CHAPTER 13

Memory Devices and Systems

OUTLINE

CHAPTER OBJECTIVES

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

- Describe basic memory concepts of address and data.
- Understand how latches and flip-flops act as simple memory devices and sketch simple memory systems based on these devices.
- Distinguish between random access read/write memory (RAM) and read only memory (ROM).
- Describe the uses of tristate logic in data bussing.
- Sketch the circuits of static and dynamic RAM cells.
- Sketch a block diagram of a static or dynamic RAM chip.
- Describe various types of ROM cells and arrays: mask-programmed, UV erasable, and electrically erasable.
- Use various types of ROM in simple applications, such as digital function generation.
- Describe the basic configuration of flash memory.
- Describe the basic configuration and operation of two types of sequential memory: first-in-first-out (FIFO) and last-in-first-out (LIFO).
- Describe how dynamic RAM is configured into high capacity memory modules.
- Sketch a basic memory system, consisting of several memory devices, an address and a data bus, and address decoding circuitry.
- Represent the location of various memory device addresses on a system memory map.
- Recognize and eliminate conditions leading to bus contention in a memory system.
- Expand memory capacity by parallel bussing and CPLD-based decoding.

In recent years, memory has become one of the most important topics in digital electronics. This is tied closely to the increasing prominence of cheap and readily available microprocessor chips. The simplest memory is a device we are already familiar with: the D flip-flop. This device stores a single bit of information as long as necessary. This simple concept is at the heart of all memory devices.

The other basic concept of memory is the organization of stored data. Bits are stored in locations specified by an "address," a unique number which tells a digital system how to find data that have been previously stored. (By analogy, think of your street address: a unique way to find you and anyone you live with.)

Some memory can be written to and read from in random order; this is called random access read/write memory (RAM). Other memory can be read only: read only memory (ROM). Yet another type of memory, sequential memory, can be read or written only in a specific sequence. There are several variations on all these basic classes.

Memory devices are usually part of a larger system, including a microprocessor, peripheral devices, and a system of tristate busses. If dynamic RAM is used in such a system, it is often in a memory module of some type. The capacity of a single memory chip is usually less than the memory capacity of the microprocessor system in which it is used. In order to use the full system capacity, it is necessary to use a method of memory address decoding to select a particular RAM device for a specified portion of system memory.

13.1 Basic Memory Concepts

KEY TERMS

Memory A device for storing digital data in such a way that they can be recalled for later use in a digital system.

Data Binary digits (0s and 1s) that contain some kind of information. The digital contents of a memory device.

Address A number, represented by the binary states of a group of inputs or outputs, uniquely defining the location of data stored in a memory device.

Write Store data in a memory device.

Read Retrieve data from a memory device.

Byte A group of 8 bits.

Nibble Half a byte; 4 bits.

Address and Data

A **memory** is a digital device or circuit that can store one or more bits of **data.** The simplest memory device, a D-type latch, shown in Figure 13.1, can store 1 bit. A 0 or 1 is stored in the latch and remains there until changed.

octal_latch.gdf
1x8mem.gdf

A simple extension of the single D-type latch is an array of latches, shown in Figure 13.2, that can store 8 bits (1 **byte**) of data. Figure 13.3 shows this octal latch used as a component in a MAX+PLUS II graphic file and configured as an 8-bit memory.

When the *WRITEn* line goes LOW, then HIGH, data at the *DATA_IN* are stored in the eight latches. Data are available at the *DATA_OUT* pins when *READ* is HIGH. Note that although the *READ* and *WRITEn* inputs are separate in this design, their functions would often be implemented as opposite logic levels of the same pin.

1x8mem.scf

Figure 13.4 shows a simulation of the 8-bit memory. The LOW pulses on *WRITEn* write the data, shown as two hexadecimal digits on the *DATA_IN* line, into the latches. To read the values stored in the eight latches, we set *READ* HIGH. In between read states, all *DATA_OUT* lines are in the high-impedance state, indicated by the notation ZZ.

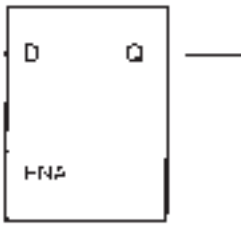


FIGURE 13.1
D-Type Latch

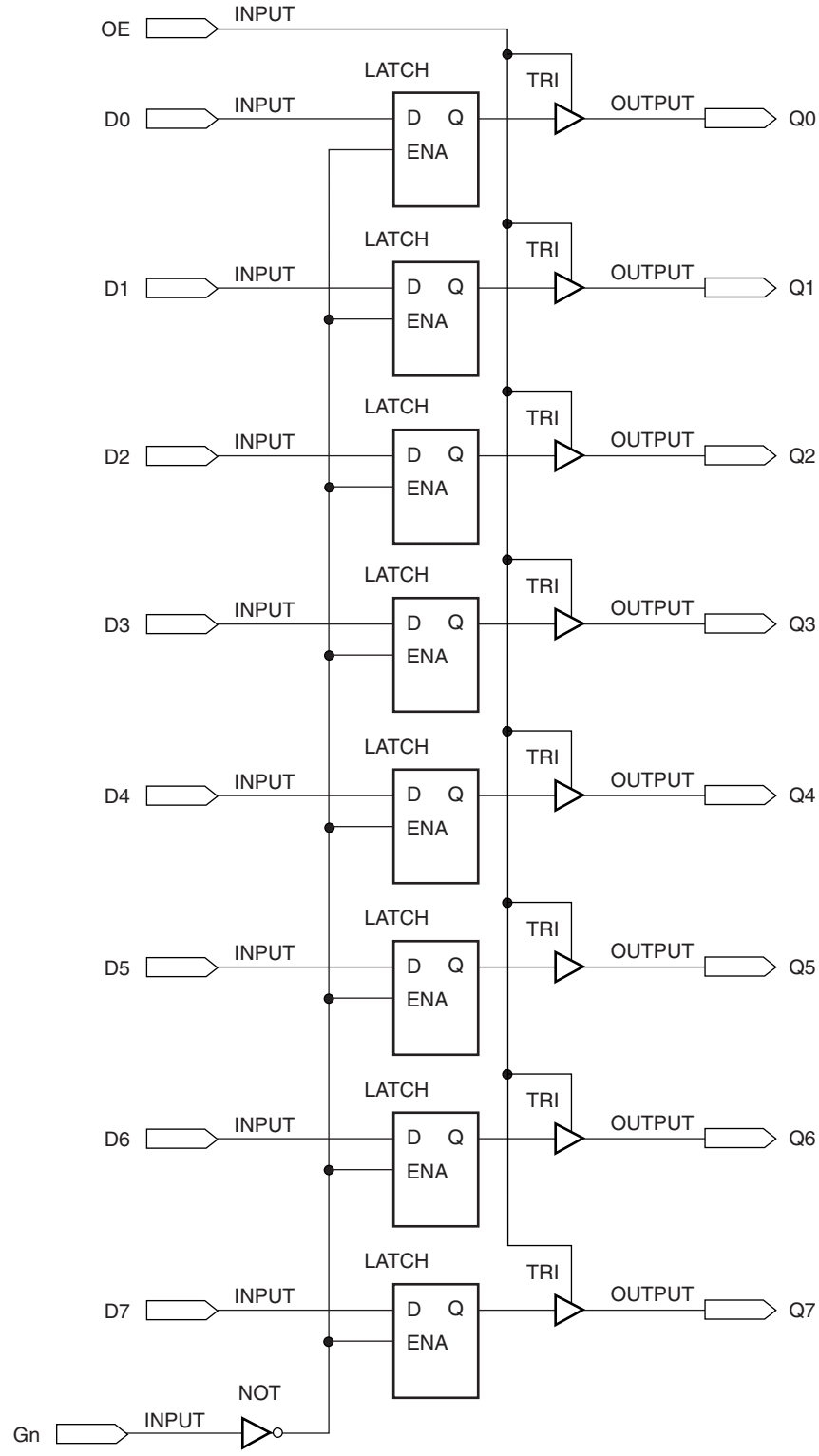
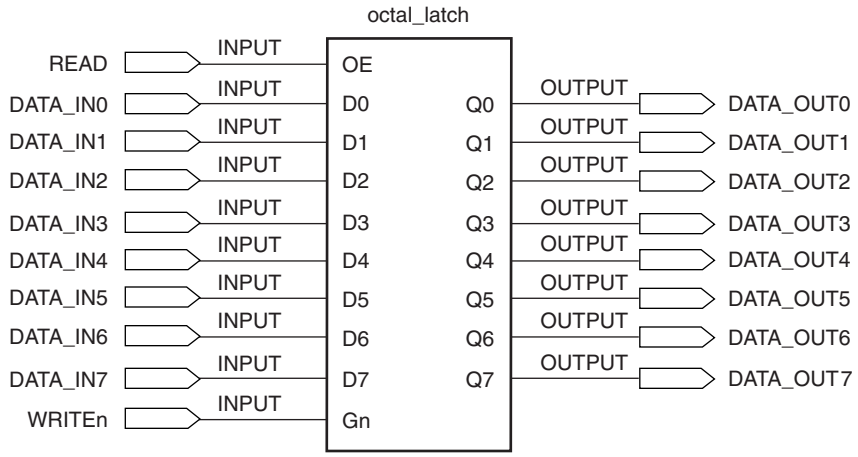


FIGURE13.2
Octal Latch

FIGURE13.3
Octal Latch as 8-bit Memory



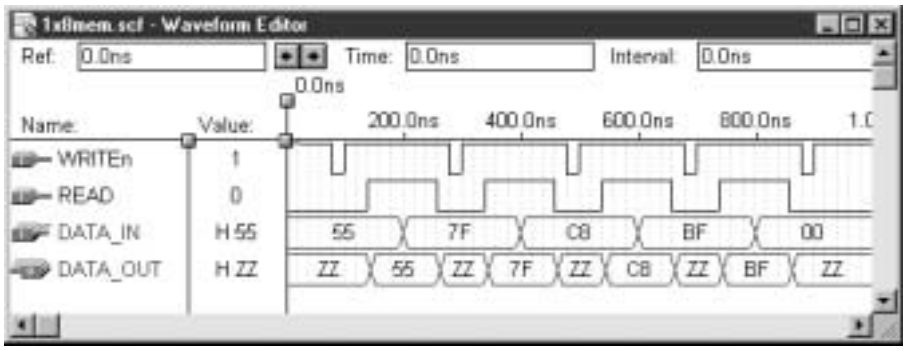


FIGURE 13.4
Simulation of 8-bit Memory

4x8reg.gdf
ltch8lpm.vhd
dcdr2to4.vhd
oct4tol.vhd

Figure 13.5 shows an expanded version of the octal latch memory circuit. Four octal latches are configured to make a 4 × 8-bit memory that can store and recall four separate 8-bit words. The octal latches are based on 8-bit latches instantiated in VHDL from the Altera Library of Parameterized Modules (LPM). The remaining components of Figure 13.5 are behaviorally-designed VHDL components.

The 8-bit input data are applied to the inputs of all four octal latches simultaneously. Data are written to a particular latch when a 2-bit **address** and a LOW on *WRITEn* cause an output of a 2-line-to-4-line decoder to enable the selected latch. For example, when *ADDR[1..0]* = 01 AND *WRITEn* = 0, decoder output *Y1* goes HIGH, activating the *ENABLE* input on latch 1. The values at *DATA_IN[7..0]* are transferred to latch 1 and stored there when *WRITEn* goes HIGH.

The latch outputs are applied to the data inputs of an octal 4-to-1 multiplexer. Recall that this circuit will direct one of four 8-bit inputs to an 8-bit output. The selected set of inputs correspond to the binary value at the MUX select inputs, which is the same as the address applied to the decoder in the write phase. The MUX output is directed to the *DATA_OUT* lines by an octal tristate bus driver, which is enabled by the *READ* line. To read the contents of latch 1, we set the address to 01, as before, and make the *READ* line HIGH. If *READ* is LOW, the *DATA_OUT* lines are in the high-impedance state.

4x8reg.scf

Figure 13.6 shows a simulation of the 4 × 8-bit memory. The address inputs change in a continuous binary sequence. For each address, a write pulse loads 8-bit data into the selected latch. After all four latches have been loaded, the latches are read in a rotating sequence. To read any new data from the memory, we would first have to write the new data into one or more of the latch locations.

