g 1 to the left of this fraction. So if F is stored as 101000 . . . 00, then the value of F is 1.10100 . . . 00, or $1\frac{5}{8}$ decimal. The value of the number stored is then

$$V = (-1)^S \times 2^{E-1023} \times F$$

Here are examples:

FLOATING-POINT FORMAT (HEXADECIMAL)	$(-1)^S \times 2^{E-1023} \times 1.F$	DECIMAL VALUE
3FD0000 . . . 00	$1 \times 2^2 \times 1.0$	0.25
C03E000 . . . 00	$-1 \times 2^4 \times 1.875$	−30
401C000 . . . 00	$1 \times 2^2 \times 1.75$	7

Since a 1 is assumed to be "invisibly" stored with each number, the representation for 0 must be special. The standard 0 is represented by all 0s in the E and F sections (there is a $+0$ and a -0). Further, infinity is represented by all 1s in the E section and all 0s in the F section. There is, therefore, also a plus infinity and a minus infinity.

When numbers are so small that they cannot be represented in normalized form because E would need to be less than 1, then the F part is handled in denormalized form and the 1 is not added when the numbers are evaluated.

The results of invalid operations are signaled by making E all 0s; S can be anything; and if F is nonzero, the 1s in F signal an illegal operation.

Other formats are less used than the two shown here. A good complete introduction to this standard can be found in Jerome T. Coonen's "An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic," *Computer*, January 1980.

PERFORMING ARITHMETIC OPERATIONS WITH FLOATING-POINT NUMBERS

5.24 A computer obviously requires additional circuitry to handle floating-point numbers automatically. Some machines come equipped with floating-point instructions. (For computers such as DEC PDP-11/45 and others, floating-point circuitry can be purchased and added to enable them to perform floating-point operations.)

To handle the floating-point numbers, the machine must be capable of extensive shifting and comparing operations. The rules for multiplying and dividing are

$$(a \times r^p) \times (b \times r^q) = ab \times r^{p+q}$$

$$(a \times r^p) \div (b \times r^q) = \frac{a}{b} \times r^{p-q}$$

The computer must be able to add or subtract the exponent sections of the floating-point numbers and to perform the multiplication or division operations on the mantissa sections of the numbers. In addition, precision is generally maintained by

shifting the numbers stored until significant digits are in the leftmost sections of the word. With each shift the exponent must be changed. If the machine is shifting the mantissa section left, for each left shift the exponent must be decreased.

For instance, in a BCD computer, consider the word

0	10	0064
sign	exponent	mantissa

To attain precision, the computer shifts the mantissa section left until the 6 is in the most significant position. Since two shifts are required, the exponent must be decreased by 2, and the resulting word is 0.08 6400. If all numbers to be used are scaled in this manner, maximum precision may be maintained throughout the calculations.

For addition and subtraction the exponent values must agree. For instance, to add 0.24×10^5 to 0.25×10^6, we must scale the numbers so that the exponents agree. Thus

$$0.024 \times 10^6 + 0.25 \times 10^6 = 0.274 \times 10^6$$

The machine must also follow this procedure. The numbers are scaled as described, so that the most significant digit of the computer mantissa section of each word contains the most significant digit of the number stored. Then the larger of the two exponents for the operands is selected, and the other number's mantissa is shifted and its exponent adjusted until the exponents for both numbers agree. The numbers may then be added or subtracted according to these rules:

$$a \times r^p + b \times r^p = (a + b) \times r^p$$
$$a \times r^p - b \times r^p = (a - b) \times r^p$$

SUMMARY

5.25 The binary full-adder forms the backbone of most arithmetic-logic units. Full-adders can be used to both add and subtract 2s or 1s complement numbers, and the layouts for these operations were shown.

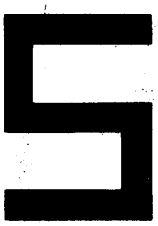
Binary full-adders are often packaged in IC packages with several per package. Several examples were shown, including one which can also perform a number of logical operations.

Binary multiplication and binary division are generally performed by using three registers and a sequence of additions and/or subtractions and shifts. The control circuitry sequences these operations. Parallel multiplication can be performed by using gate networks consisting of full-adders correctly arranged. This technique makes very high-speed multipliers.

Binary-coded-decimal addition and subtraction units use special BCD adders and complementers. Examples were shown, and their use in serial-parallel systems was explained.

Floating-point number systems make it possible to represent very large and very small numbers using fewer bits than would be required for straight binary.

Often calculations are not exact since numbers are approximated in many cases, but floating-point number systems are almost always used for large scientific computations and by higher-level languages. Several examples of floating-point number systems were explained, including the new ANSI standard.



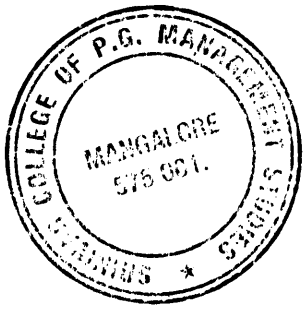
QUESTIONS

5.1 Draw a block diagram of circuitry for two registers X and Y of three flip-flops each, so that Y can be transferred into X, or the 1s complement of Y can be transferred into X.

5.2 Two parallel binary registers, designated register X and register Y, both consist of three flip-flops. Draw a block diagram of the registers and the necessary logic circuitry so that (*a*) register X can be cleared, or 1s complemented, and (*b*) the 1s complement of the contents of register Y can be transferred into register X.

5.3 If a binary computer handles numbers in the sign-plus-magnitude integer system and numbers are 5 bits (sign plus 4 bits) each, how would the following decimal numbers be represented? For example, $+5 = \underline{0}0101$.

(*a*) $+6$ (*b*) $+10$ (*c*) -12 (*d*) -16



5.4 If a binary computer represents numbers in a sign-plus-magnitude form with 5 bits per number, how would the following decimal numbers be represented.

(*a*) $+8$ (*b*) $+11$ (*c*) -7
(*d*) -4 (*e*) -15 (*f*) -12

5.5 If a binary machine handles negative numbers in the true magnitude form, how would -4 be stored in a register with a sign bit and 4 bits representing magnitude? If the same machine stored numbers in the 1s complement system, how would -4 be stored?

5.6 If a register contains five flip-flops as in Fig. 5.1(*b*) and the register contains $X_0 = 1, X_2 = 1, X_3 = 0, X_4 = 0, X_5 = 1$, give the decimal value of the number in the register if the 1s complement number system is used. What is the value if 2s complement is used?

5.7 The inputs to the full adder in Fig. 5.3 are as follows: $X = 1$, $Y = 1$, and $C = 1$. What will the output on the S and C lines represent?

5.8 If we use the $\oplus$ symbol to mean exclusive OR and define it as $X \oplus Y = \bar{X}Y + X\bar{Y}$, then the output S from a full-adder can be written as $S = X \oplus Y \oplus C_i$. Show why this is the case.

5.9 If we load the binary number $\underline{1}0011$ into the flip-flops in Fig. 5.1(*b*), that is, if $X_4 = 1, X_3 = 0, X_2 = 0, X_1 = 1$, and $X_0 = 1$, what will the value of the register be in the 1s complement number system? Give the answer in decimal. What will the value of this number be if the 2s complement number system is used?

5.10 If register X contains $\underline{0}0111$ and register Y contains $\underline{1}1011$, what do the two numbers represent in decimal if 1s complement is used? If 2s complement is used?

Add the two numbers in 1s complement, then in 2s complement, and give the results in decimal.

5.11 Register X contains $\underline{0}1100$, and register Y contains $\underline{0}1101$ (where the underscored 0s designate that the number stored is positive). If the two registers are added, what will the result be?

5.12 If 1s complement is used, for which of the following expressions will an overflow or end-around carry be generated? Why? Assume 5-bit registers, including a sign bit.

(a) $+5+(-7)$ (b) $+5+(-4)$
(c) $+12+(-13)$ (d) $+12+3$

5.13 Add a stage to the load-and-shift register in Fig. 5.17. Copy only D in your drawing, omitting A, B, and C.

5.14 A binary register consists of five binary storage devices; one stores the sign bit, and the other four store the magnitude bits. If the number stored is $\underline{0}0110$ and this number is then shifted right one binary place, what will be the result? Assume a 0 goes into the sign bit.

5.15 Design a half-adder, using only NOR gates.

5.16 Design a half-adder, using only NAND gates.

5.17 A binary half-subtracter has two inputs x and y and two outputs, which are the "difference" value $x - y$ and a "borrow" output that is 1 if the value of $x - y$ is negative ($x - y$ is then given the value 1). Draw a block diagram for a half-subtracter, using NAND gates and assuming x, $\bar{x}$, y, and $\bar{y}$ are all available as inputs.

5.18 Design a half-subtracter, using only NOR gates.

5.19 If a borrow input is added to the half-subtracter in Questions 5.17 and 5.18, a full-subtracter is formed. Design a full-subtracter, using only NAND gates.

5.20 Design a full-subtracter, using only NOR gates.

5.21 Design a full-adder, using only NAND gates.

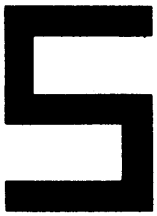
5.22 Can an overflow occur during multiplication in a binary machine with numbers stored in fixed-point sign-plus-magnitude form? Assume a double-length product.

5.23 Explain how the 2s complement of the subtrahend is formed when we subtract, using the configuration in Fig. 5.6. How is the 1 added in to form this complement?

5.24 Show how the configuration in Fig. 5.6 adds and subtracts by adding and subtracting $+21$ and $+3$ in binary, showing what each output will be.

5.25 Using the configuration in Fig. 5.6, add and subtract $+6$ and -4, showing what each output will be and checking the correctness of the sum and difference.

5.26 Draw a multiplexer using only NAND gates which selects from four inputs I_0 to I_3, using two select inputs S_0 and S_1.



5.27 Design a multiplexer for four inputs, using a two-level NOR gate combinational network. The inputs are to be X_0, X_1, X_2, and X_3. The output is to be called W. The X_0, X_1, X_2, and X_3 are "selected" by S_0 and S_1. If S_0 and $S_1 = 0$, then W should equal X_0. If S_1 and $S_0 = 01$, then W should equal X_1. If S_0 and $S_1 = 10$, then W should equal X_2. If S_0 and $S_1 = 11$, then W should equal X_3.

5.28 Add a control line NAND and circuitry to Fig. 5.20 so that the NAND of ACC and B can be transferred into AOC.

5.29 If a register containing 0110011 is logically added to a register containing 0101010, what will the result be? What will be the result if the registers are logically multiplied? If the registers are exclusive ORed?

5.30 Find a sequence of logical operations which will cause the ACC to have value 0 regardless of how ACC and B start in Fig. 5.20.

5.31 Referring to Fig. 5.20, show that a logical multiplication followed by a logical addition will transfer the contents of B to ACC.

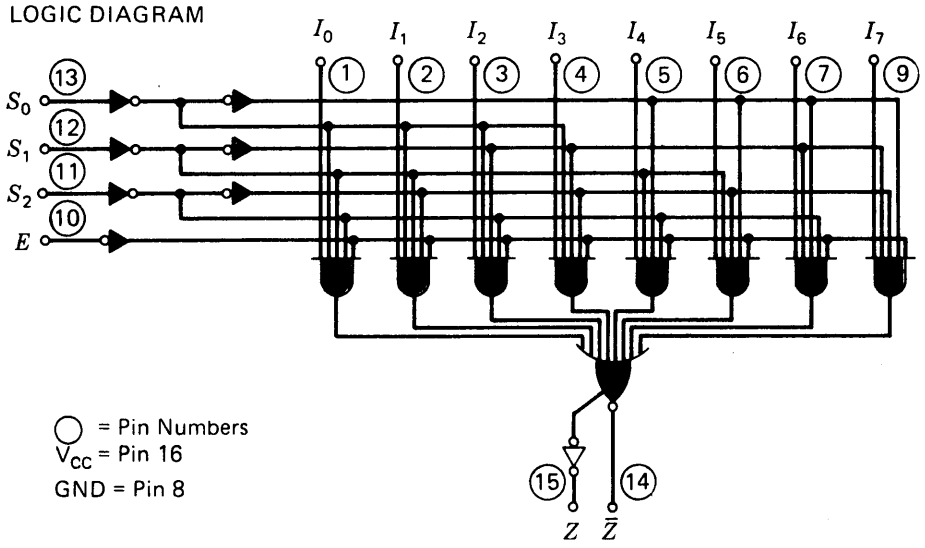
5.32 Add a control line NOR and circuitry to Fig. 5.20 so that the NOR of ACC and B can be transferred into ACC.

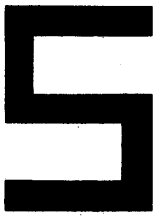
5.33 Demonstrate by means of a table of combinations that two half-adders plus an OR gate do make a full-adder, as shown in Fig. 5.4.

5.34 Add gates and a flip-flop X_{m+2} so that we can shift left and right into X_{m+1} in Fig. 5.16.

5.35 Add gates to the following drawing so that if the control input signal C is made a 1, the value $\overline{X}\overline{Y}$ will appear on the output line. If A is a 1, then $X + Y$ appears on the output; if B is a 1, then $X \cdot Y$ appears on the output; if A, B, and C are 0s, then the output is to be a 0; and only one of A, B, or C can be a 1 at a given time.

FIGURE 5.35





5.36 In what case will subtracting a negative number from 0 cause an overflow in the 2s complement system?

5.37 Design a *parallel multiplier* gate network with two inputs A_1A_0 and $B_2B_1B_0$, where A_1A_0 are two binary digits forming a binary number (with decimal values for 0 to 3) and $B_2B_1B_0$ is a 3-bit binary number (with values 0 to 7). Use only AND gates and OR gates, and see that no signal passes through more than four levels of gates. (Assume input complements are available.)

5.38 Write the boolean algebra expressions for the four least significant bits in a parallel multiplier which multiplies two 4-bit binary numbers.

5.39 Design a gating network for a module in an ALU that will add two 2-bit binary inputs A_j, A_{j-1} and B_j, B_{j-1} and an input carry-in C_{j-1}. The network is to generate the two sum digits and a carry-out C_{j+2}. Use only AND or OR gates and inverters, but assume that both complemented and uncomplemented inputs are available. The output carry C_{j+2} should have a delay of no more than three-gate delays (that is, a change in an input must pass through no more than three gates in any path to an output).

5.40 (*a*) Give the IBM floating-point representation for decimal 27.25 and -27.25.

(*b*) Give the IEEE standard format representation for $+12$ and -12.

5.41 (*a*) Give the IBM floating-point representation for $+55.4$ and -53.4.

(*b*) Give the IEEE standard single-precision floating-point representation for $+29$ and -29.

(*c*) Compare the ranges, accuracy, and other system design considerations for the above two computer floating-point number systems.

5.42 If numbers are represented in a 2s complement, 7-bit magnitude plus 1 sign-bit integer system, and we ignore overflow (that is, any result will be stored, even if it requires more magnitude bits), then the largest positive integer which can result from the addition of two numbers is _____, and the largest positive integer which can result from the subtraction of one number from another is _____. The least negative numbers which can result from an addition and subtraction are _____ and _____. As a result, any sum or difference can be stored in _____ bits.

5.43 Write all 4-bit 2s complement numbers (that is, sign plus 3-bit numbers) and their decimal values. Show that there is one more negative number than positive number. Consider 0 to be neither negative nor positive.

5.44 When addition is performed in a binary machine using the 2s complement number system to represent negative numbers, an overflow may occur in a register only when two positive or two negative numbers are added. Show that the addition of a positive number and a negative number cannot result in an overflow condition.

5.45 Show that there are as many negative as positive numbers in a 1s complement system.

5.46 The boolean algebra expressions on the output lines from the gates in Fig. 5.8 are not filled in. Develop the boolean algebra expressions for the $S1$, $S2$, and $C2$ outputs from the network.

5.47 Which of the following number systems has two 0s?
(*a*) Sign plus magnitude (*b*) 1s complement
(*c*) 2s complement

5.48 When the 2s complement number system is used and addition is performed, let us designate the carry-out of the full-adder connected into the full-adder for the sign digits as C_3 (refer to Fig. 5.6). The rule for overflow is that two numbers cause an overflow when they are added either if both numbers are positive and C_3 equals a 1 or if both numbers are negative and C_3 does not equal a 1. Therefore, by examining the sign digits of the two numbers being added and the carry-out of the full-adder which adds the two most significant digits of the magnitude of the numbers, we can form a logic network whose output will be a 1 when an overflow condition arises and a 0 if the addition is legitimate. Let X_4 store the sign digit of the addend, let Y_4 store the sign digit of the augend, and let C_3 again be the carry-out of the full-adders connected to the X_3 and Y_3 flip-flops, as in Fig. 5.6. Show that the logic equation for an overflow condition is $X_4Y_4\overline{C}_3 + \overline{X}_4\overline{Y}_4C_3 =$ overflow.

5.49 Modify Fig. 5.16 so that the complement of X can be shifted into X_m, that is, so that SHIFT LEFT causes $\overline{X}_{m-1}$ to go into X_m.

5.50 Show that when we add 7 to 9 in the BCD system using the series-parallel BCD adder in Fig. 5.10, the answer will be correct. Do this by tracing the outputs of the circuit, filling in the binary value for each X and Y shown in the figure, and by showing the values of Z_8, Z_4, Z_2 and Z_1.

5.51 Show how the BCD adder in Fig. 5.11 adds +6 to +5 by calculating each output from the gates and then the final outputs.

5.52 Check how the gates in Fig. 5.12 form a 9s complement by trying 5 and 3 in BCD at the inputs.

5.53 Explain the operation of Fig. 5.15 by explaining how 234 can be added to or subtracted from 523 in this configuration.

5.54 What is the function of the D flip-flop in Fig. 5.15?

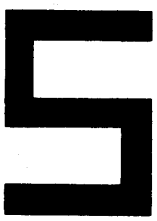
5.55 Explain how to load the input values on W, X, Y, and Z into the flip-flops in Fig. 5.17.

5.56 Explain how to cause the flip-flops in Fig. 5.17 to shift right three times. Suppose we (1) make the *reset* input a 0 then a 1, (2) hold J and K at 1 and shift at 1, and (3) apply three clock pulses to the CLOCK line. Draw the output waveforms for A, B, C, and D for this sequence of inputs.

5.57 What is the result if we multiply $\underline{0}1101$ times $\underline{0}0011$ in our generalized machine in Fig. 5.18? Give the values in each X and B flip-flop.

5.58 What is the binary number that represents -3 in the 2s complement fractional number system if we represent the number by using a sign digit plus four magnitude digits?

5.59 Using the 8,4,2,1 BCD system with a single digit for the sign digit, write



the following numbers, using a sign-plus-magnitude number system:

(*a*) +0014 (*b*) +0291
(*c*) −2346 (*d*) −0364

5.60 Using the 8,4,2,1 BCD system, write in binary form the following decimal numbers. Use a single digit for the sign digit, and express the numbers as magnitude plus sign.

(*a*) +0043 (*b*) −0222 (*c*) +1234 (*d*) −1297

5.61 Using the 8,4,2,1 code as in Question 5.59, give the same numbers but use 9s complement for the negative numbers.

5.62 Express each of the numbers in Question 5.60, using the 9s complement and the 8,4,2,1 BCD system. For example, −1024 = 11000 1001 0111 0101.

5.63 Write the binary forms of the numbers in Question 5.60, using 8,4,2,1 but 10s complements for negative numbers.

5.64 Write the decimal numbers in Question 5.60, using the 10s complement number system and again the 8,4,2,1 BCD system. For example, −1420 = 11000 0101 1000 0000.

