We've Got 4.8 stars!

STUDY MATERIAL FOR BTECH EXAMS FOR WORKING PROFESSIONALS

- ✓ 100% exam oriented
- ✓ Latest questions included
- ✓ Frequent updates
- ✓ Low fee

MORE INFO

+91-9412903929

AMIESTUDYCIRCLE.COM

AMIESTUDYCIRCLE@GMAIL.COM

CIVIL LINES, NEAR IIT, ROORKEE

BTECH NOTES SERIES

Digital System Design (Digital Electronics/Digital Logic Design)

(As Per AICTE/Technical Universities Syllabus)

CONTENT

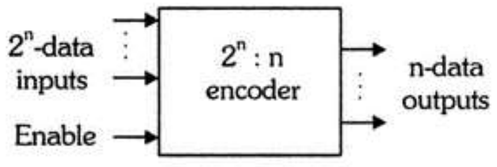
(Use bookmarks to navigate)

COMBINATIONAL CIRCUITS - II

(ENCODERS, DECODERS, MULTIPLEXERS & PLDS)

ENCODERS

An encoder is a combinational logic function that has 2^n (or fewer) input lines and n output lines. The n output lines generate the binary code for the possible 2^n input lines. Consider the case of a 2-to-1 line binary encoder. Such an encoder has two (2^1) input lines and one output lines representing the single-bit binary equivalent.



Priority Encoder

A priority encoder is an encoder circuit that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

The truth table of a four-input priority encoder is given below.

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	0	0	1
x	x	x	1	1	1	1

In addition to the two outputs x and y, the circuit has a third output designated by **V**; this is a **valid bit** indicator that is set to 1 when one or more inputs are equal to 1. If all inputs are 0, there is no valid input and V is equal to 0. The other two outputs are not inspected when V equals 0 and are specified as don't-care conditions.

Note that whereas X's in output columns represent don't-care conditions, the X's in the input columns are useful for representing a truth table in condensed form. Instead of listing all 16 minterms of four variables, the truth table uses an X to represent either 1 or 0. For example, X100 represents the two minterms 0100 and 1100.

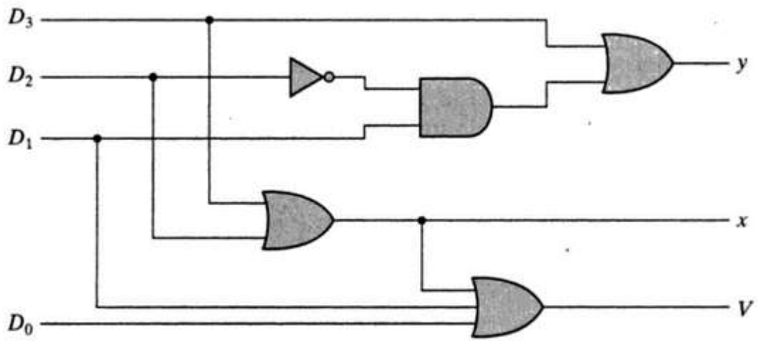
According to Table, the higher the subscript number, the higher the priority of the input. Input D_3 has the highest priority, so, regardless of the values of the other inputs, when this input is 1, the output for xy is 11 (binary 3). D_2 has the next priority level. The output is 10 if $D_2 = 1$, provided that $D_3 = 0$, regardless of the values of the other two lower priority inputs. The output for D, is generated only if higher priority inputs are 0, and so on down the priority levels.

After forming Karnaugh maps for the given truth table, we form the following equations

$$x = D_2 + D_3$$

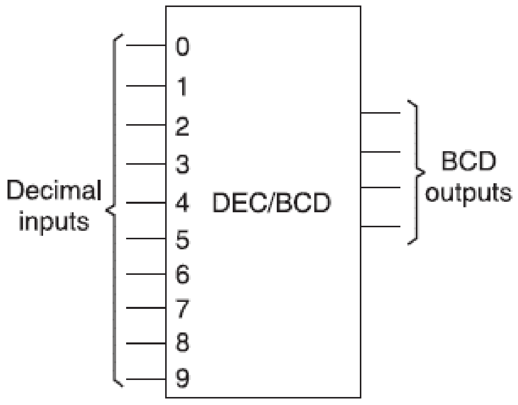
and $$y = D_3 + D_1D_2'$$

The logic diagram of a 4-bit priority encoder is given below:



Decimal-to-BCD Encoder

This type of encoder has 10 inputs—one for each decimal digit, and 4 outputs corresponding to the BCD code as shown in given figure.



This is a basic 10-line to 4-line encoder. The BCD code is listed in the truth table given below and from this we can determine the relationships between each BCD bit and the decimal digits.

Decimal inputs	Decimal no.	Output			
		A_3	A_2	A_1	A_0
D_0	0	0	0	0	0
D_1	1	0	0	0	1
D_2	2	0	0	1	0
D_3	3	0	0	1	1
D_4	4	0	1	0	0
D_5	5	0	1	0	1
D_6	6	0	1	1	0
D_7	7	0	1	1	1
D_8	8	1	0	0	0
D_9	9	1	0	0	1

There is no explicit input for a decimal 0. The BCD output is 0000 when the decimal inputs 1–9 are all 0.

From the table, we get

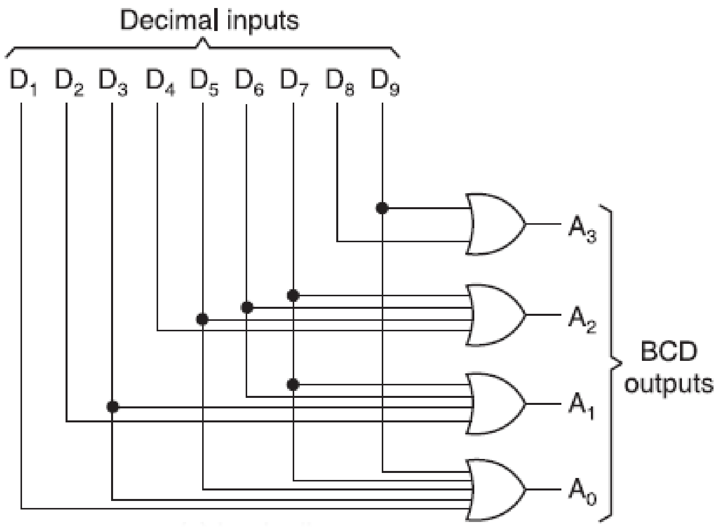
$$A_3 = D_8 + D_9$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

$$A1 = D_2 + D_3 + D_6 + D_7$$

$$A0 = D_1 + D_3 + D_5 + D_7 + D_9$$

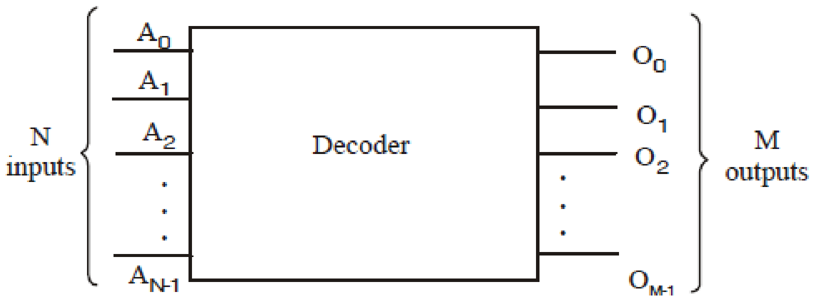
The logic circuit of decimal to BCD encoder is given below:



DECODERS

A decoder is a logic circuit that converts an N-bit binary input code into M output lines such that only one output line is activated for each one of the possible combinations of inputs. In other words, we can say that a decoder identifies or recognizes or detects a particular code.

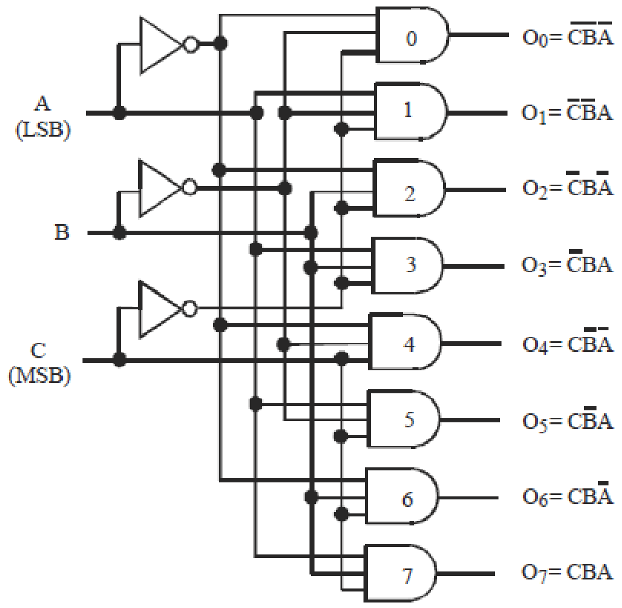
Following figure shows the general decoder diagram with N inputs and M outputs.



As seen from the diagram, it has N inputs and M outputs. Since each of the N inputs can be 0 or 1, there are 2N possible input combinations or codes. For each of these input combinations only one of the M outputs will be active (HIGH) and all the other outputs are inactive (LOW). Many decoders are designed to produce active-LOW outputs where only the selected output is LOW while all others are HIGH. This would be indicated by the presence of small circles on the output lines in the decoder diagram.

3 line to 8 line decoder (1 of 8 decoder)

Following figure shows the logic diagram for a decoder with 3 inputs and 8 (= 2^3) outputs. It uses all AND gates and so the outputs are active-HIGH.


A (LSB)
B
C (MSB)
0
1
2
3
4
5
6
7
$O_0 = \overline{C}\overline{B}\overline{A}$
$O_1 = \overline{C}\overline{B}A$
$O_2 = \overline{C}B\overline{A}$
$O_3 = \overline{C}BA$
$O_4 = C\overline{B}\overline{A}$
$O_5 = C\overline{B}A$
$O_6 = CB\overline{A}$
$O_7 = CBA$

It may be noted that for a given input code, the only output that is active (HIGH) is the one corresponding to the decimal equivalent of the binary input code. For example, output O_0 goes HIGH only when CBA = $000_2 = 0_{10}$. Similarly O_1 goes HIGH only when CBA = $001_2 = 1_{10}$ and O_5 goes HIGH when CBA = $101_2 = 5_{10}$ and so on. The complete operation of the decoder is given in the truth table shown below.

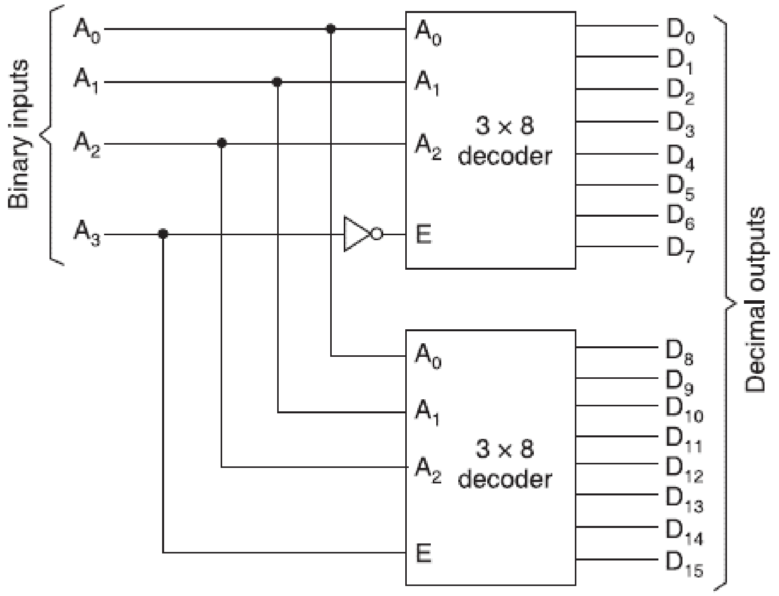
Inputs			**Outputs**							
C	**B**	**A**	**O_7**	**O_6**	**O_5**	**O_4**	**O_3**	**O_2**	**O_1**	**O_0**
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

This decoder can be referred to in several ways. It can be called a **3-line-to-8-line decoder** because it has **3 input lines and 8 output lines**. It could also be called a **binary-to-octal decoder** or converter because it takes a **3-bit binary input code** and activates one of the eight (octal) outputs corresponding to that code. It is also referred to as a **1-of-8 decoder** because only 1 of the 8 outputs is activated at one time.

4-to-16 Decoder from Two 3-to-8 Decoders

Decoders with enable inputs can be connected together to form a larger decoder circuit.

Following figure shows the arrangement for using two 74138s, 3-to-8 decoders, to obtain a 4-to-16 decoder.



Function table is given below.