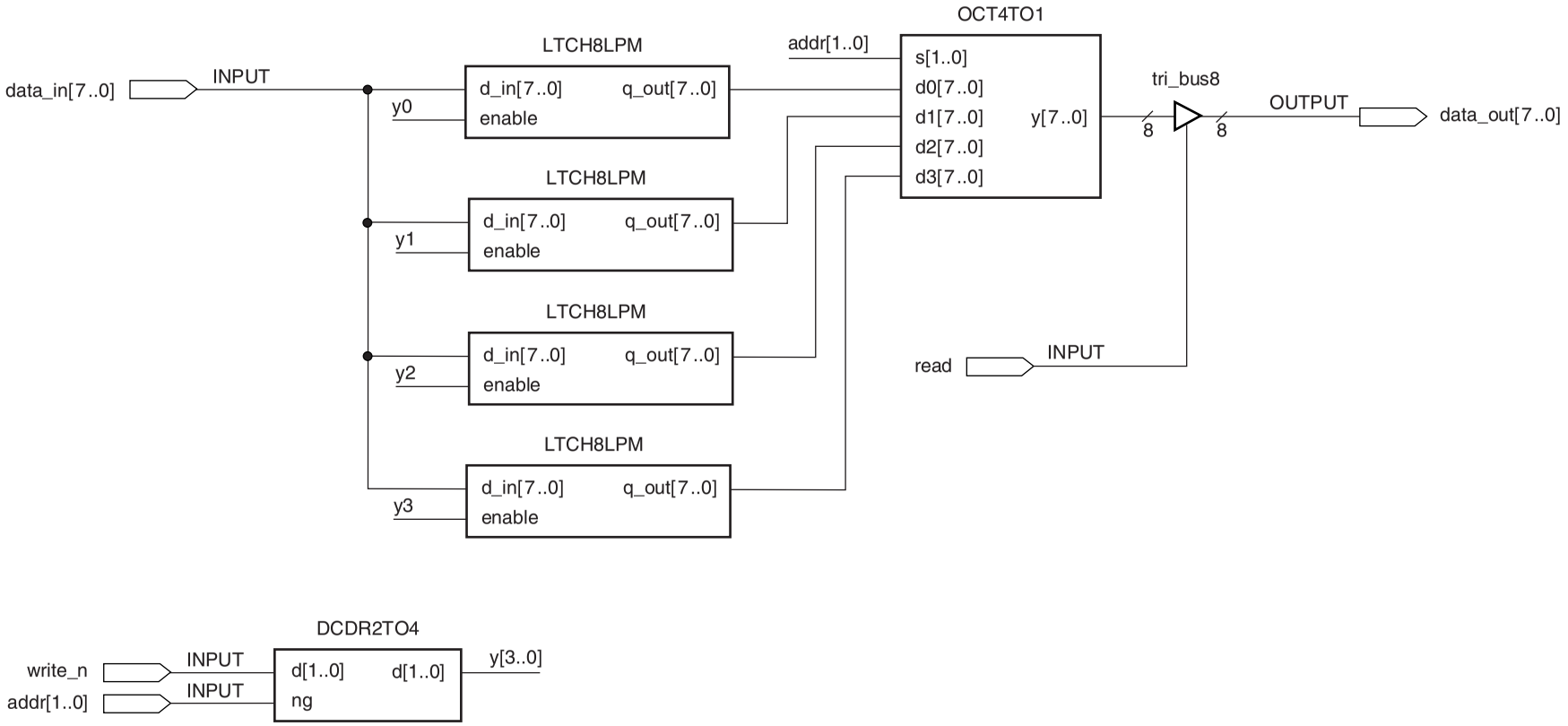


FIGURE 13.5
4 × 8-bit Memory from Octal Latches

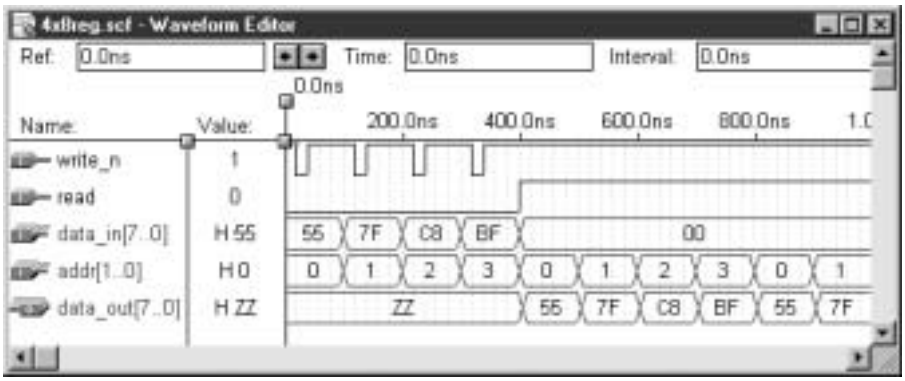


FIGURE 13.6
Simulation of 4 × 8 Memory

RAM and ROM

KEY TERMS

Random access memory (RAM) A type of memory device where data can be accessed in any order, that is, randomly. The term usually refers to random access read/write memory.

Read only memory (ROM) A type of memory where data are permanently stored and can only be read, not written.

The memory circuit in Figure 13.5 is one type of **random access memory,** or RAM. Data can be stored in or retrieved from any address at any time. The data can be accessed randomly, without the need to follow a sequence of addresses, as would be necessary in a sequential storage device such as magnetic tape.

RAM has come to mean random access read/write memory, memory that can have its data changed by a write operation, as well as have its data read. The data in another type of memory, called **read only memory,** or ROM, can also be accessed randomly, although it cannot be changed, or at least not changed as easily as RAM; there is no write function; hence the name "read only." Even though both types of memory are random access, we generally do not include ROM in this category.

Memory Capacity

KEY TERMS

b Bit.

B Byte.

K 1024 (= 2^{10}). Analogous to the metric prefix "k" (kilo-).

M 1,048,576 (= 2^{20}). Analogous to the metric prefix "M" (mega-).

The capacity of a memory device is specified by the address and data sizes. The circuit shown in Figure 13.5 has a capacity of 4 × 8 bits ("four-by-eight"). This tells us that the memory can store 32 bits, organized in groups of 8 bits at 4 different locations.

For large memories, with capacities of thousands or millions of bits, we use the shorthand designations **K** or **M** as prefixes for large binary numbers. The prefix K is analogous to, but not the same as, the metric prefix k (kilo). The metric kilo (lowercase k) indicates a multiplier of $10^3 = 1000$; the binary prefix K (uppercase) indicates a multiplier of $2^{10} = 1024$. Thus, one kilobit (Kb) is 1024 bits.

Similarly, the binary prefix M is analogous to the metric prefix M (mega). Both, unfortunately, are represented by uppercase M. The metric prefix represents a multiplier of $10^6 = 1{,}000{,}000$; the binary prefix M represents a value of $2^{20} = 1{,}048{,}576$. One megabit (Mb) is 1,048,576 bits. The next extension of this system is the multiplier G ($= 2^{30}$), which is analogous to the metric prefix G (giga; 10^9).

There is a move afoot to untangle all the inconsistencies in this notation and develop separate units for binary and metric applications, but to date, such new notation is not very widely used.

EXAMPLE 13.1

A small microcontroller system (i.e., a stand-alone microcomputer system designed for a particular control application) has a memory with a capacity of 64 Kb, organized as 8K × 8. What is the total memory capacity of the system in bits? What is the memory capacity in bytes?

Solution The total number of bits in the system memory is:

$$8\text{K} \times 8 = 8 \times 8 \times 1\text{K} = 64\text{ Kb} = 64 \times 1024\text{ bits} = 65{,}536\text{ bits}$$

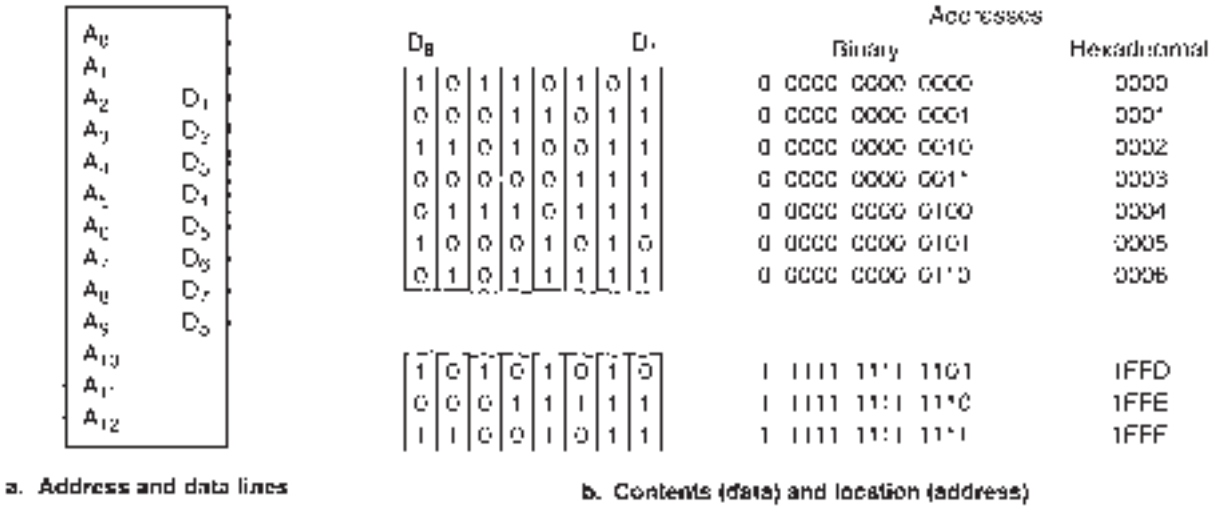
The number of bytes in system memory is:

$$\frac{64\text{ Kb}}{8\text{b/B}} = 8\text{ KB}$$

Usually, the range of numbers spanning 1K is expressed as the 1024 numbers from 0_{10} to 1023_{10} (0000000000_2 to 1111111111_2). This is the full range of numbers that can be expressed by 10 bits. In hexadecimal, the range of numbers spanning 1K is from 000H to 3FFH. The range of numbers in 1M is given as the full hexadecimal range of 20-bit numbers: 00000H to FFFFFH.

The range of numbers spanning 8K can be written in 13 bits ($8 \times 1\text{K} = 2^3 \times 2^{10} = 2^{13}$). The addresses in an 8K × 8 memory range from 0000000000000 to 1111111111111, or 0000 to 1FFF in hexadecimal. Thus, a memory device that is organized as 8K × 8 has 13 address lines and 8 data lines.

Figure 13.7 shows the address and data lines of an 8K × 8 memory and a map of its contents. The addresses progress in binary order, but the contents of any location are the



a. Address and data lines

b. Contents (data) and location (address)

FIGURE 13.7
Address and Data in an 8K × 8 Memory

last data stored there. Since there is no way to predict what those data are, they are essentially random. For example, in Figure 13.7, the byte at address 0000000000100_2 (0004H) is 01110111_2 (77H). (One can readily see the advantage of using hexadecimal notation.)

EXAMPLE 13.2

How many address lines are needed to access all addressable locations in a memory that is organized as 64K × 4? How many data lines are required?

Solution Address lines: $2^n = 64\text{K}$

$$64\text{K} = 64 \times 1\text{K} = 2^6 \times 2^{10} = 2^{16}$$

$$n = 16 \text{ address lines}$$

Data lines: There are 4 data bits for each addressable location. Thus, the memory requires 4 data lines.

Control Signals

Two memory devices are shown in Figure 13.8. The device in Figure 13.8a is a 1K × 4 random access read/write memory (RAM). Figure 13.8b shows 8K × 8 erasable programmable read only memory (EPROM). The address lines are designated by *A* and the data lines by *DQ*. The dual notation *DQ* indicates that these lines are used for both input *(D)* and output *(Q)* data, using the conventional designations of D-type latches. The input and output data are prevented from interfering with one another by a pair of opposite-direction tristate buffers on each input/output pin. One buffer goes to a memory cell input; the other comes from the memory cell output. The tristate outputs on the devices in Figure 13.8 allow the outputs to be electrically isolated from a system data bus that would connect several such devices to a microprocessor.

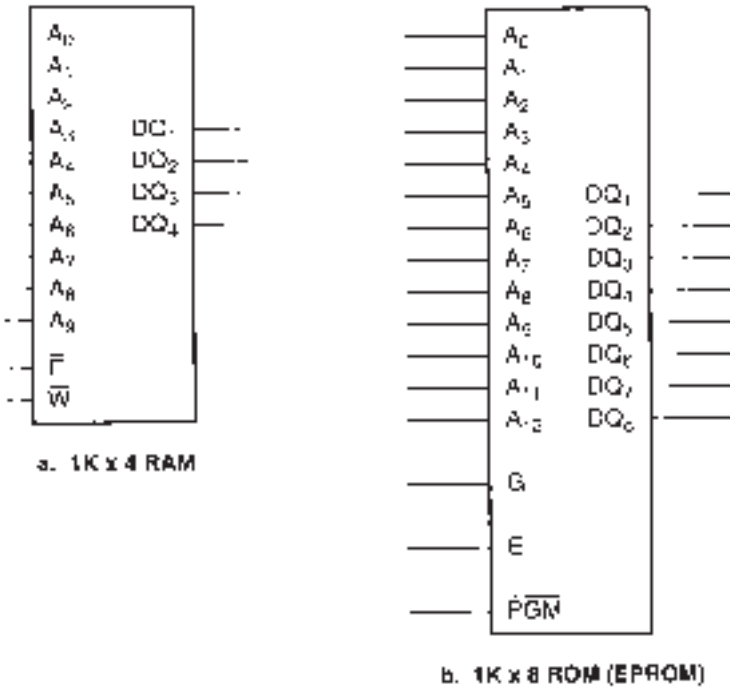


FIGURE 13.8
Address, Data, and Control Signals

In addition to the address and data lines, most memory devices, including those in Figure 13.8, have one or more of the following control signal inputs. (Different manufacturers use different notation, so several alternate designations for each function are listed.)

$\overline{E}$ (or $\overline{CE}$ or $\overline{CS}$). $\overline{\text{Enable}}$ (or $\overline{\text{Chip Enable}}$ or $\overline{\text{Chip Select}}$). The memory is enabled when this line is pulled LOW. If this line is HIGH, the memory cannot be written to or read from.

$\overline{W}$ (or $\overline{WE}$ or $R/\overline{W}$). $\overline{\text{Write}}$ (or $\overline{\text{Write Enable}}$ or Read/$\overline{\text{Write}}$). This input is used to select the read or write function when data input and output are on the same lines. When HIGH, this line selects the read (output) function if the chip is selected. When LOW, the write (input) function is selected.

$\overline{G}$ (or $\overline{OE}$). $\overline{\text{Gate}}$ (or $\overline{\text{Output Enable}}$). Some memory chips have a separate control to enable their tristate output buffers. When this line is LOW, the output buffers are enabled and the memory can be read. If this line is HIGH, the output buffers are in the high-impedance state. The chip select performs this function in devices without output enable pins.

The electrical functions of these control signals are illustrated in Figure 13.9.

