Chapter 9

Combinational Circuits

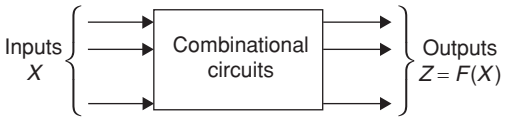
LEARNING OBJECTIVES

After reading this chapter, you will be able to understand:

- Combinational logic design
- Arithmetic circuits
- Code converters
- Decoder
- Combinational logic implementation
- Encoders
- Multiplexer
- Basic gates by using MUX
- De-multiplexer
- Memory and programmable logic
- Random access memory

Introduction

Combinational logic is a type of logic circuit whose output is a function of the present input only.



Combinational Logic Design

The design of combinational circuit starts from the problem, statement and ends with a gate–level circuit diagram.

The design procedure involves the following steps:

1. Determining the number of input variables and output variables required, from the specifications.
2. Assigning the letter symbols for input and output.
3. Deriving the truth table that defines the required relationship between input and output.
4. Obtain the simplified Boolean function for each output by using K-map or algebraic relations.
5. Drawing the logic diagram for simplified expressions.

We will discuss combinational circuits under the following categories:

- Arithmetic circuits
- Code converters
- Data processing circuits

Arithmetic Circuits

Arithmetic circuits are the circuits that perform arithmetic operation. The most basic arithmetic operation is addition.

Half Adder

Addition is an arithmetic operation, and here to implement addition in digital circuits we have to implement by logical gates. So the addition of binary numbers will be represented by the logical expressions. Half adder is an arithmetic circuit which performs the addition of two binary bits, and the result is viewed in two output—sum and carry.

The sum 'S' is the X-OR of 'A' and 'B' where A and B are inputs.

$$\therefore \quad S = A\overline{B} + B\overline{A} = A \oplus B$$

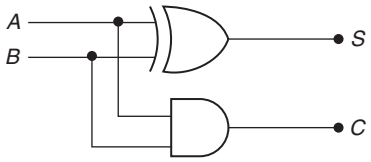
The carry 'C' is the AND of A and B.

$$\therefore C = AB$$

Truth Table

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

So, half adder can be realized by using one X-OR gate and one AND gate.



Half adder can also be realized by universal logic, such as only NAND gate or only NOR gate as given below.

NAND logic

$$S = A\bar{B} + \bar{A}B$$
$$= A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B}$$
$$= A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B})$$
$$= \overline{\overline{A\overline{AB}} \cdot \overline{B\overline{AB}}}$$
$$C = AB = \overline{\overline{A \cdot B}}$$

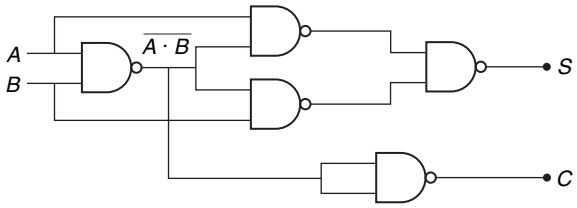


Figure 1 Half adder using NAND logic

NOR logic

$$S = A \cdot \bar{B} + \bar{A}B$$
$$= A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B}$$
$$= A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B})$$
$$= (A + B)(\bar{A} + \bar{B})$$
$$= \overline{\overline{A + B} + (\overline{\bar{A} + \bar{B}})}$$
$$C = A \cdot B = \overline{\overline{A \cdot B}} = \overline{\bar{A} + \bar{B}}$$

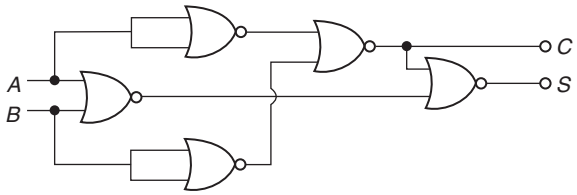


Figure 2 Half adder using NOR logic

Full Adder

Full adder is an arithmetic circuit that performs addition of two bits with carry input. The result of full adder is given by two outputs—sum and carry. The full adder circuit is used in parallel adder circuit as well as in serial adder circuit.

For full adder, if total number of 1's is odd at input lines, the sum output is equal to logic 1, and if total number of 1's at input lines are more than or equal to 2, then the carry output is logic 1.

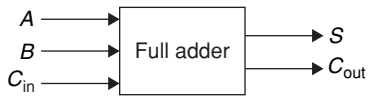


Figure 3 Block diagram

Truth Table

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\overline{C_{in}} + A\bar{B}\overline{C_{in}} + ABC_{in}$$
$$= A \oplus B \oplus C_{in}$$
$$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\overline{C_{in}} + ABC_{in}$$
$$= AB + (A \oplus B)\, C_{in}$$
$$= AB + A\, C_{in} + B\, C_{in}$$

Full adder can also be realized using universal logic gates, i.e., either only NAND gates or only NOR gates as explained below.

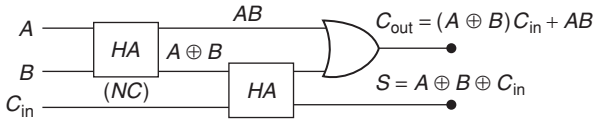


Figure 4 Block diagram of full adder, by using half adder

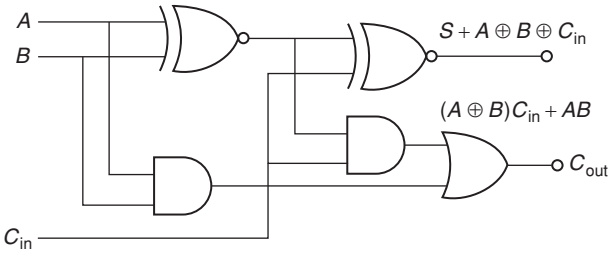


Figure 5 Logic diagram of full adder

NAND logic

$$A \oplus B = \overline{\overline{A\overline{AB}}\ \ \overline{B\overline{AB}}}$$

So $A \oplus B \oplus C_{in}$

Let $A \oplus B = x$ then $s = \overline{\overline{X \cdot \overline{XC_{in}}}\ \ \overline{C_{in}\overline{X \cdot C_{in}}}}$

$$= X \oplus C_{in}$$

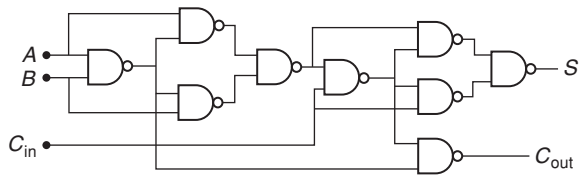


Figure 6 Logic diagram of a full adder using only 2-input NAND gates

NOR logic

Full subtractor outputs.

Sum = $a \oplus b \oplus c$, carry = $ab + bc + ac$ are self dual functions.

[$\because$ A function is called as self dual if its dual is same as the function itself $f^D = f$].

For self dual functions, the number of NAND gates are same as number of NOR gates.

By taking the dual for above NAND gate implementation, all gates will become NOR gates, and the output is dual of the sum and carry, but they are self dual ($f^D = f$).

So, output remain same, and only 9 NOR gates are required for full adder, structure similar to NAND gate circuit.

Half Subtractor

Half subtractor is an arithmetic circuit which performs subtraction of one bit (subtrahend) from other bit (minuend), and the result gives difference and borrow each of one bit. The borrow output is logic 1 only if there is any subtraction of 1 from 0.

When a bit 'B' is subtracted from another bit 'A', a difference bit (d) and a borrow bit (b) result according to the rule given below.

Truth Table

A	B	d	b
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

$$d = A\bar{B} + B\bar{A}$$
$$= A \oplus B$$
$$b = \bar{A}B$$

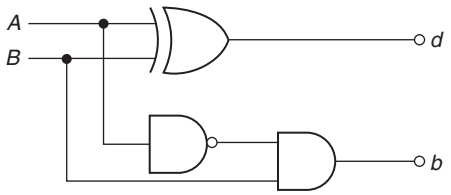


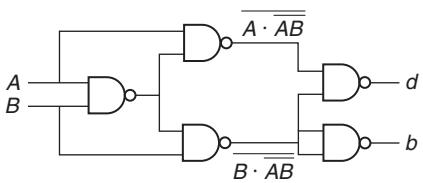
Figure 7 Logic diagram of a half subtractor

A half subtractor can also realized using universal logic either using only NAND gates or only NOR gates as explained below.

NAND logic

$$d = A \oplus B$$
$$= \overline{\overline{A\overline{AB}} \cdot \overline{B\overline{AB}}}$$

$$b = \bar{A}B = B(\bar{A} + \bar{B}) = B(\overline{AB}) = \overline{\overline{B \cdot \overline{AB}}}$$



NOR logic

$$d = A \oplus B$$
$$= A\bar{B} + \bar{A}B$$
$$= A\bar{B} + B\bar{B} + \bar{A}B + A\bar{A}$$
$$= \bar{B}(A + B) + \bar{A}(A + B)$$
$$= \overline{B + \overline{A + B}} + \overline{A + \overline{A + B}}$$
$$b = \bar{A}B$$
$$= \bar{A}(A + B)$$
$$= \overline{\overline{\bar{A}(A + B)}}$$
$$= \overline{A + \overline{(A + B)}}$$

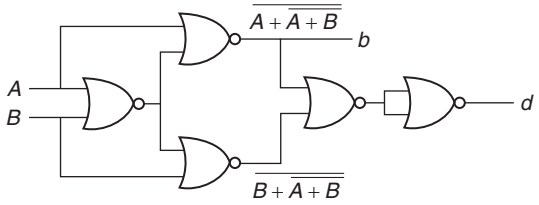


Figure 8 Logic diagram of half subtractor using NOR gate

Full Subtractor

Full subtractor is an arithmetic circuit similar to half subtractor but it performs subtraction with borrow, it involves subtraction of 3-bits—minuend, subtrahend and borrow-in, and two outputs—difference and borrow. The subtraction of 1 from 0 results in borrow to become logic 1. The presence of odd number of 1's at input lines make difference as logic 1.

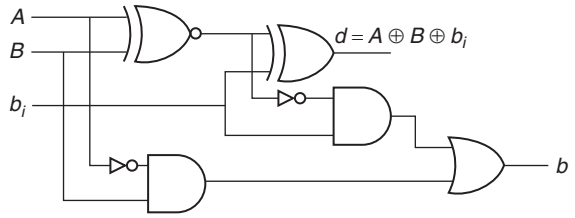
Truth Table

A	B	b_i	d	b
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$d = \bar{A}\bar{B}b_i + \bar{A}B\bar{b_i} + A\bar{B}\bar{b_i} + ABb_i$$
$$= b_i(AB + \bar{A}\bar{B}) + \bar{b_i}(A\bar{B} + \bar{A}B)$$
$$= b_i(\overline{A \oplus B}) + \bar{b_i}(A \oplus B)$$
$$= A \oplus B \oplus b_i$$

and

$$b = \bar{A}\bar{B}b_i + \bar{A}B\bar{b_i} + \bar{A}Bb_i$$
$$= \bar{A}B + (\overline{A \oplus B})b_i$$



NAND logic

$$d = A \oplus B \oplus b_i$$
$$= \overline{(A \oplus B)\overline{(A \oplus B)}b_i}\,\overline{b_i(A \oplus B)b_i},$$
$$b = \bar{A}B + b_i\overline{(A \oplus B)}$$
$$= \overline{\bar{A}B + b_i\overline{(A \oplus B)}}$$
$$= \overline{\bar{A}Bb_i}\,\overline{(A \oplus B)}$$
$$= \overline{B(\bar{A} + \bar{B})b_i}\left[\overline{\bar{b_i} + (A \oplus B)}\right]$$

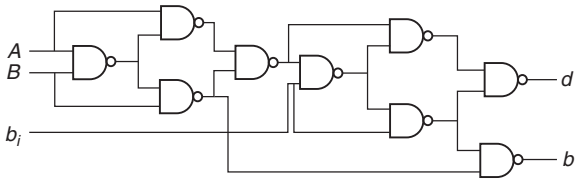


Figure 9 Logic diagram of a full subtractor using NAND logic

NOR logic

Outputs of full subtractor are also self dual in nature. Therefore, same circuit with all NAND gates replaced by NOR gates gives the NOR gate full subtractor. For this 9 NOR gates required.

