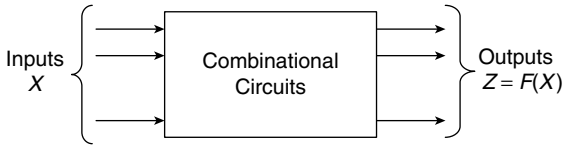
Chapter 3

Combinational Circuits

CHAPTER HIGHLIGHTS

- *Combinational Logic Design*
- *Arithmetic Circuits*
- *Half Adder*
- *NAND Logic*
- *Binary Adder*
- *N-bit Comparator*
- *Code Converters*
- *Combinational Logic Implementation*
- *Priority Encoder*
- *Multiplexer*
- *Basic Gates by Using MUX*
- *De-multiplexer*
- *Memory and Programmable Logic*

Combinational logic is a type of logic circuits 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. Determine the number of input variables and output variables required from the specifications.
2. Assign the letter symbols for input and output.
3. Derive the truth table that defines the required relationship between inputs and outputs.
4. Obtain the simplified Boolean function for each output by using K-map or algebraic relations.
5. Draw the logic diagram for simplified expressions.

We will discuss combinational circuits under three categories:

1. Arithmetic circuits
2. Code converters
3. Data processing circuits

Arithmetic Circuits

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

Half Adder

Addition is an arithmetic operation, and here, to implement addition in digital circuits, we have to implement by logical gates. Therefore, the addition of binary numbers will be represented by the logical expressions. Half adder is an arithmetic circuit that will perform the addition of two binary bits, and the result is viewed in two outputs: sum and carry.

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

Therefore, $S = A\bar{B} + B\bar{A} = A \oplus B$

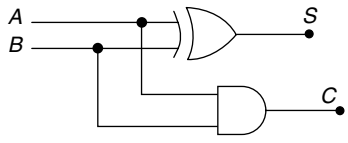
Carry 'c' is the AND of A and B

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

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

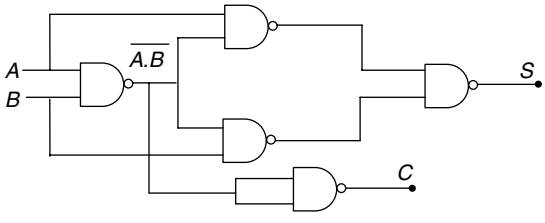


Half adder can also be realized by universal logic such as only NAND gate or only NOR gate, as in the following discussion.

NAND logic

$$\begin{aligned} S &= A\bar{B} + \bar{A}B \\ &= A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B} \\ &= A(\bar{A} + \bar{B}) + B.(\bar{A} + \bar{B}) \\ &= \overline{\overline{A.\overline{A.B}}} \;.\; \overline{\overline{B.\overline{A.B}}} \end{aligned}$$

$$C = AB = \overline{\overline{A.B}}$$

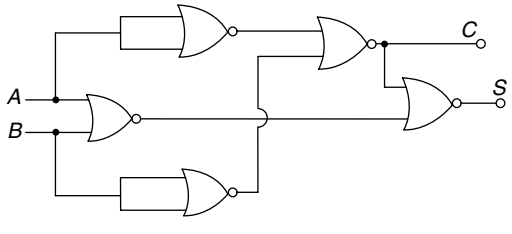


Half adder using NAND Logic

NOR logic

$$\begin{aligned} S &= A.\bar{B} + \bar{A}B \\ &= A\bar{B} + A\bar{A} + \bar{A}B + B\,\bar{B} \\ &= A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B}) \\ &= (A + B)(\bar{A} + \bar{B}) \\ &= \overline{\overline{A + B} + (\overline{\bar{A} + \bar{B}})} \end{aligned}$$

$$C = AB = \overline{\overline{A.B}} \quad = \overline{\bar{A} + \bar{B}}$$

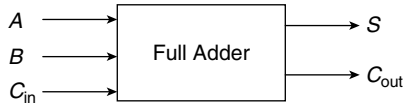


Half adder using NOR logic

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



Block Diagram

Truth Table

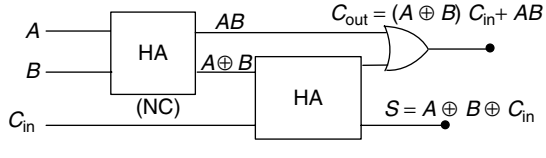
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\begin{aligned} S &= \bar{A}\,\bar{B}\,C_{in} + \bar{A}\,B\,\overline{C_{in}} + A\,\bar{B}\,\overline{C_{in}} + A\,B\,C_{in} \\ &= A \oplus B \oplus C_{in} \end{aligned}$$

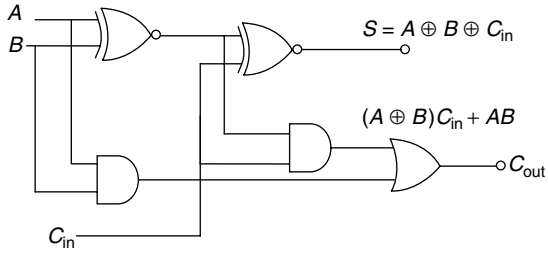
$$\begin{aligned} C_{out} &= \bar{A}\,B\,C_{in} + A\,\bar{B}\,C_{in} + A\,B\,\overline{C_{in}} + A\,B\,C_{in} \\ &= AB + (A \oplus B)\,C_{in} \\ &= AB + AC_{in} + BC_{in} \end{aligned}$$

Full adder can also be realized using universal logic gates, that is, either only NAND gates or only NOR gates, as in the following discussion.

Block Diagram of Full Adder by using *H.A*



Logic Diagram of Full Adder



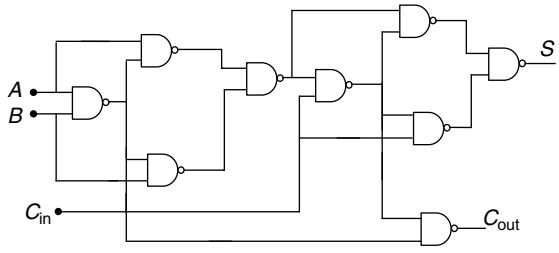
NAND logic

$$A \oplus B = \overline{\overline{A.\overline{A.B}} \quad \overline{B.\overline{A.B}}}$$

Therefore, $A \oplus B \oplus C_{in}$

Let $A \oplus B = x$, then $s = \overline{\overline{X.\overline{X\,C_{in}}} \quad \overline{C_{in}\;\overline{X.C_{in}}}}$

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



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.

[Since, 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 mentioned NAND gate implementation, all gates will become NOR gates and the output is dual of the sum and carry, but they are self-dual ($f^D = f$).

Therefore, output remains same, and only nine NOR gates are required for full adder structure similar to NAND gate circuit.

Half Subtractor

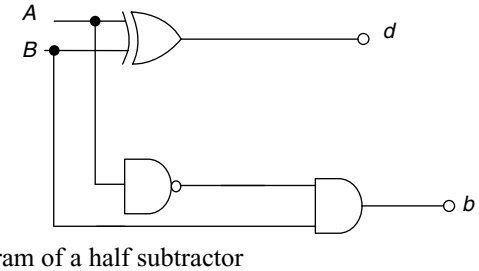
Half subtractor is an arithmetic circuit that will perform subtraction of one bit (subtrahend) from other bit (minuend), and the result gives difference and borrow each of one bit. The borrow output is logic 1 only if there is any subtraction of 1 from 0.

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

Truth Table

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

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

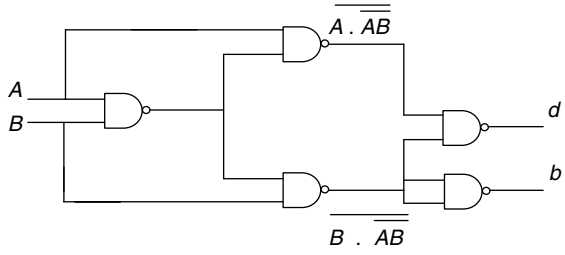


Logic Diagram of a half subtractor

A half-subtractor can also realized using universal logic either using only NAND gates or using only NOR gates, as explained in the following sections.

NAND logic

$$d = A \oplus B$$
$$= \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$
$$b = \bar{A}B] = B\left(\bar{A} + \bar{B}\right) = B\left(\overline{AB}\right) = \overline{\overline{B \cdot \overline{AB}}}$$

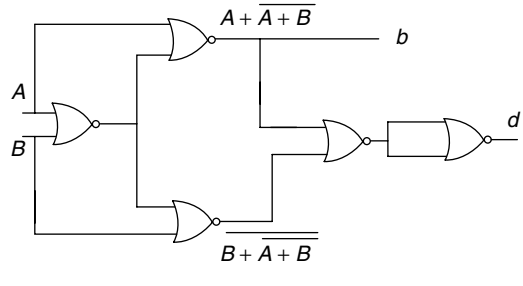


NOR Logic

$$d = A \oplus B$$
$$= A\bar{B} + \bar{A}B1$$
$$= A\bar{B} + B\bar{B} + \bar{A}B + A\bar{A}$$
$$= \bar{B}(A + B) + \bar{A}(A + B)$$
$$= \overline{B + \overline{A+B}} + \overline{A + \overline{A+B}}$$

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

Logic Diagram of Half Subtractor Using NOR Gate.



Full Subtractor

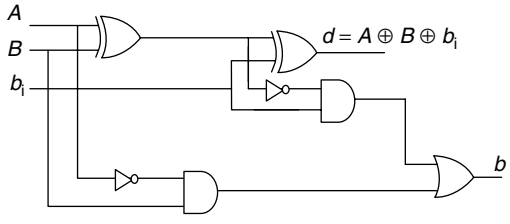
Full subtractor is an arithmetic circuit similar to half subtractor but it performs subtraction with borrow, it involves subtraction of three bits, minuend, subtrahend, and borrow-in and two outputs, that is, difference and borrow. The subtraction of 1 from 0 results in borrow to become logic 1. The 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

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

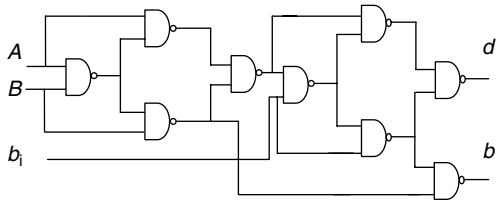
and
$$b = \bar{A}\,\bar{B}\,b_i + \bar{A}\,B\,\overline{b_i} + \bar{A}\,B\,b_i$$
$$= \bar{A}\,B + \left(\overline{A \oplus B}\right)b_i$$



NAND Logic

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

Logic Diagram of a Full Subtractor using NAND logic



NOR Logic

Outputs of full subtractor is also self-dual in nature; therefore, same circuit with all NAND gates replaced by NOR gates gives the NOR gate full subtractor. Nine NOR gates are required.