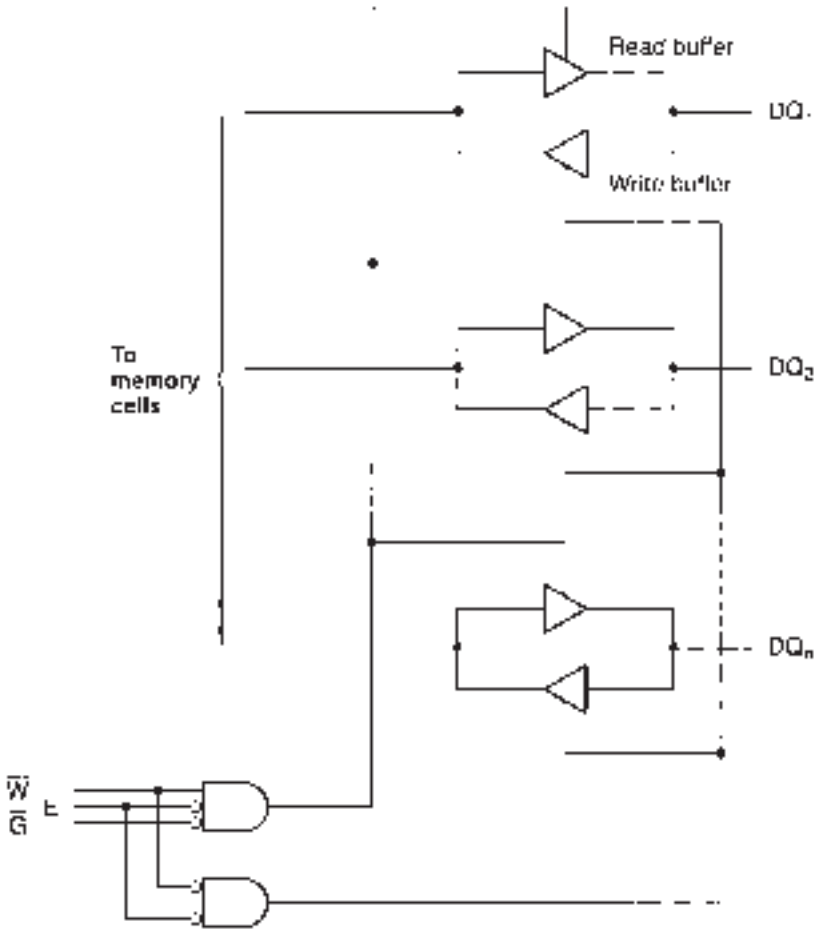


FIGURE 13.9
Memory Control Signals

13.2 Random Access Read/Write Memory (RAM)

KEY TERMS

Volatile A memory is volatile if its stored data are lost when electrical power is lost.

Static RAM A random access memory that can retain data indefinitely as long as electrical power is available to the chip.

Dynamic RAM A random access memory that cannot retain data for more than a few milliseconds without being "refreshed."

RAM cell The smallest storage unit of a RAM, capable of storing 1 bit.

Random access read/write memory (RAM) is used for temporary storage of large blocks of data. An important characteristic of RAM is that it is **volatile.** It can retain its stored data only as long as power is applied to the memory. When power is lost, so are the data. There are two main RAM configurations: static (SRAM) and dynamic (DRAM).

Static RAM (SRAM) consists of arrays of memory cells that are essentially flip-flops. Data can be stored in a static RAM cell and left there indefinitely, as long as power is available to the RAM.

A **dynamic RAM cell** stores a bit as the charged or discharged state of a small capacitor. Since the capacitor can hold its charge for only a few milliseconds, the charge must be restored ("refreshed") regularly. This makes a dynamic RAM (DRAM) system more complicated than SRAM, as it introduces a requirement for memory refresh circuitry.

DRAMs have the advantage of large memory capacity over SRAMs. At the time of this writing, the largest SRAMs have a capacity of about 4 Mb, whereas the largest DRAMs have a capacity of 256 Mb. DRAM modules, that is, groups of DRAM chips on a small circuit board, have capacities of up to 1 GB. These figures are constantly increasing and are never up to date for very long. (The most famous estimate of the growth rate of semiconductor memory capacity, Moore's law, estimates that it doubles every 18 months. My casual observation is that this is accurate to within an order of magnitude.)

Static RAM Cells

The typical static RAM cell consists of at least two transistors that are cross-coupled in a flip-flop arrangement. Other parts of the cell include pull-up circuitry that can be active (transistor switches) or passive (resistors) and some decoding/switching logic. Figure 13.10 shows an SRAM cell in three technologies: bipolar, NMOS, and CMOS.

Each of these cells can store 1 bit of data, a 0 or a 1, as the state of one of the transistors in the cell. The data are available in true or complement form, as the *BIT* and $\overline{BIT}$ outputs of the flip-flop.

All types of SRAM cells operate in more or less the same way. We will analyze the operation of the NMOS cell (Figure 13.10b) and then compare it to the other types.

Transistors Q_1 and Q_2 are permanently biased ON, making them into pull-up resistors. Channel width and length are chosen to give a resistance of about 1 kΩ. These NMOS load transistors are considered passive pull-ups, as they do not switch on and off.

A bit is stored as V_{DS3}, the drain voltage of Q_3 with respect to its source. If this voltage is HIGH, the gate of Q_4 is HIGH with respect to its source and Q_4 is biased ON. This completes a conduction path from the drain of Q_4 to its source, making V_{DS4} logic LOW. This LOW is fed back to the gate of Q_3, turning it OFF. There is no conduction path between the drain and source of Q_3, so $V_{DS3} = V_{DD}$ or logic HIGH. The cell is storing a 1.

This bit can be read by making the *ROW SELECT* line HIGH. This turns Q_5 and Q_6 ON, which puts the data onto the *BIT* and $\overline{BIT}$ lines where it can be read by other circuitry inside the RAM chip.

To change the cell contents to a 0, we make the *BIT* line LOW and the *ROW SELECT* line HIGH. The *ROW SELECT* line gives access to the cell by turning on Q_5 and Q_6, com-

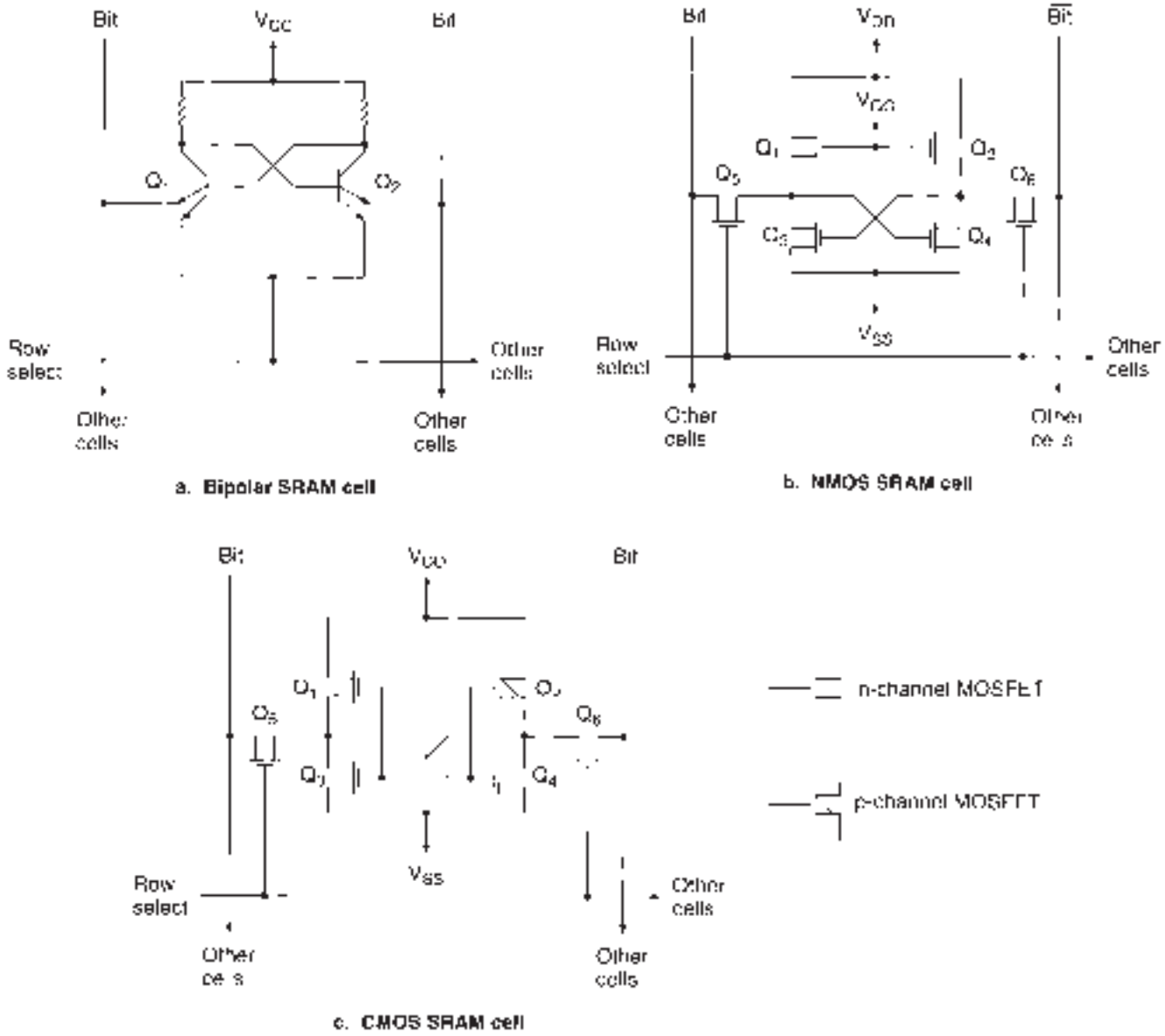


FIGURE 13.10
SRAM Cells

pleting the conduction path between the *BIT* lines and the flip-flop inputs. The LOW on the *BIT* line pulls the gate of Q_4 LOW, turning it OFF. This breaks the conduction path from Q_4 drain to source and makes $V_{DS4} = V_{DD}$, a logic HIGH. This HIGH is applied to the gate of Q_3, turning it ON. A conduction path is established between Q_3 drain and source, pulling the drain of Q_3 LOW. The cell now stores a logic 0.

NOTE

The contents of an SRAM cell must be changed by introducing a LOW on the *BIT* or the $\overline{BIT}$ line. The data cannot be changed by pulling an input HIGH without pulling the opposite input LOW. If a MOSFET gate is at the LOW state, a HIGH applied to that gate will be pulled down by the LOW level already existing there and will not cause the cell to change state.

The CMOS cell (Figure 13.10c) functions in the same way, except for the actions of Q_1 and Q_2. Q_1 and Q_3 are a complementary pair, as are transistors Q_2 and Q_4. For each of these pairs, when the *p*-channel transistor is ON, the *n*-channel is OFF, and vice versa. This arrangement is more energy efficient than the NMOS cell, since there is not the constant current drain associated with the load transistors. Power is consumed primarily during switching between states.

The main design goal of new memory technology is to increase speed and capacity while reducing power consumption and chip area. The NMOS cell has the advantage of being constructed from only one type of component. This makes it possible to manufacture more cells in the same chip area than can be done in either the CMOS or bipolar technologies. NMOS chips, however, are slower than bipolar. New advances in high-speed CMOS technologies have made possible CMOS memories that are as dense or denser than NMOS and faster. Because of this, NMOS will probably decline in importance over time.

Bipolar SRAMs can be either TTL, as shown in Figure 13.10a, or ECL, which is not shown. Of the two bipolar technologies, ECL is the faster. Historically, all bipolar SRAMs have had the advantage of speed over NMOS and CMOS chips. New CMOS devices however, have exceeded the speeds of TTL.

The bipolar SRAM cell is the least suitable for high-density memory. Both bipolar transistors and resistors are large components compared to a MOSFET. Thus, the bipolar cell is inherently larger than the CMOS or NMOS cell. Bipolar memories historically have been used when a small amount of high-speed memory is required.

The operation of the bipolar SRAM cell is similar to that of the MOSFET cells. In the quiescent state, the *ROW SELECT* line is LOW. In either the Read or the Write mode, the *ROW SELECT* line is HIGH. To change the data in the cell, pull one of the emitters LOW. When the emitter of Q_1 goes LOW, the cell contents become 0. When the emitter of Q_2 is pulled LOW, the cell contents are 1.

Static RAM Cell Arrays

KEY TERMS

Word-organized A memory is word-organized if one address accesses one word of data.

Word Data accessed at one addressable location.

Word length Number of bits in a word.

Static RAM cell arrays are arranged in a square or rectangular format, accessible by groups in rows and columns. Each column corresponds to a complementary pair of *BIT* lines and each row to a *ROW SELECT* line, as shown in Figure 13.11.

The column lines have MOSFETs configured as pull-up resistors at one end and a circuit called a sense amplifier at the other. The sense amp is a large RAM cell that amplifies the charge of an active storage cell on the same *BIT* line. Having a larger RAM cell as a sense amp allows the storage cells to be smaller, since each individual cell need not carry the charge required for a logic level output.

Figure 13.12 shows the block diagram of a 4 megabit (Mb) SRAM array, including blocks for address decoding and output circuitry. The RAM cells are arrayed in a pattern of 512 rows and 8192 columns for efficient packaging. When a particular address is applied to address lines $A_{18} \ldots A_0$, the row and column decoders select an SRAM cell in the memory array for a read or write by activating the associated sense amps for the column and the row select line for the cell.

The columns are further subdivided into groups of eight, so that one column address selects eight bits (one byte) for a read or write operation. Thus, there are 512 separate row addresses (9 bits) and 1024 separate column addresses (10 bits) for every unique group of 8 data bits, requiring a total of 19 address lines and 8 data lines. The capacity of the SRAM can be written as $512 \times 1024 \times 8$.