Example 1: How many NAND gates are required for implementation of full adder and full subtractor?
(A) 11, 10 (B) 11, 11 (C) 9, 9 (D) 9, 10

Solution: (C)
From the circuit diagrams in the previous discussion, full adder requires 9 NAND gates, and full subtractor requires 9 NAND gates.

Binary Adder

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers.

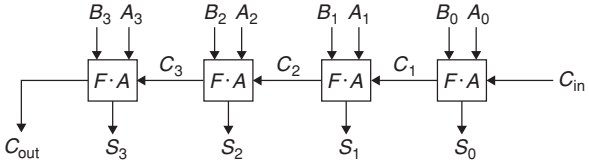


Figure 10 4-bit parallel adder

The output carry from each full adder is connected to the input carry of next full adder.

The bits are added with full adders, starting from the LSB, position, to form the sum bit and carry bit.

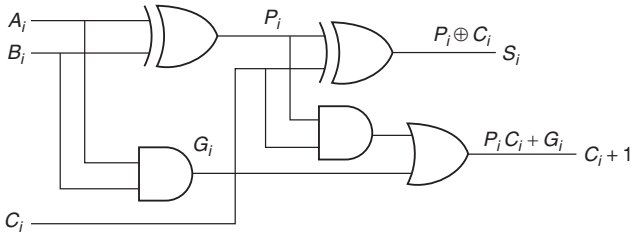
The longest propagation delay time in parallel adder is the time it takes the carry to propagate through the full adders.

For n-bit parallel adders consider t_{pds} is the propagation delay for sum of each full adder, and t_{pdc} is the propagation delay of carry.

The total time required to add all n-bits at the nth full adder is

$$T_S = t_{pds} + (n-1)t_{pdc}$$

So propagation delay increases with number of bits. To overcome this difficulty, we use look ahead carry adder. Look ahead carry adder is the fastest carry adder.



Consider the full adder circuit for ith stage, in parallel adder, with two binary variables A_i, B_i, input carry C_i are:

Carry propagate (P_i) and carry generate (G_i)

$$P_i = A_i \oplus B_i$$
$$G_i = A_i \cdot B_i$$

The output sum and carry can be expressed as

$$S_i = P_i \oplus C_i$$
$$C_{i+1} = P_i C_i + G_i$$

Now, the Boolean functions for each stage can be calculated as substitute $i = 0$

C_0 is input carry

$$C_1 = G_0 + P_0 C_0$$

Substitute $i = 1, 2 \ldots$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1 (G_0 + P_0C_0)$$
$$= G_1 + P_1G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2C_2 = G_2 + P_2 (G_1 + P_1G_0 + P_1 P_0C_0)$$
$$= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

Since the Boolean function for each output carry is expressed in SOP form, each function can be implemented with AND–OR form or two level NAND gates.

From the above equations, we can conclude that this circuit can perform addition in less time as C_3 does not have to wait for C_2 and C_1 to propagate: C_3, C_2, C_1 can have equal time delays.

The gain in speed of operation is achieved at the expense of additional complexity (hardware).

n-bit Comparator

The comparison of two numbers is an operation that determines whether one number is greater than, less than, or equal to the other number.

A magnitude comparator is a combinational circuit that compares two input numbers A and B, and specifies the output with three variables:

$$A > B, A = B, A < B$$

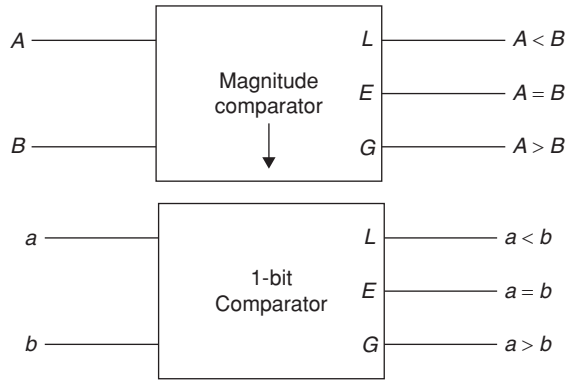


Figure 11 1-bit comparator will have only 1-bit input such as *a*, *b*.

a	b	a < b	a = b	a > b
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

By considering minterms for each output.

$$(a < b) = a'b$$
$$(a = b) = a'b' + ab = a \odot b$$
$$(a > b) = ab'$$

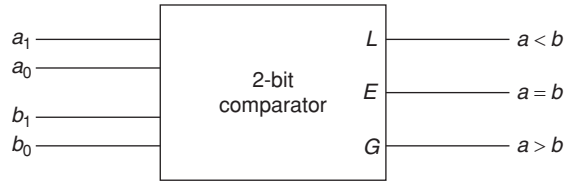
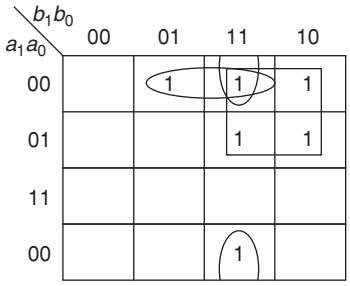


Figure 12 2-bit comparator will have 2-bit inputs, such as a_1 a_0 and b_1 b_0.

a_1	a_0	b_1	b_0	L a < b	E a = b	G a > b
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

$$(a < b) = \Sigma(1, 2, 3, 6, 7, 11)$$
$$(a > b) = \Sigma(4, 8, 9, 12, 13, 14)$$
$$(a = b) = \Sigma(0, 5, 10, 15)$$



$$a < b = a_1'a_0'b_0 + a_0'b_1b_0 + a_1'b_1$$
$$L = \overline{a_1}b_1 + (a_1 \odot b_1)\overline{a_0}b_0$$

Similarly, $a > b = a_0b_1'b_0' + a_1a_0b_0' + a_1b_1'$

$$G = a_1\overline{b_1} + (a_1 \odot b_1)a_0\overline{b_0}$$

$a = b$ is possible when $a_1 = b_1$, $a_0 = b_0$

So $(a = b) = (a_1 \odot b_1)(a_0 \odot b_0)$

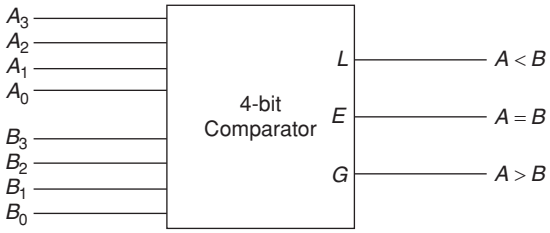


Figure 13 4-bit comparator will compare 2 input numbers each of 4-bits, as A_3 A_2 A_1 A_0 and B_3 B_2 B_1 B_0 ($A = B$) output will be 1 when each bit of input *A* is equal to corresponding bit in input *B*.

So we can write $(A = B) = (A_3 \odot B_3)\ (A_2 \odot B_2)\ (A_1 \odot B_1)\ (A_0 \odot B_0)$.

To determine whether A is greater or less than B, we inspect the relative magnitudes of pairs of significant bits, starting from MSB. If the two bits of a pair are equal, we compare the next lower significant pair of bits. The comparison continues until a pair of unequal bits is reached.

for $A < B$, $A = 0$, $B = 1$

for $A > B$, $A = 1$, $B = 0$

$$A < B = A_3'B_3 + (A_3 \odot B_3)\ A_2'B_2 + (A_3 \odot B_3)(A_2 \odot B_2) \times A_1'B_1 + (A_3 \odot B_3)(A_2 \odot B_2)\ (A_1 \odot B_1)\ A_0'B_0$$

$$A > B = A_3B_3' + (A_3 \odot B_3)\ A_2B_2' + (A_3 \odot B_3)(A_2 \odot B_2) \times A_1B_1' + (A_3 \odot B_3)(A_2 \odot B_2)\ (A_1 \odot B_1)\ A_0B_0'$$

4-bit comparator will have total 8 inputs and $2^8 = 256$ input combinations in truth table.

For 16 combinations $(A = B) = 1$, and for 120 combinations $A < B = 1$.

For remaining 120 combination $A > B = 1$.

Parity Bit Generator and Parity Bit Checker

When digital information is transmitted, it may not be received correctly by the receiver. To detect 1-bit error at receiver, we can use parity checker.

For detection of error, an extra bit known as parity bit is attached to each code word to make the number of 1's in the code even (in case of even parity) or odd (in case of odd parity).

For n-bit data, we use n-bit parity generator at the transmitter end. With 1 parity bit and n-bit data, total $n + 1$ bit will be transmitted. At the receiving end, $n + 1$ parity checker circuit will be used to check correctness of the data.

For even parity transmission, parity bit will be made 1 or 0 based on the data, so that total $n + 1$ bits will have even number of 1's. For example, if we want to transmit data 1011 by even parity transmission, then we will use parity bit as 1, so data will have even number of 1's, i.e., data transmitted will be 11011. At the receiving end, this data will be received and checked for even number of ones.

To transmit data $B_3B_2B_1B_0$ using even parity, we will transmit sequence $PB_3B_2B_1B_0$, where $P = B_3 \oplus B_2 \oplus B_1 \oplus B_0$ (equation for parity generator).

At the receiving end, we will check data received $PB_3B_2B_1B_0$ for error, $E = P \oplus B_3 \oplus B_2 \oplus B_1 \oplus B_0$ (equation for parity checker). If $E = 0$ (no error), or if $E = 1$ (1 bit error).

We use EX-OR gates for even parity generator/checker as EX-OR of bits gives output 1 if there are odd number of 1's else EX-OR output is 0.

Odd parity generator/checker is complement of even parity generator/checker. Odd parity circuits check for presence of odd number of 1's in data.

Code Converters

There are many situations where it is desired to convert from one code to another within a system. For example, the information from output of an analog to digital converter is often in gray code, before it can be processed in arithmetic unit, conversion to binary is required.

Let us consider simple example of 3-bit binary to gray code converter. This will have input lines supplied by binary codes and output lines must generate corresponding bit combination in gray code. The combination circuit code converter performs this transformation by means of logic gates.

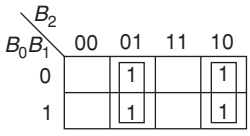
The output logic expression derived for code converter can be simplified by using the usual techniques, including don't-care, if any present. For example, *BCD* code uses only codes from 0000 to 1001, remaining combinations are treated as don't care combinations, similarly. EXS-3 uses only combinations from 0011 to 1100 remaining combinations are treated as don't care.

The relationship between the two codes is shown in the following truth table:

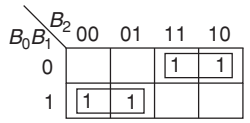
Decimal	B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

For conversion, we require to find out minimized functions of

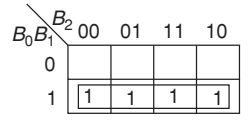
$$G_2(B_2, B_1, B_0) = \sum m(4, 5, 6, 7)$$
$$G_1(B_2, B_1, B_0) = \sum m(2, 3, 4, 5)$$
$$G_0(B_2, B_1, B_0) = \sum m(1, 2, 5, 6)$$



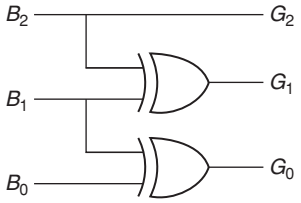
$$G_0(B_2, B_1, B_0) = B'_1B_0 + B_1B'_0 = B_1 \oplus B_0$$



$$G_1(B_2, B_1, B_0) = B'_1B_2 + B_1B'_2 = B_2 \oplus B_1$$



$$G_2(B_2, B_1, B_0) = B_2$$



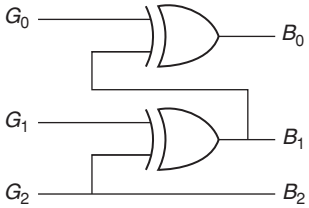
In similar fashion, we can derive n-bit binary to gray code conversion as

$$G_n = B_n$$
$$G_{n-1} = B_{n-1} \oplus B_n$$
$$G_{i-1} = B_{i-1} \oplus B_i$$

Thus conversion can be implemented by $n - 1$ X-OR gates for n-bits.