Binary inputs				**Decimal output**
A_3	**A_2**	**A_1**	**A_0**	**(active low)**
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
1	0	0	0	D_8
1	0	0	1	D_9
1	0	1	0	D_{10}
1	0	1	1	D_{11}
1	1	0	1	D_{12}
1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

The most significant input bit A_3 is connected through an inverter to $\bar{E}$ on the upper decoder (for D_0 through D_7) and directly to E on the lower decoder (for D_8 through D_{15}). Thus, when A_3

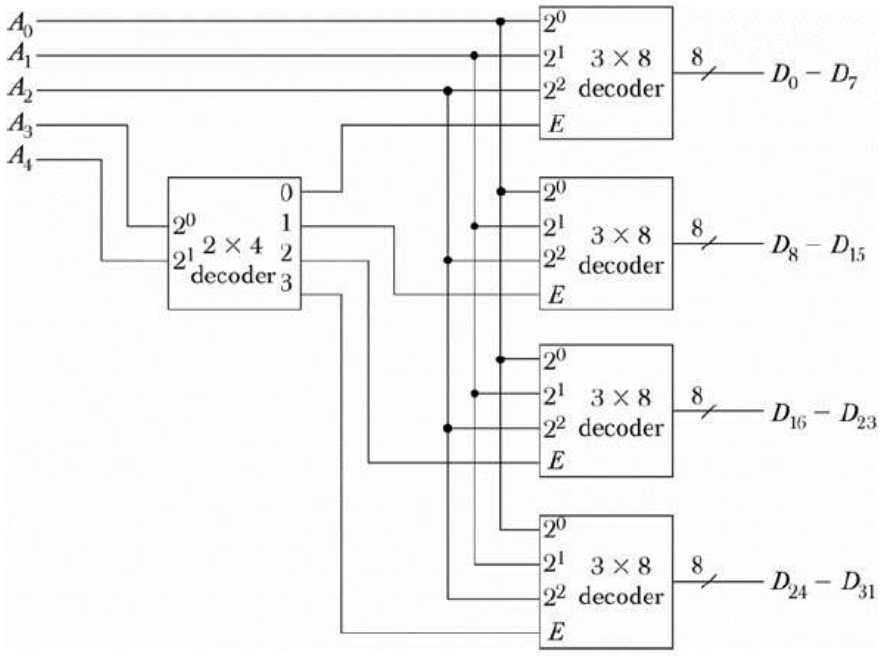
is **LOW**, the upper decoder is enabled and the lower decoder is disabled. The bottom decoder outputs all 0s, and top 8 outputs generate minterms. When A_3 is **HIGH**, the lower decoder is enabled and the upper decoder is disabled. The bottom decoder outputs generate minterms 1000 to 1111 while the outputs of the top decoder are all 0s.

Example (BPUT 2022, 10 marks)

Construct a 5 to 32 line decoder with four 3 to 8 line decoders with enable and a 2 to 4 line decoder. Use block diagrams for the components.

Solution

See following logic circuit.



Example

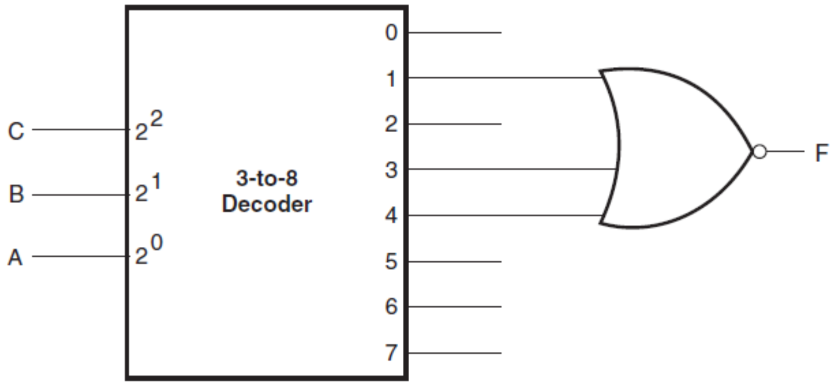
A combinational circuit is defined by F = Σ(0, 2, 5, 6, 7). Hardware implement the Boolean function F with a suitable decoder and an external OR/NOR gate having the minimum number of inputs.

Solution

The given Boolean function has five three-variable minterms. This implies that the function can be implemented with a 3-to-8 line decoder and a five-input OR gate. Also, $\overline{F}$ will have only three three-variable minterms, which means that F could also be implemented by considering minterms corresponding to the complement function and using a three-input NOR gate at the output. The second option uses a NOR gate with fewer inputs and therefore is used instead. F = Σ (0, 2, 5, 6, 7).

Therefore, $\bar{F} = \Sigma\ (1, 3, 4)$.

Following figure shows the hardware implementation of Boolean function F.



Example (PTU 2009)

Implement the following function using 3 to 8 decoder.

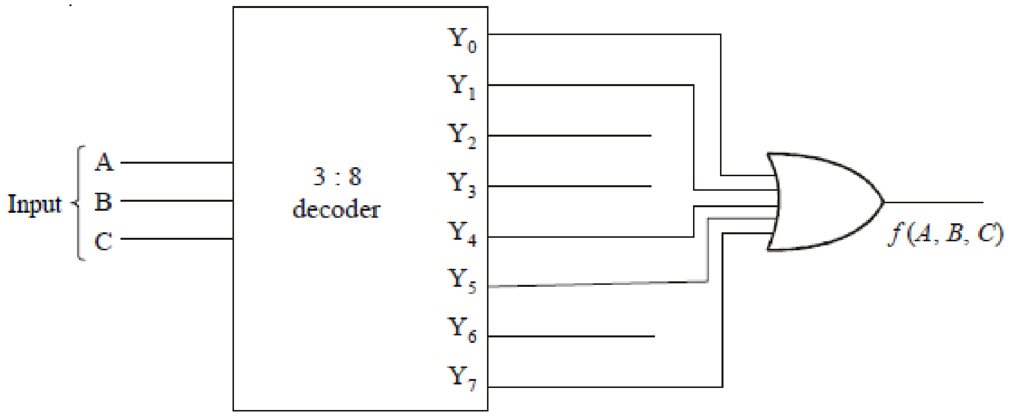
f (A, B, C) = Σm (0, 1, 4, 5, 7)

Solution

Given the function

f (A, B, C) = Σm (0, 1, 4, 5, 7)

There are three inputs and total five minterms, the implementation is shown in following figure. The decoder generates the five minterms for A, B, C.



Example

Implement using 3:8 decoders

(i) f (A, B, C) = Σm (0, 1, 5, 6)

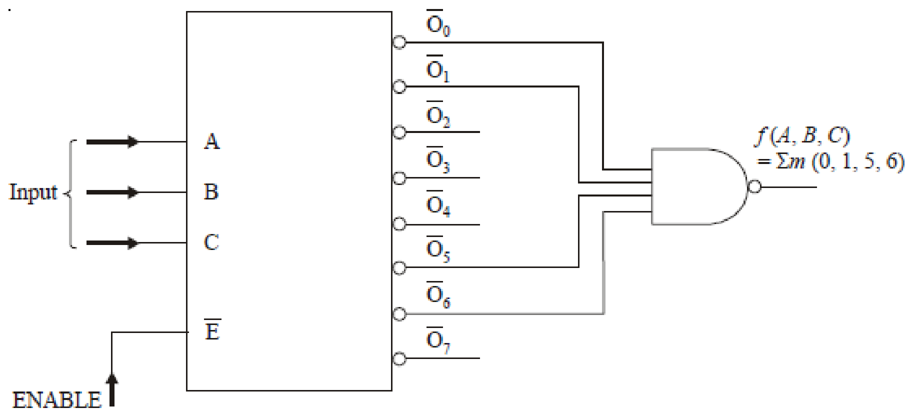
(ii) f (A, B, C) = πM (1, 2, 5, 6)

Solution

(i) Given : The function

$$f(A, B, C) = \Sigma m(0, 1, 5, 6)$$

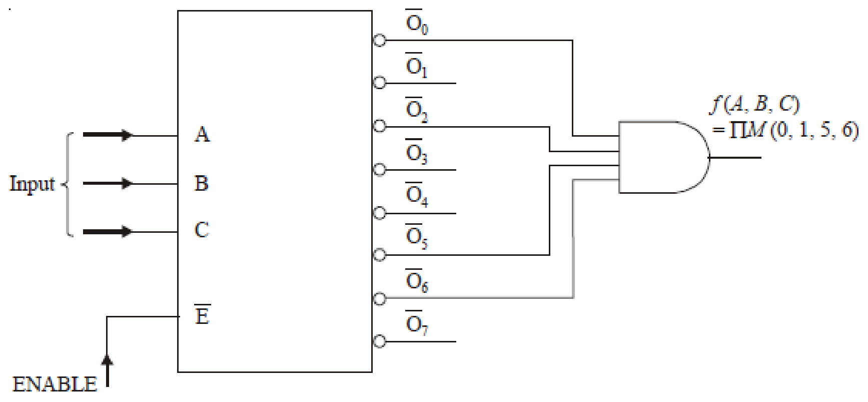
Lets consider for active low output, when decoder output is active low, the output complemented form, i.e., it generates maxterms (sum terms) for input variables. The **SOP** is achieved by connecting NAND gate. The given function is the form of SOP, the output of decoder 0, 1, 5 and 6 is connected to NAND gate as shown in following figure.



(ii) Given : The function

$$f(A, B, C) = \pi M(1, 2, 5, 6)$$

Lets consider for active low output, when decoder output is active low, the output complemented form, i.e., it generates maxterms (sum terms) for input variables. The **POS** is achieved by connecting AND gate. The given function is the form of POS, the output of decoder 1, 2, 5 and 6 is connected to AND gate as shown in following figure.



Example

Implement the function using 3 : 8 decoder

$F1 (A, B, C) = \Sigma (0, 1, 3, 5, 6)$

$F2 (A, B, C) = \Sigma (0, 2, 4, 7)$

Solution

Given : The given functions are

$F1 (A, B, C) = \Sigma (0, 1, 3, 5, 6)$

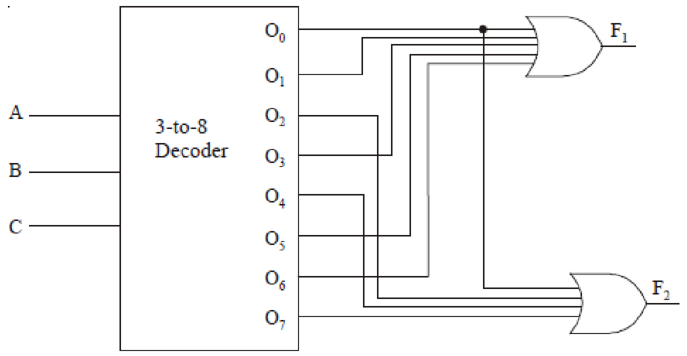
$F2 (A, B, C) = \Sigma (0, 2, 4, 7)$

From the function we know that there are three variables A, B and C and total number of term are 8. So we need 3-of-8 decoder.

he table for the output function is shown below.

Inputs			Output Function	
A	B	C	F_1	F_2
0	0	0	1	1
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	1	0	1

The output function F_1 is obtained by **ORed** the output O_0, O_1, O_3, O_5 and O_6 of the decoder output. The output function F_2 is obtained by **ORed** the output O_0, O_2, O_4 and O_7 of the decoder output as shown in following figure.



Example

Design a full adder circuit with decoder IC.

Solution

Observe the truth table of a full adder given below.

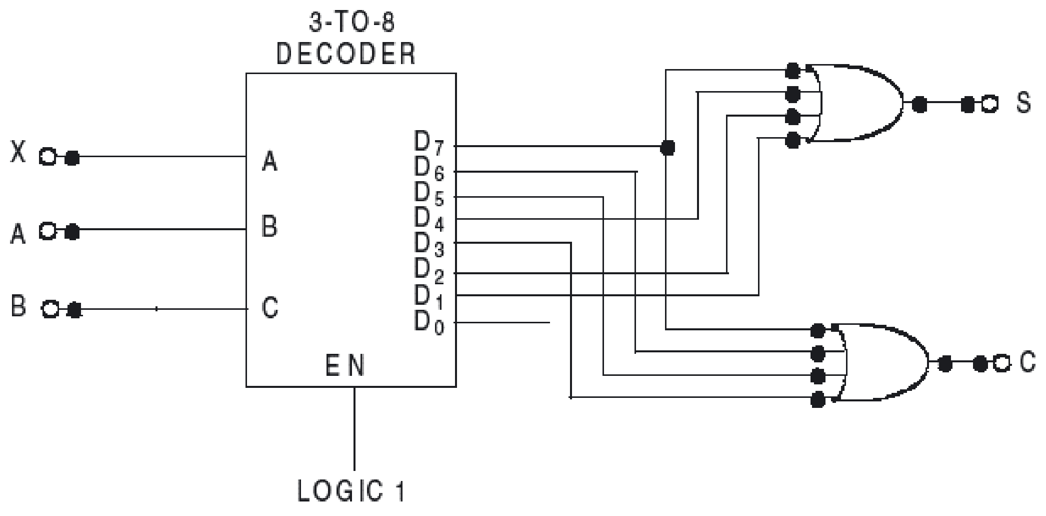
Input			Output	
X	A	B	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

In respect to minterms, the Boolean expression of sum output S and carry output C can be written as:

$$S = X'A'B + X'AB' + XA'B' + XAB \text{ and}$$

$$C = X'AB + XA'B + XAB' + XAB.$$

Logic circuit from these expressions is given below.



Example (VTU 2022, 8 marks)

Design a full subtractor circuit with decoder IC using two NAND gates.