Solved Examples

Example 1

How many NAND gates are required for implementation of full adder and full subtractor?

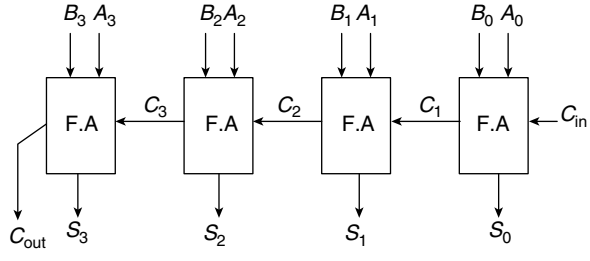
(A) 11, 10 (B) 11, 11 (C) 9, 9 (D) 9, 10

Solution

From the circuit diagrams in the previous discussion, full adder requires nine NAND gates, and full subtractor requires nine NAND gates. **Choice (C)**

Binary Adder

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



Four bit Parallel adder

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

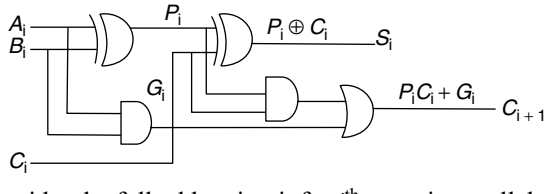
The bits are added with full adders, starting from the LSB position, to form the sum bits and carry bits. The longest propagation delay time in parallel adder is the time it takes the carry to propagate through the full adders.

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 n^{th} full adder is

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

Therefore, propagation delay increases with the 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 i^{th} stage in parallel adder, with two binary variables A_i and B_i. Input carry C_i are-

carry propagate (P_i) and carry generate (G_i).

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

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 $i = 1, 2.....$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0)$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

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 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:

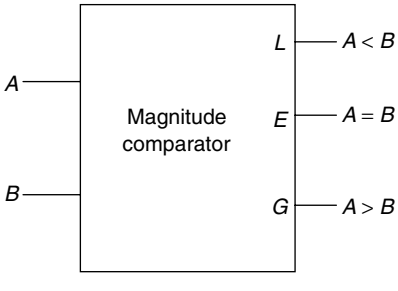
C_3, C_2, and C_1 can have equal time delays.

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

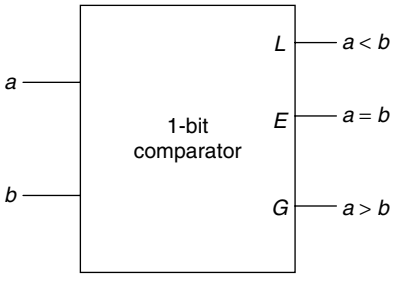
n-Bit Comparator

The comparison of two numbers is an operation that determines whether one number is greater than, lesser 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 $A > B$, $A = B$, $A < B$.



1–bit comparator will have only 1 bit input a, b

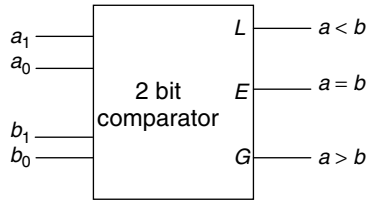


a	b	a < b	a = b	a > b
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

By considering minterms for each output.

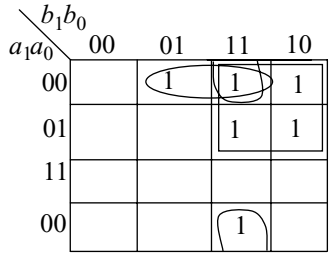
$$(a < b) = a' b$$
$$(a = b) = a' b' + a b = a \cdot b$$
$$(a > b) = a b'$$

Two-bit comparator will have two-bit inputs $a_1\ a_0$ 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

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



$$a < b = a_1{}^1\, a_0{}^1\, b_0 + a_0{}^1\, b_1\, b_0 + a_1{}^1\, b_1$$
$$L = \overline{a_1}\, b_1 + (a_1 \odot b_1)\, \overline{a_0}\, .b_0$$

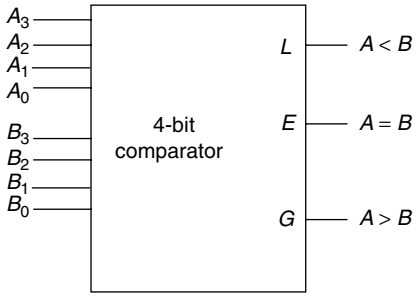
Similarly $a > b = a_0\, b_1{}^1\, b_0{}^1 + a_1\, a_0\, b_0{}^1 + a_1\, b_1{}^1$

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

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

Therefore, $(a = b) = (a_1 \odot b_1)\ (a_0 \odot b_0)$.

Four-bit comparator will compare two input numbers each of four bits $A_3\ A_2\ A_1\ A_0$ 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.

Therefore, 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 lesser 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 $A > B$, $A = 1$, $B = 0$

$$A < B = A_3^{\,1}\, B_3 + (A_3 \odot B_3)\, A_2^{\,1}\, B_2 + (A_3 \odot B_3)\, (A_2 \odot B_2)\, A_1^{\,1}\, B_1 + (A_3 \odot B_3)\, (A_2 \odot B_2)\, (A_1 \odot B_1)\, A_0^{\,1}\, B_0$$

$$A > B = A_3\, B_3^{\,1} + (A_3 \odot B_3)\, A_2\, B_2^{\,1} + (A_3 \odot B_3)\, (A_2 \odot B_2)\, A_1\, B_1^{\,1} + (A_3 \odot B_3)\, (A_2 \odot B_2)\, (A_1 \odot B_1)\, A_0\, B_0^{\,1}$$

Four-bit comparator will have total eight 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 combinations, $A > B = 1$.

Parity Bit Generator and Parity Bit Checker

When digital information is transmitted, it may not be received correctly by the receiver. To detect one 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 the case of even parity) or odd (in the 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 the 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 1's.

To transmit data $B_3B_2B_1B_0$ using even parity, we will transmit sequence $P\ B_3B_2B_1B_0$, 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 or checker as EX-OR of bits gives output 1 if there are odd number of 1's, else EXOR output is 0.

Odd parity generator or checker are complement of even parity generator or checker. The odd parity circuits check for the 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 analogue to digital converter is often in Gray code, before it can be processed in arithmetic unit and 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.

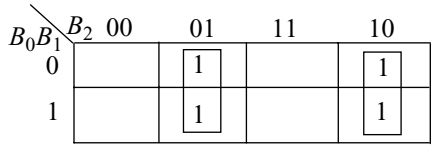
The output logic expression derived for code converter can be simplified by using the usual techniques including don't cares, if any present. For example, BCD code uses only codes from 0000 to 1001, remaining combinations are treated as don't care combinations. Similarly XS-3 uses only combinations from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

The relationship between the two codes is shown in truth table.

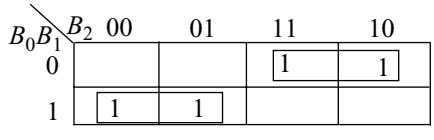
Decimal	B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

For conversion, we require to find out minimized functions of

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



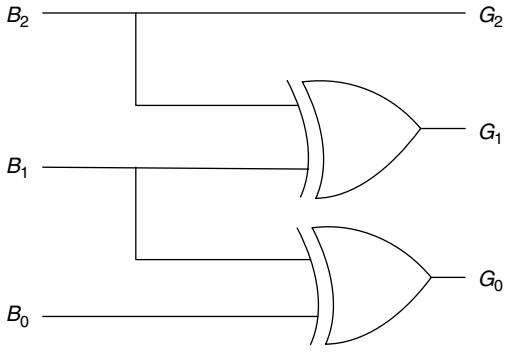
$$G_0(B_2, B_1, B_0) = B'_1B_0 + B_1B'_0 = B_1 \oplus B_0$$



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

B_0B_1 \ B_2	00	01	11	10
0				
1	1	1	1	1

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

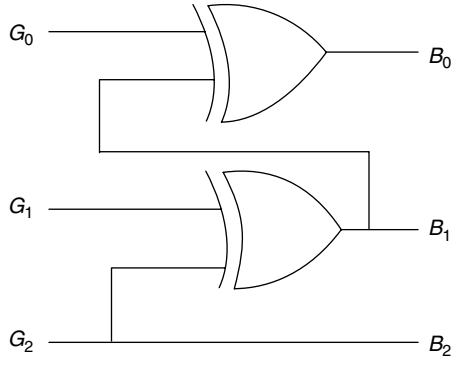


In similar fashion, we can derive n-bit binary to Gray code conversion as

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

Thus, conversion can be implemented by $(n - 1)$ XOR 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, we get

$B_i = G_n \oplus G_{n-1} \oplus G_{n-2} \ldots \oplus G_{i-1} \oplus G_i$
$B_n = G_n$ (MSB is same in Gray and binary). It also requires $(n - 1)$ XOR 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 need 4 bits to represent. The following table gives the relation between these codes. 84-2-1 is a weighted code, that is, 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, remaining minterms 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.

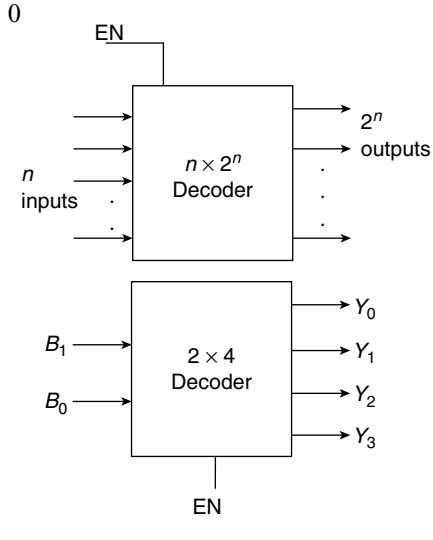


Figure 3.1 2 × 4 Decoder

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

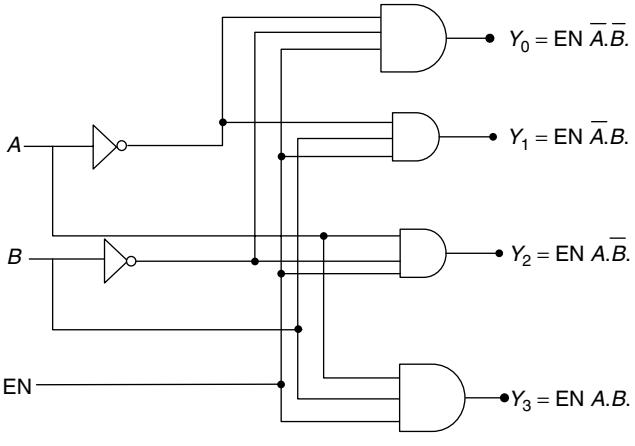
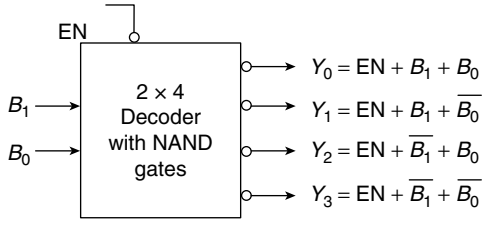


Figure 3.2 2 × 4 Decoder

Decoder outputs are implemented by AND gates, but realization of AND gates at circuit level is done by the NAND gates (universal gates). Therefore, the decoders available in IC form are implemented with NAND gates, that is, the outputs are in complemented form. Further, outputs are maxterms of the inputs rather than minterms 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.

