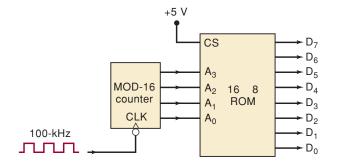


FIGURE 12-47 Problem 12-17.

N 12-18. Another ROM application is the generation of timing and control signals. Figure 12-48 shows a 16×8 ROM with its address inputs driven by a MOD-16 counter so that the ROM addresses are incremented with





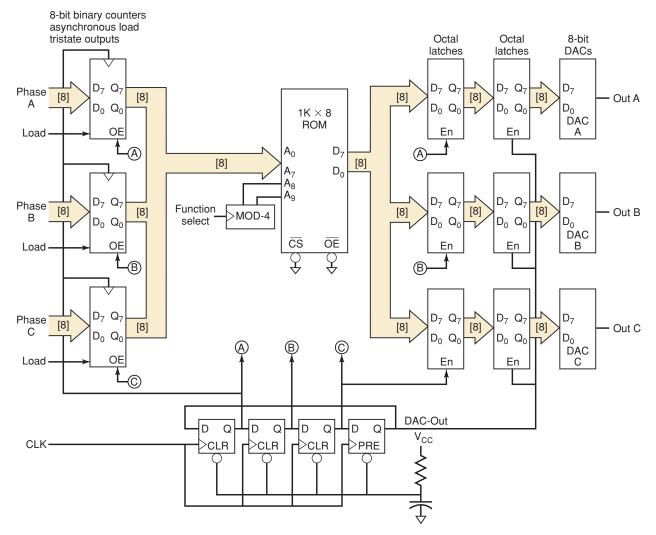
each input pulse. Assume that the ROM is programmed as in Figure 12-6, and sketch the waveforms at each ROM output as the pulses are applied. Ignore ROM delay times. Assume that the counter starts at 0000.

D 12-19.* Change the program stored in the ROM of Problem 12-18 to generate the D_7 waveform of Figure 12-49.

FIGURE 12-49 Problem 12-19.

D ₇						_
→						
10 <i>µ</i> s						

- **D** 12-20.* Refer to the function generator of Figure 12-17.
 - (a) What clock frequency will result in a 100-Hz sine wave at the output?
 - (b) What method could be used to vary the peak-to-peak amplitude of the sine wave?
- **C** 12-21. For the ML2035 of Figure 12-18, assume that a value of 038E (hex) in the latch will produce the desired frequency. Draw the timing diagram for the *LATI*, *SID*, and *SCK* inputs, and assume that the LSB is shifted in first.
- **N, C** 12-22* The system shown in Figure 12-50 is a waveform (function) generator. It uses four 256-point look-up tables in a 1-Kbyte ROM to store one cycle each of a sine wave (address 000–0FF), a positive slope ramp (address 100–1FF), a negative slope ramp (200–2FF), and a triangle wave (300–3FF). The phase relationship among the three output channels is controlled by the values initially loaded into the three counters. The critical timing parameters are $t_{pd(ck-Q \text{ and } OE-Q \text{ max})}$, counters = 10 ns, latches = 5 ns, and t_{ACC} ROM = 20 ns. Study the diagram until you understand how it operates and then answer the following:
 - (a) If counter A is initially loaded with 0, what values must be loaded into counters B and C so that A lags B by 90° and A lags C by 180°?
 - (b) If counter A is initially loaded with 0, what values must be loaded into counters B and C to generate a three-phase sine wave with 120° shift between each output?
 - (c) What must the frequency of pulses on DAC_OUT be in order to generate a 60-Hz sine wave output?
 - (d) What is the maximum frequency of the CLK input?
 - (e) What is the maximum frequency of the output waveforms?
 - (f) What is the purpose of the function select counter?





- 12-23. (a) Draw the logic symbol for an MCM101514, a CMOS static RAM organized as a $256K \times 4$ with separate data in and data out, and an active-LOW chip enable.
 - (b) Draw the logic symbol for an MCM6249, a CMOS static RAM organized as a $1M \times 4$ with common I/O, an active-LOW chip enable, and an active-LOW output enable.