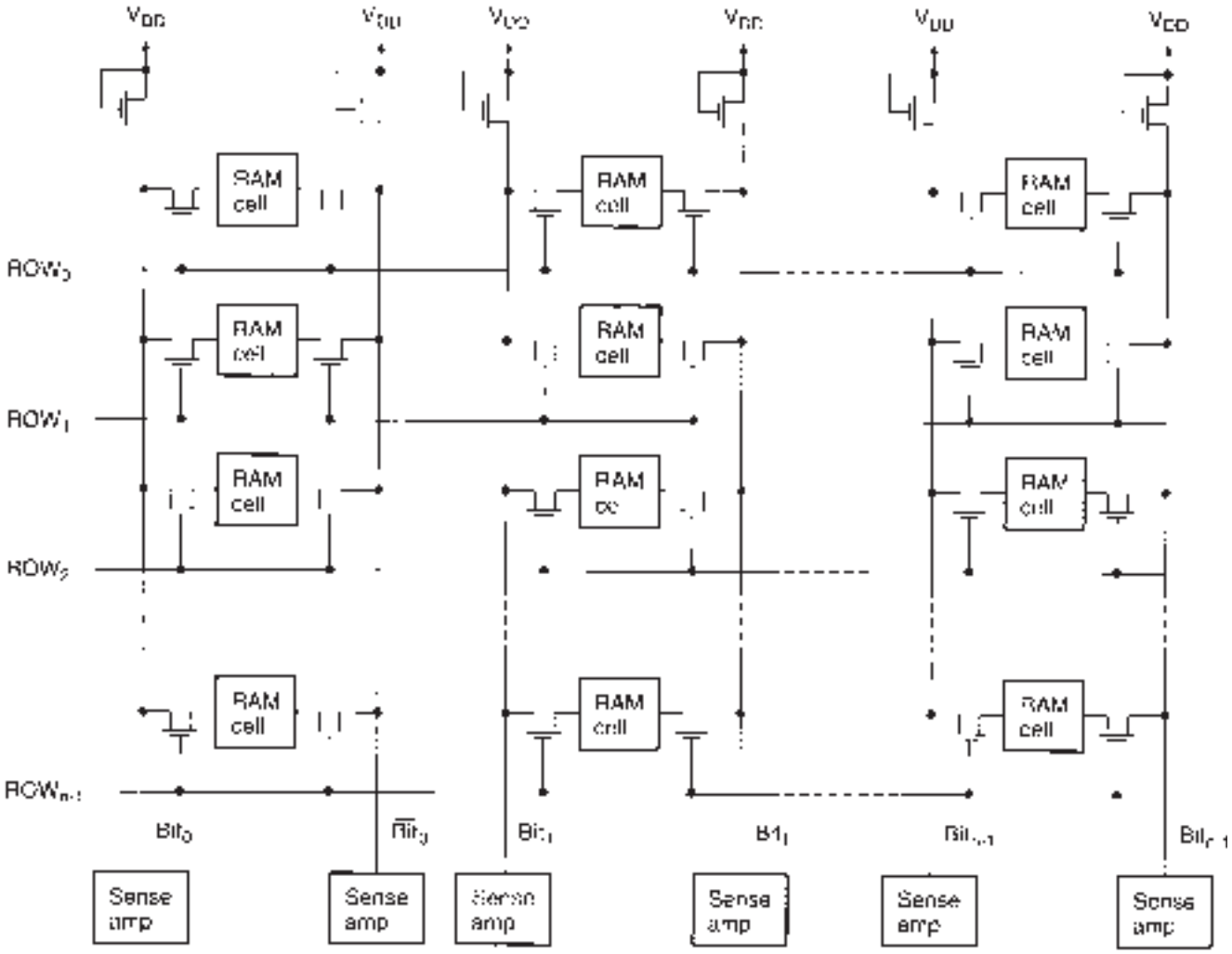
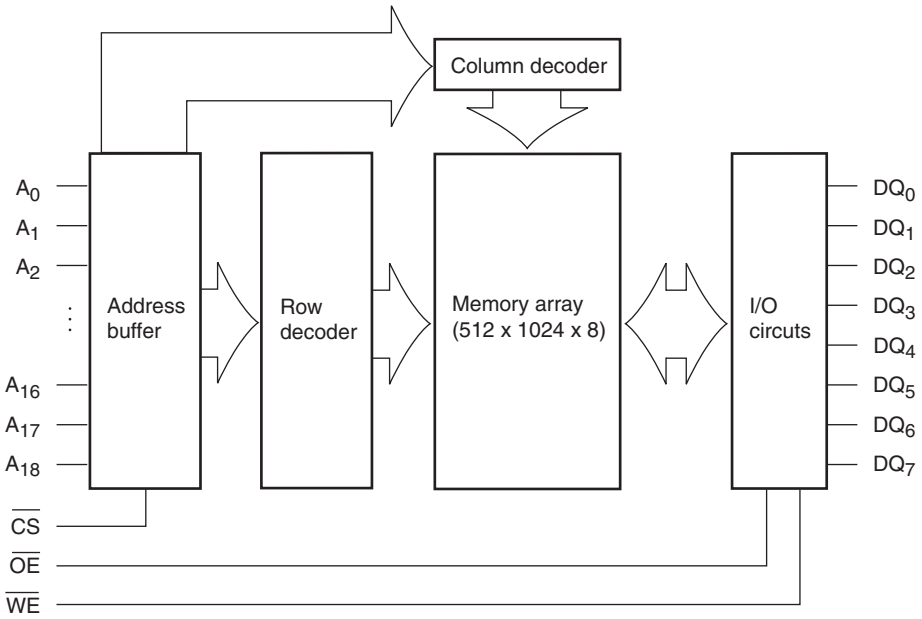


FIGURE 13.11
SRAM Cell Array

FIGURE 13.12
Block Diagram of a 4Mb (512 KB) SRAM



Since one address reads or writes 8 cells, we say that the SRAM in Figure 13.12 is **word-organized** and that the **word length** of the SRAM is 8 bits. Other popular word lengths for various memory arrays are 4, 16, 32, and 64 bits.

SECTION 13.2A REVIEW PROBLEM

13.1 If an SRAM array is organized as 512 × 512 × 16, how many address and data lines are required? How does the bit capacity of this SRAM compare to that of Figure 13.12?

Dynamic RAM Cells

KEY TERM

Refresh cycle The process that periodically recharges the storage capacitors in a dynamic RAM.

A dynamic RAM (DRAM) cell consists of a capacitor and a pass transistor, as shown in Figure 13.13. A bit is stored in the cell as the charged or discharged state of the capacitor. The bit location is read from or written to by activating the cell MOSFET via the Word Select line, thus connecting the capacitor to the *BIT* line.

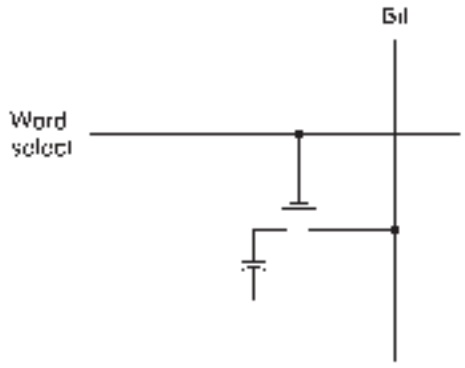


FIGURE 13.13
Dynamic RAM Cell

The major disadvantage of dynamic RAM is that the capacitor will eventually discharge by internal leakage current and must be recharged periodically to maintain integrity of the stored data. The recharging of the DRAM cell capacitors, known as refreshing the memory, must be done every 8 to 64 ms, depending on the device.

The **refresh cycle** adds an extra level of complication to the DRAM hardware and also to the timing of the read and write cycles, since the memory might have to be refreshed between read and write tasks. DRAM timing cycles are much more complicated than the equivalent SRAM cycles.

This inconvenience is offset by the high bit densities of DRAM, which are possible due to the simplicity of the DRAM cell. Up to 256 megabits of data can be stored on a single chip.

DRAM Cell Arrays

KEY TERMS

Bit-organized A memory is bit-organized if one address accesses one bit of data.

Address multiplexing A technique of addressing storage cells in a dynamic RAM that sequentially uses the same inputs for the row address and column address of the cell.

$\overline{\text{RAS}}$ Row address strobe. A signal used to latch the row address into the decoding circuitry of a dynamic RAM with multiplexed addressing.

$\overline{\text{CAS}}$ Column address strobe. A signal used to latch the column address into the decoding circuitry of a dynamic RAM with multiplexed addressing.

Dynamic RAM is sometimes **bit-organized** rather than word-organized. That is, one address will access one bit rather than one word of data. A bit-organized DRAM with a large capacity requires more address lines than a static RAM (e.g., 4 Mb × 1 DRAM requires 22 address lines (2^{22} = 4,194,304 = 4M) and 1 data line to access all cells).

In order to save pins on the IC package, a system of **address multiplexing** is used to specify the address of each cell. Each cell has a row address and a column address, which use the same input pins. Two negative-edge signals called **row address strobe** *(RAS)* and **column address strobe** *(CAS)* latch the row and column addresses into the DRAM's decoding circuitry. Figure 13.14 shows a simplified block diagram of the row and column addressing circuitry of a 1 Mb × 1 dynamic RAM.

Figure 13.15 shows the relative timing of the address inputs of a dynamic RAM. The first part of the address is applied to the address pins and latched into the row address buffers

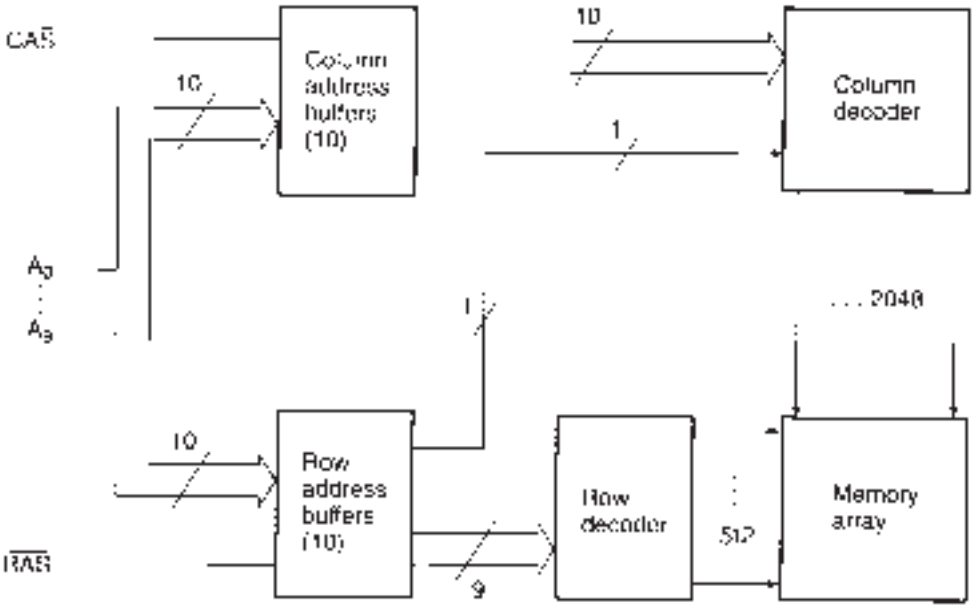


FIGURE 13.14
Row and Column Decoding in a 1M × 1 Dynamic RAM

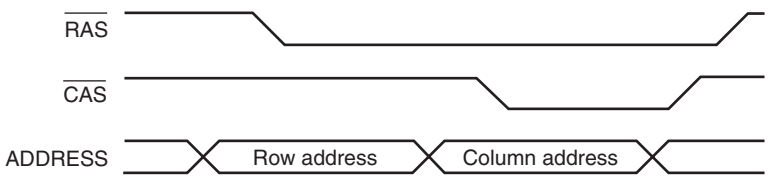


FIGURE 13.15
DRAM Address Latch Signals

when $\overline{\text{RAS}}$ goes LOW. The second part of the address is then applied to the address pins and latched into the column address buffers by the $\overline{\text{CAS}}$ signal. This allows a 20-bit address to be implemented with 12 pins: 10 address and 2 control lines. Adding another address line effectively adds 2 bits to the address, allowing access to 4 times the number of cells.

The memory cell array in Figure 13.14 is rectangular, not square. One of the Row Address lines is connected internally to the Column Address decoder, resulting in a 512-row-by-2048-column memory array.

One advantage to the rectangular format shown is that it cuts the memory refresh time in half, since all the cells are refreshed by accessing the rows in sequence. Fewer rows means a faster refresh cycle. All cells in a row are also refreshed by normal read and write operations.

SECTION 13.2B REVIEW PROBLEMS

13.2 How many address and data lines are required for the following sizes of dynamic RAM, assuming that each memory cell array is organized in a square format, with common Row and Column Address pins?

a. 1M × 1

b. 1M × 4

c. 4M × 1

13.3 Read Only Memory (ROM)

KEY TERMS

Hardware The electronic circuit of a digital or computer system.

Software Programming instructions required to make hardware perform specified tasks.

Firmware Software instructions permanently stored in ROM.

The main advantage of read only memory (ROM) over random access read/write memory (RAM) is that ROM is nonvolatile. It will retain data even when electrical power is lost to the ROM chip. The disadvantage of this is that stored data are difficult or impossible to change.

ROM is used for storing data required for tasks that never or rarely change, such as **software** instructions for a bootstrap loader in a personal computer or microcontroller. (The bootstrap loader—a term derived from the whimsical idea of pulling oneself up by one's bootstraps, that is, starting from nothing—is the software that gives the personal computer its minimum startup information. Generally, it contains the instructions needed to read a magnetic disk containing further operating instructions. This task is always the same for any given machine and is needed every time the machine is turned on, thus making it the ideal candidate for ROM storage.)

Software instructions stored in ROM are called **firmware.**

Mask-Programmed ROM

KEY TERM

Mask-programmed ROM A type of read only memory (ROM) where the stored data are permanently encoded into the memory device during the manufacturing process.

The most permanent form of read only memory is the **mask-programmed ROM,** where the stored data are manufactured into the memory chip. Due to the inflexibility of this type of ROM and the relatively high cost of development, it is used only for well-developed high-volume applications. However, even though development cost of a mask-programmed ROM is high, volume production is cheaper than for some other types of ROM.

Examples of applications suitable to mask-programmed ROM include:

- Bootstrap loaders and BIOS (basic input/output system) for PCs.
- Character generators (decoders that convert ASCII codes into alphanumeric characters on a CRT display)
- Function lookup tables (tables corresponding to binary values of trigonometric, exponential, or other functions)
- Special software instructions that must be permanently stored and never changed (firmware)

Figure 13.16 shows a ROM based on a matrix of MOSFETs. Each cell is manufactured with a MOSFET and its gate and source connections. LOWs are programmed by making a connection between the drain of the cell's MOSFET and the corresponding Bit

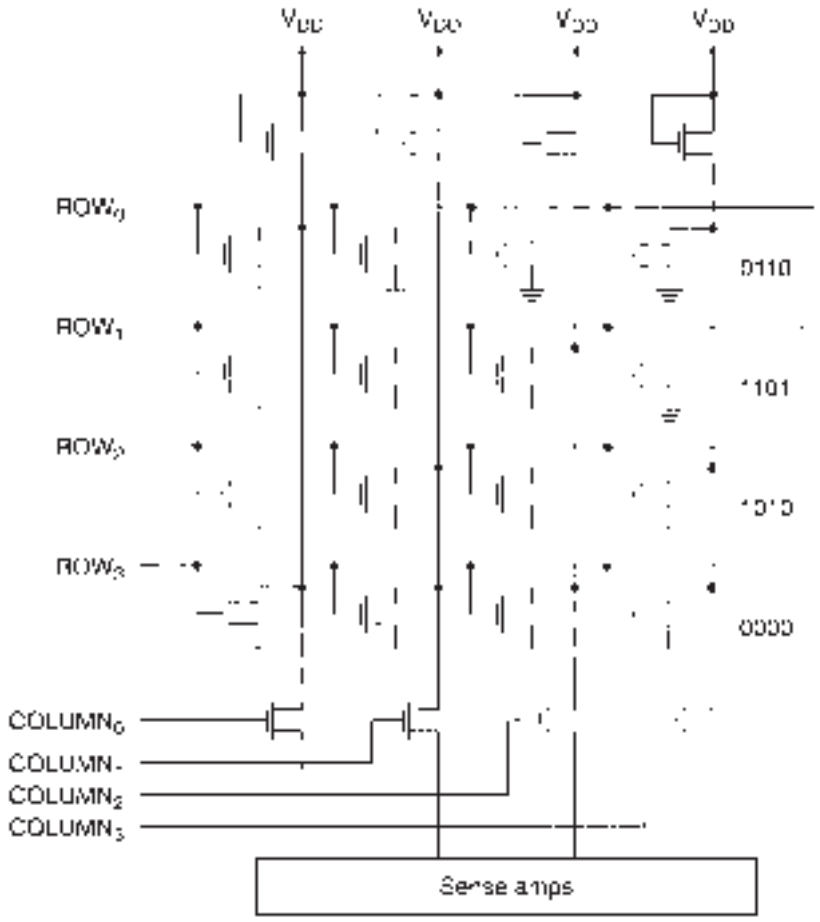


FIGURE 13.16
Mask-Programmed ROM

line. When the appropriate Row Select goes HIGH, the MOSFET turns ON, providing a path to ground from the selected Bit line. Cells programmed HIGH have no connection between the MOSFET drain and the Bit line, which thus cannot be pulled LOW when the cell is selected.