Solution

Truth table is given below

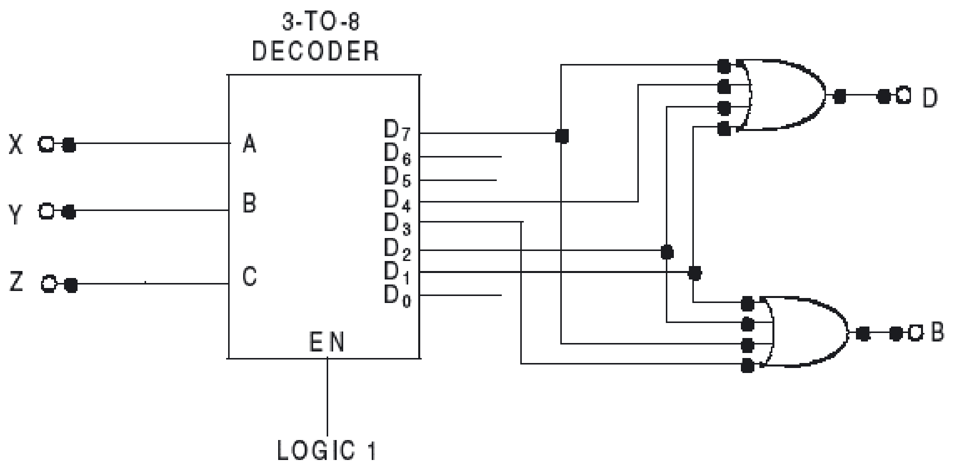
Input			Output	
X	A	B	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The Difference D and Borrow B outputs may be written as

$$D = X'Y'Z + X'YZ' + XY'Z' + XYZ \text{ and}$$

$$B = X'Y'Z + X'YZ' + X'YZ + XYZ.$$

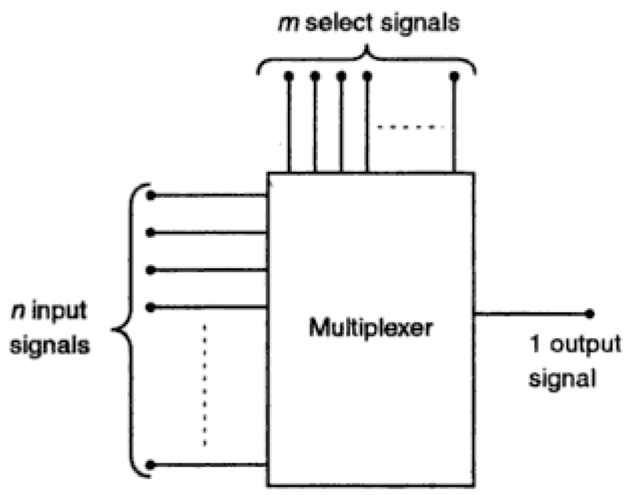
The combinational circuit with decoder is shown in following figure.



MULTIPLEXER

The term **'multiplex'** means "**many into one**." Multiplexing is the process of transmitting a large number of information over a single line. A digital multiplexer (MUX) is a combinational circuit that selects one digital information from several sources and transmits the selected information on a single output line. A **multiplexer is also called a data selector** since it selects one of many inputs and steers the information to the output.

The multiplexer has several data-input lines and a single output line. The selection of a particular input line is controlled by a set of selection lines. The block diagram of a multiplexer with n input lines, m select signals and one output line is shown in following figure.

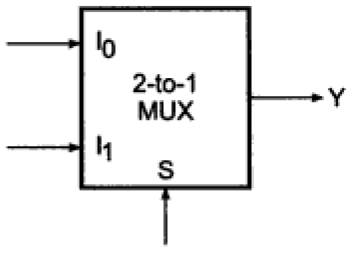


The selection lines decide the number of input lines of a particular multiplexer. If the number of n input lines is equal to 2^m, then m select lines are required to select one of the n input lines. For example, to select 1 out of 4 input lines, two select lines are required; to select 1 of 8 input lines, three select lines are required and so on.

2 TO 1 MULTIPLEXER

We will briefly describe the type of combinational logic circuit found inside a multiplexer by considering the 2-to-l multiplexer of Figure (a), the functional table of which is shown in Figure (b). Figure (c) shows the possible logic diagram of this multiplexer. The circuit functions as follows.

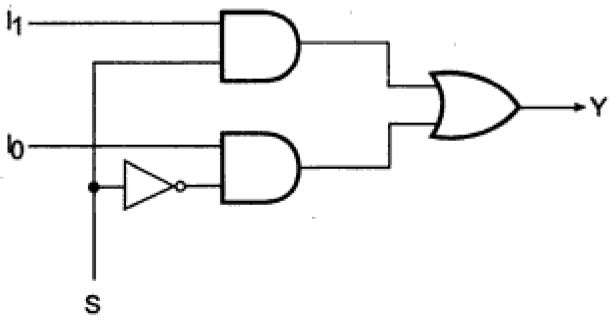
1. For S = 0, the Boolean expression for the output becomes $Y = I_0$
2. For S = 1, the Boolean expression for the output becomes $Y = I_1$



(a)

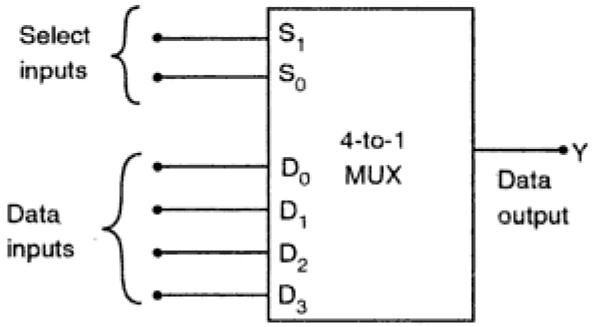
S	Y
0	I_0
1	I_1

(b)



BASIC FOUR-INPUT MULTIPLEXER

The logic symbol of a 4-to-l multiplexer is shown in following figure.



It has four data input lines (D_0 –D_3),), a single output line (Y) and two select lines (S_0 and S_1) to select one of the four input lines. The truth table for a 4-to-l multiplexer is shown in following table.

Data select inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

From this truth table a logical expression for the output in terms of the data input and the select inputs can be derived as follows:

The data output Y - data input D_0, if and only if $S_1 = 0$ and $S_0 = 0$,

Therefore, $Y = D_0\bar{S}_1\bar{S}_0 = D_0\bar{0}.\bar{0} = D_0 1.1 = D_0$

The data output Y = D_1, if and only if $S_1 = 0$ and $S_0 = 1$.

Therefore, $Y = D_1\bar{S}_1 S_0 = D_1, when\, S_1S_0 = 01$

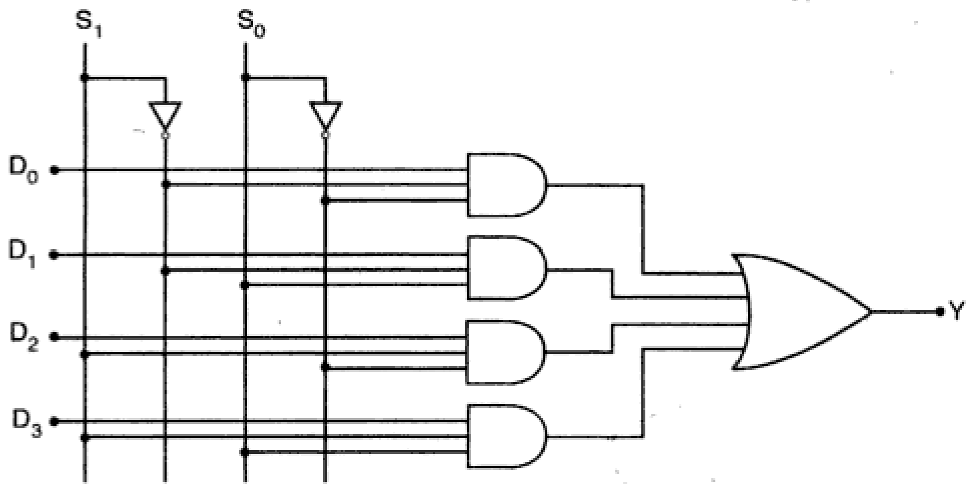
Similarly, $Y = D_2 S_1\bar{S}_0 = D_2, when\, S_1S_0 = 10\, and$

$Y = D_3 S_1 S_0 = D_3, when\, S_1S_0 = 11$

If the above terms are ORed, then the final expression for the data output is given by

$$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1 S_0 + D_2 S_1\bar{S}_0 + D_3 S_1 S_0$$

Using the above expression, the 4-to-l multiplexer can be implemented using two NOT gates, four 3-input AND gates and one 4-input OR gate as shown in figure.



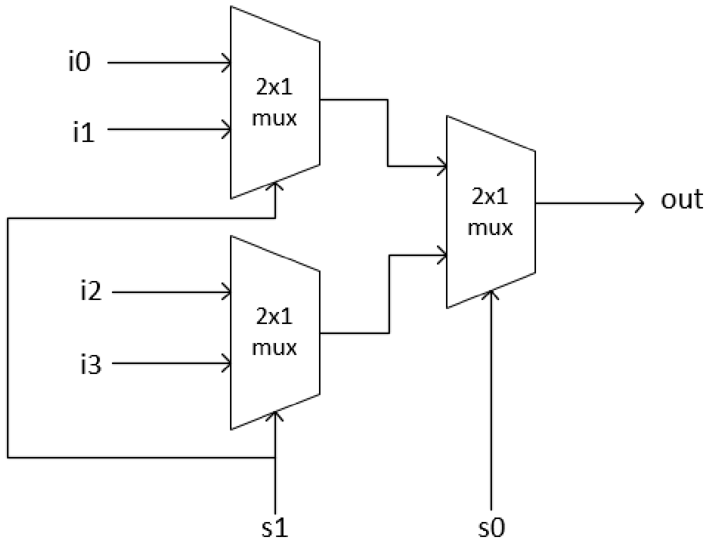
Here, each of the four data input lines is applied to any one input of an AND gate and the AND gate outputs are connected with the inputs of OR gate to generate the output Y.

Example (AKTU 2020, 2021, 2 marks)

Implement a 4:1 multiplexer using 2: 1 multiplexer.

Solution

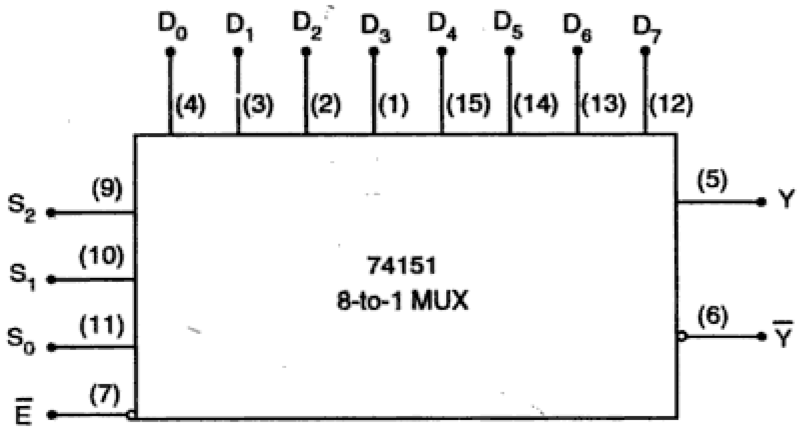
Logic circuit is given below:



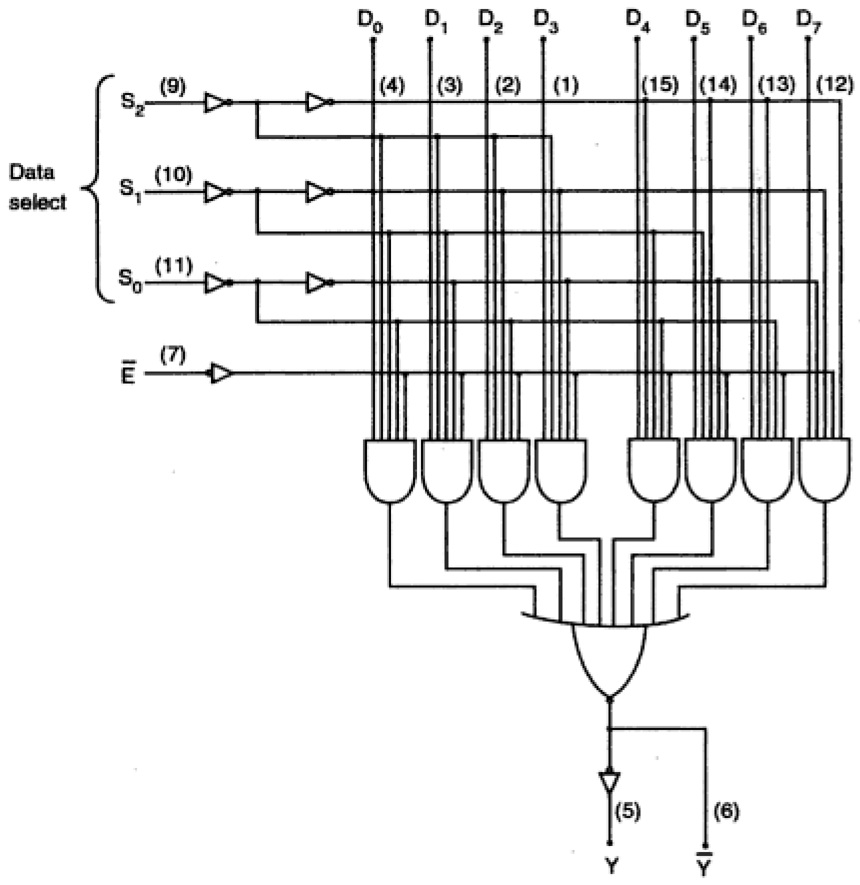
When S1 is set to HIGH it will select i1 and i3 now if s0 is LOW output will have i1 otherwise i3 and similar for i0 and i2.