5.65 If we add two 20-digit binary numbers, using the full-adders shown in Fig. 5.8, and if 3 ns is required for a signal to pass through a gate, what is the maximum time it will require a CARRY signal to propagate from the lowest-order bits to the highest-order bits, assuming a full-adder which is parallel as in Fig. 5.6?

5.66 Explain how you would add gates and inputs to Fig. 5.17 so that the flip-flop register could be shifted left as well as right.

5.67 If we multiply 6 × 11 in the registers in Fig. 5.18, show the placement of binary digits at the start and end of the multiplication.

5.68 Show how 7 × 9 and 5 × 5 would be multiplied in the registers in Fig. 5.18.

5.69 Draw a flowchart (as in Fig. 5.19) for the binary multiplication procedure described.

5.70 If we divide 23 by 6 in the registers in Fig. 5.18, show the beginning positioning of the numbers (in binary) and the result at the end. (Show where the quotient and the remainder are placed.)

5.71 Explain how to divide 14 by 4 by using the registers in Fig. 5.18 and showing how the quotient and the remainder are placed after the division.

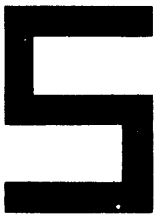
5.72 Go through the division of 14 by 3, using the technique shown in the text.

5.73 Using the algorithm shown in Fig. 5.19, show how to divide 11 by 4.

5.74 Show how to represent +6 in the 12-bit floating-point word in Fig. 5.31.

5.75 Show how to represent −14 in the 12-bit floating-point word in Fig. 5.31.

5.76 (*a*) Give the IBM floating-point representation for 57.5 and 54.5.
(*b*) Give the IEEE standard format floating-point representation for +25.0 and −25.0.



(*c*) Compare the ranges, accuracy, and other system design considerations for the above two computer floating-point number systems.

5.77 Show two decimal numbers which, when converted to the IBM 370 floating-point number system, will have .0011 in bits 8, 9, 10, and 11.

5.78 Can you give any reasons that might be behind the decision of the systems architects at IBM to use hexadecimal as the base for the IBM series floating-point number system instead of conventional base 2? That is, in system 370 a floating-point fullword of 32 bits has as characteristic a 7-bit integer with value C and as fraction a binary signed-magnitude fraction 25-bit number with value F. The value of the number represented is then $(16^{C-64})F$. Why 16^{C-64} rather than 2^{C-64}? Give system design considerations.

5.79 Give the value of a positive nonzero *integer* less than 16^{62} which cannot be represented in the IBM 370 floating-point number system (using a single 32-bit word). Do not be afraid to use an expression such as $2^{16} + 3$ for your answer, but explain why you think your answer is correct.

5.80 A binary computer with a basic 16-bit code uses an integer 2s complement number system. The arithmetic element contains an accumulator and *MQ* register, each containing 16 flip-flops (or bits). When a multiplication is performed, the 16-bit word taken from the memory is multiplied by the 16-bit word in the accumulator, and the product is stored in the combined ACC-*MQ* with the least significant bit in the rightmost bit of the *MQ*. The sign bit of the *MQ* is set the same as the sign of the ACC and is not used to store a magnitude bit. In what bit of the ACC does the most significant bit of a product appear for numbers of maximum magnitude?

5.81 For the 74S181 chip in Fig. 5.21, write the boolean algebra expression for $\overline{F}_0$ in terms of A_0, B_0, and C_n if M, S_3, S_1, S_0 are 1s and S_2 is a 0.

5.82 For the 74S181 chip in Fig. 5.21, how would you set M and S_0, S_1, S_2, and S_3 to subtract B from A?

5.83 For the 74S181 chip in Fig. 5.21, if we set $M = 1$, $S_3 = S_1 = S_0 = 1$, and $S_2 = 0$, the chip will add A to B. Write the boolean algebra expression for F_1 in terms of the A, B, C_{n+4} inputs.

5.84 How would you set the M, S_0, S_1, S_2, S_3 inputs to the 74S181 chip in Fig. 5.21 to form the AND of A and B?

5.85 How would you set the M, S_0, S_1, S_2, S_3 inputs to perform an OR of the A and B inputs for the chip in Fig. 5.21?

5.86 Explain how the carry output C_n is formed for the chip in Fig. 5.21.

5.87 The statement was made that the maximum delay through the 74S181 chip in Fig. 5.21 is 11 ns. This means, for instance, that if all the inputs are held in the same state except for $\overline{A}_0$, then from the time $\overline{A}_0$ is changed, the maximum time for C_{n+4} to change will be 11 ns. Find the maximum number of gates through which this delay must propagate. Then determine typical single-gate delays.

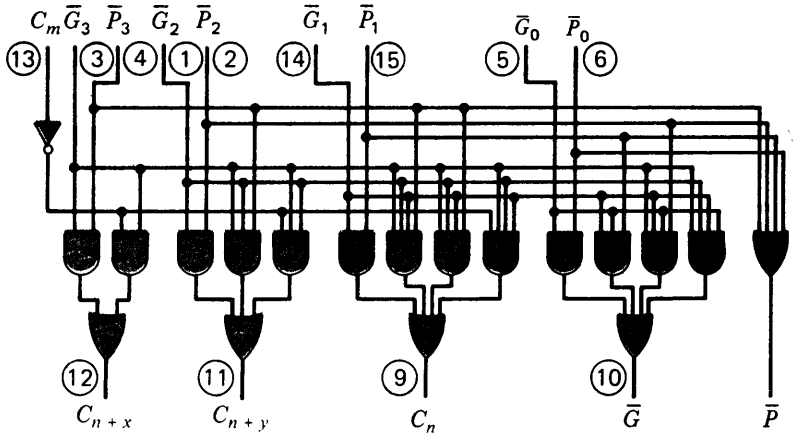
5.88 The inputs to the 74S181 chip in Fig. 5.21 are each complemented, as are

the outputs. Suppose we connect the A and B inputs to the uncomplemented outputs of the A and B flip-flops, also changing the diagram by removing the bars over the A and B inputs and the F outputs. Now if M is a 1, the functions yielded by each S_0, S_1, S_2, and S_3 state (input combination) will be different from those shown in Fig. 5.21. For instance, $M = S_3 = S_2 = S_1 = 1$ and $S_0 = 0$ will cause the circuit to form $A + B$. Give three more different output functions and their S_0, S_1, S_2, and S_3 values.

5.89 Repeat the setup with Question 5.88 and give three more functions.

5.90 The following is the block diagram for a 74S182 chip in the TTL series. This works with four 74S181 chips as shown in Fig. 5.21. The $\overline{G}_0$, $\overline{P}_0$, $\overline{G}_1$, $\overline{P}_1$, . . . inputs here are connected to the $\overline{G}$ and $\overline{P}$ outputs for the four 74S181 chips to be used. The 74S182 chip forms a carry-look-ahead generator for all four chips, forming high-speed carries for inputs to these chips. A complete 16-bit addition can be performed in 22 ns. Explain how this works.

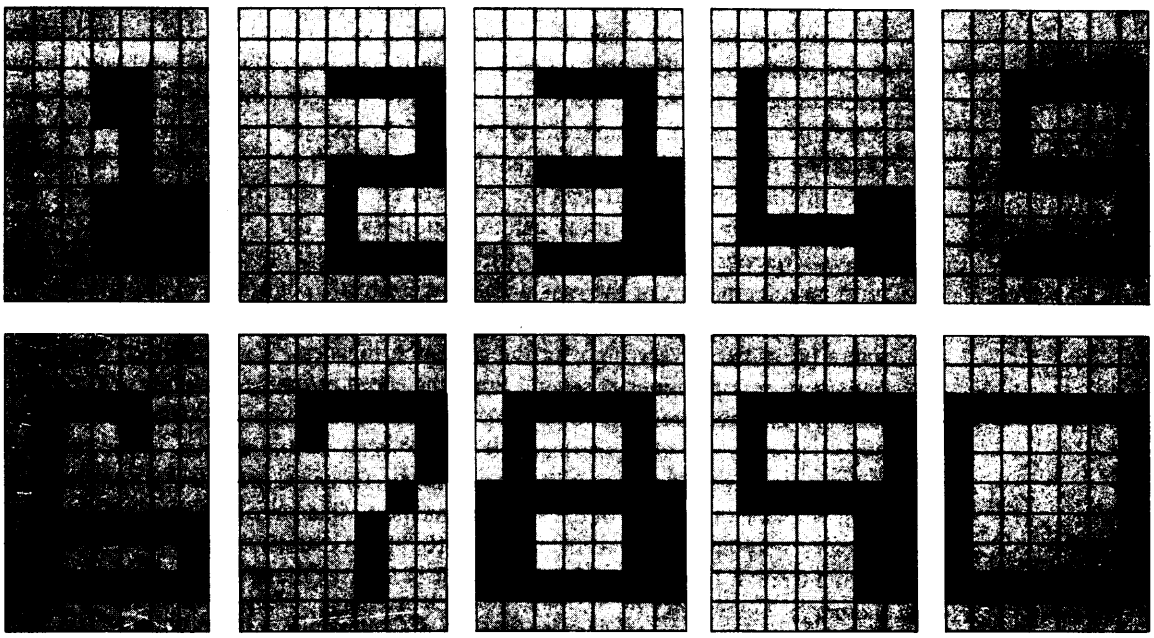
FIGURE 5.90



5.91 For the carry generator in Question 5.90 and four 74S181 chips as in Fig. 5.21, find the longest carry propagation path and determine how many gates are in it.

5.92 Develop the boolean algebra expression for the carry output in Fig. 5.29.

5.93 Convert the boolean algebra expression for the carry-out in a 3-bit adder to two-level AND-to-OR gate form.



THE MEMORY ELEMENT

The memory of a computer is not actually concentrated in one place; storage devices are scattered throughout the machine. For instance, the *operation registers* are flip-flop registers which are used in the arithmetic and control units of the computer. Arithmetic operations including additions, multiplications, shifts, etc., are all performed in these registers of the machine. The actual processing of information is performed in, and at the direction of, these registers.

Looking outward, the next category of storage device encountered is called the *high-speed memory, inner memory,* or *main memory*. This section of the computer's memory consists of a set of storage registers, each of which is identified with an address that enables the control unit either to write into or read from a particular register.

It is desirable that the operating speed of this section of the computer's memory be as fast as possible, for most of the transfers of data to and from the information processing section of the machine will be via the main memory. For this reason, storage devices with very fast access times are generally chosen for the main memory; unfortunately the presently available devices which are fast enough to perform this function satisfactorily do not possess the storage capacity that is sometimes required. As a result, additional memory, which is called the *auxiliary memory,* or *secondary memory,* is added to most computers. This section of the computer's memory is characterized by low cost per digit stored, but it generally has an operating speed far slower than that of either the operation registers or the main memory. This section of the memory is sometimes designated the *backup memory,* for its function is to handle quantities of data in excess of those that may be stored in the inner memory.

6

THE MEMORY ELEMENT

The final and outermost storage devices are used to introduce information to the computer from the "outside world" and to store results from the computer to the computer user. The storage media in this case generally consist of such input media as punched cards or perforated paper tape, and the outputs from the machine generally consist of printed characters. Again, the cost per bit is low, but the operating speeds of the tape and card readers, printers, etc., are liable to be on the order of 1000 times slower than the speeds of the operation registers. These devices are described in Chap. 8 under input-output devices. This chapter is limited to the *internal storage* of the machine, which is defined as those storage devices that form an integral part of the machine and are directly controlled by the machine.

Each division of memory has certain characteristics. For instance, the premium on speed is very high for the operation registers. These registers generally must perform operations at several times the speed of the main memory. The main memory also requires high operating speeds, but because it is desirable to store larger quantities of data (perhaps 10^5 to 10^9 bits) in this section, a compromise between cost and speed generally must be made. Often the same sort of compromise must be made in the case of the auxiliary memory. In a large machine, the auxiliary memory may have to store from 10^8 to 10^{14} binary digits, and in these instances it might prove too expensive to use devices such as those employed in the main memory.

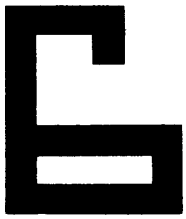
An important point to note when operating speed is considered is that, before a word can be read, it is necessary to locate it. The time required to locate and read a word from memory is called the *access time*. The procedures for locating information may be divided into two classes, random access and sequential access. A *random-access* storage device is one in which any location in the device may be selected at random, access to the information stored is direct, and approximately equal access time is required for each location. A flip-flop register is an example of a random-access storage device, as are the IC memories, which will be described. A *sequential-access* device is one in which the arrival at the location desired may be preceded by sequencing through other locations, so that access time varies according to location.[1] For instance, if we try to read a word stored on a reel of magnetic tape and the piece of tape on which the word is stored is near the center of the reel, it is necessary to sequence through all the intervening tape before the word can be read.

Another way to subdivide storage devices is according to whether they are static or dynamic. A *static* storage device is one in which the information does not change position; flip-flop registers and even punched cards or tape are examples of static storage devices. *Dynamic* storage devices are devices in which the information stored is continually changing position. Circulating registers utilizing charge-coupled device (CCD) delay lines are examples of dynamic storage devices.

This chapter will concentrate on the three most frequently used devices for storing digital information in the internal memory sections of computers: IC memo-

[1]Sequential-access devices are further separated into *direct-access storage devices* (DASD) and *serial-access devices*. Direct-access storage devices have addresses, but the access time to reach the data at a given address may vary. For instance, the time to locate on a movable-head disk (to be explained) depends on the head position and disk position when the address is given. Serial-access devices are truly serial in their access properties; magnetic tape is the classic example.

ries, which are high speed and of moderate cost; magnetic disk memories, which are direct-access storage devices generally used for auxiliary storage; and magnetic-tape memories, which are used almost exclusively as an auxiliary, or backup, storage but which are capable of storing large quantities of information at low cost. Following the sections on disk and magnetic-tape devices, the techniques used to record digital information on a magnetic surface are described.



OBJECTIVES

1 Memory cells are distributed throughout a computer. The faster memory devices are in the central processing unit which uses flip-flops. The main memory then uses IC circuits (or cores), but in arrays which are slower than the very high-speed flip-flops used in the CPU. The backup, or secondary, memory uses magnetic devices such as disks or tape. All these are discussed from a general organizational viewpoint.

2 IC memories are now used in the main high-speed memory in most computers. The selection of specific cells from an array of memory cells is an important factor in memory design, and this subject is treated first. Then, how IC memory chips are organized into a memory is described. Finally, the types of IC memories now in active use are discussed.

3 Disk and tape memories are the most used large memories, and the organization and construction of these are explained along with some of their operating and cost characteristics. Bubble memories and CCD memories are also discussed.

RANDOM-ACCESS MEMORIES

6.1 The main memory of a computer is organized in a way which is particularly desirable. Figure 6.1 shows that a high-speed main memory in a computer is organized into words of fixed lengths. As the figure indicates, a given memory is

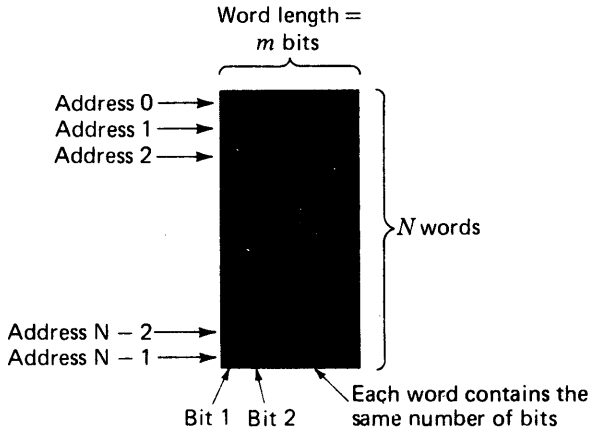
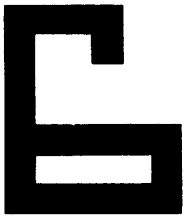


FIGURE 6.1

Words in high-speed memory.



THE MEMORY ELEMENT

divided into N words, where N generally is some power of 2, and each word is assigned an *address*, or *location*, in the memory. Each word has the same number of bits, called the *word length*. And if we read, for instance, the word at location 72, we receive a word from the memory with this word length.

The addresses, or address numbers, in the memory run consecutively, starting with the address 0 and running up to the largest address. Thus at address 0 we find a word, at address 1 a second word, at address 2 a third word, and so on up to the final word at the largest address.

Generally, the computer can read a word from or write a word into each location in the memory. For a memory with an 8-bit word, if we write the word 01001011 into memory address 17 and later read from this same address, we shall read the word 01001011. If we again read from this address at a later time (and have not written another word in), then the word 01001011 will be read again. This means the memory is a *nondestructive read*, in that reading does not destroy or change a stored word.

It is important to understand the difference between the *contents* of a memory address and the address itself. A memory is like a large cabinet containing as many drawers as there are addresses in memory. In each drawer is a word, and the address of each word is written on the outside of the drawer. If we write or store a word at address 17, it is like placing the word in the drawer labeled 17. Later, reading from address 17 is like looking in that drawer to see its contents. We do not remove the word at an address when we read. We change the contents at an address only when we store or write a new word.

From an exterior viewpoint, a high-speed main memory looks very much like a "black box" with a number of locations or addresses into which data can be stored or from which data can be read. Each address, or location, contains a fixed number of binary bits, the number being called the *word length* for the memory. A memory with 4096 locations, each with a different address and with each location storing 16 bits, is called a *4096-word 16-bit memory*, or, in the vernacular of the computer trade, a *4K 16-bit memory*. (Since memories generally come with a number of words equal to 2^n for some n, if a memory has $2^{14} = 16{,}384$ words, computer literature and jargon would refer to it as a 16K memory, because it is always understood that the full 2^n words actually occur in the memory. Thus, a 2^{15}-word 16-bit memory is called a 32K 16-bit memory.)

Memories can be read from (that is, data can be taken out) or written into (data can be entered into the memory). Memories which can be both read from and written into are called *read-write memories*. Some memories have programs or data permanently stored and are called *read-only memories*.

A block diagram of a read-write memory is shown in Fig. 6.2. The computer places the address of the location into which the data are to be read into the *memory address register*. This register consists of n binary devices (generally flip-flops), where 2^n is the number of words that can be stored in the memory. The data to be written into the memory are placed in the *memory buffer register*, which has as many binary storage devices as there are bits in each memory word. The memory is told to write by means of a 1 signal on the WRITE line. Then the memory will store the contents of the memory buffer register in the location specified by the memory address register.

Words are read by placing the address of the location to be read from into

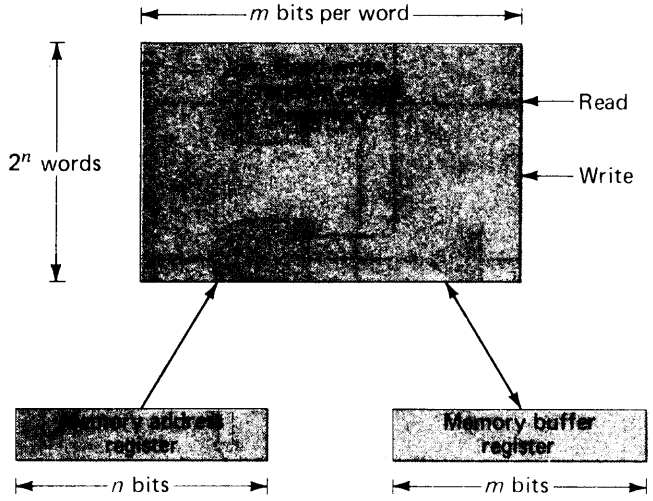


FIGURE 6.2

Read-write random-access memory.

the memory address register. Then a 1 signal is placed on the READ line, and the contents of that location are placed by the memory in the memory buffer register.