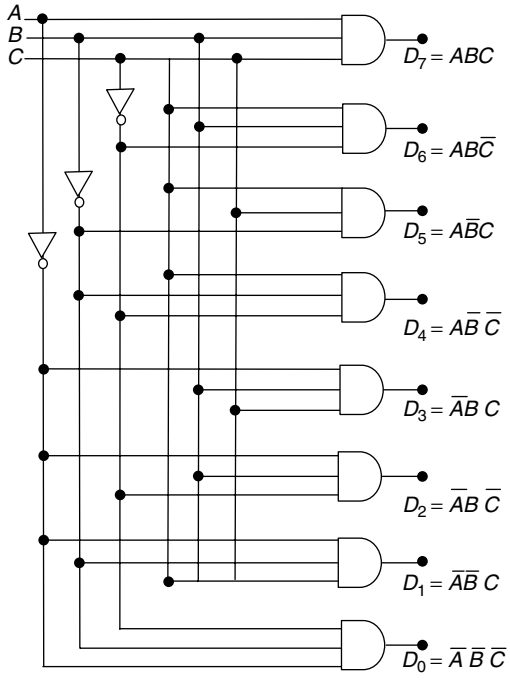
The decoder is enabled when EN is equal to 0. As shown in the truth table, only one output can be equal to 0 at any given time, all other outputs are equal to 1. The output whose value is equal to 0 represents the minterm selected by inputs enable.

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	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

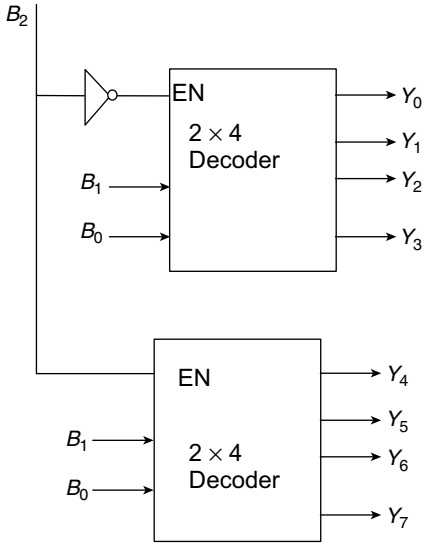
A 3 to 8 decoder has three input lines and eight output lines, based on the combination of inputs applied for the three inputs. One of the eight output line will be made logic 1, as shown in the truth table. Therefore, each output will have only one minterm.



Designing High Order Decoders from Lower Order Decoders

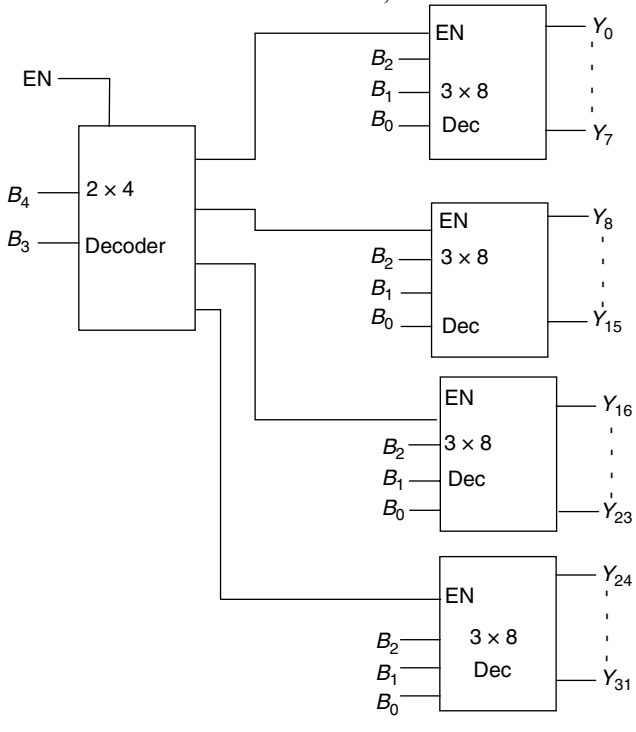
Decoder with enable input can be connected together to form larger decoder circuit.

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.



5 × 32 Decoder will have 5 inputs $B_4\ B_3\ B_2\ B_1\ B_0$. 3 × 8 Decoder will have eight outputs, and therefore, 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 above mentioned circuit.

Combinational Logic Implementation

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

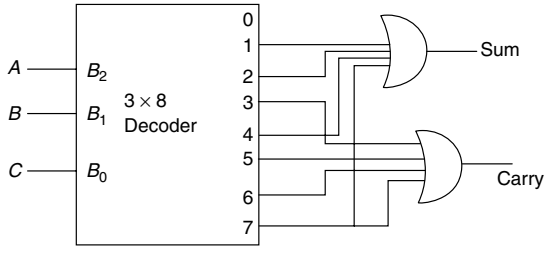


Figure 3.3 Implementation of full adder circuit with decoder.

Solution

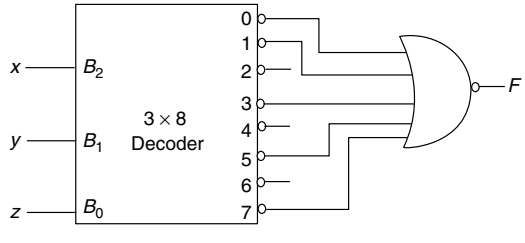
The 3 × 8 decoder generates the eight minterms for A, B, and C. The OR gate for output sum forms the logical sum of minterms 1, 2, 4, and 7. The OR gate for output carry forms the logical sum of minterms 3, 5, 6, and 7.

Example 4

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

(A) $x^1 y + z^1$ (B) $x^1 y^1 + z^1$

(C) $x^1 y^1 + z$ (D) $x^1 + y^1 z$



Solution

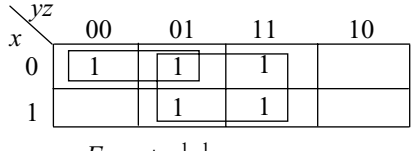
The output of decoder are in active low state. Therefore, we can express outputs as $\overline{Y_7}$, $\overline{Y_6}$ ..... $\overline{Y_0}$

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

Therefore, $F = \overline{\overline{Y_0}.\overline{Y_1}.\overline{Y_3}.\overline{Y_5}.\overline{Y_7}}$

$= Y_0 + Y_1 + Y_3 + Y_5 + Y_7 = \Sigma\,(0, 1, 3, 5, 7)$

By using K-maps, we get



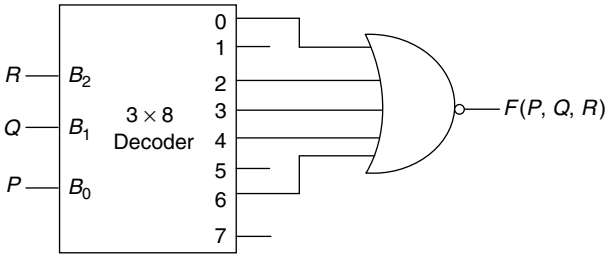
$F = z + x^1 y^1$ **Choice (C)**

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

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

Therefore, $F = \overline{Y_0 + Y_2 + Y_3 + Y_4 + Y_6} = \overline{Y_0}.\overline{Y_2}.\overline{Y_3}.\overline{Y_4}.\overline{Y_6}$

$Y_0, Y_1 ..... Y_7$ indicate minterms, whereas $\overline{Y_0}$, $\overline{Y_1}$ ..... $\overline{Y_7}$ are maxterms.

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

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

By using K-map, we get

R \ QP	00	01	11	10
0	0		0	0
1	0			0

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

Encoders

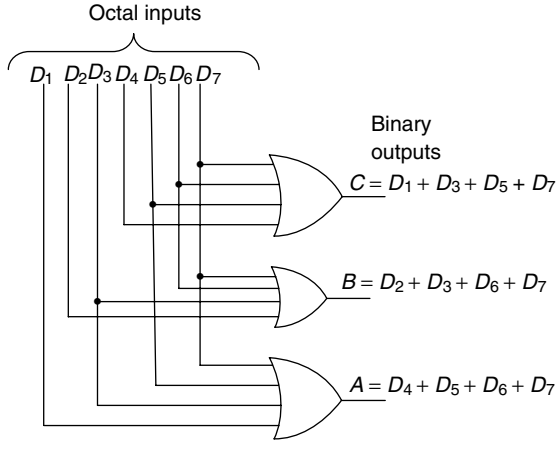
It is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n (or fewer) input lines and n output lines. It is also known as an octal to binary converter.

Consider an 8-to-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

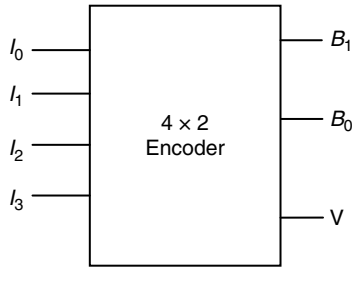
Logic Diagram



Priority Encoder

A priority encoder is an encoder circuit that includes the priority function. When two or more inputs are present, the input with high 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_1B_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, and I_2 has the next priority level. Input I_0 has the lowest priority level. The Boolean expressions for output B_1B_0 are

$$B_1 = I_3 + \overline{I_3}\,I_2 = I_3 + I_2$$

$$B_0 = I_3 + \overline{I_3}\,\overline{I_2}\,I_1 p1$$

$$= I_3 + \overline{I_2}\,I_1$$

$$V = I_3 + I_2 + I_1 + I_0$$

MULTIPLEXER

A multiplexer 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.

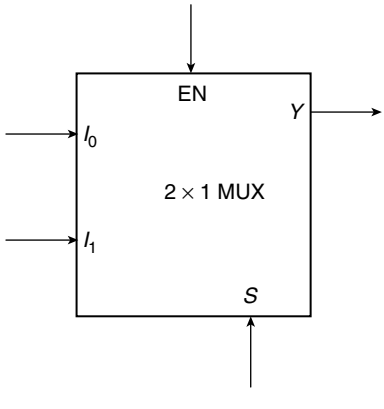
2^n inputs will be controlled by n selection lines and multiplexer will have 1 output, and we denote it as $2^n \times 1$ multiplexer (data selector).