For reverse conversion of gray to binary, by following similar standard principle of conversion, we will get

$$B_0 = G_0 \oplus G_1 \oplus G_2, B_1 = G_1 \oplus G_2, B_2 = G_2$$



In general for n-bit gray to binary code conversion

$$B_i = G_n \oplus G_{n-1} \oplus G_{n-2} \ldots \oplus G_{i-1} \oplus G_i$$

$B_n = G_n$ (MSB is same in gray and binary). It also requires $n-1$ X-OR gates for n-bits.

Example 2: Design 84-2-1 to XS-3 code converter

Solution: Both 84-2-1 and XS-3 are *BCD* codes, each needs 4-bits to represent. The following table gives the relation between these codes. 84-2-1 is a weighted code, i.e., each position will have weight as specified. XS-3 is non weighted code; the binary code is three more than the digit in decimal.

Decimal	84-2-1 $B_3B_2B_1B_0$	XS-3 $X_3X_2X_1X_0$
0	0000	0011
1	0111	0100
2	0110	0101
3	0101	0110
4	0100	0111
5	1011	1000
6	1010	1001
7	1001	1010
8	1000	1011
9	1111	1100

We will consider minterm don't care combinations as 1, 2, 3, 12, 13, 14. For these combinations 84-2-1 code will not exist, and the remaining minimum terms can be found from truth table.

$$X_0(B_3, B_2, B_1, B_0) = \sum m(0, 4, 6, 8, 10) + \sum \Phi(1, 2, 3, 12, 13, 14) = \overline{B_0}$$

$$X_1(B_3, B_2, B_1, B_0) = \sum m(0, 4, 5, 8, 9, 15) + \sum \Phi(1, 2, 3, 12, 13, 14) = \overline{B_1}$$

$$X_2(B_3, B_2, B_1, B_0) = \sum m(4, 5, 6, 7, 15) + \sum \Phi(1, 2, 3, 12, 13, 14) = B_2$$

$$X_3(B_3, B_2, B_1, B_0) = \sum m(8, 9, 10, 11, 15) + \sum \Phi(1, 2, 3, 12, 13, 14) = B_3$$

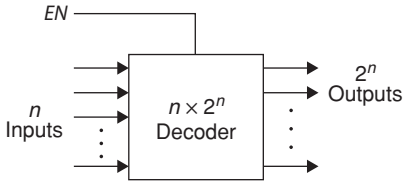
Decoder

A binary code of n-bits is capable of representing up to 2^n elements of distinct elements of coded information.

The three inputs are decoded into eight outputs, each representing one of the minterms of the three input variables.

A decoder is a combinational circuit that converts binary information from n input lines to a maximum 2^n unique output lines.

A binary decoder will have n inputs and 2^n outputs.



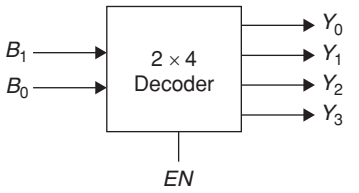


Figure 14 2 × 4 decoder

Truth Table

EN	B_1	B_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

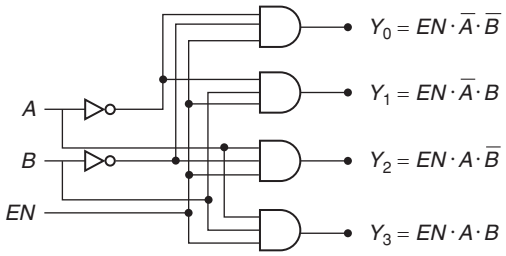
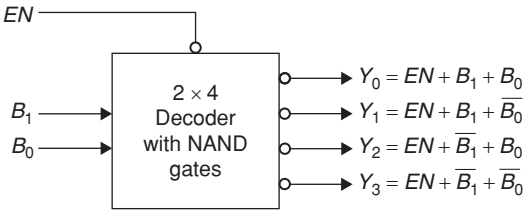


Figure 15 2 × 4 decoder

Decoder outputs are implemented by AND gates, but realization of AND gates at circuit level is done by the NAND gates (universal gates). So, the decoders available in IC form are implemented with NAND gates, i.e., the outputs are in complemented form and outputs are maximum terms of the inputs rather than minimum terms of inputs as in AND gate decoders.

Furthermore, decoders include one or more enable inputs to control the circuit operation. Enable can be either active low/high input.

Active low 2 × 4 decoder:



Truth Table

EN	B_1	B_0	Y_3	Y_2	Y_1	Y_0
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

The block diagram shown here is 2 × 4. Decoder with active low output and active low enable input.

The logic diagram is similar to the previous 2 × 4 decoder, except, all AND gates are replaced by NAND gates and *EN* will have inverter, *EN* is connected to all NAND inputs, as $\overline{EN}$ is active low input for this circuit.

The decoder is enabled when *EN* is equal to 0.

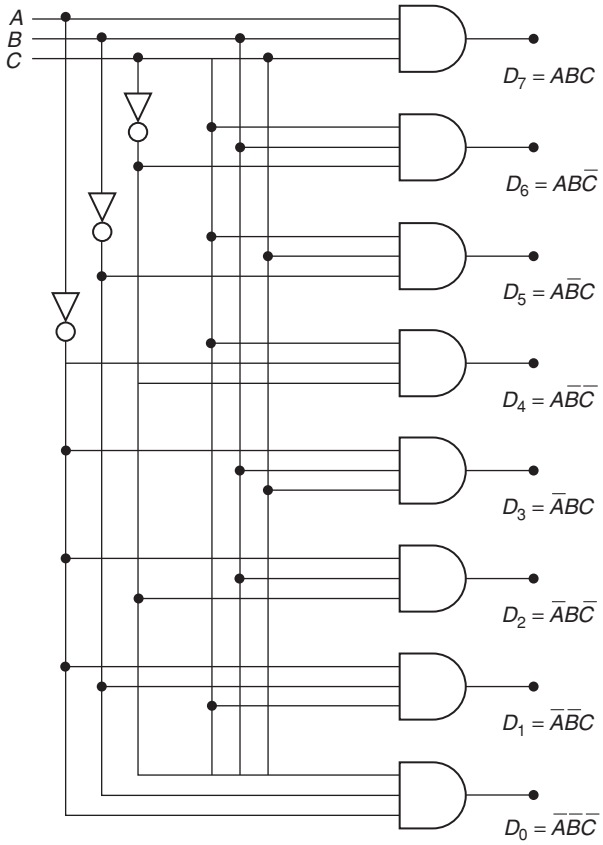
As shown in the truth table, only one output can be equal to 0 at any given time, all other outputs are equal to 1. The output whose value is equal to 0 represents the minimum term selected by inputs, enable.

Consider a 3 to 8 line decoder

Truth Table

Inputs			Outputs							
A	*B*	*C*	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

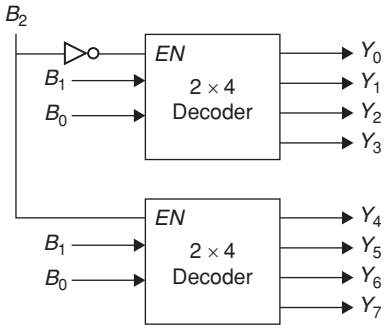
A 3 to 8 decoder has 3 input lines and 8 output lines, based on the combination of inputs applied for the 3 inputs, one of the 8 output lines will be made logic 1 as shown in the truth table. So, each output will have only one minimum term.



Designing High Order Decoders from Lower Order Decoders

Decoder with enable input can be connected together to form larger decoder circuit.

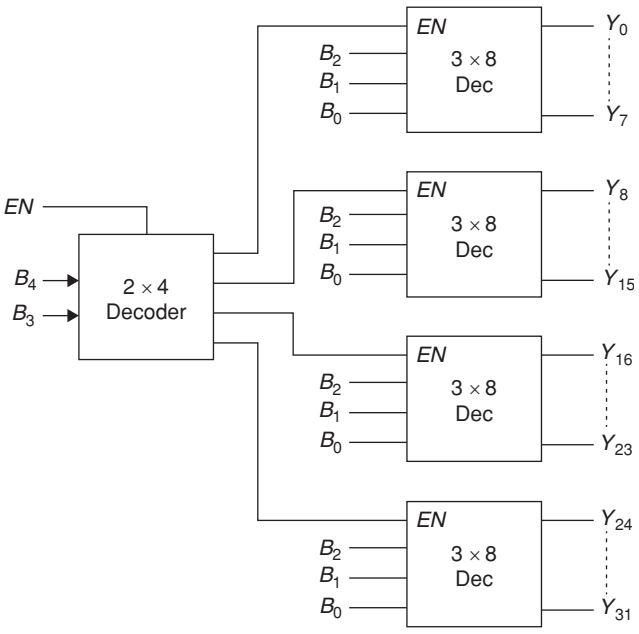
The following configuration shows 3 × 8 decoder with 2 × 4 decoders.



When $B_2 = 0$, top decoder is enabled and other is disabled, for 000 to 011 inputs, outputs are Y_0 to Y_3, respectively, and other outputs are 0.

For $B_2 = 1$, the enable conditions are reversed.

The bottom decoder outputs generates minterms 100 to 111, while the outputs of top decoder are all 0's. 5 × 32 decoder with 3 × 8 decoders, 2 × 4 decoders.



A 5 × 32 decoder will have 5 inputs $B_4\,B_3\,B_2\,B_1\,B_0$. A 3 × 8 decoder will have 8 outputs, so 5 × 32 requires four 3 × 8 decoders, and we need one of the 2 × 4 decoders to select one 3 × 8 decoders and the connections are as shown in the circuit above.

Combinational Logic Implementation

An $n \times 2^n$ decoder provides 2^n minimum terms of n input variables. Since any Boolean function can be expressed in sum of minimum terms form, a decoder that generates the

minimum terms of the function, together with an external OR gate that forms their logical sum, provides a hardware implementation of the function.

Similarly, any function with n inputs and m outputs can be implemented with $n \times 2^n$ decoders and m OR gates.

Example 3: Implement full adder circuit by using 2 × 4 decoder.

Sum = Σ(1, 2, 4, 7), Carry = Σ(3, 5, 6, 7)

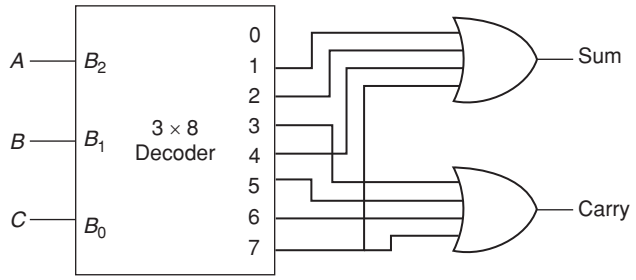


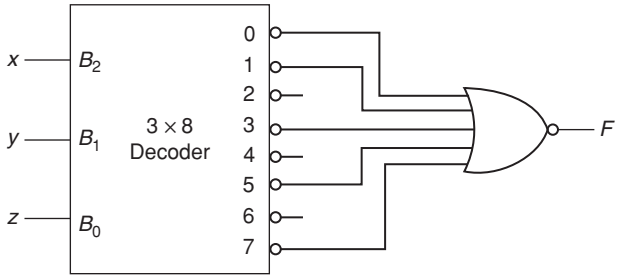
Figure 16 Implementation of full adder circuit with decoder

The 3 × 8 decoder generates the 8 minimum terms for A, B, and C. The OR gate for output sum forms the logical sum of minimum terms 1, 2, 4 and 7. The OR gate for output carry forms the logical sum of minimum terms 3, 5, 6 and 7.