These connections can be made by a custom overlay of connections (a mask) on top of the standard-cell layer. The standard-cell-plus-custom-overlay format is cheaper to manufacture than custom cells for each bit, even if many of the MOSFETs are never used.

EPROM

KEY TERMS

EPROM Erasable programmable read only memory. A type of ROM that can be programmed ("burned") by the user and erased later, if necessary, by exposing the chip to ultraviolet radiation.

FAMOS FET Floating-gate avalanche MOSFET. A MOSFET with a second, "floating" gate in which charge can be trapped to change the MOSFET's gate-source threshold voltage.

Mask-programmed ROM is useful because of its nonvolatility, but it is hard to program and impossible to erase. **Erasable programmable read only memory** (EPROM) combines the nonvolatility of ROM with the ability to change the internal data if necessary.

This erasability is particularly useful in the development of a ROM-based system. Anyone who has built a complex circuit or written a computer program knows that there is no such thing as getting it right the first time. Modifications can be made easily and cheaply to data stored in an EPROM. Later, when the design is complete, a mask ROM version can be prepared for mass production. Alternatively, if the design will be produced in small numbers, the ROM data can be stored in EPROMs, saving the cost of preparing a mask-programmed ROM.

The basis of the EPROM memory cell is the **FAMOS FET,** whose circuit symbol is shown in Figure 13.17. FAMOS stands for floating-gate avalanche metal-oxide-semiconductor. ("Avalanche" refers to electron behavior in a semiconductor under certain bias conditions.) This is a MOSFET with a second, or floating, gate that is insulated from the first by a thin oxide layer.

The floating gate has no electrical contact with either the first gate or the source and drain terminals. As is the case in a standard MOSFET, conduction between drain and

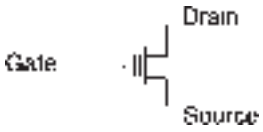


FIGURE 13.17
FAMOS FET

source terminals is effected by the voltage of the gate terminal with respect to the source. If this voltage is above a certain threshold level, the transistor will turn ON, allowing current to flow between drain and source.

In the unprogrammed state the FAMOS transistor's threshold voltage is low enough for the transistor to be turned ON by a 5-V read signal on the Row Select line. During the

programming operation, a relatively high voltage pulse (about 12 V to 25 V, depending on the device) on the Row Select line drives high-energy electrons into the floating gate and traps them there. This raises the threshold voltage of the programmed cell to a level where the cell won't turn ON when selected by a 5-V read.

The EPROM cells are configured so that an unprogrammed location contains a logic HIGH and the programming signal forces it LOW.

To erase an EPROM, the die (i.e., the silicon chip itself) must be exposed for about 20 to 45 minutes to high-intensity ultraviolet light of a specified wavelength (2537 angstroms) at a distance of 2.5 cm (1 inch). The high-energy photons that make up the UV radiation release the electrons trapped in the floating gate and restore the cell threshold voltages to their unprogrammed levels.

EPROMS are manufactured with a quartz window over the die to allow the UV radiation in. Since both sunlight and fluorescent light contain UV light of the right wavelength to erase the EPROM over time (several days to several years, depending on the intensity of the source), the quartz window should be covered by an opaque label after the EPROM has been programmed.

EPROM Application: Digital Function Generator

An EPROM can be used as the central component of a digital function generator. Other components in the system include a clock generator, a counter, a digital-to-analog converter, and an output op amp buffer. The portion of the circuit including the last three of these components is shown in Figure 13.18.

The generator can produce the usual analog waveforms—sine, square, triangle, sawtooth—and any other waveforms that you wish to store in the EPROM. A single cycle of

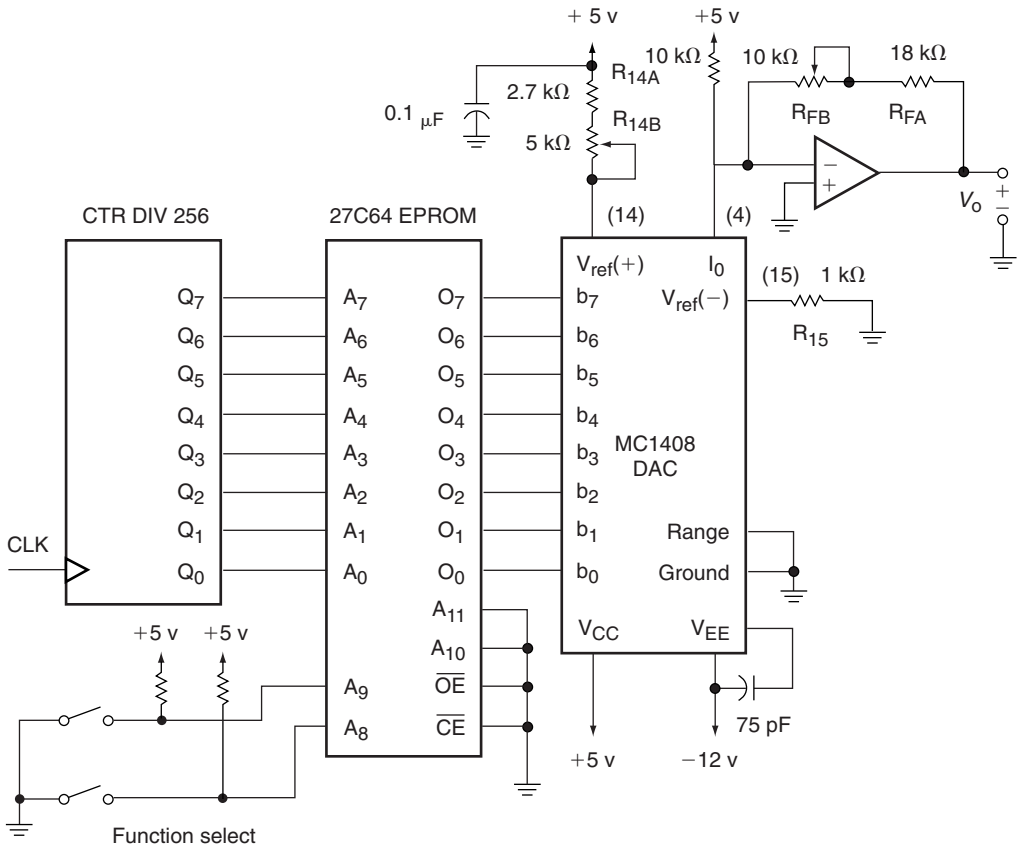


FIGURE 13.18
Digital Function Generator

each waveform is stored as 256 consecutive 8-bit numbers. For example, the data for one cycle of the sine waveform are stored at addresses 0000H to FFFFH, as shown in hex form in Table 13.1. (FF is maximum positive, 80 is zero, and 00 is maximum negative.) The square wave data are stored at addresses 0100 to 01FF, also shown in Table 13.1. The data for other functions, stored in subsequent 256-byte blocks, are not shown. A full list of the function data and an ANSI C program to generate an EPROM record file (Intel format) are included in Appendix E.

The counter and EPROM can also be implemented in a CPLD. Alternatively, a VHDL-designed state machine can replace the counter and EPROM, except for the sine function.

Table 13.1 EPROM Sine and Square Wave Data

SINE

Base Address	Byte Addresses															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	80	83	86	89	8C	8F	92	95	98	9C	9F	A2	A5	A8	AB	AE
0010	B0	B3	B6	B9	BC	BF	C1	C4	C7	C9	CC	CE	D1	D3	D5	D8
0020	DA	DC	DE	E0	E2	E4	E6	E8	EA	EC	ED	EF	F0	F2	F3	F5
0030	F6	F7	F8	F9	FA	FB	FC	FC	FD	FE	FE	FF	FF	FF	FF	FF
0040	FF	FF	FF	FF	FF	FF	FE	FE	FD	FC	FC	FB	FA	F9	F8	F7
0050	F6	F5	F3	F2	F0	EF	ED	EC	EA	E8	E6	E4	E2	E0	DE	DC
0060	DA	D8	D5	D3	D1	CE	CC	C9	C7	C4	C1	BF	BC	B9	B6	B3
0070	B0	AE	AB	A8	A5	A2	9F	9C	98	95	92	8F	8C	89	86	83
0080	7F	7C	79	76	73	70	6D	6A	67	63	60	5D	5A	57	54	51
0090	4F	4C	49	46	43	40	3E	3B	38	36	33	31	2E	2C	2A	27
00A0	25	23	21	1F	1D	1B	19	17	15	13	12	10	0F	0D	0C	0A
00B0	09	08	07	06	05	04	03	03	02	01	01	00	00	00	00	00
00C0	00	00	00	00	00	00	01	01	02	03	03	04	05	06	07	08
00D0	09	0A	0C	0D	0F	10	12	13	15	17	19	1B	1D	1F	21	23
00E0	25	27	2A	2C	2E	31	33	36	38	3B	3E	40	43	46	49	4C
00F0	4F	51	54	57	5A	5D	60	63	67	6A	6D	70	73	76	79	7C

SQUARE

Base Address	Byte Addresses															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0100	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0110	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0120	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0130	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0140	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0150	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0160	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0170	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

These configurations are designed and built as exercises in the lab manual that accompanies this book.

The most significant bits of the EPROM address select the waveform function by selecting a block of 256 address. The 8 least significant bits of the EPROM address are connected to an 8-bit (mod-256) counter, which continuously cycles through the 256 selected addresses. A 27C64 EPROM (8K × 8) has 13 address lines. After the eight lower lines are accounted for, the remaining five lines can be used to select up to 32 digital functions. With the two binary Function Select switches, we can potentially select 4 functions.

For example, to select the Sine function, inputs A_9 and A_8, which comprise the most significant digit of the EPROM address, are set to 00. Thus, the 8-bit counter cycles through addresses 0000–00FF, the location of the sine data. The Square Wave function is selected by setting A_9 and A_8 to 01, thus selecting the address block 0100–01FF. Other functions can be similarly selected.

The data at each address are sent to the D/A converter (MC1408), which, in combination with the op amp, is configured to produce a bipolar (both positive and negative) output. (We use a high slew rate op amp so that the generated square waves will have vertical sides.) The circuit generates a continuous waveform by retracing the data points in one 256-byte section of the EPROM over and over.

The DAC/op amp combination produces a maximum negative voltage for a hex input of 00, a 0 V output for an input of 80, and a maximum positive voltage for an input of FF. (You might wish to refer to the section Bipolar Operation of MC1408 in Chapter 12 for details of the DAC operation.)

You can see from the Sine function data in Table 13.1 that 8 bits are not sufficient to represent each of the 256 steps of a digital sine function as a unique number. The peaks of the waveform are changing too slowly to be represented accurately by an 8-bit quantization, and as a result, the top of the sine wave is flat for several clock pulses. (Mathematically, a sine function is tangential to a horizontal line at its peak. However, since tangential means touching at one point, the flat top is a distortion.) A unique number for each of 256 steps of a sine function needs at least 13 bits,[1] but this requires additional bits on the D/A converter input, and therefore a different DAC and an expanded memory word length.

The output frequency of the function generator is 1/256 of the clock rate. Given that the settling time of the MC1408 DAC is about 300ns, the maximum clock rate of the circuit is 1/300 ns = 3.33 MHz. At this rate, the output frequency is 3.33 MHz/256 = 13 kHz.

EEPROM

KEY TERM

EEPROM (or E^{2}PROM) Electrically erasable programmable read only memory. A type of read only memory that can be field-programmed and selectively erased while still in a circuit.

As was discussed in the previous section, EPROMs have the useful property of being erasable. The problem is that they must be removed from the circuit for erasure, and bits

[1]Bits required:
- 360°/256 steps = 1.40625°/step.
- Sine function changes most slowly at peak, so calculate $A \sin(90° \pm 1.40625°)$ to find smallest amplitude change.
- The smallest power-of-2 amplitude, A, for which $A \sin(90°) - A \sin(90° \pm 1.40625°) \geq 1$ is 4096.
- The amplitude range $-4096 \leq A \leq 4095$ can be represented by a 13-bit number.

cannot be selectively erased; the whole memory cell array is erased as a unit.

Electrically erasable programmable read only memory (EEPROM or E^2PROM) provides the advantages of EPROM along with the additional benefit of allowing erasure of selected bits while the chip is in the circuit; it combines the read/write properties of RAM with the nonvolatility of ROM. EEPROM is useful for storage of data that need to be changed occasionally, but that must be retained when power is lost to the EEPROM chip. One example is the memory circuit in an electronically tuned car radio that stores the channel numbers of local stations.

Like the UV-erasable EPROM, the memory cell of the EEPROM is based on the FAMOS transistor. Unlike the EPROM, the FAMOS FET is coupled with a standard MOSFET, as shown in Figure 13.19.