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 two 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

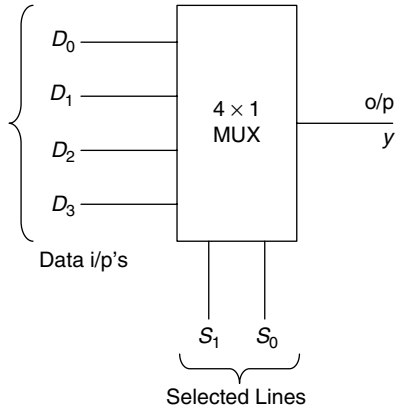


The output equation of 2 × 1 multiplexer is $Y = \text{EN}\,(I_0\overline{S} + I_1 S)$.

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 $Y = \overline{EN}\,(I_0\overline{S} + I_1 S)$.

4 × 1 Multiplexer

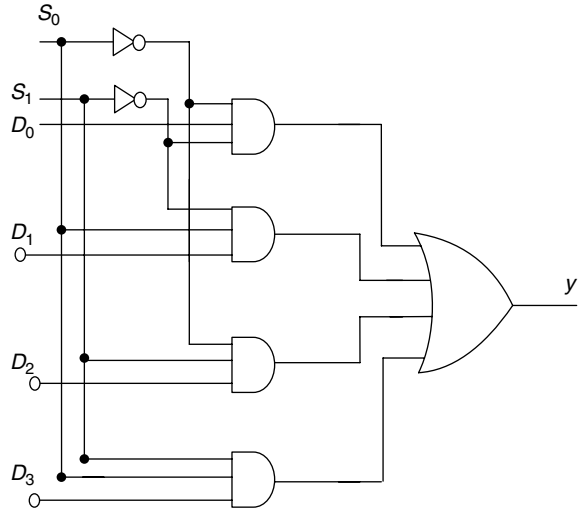


If a binary zero $S_1 = 0$ and $S_0 = 0$ as applied to the data select line, the data input D_0 appears on the data output line and so on.

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

$$y = \overline{S_1}\,\overline{S_0}\,D_0 + \overline{S_1}\,S_0\,D_1 + S_1\,\overline{S_0}\,D_2 + S_1\,S_0\,D_3$$

Logic Diagram

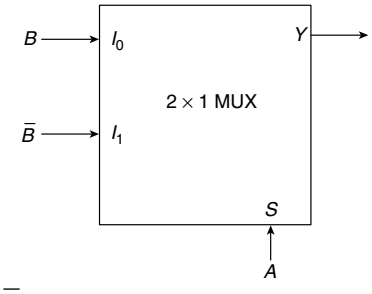


For 8 × 1 multiplexer with eight inputs from I_0 to I_7 based on selection inputs S_2, S_1, and S_0, the equation for output

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

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

Basic Gates by using MUX



$Y = \overline{A}B + A\overline{B}$ = XOR gate, we can interchange inputs A and B also. By interchanging inputs I_0 and I_1, then $Y = \overline{A}\overline{B} + AB$, XNOR gate.

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

Example 6

If $I_0 = 1$, $I_1 = 0$, and $S = A$, then Y is?

Solution

$= (I_0\overline{S} + I_1 S) = \overline{A}$. It Implements NOT gate.

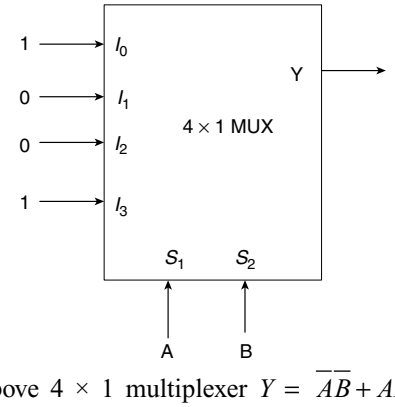
Example 7

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

Solution

$Y = \overline{AB} = \overline{A} + \overline{B} = \overline{A} + A\overline{B} = 1.\overline{A} + \overline{B}.A$

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



For the above 4 × 1 multiplexer $Y = \overline{A}\overline{B} + AB$ = XNOR 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 (borrow) to be implemented by using multiplexer. Full subtractor borrow (B) is a function of three inputs X, Y, and Z. The truth table is as follows:

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 borrow by using 8 × 1 multiplexer, connect the three variables X, Y, and Z directly to selection lines of the multiplexer, and connect the corresponding values of B to inputs, that is, for $I_0 = 0$, $I_1 = 1$, $I_2 = 1$… as per above mentioned truth table.

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

To implement the function by using 2 × 1 multiplexer, connect one variable as selection line (in this case, consider X) and write output (B) in terms of other variables; for $X = 0$, output B varies as $B = Y + Z$, and therefore, connect $I_0 = Y + Z$. For $X = 1$, output B varies as $B = YZ$, connect $I_1 = YZ$.

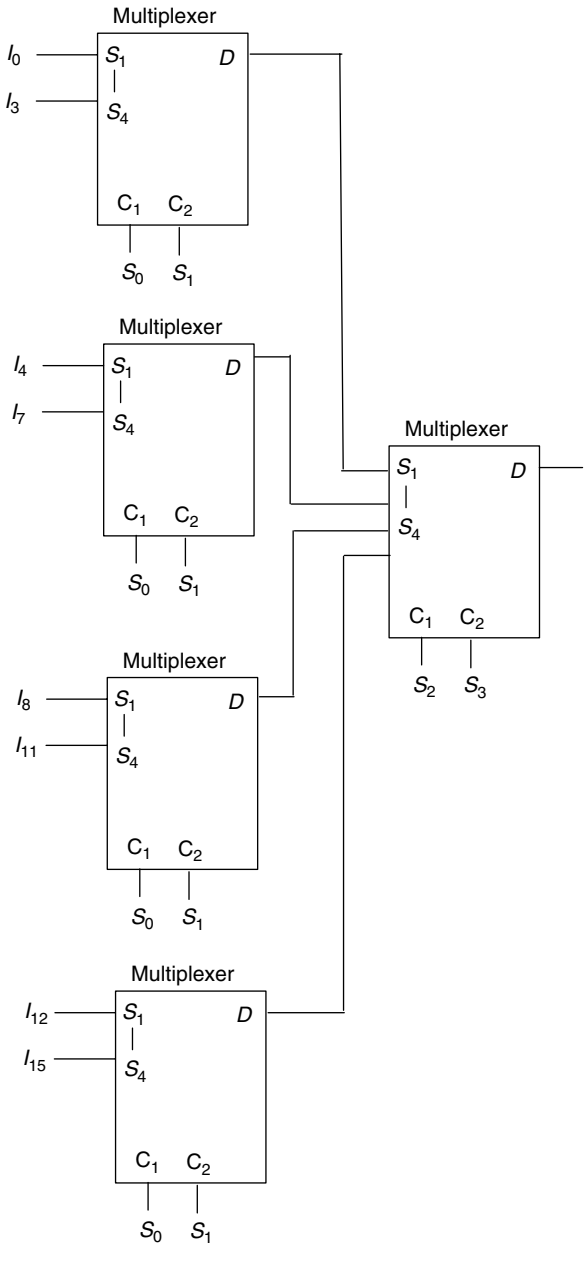
N-variable function can be implemented by using 2^{N-1} × 1 multiplexer without any extra hardware.

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 four input lines, and two selection lines. Therefore, 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). Therefore, total of 5, 4 × 1 multiplexers are required to implement 16 × 1 MUX.

In 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.

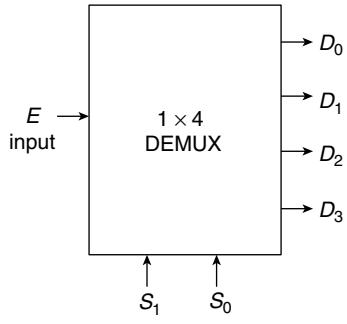


Demultiplexer

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

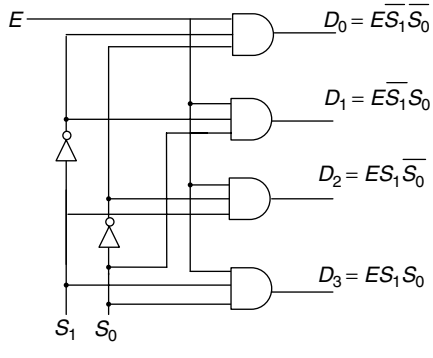
The other name for demultiplexer is data distributor, as it receives information on a single line and distributes it to a possible 2^n output lines, where n is the number of selection lines, and value of n selects the line.

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).

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: Random Access Memory (RAM) and Read Only Memory (ROM)

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

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).

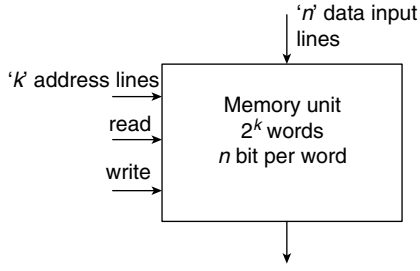
A PLD is an integrated circuit with internal logic gates connected through electronic paths that behave similar to fuses.

Random Access Memory

If the time taken to read or write data from any memory location is same, then we call it as Random Access Memory. While 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 to 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, 4 K × 16 memory has 12 bits in address and 16 bits in each word. Similarly, 32 K × 16 memory has 15 bits as address lines and 16 bits in each word.

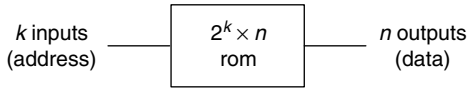
The memory enable (chip select) is used to enable the particular memory chip in a multichip implementation of a large memory.

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

An ROM is a type of memory that stores data permanently or semi-permanently, that is, data can be erasable. As the name indicates, data stored in ROM can only be read. The data that 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.

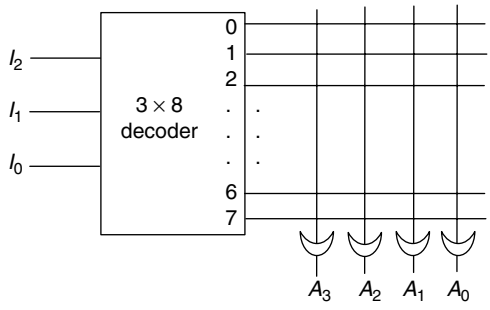
ROM also have address inputs similar to RAM to access one of the memory location, and to read the data from that memory location, data output lines are available.



The number of words in an 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 8 words of 4 bit each. There are three input lines from the binary numbers 0 to 7 for address.

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



Each OR gate must be considered as having 8 inputs. Each output of the decoder is connected to one of the inputs of each OR gate. Since each OR gate has 8 input connections and there are 4 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.