Example 4: The minimized SOP form of output $F(x, y, z)$ is

(A) $x'y + z'$ (B) $x'y' + z'$

(C) $x'y' + z'$ (D) $x' + y'z$



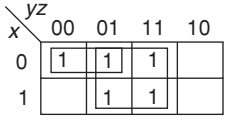
Solution: (C)

The outputs of decoder are in active low state. So, we can express outputs as $\overline{Y_7}, \overline{Y_6} \ldots \overline{Y_0}$

Outputs 0, 1, 3, 5, 7 are connected to NAND gate to form function $F(x, y, z)$

So, $$F = \overline{\overline{Y_0} \cdot \overline{Y_1} \cdot \overline{Y_3} \cdot \overline{Y_5} \cdot \overline{Y_7}}$$
$$= Y_0 + Y_1 + Y_3 + Y_5 + Y_7$$
$$= \Sigma(0, 1, 3, 5, 7)$$

By using K-maps

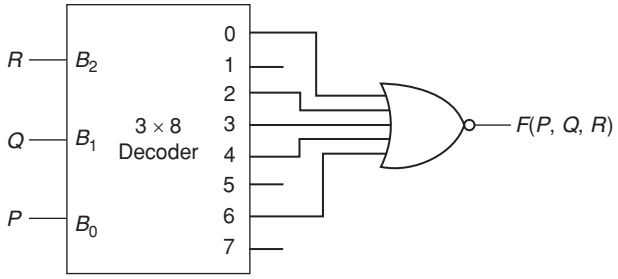


$$F = z + x'y'$$

Example 5: The minimal POS form of output function $f(P, Q, R)$ is

(A) $P\overline{Q} + PR$ (B) $P + \overline{Q}R$

(C) $P(\overline{Q} + R)$ (D) $Q(\overline{P} + R)$



Solution: (C)

The outputs of decoder are in normal form. 0, 2, 3, 4, 6 outputs are connected to NOR gate to form $F(P, Q, R)$

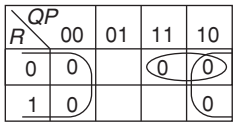
So $$F = \overline{Y_0 + Y_2 + Y_3 + Y_4 + Y_6}$$
$$= \overline{Y_0} \cdot \overline{Y_2} \cdot \overline{Y_3} \cdot \overline{Y_4} \cdot \overline{Y_6}$$

$Y_0, Y_1 \ldots Y_7$ indicate minimum terms, whereas $\overline{Y_0}, \overline{Y_1} \ldots \overline{Y_7}$ are maximum terms.

So $F = \pi(0, 2, 3, 4, 6)$

Here, from the decoder circuit MSB is R, LSB is P.

By using K-map



$$F(P, Q, R) = P(R + \overline{Q})$$

ENCODERS

It is a digital circuit that performs the inverse operation of a decoder.

An encoder has 2^n (or fewer) input lines and n output lines.

It is also known as an octal to binary converter.

Consider an 8–3 line encoder:

Truth Table

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	1	0	0	1	1
0	0	0	0	1	0	0	1	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

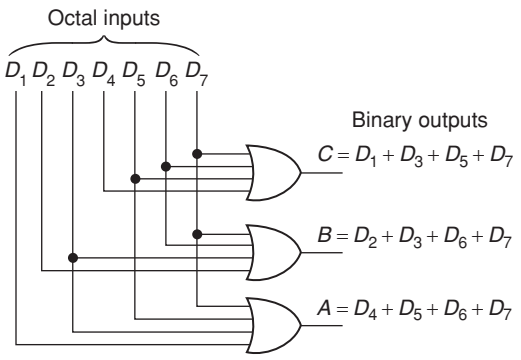


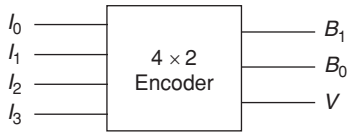
Figure 17 Logic diagram

Priority Encoder

A priority encoder is an encoder circuit that includes the priority function.

When two or more inputs are present, the input with higher priority will be considered.

Consider the 4 × 2 priority encoder.



I_3	I_2	I_1	I_0	B_1	B_0	V
1	X	X	X	1	1	1
0	1	X	X	1	0	1
0	0	1	X	0	1	1
0	0	0	1	0	0	1
0	0	0	0	X	X	0

I_3–I_0 are inputs and B_1 B_0 are binary output bits, valid (V) output is set to 1 when at least one input is present at input (I_3–I_0).

When there is no input present, (I_3–I_0 = 0000) then $V = 0$, for this combination the output B_1B_0 will not be considered.

The higher the subscript number, the higher the priority of the input. Input I_3 has the highest priority, I_2 has the next priority level. Input I_0 has lowest priority level. The Boolean expressions for output B_1 B_0 are

$$B_1 = I_3 + \overline{I}_3 I_2$$
$$= I_3 + I_2$$
$$B_0 = I_3 + \overline{I}_3 \overline{I}_2 I_1$$
$$= I_3 + \overline{I}_2 I_1$$
$$V = I_3 + I_2 + I_1 + I_0$$

Multiplexer

A multiplexer (MUX) is a device that allows digital information from several sources to be converted on to a single line for transmission over that line to a common destination.

The MUX has several data input lines and a single output line. It also has data select inputs that permits digital data on any one of the inputs to be switched to the output line.

Depending upon the binary code applied at the selection inputs, one (out of 2^n) input will be gated to single output. It is one of the most widely used standard logic circuits in digital design. The applications of multiplexer include data selection, data routing, operation sequencing, parallel to serial conversion, and logic function generation.

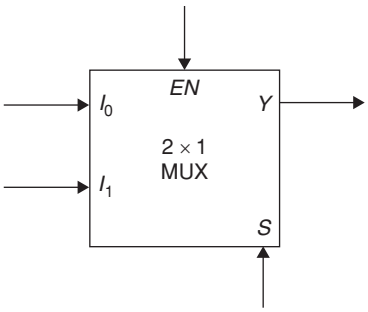
2^n inputs will be controlled by n selection lines and multiplexer will have 1 output, we denote it as $2^n \times 1$ multiplexer (data selector).

In other words, a multiplexer selects 1 out of n input data sources and transmits the selected data to a single output channel, this is called as multiplexing.

Basic 2 × 1 Multiplexer

The figure shows 2 × 1 multiplexer block diagram; it will have 2 inputs I_0 and I_1, one selection line S, and one output Y. The function table is as shown here.

EN	S	Y
0	x	0
1	0	I_0
1	1	I_1

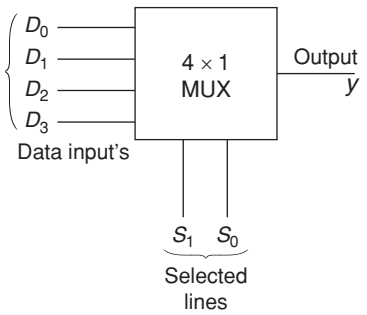


The output equation of 2 × 1 multiplexer is $Y = EN(I_0\overline{S} + I_1S)$.

When enable is 1, the multiplexer will work in normal mode, else the multiplexer will be disabled.

Sometimes enable input will be active low enable $\overline{EN}$, then $Y = \overline{EN}(I_0\overline{S} + I_1S)$.

The 4 × 1 Multiplexer



If a binary zero $S_1 = 0$ and $S_0 = 0$ as applied to the data select line the data input D_0 appear on the data output line and so on.

S_1	S_0	y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$y = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$$

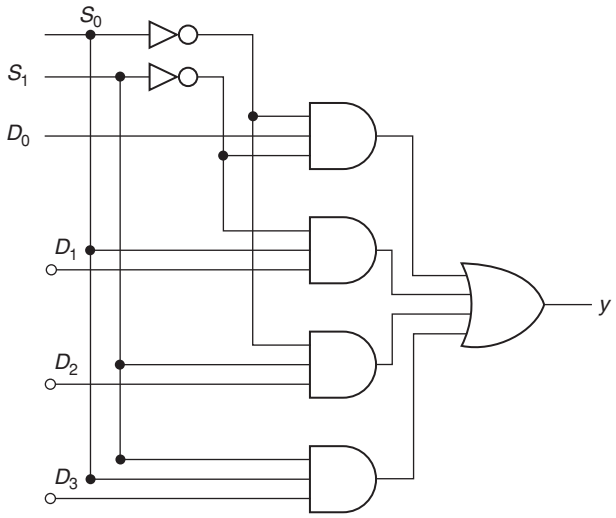


Figure 18 Logic diagram

For 8 × 1 multiplexer with 8 inputs from I_0 to I_7 based on selection inputs $S_2S_1S_0$, the equation for output

$$Y = I_0\overline{S_2}\,\overline{S_1}\,\overline{S_0} + I_1\overline{S_2}\,\overline{S_1}S_0 + I_2\overline{S_2}S_1\overline{S_0} + I_3\overline{S_2}S_1S_0$$
$$+I_4S_2\overline{S_1}\,\overline{S_0} + I_5S_2\overline{S_1}S_0 + I_6S_2S_1\overline{S_0} + I_7\overline{S_2}S_1S_0$$

From multiplexer equation we can observe, each input is associated with its minterm (in terms of selection inputs).

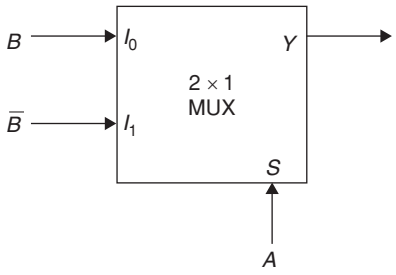


Figure 19 Basic gates by using MUX

$Y = \overline{A}B + A\overline{B} = X$-OR gate, we can interchange inputs A and B also,

By interchanging inputs I_0 and I_1, $Y = \overline{A}\,\overline{B} + AB$, X-NOR gate.

Similarly, we can build all basic gates by using 2 × 1 multiplexer.

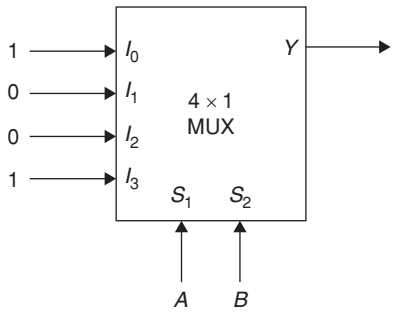
Example 6: If $I_0 = 1$, $I_1 = 0$, $S = A$, then Y is?

Solution: $= (I_0\overline{S} + I_1S) = \overline{A}$. It implements NOT gate.

Example 7: What should be the connections to implement NAND gate by using 2 × 1 MUX?

Solution: $Y = \overline{AB} = \overline{A} + \overline{B} = \overline{A} + A\overline{B} = 1\cdot\overline{A} + \overline{B}\cdot A$

By considering $I_0 = 1$, $I_1 = \overline{B}$, $S = A$, we can implement NAND gate, or by interchanging A and B also we can get the same answer.



For the above 4 × 1 multiplexer $Y = \overline{AB} + AB = X$-NOR gate. Similarly to implement 2-input gates by using 4 × 1 multiplexer, the inputs I_0, I_1, I_2, I_3 should be same as the terms in the truth table of that gate.

Logic Function Implementation by Using Multiplexer

Let us consider a full subtractor circuit (Barrow) to be implemented by using multiplexer.

Full subtractor Barrow (B) is a function of 3-inputs X, Y, Z. The truth table is

X	Y	Z	B	4 × 1 MUX	2 × 1 MUX
0	0	0	0	$B = Z$	$B = Y + Z$
0	0	1	1		
0	1	0	1	$B = 1$	
0	1	1	1		
1	0	0	0	$B = 0$	$B = YZ$
1	0	1	0		
1	1	0	0	$B = Z$	
1	1	1	1		

To implement Barrow by using 8 × 1 multiplexer, connect the three variables X, Y, Z directly to selection lines of the multiplexer, and connect the corresponding values of B to inputs, i.e., for $I_0 = 0$, $I_1 = 1$, $I_2 = 1$, etc. as per above truth table.

To implement Barrow by using 4 × 1 multiplexer, connect any two variables to selection lines (in this case X, Y) and write output (B) in terms of other variable, for $XY = 00$, output B is same as Z, so connect $I_0 = Z$, similarly 1, 0, Z for remaining inputs.