As can be seen, the computer communicates with the memory by means of the memory address register, the memory buffer register, and the READ and WRITE inputs. Memories are generally packaged in separate modules or packages. It is possible to buy a memory module of a specified size from a number of different manufacturers. For instance, a 64K 16-bit memory module can be purchased on a circuit board ready for use. Similarly, if a computer is purchased with a certain amount of main memory, generally more memory can be added later by purchasing additional modules and "plugging them in."

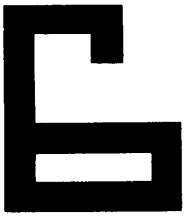
If it is possible to read from or write into any location "at once," that is, if there is no more delay in reaching one location than another, the memory is called a *random-access memory* (RAM). Computers almost invariably use random-access read-write memories for their high-speed main memory and then use backup or slower-speed memories to hold auxiliary data.

LINEAR-SELECT MEMORY ORGANIZATION

6.2 The most used random-access memories are IC memories. To present the basic principles, an idealized IC memory is shown, followed by details of several actual commercial memories.

In any memory there must be a basic memory cell. Figure 6.3 shows a basic memory cell consisting of an *RS* flip-flop with associated control circuitry. To use this cell in a memory, however, a technique for selecting those cells addressed by the memory address register must be used, as must a method to control whether the selected cells are written into or read from.

Figure 6.4 shows the basic memory organization for a *linear-select* IC memory. This is a four-address memory with 3 bits per word. The memory address register (MAR) selects the memory cells (flip-flops) to be read from or written into through



THE MEMORY ELEMENT

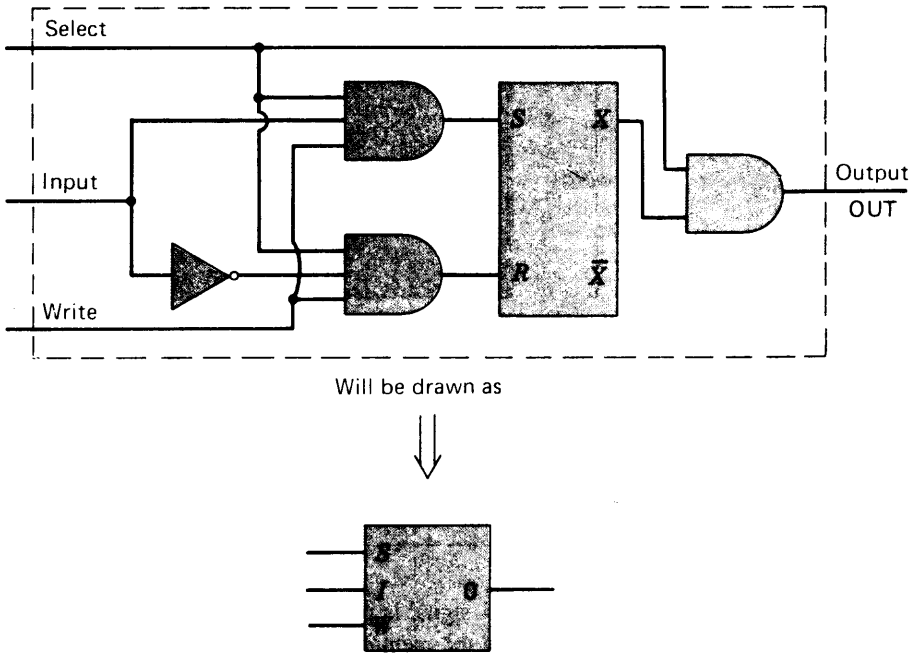


FIGURE 6.3

Basic memory cell.

a *decoder* which selects three flip-flops for each address that can be in the memory address register.

Figure 6.5(*a*) shows the decoder in expanded form. It has an input from each flip-flop (bit) to be decoded. If there are 2 input bits as in Fig. 6.5(*a*), then there will be four output lines, one for each state (value) the input register can take. For instance, if the MAR contains 0 in both flip-flops, then the upper line of the decoder will be a 1 and the remaining three lines a 0. Similarly, if both memory cells contain a 1, the lowest output line will be a 1 and the remaining three lines a 0. Similar reasoning will show that there will be a single output line with a 1 output for each possible input state, and the remaining lines will always be a 0.

Figure 6.5(*b*) shows a decoder for three inputs. The decoder has eight output lines. In general, for n input bits a decoder will have 2^n output lines.

The decoder in Fig. 6.5(*b*) operates in the same manner as that in Fig. 6.5(*a*). For each input state the decoder will select a particular output line, placing a 1 on the selected line and a 0 on the remaining lines.

Returning to Fig. 6.4, we now see that corresponding to each value that can be placed in the MAR, a particular output line from the decoder will be selected and carry a 1 value. The remaining output lines from the decoder will contain 0s, not selecting the AND gates at the inputs and outputs of the flip-flops for these rows. (Refer also to Fig. 6.3.)

The memory in Fig. 6.4 is organized as follows: There are four words, and each row of three memory cells comprises a word. At any given time the MAR selects a word in memory. If the READ line is a 1, the contents of the three cells in the selected word are read out on the O_1, O_2, and O_3 lines. If the WRITE line is a 1, the values on I_1, I_2, and I_3 are read into the memory.

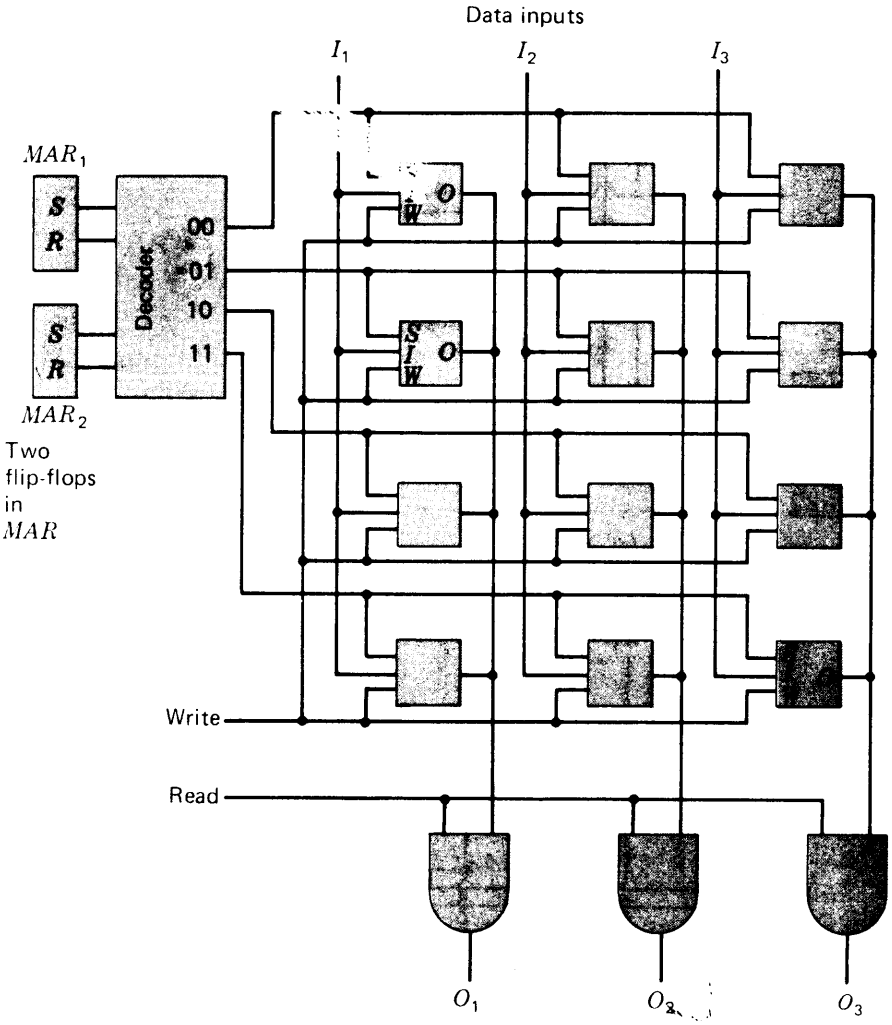


FIGURE 6.4

Linear-select IC memory.

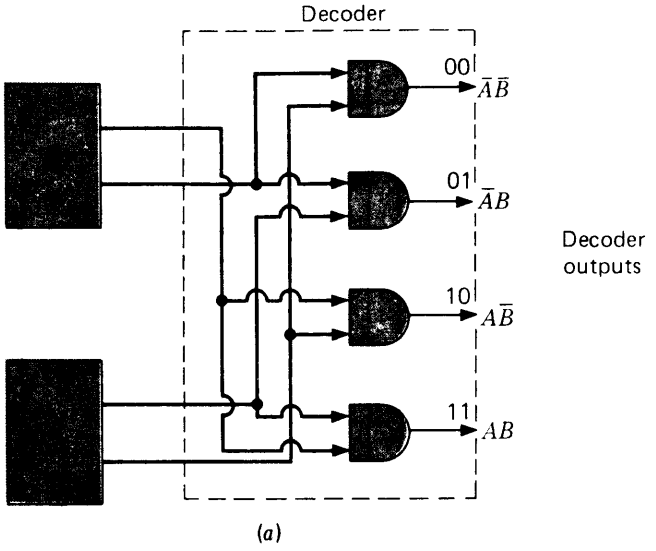
The AND gates connected to the OUT lines on the memory cells in Fig. 6.3 must have the property that when a number of AND gate output lines are connected, the output goes to the highest level. (If any OUT is a 1, the line goes to 1; otherwise, it is a 0.) This is called a *wired OR*. In Fig. 6.4 all four memory cells in the first column are wire-ORed; so if any output line is a 1, the entire line will be a 1. (Memory cells in IC memories are constructed in this manner.)

Now if the READ line is a 1 in Fig. 6.4, then the output values for the flip-flops in the selected row will all be gated onto the output line for each bit in the memory.

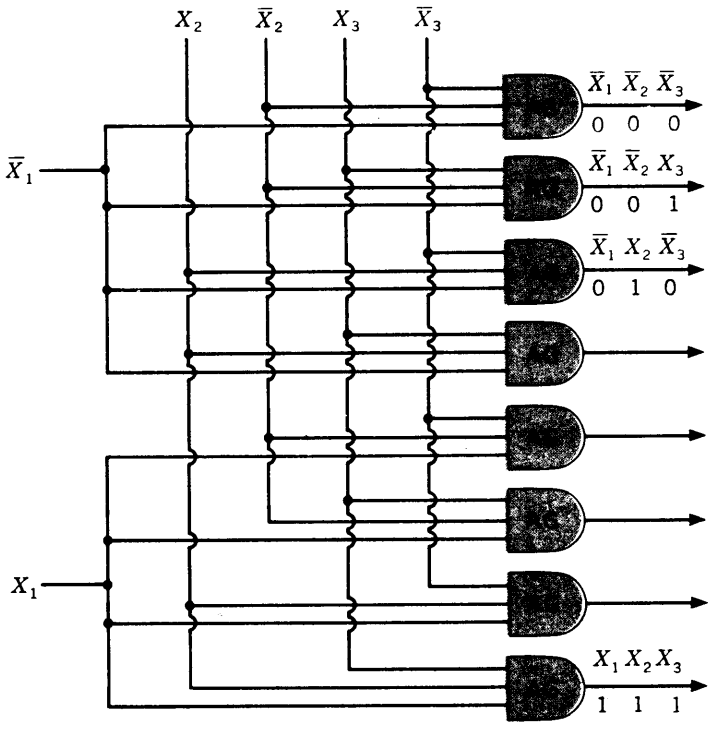
For example, if the second row in the memory contains 110 in the three memory cells, and if the MAR contains 01, then the second output line from the decoder (marked 01) will be a 1, and the input gates and output gates to these three memory cells will be selected. If the READ line is a 1, then the outputs from the

6

THE MEMORY ELEMENT



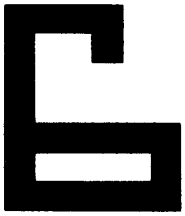
(*a*)



(*b*)

FIGURE 6.5

(*a*) Four-output decoder. (*b*) Parallel decoder.



three memory cells in the second row will be 110 to the AND gates at the bottom of the figure, which will transmit the value 110 as an output from the memory.

If the WRITE line is a 1 and the MAR again contains 01, the second row of flip-flops will have selected inputs. Then the input values on I_1, I_2, and I_3 will be read into the flip-flops in the second row.

As may be seen, this is a complete memory, fully capable of reading and writing. The memory will store data for an indefinite period and will operate as fast as the gates and flip-flops will permit. There is only one problem with the memory—its complexity. The basic memory cell (the flip-flop with its associated circuitry) is complicated, and for large memories the decoder will be large.

To further explore memory organization, we examine decoder construction in more detail, the selection schemes that are commonly used, and finally some examples of IC memories now in production.

DECODERS

6.3 An important part of the system which selects the cells to be read from and written into is the decoder. This particular circuit is called a *many-to-one decoder*, a *decoder matrix*, or simply a decoder. It has the characteristic that for each of the possible 2^n binary input numbers which can be taken by the n input cells, the matrix will have a unique one of its 2^n output lines selected.

Figure 6.5(*b*) shows a decoder which is completely parallel in construction and designed to decode three flip-flops. There are $2^3 = 8$ output lines; and for each of the eight states which the three inputs (flip-flops) may take, a unique output line will be selected. This type of decoder is often constructed by using diodes (or transistors) in the AND gates. The rule is: the number of diodes (or transistors) used in each AND gate is equal to the number of inputs to each AND gate.[2] For Fig. 6.5(*b*) this is equal to the number of input lines (flip-flops which are being decoded). Further, the number of AND gates is equal to the number of output lines, which is equal to 2^n (n is the number of input flip-flops being decoded). So the total number of diodes is equal to $n \times 2^n$. And for the binary decoding matrix in Fig. 6.5(*b*), 24 diodes are required to construct the network. As may be seen, the number of diodes required increases sharply with the number of inputs to the network. For instance, to decode an eight-flip-flop register, we would require $8 \times 2^8 = 2048$ diodes if the decoder were constructed in this manner.

As a result, several other types of structures are often used in building decoder networks. One such structure, called a *tree-type* decoding network, is shown in Fig. 6.6. This tree network decodes four flip-flops and so has $2^4 = 16$ output lines, a unique one of which is selected for each state of the flip-flops. An examination will show that 56 diodes are required to build this particular network, while $2^4 \times 4 = 64$ diodes would be required to build the parallel decoder type shown in Fig. 6.5.

Still another type of decoder network is shown in Fig. 6.7. It is called a *balanced multiplicative decoder network*. Notice that this network requires only 48

[2]The rule for transistors is the same: The number of transistors required equals the number of inputs.

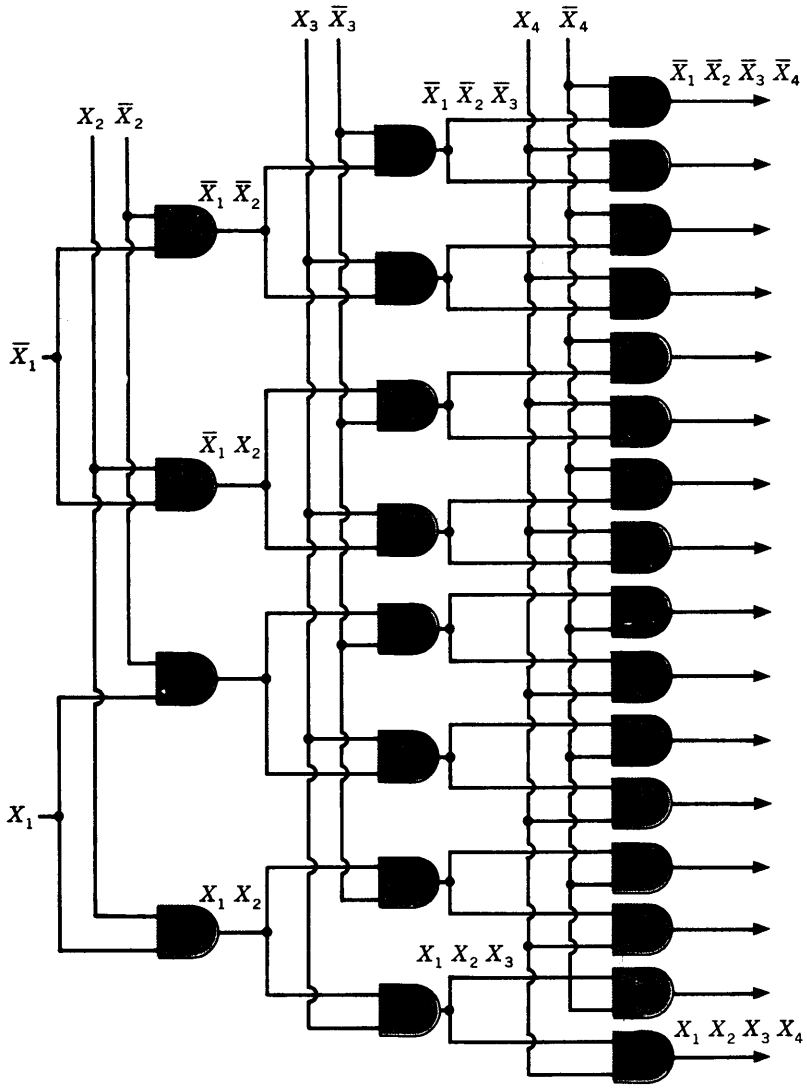


FIGURE 6.6

Tree decoder.

diodes. It can be shown that the type of decoder network illustrated in Fig. 6.7 requires the minimum number of diodes for a complete decoder network. The difference in the number of diodes, or decoding elements, to construct a network such as shown in Fig. 6.7, compared with those in Figs. 6.5 and 6.6, becomes more significant as the number of flip-flops to be decoded increases. The network shown in Fig. 6.5, however, has the advantage of being the fastest and most regular in construction of the three types.