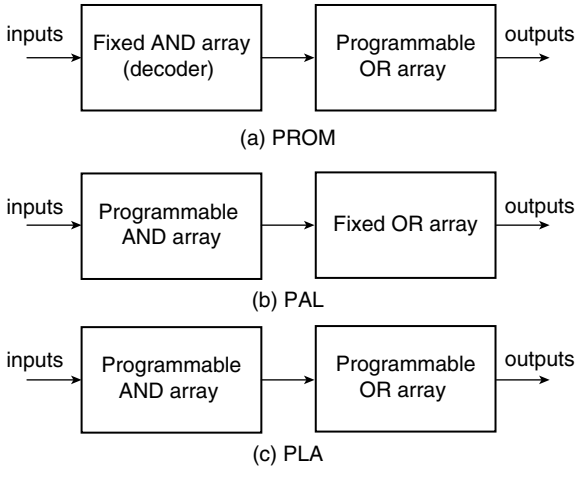
These intersections are programmable. A programmable connection between two lines is logically equivalent to a switch that can be altered to be either closed or open.

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 separately as the output of a Boolean function expressed as a sum of minterms.

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.



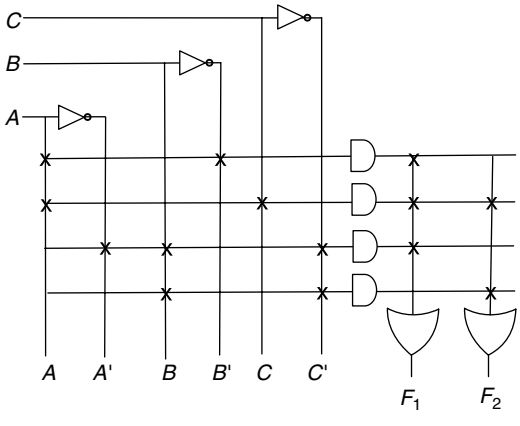
The PROM has fixed AND array constructed as a decoder and a programmable OR array. The programmable OR gates implement the Boolean function in sum of minterms form.

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 does not generate all the minterms.

PLA for the functions

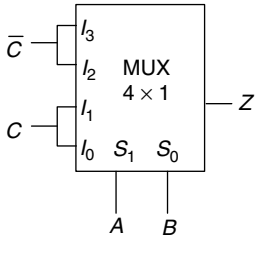
$$F_1 = AB^1 + AC + A^1\,B\,C^1$$

$F_2 = AC + BC^1$ is shown in the following figure.



Example 8

The multiplexer shown in the figure is a 4:1 multiplexer. Find 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, $Z = \overline{A}\,\overline{B}\,C + \overline{A}BC + AB\overline{C} + A\,\overline{B}\,\overline{C}$

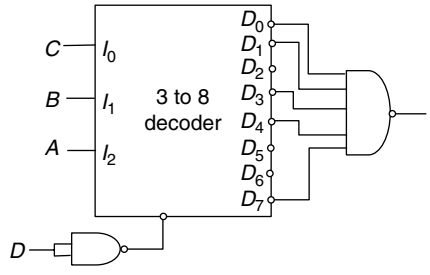
$$= \overline{B}(\overline{A}C + A\overline{C}) + B(\overline{A}C + A\overline{C})$$

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

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

Example 9

The logic circuit shown in figure implements



Solution

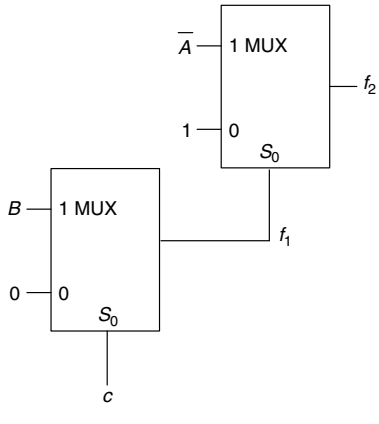
$$z = D\left(\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}BC + A\,\overline{B}\,\overline{C} + ABC\right)$$

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

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

Example 10

The network shown in figure implements.



Solution

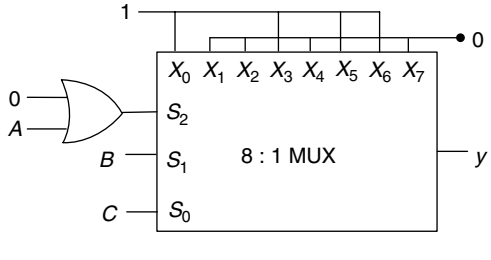
$f_1 = \overline{C}0 + CB = CB\ , f_1 = CB$

$F_2 = \overline{f_1} + f_1\overline{A} = \overline{A}CB + \overline{CB} = \overline{A} + \overline{CB} = \overline{A} + \overline{C} + \overline{B} = \overline{ABC}.$

Therefore, NAND gate.

Example 11

In the TTL circuit in the 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

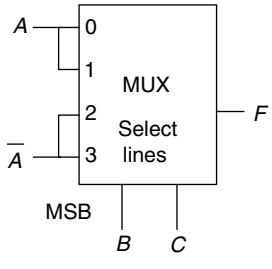
$$S_2 = A, S_1 = B, S_0 = C$$

$S_2(A)$	$S_1(B)$	$S_0(C)$	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

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

Example 12

Find the logic realized by the adjoining circuit.



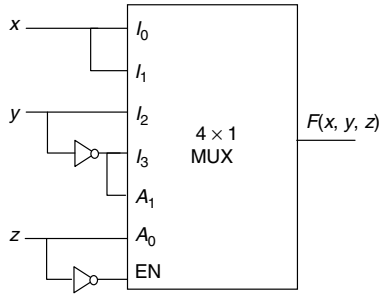
Solution

$$F = \overline{B}\overline{C}A + \overline{B}CA + B\overline{C}\overline{A} + BC\overline{A}$$

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

Example 13

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



Solution

$$A_1 = \overline{y}\,.\,A_0 = z,\ \text{EN} = \overline{z}$$

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

$$f(x, y, z) = \text{S.I} = (xy + 0 + \overline{y}z).\text{EN} = xy.\overline{z}$$

Exercises

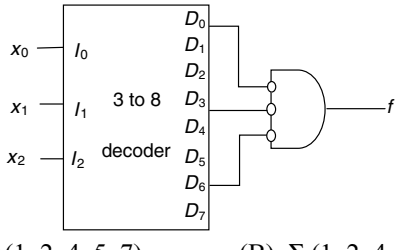
Practice Problem I

Direction 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, XNOR
(C) 6, XOR (D) 5, XOR

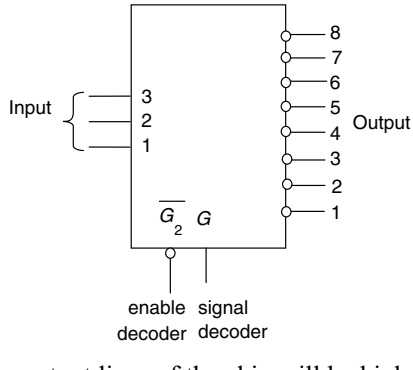
2. The number of 4-line to 16-line decoder required to make an 8-line to 256-line decoder is
(A) 16 (B) 17 (C) 32 (D) 64

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



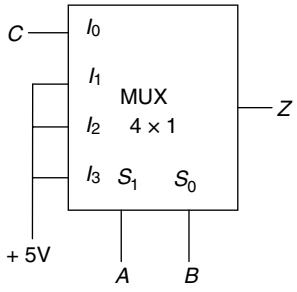
(A) $\pi(1, 2, 4, 5, 7)$ (B) $\Sigma(1, 2, 4, 5, 7)$
(C) $\Sigma(0, 3, 6)$ (D) $\pi(0, 2, 3, 6)$

4. A 3-to-8 decoder is shown in the following figure



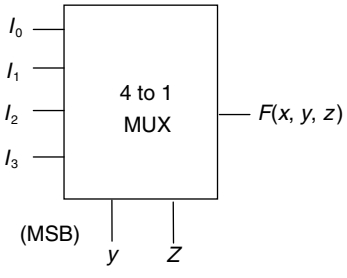
All the output lines of the chip will be high except pin 8, when all the inputs 1, 2, and 3
(A) are high; and G, G_2 are low
(B) are high; and G is low G_2 is high
(C) are high; and G, G_2 are high
(D) are high; and G is high; G_2 is low

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



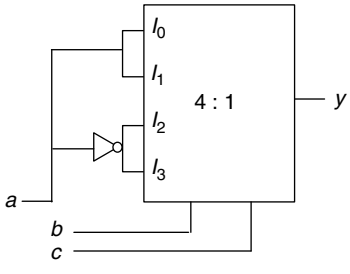
(A) $A\,B\,C$ (B) $A \oplus B \oplus C$
(C) $A\,\Theta\,B\,\Theta\,C$ (D) $A + B + C$

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



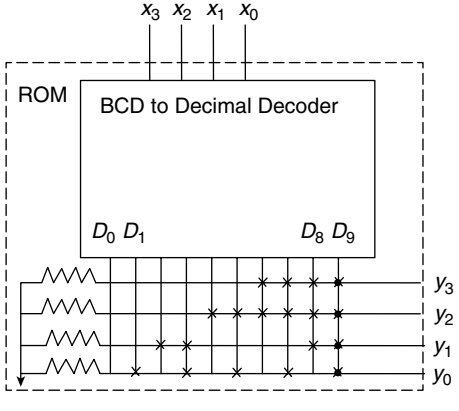
(A) $I_0 = X, I_1 = 0, I_2 = X\ I_3 = X$
(B) $I_0 = 0, I_1 = 1, I_2 = Y_1, I_3 = X$
(C) $I_0 = X, I_1 = 1, I_2 = 0, I_3 = X$
(D) $I_0 = X, I_1 = 0, I_2 = X, I_3 = Z$

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 XOR gates with a and b inputs
(D) two input XNOR gate with b and c as inputs.

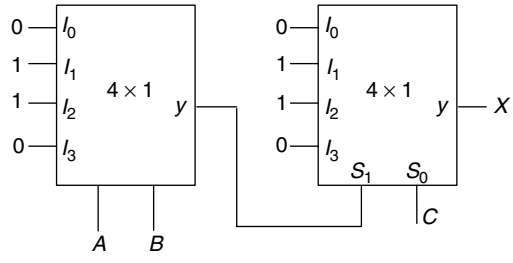
8. If the input x_3, x_2, x_1, x_0 to the ROM in the figure are 8421 BCD numbers, then the outputs y_3, y_2, y_1, y_0 are



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

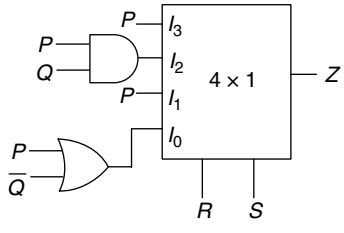
9. A 4-bit parallel full adder without input carry requires
(A) 8 H.A, 4 OR gates (B) 8 H.A, 3 OR gates
(C) 7 H.A, 4 OR gates (D) 7 H.A, 3 OR gates

10. In the circuit, find X.



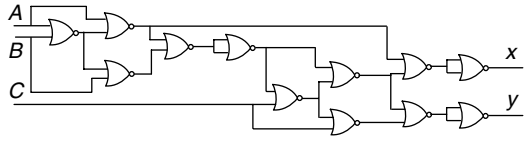
(A) $A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\,\bar{B}C + ABC$
(B) $\bar{A}BC + A\bar{B}C + AB\bar{C} + \bar{A}\,\bar{B}\,\bar{C}$
(C) $AB + BC + AC$
(D) $\bar{A}\,\bar{B} + \bar{B}\,\bar{C} + \bar{A}\,\bar{C}$

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

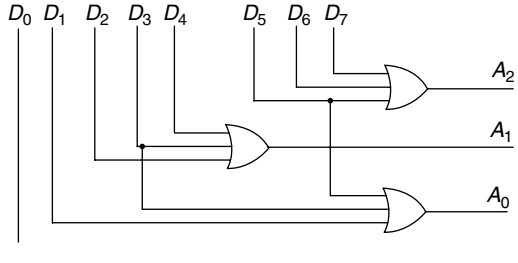
12. Which function is represented by the given circuit?