To implement the function by using 2 × 1 multiplexer, connect 1 variable as selection line (in this case consider X) and write output (B) in terms of other variables, for $X = 0$,

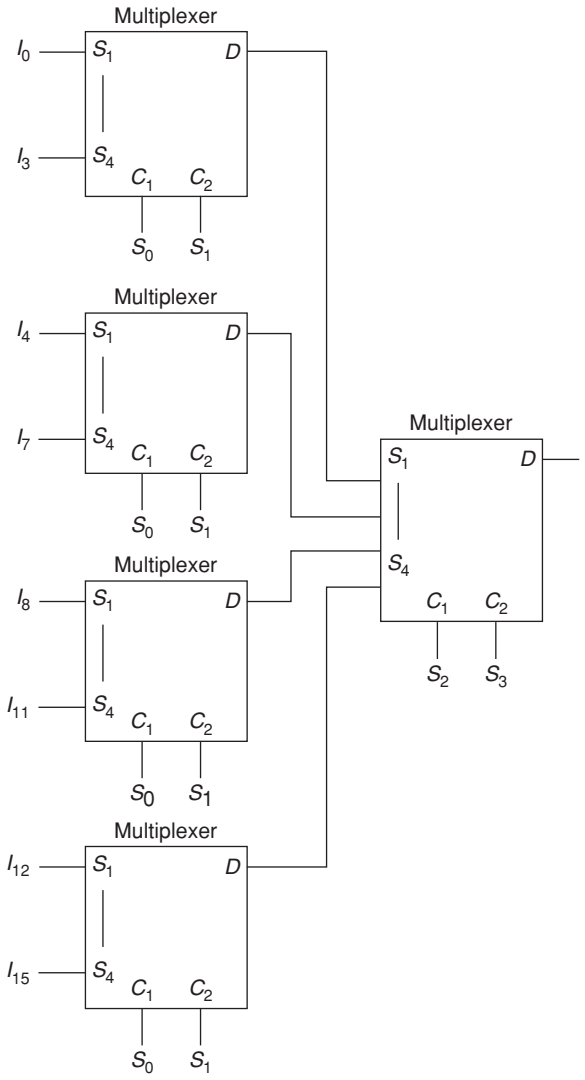
output B is varies as $B = Y + Z$, so connect $I_0 = Y + Z$. For $X = 1$, output B varies as $B = YZ$, connect $I_1 = YZ$.

N variable function can be implemented by using $2^{N-1} \times 1$ multiplexer without any extra hardware.

Implementation of Higher Order Multiplexer by Using Lower Order Multiplexers

By using lower order multiplexers, we can implement higher order multiplexers, for example by using 4×1 multiplexer, we can implement 8×1 MUX or 16×1 MUX or other higher order multiplexers.

Let us consider implementation of 16×1 MUX by using 4×1 MUX. 16×1 MUX will have inputs I_0 to I_{15} and selection lines S_0 to S_3, whereas 4×1 MUX will have only 4 input lines, and 2 selection lines, so we require four 4×1 MUX to consider all inputs I_0 to I_{15}, and again to select one of the four outputs of these four multiplexers one more 4×1 multiplexer is needed (for which we will connect higher order selection lines S_2 and S_3). So, total of 5, 4×1 multiplexers are required to implement 16×1 MUX.

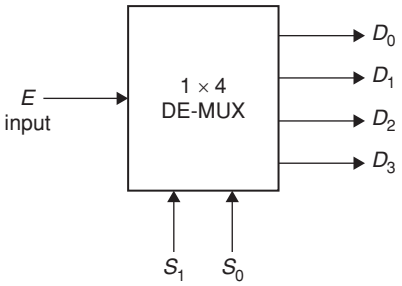


In a similar fashion, to design 4×1 MUX, we require three, 2×1 multiplexers, and to design 8×1 multiplexer, we require seven, 2×1 multiplexers.

De-multiplexer

The de-multiplexer [DeMUX] basically serves opposite of the multiplexing function. It takes data from one line and distributes them to a given number of output lines.

The other name for de-multiplexer is data distributor, as it receives information on a single line and distributes it to a possible 2^n output lines, where n is the number of selection lines, and value of n selects the line.



S_1	S_0	D_3	D_2	D_1	D_0
0	0	0	0	0	E
0	1	0	0	E	0
1	0	0	E	0	0
1	1	E	0	0	0

When $S_1S_0 = 10$; D_2 will be same as input E, and other outputs will be maintained at zero (0).

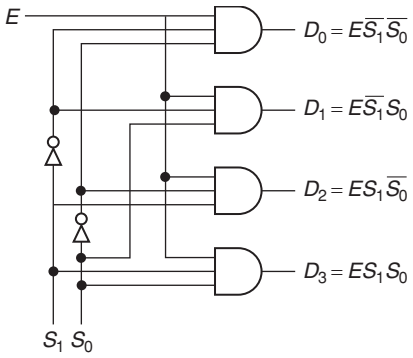


Figure 20 Logic diagram

Memory and Programmable Logic

A memory unit is a device to which binary information is transferred for storage and from which information is retrieved when needed for processing.

There are two types of memories that are used in digital systems. (1) Random Access Memory (RAM) and (2) Read Only Memory (ROM).

RAM stores new information for later use, RAM can perform both write and read operation, whereas ROM can perform only the read operation.

ROM is one example of a programmable logic device (PLD). Other such units are programmable logic array (PLA), programmable array logic (PAL) and field programmable logic array (FPGA).

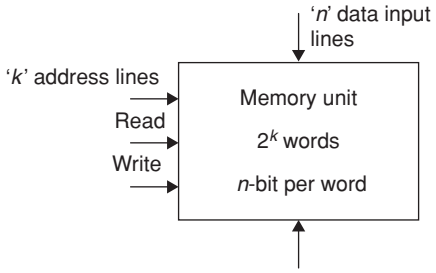
A PLD is an integrated circuit with internal logic gates connected through electronic paths that behave similar to fuses.

Random Access Memory

If the time taken to read or write data from any memory location is the same, then we call it as *random access memory*, whereas in *serial access memory* like magnetic tapes, the time taken to access different locations is different.

Any memory element will have address selection inputs to locate each word in memory, and the control input *Read/Write* specify the operation as well as data input or output lines for data writing or reading operation.

Each word in memory is assigned with an address, starting from 0 to 2^{k-1}, where k is the number of address lines.



The selection of a specific word inside memory is done by applying the k-bit address to the address lines.

For example, 4k × 16 memory has 12-bits in address and 16-bit in each word. Similarly, 32k × 16 memory has 15-bits as address lines and 16-bit in each word.

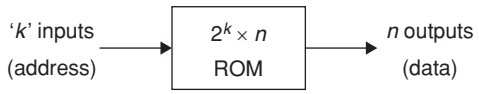
The memory enable (chip select) is used to enable the particular memory chip in a multichip implementation of a large memory.

Memory Enable	Read/Write	Memory Operation
0	X	No operation
1	0	Write to selected location
1	1	Read from selected location

Read-only Memory

A read-only memory (ROM) is a type of memory which stores data permanently or semi-permanently, i.e., data can be erasable. As the name indicates, data stored in ROM can only be read. The data that a user wanted to store on ROM will be given to manufacturer to fabricate the masked ROM which stores permanently. Or the user can programme the ROM in lab according to their specifications in the case of PROM. EPROM/EEPROM are the type of ROMs where user can erase data, and rewrite new data to ROM.

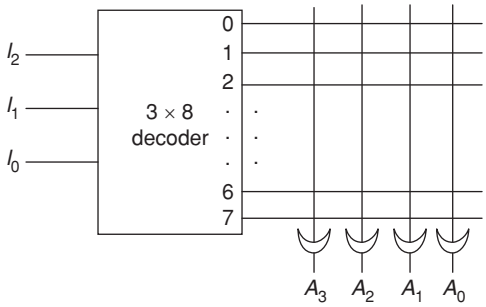
ROM also have address inputs similar to RAM, to access one of the memory location, and to read the data from that memory location, data output lines are available.



The number of words in a ROM is determined from the fact that k address input lines are needed to specify 2^k words.

ROM does not have data inputs because it does not have write operation.

Consider for example 8 × 4 ROM, the unit consists of eight words of 4-bit each. There are three input lines that from the binary numbers from 0 to 7 for address.



The above figure shows the internal logic construction of ROM, the three inputs are decoded into eight district outputs by means of 3 × 8 decoder, each output of the decoder represents a memory address. The eight outputs of decoder are connected to each of the four OR gates.

Each OR gate must be considered as having eight inputs. Each output of the decoder is connected to one of the inputs of each OR gate. Since each OR gate has eight input connections and there are four OR gates, the ROM contains 8 × 4 = 32 internal connections.

In general, a $2^{m \times n}$ ROM will have an internal $m \times 2^m$ decoder and n OR gates. Each OR gate has 2^m inputs, which are connected to each of the outputs of the decoder.

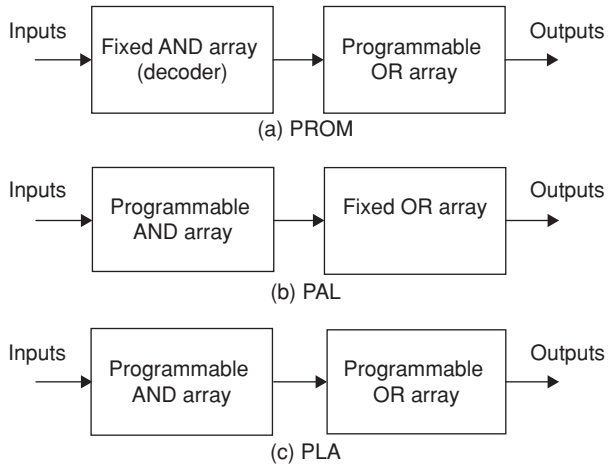
These intersections are programmable. A programmable connection between two lines is logically equivalent to a switch that can be altered to be either closed or open.

The internal binary storage of ROM is specified by a truth table that shows the word content in each address.

Combinational circuits can be implemented by ROM. For this, each output terminal of PROM is considered as separately as, the output of a Boolean function expressed as a sum of min terms.

The PROM is a combinational programmable logic device (PLD)—an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND–OR sum of product implementation.

There are three major types of combinational PLDS, differing in the placement of the programmable connections in the AND–OR array.



The PROM has fixed AND array constructed as a decoder, and a programmable OR array. The programmable OR gates implement the Boolean function in sum of min terms form.

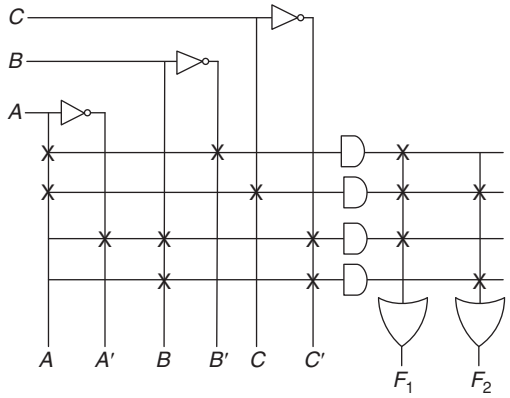
Most flexible PLD is PLA, in which both the AND and OR arrays can be programmed.

The PLA is similar in concept to the PROM, except that the PLA does not provide full decoding of the variables and doesn't generate all the min terms.

PLA for the functions,

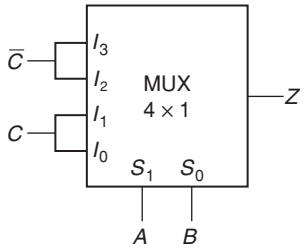
$F_1 = AB' + AC + A'\,B\,C'$

$F_2 = AC + BC'$ is shown below.