Having studied the three types of decoding matrices now used in digital machines, we simply draw the decoder networks as a box with n inputs and 2^n outputs, with the understanding that one of the three types of circuits shown in Figs. 6.5 to 6.7 will be used in the box. Often only the uncomplemented inputs

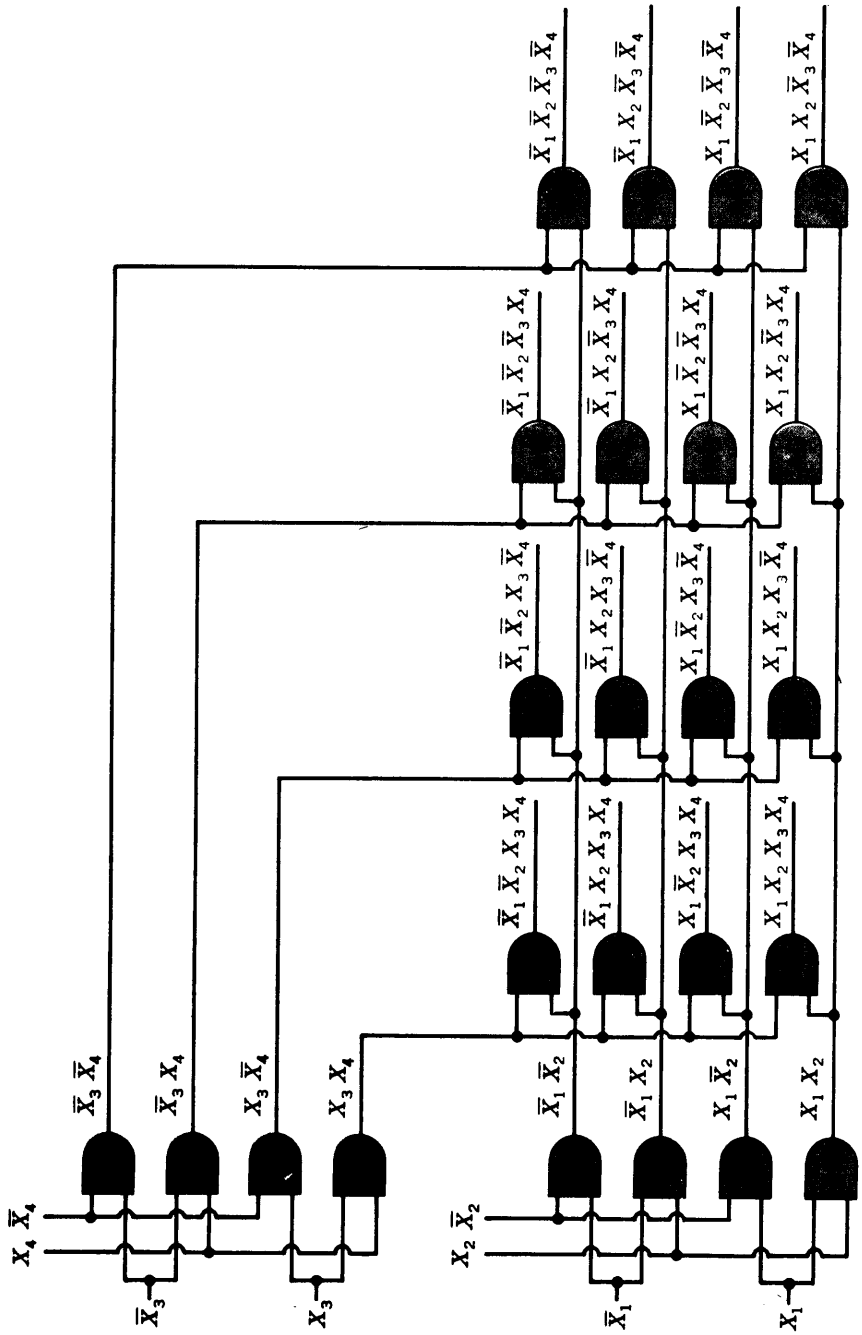
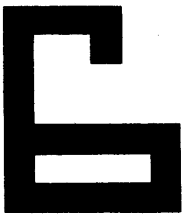


FIGURE 6.7

Balanced decoder.



are connected to decoders, and inverters are included in the decoder package. Then a three-input (or three-flip-flop) decoder will have only three input lines and eight outputs.

DIMENSIONS OF MEMORY ACCESS

6.4 The memory organization in Fig. 6.4 has a basic linear-select (one-dimensional) selection system. This is the simplest organization. However, the decoder in the selection system becomes quite large as the memory size increases.

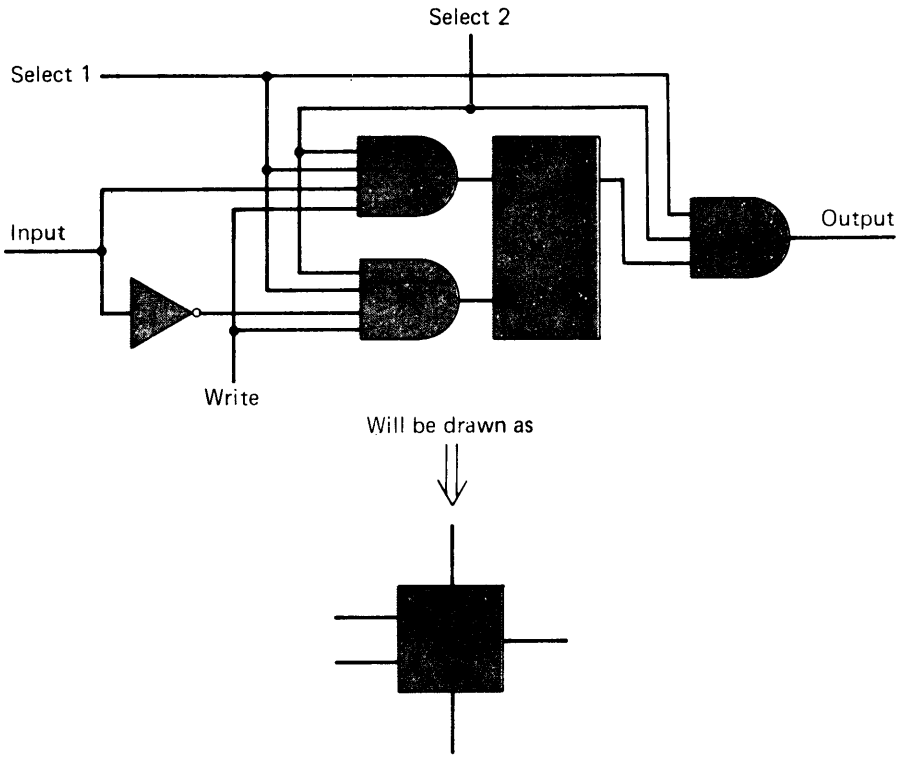
As an example, we assume a parallel decoder as shown in Fig. 6.5(*b*). These are widely used in IC packages because of their speed and regular (symmetric) construction.

Consider now a decoder for a 4096-word memory, a common size for an IC package. There will be 12 inputs per AND gate, and 4096 AND gates are required. If a diode (or transistor) is required at each AND gate's input, then $12 \times 4096 = 49{,}152$ diodes (or transistors) will be required. This large number of components is the primary objection to this memory organization.

Let us now consider a *two-dimensional selection system*. First we need to add another SELECT input to our basic memory cell. This is shown in Fig. 6.8. Now both the SELECT 1 and the SELECT 2 must be 1s for a flip-flop to be selected.

FIGURE 6.8

Two-dimensional memory cell.



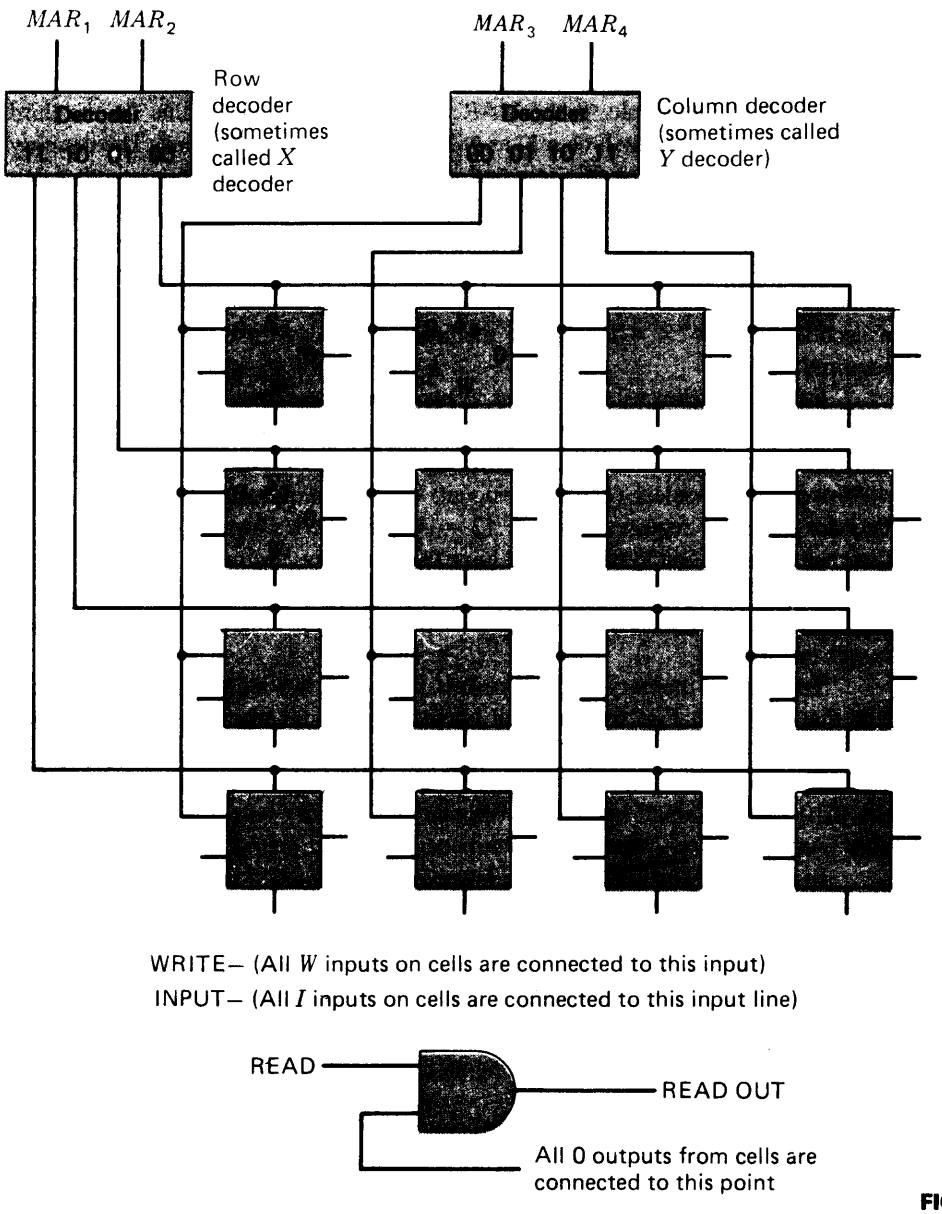
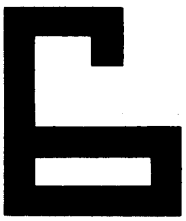


FIGURE 6.9

Two-dimensional IC memory organization.

Figure 6.9 shows a two-dimensional memory selection system using this cell. Two decoders are required for this memory, which has 16 words of only 1 bit per word (for clarity of explanation). The MAR has 4 bits and thus 16 states. Two of the MAR inputs go to one decoder and two to the other.

To illustrate the memory's operation, if the MAR contains 0111, then the value 01 goes to the left decoder and 11 goes to the upper decoder. This will select the second row (line) from the left decoder and the rightmost column from the top decoder. The result is that only the cell (flip-flop) at this intersection of the second



row and the rightmost column will have both its SELECT lines (and as a result its AND gates) enabled. As a result, only this particular single cell will be selected, and only this flip-flop can be read from or written into.

As another example, if the MAR contains 1001, then the line for the third row of the left decoder will be a 1, as will be the second column line. The memory cell at the intersection of this row and column will be enabled, but no other cell will be enabled. If the READ line is a 1, the enabled cell will be read from; if the WRITE line is a 1, the enabled cell will be written into.

Now let us examine the number of components used. If a 16-word 1-bit memory were designed by using the linear-select, or one-dimensional, system, then a decoder with 16 × 4 inputs and therefore 64 diodes (or transistors) would be required.

For the two-dimensional system, 2 two-input four-output decoders are needed, each requiring 8 diodes (transistors); so 16 diodes are required for both decoders.

For a 4096-word 1-bit-per-word memory, the numbers are more striking. A 4096-word linear-select (one-dimensional) memory requires a 12-bit MAR. This decoder therefore requires 4096 × 12 = 49,152 diodes or transistors. The two-dimensional selection system would have two decoders, each with six inputs. Each would require $2^6 \times 6 = 384$ diodes or transistors, that is, a total of 768 diodes or transistors for the decoders. This is a remarkable saving and extends to even larger memories.

To make a memory with more bits per word, we simply make a memory like that shown in Fig. 6.9 for each bit in the word (except that only one MAR and the original two decoders are required).

The above memory employs a classic two-dimensional selection system. Figure 6.10 shows a small high-speed (bipolar) IC memory with 256 bits on a single chip. As can be seen, this is a two-dimensional select memory.

In a two-dimensional memory, however, simplification in decoder complexity is paid for with cell complexity. In some cases this extra cell complexity is inexpensive, but it can be a problem, and so a variation of this scheme is used. The most used variation on the basic two-dimensional selection system is illustrated in Fig. 6.11. This memory uses two decoders, as in the previous scheme; however, the memory cells are basic memory cells, as shown in Fig. 6.3.

The selection scheme uses gating on the READ and WRITE inputs to achieve the desired two-dimensionality.

Let us consider a WRITE operation. First assume that the MAR contains 0010. This will cause the 00 output from the upper decoder to be a 1, selecting the top row of memory cells. In the lower decoder the 10 output will become a 1, and this is gated with an AND gate near the bottom of the diagram, turning on the *W* inputs in the third column. As a result, for the memory cell in the top row and third column, the *S* input and the *W* input will be a 1. For no other memory cell will both *S* and *W* be a 1, and so no other memory cell will have its *RS* flip-flop set to the input value. (Notice that all *I* inputs on the memory cells are connected to the input value D_I.)

Consideration of other values for the MAR will indicate that for each value a unique memory cell will be selected for the write operation. Therefore, for each MAR state only one memory cell will be written into.

The read operation is similar. If the MAR contains 0111, then the upper

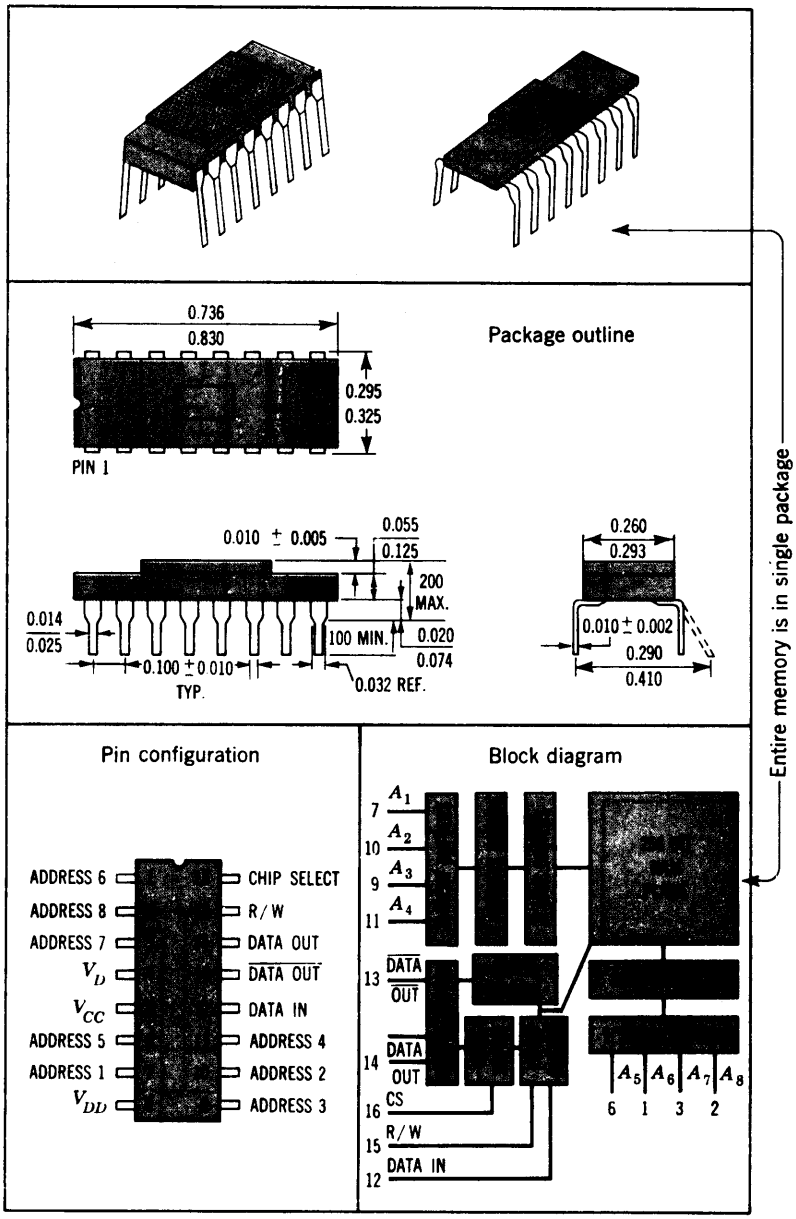


FIGURE 6.10

Single-chip 256-bit memory. (*Intel Corp.*)

decoder's 01 line will be a 1, turning on the *S* inputs in the second row of memory cells. As a result only these four cells in the entire array are capable of writing a 1 on the output lines. (Again, the memory cells are wire-ORed by having their outputs connected, this time in groups of four.)

The lower decoder will have input 11, and so its lowest output line will carry a 1. This 1 turns on the rightmost AND gate in the lowest row, which enables the output from the rightmost column of memory cells. Only the second cell down has

6

THE MEMORY ELEMENT

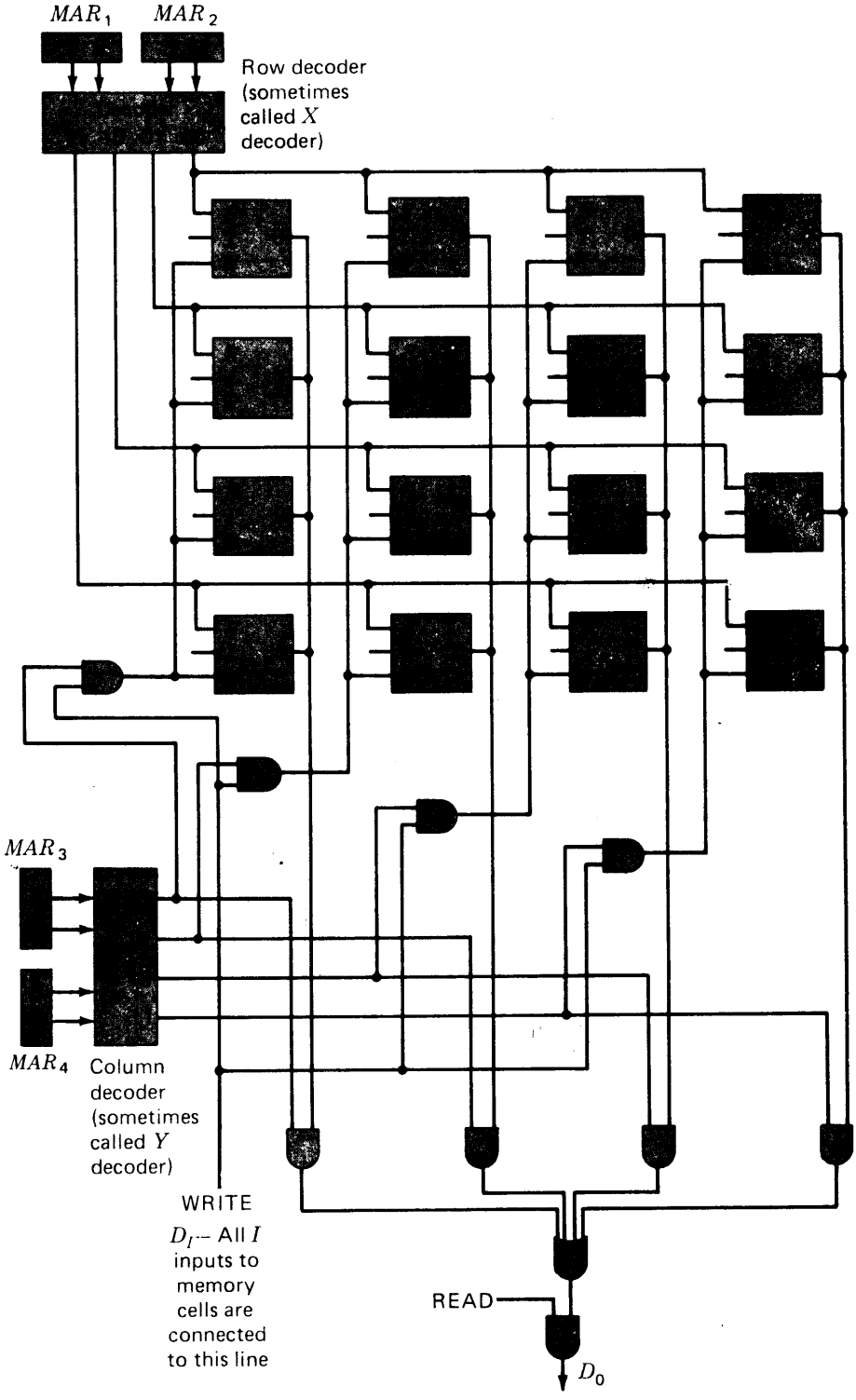
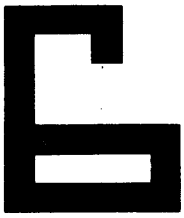


FIGURE 6.11

IC memory chip layout.



its output enabled, however, and so the output from the rightmost AND gate will have the value in the cell. This value then goes through the OR gate and the AND gate at the bottom of the diagram, the AND gate having been turned on by the READ signal.