SECTION 12-12

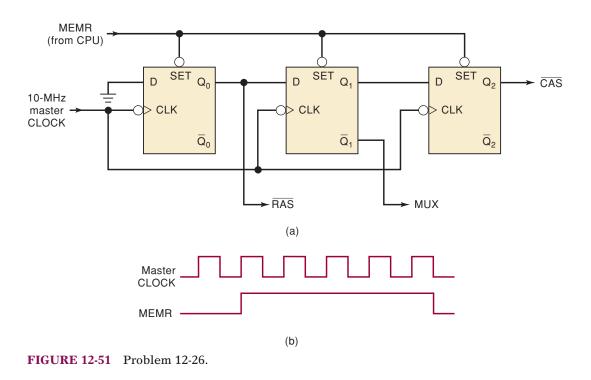
12-24.*A certain static RAM has the following timing parameters (in nanoseconds):

$t_{\rm RC} = 100$	$t_{\rm AS} = 20$
$t_{\rm ACC} = 100$	$t_{\rm AH}$ = not given
$t_{\rm CO} = 70$	$t_{\rm W} = 40$
$t_{\rm OD} = 30$	$t_{\rm DS}=10$
$t_{\rm WC} = 100$	$t_{\rm DH}=20$

- (a) How long after the address lines stabilize will valid data appear at the outputs during a read cycle?
- (b) How long will output data remain valid after \overline{CS} returns HIGH?
- (c) How many read operations can be performed per second?
- (d) How long should R/\overline{W} and \overline{CS} be kept HIGH after the new address stabilizes during a write cycle?
- (e) What is the minimum time that input data must remain valid for a reliable write operation to occur?
- (f) How long must the address inputs remain stable after R/\overline{W} and \overline{CS} return HIGH?
- (g) How many write operations can be performed per second?

SECTIONS 12-13 TO 12-17

- 12-25. Draw the logic symbol for the TMS4256, which is a $256K \times 1 DRAM$. How many pins are saved by using address multiplexing for this DRAM?
- **D** 12-26. Figure 12-51(a) shows a circuit that generates the \overline{RAS} , \overline{CAS} , and MUX signals needed for proper operation of the circuit of Figure 12-28(b). The 10-MHz master clock signal provides the basic timing for the computer. The memory request signal (*MEMR*) is generated by the CPU in synchronism with the master clock, as shown in part (b) of the figure. *MEMR* is normally LOW and is driven HIGH whenever the CPU wants to access memory for a read or a write operation. Determine the waveforms at Q_0 , $\overline{Q_1}$, and Q_2 , and compare them with the desired waveforms of Figure 12-29.



D 12-27. Show how to connect two 74157 multiplexers (Section 9-6) to provide the multiplexing function required in Figure 12-28(b).

- 12-28. Refer to the signals in Figure 12-30. Describe what occurs at each of the labeled time points.
- 12-29. Repeat Problem 12-28 for Figure 12-31.
- **C** 12-30.* The 21256 is a 256K \times 1 DRAM that consists of a 512 \times 512 array of cells. The cells must be refreshed within 4 ms for data to be retained. Figure 12-33(a) shows the signals used to execute a \overline{CAS} -before- \overline{RAS} refresh cycle. Each time a cycle such as this occurs, the on-chip refresh circuitry will refresh a row of the array at the row address specified by a refresh counter. The counter is incremented after each refresh. How often should \overline{CAS} -before- \overline{RAS} cycles be applied in order for all of the data to be retained?
 - 12-31.*Study the functional block diagram of the TMS44100 DRAM in Figure 12-27.
 - (a) What are the actual dimensions of the DRAM cell array?
 - (b) If the cell array were actually square, how many rows would there be?
 - (c) How would this affect the refresh time?

- D 12-32. Show how to combine two 6206 RAM chips (Figure 12-20) to produce a 32K × 16 module.
- D 12-33. Show how to connect two of the 6264 RAM chips symbolized in Figure 12-23 to produce a 16K × 8 RAM module. The circuit should not require any additional logic. Draw a memory map showing the address range of each RAM chip.
- **D** 12-34.*Describe how to modify the circuit of Figure 12-37 so that it has a total capacity of $16K \times 8$. Use the same type of PROM chips.
- **D** 12-35. Modify the decoding circuit of Figure 12-37 to operate from a 16-line address bus (i.e., add A_{13} , A_{14} , and A_{15}). The four PROMs are to maintain the same hex address ranges.
- C 12-36. For the memory system of Figure 12-38, assume that the CPU is storing one byte of data at system address 4000 (hex).
 - (a) Which chip is the byte stored in?
 - (b) Is there any other address in this system that can access this data byte?
 - (c) Answer parts (a) and (b) by assuming that the CPU has stored a byte at address 6007. (*Hint:* Remember that the EEPROM is not completely decoded.)
 - (d) Assume that the program is storing a sequence of data bytes in the EEPROM and that it has just completed the 2048th byte at address 67FF. If the programmer allows it to store one more byte at address 6800, what will be the effect on the first 2048 bytes?
- D 12-37. Draw the complete diagram for a 256K × 8 memory that uses RAM chips with the following specifications: 64K × 4 capacity, common in-put/output line, and two active-LOW chip select inputs. [*Hint:* The circuit can be designed using only two inverters (plus memory chips).]