Solved Examples

Example 1: The multiplexer shown in the figure is a 4 : 1 multiplexer, where the output 'z' is



Solution:

A_1	B_0	Z
0	0	C
0	1	C
1	0	$\overline{C}$
1	1	$\overline{C}$

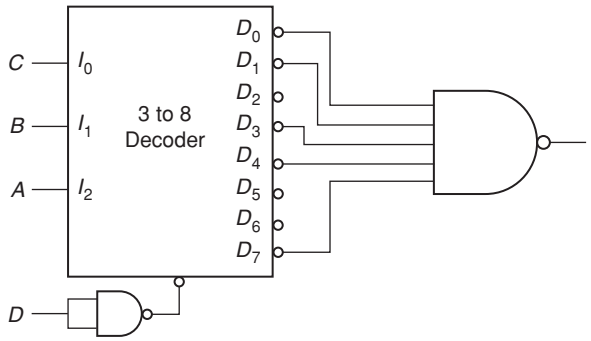
$$\therefore \quad Z = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + A\overline{B}\overline{C}$$

$$= \overline{B}(\overline{A}C + A\overline{C}) + B(\overline{A}C + A\overline{C})$$

$$= (\overline{A}C + A\overline{C})(B + \overline{B})(x + \overline{x} = 1)$$

$$\therefore \quad = \overline{A}C + A\overline{C} = A \oplus C$$

Example 2: The logic circuit shown in figure implements



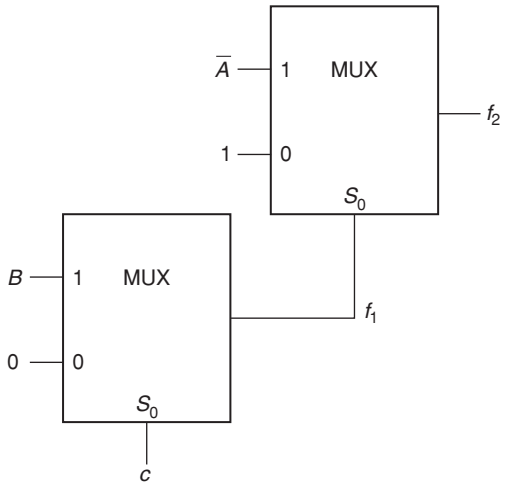
Solution: $z = D(\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}\,\overline{C} + ABC)$

$$= D(\overline{A}\,\overline{B}(C + \overline{C}) + BC(\overline{A} + A) + A\overline{B}\,\overline{C})$$

$$\times D(\overline{B}\,\overline{A} + \overline{B}\,\overline{C} + BC)$$

$$= D(B \Theta C + \overline{A}\,\overline{B})$$

Example 3: The network shown in figure implements



Solution: $f_1 = \overline{C}_0 + CB = CB,\ f_1 = CB$

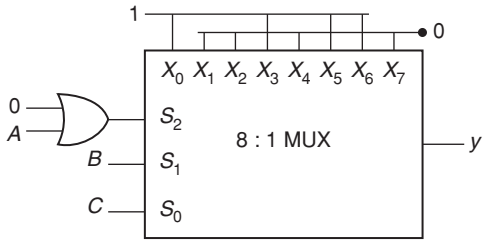
$$F_2 = \overline{f_1} + f_1\overline{A} = \overline{A} \cdot CB + \overline{CB}$$

$$= \overline{A} + \overline{CB}$$

$$= \overline{A} + \overline{C} + \overline{B} = \overline{ABC}$$

$\therefore$ NAND gate

Example 4: In the TTL circuit in figure, S_2 to S_0 are select lines and x_7 to x_0 are input lines. S_0 and X_0 are LSBs. The output y is

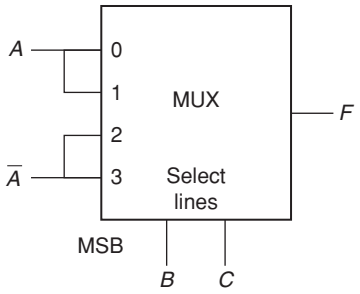


Solution: $S_2 = A, S_1 = B, S_0 = C$

$S_2(A)$	$S_1(B)$	$S_0(C)$	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

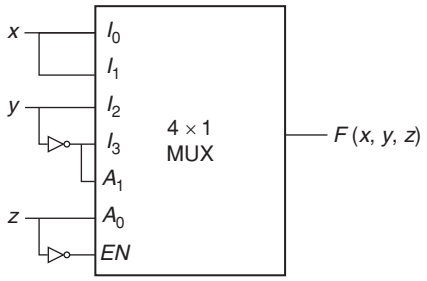
$$Y = \overline{A}\overline{B}\overline{C} + AB\overline{C} + A\overline{B}C + \overline{A}BC$$
$$= \overline{C}(\overline{A}B + AB) + C(A\overline{B} + \overline{A}B)$$
$$Y = \overline{C}(\overline{A} \oplus B) + C(A \oplus B) = A \oplus B \odot C$$

Example 5: The logic realized by the adjoining circuit is



Solution: $F = \overline{B}\overline{C}A + \overline{B}CA + B\overline{C}\overline{A} + BC\overline{A}$
$= \overline{C}(\overline{B}A + B\overline{A}) + C(\overline{B}A + B\overline{A})$
$= A\overline{B} + \overline{A}B(C + \overline{C}) = A \oplus B$

Example 6: Consider the following multiplexer, where I_0, I_1, I_2, I_3 are four date input lines selected by two address line combinations A_1A_0 = 00, 01, 10, 11, respectively and f is the output of the multiplexer. EN is the enable input, the function $f(x, y, z)$ implemented by the below circuit is



Solution: $A_1 = \overline{y} \cdot A_0 = z, EN = \overline{z}$

A_1	A_0	S	I
0	0	$(y\overline{z})$	x
0	1	$(y\ z)$	x
1	0	$(\overline{y}\overline{z})$	y
1	1	$(\overline{y}z)$	$\overline{y}$

$$f(x, y, z) = \text{SI} = (xy + 0 + \overline{y}z) \cdot EN$$
$$= xy \cdot \overline{z}$$

EXERCISES

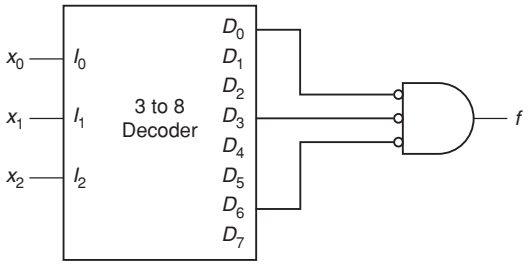
Practice Problems I

Directions for questions 1 to 25: Select the correct alternative from the given choices.

1. The binary number 110011 is to be converted to gray code. The number of gates and type required are
(A) 6, AND (B) 6, X-NOR
(C) 6, X-OR (D) 5, X-OR

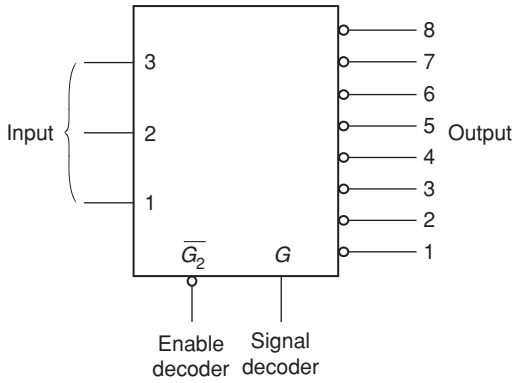
2. The number of 4- to 16-line decoder required to make an 8 to 256-line decoder is
(A) 16 (B) 17
(C) 32 (D) 64

3. $f(x_2, x_1, x_0) = ?$



(A) $\pi(1, 2, 4, 5, 7)$ (B) $\Sigma(1, 2, 4, 5, 7)$
(C) $\Sigma(0, 3, 6)$ (D) $\pi(0, 2, 3, 6)$

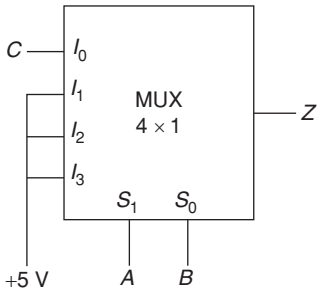
4. A 3 to 8 decoder is shown below:



All the output lines of the chip will be high except pin 8, when all the inputs 1, 2, and 3

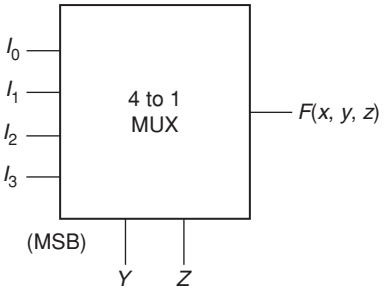
(A) are high; and G, G_2 are low
(B) are high; and G is low G_2 is high
(C) are high; and G, G_2 are high
(D) are high; and G is high G_2 is low

5. The MUX shown in figure is 4 × 1 multiplexer the output z is



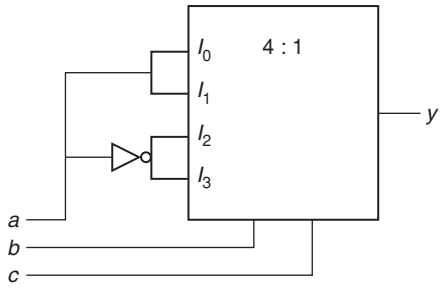
(A) $A\,B\,C$
(B) $A \oplus B \oplus C$
(C) $A\,\Theta\,B\,\Theta\,C$
(D) $A + B + C$

6. If a 4 to 1 MUX (shown below) realizes a three variable function $f(x, y, z) = xy + x\bar{z}$, then which of the following is correct?



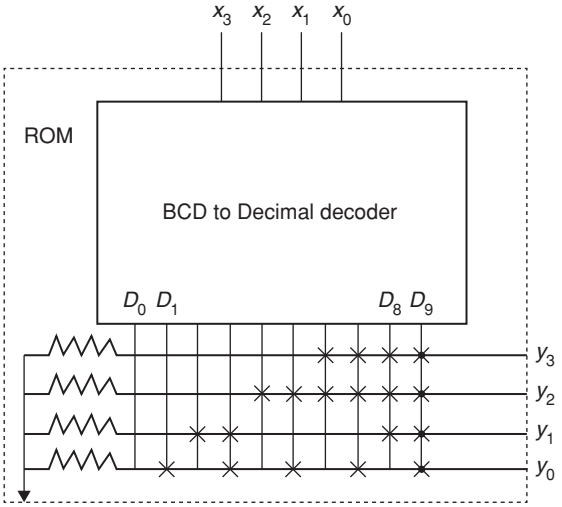
(A) $I_0 = X, I_1 = 0, I_2 = X, I_3 = X$
(B) $I_0 = 0, I_1 = 1, I_2 = Y_1, I_3 = X$
(C) $I_0 = X, I_1 = 1, I_2 = 0, I_3 = X$
(D) $I_0 = X, I_1 = 0, I_2 = X, I_3 = Z$

7. The circuit shown in the figure is same as



(A) Two input NAND gate with a and c inputs
(B) Two input NOR gate with a and c inputs
(C) Two input X-OR gates with a and b inputs
(D) Two input X-NOR gate with b and c as inputs.

8. If the input x_3, x_2, x_1, x_0 to the ROM in the figure are 8421 BCD numbers, then the outputs y_3, y_2, y_1, y_0 are

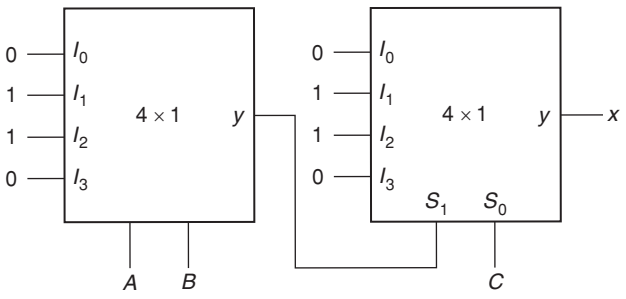


(A) Gray code numbers
(B) 2421 BCD
(C) Excess – 3 code numbers
(D) 84 – 2 – 1

9. A 4-bit parallel full adder without input carry requires

(A) 8 HA, 4 OR gates
(B) 8 HA, 3 OR gates
(C) 7 HA, 4 OR gates
(D) 7 HA, 3 OR gates

10. In the circuit find X.



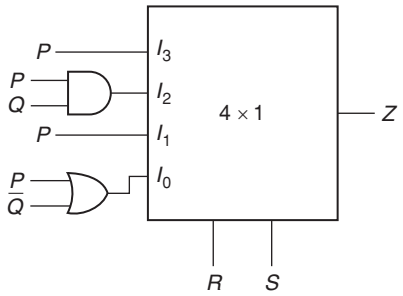
(A) $A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC$