8-to-1 Multiplexer

IC 74I51 is an 8-to-l multiplexer with eight data inputs ($D_0 – D_7$), three select input lines ($S_2 – S_0$) and a single output (Y). It also has an enable input $\bar{E}$ and provides both normal and inverted outputs (i.e. Y and $\bar{Y}$). Since $2^3 = 8$, three bits are required to select any one of the eight data bits. When $\bar{E} = 0$, the select inputs $S_2S_1S_0$ will select one of the data input to pass through the output Y. When $\bar{E} = 1$, the multiplexer is disabled. The logic symbol of IC 74151 is shown in following figure.



Its logic diagram is shown in following figure.



The operation of this IC is summarised in truth table.

Inputs				Outputs	
$\bar{E}$	S_2	S_1	S_0	Y	$\bar{Y}$
1	X	X	X	1	0
0	0	0	0	D_0	$\bar{D}_0$
0	0	0	1	D_1	$\bar{D}_1$
0	0	1	0	D_2	$\bar{D}_2$
0	0	1	1	D_3	$\bar{D}_3$
0	1	0	0	D_4	$\bar{D}_4$
0	1	0	1	D_5	$\bar{D}_5$
0	1	1	0	D_6	$\bar{D}_6$
0	1	1	1	D_7	$\bar{D}_7$

Example (GTU 2022, 2023, 4 marks)

Implement the 8×1 MUX using two 4×1 MUX.

Solution

When S_2 is zero then the first MUX is activated and the second will remain inactive. Again when S_2 = l, then the first MUX will remain inactive and the second will be activated.

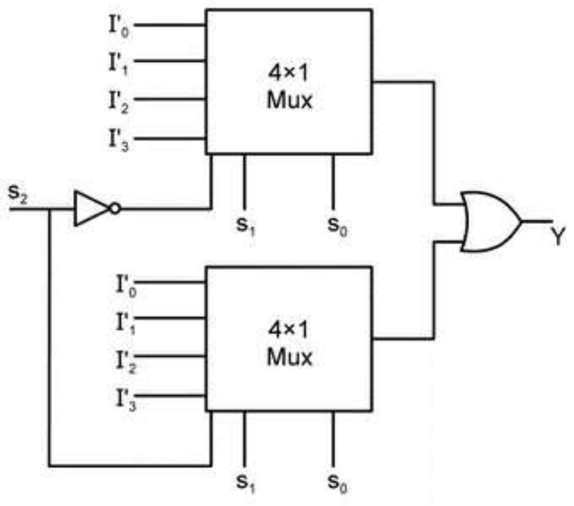
At the **output** we get the addition of two mux's output. Here total selection input are S_0, S_1 and S_2 and general input = 8.

Therefore, it becomes 8:1 MUX.

Truth table is given below.

S_2	S_1	S_0	O/P
0	0	0	I_0'
0	0	1	I_1'
0	1	0	I_2'
0	1	1	I_3'
1	0	0	I_4'
1	0	1	I_5'
1	1	0	I_6'
1	1	1	I_7'

The logic diagram is given below:



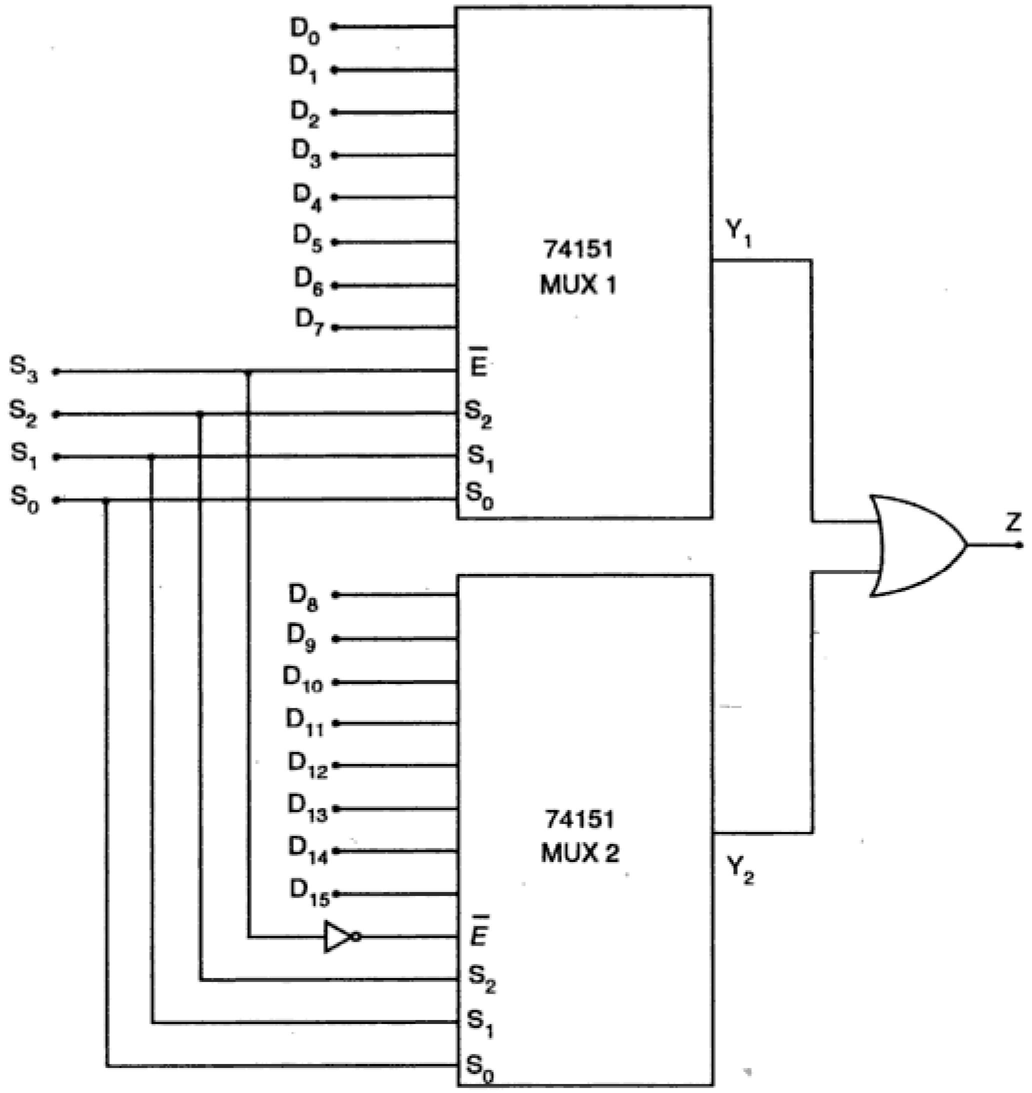
Example

Implement a 16-to-l multiplexer using two 8-to-l multiplexer ICs (74151).

Solution

A 16-to-l multiplexer can be implemented using two IC 74151 —8-to-l multiplexers and one 2-input OR gate as shown in figure.

Here, to select one of the 16 inputs, four select lines ($S_3S_2S_1S_0$) are required. Among the four select lines, the least significant three select lines ($S_2S_1S_0$) are connected with three select inputs of both the multiplexer ICs. The most significant select line S_3 is connected directly to the $\bar{E}$ input of MUX I while the same is connected through an inverter to the $\bar{E}$ input of MUX2. Therefore, when $S_3 = 0$. MUX 1 is selected and the inputs (D_0 to D_7) are multiplexed to the output Z and MUX2 is disabled. When $S_3 = 1$, the MUX1 is disabled while MUX2 enabled and the inputs (D_8 to D_{15}) are multiplexed to the output Z. Also, note that the outputs of MUX1 and MUX2 (i.e. Y_1 and Y_2) are ORed using an OR gate to generate output Z.

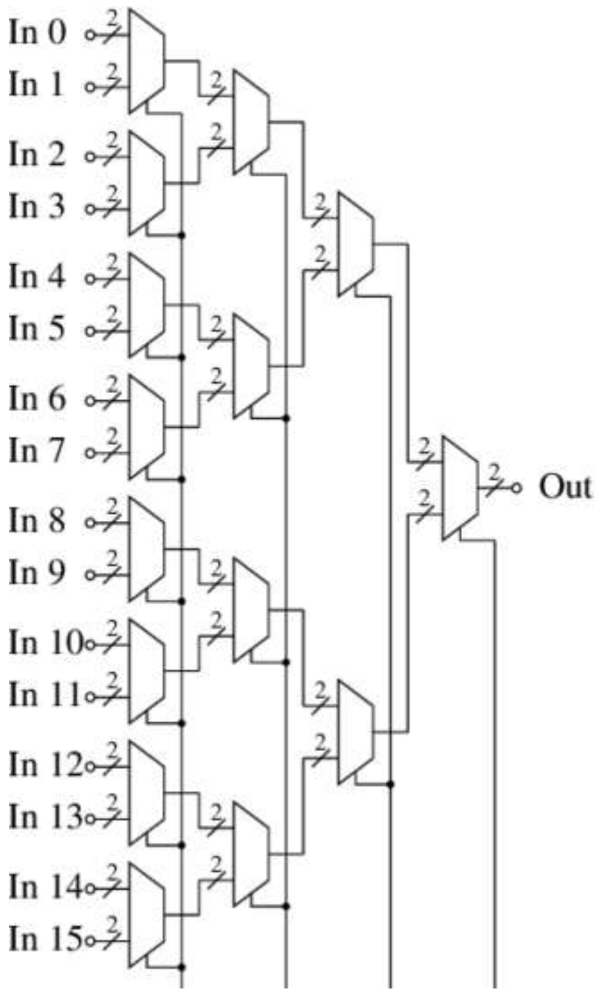


Example (RGPV 2022, 8 marks)

Design and explain a 16:1 mux using 2:1 mux.

Solution

See following logic diagram.

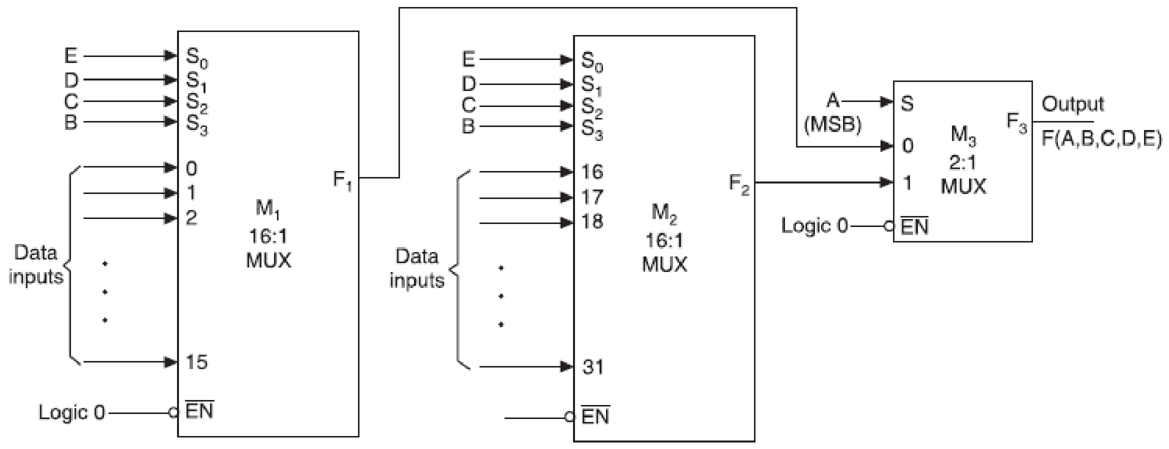


Example (JNTUH 2020, 7 marks)

Design a 32:1 Multiplexer using two 16:1 and 2:1Multiplexers.

Solution

The arrangement to obtain a 32:1 mux using two 16:1 muxes and one 2:1 mux is shown in following figure.



A 32:1 mux has 32 data inputs. So it requires five data select lines.

Since a 16:1 mux has only four data select lines, the inputs B,C,D,E are connected to the data select lines of both the 16:1 muxes and the most significant input A is connected to the single data select line of the 2:1 mux.

For the values of BCDE = 0000 to 1111, inputs 0 to 15 will appear at the input terminal 0 of the 2:1 mux through the output F_1 of the first 16:1 mux and inputs 16 to 31 will appear at the input terminal 1 of the 2:1 mux through the output F_2 of the second 16:1 mux.

For A = 0, output F = F_1.

For A = 1, output F = F_2.

Example (AKTU 2020, 10 marks)

Implement a full adder by using 8:1 multiplexer.

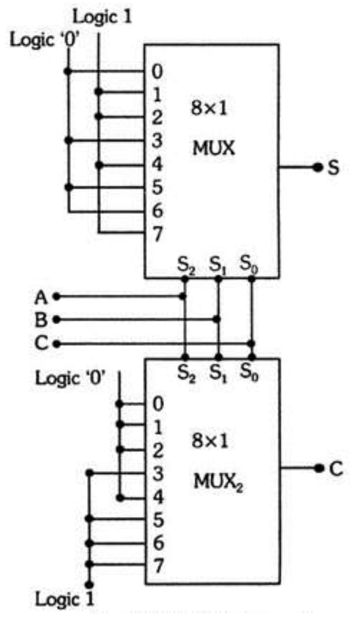
Solution

The full adder has two outputs-sum (S) and carry (C) and they are represented by

$S = \Sigma m\ (1, 2, 4, 7)$ and

$C = \Sigma m\ (3, 5, 6, 7)$.