The FAMOS FET is programmed in the same way as UV-erasable EPROM: a programming voltage pulse (V_{PP}) drives high-energy electrons into the floating gate of the

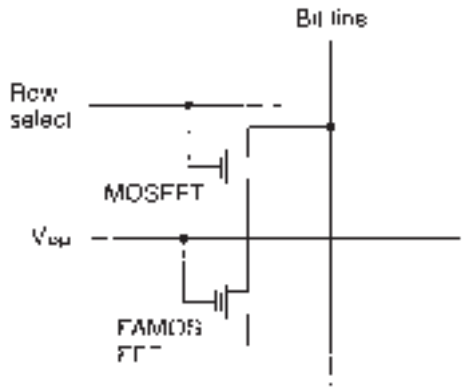


FIGURE 13.19
EEPROM Cell

FAMOS transistor, where they remain trapped and change the threshold voltage of the transistor. The cell is read by keeping the programming line at 5 V and making the cell's Row Select line HIGH. The FAMOS transistor will or will not turn on, depending on its programmed state.

The FAMOS transistors used in EPROM and EEPROM differ in one important respect. The EEPROM transistor is manufactured with a very thin oxide layer between the drain and the upper (nonfloating) gate. This construction allows trapped electrons in the floating gate to be forced out electrically, thus erasing the cell contents.

Given the obvious advantages of EEPROM, why doesn't it replace all other types of memory? There are several reasons:

1. EEPROM has a much slower access time than RAM and is thus not good for high-speed applications.
2. The currently available EEPROMs have significantly smaller bit capacities than commercially available RAM (especially dynamic RAM) and EPROM.
3. EEPROM has a fixed number of write/erase cycles, typically 100,000. After that, new data cannot be programmed into the device.

Flash Memory

KEY TERMS

Flash memory A nonvolatile type of memory that can be programmed and erased in sectors, rather than byte-at-a-time.

Sector A segment of flash memory that forms the smallest amount that can be erased and reprogrammed at one time.

Boot block A sector in a flash memory reserved for primary firmware.

Top boot block A boot block sector in a flash memory placed at the highest address in the memory.

Bottom boot block A boot block sector in flash memory paced at the lowest address in the memory.

A popular variation on EEPROM is **flash memory.** This type of nonvolatile memory generally has a larger byte capacity (e.g., 8 Mb) than EEPROM devices and thus can be used to store fairly large amounts of firmware, such as the BIOS (basic input/output system) of a PC.

A flash memory is divided into **sectors,** groups of bytes that are programmed and erased at one time. One sector is designated as the **boot block,** which is either the sector with the highest **(top boot block)** or lowest **(bottom boot block)** address. The primary firmware is usually stored in the boot block, with the idea that the system using the flash memory is configured to look there first for firmware instructions. The boot block can also be protected from unauthorized erasure or modification (e.g., by a virus), thus adding a security feature to the device.

Figure 13.20 shows the arrangement of sectors of a 512K × 8-bit (4 Mb) flash memory with a bottom boot block architecture. The range of addresses are shown alongside the blocks. For example, sector S0 (the boot block) has a 16 KB address range of 00000H to 03FFFH. Sector S1 has an 8 KB address range from 04000H to 05FFFH. The first 64 KB of the memory are divided into one 16 KB, two 8 KB, and one 32 KB sectors. The remainder of the memory is divided into equal 64 KB sectors. Note that even though the boot block is drawn at the top of Figure 13.20, it is a bottom boot block because it is the sector with the lowest address.

A flash memory with a top boot block would have the same proportions given over to its sectors, but mirror-image to the diagram in Figure 13.20. That is, S10 (boot block) would be a 16 KB sector from 7C000H to 7FFFFH. The other sectors would be identical to the bottom boot block architecture, but in reverse order.

As with other EEPROM devices, a flash memory can be erased and reprogrammed while installed in a circuit. The memory cells in a flash device have a limited number of program/erase cycles, like other EEPROMs. The sector architecture of the flash memory makes it faster to erase and program than other EEPROM-based memories which must erase or program bytes one at a time. This same characteristic makes it unsuitable for use as system RAM, which must be able to program single bytes.

SECTION 13.3 REVIEW PROBLEM

13.3 A flash memory has a capacity of 8 Mb, organized as 1M × 8-bit. List the address range for the 32 KB boot block sector of the memory if the device has a bottom boot block architecture and if it has a top boot block architecture.

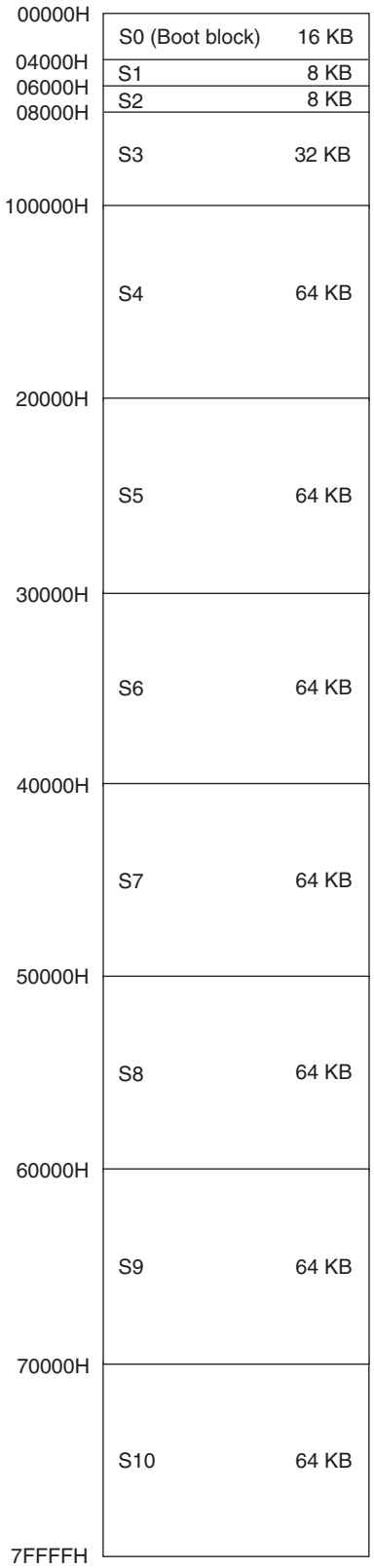


FIGURE 13.20
Sectors in a 512K × 8b Flash Memory (Bottom Boot Block)

13.4 Sequential Memory: FIFO and LIFO

KEY TERMS

Sequential memory Memory in which the stored data cannot be read or written in random order, but must be addressed in a specific sequence.

FIFO First-in first-out. A sequential memory in which the stored data can be read only in the order in which it was written.

Queue A FIFO memory.

LIFO Last-in first-out. A sequential memory in which the last data written are the first data read.

Stack A LIFO memory.

The RAM and ROM devices we have examined up until now have all been random access devices. That is, any data could be read from or written to any sequence of addresses in any order. There is another class of memory in which the data must be accessed in a particular order. Such devices are called **sequential memory**.

There are two main ways of organizing a sequential memory—as a **queue** or as a **stack**. Figure 13.21 shows the arrangement of data in each of these types of memory.

A queue is a **first-in first-out** (FIFO) memory, meaning that the data can be read only in the same order they are written, much as railway cars always come out of a tunnel in the same order they go in.

One common use for FIFO memory is to connect two devices that have different data rates. For instance, a computer can send data to a printer much faster than the printer can use it. To keep the computer from either waiting for the printer to print everything or periodically interrupting the computer's operation to continue the print task, data can be sent in a burst to a FIFO, where the printer can read them as needed. The only proviso is that there

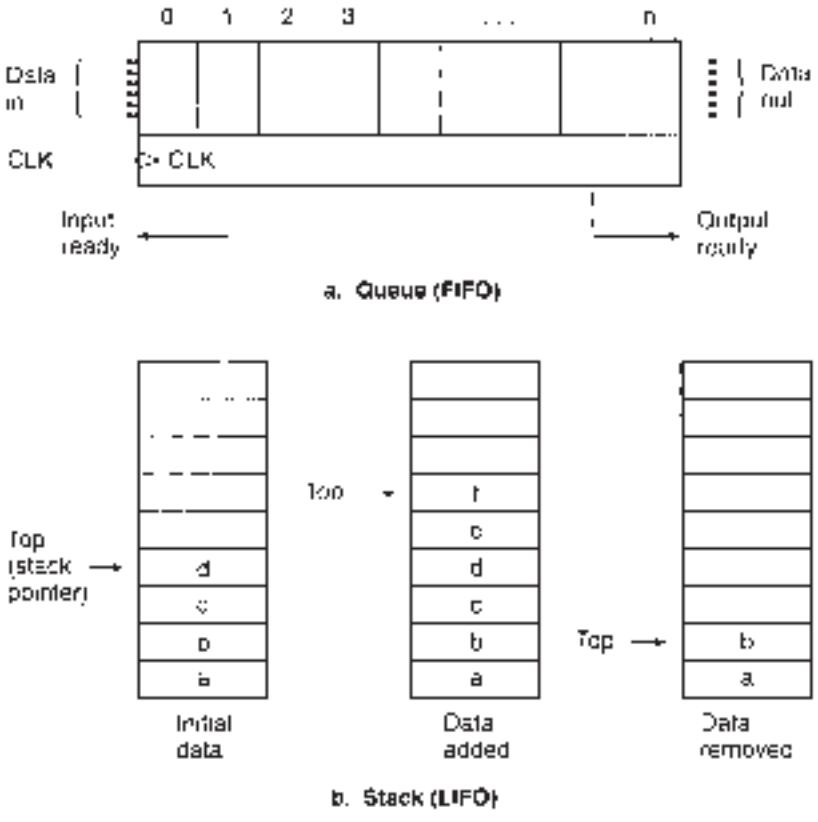


FIGURE 13.21
Sequential Memory

must be some logic signal to the computer telling it when the queue is full and not to send more data and another signal to the printer letting it know that there are some data to read from the queue.

The **last-in-first-out** (LIFO), or stack, memory configuration, also shown in Figure 13.21, is not available as a special chip, but rather is a way of organizing RAM in a memory system.

The term "stack" is analogous to the idea of a spring-loaded stack of plates in a cafeteria line. When you put a bunch of plates on the stack, they settle into the recessed storage area. When a plate is removed, the stack springs back slightly and brings the second plate to the top level. (The other plates, of course, all move up a notch.) The top plate is the only one available for removal from the stack, and plates are always removed in reverse order from that in which they were loaded.

Figure 13.21b shows how data are transferred to and from a LIFO memory. A block of addresses in a RAM is designated as a stack, and one or two bytes of data in the RAM store a number called the stack pointer, which is the current address of the top of the stack.

In Figure 13.21, the value of the stack pointer changes with every change of data in the stack, pointing to the last-in data in every case. When data are removed from the stack, the stack pointer is used to locate the data that must be read first. After the read, the stack pointer is modified to point to the next-out data. Some stack configurations have the stack pointer painting to the next empty location on the stack.

The most common application for LIFO memory is in a computer system. If a program is interrupted during its execution by a demand from the program or some piece of hardware that needs attention, the status of various registers within the computer are stored on a stack and the computer can pay attention to the new demand, which will certainly change its operating state. After the interrupting task is finished, the original operating state of the computer can be taken from the top of the stack and reloaded into the appropriate registers, and the program can resume where it left off.

SECTION 13.4 REVIEW PROBLEM

13.4 State the main difference between a stack and a queue.

13.5 Dynamic RAM Modules

KEY TERMS

Memory module A small circuit board containing several dynamic RAM chips.

Single in-line memory module (SIMM) A memory module with DRAMs and connector pins on one side of the board only.

Dual in-line memory module (DIMM) A memory module with DRAMs and connector pins on both sides of the board.

Dynamic RAM chips are often combined on a small circuit board to make a **memory module**. This is because the data bus widths of systems requiring the DRAMs are not always the same as the DRAMs themselves. For example, Figure 13.22 shows how four 64M $\times$ 8 DRAMs are combined to make a 64M $\times$ 32 memory module. The block diagram of the module is shown in Figure 13.22, and the mechanical outline is shown in Figure 13.23. The data input/output lines are separate from one another so that there are 32 data I/Os *(DQ)*. The address lines (ADDR[12..0]) for the module are parallel on all chips. With address multiplexing, this 13-bit address bus yields a 26-bit address, giving a 64M address range. Chip selects (CS) for all devices are connected together so that selecting the module selects all chips on the module.

This particular memory module is configured as a **single in-line memory module (SIMM)**, which has the DRAM chips and pin connections on one side of the board only. A

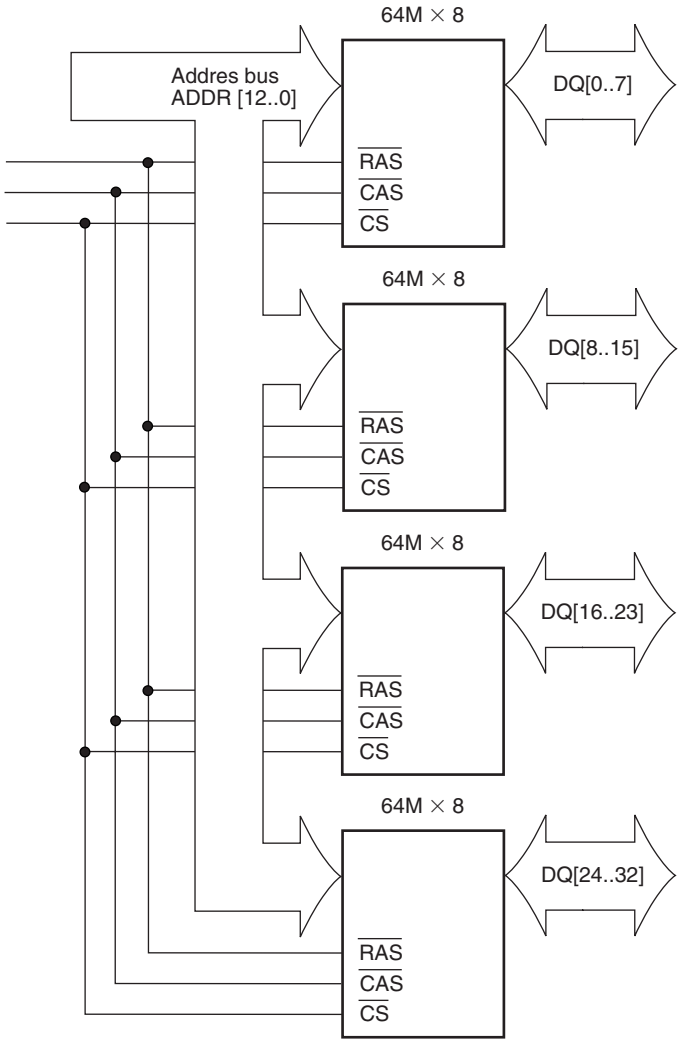


FIGURE 13.22
SIMM Block Diagram

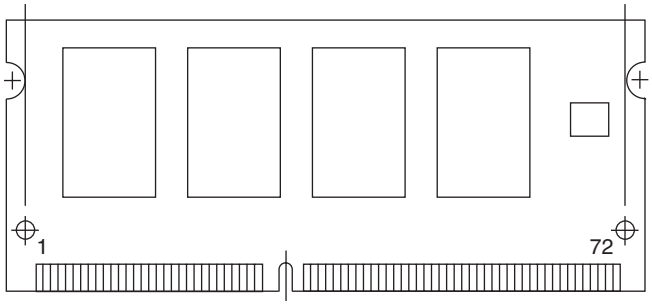


FIGURE 13.23
SIMM Layout

dual in-line memory module (DIMM) has the DRAMs mounted on both sides of the circuit board and pin connections on both sides of the board as well.