SECTION 12-20

12-38.*Modify the RAM circuit of Figure 12-42 as follows: change the OR gate to an AND gate and disconnect its output from *C*; connect the

AND output to E_3 ; connect *C* to ground. Determine the address range for each RAM module.

- **C**, **D** 12-39. Show how to expand the system of Figure 12-42 to an 8K × 8 with addresses ranging from 0000 to 1FFF. (*Hint:* This can be done by adding the necessary memory modules and modifying the existing decoding logic.)
 - **T** 12-40.* A dynamic test is performed on the decoding logic of Figure 12-42 by keeping E = 1 and connecting the outputs of a six-bit counter to address inputs A_{10} to A_{15} . The decoder outputs are monitored with an oscilloscope (or a logic analyzer) as the counter is continuously pulsed by a 1-MHz clock. Figure 12-52(a) shows the displayed signals. What are the most probable faults?

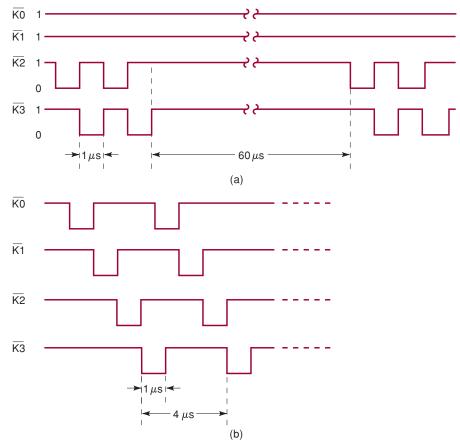


FIGURE 12-52 Problems 12-40 and 12-41.

- C, T 12-41. Repeat Problem 12-40 for the decoder outputs shown in Figure 12-52(b).
- C, D 12-42* Consider the RAM system of Figure 12-42. The checkerboard pattern test will not be able to detect certain types of faults. For instance, assume that there is a break in the connection to the A input to the decoder. If a checkerboard pattern SELF-TEST is performed on this circuit, the displayed messages will state that the memory is OK.
 - (a) Explain why the circuit fault was not detected.
 - (b) How would you modify the SELF-TEST so that faults such as this will be detected?

T 12-43.*Assume that the $1K \times 8$ modules used in Figure 12-42 are formed from two $1K \times 4$ RAM chips. The following messages are printed out when the power-up self-test is performed on this RAM system:

```
module-0 test OK
module-1 test OK
address 0800 faulty at bits 4-7
address 0801 faulty at bits 4-7
address 0802 faulty at bits 4-7
   . . .
             .
                   .
   . . .
             .
                   .
                         .
     .
        .
             .
                   .
address OBFE faulty at bits 4-7
address OBFF faulty at bits 4-7
module-3 test OK
```

Examine these messages and list the possible faults.

T 12-44.* The following messages are printed out when the power-up self-test is performed on the RAM system of Figure 12-42.

Examine these messages and list the possible faults.

T 12-45. What messages would be printed out when a power-up self-test is performed on the RAM system of Figure 12-42 if there is a short between the decoder's $\overline{K2}$ and $\overline{K3}$ outputs?

SECTION 12-21

T 12-46.*Consider the 16×8 ROM in Figure 12-6. Replace the data word stored at address location 1111 with a checksum calculated from the other 15 data words.

ANSWERS TO SECTION REVIEW QUESTIONS

SECTION 12-1

1. See text. 2. 16 bits per word; 8192 words; 131,072 bits or cells 3. In a read operation, a word is taken from a memory location and is transferred to another device. In a write operation, a new word is placed in a memory location and replaces the one previously stored there. 4. True 5. SAM: Access time is not constant but depends on the physical location of the word being accessed. RAM: Access time is the same for any address location. 6. RWM is memory that can be read from or written to with equal ease. ROM is memory that is mainly read from and is written into very infrequently. 7. False; its data must be periodically refreshed.