To design the full adder we need two 8x1 multiplexers, one for sum and another for carry as shown in following figure.



Implementation of Boolean Expression using Multiplexers

Any boolean or logical expression can be easily implemented using a multiplexer. If a boolean expression has (n + 1) variables, then n of these variables can be connected to the select lines of the multiplexer. The remaining single variable along with constants 1 and 0 is used as the input of the multiplexer. For example, if A is the single variable, then the inputs of the multiplexer are A, $\bar{A}$, 1 and 0. By this method, any logical expression can be implemented. In general, a boolean expression of (n + 1) variables can be implemented using a multiplexer with 2" inputs. To demonstrate this procedure, consider the function

$$F(A, B,C,D) = \Sigma(0,1,3,4,8,9,15)$$

As the given function is a four-variable function, we need a multiplexer with 3 select lines and eight inputs. Apply variables B, C, and D to the select lines. The procedure for implementing the function are: (i) list the input of the multiplexer and (ii) list under them ail the minterms in two rows as shown in table.

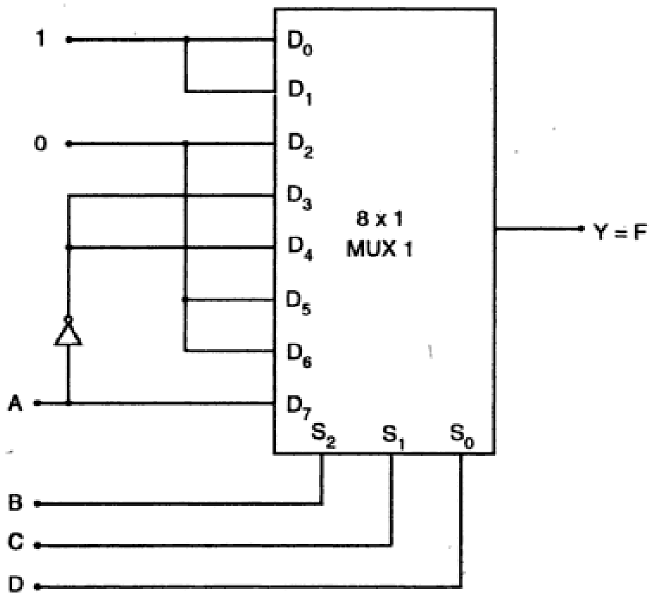
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

The first half of the minterms are associated with $\bar{A}$ and the second half with A. The given function is implemented by circling the minterms of the function and applying the following rules to find the values for the inputs of the multiplexer.

1. If both the minterms in a column are not circled, apply 0 to the corresponding input

2. If both the minterms in a column are circled, apply 1 to the corresponding input.
3. If the bottom minterm is circled and the top is not circled, apply A to the input.
4. If the top minterm is circled and the bottom is not circled, apply $\bar{A}$ to the input.

Now, using the procedure and the table, the given function can be implemented using an 8-to-l multiplexer as shown below.



It is not necessary to choose the most significant variable as an input to the multiplexer. One can choose any one of the variables as an input and accordingly the multiplexer implementation table has to be modified.

Example

Implement the following function using a multiplexer:

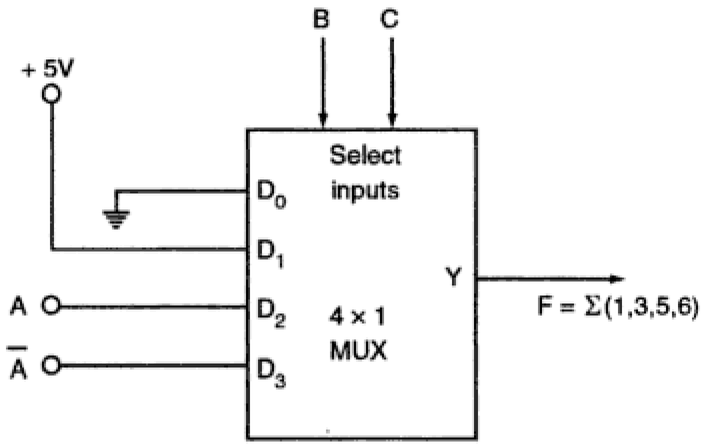
$$F(A,B,C) = \Sigma(1,3,5.6)$$

Solution

The given function has three variables. Hence, it can be implemented using a multiplexer with two select inputs and four data inputs. The implementation table of the given function is shown in following table.

	D_0	D_1	D_2	D_3
$\bar{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	$\bar{A}$

Now, the given three variable functions can be implemented using 4-to-l multiplexer as shown in following figure.



Example

Implement the following Boolean function using 4x1 multiplexer.

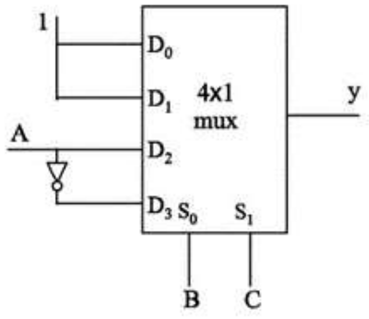
$$F(A, B, C) = \Sigma(0, 1, 3, 4, 5, 6)$$

Solution

The implementation table of the given function is shown in following table.

	D_0	D_1	D_2	D_3
$\bar{A}$	(0)	(1)	2	(3)
A	(4)	(5)	(6)	7
	1	1	A	$\bar{A}$

Now, the given three variable functions can be implemented using 8-to-l multiplexer as shown in following figure.



Example

Implement the following expressions using 8x1 mux.

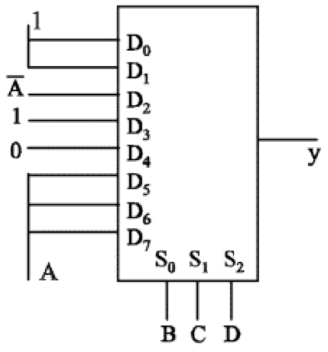
F(A,B,C,D) = Σ (0, 1, 2, 3, 8, 9, 11, 13, 14, 15)

Solution

The implementation table of the given function is shown in following table.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	(2)	(3)	4	5	6	7
A	(8)	(9)	10	(11)	12	(13)	(14)	(15)
	1	1	$\bar{A}$	1	0	A	A	A

Now, the given four variable functions can be implemented using 8-to-l multiplexer as shown in following figure.



Example

Implement the following using 8 x 1 mus.

$$F(A,B,C,D) = AB\bar{C} + A\bar{B}D + A\bar{C}D + ABCD$$

Solution

$$F(A,B,C,D) = AB\bar{C} + A\bar{B}D + A\bar{C}D + ABCD$$

$$= AB\bar{C}(D+\bar{D}) + A\bar{B}(C+\bar{C}) + A\bar{C}D(B+\bar{B}) + ABCD$$
$$AB\bar{C}D + AB\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}D + AB\bar{C}D + A\bar{B}\bar{C}D + ABCD$$

= Σ (9, 11, 12, 13, 15) [From 4 variable truth table given below]

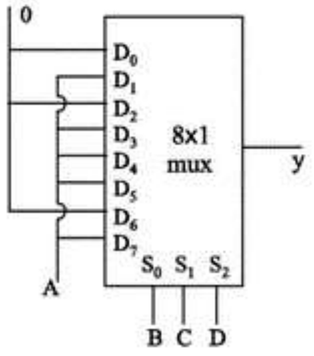
Decimal number	**A**	**B**	**C**	**D**	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	

3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	$A\bar{B}\bar{C}D$
10	1	0	1	0	
11	1	0	1	1	$A\bar{B}CD$
12	1	1	0	0	$AB\bar{C}\bar{D}$
13	1	1	0	1	$AB\bar{C}D$
14	1	1	1	0	
15	1	1	1	1	ABCD

The implementation table of the given function is shown in following table.

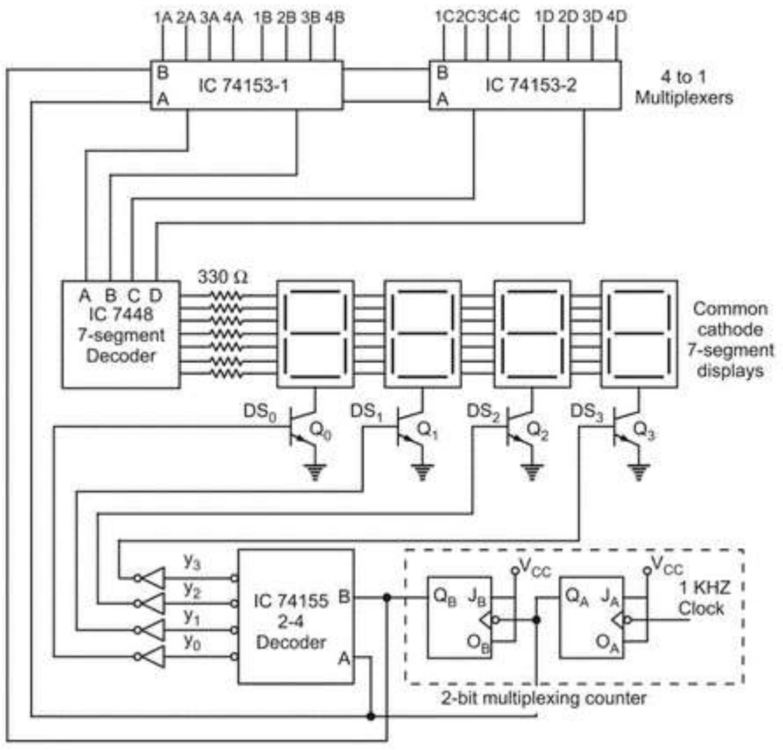
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	(9)	10	(11)	(12)	(13)	14	(15)
	0	A	0	A	A	A	0	A

Now, the given four variable functions can be implemented using 8-to-l multiplexer as shown in following figure.



4-DIGIT MULTIPLEXED DISPLAY SYSTEM

The block diagram of 4-digit multiplexed common cathode 7-segment display system is shown in the following figure.



Here, the dual 4-bit BCD inputs to be displayed in a **multiplexed** manner is connected to the inputs of four 4 to 1 multiplexers (IC 74153) as:

- 1A IB IC ID (MSD);
- 2A 2B 2C 2D (second MSD);
- 3A 3B 3C 3D (Third digit) and
- 4A 4B 4C 4D (LSD).

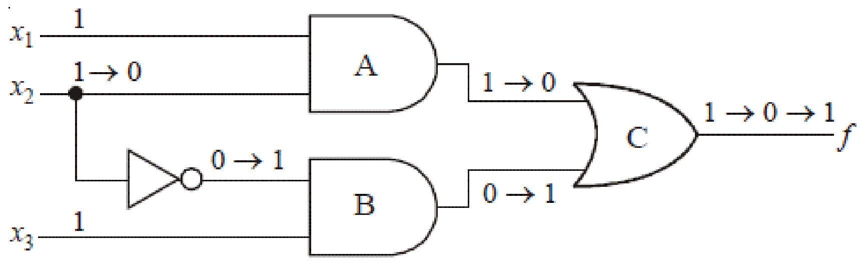
The select lines A and B of these multiplexer ICs are connected to the outputs of 2-bit multiplexing counter, constructed using JK flip-flops. The output of multiplexing counter also acts as the input for a 2 to 4 decoder IC 74155 whose outputs are digit select inputs (DS_0-DS_3) of the four 7-segment displays.

HAZARDS IN COMBINATIONAL CIRCUITS

A hazard is a condition where a single variable change produces a momentary output change when no output change should occur.

Static 1 hazard

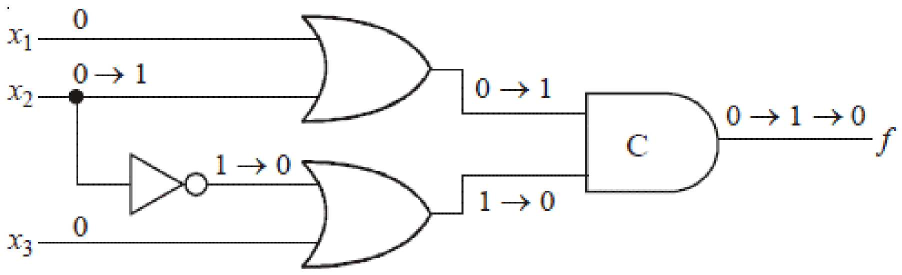
To illustrate a static 1- hazard, consider the gate network of following figure.



Assume the input state $x_1x_2x_3$ = 111 is initially applied to the network. Then the upper AND gate A has a 1 output, and the lower AND gate B has a 0 output and final output f = 1. If the input state is next changed to $x_1x_2x_3$ = 101. The value of x_2 changes, then the output of the upper AND gate A becomes 0 and the lower AND gate B becomes 1 output and final output is same f = 1.

Static 0 hazard

To understand a static 0-hazard, consider the network shown in following figure assume that the NOT gate has an appreciably greater propagation delay time than the other gates.