SECTION 13.5 REVIEW PROBLEM

13.5 A SIMM has a capacity of 16M × 32. How many 16M × 8 DRAMs are required to make this SIMM? How many address lines does the SIMM require? How should the DRAMs be connected?

13.6 Memory Systems

KEY TERMS

Address decoder A circuit enabling a particular memory device to be selected by the address bus of a larger memory system.

Address space A block of addresses in a memory system.

Bus contention The condition that results when two or more devices try to send data to a bus at the same time. Bus contention can damage the output buffers of the devices involved.

Memory map A diagram showing the total address space of a memory system and the placement of various memory devices within that space.

In the section on memory modules, we saw how multiple memory devices can be combined to make a system that has the same number of addressable locations as the individual devices making up the system, but with a wider data bus. We can also create memory systems where the data I/O width of the system is the same as the individual chips, but where the system has more addressable locations than any chip within the system.

In such a system, the data I/O and control lines from the individual memory chips are connected in parallel, as are the lower bits of an address bus connecting the chips. However, it is important that only one memory device be enabled at any given time, in order to avoid **bus contention**, the condition that results when more than one output attempts to drive a common bus line. To avoid bus contention, one or more additional address lines must be decoded by an **address decoder** that allows only one chip to be selected at a time.

Figure 13.24 shows two 32K × 8 SRAMs connected to make a 64K × 8 memory system. A single 32K × 8 SRAM, as shown in Figure 13.24a, requires 15 address lines, 8 data lines, a write enable (WE), and chip select (CS) line. To make a 64K × 8 SRAM system, all of these lines are connected in parallel, except the CS lines. In order to enable only one at a time, we use one more address line, A_{15}, and enable the top SRAM when $A_{15} = 0$ and the bottom SRAM when $A_{15} = 1$.

The address range of one 32K × 8 SRAM is given by the range of states of the address lines A[14..0]:

Lowest single-chip address: 000 0000 0000 0000 = 0000H
Highest single-chip address: 111 1111 1111 1111 = 7FFFH

The address range of the whole system must also account for the A_{15} bit:

Lowest system address: 0000 0000 0000 0000 = 0000H
Highest system address: 1111 1111 1111 1111 = FFFFH

Within the context of the system, each individual SRAM chip has a range of addresses, depending on the state of A_{15}. Assume $SRAM_0$ is selected when $A_{15} = 0$ and $SRAM_1$ is selected when $A_{15} = 1$.

Lowest $SRAM_0$ address: 0000 0000 0000 0000 = 0000H
Highest $SRAM_0$ address: 0111 1111 1111 1111 = 7FFFH
Lowest $SRAM_1$ address: 1000 0000 0000 0000 = 8000H
Highest $SRAM_1$ address: 1111 1111 1111 1111 = FFFFH

Figure 13.25 shows a **memory map** of the 64K × 8 SRAM system, indicating the range of addresses for each device in the system. The total range of addresses in the system is called the **address space**.

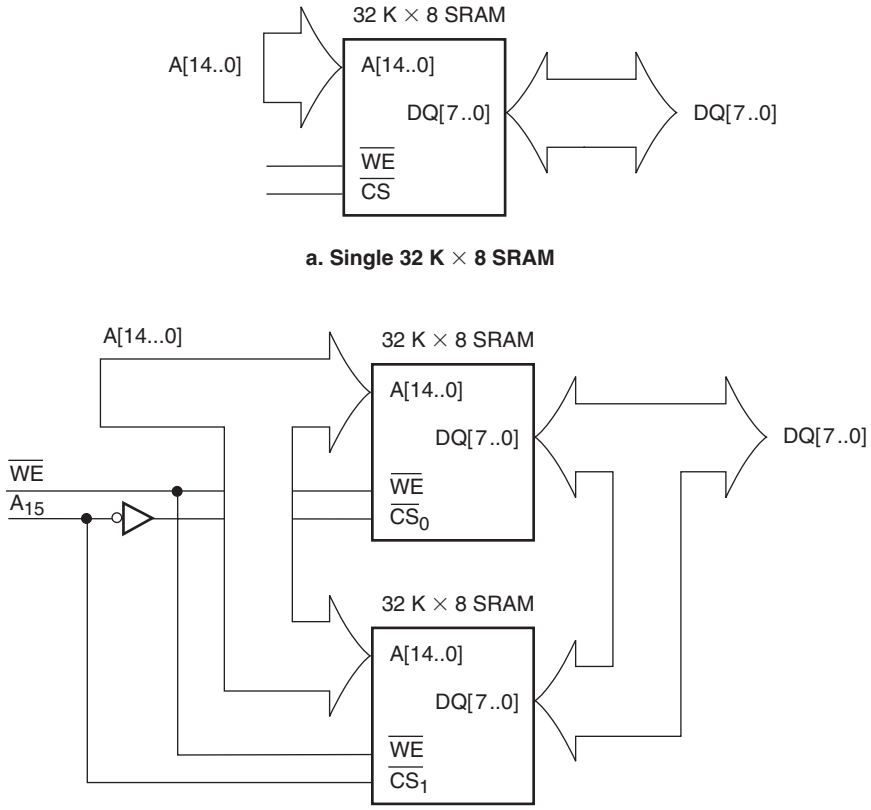


FIGURE 13.24
Expanding Memory Space

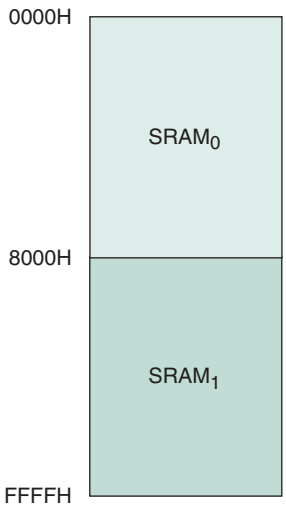


FIGURE 13.25
Memory Map

EXAMPLE 13.3

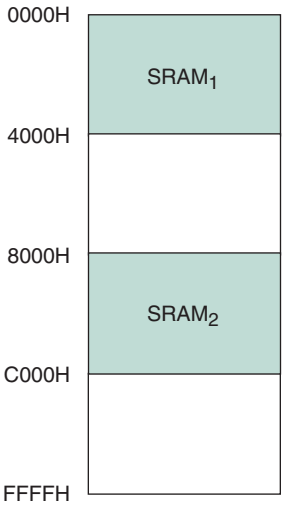


FIGURE 13.26
Memory Map Showing Non-contiguous Decoded Blocks.

Figure 13.26 shows a memory map for a system with an address space of 64K (16 address lines). Two 16K × 8 blocks of SRAM are located at start addresses of 0000H and 8000H, respectively. Sketch a memory system that implements the memory map of Figure 13.26.

Solution A 16K address block requires 14 address lines, since

$$16\text{K} = 16 \times 1024 = 2^4 \times 2^{10} = 2^{14}$$

The entire 64K address space requires 16 address lines, since

$$64\text{K} = 64 \times 1024 = 2^6 \times 2^{10} = 2^{16}$$

The highest address in a block is the start address plus the block size.

	16K block size:	11 1111 1111 1111 = 3FFFH
$SRAM_0$:	Lowest address:	0000 0000 0000 0000 = 0000H
	Highest address:	0011 1111 1111 1111 = 3FFFH
$SRAM_2$:	Lowest Address:	1000 0000 0000 0000 = 8000H
	Highest Address:	1011 1111 1111 1111 = BFFFH

$A_{15}A_{14} = 00$ for the entire range of the $SRAM_0$ block. $A_{15}A_{14} = 10$ for the entire $SRAM_2$ range. These can be decoded by the gates shown in Figure 13.27.

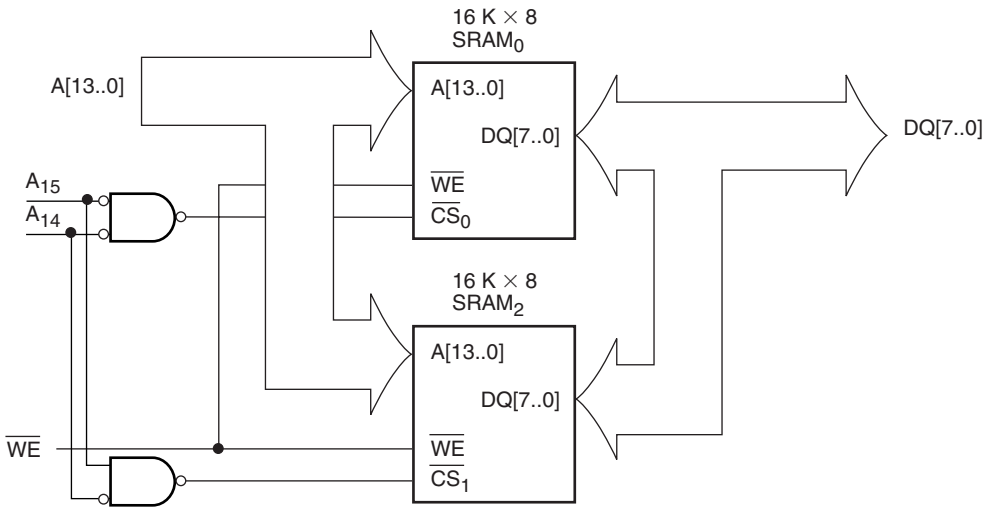


FIGURE 13.27
Example 13.3
32K × 8 SRAM with non-continguous blocks.

Address Decoding with *n*-line-to-*m*-line Decoders

Figure 13.28 shows a 64K memory system with four 16K chips: one EPROM at 0000H and three SRAMs at 4000H, 8000H, and C000H, respectively. In this circuit, the address decoding is done by a 2-line-to-4-line decoder, which can be an off-the-shelf MSI decoder, such as a 74HC139 decoder or a PLD-based design.

Table 13.2 shows the address ranges decoded by each decoder output. The first two address bits are the same throughout any given address range. Figure 13.29 shows the memory map for the system.

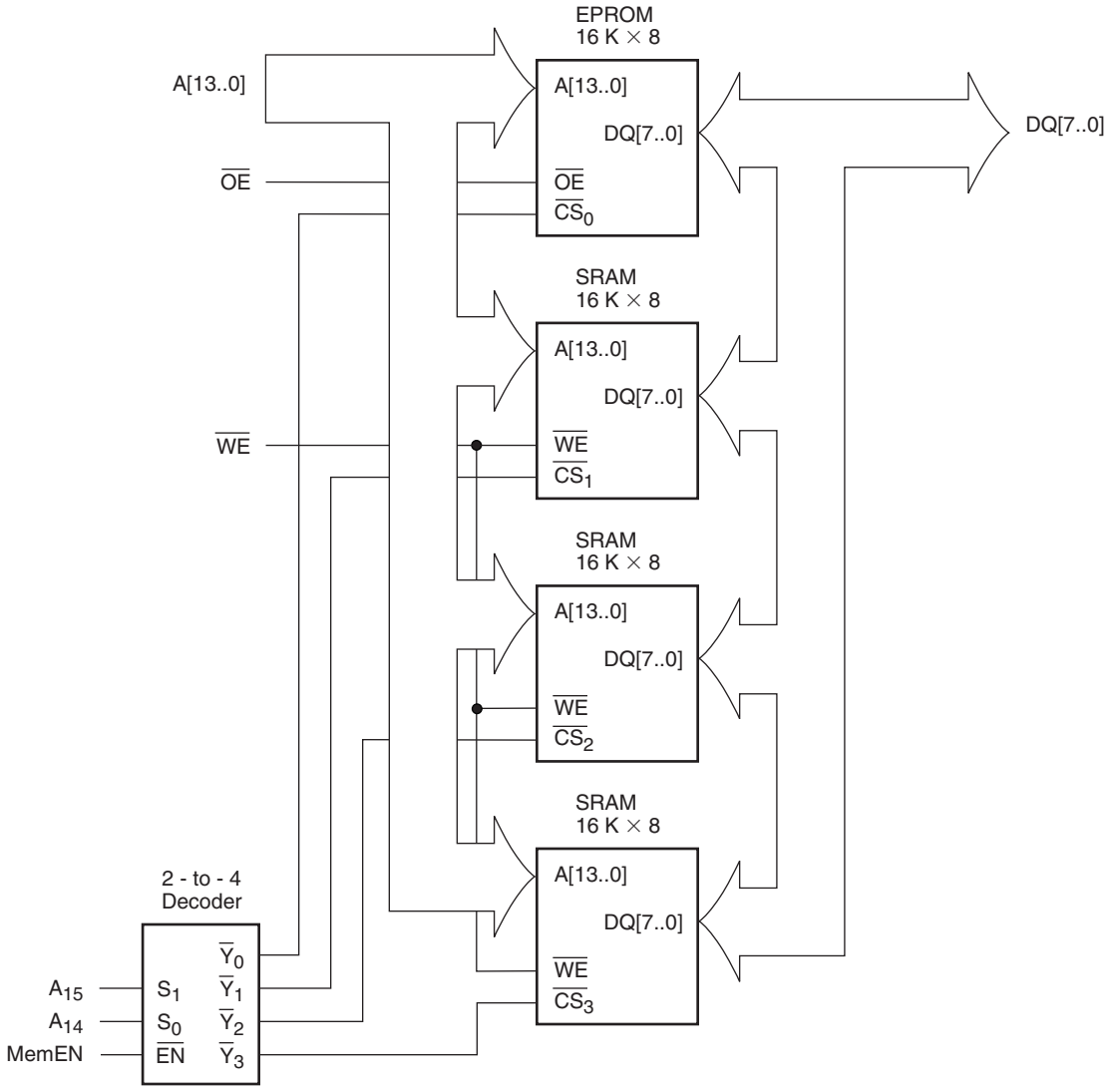
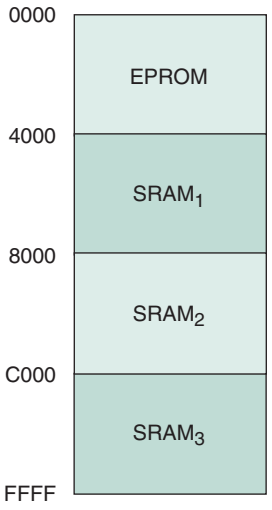


FIGURE 13.28
64K Memory System

Table 13.2 Address Decoding for Figure 13.28

A_{15}	A_{14}	Active Decoder Output	Device	Address Range
0	0	Y_0	EPROM	0000 0000 0000 0000 = 0000H 0011 1111 1111 1111 = 3FFFH
0	1	Y_1	$SRAM_1$	0100 0000 0000 0000 = 4000H 0111 1111 1111 1111 = 7FFFH
1	0	Y_2	$SRAM_2$	1000 0000 0000 0000 = 8000H 1011 1111 1111 1111 =

FIGURE 13.29
Memory Map for Figure 13.28



SECTION 13.6 REVIEW PROBLEM

13.6 Calculate the number of 128K memory blocks will fit into a 1M address space. Write the start addresses for the blocks.