(B) $\bar{A}BC + A\bar{B}C + AB\bar{C} + \bar{A}\bar{B}\bar{C}$

(C) $AB + BC + AC$

(D) $\bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{C}$

11. Find the function implemented.



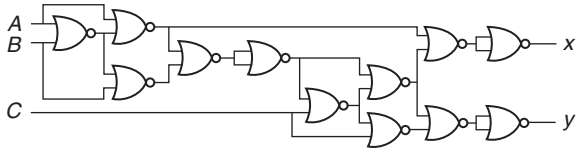
(A) $PQ + PS + \bar{Q}\bar{R}\bar{S}$

(B) $P\bar{Q} + PQ\bar{R} + \bar{P}\bar{Q}\bar{S}$

(C) $P\bar{Q}\bar{R} + \bar{P}QR + PQRS + \bar{Q}\bar{R}\bar{S}$

(D) $PQ\bar{R} + PQR\bar{S} + P\bar{Q}\bar{R}S + \bar{Q}\bar{R}\bar{S}$

12. Which function is represented by the given circuit?

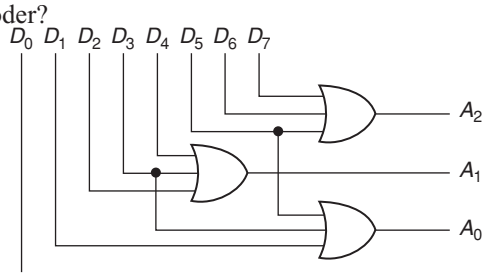


(A) Full adder (B) Full subtractor

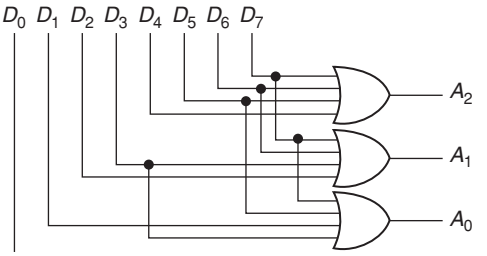
(C) Comparator (D) Parity generator

13 Which of the following represents octal to binary encoder?

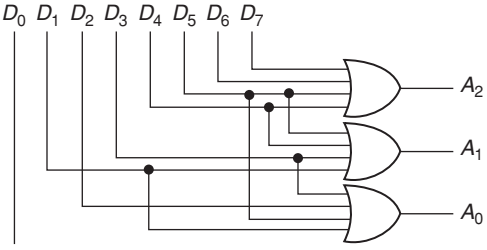
(A)



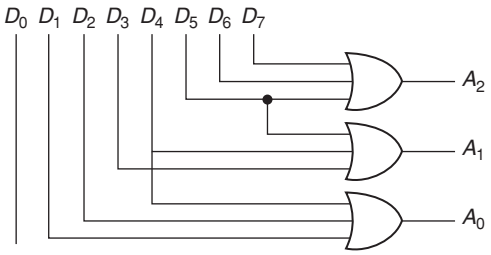
(B)



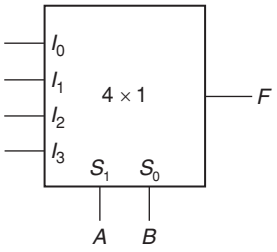
(C)



(D)



14. For a MUX to function as a full adder what should be the input provided to the I_0, I_1, I_2, I_3 if A and B are the select lines?



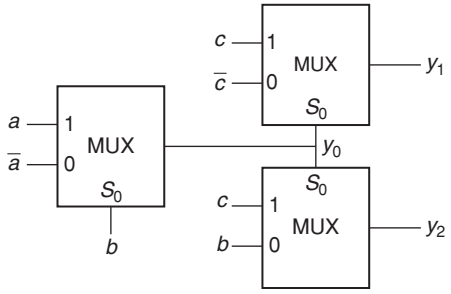
(A) $I_0 = I_1 = C_{in}; I_2 = I_3 = \overline{C_{in}}$

(B) $I_0 = I_1 = \overline{C_{in}}; I_2 = I_3 = C_{in}$

(C) $I_0 = I_3 = C_{in}; I_1 = I_2 = \overline{C_{in}}$

(D) $I_0 = I_3 = \overline{C_{in}}; I_1 = I_2 = C_{in}$

15. The given circuit act as



(A) full adder (B) half adder

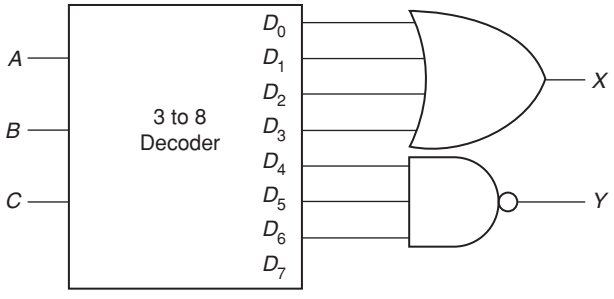
(C) full subtractor (D) half subtractor

16. For a 4 × 16 decoder circuit, the outputs of decoder (y_0, y_1, y_4 . y_5 . y_{10} . y_{11} . y_{14} . y_{15}) are connected to 8 input NOR gate. The expression of NOR gate output is?

(A) $A \oplus D$ (B) $A \odot D$

(C) $A \odot C$ (D) $A \oplus C$

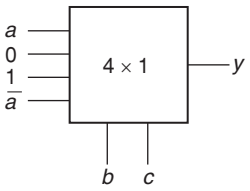
17. The function implemented by decoder is.

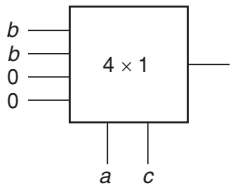


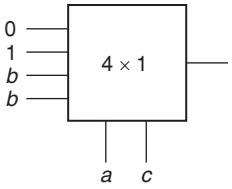
(A) $X = A'BC' + B'C', y = A + B$
(B) $X = A'C' + B'C', y = 1$
(C) $X = \overline{A}, y = 0$
(D) $X = \overline{A}, y = 1$

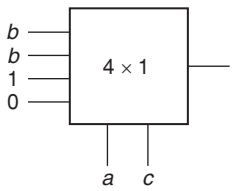
18. A relay is to operate with conditions that it should be on when the input combinations are 0000, 0010, 0101, and 0111. The states 1000, 1001, 1010 doesn't occur. For rest of the status relay should be off. The minimized Boolean expression notifying the relationship is
(A) $BC + ACD$ (B) $\overline{B}\overline{D} + \overline{A}BD$
(C) $BD + AC$ (D) $AB + CD$

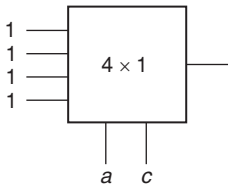
19. If a function has been implemented using MUX as shown, implement the same function with a and c as the select lines



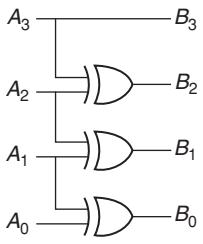
(A)


(B)


(C)


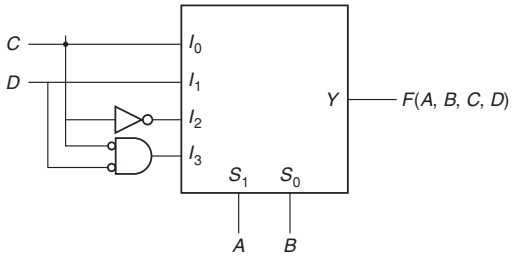
(D)


20. The circuit is used to convert one code to another. Identify it.



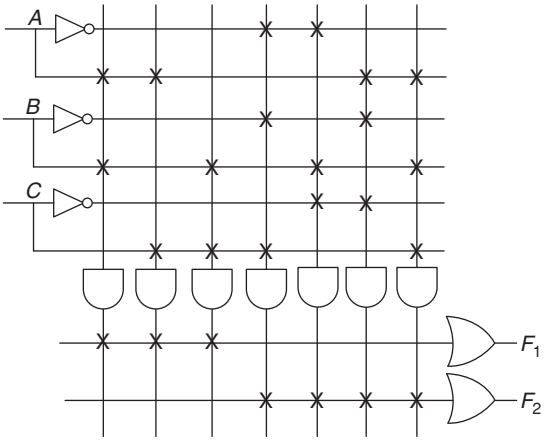
(A) Binary to grey
(B) Grey to binary
(C) Grey to XS – 3
(D) Grey to 8421

21. The Boolean function realised by logic circuit is



(A) $F = \Sigma m(0, 1, 3, 5, 9, 10, 14)$
(B) $F = \Sigma m(2, 3, 5, 7, 8, 12, 13)$
(C) $F = \Sigma m(1, 2, 4, 5, 11, 14, 15)$
(D) $F = \Sigma m(2, 3, 5, 7, 8, 9, 12)$

22. A circuit receives a 4-bit excess three code. The function to detect the decimal numbers 0, 1, 4, 6, 7, 8 is (assume inputs as A, B, C, D)?
(A) $ABC + \overline{A}C + BCD + AD$
(B) $CD + AD + AC + ACD$
(C) $CD + AD + \overline{A}\overline{C} + AC\overline{D}$
(D) $CD + AD + AC + \overline{A}\overline{C}\overline{D}$



Common Data for Questions 23 and 24:

23. Identify the function implemented.
(A) $f_1(A, B, C) = AB + BC + AC$
(B) $f_1(A, B, C) = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C}$
(C) $f_1(A, B, C) = A + B + C$
(D) None of these

24. From the above, PLD implemented is
(A) PLA
(B) PROM
(C) PAL
(D) CPLD

25. The circuit implemented using the PLA in the above figure is
(A) full adder
(B) full subtractor
(C) half adder
(D) half subtractor

Practice Problems 2

Directions for questions 1 to 24: Select the correct alternative from the given choices.

1. For a binary half subtractor having two input A and B, the correct set of logical expression for the outputs $D = (A \text{ minus } B)$ and X (borrow) are

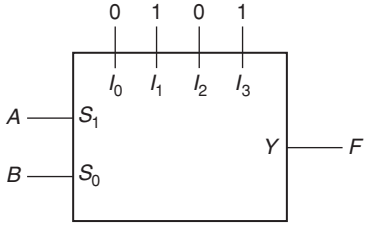
(A) $D = AB + \overline{AB},\ X = \overline{A}B$

(B) $D = \overline{AB} + A\overline{B} +,\ A\overline{B},\ X = \overline{A}B$

(C) $D = \overline{A}B + A\overline{B},\ X = \overline{A}B$

(D) $D = AB + \overline{A}\overline{B},\ X = \overline{A}B$

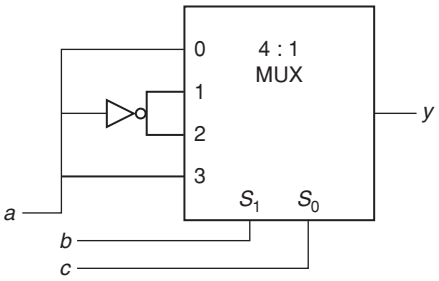
2. The function 'F' implemented by the multiplexer chip shown in the figure is



(A) A (B) B

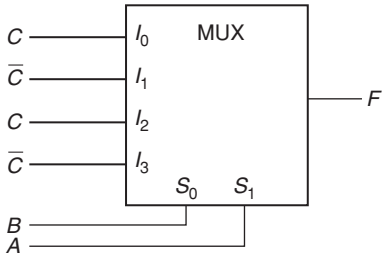
(C) $\overline{A}B$ (D) $A\overline{B} + \overline{A}B$

3. The following multiplexer circuit is equal to



(A) implementation of sum equation of full adder
(B) implementation of carry equation of full adder
(C) implementation of Borrow equation of full substractor
(D) All of the above

4. The output 'F' of the multiplexer circuit shown in the figure will be



(A) $AB + B\overline{C} + \overline{C}A + \overline{B}\overline{C}$ (B) $A \oplus B \oplus C$

(C) $A \oplus B$ (D) $B \oplus C$

5. Full subtractor can be implemented by using

(A) 3 to 8 line decoder only
(B) 3 to 8 line decoder and one OR gate
(C) 3 to 8 line decoder and two OR gates
(D) None

6. What are the difference and Borrow equations for the above circuit?

(A) $D = x\ \Theta\ y\ \Theta\ z,\ B = x'y + yz + zx'$
(B) $D = X \oplus y \oplus z,\ B = xy + yz + zx$
(C) $D = x \oplus y \oplus z,\ B = x'y + yz + zx'$
(D) A, C both

7. Combinational circuits are one in which output depends _____, whereas sequential circuit's output depends _____

(A) only on present input, only on past input
(B) only on present input, only on past and future input
(C) only on present input, only on present input and past output
(D) on present input, on past and present output

8. The sum output of the half adder is given by (assume A and B as inputs)

(A) $S = AB(\overline{A+B})$ (B) $S = (A+B)\overline{AB}$

(C) $S = (A+B)(AB)$ (D) $S = (\overline{A}+\overline{B})(\overline{A}B)$

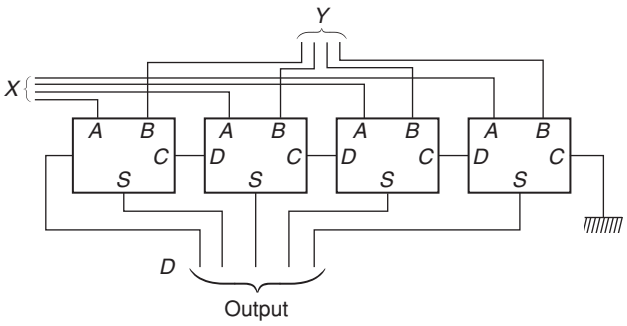
9. MUX implements which of the following logic?