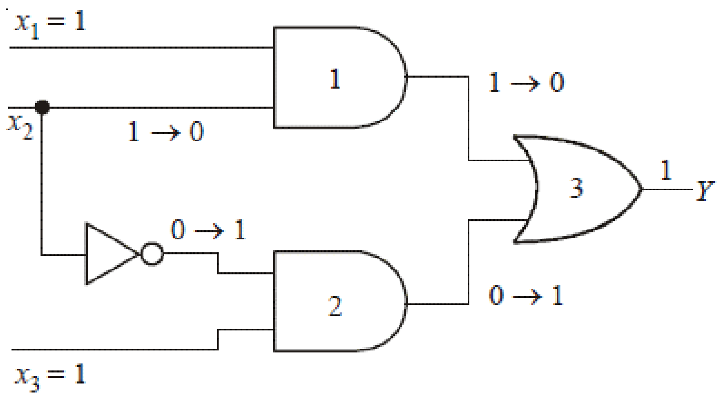


In this case there is a static 0-hazard in the transition between the input states $x_1x_2x_3$ = 000 and $x_1x_2x_3$ = 010. Since it is possible for a logic-1 signal to appear at both input terminal of the AND gate for a short duration.

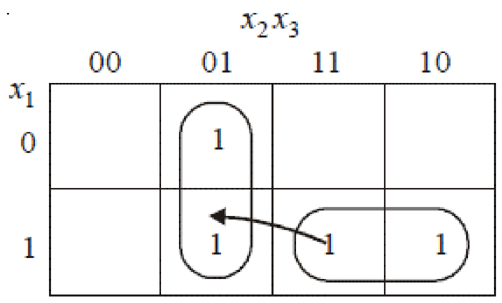
Detection of Static Hazards

The occurrence of the hazard can be detected by inspecting the map of the particular circuit.

Consider the circuit shown in following figure.



The K-map of this circuit is shown below.



The change in x_2 from 1 to 0 moves the circuit from minterm 111 to minterm 101.

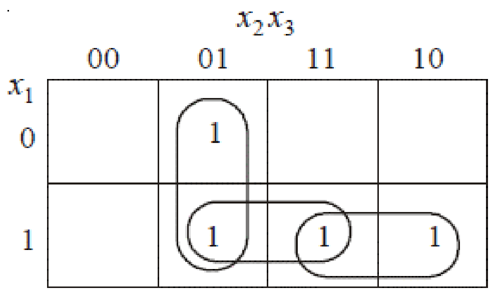
The hazard exists because the change of input results in a different product term covering the two minterms. Minterm 111 is covered by the product term implemented in AND gate 1, and minterm 101 is covered by the product term implemented in AND gate 2.

Whenever the circuit must move from one product term to another, there is a *possibility* of a momentary interval when neither term is equal to 1, giving rise to an undesirable 0 output. The output function of the circuit is,

$$y = x_1x_2 + \bar{x}_2x_3$$

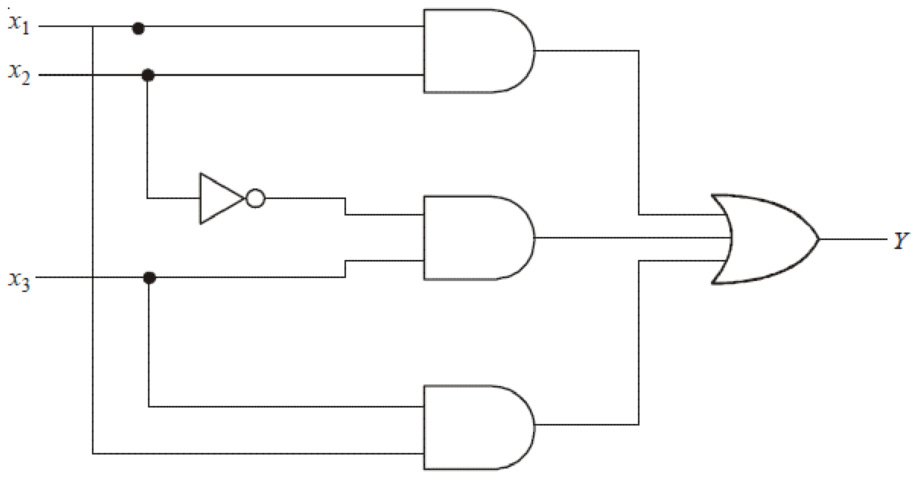
Elimination of Static Hazards

The above discussion on the detection of static hazards also suggests a method for their elimination. Additional logic is used to provide the holding action that is needed to eliminate the static hazards. The additional logic is determined by ensuring that any two adjacent cells on a Karnaugh map with the same logic-value entry are included in a common square. This is shown in the map given below.



Where the two minterms that cause the hazard are combined into one product term.

The hazard-free circuit obtained by this configuration is shown below.



The **extra gate** in the circuit generates the product term x_1, x_3. In general, hazards in combinational circuits can be removed by covering any two min terms that may produce a hazard with a product term common to both. The removal of hazards requires the addition of redundant gates to the circuit. The output of the circuit

$$y = x_1x_2 + \bar{x}_2x_3 + x_1x_3$$

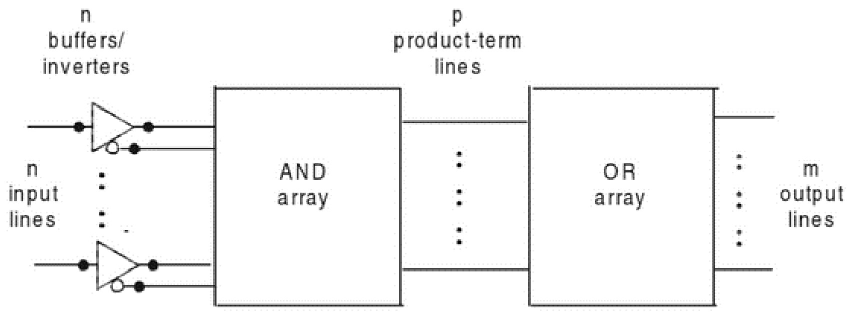
PROGRAMMABLE LOGIC DEVICE (OR PLD)

- A programmable logic device (or PLD) is a device which is used in many digital electronic designs.
- Unlike a logic gate, which has a fixed function, these PLDs are very flexible and can implement both combinational and sequential logic functions, such as AND, OR, NAND, counter and shift registers on the same chip.
- PLD performs most of the complicated logic functions which are programmed into a single chip.

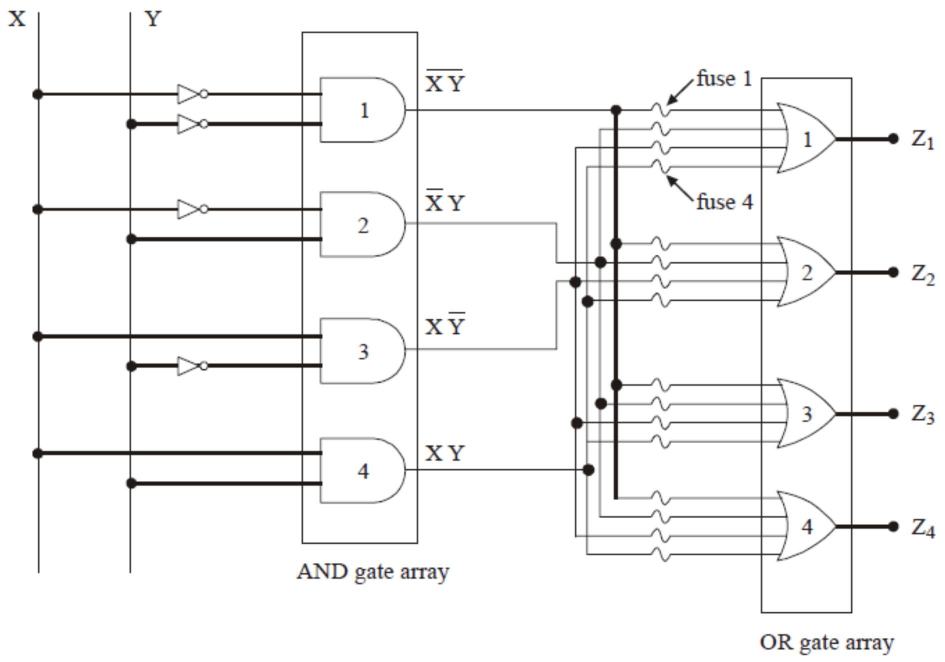
- Using these PLDs, an effective digital design circuit will be achieved. For the PLD programming we have to select inputs, outputs and the logical relationships of our required system. Then these details will be programmed into a single IC and that does the necessary operation.

PLD Logic Circuit

Basically a programmable logic device (PLD) consists of two types of logic array. One is an array of AND gates and the other is an array of OR gates.



These arrays are programmable to meet the desired logic functions. Following figure shows an example of small PLD.



The AND array produces the product term whereas the OR array produces sum of these products. Thus the output of PLD is called sum-of-product.

The connection between the AND gates and OR gates are in the form of programmable fuses. For a specific function few of the fuses are broken selectively while the other remains intact.

The **blowing of the fuses** is done by the manufacturer under the instruction of designer. The process of blowing fuse is called **programming** because it produces the desired circuit pattern interconnecting the gates.

Based on our requirement the PLDs are programmed. But once the PLD is programmed, the device will permanently generate the selected output functions.

Types of PLD

Programmable logic device may be classified into many types. They are namely,

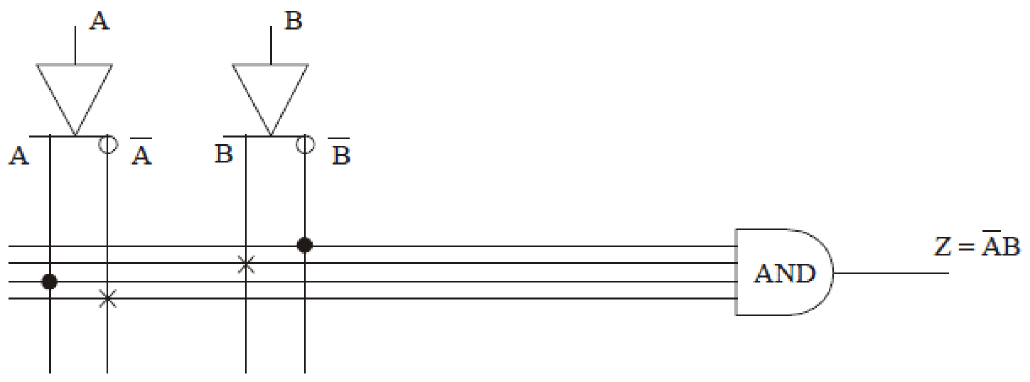
- Programmable read-only memory (PROM)
- Programmable array logic (PAL)
- Programmable logic array (PLA)
- Generic Array Logic (GAL)
- Complex programmable logic device (CPLD)
- Field programmable gate array (FPGA)

Advantages of Programmable Logic Devices

- Low space requirement i.e., high density.
- Low development cost.
- Low power consumption.
- Design security and flexibility.
- Easy programming.
- Compact circuitry.
- High switching speed

PLD Symbol

PLD manufacturers have adopted a simplified symbol to represent the internal circuitry. Following figure shows the circuit symbol of a 2-input PLD.



As seen from this figure, the two logic input gates A and B have two outputs each (i.e., normal output and its complement). These outputs are connected to a 4-input AND gate by connecting the wires like 'x' or '•'. Here 'x' means an intact fuse. A dot (•) means a hardwire connection. This type of connection cannot be changed.

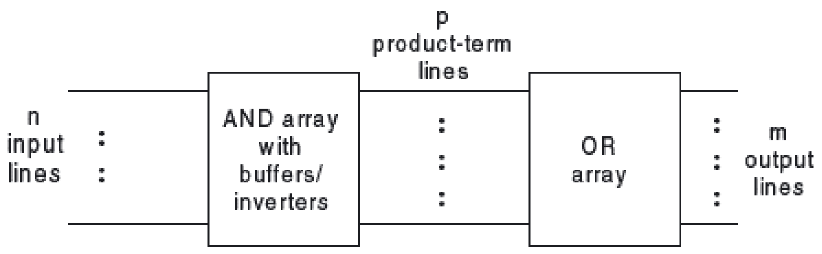
From figure, the inputs A and $\bar{B}$ are connected to generate the product AB (i.e., Z = A$\bar{B}$). Here A and B are not connected to the AND gate (i.e., intact fuse). So they do not have any effect on its output.

ROM (READ ONLY MEMORY)

A ROM is essentially a memory device for storage purpose in which a fixed set of binary information is stored. An user must first specify the binary information to be stored and then it is embedded in the unit to form the required interconnection pattern.

ROM contains special internal links that can be fused or broken. Certain links are to be broken or blown out to realize the desired interconnections for a particular application and to form the required circuit path. Once a pattern is established for a ROM, it remained fixed even if the power supply to the circuit is switched off and then switched on again.

A block diagram of ROM is shown in following figure.



It consists of **n** input lines and **m** output lines. Each bit combination of input variables is called an **address** and each bit combination that is formed at output lines is called a **word**. Thus, an address is essentially a binary number that denotes one of the minterms of **n** variables and the number of bits per word is equal to the number of output lines **m**. It is possible to generate p = 2^n number of distinct addresses from **n** number of input variables.

A ROM is characterized by the number of words $\mathbf{2^n}$ and number of bits per word m and denoted as $\mathbf{2^n \times m}$ ROM.