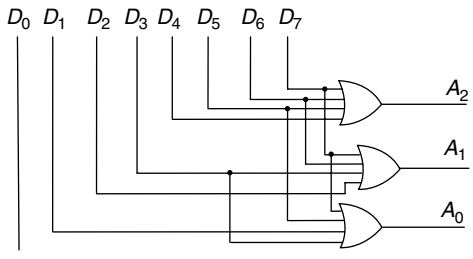
(A) full adder (B) full subtractor
(C) comparator (D) Parity generator

13. Which of the following represents octal to binary encoder?

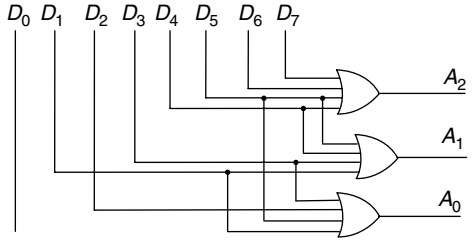
(A)



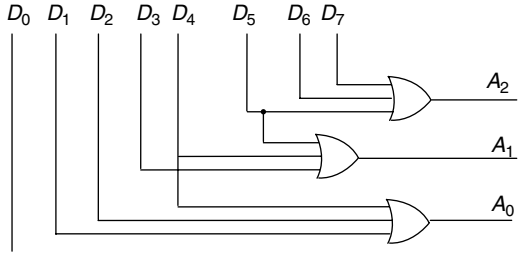
(B)



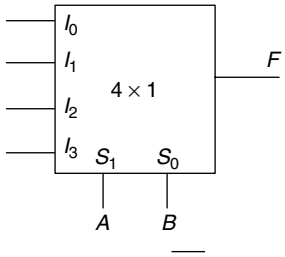
(C)



(D)



14. For an MUX to function as a full adder, what should be the input provided to the I_0, I_1, and I_2I_3 if the A and B are the select lines?



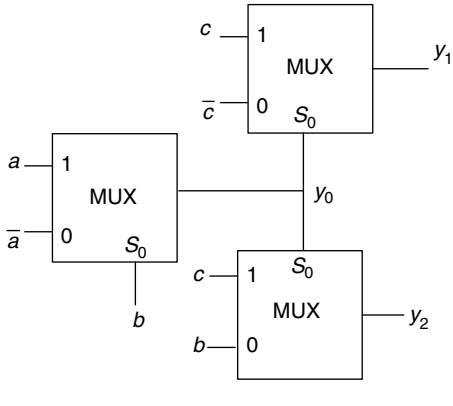
(A) $I_0 = I_1 = C_{in}; I_2 = I_3 = \overline{C_{in}}$

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

(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

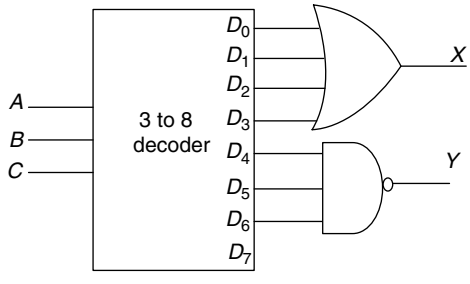
(C) full subtractor (D) half subtractor

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

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

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

17. The function implemented by decoder.



(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, and 1010 do not occur. For rest of the status, relay should be OFF. The minimized Boolean expression notifying the relationship is

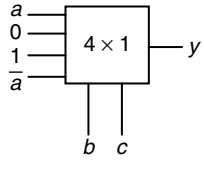
(A) $BC + ACD$

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

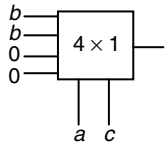
(C) $BD + AC$

(D) $AB + CD$

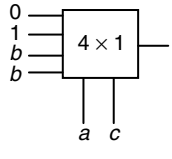
19. If a function has been implemented using MUX as shown in the figure, implement the same function with a and c as the select lines



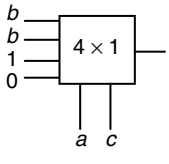
(A)



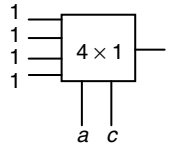
(B)



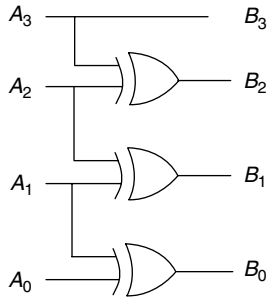
(C)



(D)

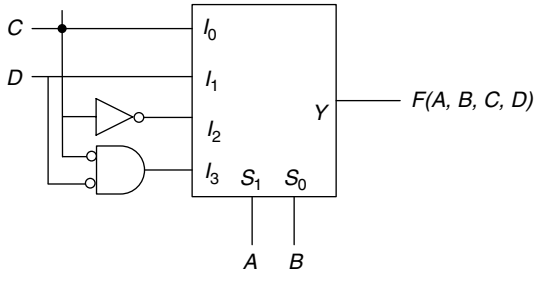


20. Identify the circuit that is used to convert one code to another.



(A) Binary to Gray (B) Gray to Binary
(C) Gray to XS-3 (D) Gray to 8421

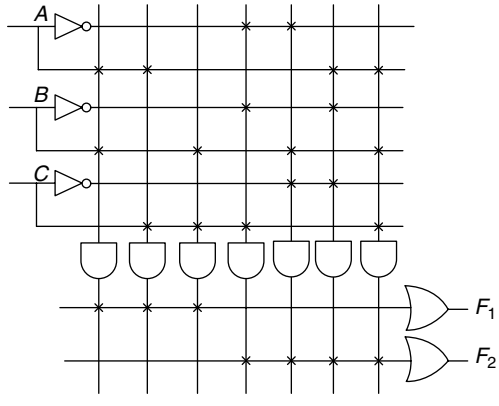
21. The Boolean function realized by logic circuit is



(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 received a 4 bit excess 3 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) $CD + AD + AC + \overline{A}\,\overline{C}\,\overline{D}$

Direction for questions 23 and 25:



23. Identify the function implemented.
(A) $f_1(A, B, C) = AB + BC + AC$
(B) $f_1(A, B, C) = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C}$
(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

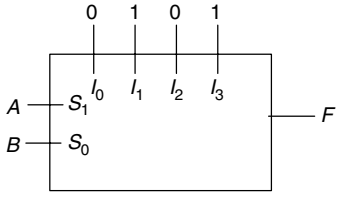
25. The circuit implemented using the PLA in the above mentioned figure is
(A) full adder (B) full subtractor
(C) half adder (D) half subtractor

Practice Problem 2

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

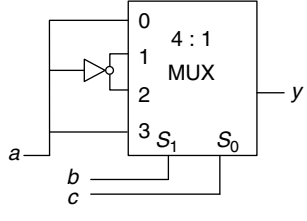
1. For a binary half subtractor having two input A and B, the correct set of logical expression for the outputs $D(= A - B)$ and X (borrow) are
(A) $D = AB + \overline{A}\,\overline{B}, X = \overline{A}B$
(B) $D = \overline{A}\,\overline{B} + A\overline{B}, A\overline{B}, X = \overline{A}B$
(C) $D =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



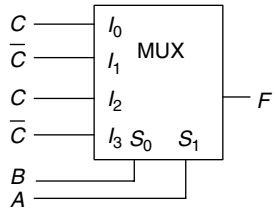
(A) $\overline{A}B$ (B) B
(C) (D) $A\overline{B} + \overline{A}B$

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 subtractor
(D) All 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 mentioned 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)

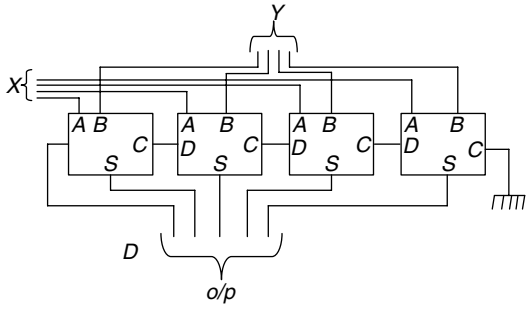
(A) $S = AB(\overline{A+B})$ (B) $S = (A+B)\overline{AB}$

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

9. MUX implements which of the following logic?
(A) NAND–XOR (B) AND–OR
(C) OR–AND (D) XOR–NOT

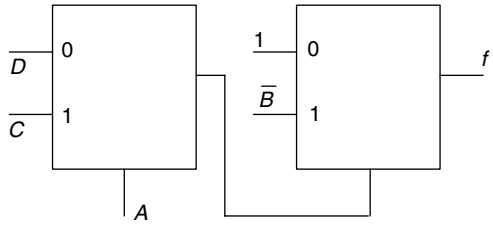
10. A DEMUX can be used as a
(A) comparator (B) encoder
(C) decoder (D) adder

11. If we have inputs as A, B, and C and output as S and D, we are given that $S = A \oplus B \oplus C$. $D = BC + \overline{A}B + \overline{A}C$, then 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. Find the Boolean function f implemented in the figure using 2-input multiplexers.



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

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

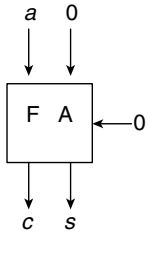
(C) $\overline{B} + A\overline{C} + \overline{A}\,\overline{D} + \overline{D}\,\overline{C}$

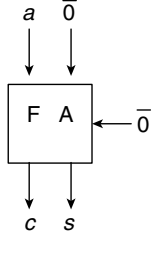
(D) $AC + AD + A + B$

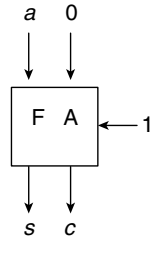
13. The carry generate and carry propagate function of the look-ahead carry adder is
(A) $CG = A + B, CP = A \oplus B$
(B) $CG = A \oplus B, CP = A + B$
(C) $CG = AB, CP = A \oplus B$
(D) $CG = AB, CP = A + B$

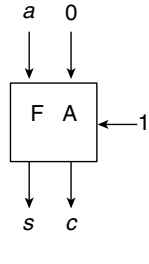
14. If we have a comparator and if E represents the condition for equality, that is, $(A_n \oplus B_n)$, if A_n and B_n are to be compared, then the expression $A_3\overline{B}_3 + E_3A_2\overline{B}_2 + E_3E_2A_1\overline{B}_1 + E_3E_2E_1A \cdot \overline{B}$ represents which of the condition for a 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 incrementer, which of the circuit configurations must be used?