SUMMARY

1. A memory is a device that can accept data and store them for later recall.
2. Data are located in a memory by an address, a binary number at a set of address inputs that uniquely locates the block of data.
3. The operation that stores data in a memory is called the write function. The operation that recalls the stored data is the read function. These functions are controlled by functions such as write enable ($\overline{WE}$), chip select ($\overline{CS}$), and output enable ($\overline{OE}$).
4. RAM is random access memory. RAM can be written to and read from in any order of addresses. RAM is volatile. That is, it loses its data when power is removed from the device.
5. ROM is read only memory. Original ROM devices could not be written to at all, except at the time of manufacture. Modern variations can also be written to, but not as easily as RAM. ROM is nonvolatile; it retains its data when power is removed from the device.
6. Memory capacity is given as $m \times n$ for m addressable locations and an n-bit data bus. For example, a 64K $\times$ 8 memory has 65,536 addressable locations, each with 8-bit data.
7. Large blocks of memory are designated with the binary prefixes K ($2^{10} = 1024$), M ($2^{20} = 1{,}048{,}576$), and G ($2^{30} = 1{,}073{,}741{,}824$).
8. RAM can be divided into two major classes: static RAM (SRAM) and dynamic RAM (DRAM). SRAM retains its data as long as power is applied to the device. DRAM requires its data to be refreshed periodically.
9. Typically DRAM capacity is larger than SRAM because DRAM cells are smaller than SRAM cells. An SRAM cell is essentially a flip-flop consisting of several transistors. A DRAM cell has only one transistor and a capacitor.
10. RAM cells are arranged in rectangular arrays for efficient packaging. Internal circuitry locates each cell at the intersection of a row and column within the array.
11. For packaging efficiency, DRAM addresses are often multiplexed so that the device receives half its address as a row address, latched in to the device by a $\overline{RAS}$ (row address strobe) signal and the second half as a column address, latched in by a $\overline{CAS}$ (column address strobe) signal.
12. Read only memory (ROM) is used where it is important to retain data after power is removed.
13. Mask-programmed ROM is programmed at the time of manufacture. Programming is done by making a custom overlay of connections onto a standard cell array. Data cannot be changed. This is suitable for mature designs in high-volume production.
14. Erasable programmable read only memory (EPROM) can be programmed by the user and erased by exposure to ultraviolet light of a specified frequency and intensity. An EPROM must be removed from its circuit for erasing and reprogramming.
15. Electrically erasable read only memory (EEPROM or E^2PROM) can be programmed and erased in-circuit. It is nonvolatile, but unsuitable for use as system RAM due to

its long programming/erase times and finite number of program/erase cycles.

16. Flash memory is a type of EEPROM that is organized into sectors that are erased all at once. This is faster than other EEPROM, which must be erased byte-by-byte.
17. Flash memory is often configured with one sector as a boot block, where primary firmware is stored. A bottom boot block architecture has the boot block at the lowest chip address. A top boot block architecture has the boot block at the highest chip address.
18. Sequential memory must have its data accessed in sequence. Two major classes are first-in first-out (FIFO) and last-in first-out (LIFO). FIFO is also called a queue and LIFO is called a stack.
19. Dynamic RAM chips are often configured as memory modules, small circuit boards with multiple DRAMs. The modules usually have the same number of address locations as the individual chips on the module, but a wider data bus.
20. Memory systems can be configured to have the same data width as individual memory devices comprising the system, but with more addressable locations than any chip in the system. The additional addresses require additional system address lines, which are decoded to enable one chip at a time within the system.

GLOSSARY

Address A number, represented by the binary states of a group of inputs or outputs, uniquely defining the location of data stored in a memory device.

Address decoder A circuit enabling a particular memory device to be selected by the address bus of a larger memory system.

Address multiplexing A technique of addressing storage cells in a dynamic RAM which sequentially uses the same inputs for row address and column address of the cell.

Address Space A block of addresses in a memory system.

b Bit.

B Byte.

Bit-organized A memory is bit-organized if one address accesses one bit of data.

Boot block A sector in a flash memory reserved for primary firmware.

Bottom boot block A boot block sector in flash memory paced at the lowest address in the memory.

Bus A group of parallel conductors carrying related logic signals, such as multi-bit data or addresses.

Bus contention The condition that results when two or more devices try to send data to a bus at the same time. Bus contention can damage the output buffers of the devices involved.

Byte A group of 8 bits.

$\overline{\text{CAS}}$ Column address strobe. A signal used to latch the column address into the decoding circuitry of a dynamic RAM with multiplexed addressing.

Data Binary digits (0s and 1s) which contain some kind of information. In the context of memory, the digital contents of a memory device.

Dual in-line memory module (DIMM) A memory module with DRAMs and connector pins on both sides of the board.

Dynamic RAM A random access memory which cannot retain data for more than a few (e.g., 64) milliseconds without being "refreshed."

EEPROM (or E^2PROM) Electrically erasable programmable read only memory. A type of read only memory that can be field-programmed and selectively erased while still in a circuit.

EPROM Erasable programmable read only memory. A type of ROM that can be programmed ("burned") by the user and erased later, if necessary, by exposing the chip to ultraviolet radiation.

FAMOS FET Floating-gate avalanche. MOSFET. A MOSFET with a second, "floating" gate in which charge can be trapped to change the MOSFET's gate-source threshold voltage. A FAMOS transistor is the memory element in an EPROM cell.

FIFO First-in first-out. A sequential memory in which the stored data can only be read in the order in which it was written.

Firmware Software instructions permanently stored in ROM.

Flash memory A nonvolatile type of memory that can be programmed and erased in sectors, rather than byte-at-a-time.

Hardware The electronic circuit of a digital or computer system.

I/O Input/output.

K 1024 ($=2^{10}$) Analogous to the metric prefix "k" (kilo).

LIFO Last-In first-out. A sequential memory in which the last data written is the first data read.

M 1,048,576 ($=2^{20}$) Analogous to the metric prefix "M" (mega).

Mask-programmed ROM A type of read only memory (ROM) where the stored data are permanently encoded into the memory device during the manufacturing process.

Memory A device for storing digital data in such a way that it can be recalled for later use in a digital system.

Memory map A diagram showing the total address space of a memory system and the placement of various memory devices within that space.

Memory module A small circuit board containing several dynamic RAM chips.

Nibble Half a byte; 4 bits.

PROM Programmable read only memory. A type of ROM whose data need not be manufactured into the chip, but can be programmed by the user.

Queue A FIFO memory.

RAM cell The smallest storage unit of a RAM, capable of storing one bit.

Random access memory (RAM) A type of memory device where data at any address can be accessed in any order, that is, randomly. The term usually refers to random access read/write memory.

$\overline{\text{RAS}}$ Row address strobe. A signal used to latch the row address into the decoding circuitry of a dynamic RAM with multiplexed addressing.

Read Retrieve data from a memory device.

Read only memory (ROM) A type of memory where data is permanently stored and can only be read, not written.

Refresh cycle The process which periodically recharges the storage capacitors in a dynamic RAM.

Sector A segment of flash memory that forms the smallest amount that can be erased and reprogrammed at one time.

Sequential memory Memory in which the stored data cannot be read or written in random order, but must be addressed in a specific sequence.

Single in-line memory module (SIMM) A memory module with DRAMs and connector pins on one side of the board only.

Software Programming instructions required to make hardware perform specified tasks.

Stack A LIFO memory.

Static RAM A random access memory which can retain data indefinitely as long as electrical power is available to the chip.

Top boot block A boot block sector in a flash memory placed at the highest address in the memory.

Volatile A memory is volatile if its stored data is lost when electrical power is lost.

Word Data accessed at one addressable location.

Word length Number of bits in a word.

Word-organized A memory is word-organized if one address accesses one word of data.

Write Store data in a memory device.

PROBLEMS

Section 13.2 Basic Memory Concepts

13.1 How many address lines are necessary to make an 8 × 8 memory similar to the 4 × 8 memory in Figure 13.5? How many address lines are necessary to make a 16 × 8 memory?

13.2 Briefly explain the difference between RAM and ROM.

13.3 Calculate the number of address lines and data lines needed to access all stored data in each of the following sizes of memory:

a. 64K × 8

b. 128K × 16

c. 128K × 32

d. 256K × 16

Calculate the total bit capacity of each memory.

13.4 Explain the difference between the chip enable ($\overline{E}$) and the output enable ($\overline{G}$) control functions in a RAM.

13.5 Refer to Figure 13.9. Briefly explain the operation of the $\overline{W}$, $\overline{E}$, and $\overline{G}$ functions of the RAM shown.

Section 13.2 Random Access Read/Write Memory (RAM)

13.6 Draw the circuit for an NMOS static RAM cell. Label one output *BIT* and the other $\overline{BIT}$.

13.7 Refer to the NMOS static RAM cell drawn in Problem 13.6. Assume that *BIT* = 1. Describe the operation required to change *BIT* to 0.

13.8 Describe the main difference between a CMOS and an NMOS static RAM cell.

13.9 Explain how a particular RAM cell is selected from a group of many cells.

13.10 How many address lines are required to access all elements in a 1M × 1 dynamic RAM with address multiplexing?

13.11 What is the capacity of an address-multiplexed DRAM with one more address line than the DRAM referred to in Problem 13.10? With two more address lines?

13.12 How many address lines are required to access all elements in a 256M × 16 DRAM with address multiplexing?

Section 13.3 Read Only Memory (ROM)

13.13 Briefly list some of the differences between mask-programmed ROM, UV-erasable EPROM, EEPROM, and flash memory.

13.14 Briefly describe the programming and erasing process of a UV-EPROM.

13.15 Briefly explain the difference between flash memory and other EEPROM. What is the advantage of each configuration?

13.16 A flash memory has a capacity of 8 Mb, organized as 512K × 16-bit. List the address range for the 16 KB boot block sector of the memory if the device has a bottom boot block architecture and if it has a top boot block architecture.

13.17 Briefly state why EEPROM is not suitable for use as system RAM.

13.18 Briefly state why flash memory is unsuitable for use as system RAM.

Section 13.4 Sequential Memory

13.19 State one possible application for a FIFO and for a LIFO memory.

Section 13.5 Memory Modules

13.20 A SIMM has a capacity of 32M × 64. How many 32M × 8 DRAMs are required to make this SIMM? How many address lines does the SIMM require? How should the DRAMs be connected?

Section 13.6 Memory Systems

13.21 A microcontroller system with a 16-bit address bus is connected to a 4K × 8 RAM chip and an 8K × 8 RAM chip. The 8K address begins at 6000H. The 4K address block starts at 2000H.

Calculate the end address for each block and show address blocks for both memory chips on a 64K memory map.

13.22 Draw the memory system of Problem 13.21.

13.23 A microcontroller system with a 16-bit address bus has the following memory assignments:

Memory	Size	Start Address
RAM_0	16K	4000H
RAM_1	8K	8000H
RAM_2	8K	A000H

Show the blocks on a 64Kmemory map.

13.24 Draw the memory system described in Problem 13.23.

13.25 The memory map of a microcontroller system with a 16-bit address bus is shown in Figure 13.30. Make a table of start and end addresses for each of the blocks shown. Indicate the size of each block.

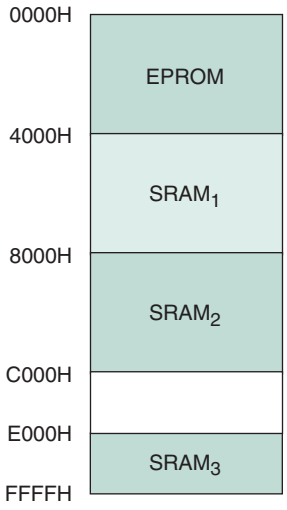


FIGURE 13.30
Problem 13.25

13.26 Sketch the memory system described in Problem 13.25.

13.27 How many 16M × 32 DIMMs are required to make a 256M × 32 memory system? Make a table showing the start and end addresses of each block.

ANSWERS TO SECTION REVIEW PROBLEMS

Section 13.2a

13.1 18 address lines, 16 data lines; capacity = 4Mb, same as Figure 3.12

Section 13.2b

13.2 **a.** 10 address, 1 data; **b.** 10 address, 4 data; **c.** 11 address, 1 data.

Section 13.3

13.3 Bottom boot block: 00000H to 07FFFH; top boot block: F8000H to FFFFFH.

Section 13.4

13.4 A stack is a last-in first-out (LIFO) memory and a queue is a first-in first-out (FIFO) memory.

Section 13.5

13.5 Four DRAMs. 12 address lines. Address and control lines are in parallel with all DRAMs. Data I/O lines are separate.

Section 13.6

13.6 Eight blocks. Start addresses: 00000H, 20000H, 40000H, 60000H, 80000H, A0000H, C0000H, E0000H.