☞ As an example, consider a **32 × 8 ROM**. The device contains 32 words of 8 bits each. This means there are eight output lines and there are 32 numbers of distinct words stored in that unit, each of which is applied to the output lines. The particular word selected from the presently available output lines is determined by **five input variables**, as there are five input lines for a 32 × 8 ROM, because $2^5 = 32$. Five input variables can specify 32 addresses or minterms and for each address input there is a unique selected word. Thus, if the input address is 0000, word number 0 is selected. For address 0001, word number 1 is selected and so on.

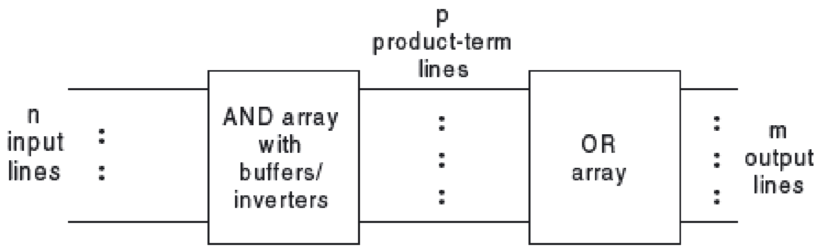
☞ A ROM is sometimes specified by the total number of bits it contains, which is **2^n × m**. For example, a **4,096-bit ROM** may be organized as **512 words of 8 bits** each. That means the device has 9 input lines (2^9 × m = 512) and 8 output lines.

Example

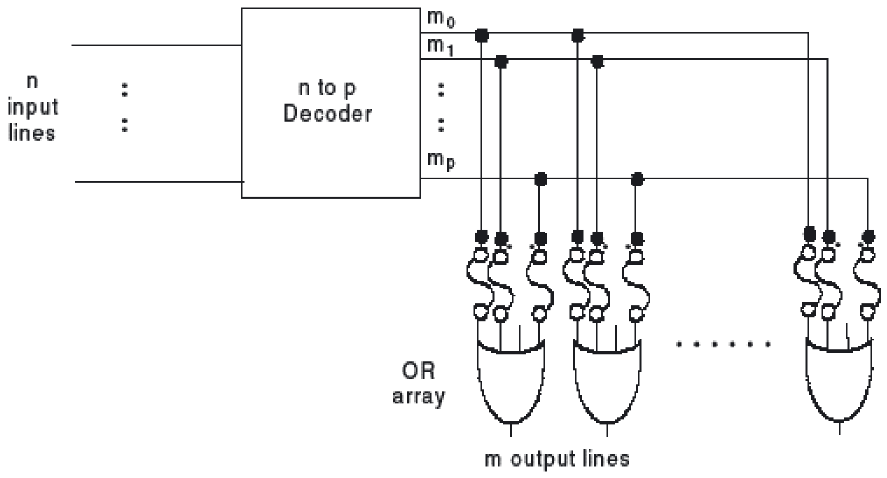
Draw the equivalent logic diagram of a 2^n x m ROM is shown and its logic diagram with PLD notation.

Solution

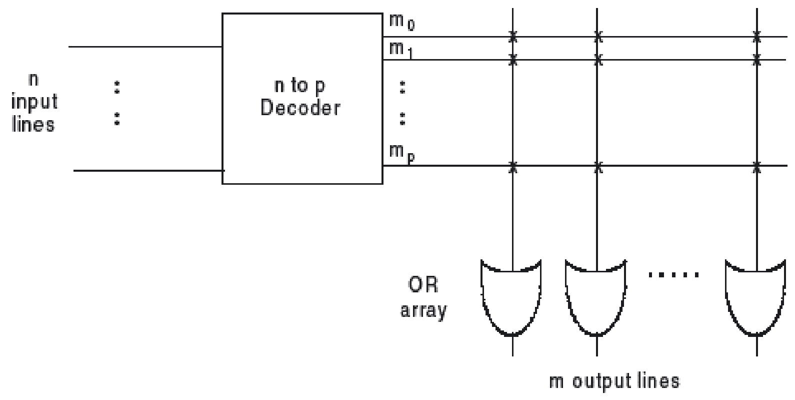
In following figure, the block consisting of an AND array with buffers/inverters is equivalent to a decoder.



The decoder basically is a combinational circuit that generates 2^n numbers of minterms from n number of input lines. 2^n or p numbers of minterms are realized from n number of input variables with the help of n numbers of buffers, n numbers of inverters, and 2^n numbers of AND gates. Each of the minterms is applied to the inputs of m number of OR gates through fusible links. Thus, m numbers of output functions can be produced after blowing of some selected fuses. The equivalent logic diagram of a 2^n x m ROM is shown below.



Its PLD notation is redrawn in following figure.



Programmable Read Only Memory (PROM)

Programmable Read Only Memory (PROM). It is more economic in cases requiring small quantities. In this method the manufacturer provides the PROM with all 0s (or all 1s) in every bit of the stored words. The required links are broken by application of current pulses. This allows the user to program the device in his own laboratory to obtain the desired relationship between input addresses and stored words.

Example

Implement the following Boolean functions are to be developed using ROM.

F_1 (A, B, C) = (0, 1, 2, 5, 7) and

F_2 (A, B, C) = (1, 4, 6).

Solution

When a combinational circuit is developed by means of a ROM, the functions must be expressed in the **sum of minterms** or **by a truth table**.

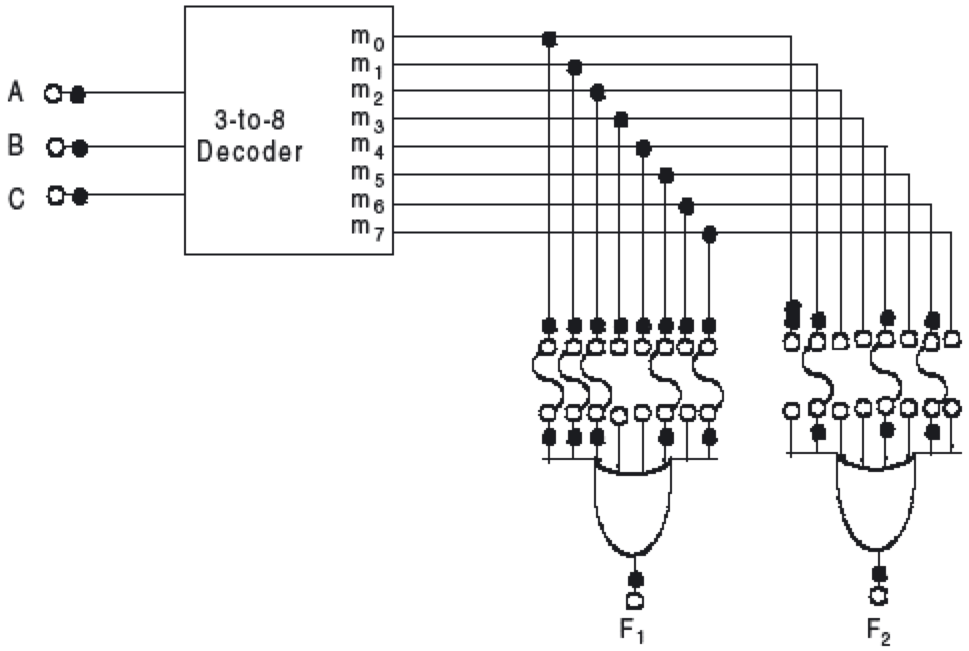
The truth table of the above functions is shown below.

Decimal	Input			Output	
	A	B	C	F_1	F_2
0	0	0	0	1	0
1	0	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	0	1

7	1	1	1	1	0

Since there are three input variables, a ROM containing a **3-to-8 line decoder** is needed.

In addition, since there are **two output functions**, the OR array must contain at least two OR gates. That means, a $2^3 \times 2$ ROM or 8×2 ROM is to be employed to realize the above functions. The logic diagram of the ROM after blowing off the appropriate fuses is shown below.



Example

Implement the following Boolean expressions using a PROM.

$f_1 (x_2, x_1, x_0) = \Sigma m (0, 1, 2, 5, 7)$

$f_2 (x_2, x_1, x_0) = \Sigma m (1, 2, 4, 6)$

Solution

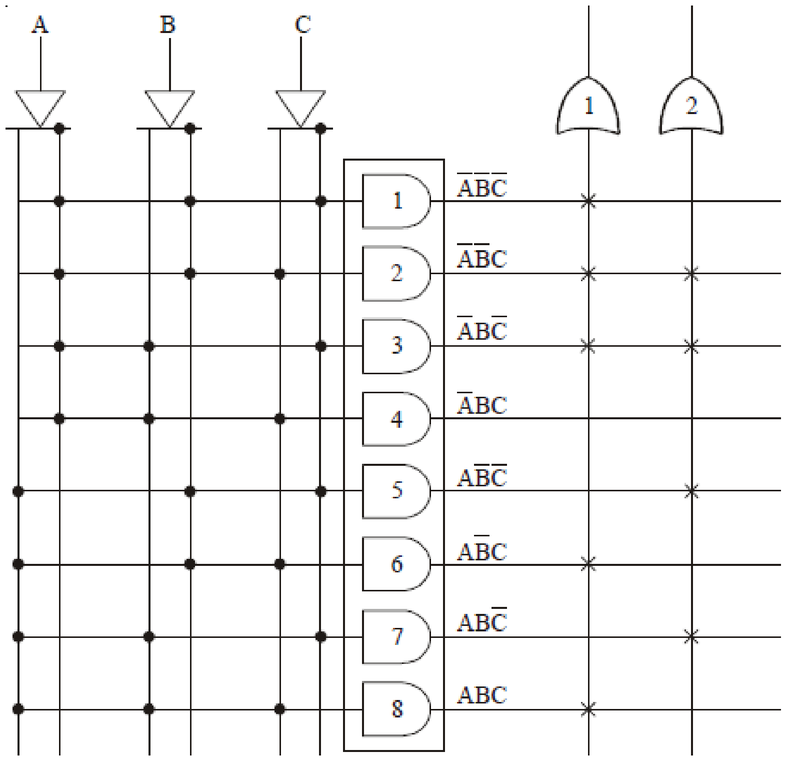
By seeing the given output function then an three variables x_2, x_1 and x_0. Therefore PROM is required to have minimum of 2^n $(2^3) = 8$ locations. Following table shows the data in each location of PROM.

Decimal	Input			Output	Data bit	
	A	B	C		D_1	D_2
0	0	0	0	$\bar{A}\bar{B}\bar{C}$	1	0
1	0	0	1	$\bar{A}\bar{B}C$	1	1

2	0	1	0	$\bar{A}B\bar{C}$	1	0
3	0	1	1	$\bar{A}BC$	0	0
4	1	0	0	$A\bar{B}\bar{C}$	0	1
5	1	0	1	$A\bar{B}C$	1	0
6	1	1	0	$AB\bar{C}$	0	1
7	1	1	1	ABC	1	0

By seeing the truth table, database D_0 represent function f_1 and D_1 represent function f_2.

For the desired output expression f_1, fuses links from AND gate 1, 2, 3, 6, 7 to OR gate 1 should be blown out. This make the output at f_1. Similarly, for the desired output f_2, fuses links from AND gate 2, 3, 5, 7 to OR gate 2 should be blown out as shown in given figure.



Example

Implement the following boolean expression using a ROM and also a decoder.

$d_0 (a_1 , a_0) = \Sigma m\ (1, 2)\ (11.1)$

$d_1 (a_1 , a_0) = \Sigma m\ (3)$

Solution

See following truth table.

Decimal	Input		Output	Data bit	
	a_1	a_0		d_0	d_1
0	0	0	$\overline{a}_1\overline{a}_0$	1	0
1	0	1	$\overline{a}_1 a_0$	1	1
2	1	0	$a_1\overline{a}_0$	1	0
3	1	1	$a_1 a_0$	0	0

For ROM design, there is need of two inputs and two outputs.

Number of inputs or address lines, n = 2.

Number of outputs or data lines, m = 2.

Number of AND gates required = 2^n = 22 = 4.

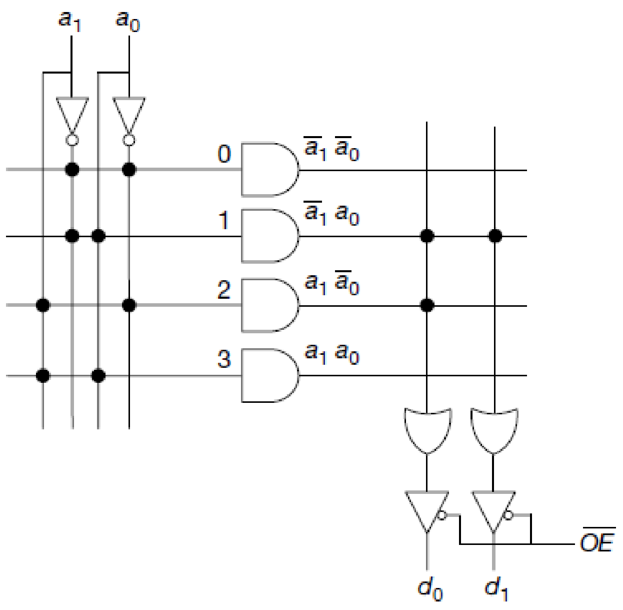
Number of fixed links before AND gate = $2 \times (n \times 2^n) = 2 \times (2 \times 2^2) = 16$.