Examination will show that each input value from the MAR will select a unique memory cell to be read from, and that cell will be the same as would have been written into if the operation were a write operation.

This is basically the organization used by most IC memories at this time. The chips contain up to 256K bits. The number of rows versus the number of columns in an array is determined by the designers who choose the numbers that will reduce the overall component count.

All the circuits necessary for a memory are placed on the same chip, except for the MAR flip-flops which sometimes are not placed on the chip; the inputs go directly to the decoders. This will be clearer when interfacing with a bus has been discussed.

CONNECTING MEMORY CHIPS TO A COMPUTER BUS

6.5 The present trend in computer memory connection is to connect the computer central processing unit (CPU), which does the arithmetic, generates control, etc.,[3] to the memory by means of a *bus*. The bus is simply a set of wires which are shared by all the memory elements to be used.

Microprocessors always use a bus to interface memory. In this case the memory elements will be IC chips, which are in IC containers just like those described in Chap. 4 and shown in Fig. 6.10.

The bus used to connect the memories generally consists of (1) a set of *address lines* to give the address of the word in memory to be used (these are effectively an output from a MAR on the microprocessor chip); (2) a set of *data wires* to input data from the memory and output data to the memory; and (3) a set of *control wires* to control the read and write operations.

Figure 6.12 shows a bus for a microcomputer. To simplify drawings and clarify explanations, we use a memory bus with only three address lines, three output data lines, two control signals, and three input data lines. So the memory to be used is an 8-word 3-bit-per-word memory.

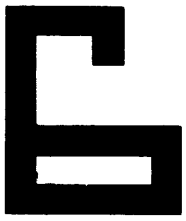
The two control signals work as follows. When the read-write ($R/\overline{W}$) line is a 1, the memory is to be read from; when the $R/\overline{W}$ line is a 0, the memory is to be written into.[4] The MEMORY ENABLE signal ME is a 1 when the memory is either to be read from or to be written into; otherwise, it is a 0.

The IC memory package is shown in Fig. 6.13. Each IC package has three address inputs A_0, A_1, and A_2, and $R/\overline{W}$ input, an output bit D_O, an input bit D_I, and a CHIP SELECT $\overline{CS}$. Each package contains an 8-word 1-bit memory.

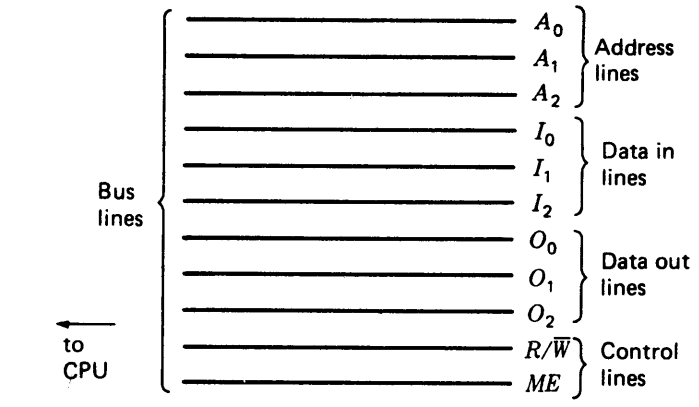
The IC memory chip works as follows. The address lines A_0, A_1, and A_2

[3] A CPU includes the arithmetic and control sections of a computer.

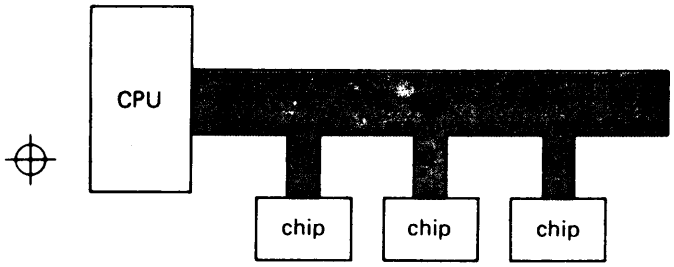
[4] This is quite similar to the READ and WRITE signals used in prior memory description. An AND gate connected to ME and $R/\overline{W}$ will generate a READ signal, and an inverted $R/\overline{W}$ which is ANDed with ME will give a WRITE signal.



THE MEMORY ELEMENT



(a)



(b)

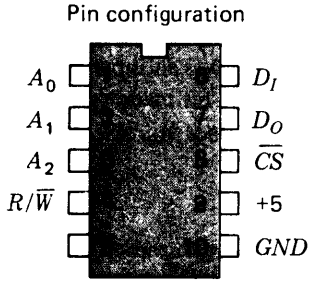
FIGURE 6.12

Bus for computer system. (*a*) Bus lines. (*b*) Bus, CPU, and memory organization.

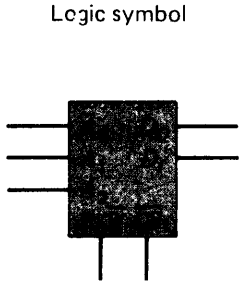
must be set to the address to be read from or written into (refer to Fig. 6.13). If the operation is a read, the R/$\overline{\text{W}}$ line is set to a 1 and the $\overline{\text{CS}}$ line is brought to 0 (the $\overline{\text{CS}}$ line is normally a 1). The data bit may then be read on line D_O. Certain timing constraints must be met, however, and these will be supplied by the IC manufacturer. Figure 6.14 shows several of these. The value T_R is the minimum cycle time a read operation requires. During this period the address lines must be stable. The value T_A is the access time, which is the minimum time from when the address lines are stable until data can be read from the memory. The value T_{CO} is the minimum time from when the $\overline{\text{CS}}$ line is made a 0 until data can be read.

FIGURE 6.13

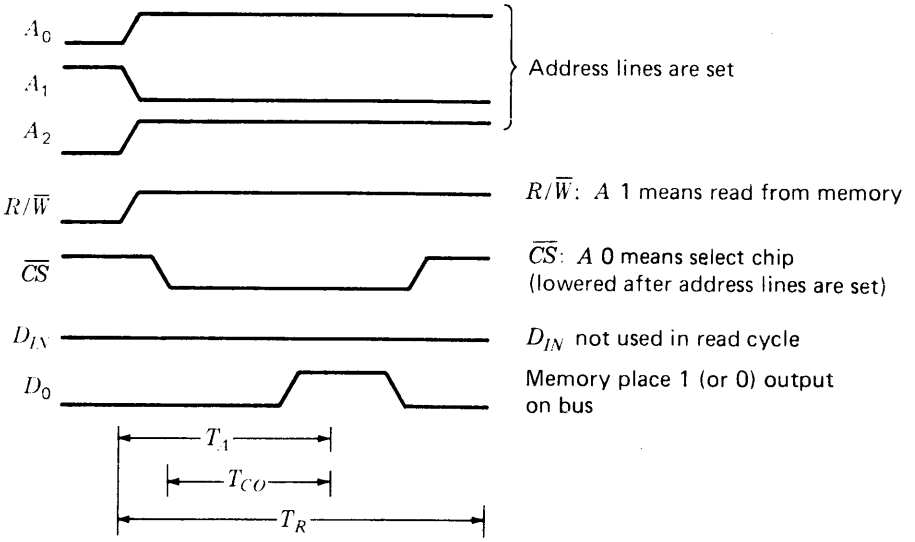
IC package and block diagram symbol for RAM chip. (*a*) Pin configuration. (*b*) Logic symbol.



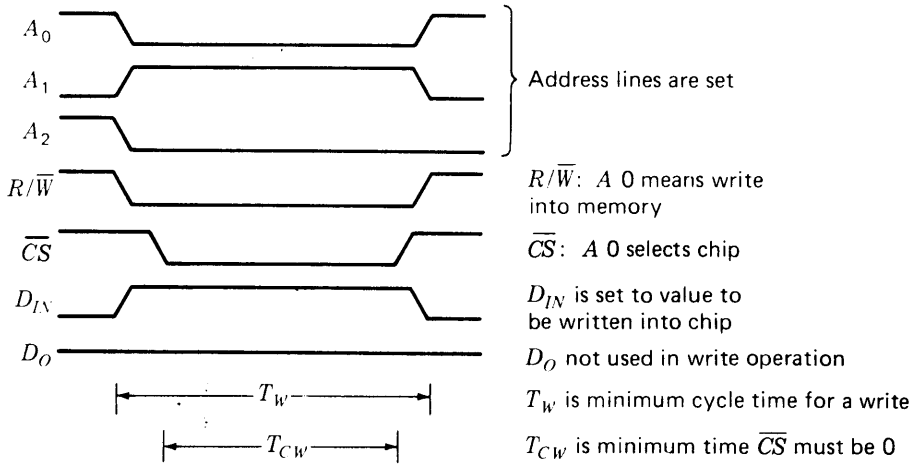
(a)



(b)



(*a*) READ cycle



(*b*) WRITE cycle

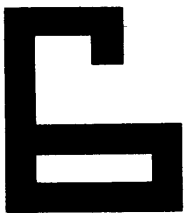
FIGURE 6.14

Timing for bus IC memory. (*a*) READ cycle. (*b*) WRITE cycle.

The bus timing must accommodate the above times. It is important that the bus not operate too fast for the chip and that the bus wait for at least the time T_A after setting its address lines before reading and wait at least T_{C0} after lowering the $\overline{CS}$ line before reading. Also, the address line must be held stable for at least the period T_R.

For a write operation, the address to be written into is set up on the address lines, the R/$\overline{W}$ line is made a 0, $\overline{CS}$ is brought down, and the data to be read are placed on the D_{IN} line.

The time interval T_W is the minimum time for a WRITE cycle; the time T_H is the time the data to be written into the chip must be held stable. Different types



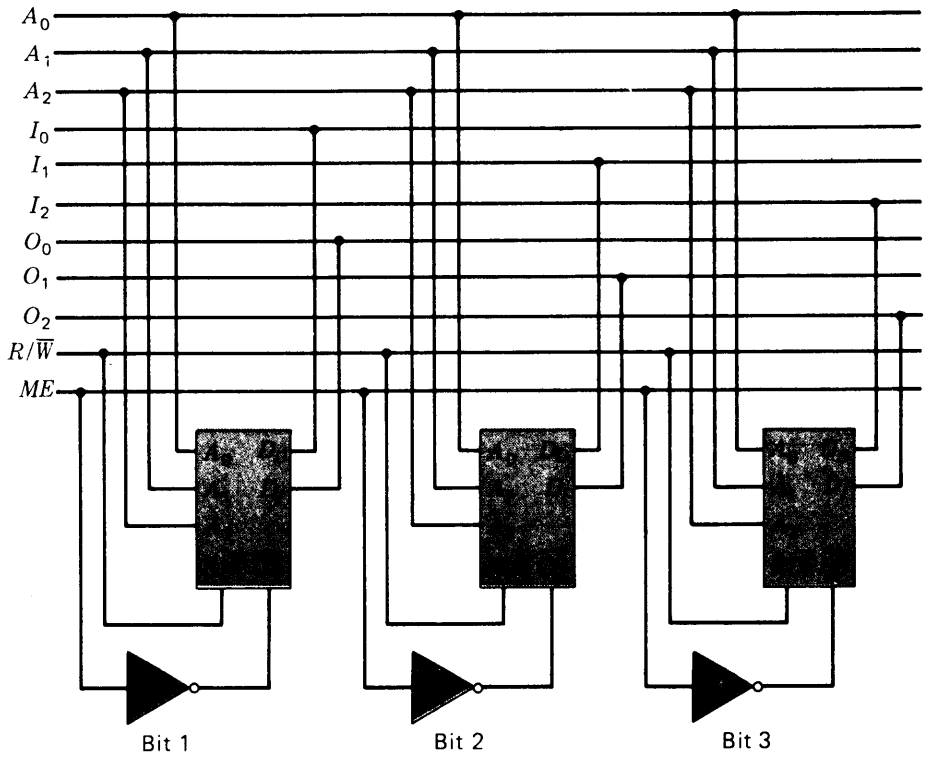


FIGURE 6.15

Interfacing chips to a bus.

of memories have different timing constraints which the bus must accommodate. We assume that our bus meets these constraints.[5]

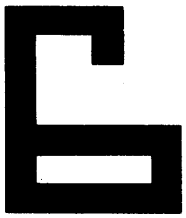
To form an 8-word 3-bit memory from these IC packages (chips), the interconnection scheme in Fig. 6.15 is used. Here the address line to each chip is connected to a corresponding address output on the microcomputer bus. The CHIP SELECT input $\overline{CS}$ of each chip is connected to the MEMORY ENABLE output ME from the microprocessor via an inverter, and the $R/\overline{W}$ bus line is connected to the $R/\overline{W}$ input on each chip.

If the microprocessor CPU wishes to read from the memory, it simply places the address to be read from on the address lines, puts a 1 on the $R/\overline{W}$ line, and raises the ME line. Then each chip reads the selected bit onto its output line, and the CPU can read these values on its I_1, I_2, and I_3 lines. (Notice that a chip's output is a bus input.)

Similarly, to write a word into the memory, the CPU places the address to be written into on the address lines, the bits to be written on the O_1, O_2, O_3 lines, lowers $R/\overline{W}$, and then raises ME.

In practice, for microprocessors the memory words now generally contain 8 bits each. (Some new large microprocessors have 16-bit words.) There are generally

[5]On the other hand, if a specific microprocessor is used, the memory must be fast enough to accommodate the bus.



16 address lines, and so 2^{16} words can be used in the memory. However, memory chips tend to have from 8 to 14 (at most) memory address lines. Fortunately there is a simple way to expand memories, and this is shown for our small system in Fig. 6.16.

In this example the chips again have three address lines, but the microprocessor bus has five lines. To enable connection, a two-input decoder is connected to the two most significant bits of the address section of the bus, while the three least significant bits are connected to the chip address buses as before.

Now the decoder outputs are each gated with the ME control signal by a NAND gate; so when ME is raised, a single CHIP SELECT line is lowered (the outputs from the NAND gates are normally high). The decoder therefore picks the chip that is enabled, and the address lines on the enabled chip select the memory cell to be written into or read from. The decoding on the chip then selects the particular memory cell to be read from or written into.

The principle shown in Fig. 6.16 is widely used in computers. Memory chips almost invariably have fewer address inputs than buses, and so this expansion technique is necessary to memory usage. Notice that only 1 bit of the memory word is completely drawn in Fig. 6.16. (One chip from the second bit is also shown.) An entire 32-word 3-bit memory would require 12 chips of the type shown here. Most buses combine the input and output data lines into a single set of lines in a way to be explained in the section on buses.

As may be seen, a microcomputer or minicomputer (or any computer) can be purchased with a minimal memory, and then the memory can be expanded by adding more chips, up to the size that the bus address lines can accommodate.

The remainder of this chapter is structured as follows. Read-only memories are explained; then disk, drum, and tape memories are covered. Finally, some information on recording techniques is presented.

RANDOM-ACCESS SEMICONDUCTOR MEMORIES

6.6 The ability to fabricate large arrays of electronic components using straightforward processing techniques and to make these arrays in small containers at reasonable prices has made semiconductor memories the most popular at this time.

Although a number of different schemes and devices are available, at present there are six main categories of IC memories:

1 *Bipolar memories* These are essentially flip-flop memories with the flip-flops fabricated using standard *pn*-junction transistors. These memories are fast but tend to be expensive.

2 *Static MOS memories* These are fabricated by using MOS field-effect devices to make flip-flop circuits. These memories are lower in speed than the bipolar memories, but cost less, consume less power, and have high packing densities.

3 *Dynamic MOS memories* These are fabricated using MOS devices. But instead of a flip-flop being used for the basic memory cell, a charge is deposited on a capacitor fabricated on the IC chip, and the presence or absence of this charge determines the state of the cell. The MOS devices are used to sense and deposit

6

THE MEMORY ELEMENT

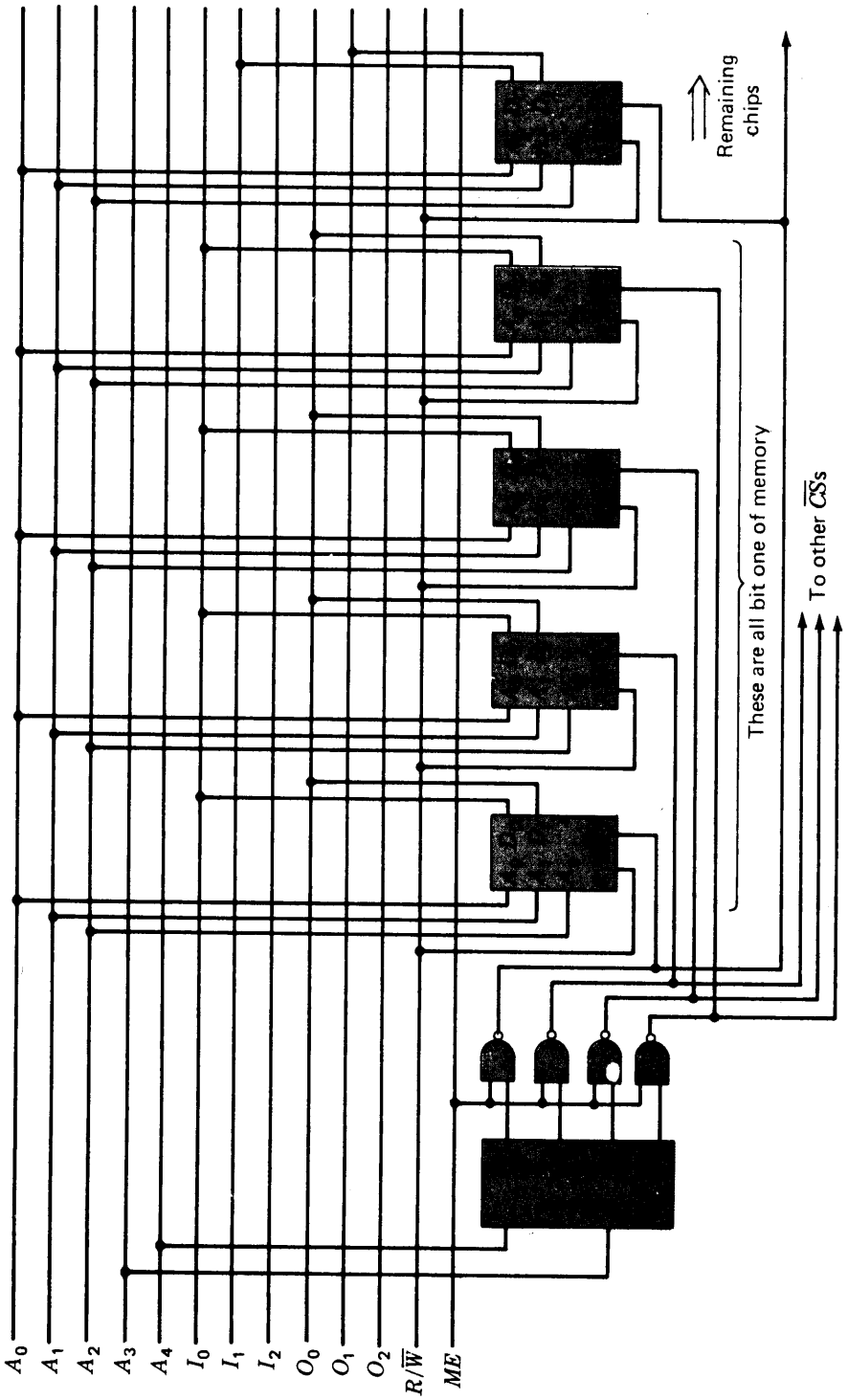


FIGURE 6.16

Layout for adding memory to a bus.