1. 14, 12, 12 2. Commands the memory to perform either a read operation or a write operation 3. When in its active state, this input enables the memory to perform the read or the write operation selected by the R/\overline{W} input. When in its inactive state, this input disables the memory so that it cannot perform the read or the write function.

SECTION 12-3

1. Address lines, data lines, control lines 2. See text. 3. See text.

SECTION 12-4

True
 Apply desired address inputs; activate control input(s); data appear at data outputs.
 Process of entering data into ROM

SECTION 12-5

1. $A_3A_2A_1A_0 = 1001$ 2. The row-select decoder activates one of the enable inputs of all registers in the selected row. The column-select decoder activates one of the enable inputs of all registers in the selected column. The output buffers pass the data from the internal data bus to the ROM output pins when the *CS* input is activated.

SECTION 12-7

 False; by the manufacturer
 A PROM can be programmed once by the user. It cannot be erased and reprogrammed.
 True
 By exposure to UV light
 True
 Automatically programs data into memory cells one address at a time
 An EEPROM can be electrically erased and reprogrammed without removal from its circuit, and it is byte erasable.
 Low density; high cost
 EEPROM
 One

SECTION 12-8

Electrically erasable and programmable in circuit
 Higher density; lower
 Short erase and programming times
 For the erase and programming operations
 The contents of this register control all internal chip functions.
 To confirm that a memory address has been successfully erased (i.e., data = all 1s)
 To confirm that a memory address has been programmed with the correct data

SECTION 12-9

1. On power-up, the computer executes a small bootstrap program from ROM to initialize the system hardware and to load the operating system from mass storage (disk). 2. Circuit that takes data represented in one type of code and converts it to another type of code 3. Counter, ROM, DAC, low-pass filter 4. They are nonvolatile, fast, reliable, small, and low-power.

SECTION 12-11

1. Desired address applied to address inputs; $R/\overline{W} = 1$; *CS* or *CE* activated 2. To reduce pin count 3. 24, including V_{CC} and ground

SECTION 12-12

1. SRAM cells are flip-flops; DRAM cells use capacitors. 2. CMOS 3. Memory 4. CPU 5. Read- and write-cycle times 6. False; when \overline{WE} is LOW, the I/O pins act as data inputs regardless of the state of \overline{OE} (second entry in mode table). 7. A_{13} can remain connected to pin 26. A_{14} must be removed, and pin 27 must be connected to +5 V.

1. Generally slower speed; need to be refreshed2. Low power; high capacity;lower cost per bit3. DRAM

SECTION 12-14

1. 256 rows × 256 columns 2. It saves pins on the chip. 3. $1M = 1024K = 1024 \times 1024$. Thus, there are 1024 rows by 1024 columns. Because $1024 = 2^{10}$, the chip needs 10 address inputs. 4. \overline{RAS} is used to latch the row address into the DRAM's row address register. \overline{CAS} is used to latch the column address into the column address register. 5. *MUX* multiplexes the full address into the row and column addresses for input to the DRAM.

SECTION 12-15

1. (a) True (b) False (c) False (d) True 2. MUX

SECTION 12-16

1. (a) True (b) False2. It provides row addresses to the DRAM during refreshcycles.3. Address multiplexing and the refresh operation4. (a) False (b) True

SECTION 12-17

1. No 2. Memory locations with same upper address (same row) 3. Only the column address must be latched. 4. Extended data output 5. *Burst* 6. The system clock

SECTION 12-18

1. Sixteen 2. Four 3. False; when expanding memory capacity, each chip is selected by a different decoder output (see Figure 12-43). 4. True

SECTION 12-19

1. Battery backup for CMOS RAM; flash memory 2. Economics 3. Data are read out of memory in the same order they were written in. 4. A FIFO used to transfer data between devices with widely different operating speeds 5. Circular buffers "wrap around" from the highest address to the lowest, and the newest datum always overwrites the oldest.

SECTION 12-20

1. Prevents decoding glitches by disabling the decoder while the address lines are changing 2. A way to test RAM by writing a checkerboard pattern (first 01010101, then 10101010) into each memory location and then reading it. It is used because it will detect any shorts or interactions between adjacent cells. 3. An automatic test of RAM performed by a computer on power-up

SECTION 12-21

1. A code placed in the last one or two ROM locations that represents the sum of the expected ROM data from all other locations. It is used as a means to test for errors in one or more ROM locations.