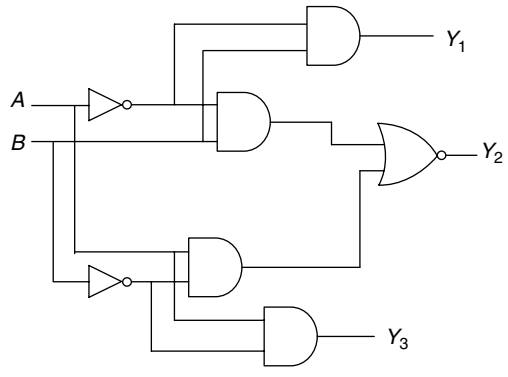
(A)


(B)


(C)


(D)


16. Identify the circuit.



(A) half adder
(B) full adder
(C) one-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) $2n \times 1$ (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) $\overline{A.\overline{AB}.B.\overline{AB}}$
(C) $\overline{\overline{A.\overline{AB}}.\overline{B.\overline{AB}}}$ (D) $\overline{\overline{A.\overline{AB}}\,B.\overline{AB}}$

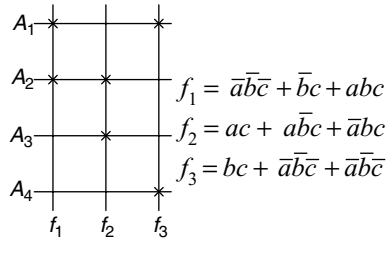
20. For a DEMUX to act as a decoder, what is the required condition?
(A) input should be left unconnected and select lines behave as a input to decoder
(B) input should be always 0 and select line behave as inputs to decoder
(C) Both are same
(D) input should become enable and select lines behave as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

21. For a full subtractor, which of the combination will give the difference?
(A) $\overline{\overline{(A \oplus B)(A \oplus B)b_1}}\ \overline{b_1(A \oplus B)b_1}$
(B) $\overline{\overline{B \cdot \overline{AB} \cdot b_i (A \oplus B)}}$
(C) $\overline{\overline{A} + \overline{B}} + \overline{\overline{b_1} + \overline{A \oplus B}}$
(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 an OR gate array. When the input to the gate array must be product of one or two variables, the terms A_1, A_2, and A_3 should be

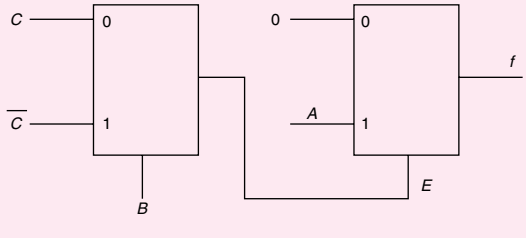


(A) $ab, ac, \bar{a}\bar{b}, bc$ (B) $ac, \bar{a}b, \bar{a}\bar{b}, b\bar{c}$
(C) $\bar{a}\bar{b}, ac, \bar{a}b, bc$ (D) None of these

Previous Years' Questions

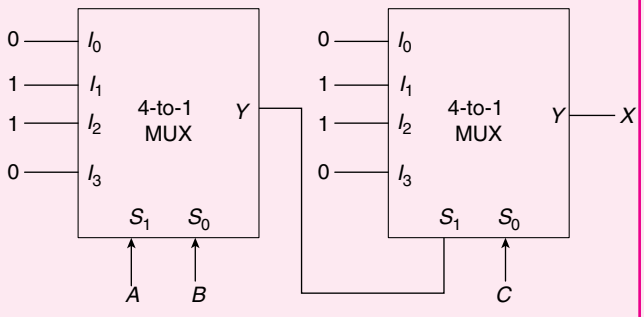
1. A Boolean function f of two variables X and Y is defined as follows.
$F(0, 0) = f(0, 1) = f(1, 1) = 1; f(1, 0) = 0$
Assuming complements of X and Y are not available, a minimum cost solution for realizing f using only 2-input NOR gates and 2-input OR gates (each having unit cost) would have a total cost of **[2004]**
(A) 1 unit (B) 4 unit
(C) 3 unit (D) 2 unit

2. The Boolean function f implemented in figure using 2-input multiplexers is **[2005]**



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

3. In the following circuit, X is given by **[2007]**



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

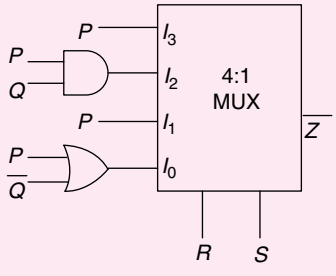
(B) $X = \overline{A}\,BC + A\,\overline{B}C + A\,B\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C}$

(C) $X = AB + BC + AC$

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

4. For the circuit shown in the following figure, $I_0 - I_3$ are inputs to the 4:1 multiplexer. R(MSB) and S are control bits

[2008]



The output Z can be represented by

(A) $PQ + P\overline{Q}S + \overline{Q}\,\overline{R}\,\overline{S}$

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

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

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

5. What are the minimum number of 2 to 1 multiplexers required to generate a 2-input AND gate and a 2-input Ex-OR gate? **[2009]**

(A) 1 and 2 (B) 1 and 3

(C) 1 and 1 (D) 2 and 2

Direction for questions 6 and 7:

Two products are sold from a vending machine, which has two push buttons P_1 and P_2. When a button is pressed, the price of the corresponding product is displayed in a 7-segment display.

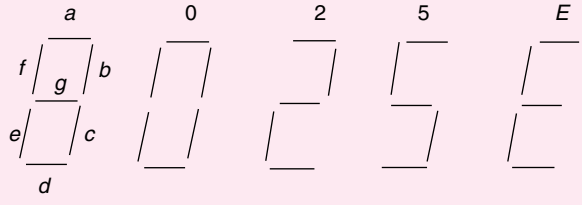
If no buttons are pressed, '0' is displayed, signifying 'Rs. 0'.

If only P_1 is pressed, '2' is displayed, signifying 'Rs. 2'.

If only P_2 is pressed, '5' is displayed, signifying 'Rs. 5'.

If both P_1 and P_2 are pressed, 'E' is displayed, signifying 'Error'.

The names of the segments in the 7-segment display, and the glow of the display for '0', '2', '5', and 'E', are shown in the following figure.



Consider

(i) push button pressed/not pressed is equivalent to logic 1/0, respectively

(ii) a segment glowing/not glowing in the display is equivalent to logic 1/0, respectively

6. If segments a to g are considered as functions of P_1 and P_2, then which of the following is correct? **[2009]**

(A) $g = \overline{P_1} + P_2, d = c + e$ (B) $g = P_1 + P_2, d = c + e$

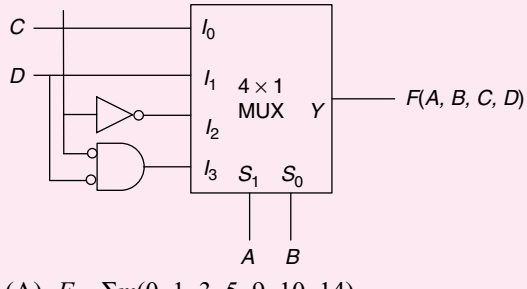
(C) $g = \overline{P_1} + P_2, e = b + c$ (D) $g = P_1 + P_2, e = b + c$

7. What are the minimum numbers of NOT gates and 2-input OR gates required to design the logic of the driver for this 7-segment display? **[2009]**

(A) 3 NOT and 4 OR (B) 2 NOT and 4 OR

(C) 1 NOT and 3 OR (D) 2 NOT and 3 OR

8. The Boolean function realized by the logic circuit shown is **[2010]**



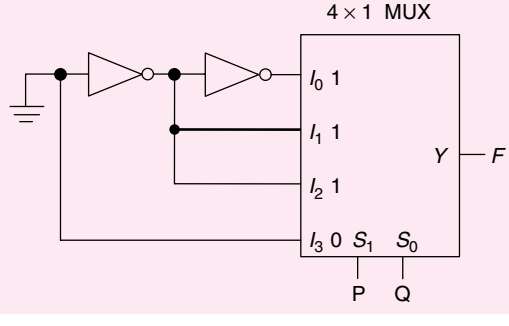
(A) $F = \Sigma m(0, 1, 3, 5, 9, 10, 14)$

(B) $F = \Sigma m(2, 3, 5, 7, 8, 12, 13)$

(C) $F = \Sigma m(1, 2, 4, 5, 11, 14, 15)$

(D) $F = \Sigma m(2, 3, 5, 7, 8, 9, 12)$

9. The logic function implemented by the following circuit is (ground implies a logic '0') **[2011]**



(A) $F = \text{AND}(P, Q)$ (B) $F = \text{OR}(P, Q)$
(C) $F = \text{XNOR}(P, Q)$ (D) $F = \text{XOR}(P, Q)$

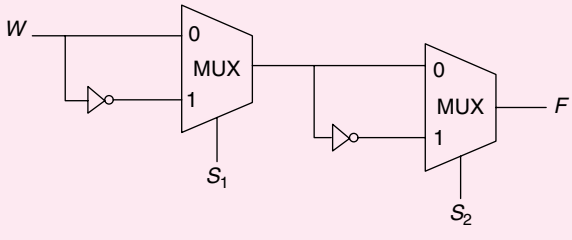
10. The output Y of a 2-bit comparator is logic 1 whenever the 2-bit input A is greater than the 2-bit input B. The number of combinations for which for the output is logic 1 is **[2012]**

(A) 4 (B) 6
(C) 8 (D) 10

11. In a half-subtractor circuit with X and Y as inputs, the borrow (M) and difference ($N = X - Y$) are given by **[2014]**

(A) $M = X \oplus Y, N = XY$
(B) $M = XY, N = X \oplus Y$
(C) $M = \overline{X}Y, \; N = X \oplus Y$
(D) $M = X\overline{Y}, \; N = \overline{X \oplus Y}$

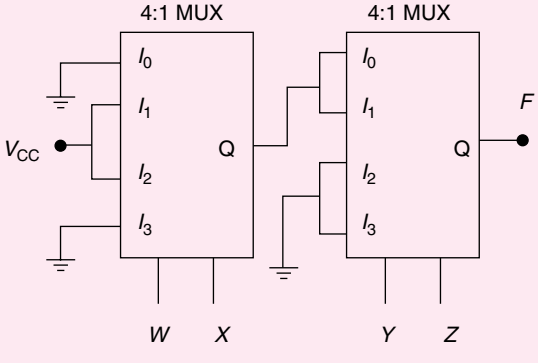
12. Consider the multiplexer-based logic circuit shown in the figure. **[2014]**



Which one of the following Boolean functions is realized by the circuit?