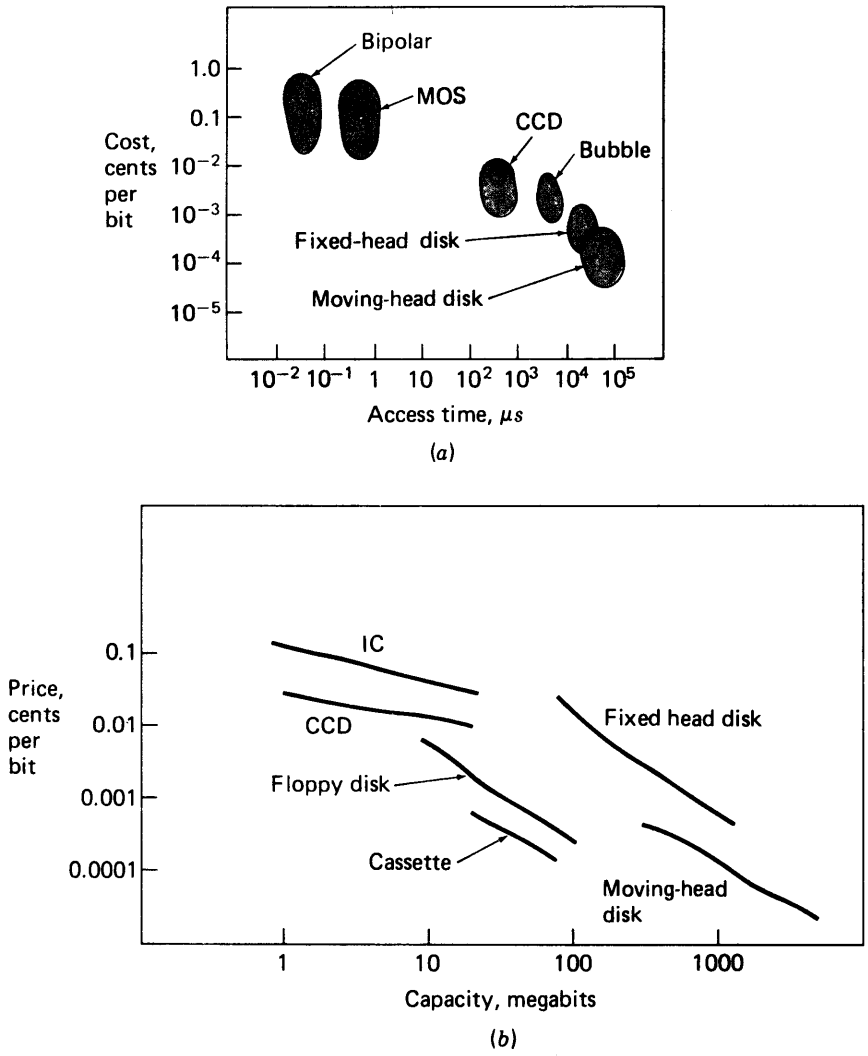


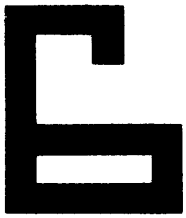
FIGURE 6.17

Characteristics of memory devices. (*a*) Access time versus cost for some memory devices. (*b*) Cost versus capacity in megabits.

the charge on the capacitors used. Since the charge used will slowly dissipate in time, it is necessary to periodically refresh this charge, and so the memories are called dynamic memories. (MOS or bipolar flip-flop memories are called static memories.) These memories tend to be slower than the other types, but they are also less expensive, consume less power, and have a high packing density.

4 *CMOS memories* CMOS utilizes both *p*- and *n*-channel devices on the same substrate. As a result, it involves a more complex manufacturing process. CMOS has improved speed and power output figures over *n*- and *p*-channel MOS, but it costs more.

5 *Silicon-on-sapphire (SOS) memories* SOS is similar to CMOS. Devices are formed on an insulating substrate of sapphire. This reduces the device capacitance and improves speed. However, SOS is costly.



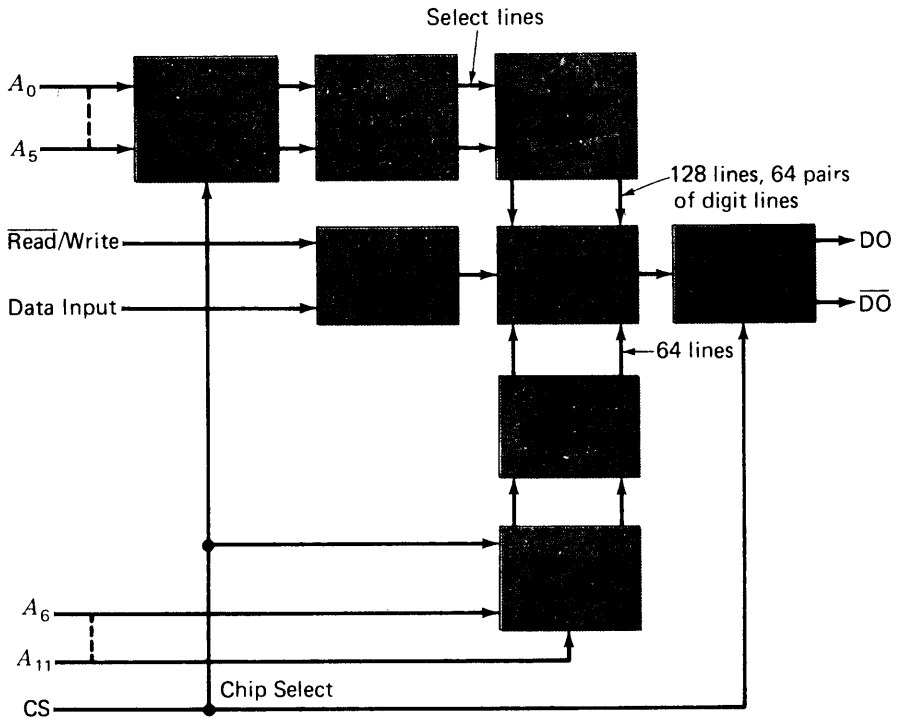


FIGURE 6.18

Block diagram of MOS static memory. (*Electronic Memories and Magnetics Corp.*)

6 *Integrated injection logic (IIL) memories* The IIL circuits eliminate the load resistors and current sources of TTL circuits. This reduces power consumption over bipolar memories, giving greater packing density than bipolar devices. As a result, IIL mixes the speed of bipolar memories with the packing density of MOS. It is medium-cost.

Figure 6.17 shows the characteristics of the basic memory devices in this chapter. Figure 6.17(*a*) shows cost per bit versus access time for these devices, and Fig. 6.17(*b*) shows price versus storage capacity. Notice the semiconductor memories are faster but more expensive than the disk devices. Magnetic-tape devices provide still slower operation but at a still lower cost per bit.

STATIC RANDOM-ACCESS MEMORIES

6.7 If we compare bipolar and MOS technologies, bipolar offers a speed advantage, although, until recently, limitations imposed by the need for isolation between transistors have limited packing density and hence per-chip storage capacity. Bipolar components can provide access times of under 10 ns, in contrast with 300 ns or more for PMOS and 20 ns for NMOS. MOS devices have relatively high internal capacitance and impedance, leading to longer time constants and access times.

The operation of a typical 4096-bit static NMOS memory chip is detailed in Figs. 6.18 through 6.20. The 4096 bits of memory are organized in an array of 64

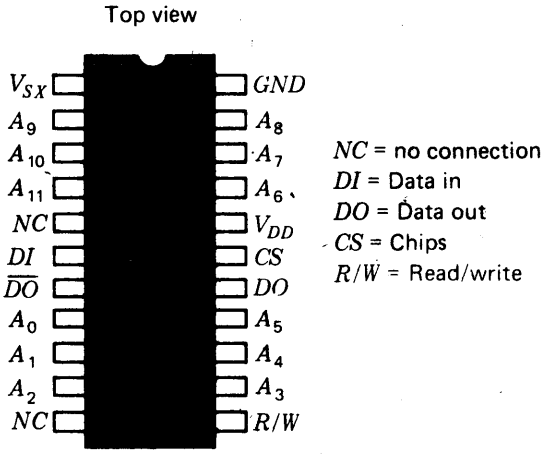


FIGURE 6.19

MOS static memory pin assignment for Fig. 6.18. (*Electronic Memories and Magnetics Corp.*)

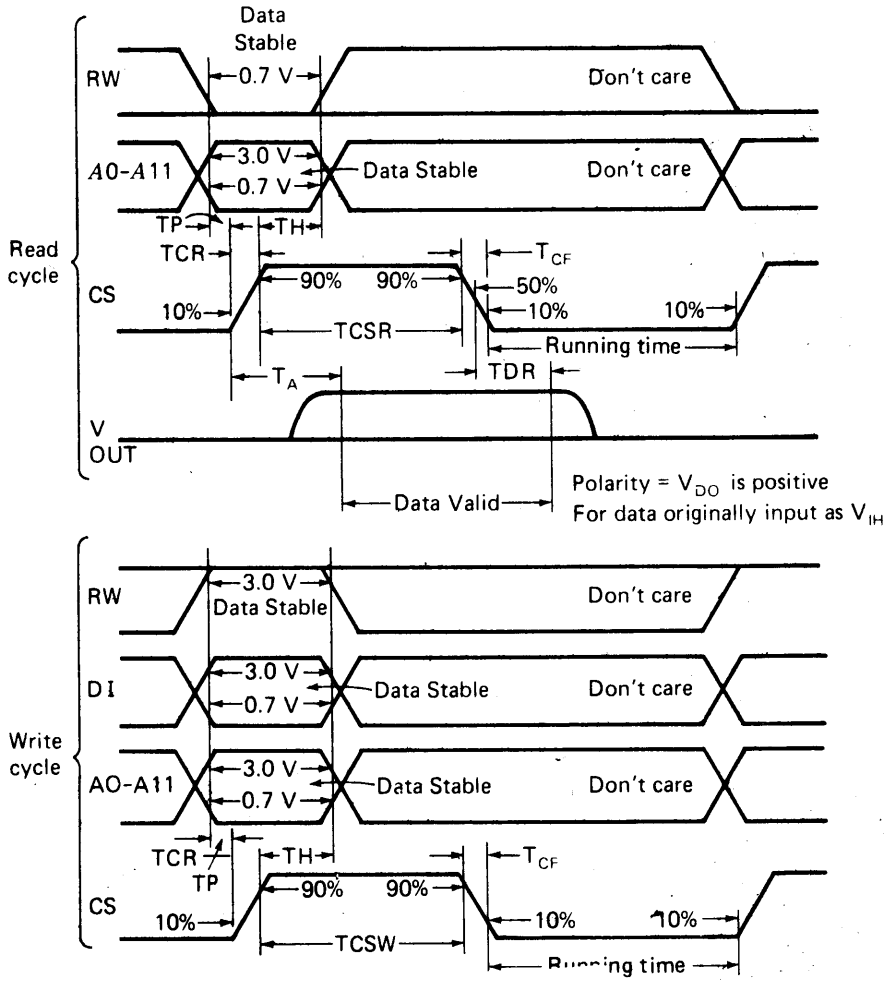


FIGURE 6.20

Timing diagrams. (*Electronic Memories and Magnetics Corp.*)

TABLE 6.1 STATIC MEMORY CHARACTERISTICS

MANUFACTURER AND PART NUMBER	ORGANIZATION	ACCESS TIME, ns	POWER, MW	PINS	TYPE
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

rows by 64 columns. The memory bits are accessed by simultaneously decoding the X address A_0–A_5 for the rows and the Y address A_6–A_{11} for the columns.

The basic organization is the same as that in Fig. 6.11.

The CHIP SELECT (CS) input controls the operation of the memory. When CS is low, the input address buffers, decoders, sensing circuits, and output stages are held in the off state, and power is supplied only to the memory elements. When the CS goes high, the memory is enabled. The CS pulse clocks the TTL level addresses, READ-WRITE, and DATA input into D flip-flops and enables the output stage. When a cell is read from, one of the two outputs will be a 1 (DO for data originally input as a 1, $\overline{\text{DO}}$ for data originally input as a 0).

As shown in Fig. 6.19, this memory chip is packaged in a 22-pin dual-in-line package. By assembling a number of these chips, a large, moderately fast memory can be constructed. Memory cycle times for this chip are on the order of 30 ns, and Fig. 6.20 shows the timing for the chips.

The characteristics of several static memory chips are shown in Table 6.1.

DYNAMIC RANDOM-ACCESS MEMORIES

6.8 The preceding memories have all used flip-flops for the basic memory cells in an array. There is another class of random-access memory called *dynamic random-access memories* (DRAMs). These memories have individual cells composed of from one to three MOS transistors plus a capacitor. The cell's state is determined by placing or not placing a positive charge on the capacitor. Thus, if the capacitor has no charge, the cell may represent a 0, and with a positive charge the cell may represent a 1. The advantage of this kind of memory is that the individual cells are simpler than flip-flops, requiring less area on the chips and having lower power consumption. The disadvantages are that DRAMs are slower and the charge slowly leaks from the capacitor. Thus the contents of each cell must be rewritten into each cell periodically. This is called refreshing the memory. Despite the difficulties in refreshing dynamic memories and their relatively slow speed (100-ns access times are representative), the memory costs are enough lower than for static memories for dynamic memories to be widely used in present-day systems.

Dynamic random-access memories are organized in the same manner as flip-flop memories (the organization in Fig. 6.11 is the most frequently used except that some circuit "tricks" enable the combining of each column input line and output line into a pair of lines called *bit lines*).

Two-dimensional selection with decoders is standard. Dynamic memory chips now contain up to 256K of memory per chip; however, 16K and 64K chips are more widely used.

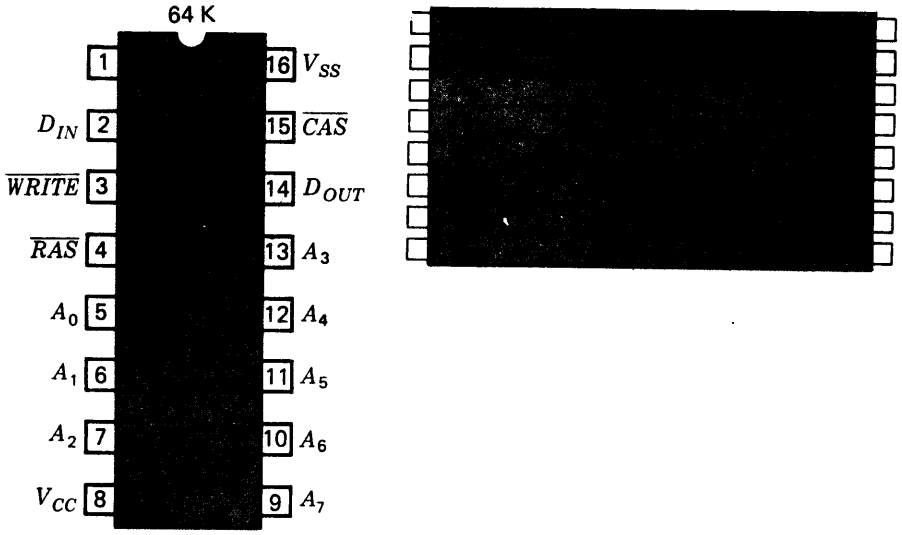


FIGURE 6.21

Pin-out for dynamic memory chip.

Figure 6.21 shows the standard pin layout for a 64K-bit dynamic RAM. There are not enough address lines into this chip, and so the addresses are time-multiplexed (that is, put on in two sections, one right after the other). This is shown in Fig. 6.22. First the row address[6] is placed on A_0–A_7 and $\overline{RAS}$ is lowered; then the column address is placed on A_0–A_7 and $\overline{CAS}$ is lowered. To use RAMs of this kind, extra circuitry for multiplexing address lines and to generate the REFRESH signal must be used.[7] Nevertheless because of their high packing density and low cost, the extra complexity of these specialized circuits is compensated for, and these memories are widely used.

As was stated, the primary advantages of dynamic MOS memories lie in the simplicity of the individual cells. There is a secondary advantage in that power

[6]The row address is the first half of the field 16-bit address, and the column address is the second half.
[7]IC manufacturers make special chips for this purpose. For these particular memories, refresh is done a column at a time so only the rows must be sequenced through. To refresh a row, the row address is placed on the address lines and the $\overline{RAS}$ is lowered. Each row must be refreshed in a 2-ms period.

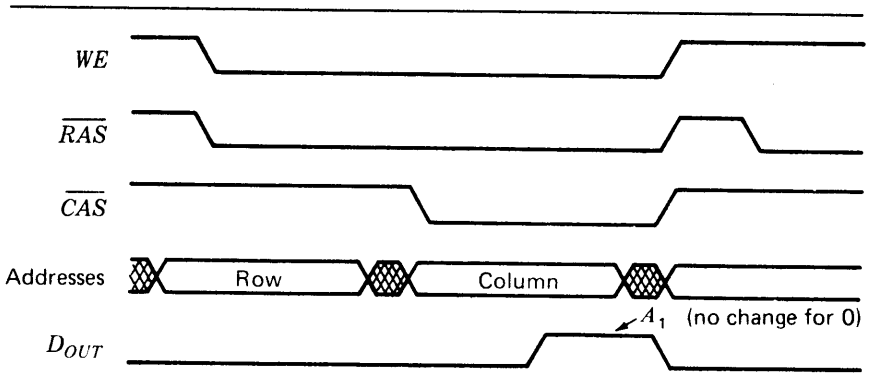


FIGURE 6.22

Pin layout and address timing for 64K dynamic memory.

TABLE 6.2 DYNAMIC RANDOM-ACCESS MEMORY CHARACTERISTICS

MANUFACTURER AND PART NUMBER	ORGANIZATION	ACCESS TIME, ns	POWER, MW		REFRESH RATE
			ACTIVE	STANDBY	
TI TM SY164	64K × 1	150	200	27	128 cycles/2 ms
Motorola MCM 6664	64K × 1	120	275	30	128 cycles/2 ms
TI TMS S4416	16K × 4	150	125	18	256 cycles/4 ms
Mostek MK4332	32K × 1	200	462	40	128 cycles/2 ms
Hitachi	8K × 8	120	170	100	256 cycles/4 ms
Fujitsu MB8264	64K × 1	150	300	28	256 cycles/4 ms
TI	8K × 8	100	200	100	128 cycles/2 ms

need not be applied to the cells when they are not being read from or written into, and so the power is conserved. This makes for higher packing densities per chip. The obvious disadvantage lies in the need to refresh these cells every few milliseconds since charge continually leaks from the capacitors. External circuitry to control the refresh rewrite is generally required, or sometimes the memories may include special circuits to refresh when commanded. As a result, extra refresh memory cycles are required, but these occupy only a small percentage of the overall operating time.[8] Table 6.2 lists some of the characteristics of dynamic memories.