(A) NAND-XOR (B) AND-OR
(C) OR-AND (D) XOR-NOT

10. A DeMUX can be used as a

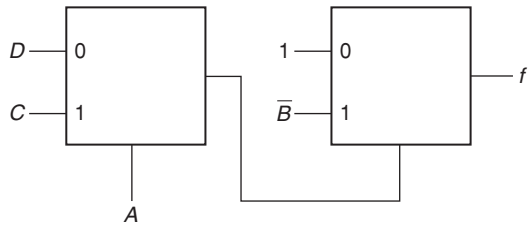
(A) comparator (B) encoder
(C) decoder (D) adder

11. If we have inputs as A, B and C and output as S and D. We are given that $S = A \oplus B \oplus C$. $D = BC + \overline{A}B + \overline{A}C$. Which of the circuit is represented by it?



(A) 4-bit adder giving $X + Y$
(B) 4-bit subtractor giving $X - Y$
(C) 4-bit subtractor giving $Y - X$
(D) 4-bit adder giving $X + Y + S$

12. The Boolean function f implemented in the figure using two input multiplexers is

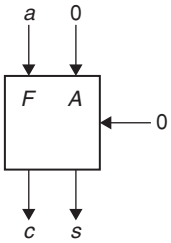


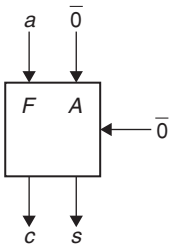
(A) $A\overline{C} + \overline{A}\overline{D} + \overline{D}C + \overline{A}\overline{B}D + A\overline{B}C$
(B) $\overline{A} + A\overline{C} + \overline{A}\overline{D} + \overline{D}\overline{C}$
(C) $\overline{B} + A\overline{C} + \overline{A}\overline{D} + \overline{D}\overline{C}$
(D) $AC + AD + A + B$

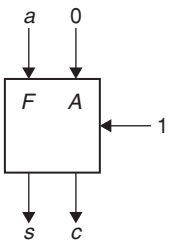
13. The Carry generate and Carry Propagate function of the look ahead carry adder is
(A) $CG = A + B$, $CP = A \oplus B$
(B) $CG = A \oplus B$, $CP = A + B$
(C) $CG = AB$, $CP = A \oplus B$
(D) $CG = AB$, $CP = A + B$

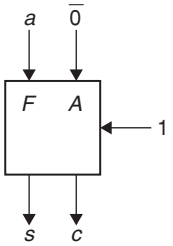
14. If we have a comparator and if E represents the condition for equality i.e., $(A_n \oplus B_n)$, if A_n and B_n are to be compared, then the expression $A_3\overline{B}_3 + E_3A_2\overline{B}_2 + E_3E_2A_1\overline{B}_1 + E_3E_2E_1A \cdot \overline{B}$. represents which of the condition for a 4-bit number?
(A) $A > B$ (B) $B > A$
(C) $A = B$ (D) None of these

15. When full adder is used to function as a 1-bit incrementor, which of the circuit configurations must be used?

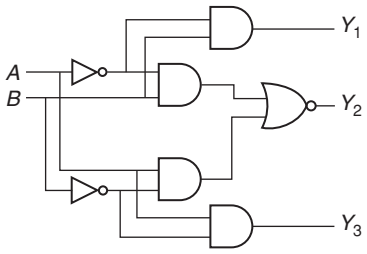
(A)


(B)


(C)


(D)


16. Identify the circuit.



(A) half adder
(B) full adder
(C) 1-bit magnitude comparator
(D) parity generator

17. In order to implement n variable function (without any extra hardware), the minimum order of MUX is
(A) $2^n \times 1$ (B) $2^n \times 1$
(C) $(2^n - 1) \times 1$ (D) $(2^{n-1}) \times 1$

18. A full adder circuit can be changed to full subtractor by adding a
(A) NOR gate (B) NAND gate
(C) Inverter (D) AND gate

19. The half adder when implemented in terms of NAND logic is expressed as
(A) $A \oplus B$ (B) $\overline{A \cdot \overline{AB} \cdot B \cdot \overline{AB}}$
(C) $\overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$ (D) $\overline{\overline{A \cdot \overline{ABB}} \cdot \overline{AB}}$

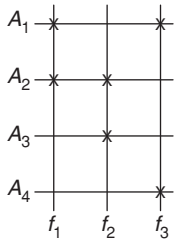
20. For a DeMUX to act as a decoder, what is the required condition?
(A) Input should be left unconnected and select lines behave as a input to decoder
(B) Input should be always 0 and select line behave as inputs to decoder
(C) Both are same
(D) Input should become enable and select lines behave as inputs to decoder

21. For a full subtractor, which of the combination will give the difference?
(A) $\overline{\overline{(A \oplus B)\overline{(A \oplus B)}b_i} \cdot \overline{b_i\overline{(A \oplus B)}b_i}}$
(B) $\overline{\overline{B \cdot \overline{AB}} \cdot \overline{b_i(A \oplus B)}}$
(C) $\overline{\overline{A} + \overline{B}} + \overline{\overline{b_i} + \overline{A \oplus B}}$
(D) None of these

22. A PROM contains
(A) a fixed AND array and programmable OR array
(B) a programmable AND array and OR array
(C) reprogrammable AND and OR array
(D) programmable AND array and fixed OR array

23. Which of the following is true of dynamic memories?
a. The power dissipation is slightly lower than that in static ROM
b. Refreshing operation of data is required to store data permanently
c. The clock will be needed
d. They will contain energy storage elements
(A) a, b, c, d (B) a, b, d
(C) a, b, c (D) c, b, d

24. For the given functions, we have to implement them using a OR gate array. When the input to the gate array must be product of one or two variables the terms A_1, A_2, A_3 should be



$f_1 = \bar{a}b\bar{c} + \bar{b}c + abc$

$f_2 = ac + a\bar{b}c + \bar{a}bc$

$f_3 = bc + \bar{a}b\bar{c} + \bar{a}b\bar{c}$

(A) $ab, ac, \bar{a}\bar{b}, bc$

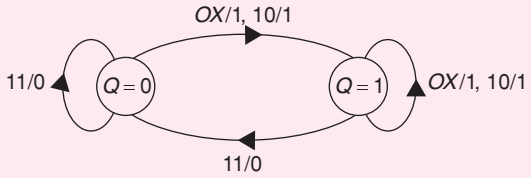
(B) $ac, \bar{a}b, \bar{a}\bar{b}, b\bar{c}$

(C) $\bar{a}\bar{b}, ac, \bar{a}b, bc$

(D) None of these

Previous Years' Questions

1. State diagram of a logic gate which exhibits a delay in the output is shown in the figure, where X is the don't care condition, and Q is the output representing the state. **[2014]**

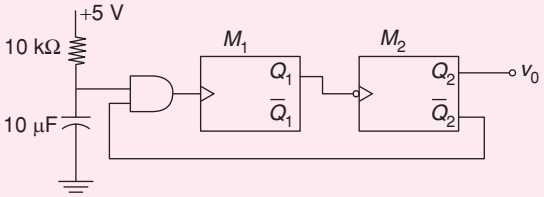


The logic gate represented by the state diagram is

(A) X-OR (B) OR

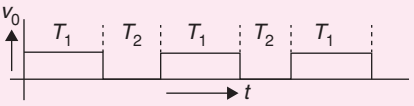
(C) AND (D) NAND

2. Two monoshot multivibrators, one positive edge triggered (M_1) and another negative edge triggered (M_2), are connected as shown in figure. **[2014]**

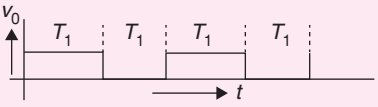


The monoshots M_1 and M_2 when triggered produce pulses of width T_1 and T_2, respectively, where $T_1 > T_2$. The steady state output voltage vo of the circuit is

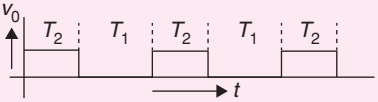
(A)



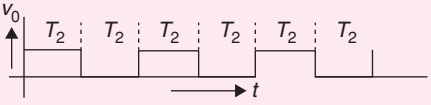
(B)



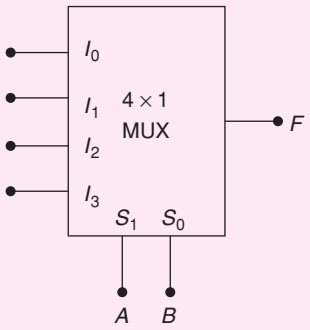
(C)



(D)



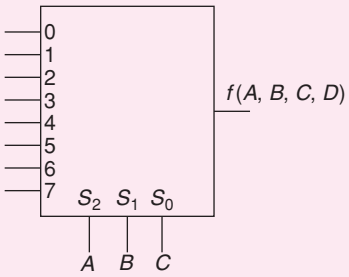
3. In the 4×1 multiplexer, the output F is given by $F = A + B$. Find the required input '$I_3\ I_2\ I_1\ I_0$'. **[2015]**



(A) 1010 (B) 0110

(C) 1000 (D) 1110

4. A Boolean function $f(A, B, C, D) = \prod(1, 5, 12, 15)$ is to be implemented using an 8×1 multiplexer (A is MSB). The inputs ABC are connected to the select inputs $S_2\ S_1\ S_0$ of the multiplexer respectively.



Which one of the following options gives the correct inputs to pins 0, 1,2,3,4,5,6,7 in order? **[2015]**

(A) $D, 0, D, 0, 0, 0, \bar{D}, D$

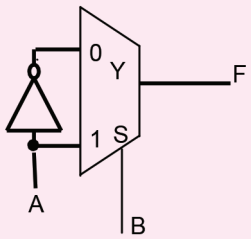
(B) $\bar{D}, 1, \bar{D}, 1, 1, 1, D, \bar{D}$

(C) $D, 1, D, 1, 1, 1, \bar{D}, D$

(D) $\bar{D}, 0, \bar{D}, 0, 0, 0, D, \bar{D}$

5. Consider the following circuit which uses a 2-to-1 multiplexer as shown in the figure below. The Boolean expression for output F in terms of A and B is

[2016]



(A) $A \oplus B$ (B) $\overline{A+B}$

(C) $A+B$ (D) $\overline{A \oplus B}$

Answer Keys

Exercises

Practice Problems 1

1. D	**2.** B	**3.** B	**4.** D	**5.** D	**6.** A	**7.** C	**8.** B	**9.** D	**10.** A
11. A	**12.** B	**13.** B	**14.** C	**15.** C	**16.** D	**17.** D	**18.** B	**19.** C	**20.** A
21. D	**22.** D	**23.** A	**24.** A	**25.** A					

Practice Problems 2

1. C	**2.** B	**3.** A	**4.** D	**5.** C	**6.** C	**7.** C	**8.** B	**9.** B	**10.** C
11. B	**12.** C	**13.** C	**14.** A	**15.** C	**16.** C	**17.** C	**18.** C	**19.** C	**20.** D
21. A	**22.** A	**23.** A	**24.** C						

Previous Years' Questions

1. D **2.** C **3.** B **4.** B **5.** D