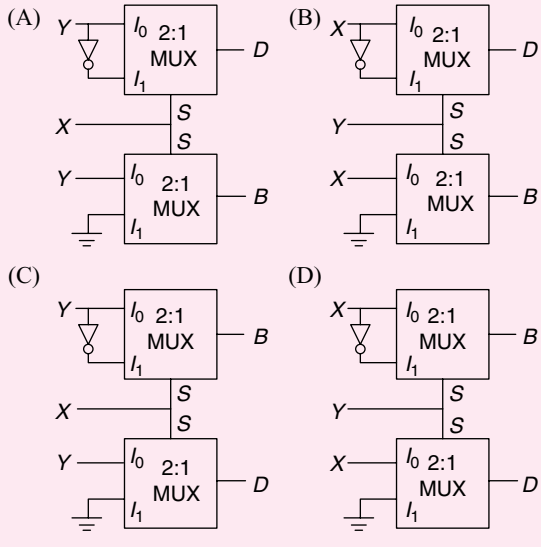
(A) $F = W\overline{S_1}\;\overline{S_2}$ (B) $F = WS_1 + WS_2 + S_1S_2$
(C) $F = \overline{W} + S_1 + S_2$ (D) $F = W \oplus S_1 \oplus S_2$

13. In the circuit shown, W and Y are MSBs of the control inputs.
The output F is given by **[2014]**

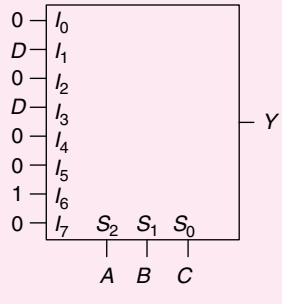


(A) $F = W\overline{X} + \overline{W}X + \overline{Y}\,\overline{Z}$
(B) $F = W\overline{X} + \overline{W}X + \overline{Y}Z$
(C) $F = W\overline{X}\;\overline{Y} + \overline{W}X\overline{Y}$
(D) $F = \left(\overline{W} + \overline{X}\right)\overline{Y}\;\overline{Z}$

14. If X and Y are inputs and the difference ($D = X - Y$) and the borrow (B) are the outputs, which one of the following diagrams implements a half subtractor? **[2014]**

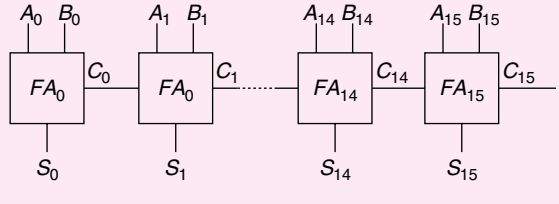


15. An 8-to-1 multiplexer is used to implement a logical function Y, as shown in the figure. The output Y is given by **[2014]**

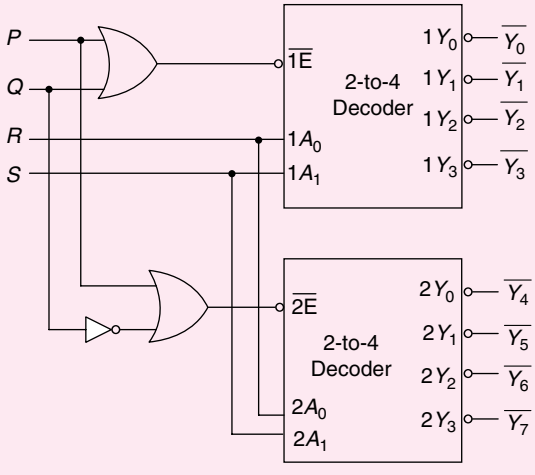


(A) $Y = A\overline{B}C + A\overline{C}D$ (B) $Y = \overline{A}BC + A\overline{B}D$
(C) $Y = AB\overline{C} + \overline{A}CD$ (D) $Y = \overline{A}\,\overline{B}D + A\overline{B}C$

16. A 16-bit ripple carry adder is realized using 16 identical full adders (FA), as shown in the figure. The carry-propagation delay of each FA is 12 ns and the sum-propagation delay of each FA is 15 ns. The worst case delay (in ns) of this 16-bit adder will be_____. **[2014]**

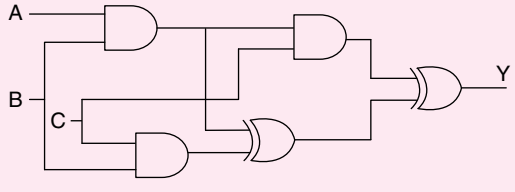


17. A 1-to-8 demultiplexer with data input Din, address inputs S_0, S_1, S_1 (with S_0 as the LSB) and $\overline{Y}_0$ *to* $\overline{Y}_7$ as the eight demultiplexed outputs, is to be designed using two 2-to-4 decoders (with enable input $\overline{E}$ and address inputs A_0 and A_1) as shown in the figure. Din, S_0, S_1, and S_2 are to be connected to P, Q, R, and S, but not necessarily in this order. The respective input connections to P, Q, R, and S terminals should be **[2015]**



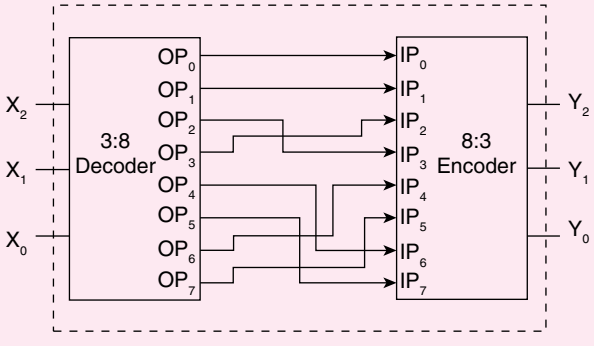
(A) S_2, D_{in}, S_0, S_1 (B) S_1, D_{in}, S_0, S_2
(C) D_{in}, S_0, S_1, S_2 (D) D_{in}, S_2, S_0, S_1

18. The output of the combinational circuit given below is: **[2016]**



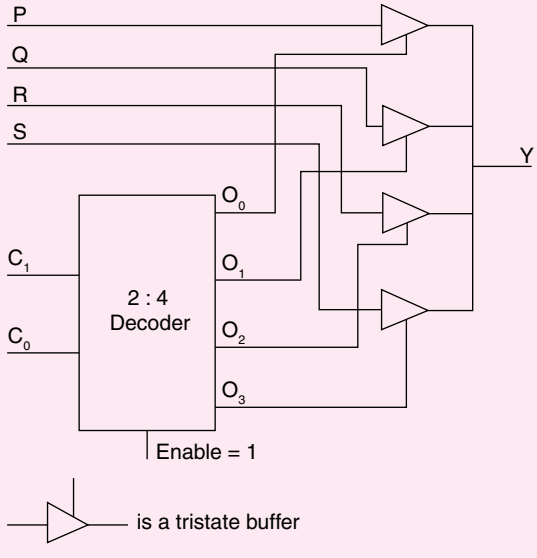
(A) A + B + C (B) A(B + C)
(C) B(C + A) (D) C(A + B)

19. Identify the circuit below:



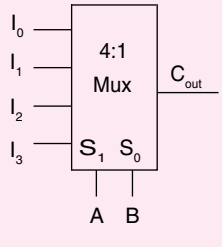
(A) Binary to Gray code converter
(B) Binary to XS3 converter
(C) Gray to Binary converter
(D) XS3 to Binary converter

20. The functionality implemented by the circuit below is: **[2016]**



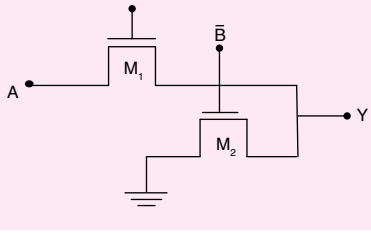
(A) 2 to 1 multiplexer
(B) 4 to 1 multiplexer
(C) 7 to 1 multiplexer
(D) 6 to 1 multiplexer

21. A 4:1 multiplexer is to be used for generating the output carry of a full adder. A and B are the bits to be added while C_{in} is the input carry and C_{out} is the output carry. A and B are to be used as the select bits with A being the more significant select bit. **[2016]**



Which one of the following statements correctly describes the choice of signals to be connected to the inputs I_0, I_1, I_2, and I_3 so that the output is C_{out}?
(A) $I_0 = 0$, $I_1 = C_{in}$, $I_2 = C_{in}$ and $I_3 = 1$
(B) $I_0 = 1$, $I_1 = C_{in}$, $I_2 = C_{in}$ and $I_3 = 1$
(C) $I_0 = C_{in}$, $I_1 = 0$, $I_2 = 1$ and $I_3 = C_{in}$
(D) $I_0 = 0$, $I_1 = C_{in}$, $I_2 = 1$ and $I_3 = C_{in}$

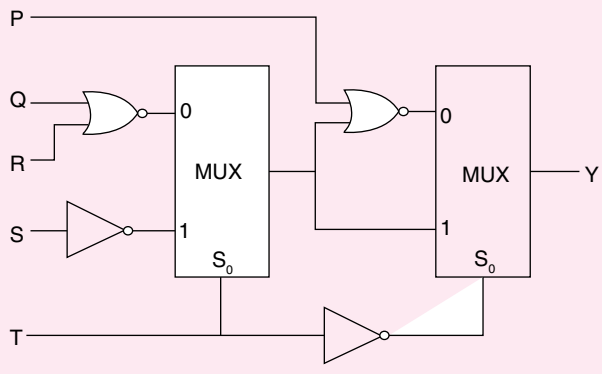
22. The logic functionality realized by the circuit shown below is: **[2016]**



(A) OR (B) XOR
(C) NAND (D) AND

23. The minimum number of 2-input NAND gates required to implement a 2-input XOR gate is **[2016]**
(A) 4 (B) 5
(C) 6 (D) 7

24. For the circuit shown in the figure, the delays of NOR gates, multiplexers and inverters are 2ns, 1.5ns and 1ns respectively. If all the inputs P, Q, R, S and T are applied at the same time instant, the maximum propagation delay (in ns) of the circuit is _____.
[2016]



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.** B	**18.** C	**19.** C	**20.** D
21. A	**22.** A	**23.** A	**24.** C						

Previous Years' Questions

1. D	**2.** A	**3.** A	**4.** A	**5.** A	**6.** B	**7.** D	**8.** D	**9.** D	**10.** B
11. C	**12.** D	**13.** C	**14.** A	**15.** C	**16.** 195 ns	**17.** D	18. C	19. A	**20.** B
21. A	**22.** D	**23.** A	**24.** 6ns						