Memory controller chips are used to control several dynamic memory chips assembled into a memory. These controller chips handle the sequencing of the row and column addresses into the individual chips during a normal memory access and also control necessary refreshing of the memory. The refreshing generally involves a counter which sequences through all possible states during a 2-ms period. The memory controller chip(s) places the counter outputs on the address lines of the dynamic memory chips and lowers the $\overline{\text{RAS}}$ inputs, loading the counter addresses into the row select latches (flip-flops). The controller attempts to perform refresh operations between memory accesses when this is possible, thus reducing the time lost to memory refreshes. (These are called "hidden" refreshes.)

Use of dynamic memory controller chips simplifies dynamic memory design and optimizes memory operation.

READ-ONLY MEMORIES

6.9 A type of storage device called a *read-only memory* (ROM) is widely used. ROMs have the unique characteristic that they can be read from, but not written into. Thus the information stored in these memories is introduced into the memory in some manner such that the information is semipermanent or permanent. Sometimes the information stored in a ROM is placed in the memory at the time of construction, and sometimes devices are used in which the information can be changed. In this section we study several types of ROMs. These are characteristic of this particular class of memory devices, and most devices are variations on the principles presented.

[8]There are even some interesting internal refresh mechanisms called *charge pumps* which require application of a sine wave on one of the inputs and make refreshing transparent (invisible) to the user. Other refresh strategies rewrite entire rows by using a single REFRESH pulse.

TABLE 6.3 BINARY-TO-GRAY CODE VALUES

INPUT				OUTPUT			
X_1	X_2	X_3	X_4	Z_1	Z_2	Z_3	Z_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Basically a ROM is a device with several input and output lines such that for each input value there is a unique output value. Thus a ROM physically realizes a truth table, or table of combinations. A typical (small) table is shown in Table 6.3. This list of input-output values is actually a list of binary-to-Gray code values. (The Gray code is discussed in the next chapter.) It is important to see that the list can be looked at in two ways: (1) as a table for a gating network with four inputs and four outputs and (2) as a list of addresses from 0 to 15, given by the X values, and the contents of each address, given by the values of Z. Thus we might construct a gating network as in Fig. 6.23 which would give the correct Z output for each X input. (The boxes with $\oplus$ are mod 2 adders.)

Table 6.2 could also be realized by a 16-word 4-bit-per-word IC memory into which we had read 0000 at address 0; 0001 at address 1; 0011 in the next address; and so on until 1000 at the last address. If we never wrote into this memory

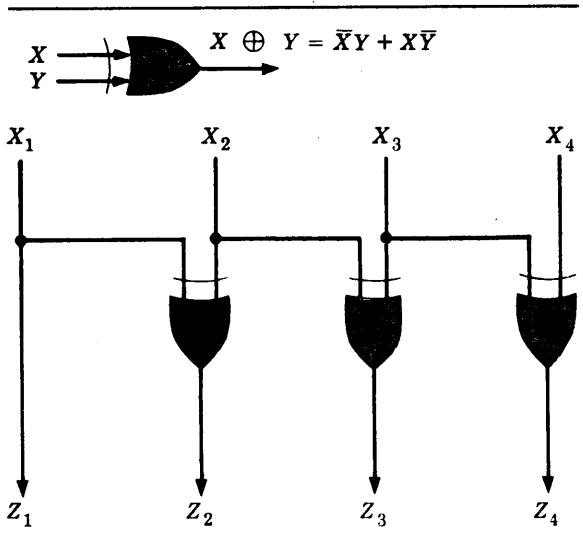
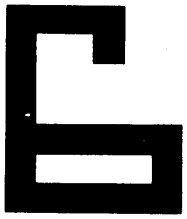


FIGURE 6.23

Combinational network for binary-to-Gray code.



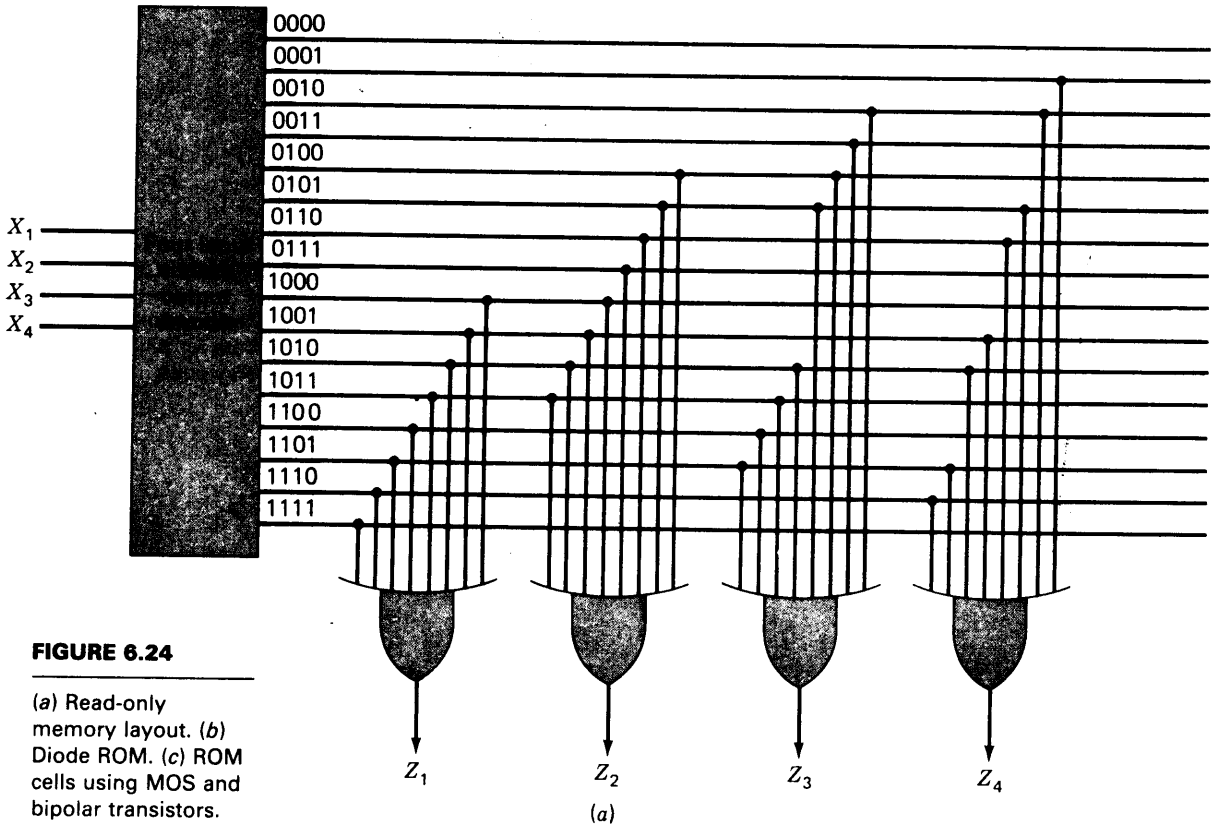
THE MEMORY ELEMENT

afterward, it would be a ROM memory and would serve the same purpose as the gating network in Fig. 6.23.

Figure 6.24(*a*) shows a scheme for implementing Table 6.2 by using a decoder network with four inputs X_1, X_2, X_3, and X_4 and four OR gates. With a given input combination (or address), a single output line from the decoder will be high. Let us assume that the input value is $X_1 = 0$, $X_2 = 1$, $X_3 = 1$, $X_4 = 1$. This corresponds to 0111 on the decoder output in Fig. 6.24(*a*). OR gates are connected to this line for outputs which are 1s, and no OR gates are placed where 0s are to appear. Thus for the input 0111 we have a single OR gate connected to output line Z_2, since the desired output is 0100. Similarly, for the input 0110 we connect OR gates to Z_2 and Z_4 since the output is to be 0101.

The entire scheme outlined above realizes the ROM with OR gates. By using LSI techniques, arrays of this sort can be inexpensively fabricated in small containers at low prices. The 4K-word 8-bit memories are of about average size for the large-scale integration ROMs; these memories effectively store 32K bits in all.

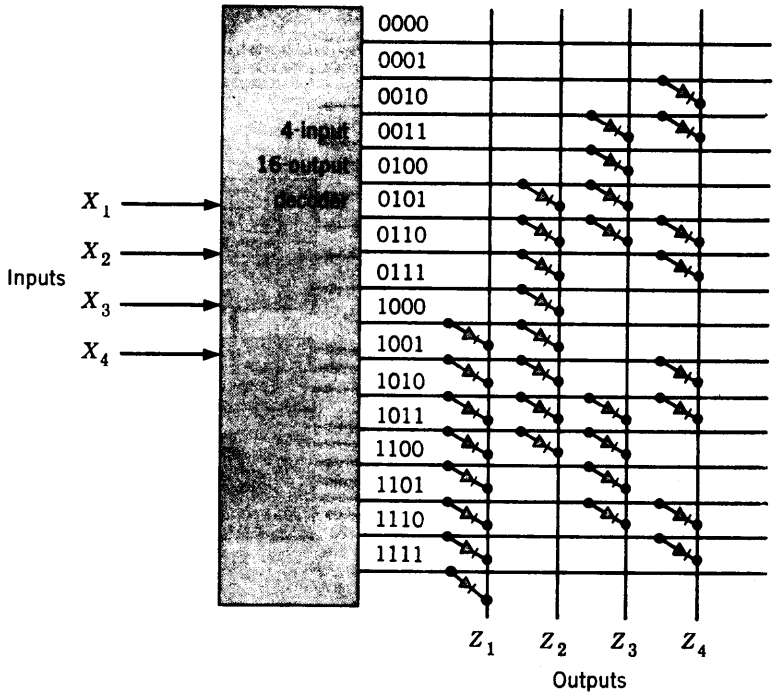
Figure 6.24(*b*) shows how Fig. 6.24(*a*) might be implemented by using diodes to form the OR gates. The diodes OR the inputs to which they are connected. The manufacturer of a conventional ROM will have an IC layout in which diodes can be placed between every input and output line; then only the specified diodes would be actually used, the remainder being omitted during manufacture.



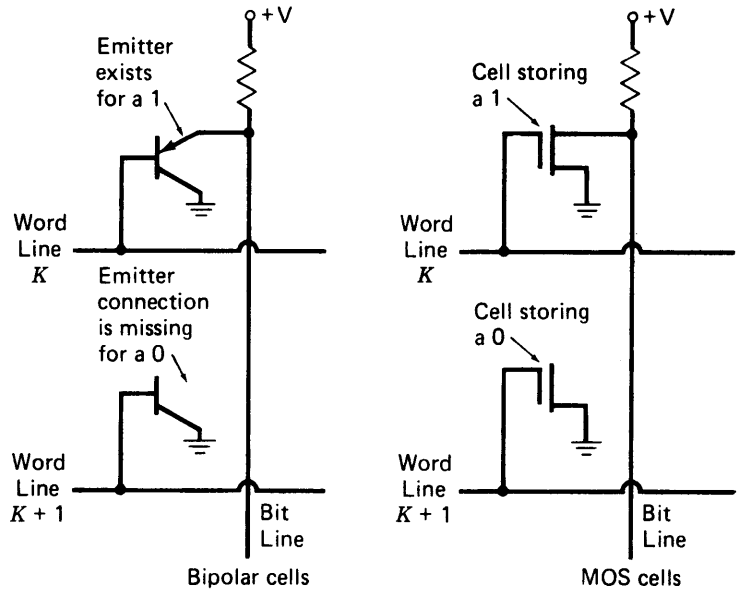
(*a*)

FIGURE 6.24

(*a*) Read-only memory layout. (*b*) Diode ROM. (*c*) ROM cells using MOS and bipolar transistors.


4-input
16-output
decoder
Inputs
X_1
X_2
X_3
X_4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
Z_1
Z_2
Z_3
Z_4
Outputs

(*b*)


+V
Emitter
exists
for a 1
Word
Line
K
Emitter
connection
is missing
for a 0
Word
Line
$K + 1$
Bit
Line
Bipolar cells
+V
Cell storing
a 1
Word
Line
K
Cell storing
a 0
Word
Line
$K + 1$
Bit
Line
MOS cells

(*c*)

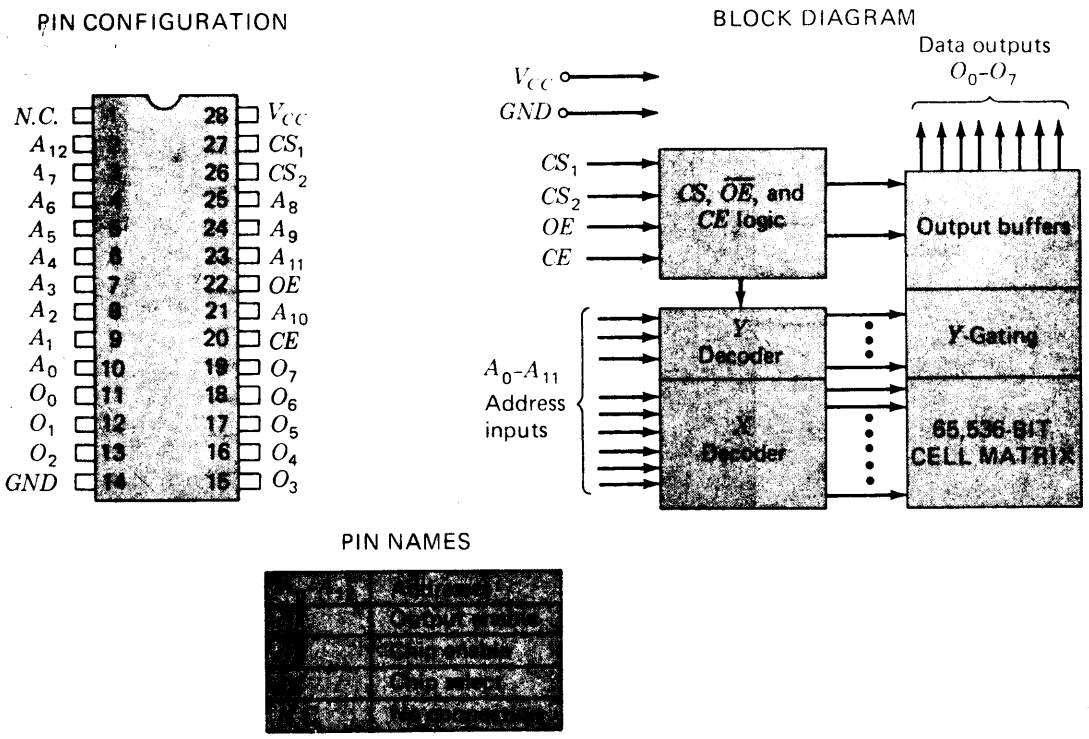


FIGURE 6.25

A 64K ROM. (*Intel Corp.*)

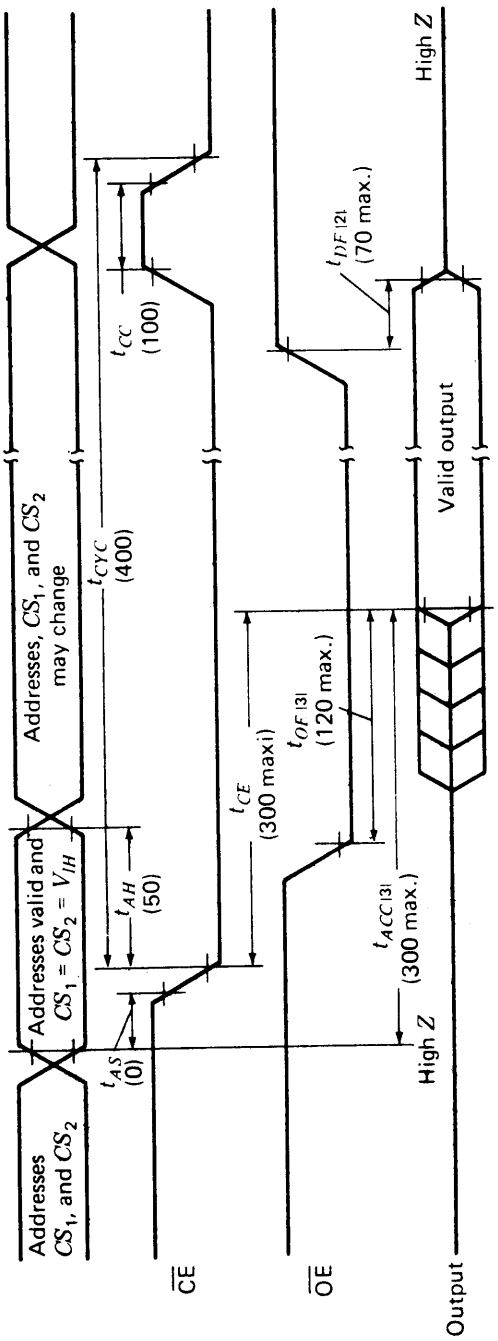
The diodes in Fig. 6.24(*b*) are often transistors, and the manufacturing processes are of various types. Both one- and two-dimensional selection is used in ROMs. Generally MOS ROMs have 10- to 200-ns access times; access times of bipolar memories range from 10 to 100 ns.

When a ROM is constructed so that the user can electrically (or by using other techniques) write in the contents of the memory, the memory is called a *programmable ROM*, or *PROM*.[9] Often a scheme is used in which a memory chip is delivered with 1s in every position, but 0s can be introduced at given positions by placing an address on the input lines and then raising each output line which is to be a 0 to a specified voltage, thus destroying a connection to the selected cell. (Sometimes the memories contain all 0s, and 1s are written in by the user.) Devices are also manufactured which program PROMs by reading paper tapes, magnetic tapes, punched cards, etc., and placing their contents into the PROM.

Custom ROM manufacturers provide forms in which a user can fill 1s and 0s and then they will produce a custom-made mask and LSI chips which will realize the memory contents specified by the user. A single chip may cost more for such a memory, but with large production runs generally the cost per chip is less. These devices come with up to 256K bits per IC package.

Figure 6.25 shows a block diagram of a 64K-bit MOS memory which is organized as a 4096-word 8-bit-per-word memory. The user places the address on

[9]These memories are also called *field-programmable ROMs*.



Notes
1. All times shown in parentheses are minimum times and are nonoseconds unless otherwise specified.
2. t_{DF} is specified from $\overline{OE}$ or $\overline{CE}$, whichever occurs first.
3. t_{ACC} may be delayed up to 180 ns after the falling edge of $\overline{CE}$ without impact on t_{ACC}.

FIGURE 6.25 (*Cont.*)

TABLE 6.4 CHARACTERISTICS OF READ-ONLY MEMORIES

MANUFACTURER AND PART NUMBER	ORGANIZATION	TYPE	ACCESS TIME, ns
Monolithic Memories 6280	1K × 8	ROM	100
National Semiconductor DM85529	1K × 8	ROM	90
Intel 2732	4K × 8	EPROM	200
TI 2532	4K × 8	EPROM	400
NITRON NC7714	256K × 4	EAROM	450
GI ER2805	2K × 4	EAROM	500
RCA 1842	256K × 8	EEPROM	250
NEC D458	1K × 8	EEPROM	450

the 13 input lines A_0 to A_{12} and then raises CS_1 and CS_2 ($\overline{CE}$ must be low). This will enable the output, and the desired word will appear on lines O_0 to O_7. This is a custom-made memory in which the desired memory contents are supplied to the manufacturer by the user on a form. Then the manufacturer makes a mask to create the desired bit patterns on an IC chip and manufactures ROMs with this pattern to order. The delay time for the memory is on the order of 75 ns.

When a ROM is manufactured so that the memory's contents can be set as desired by the user and the memory can later have the contents erased and new values written in, the ROM is said to be *erasable and reprogrammable* and is often called an *EPROM*.

For example, some memory chips are made with a transparent lid. Exposing the semiconductor chip (through the lid) to ultraviolet light[10] will erase the pattern on the chip, and a new pattern can be written in electrically. This can be repeated as often as desired.

Table 6.4 shows characteristics of some bipolar ROMs and of several electrically programmable ROMs (EPROMs) which can have their contents erased by exposure to ultraviolet light and can be programmed (written into) by placing designated voltages on inputs. Also shown in Table 6.4 are some characteristics for electrically alterable ROMs (EAROMs) which can have selected contents rewritten while in place in a circuit by means of properly applied input voltages and EEPROMs (electrically erasable programmable ROMs). The difference between EAROMs and EEPROMs is that EAROMs can have their contents altered while in place in the circuit and EEPROMs must be removed to be erased and rewritten.

Several companies make devices for programming PROMs and EPROMs. Some of these devices are operated from a keyboard, others from tape, and still others from external inputs such as microprocessors.

When an EPROM, EEPROM, EAPROM, or a PROM chip is to be programmed, there is generally a write enable to be raised, which makes output lines able to accept data. Then address lines are set to the location to be written into. Next, for some chips, the output lines to have 1s are raised to high voltages (or a sequence of large-amplitude pulses are applied to them); or, for some chips, the normal logic levels are placed on the output lines, and a special program input is placed with a sequence of high-voltage (25-V) pulses. In either case, each memory

[10]Standard ultraviolet lamps can be used to erase the memory. About a 20- to 30-min exposure is required.

location must be written into by setting the address lines and then writing the desired contents into the output lines. Applying an erase (for example, applying an ultraviolet light to the lid for some specified time) generally erases all the contents of the memory except for EAROMs, in which the individual locations can be changed.

MAGNETIC DISK MEMORIES

6.10 The magnetic disk memory provides large storage capabilities with moderate operating speeds. Quite a number of different types of magnetic disk memories are now on the market. Although they differ in specific details, all are based on the same principles of operation.

Magnetic disk memories store information on one or more circular platters, or disks, which are continually spinning. These rotating disks are coated with a magnetic material and stacked with space between them (refer to Fig. 6.26). Information is recorded on the surface of the rotating disks by magnetic heads such

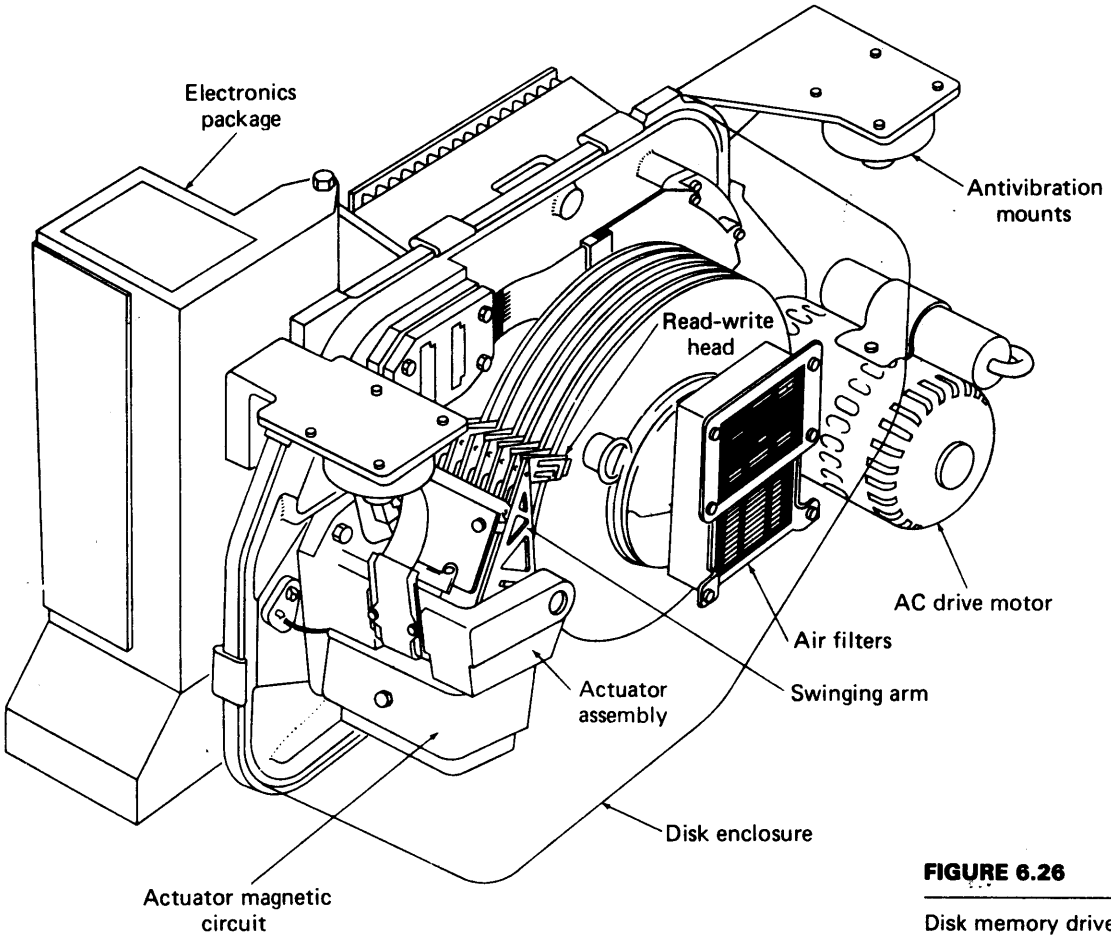


FIGURE 6.26

Disk memory drive.

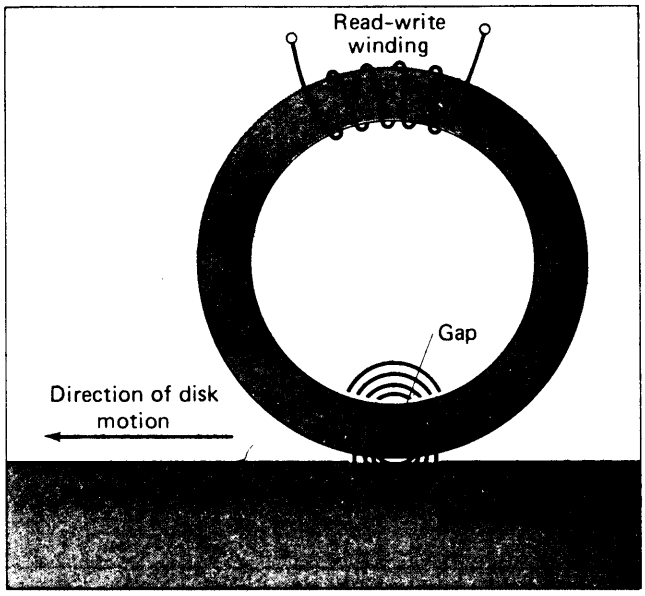


FIGURE 6.27

Read-write head for a magnetic disk memory. *Note*: Passing a film of magnetic material by a head while passing current through the read-write winding will magnetize the surface of the material, forming simple magnets. Passing this same area by a head at a later time will induce currents in the read-write winding.

as that in Fig. 6.27. These heads are mounted on *access arms*. (Information is recorded in bands rather than on a spiral.) Each band of information on a given disk is called a *track*. On one side of a typical disk there can be several thousand data tracks. Bits are recorded along a track at a density of 500 to 24,000 bits/in. In some systems the outer tracks contain more bits than the inner tracks, because the circumference of an outer track is greater than that of an inner track; but many disks have the same number of bits around each track, The rotational speed of the disks varies, of course, with the manufacturer, but typical speeds are on the order of 3600 rpm.

Since each disk contains a number of tracks of information and there may be several disks in a given memory, several techniques have evolved for placing a magnetic read-write head in the correct position on a selected track. Since the same head is generally used for reading and for writing, the problem becomes that of placing this head accurately and quickly on the selected track.

There are two basic types of disk head placement systems. In the first type, the heads are fixed in position on each track. These are called *fixed-head systems*. In the second kind of system, one or more pairs of read-write heads exist for each pair of adjacent disk surfaces (because information is generally written on both the top and the bottom of each disk). These read-write heads are mounted on arms which can be moved in and out. These are called *movable-head systems*.[11]

The positioning of the heads by means of the mechanical movement of arms is a difficult and tricky business, particularly since the tracks are often recorded

[11]A few systems have been made with only one pair of read-write heads for the entire memory. In these systems, the two recording heads are positioned on an arm which first is moved between the correct pair of disks and then selects the correct surface of the adjacent surfaces (again because information is written on both the top and the bottom of each disk). The read-write head is finally placed upon the correct selected track.

thousandths of an inch apart on the disk. Clearly disk memories with many heads can locate and record on or read from a selected track faster than the ones with only a few heads, since the amount of mechanical movement before the track is reached will be less for the multihead system.

The total time it takes to begin reading selected data or to begin writing on a selected track in a particular place is called the *access time*.

The time it takes to position a head on the selected track is called the *seek time*, and it is generally several milliseconds. The other delay in locating selected data is the *latency*, or rotational, delay, which is the time required for the desired data to reach the magnetic head once the head is positioned. Thus the total access time for a disk is the seek time plus the latency.

For a rotational speed of 2400 rpm, for example, latency is a maximum of 25 ms and averages 12.5 ms. Latency represents a lower limit in systems using fixed heads. As a result, for minimum access time, fixed heads are used. In fixed-head systems, typically heads are arranged in groups of eight or nine, perhaps including a spare, and are carefully aligned in fixed positions with respect to the disk. Although head spacing in each group is typically 8 to 16 per inch, track densities of 30 to 60 per inch can be achieved by interlacing groups.

Although they are faster, fixed-head systems provide less storage capacity than moving-head systems with comparable disk recording areas because the moving-head systems have more tracks per inch. Further, the large number of heads required can increase cost for a given capacity, and fixed-head systems are used only in special applications.

An important advantage of moving magnetic heads concerns their alignment with very closely spaced data tracks. Although track spacing is limited by *crosstalk* between adjacent tracks and mechanical tolerances, spacing of 3 mils between adjacent tracks is common. Further, track widths of approximately 0.5 mil are consistent with head-positioning accuracies of 0.25 mil.

The read-write heads used on magnetic disk memories are almost invariably of the type called *flying heads*. A simplified diagram of a flying head is shown in Fig. 6.28. When a disk rotates at a high speed, a thin but resilient boundary layer of air rotates with the disk. The head is shaped so that it rides on this layer of rotating air, which causes the disk to maintain separation from the head, thus

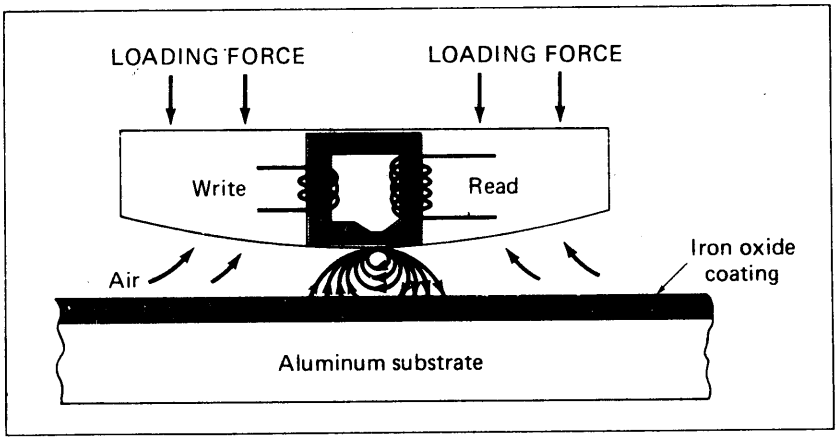


FIGURE 6.28

Flying head in a disk memory. *Note*: The loading force is applied by the access arm pressing down, using a springlike mechanism. The flying height is determined by the amount of loading force.

preventing wear on the disk surface. In effect, the layer of air rotating with the disk acts like a spring with a stiffness exceeding several thousand pounds per inch, thus forcing the head away from the surface of the disk. To force the head into correct proximity with the disk, a number of mechanisms have been used, but current memories use a spring-loading system based on a flexible arm on which the read-write head is mounted. These floating heads are normally about 10 microinches (μin.) from the disk surface.

There are many sizes and speeds of disk memories. Because of the large market for these memories and the seemingly infinite variety of configurations in which they can be manufactured, the system user is afforded a wide choice.

Some disk memories have changeable disk packs (or modules[12]). Each disk pack contains a set of disks which rotate together. These disk packs can be changed and are inexpensive enough that a user can store records or programs on them and keep them in files. The cost per bit is much higher than for tape, however, and so disk packs are used most in applications requiring high operating speeds.

In the mid-1970s, disk units with changeable packs, or modules, were the most used devices. At that time, however, an IBM unit with fixed disks, the 3350,[13] brought in a new disk technology with greater recording density, more tracks per data surface, and a faster transfer rate. This disk technology is referred to as *Winchester* technology and features a low head loading force [10 versus 300 grams (g) on earlier devices] and a low-mass head. Also, since these disks are generally not changeable, alignment and other tracking problems are reduced. Further, the disks are lubricated so that the lightweight heads can "crash" without damage, and they are stored with power off against the disk. Winchester disk drives require clean air and are sealed and have circulating air which is filtered. This is because the heads float so close to the disk (10 to 20 μin.). A particle of smoke, for instance, is 50 μin., and if one got between the disk and head, it could cause an error. The original Winchester disk drives were used for large storage systems. However, the technology was quickly picked up by manufacturers of smaller drives, and so now Winchester drives are made by many companies. Winchester fixed-disk systems are the most popular with minicomputer and microcomputer systems as well as with large systems. Some data on Winchester drives is given in Tables 6.5 and 6.6.

The Winchester disk drives manufactured by most concerns use 14-, 8-, $5\frac{1}{4}$-, or 3.5-in. disks. Capacities range from 5 megabytes (Mbytes) for several 3.5-in. drives to 2500 Mbytes for the Storage Technologies 8380 and the IBM 3380. Bit densities along a track range from 200 to 300 bits/in. for the less expensive drives up to 20,000 bits/in. for large thin-film disks. Track densities vary from 400 (for stepper motor access arms) to 1000 per inch for voice-coil-type access arms.

Some Winchester drives have both fixed disk(s) and a removable cartridge.

Some general data on Winchester-type tape drives follow. The head-to-disk spacing is generally 10 to 20 μin. The magnetic material on the surface of the

[12]A module is a disk pack which has the read-write heads and positioning arms all packaged together with the disks. These modules are costly, but enable a higher performance system.

[13]The 3350 improved an earlier Winchester technology with changeable disk packs (the 3340).

TABLE 6.5 LARGE FIXED-DISK DRIVE CHARACTERISTICS FOR CDC 9776†

CHARACTERISTIC	SPECIFICATION
Number of spindles per cabinet	2
Capacity per spindle	400 Mbytes (movable head) 1.72 Mbytes (fixed head)
Data rate	9.6 MHz
Average access time	25 ms
Rotational speed	3600 rpm

†Courtesy CDC Corp.

TABLE 6.6 TYPICAL FIXED-DISK WINCHESTER DRIVE CHARACTERISTICS (FOR SMALL TO MEDIUM SYSTEMS)

CHARACTERISTIC	SPECIFICATION
Number of disks	1 to 4
Data surfaces	1 to 7
Bit density	8400 bits/in.
Track density	500 tracks/in.
Tracks per surface	1200
Surface capacity	7.6 Mbytes
Rotational speed	3600 rpm
Four-disk capacity	30.2 Mbytes

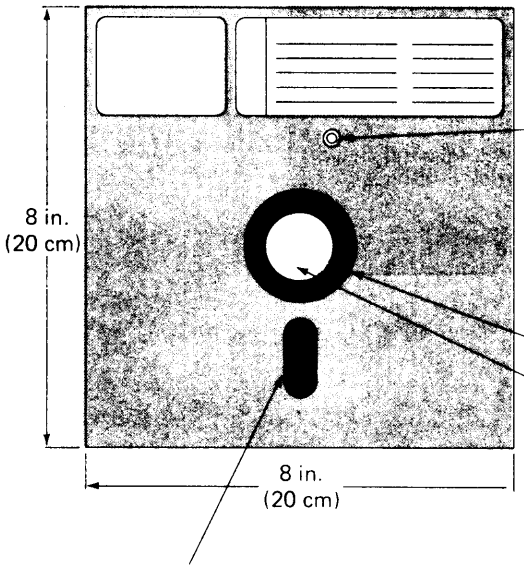
disks is about 0.075 in. thick except for some new thin-film disks. The access arms on inexpensive drives are positioned by using stepper motors and on more expensive drives by using servo (voice-coil-like) motors. The head type plays a large role in determining bits per inch along a track. Most heads are ferrite, but thin-film heads fabricated by using vacuum disposition through masks or through photolithography increase the number of bits per inch possible.

Because of the relative inexpensiveness per bit of information stored in disk memories and because of the relatively low access times and the high transfer rates attainable when data are read from or written into a disk file, magnetic disk memories have become one of the most important storage devices in modern digital computers.

FLEXIBLE-DISK STORAGE SYSTEMS—THE FLOPPY DISK

6.11 An innovation in disk storage, originally developed at IBM, uses a flexible, "floppy" disk with a plastic base in place of the more conventional rigid, metal-based disk. This storage medium is approximately the size and shape of a 45-rpm record (or smaller) and can be "plugged in" about as easily as a tape cartridge.

The floppy disks are changeable, and each disk comes in an envelope, as shown in Fig. 6.29. The disks are mounted on the disk drive with the envelope in place, and information is written and read through an aperture in the envelope. (A



Index hole
The outer circle shows a hole in the jacket; the inner circle shows the index hole in the disk. When these two holes are aligned as the disk revolves during data processing operations, a beam of light shining on one side of the diskette is sensed from the other side and used for timing functions.

Drive access opening in jacket

Drive spindle hole in disk
After the diskette has been placed in the machine and the disk drive spindle has been inserted into the drive spindle hole in the disk, the drive mechanism clamps onto a portion of the disk exposed by the drive access opening in the jacket.

Pressure pad slot
(A similar opening on the other side of the diskette is called the head slot.)
The head slot exposes the recording surface of the disk as the disk turns in its jacket in the machine. The data recording and sensing unit of the disk drive, which is called a *read-write head* and is similar to the record/playback head in a tape recorder, moves to specified positions along the length of the slot. Moving to a specified position is called *accessing a track*. (Data are recorded only on the side of the diskette that contains the head slot.)

FIGURE 6.29

Floppy disk with envelope.

few systems require removing the envelope.) Several manufacturers now provide complete systems for well under $300 for use with these disks. Convenience in use and low prices have broadened the use of floppy-disk memories in many applications.

On most floppy-disk drives, the read-write head assembly is in actual physical contact with the recording material. (For increased life, head contact is generally maintained only during reading or writing.) Track life on a diskette is generally on the order of 3 to 5 million contact revolutions. There are three standard sizes for floppy disks. The original-size flexible disk is enclosed in an 8-in. square jacket, and the disk has a diameter of 7.88 in. The recording surface is a 100-in.-thick layer of magnetic oxide on a 0.003-in.-thick polyester substrate. The jacket gives handling protection; in addition, it has a special liner that provides a wiping action to remove wear products and other dirt which would be abrasive to the media and head if left on the surface.

In the original IBM version, there is a single 0.100-in.-diameter index hole 1.5 in. from the center of the disk to indicate the start of a track. A written track