Number of OR gates required = m = 2

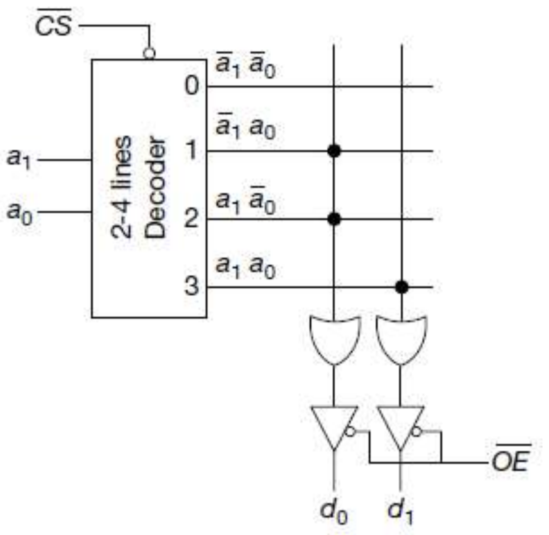
Number of fixed links after AND gate array and before OR gate array

$$= 2^n \times m = 2^2 \times 2 = 8.$$

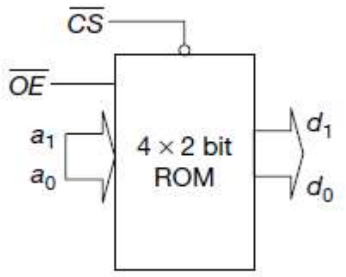
Complete diagram is given below. AND gates and OR gates are having fixed contacts or links.



Following figure gives the diagram using **decoder**.



Block diagram of ROM is given below.

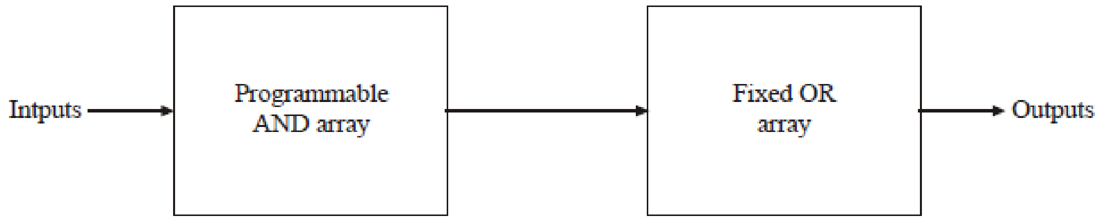


Two number of tristate buffers are used to read data. Active low signal, $\overline{OE}$ acts as read signal.

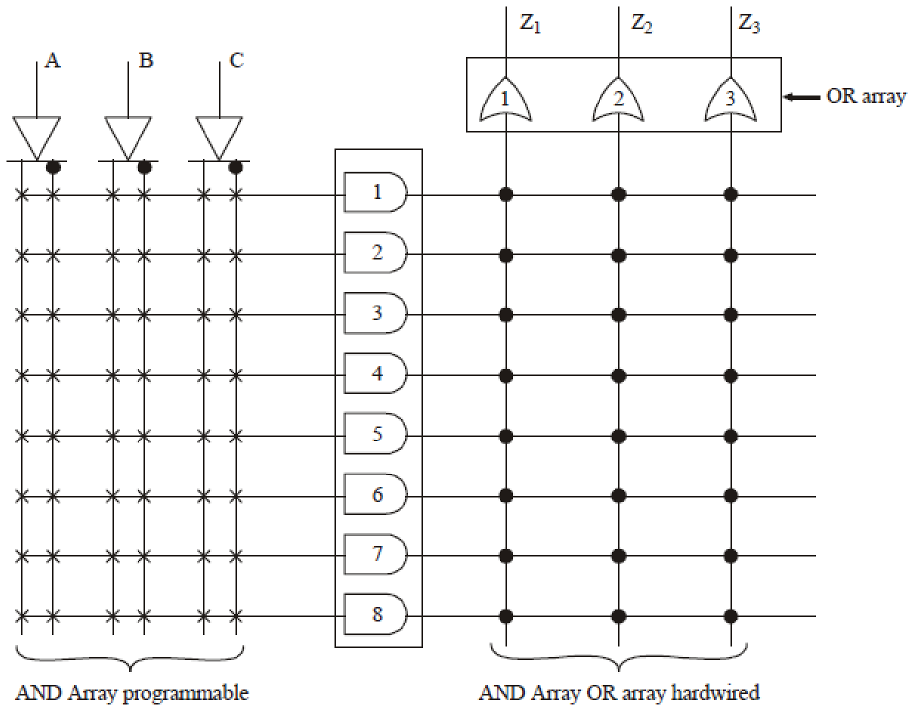
PROGRAMMING ARRAY LOGIC (PAL)

Like in the PROM PLDs, it has the same AND and OR gate arrays in it. But **inputs** to AND arrays are **programmable** and inputs to **OR** arrays are **hardwired**.

Following figure shows the simple block diagram of a PAL device.



Following figure shows the configuration of AND and OR arrays for PAL architecture.



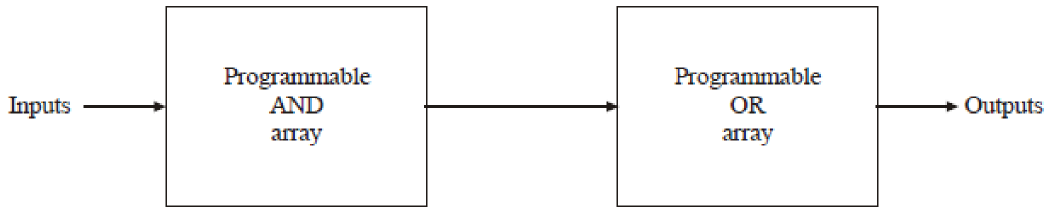
PAL architecture

As seen from the figure, there are three inputs A, B and C connected to the buffer gates. The outputs of these buffer gates are suitably connected to 8 AND gates that are programmable one. Then the outputs of these AND gates are connected to the fixed OR gate arrays.

PROGRAMMABLE LOGIC ARRAY (PLA OR FPLA)

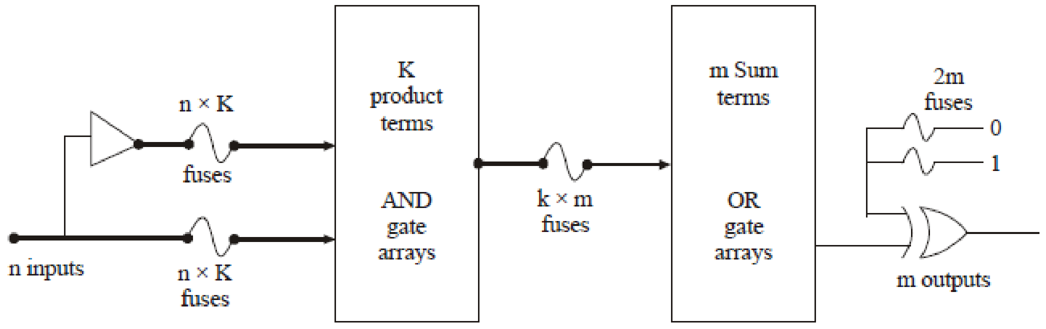
Programmable Logic Array (or PLA) is having the characteristics of PROM and PAL. It has the programmable connections for both AND and OR arrays. So it is the most **flexible** type of PLDs compared to PROMs and PALs.

Following figure shows a simple block diagram of PLA.



Normally in PLAs, inputs are fed into a number (K) of AND gates where $K < 2^n$, (n is the number of inputs). Every AND gate can be programmed to generate a product term of the input variables and does not generate all the minterms as in the ROM. The AND and OR gates are connected with the fuses. The specific boolean functions are implemented in sum of products form by opening appropriate fuses and leaving the desired connections.

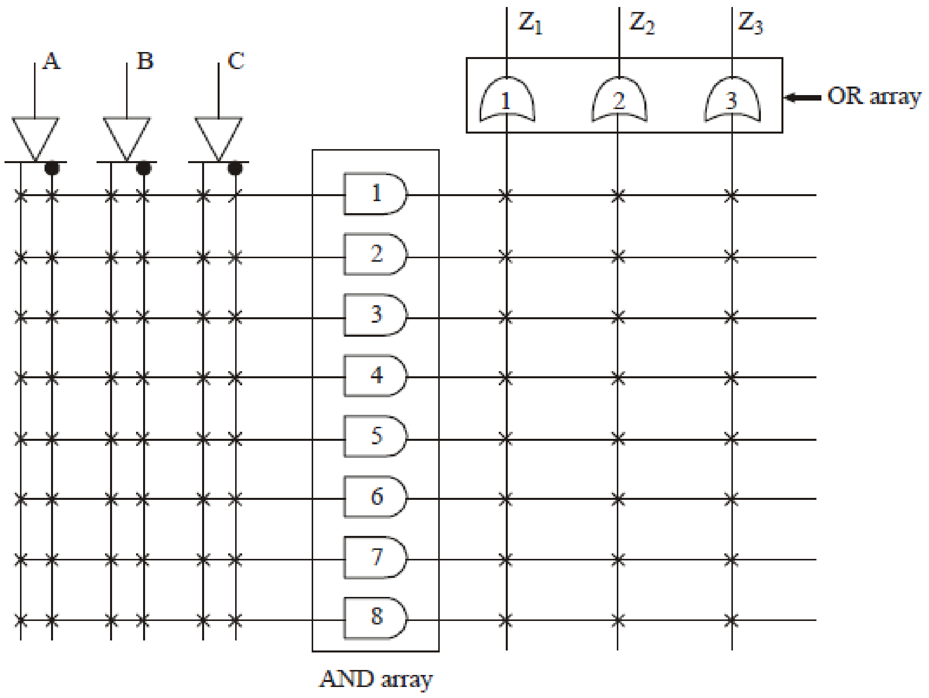
Following figure shows the implementation of PLA. It consists of 'n' inputs, 'm' outputs and 'K' product terms. The product term consists a group of 'K' AND gates each of 2^n inputs.



Here the fuses are inserted between all n inputs and their complement values to each of the AND gates. Fuses are provided between the outputs of the AND gates and the inputs of the OR gates. Since the PLA has m outputs, the number of OR gates is m. The output of each OR gate goes to a XOR gate, where the other input has two sets of fuses, one connected to logic 0 and the other to logic 1. This allows the output function to be generated either in the true form or in the complement form. The output is inverted when the XOR input is connected to 1 (Since $X \oplus 1 = X$). The output does not change when XOR input is connected to 0 (Since $X \oplus 0 = X$).

So, the total number of programmable fuses = $2^n + K + K \times m + 2\,m$.

Following figure shows the architecture of the PLA.



There are **three inputs** passing through the **buffer gates**. The outputs from the buffer gates are passed to the 8 AND gates array. Then the outputs of AND gates are fed to the OR gate array. From the OR gate, we will get the actual logic functions.

Here **both AND arrays and OR gate arrays are programmable**. So it considers two sets of fuses (i.e., one set for AND array and one set for OR array). The AND gate array provides the product terms and the OR gate logically sum these product terms and gives the required logic function.

Since it has two sets of fuses, it is **difficult to program** the PLA. PLAs are also often referred as **FPLA (field programmable logic array)**.

GENERIC ARRAY LOGIC

The continuous development in design and development of PAL device has led to the introduction of configurable PAL (or CPAL) to enhance the output capabilities. The word configurable is also known as generic. Nowadays PALs are no longer used, because GALs provide a superior functionality and can do all the functions of PAL. GALs are relatively small, easily available and inexpensive.

A Generic Array Logic (or GAL) is one type of configurable PAL. It can be used as a pin-to-pin replacement for many PAL devices. These devices can be erased and re-programmed whenever a need arises.

Most of the GAL's logic functions are performed by programming the AND array. But selecting flip-flops, input/output configurations and polarities of ORs are done by programming a configurable Input/Output and feedback structure. This is called as macrocell.

ASSIGNMENT

ENCODERS & DECODERS

Q.1. (BPUT 2022, 2 marks): Differentiate between a priority encoder and conventional encoder.

Answer: A priority encoder has a priority function which allows it to produce an output corresponding to the highest-order input.

Q.2. (AKTU 2023, 2 marks): What is the difference between Multiplexer and Encoder.

Answer: The digital circuits that perform encoding of digital information are called encoders while digital circuits that decode the coded digital information are called decoders. An encoder with enable pins is called multiplexer while a decoder with enable pins is called demultiplexer.

Q.3. (JNTUH 2023, 2 marks): What is the difference between Decoder and Demultiplexer?

Answer: Already answered.

Q.4. (RGPV 2022, RTU 2019, 6 marks): Design and explain a 4-bit priority encoder.

Answer: Described in this module.

Q.5. (UTU 2023, 5 marks): Design and implement decimal to BCD encoder.

Answer: Described in this module.

Q.6. (VTU 2022, 6 marks): With the aid of general structure, clearly distinguish between a decoder and encoder.

Q.7. (GTU 2020, 2022, 4 marks): Explain briefly 3 line to 8 line decoder.

Answer: Described in this module.

Q.8. (GTU 2023, 4 marks): Implement 4x16 decoder using two 3x8 decoder.

Answer: Described in this module.

Q.9. (BPUT 2022, 10 marks): Construct a 5 to 32 line decoder with four 3 to 8 line decoders with enable and a 2 to 4 line decoder. Use block diagrams for the components.

Answer: Solved in this module.

Q.10. (JNTUH 2020, 8 marks): Implement the multiple output combinational logic circuit using a 4 line to 16 line decoder.

$f1 = \Sigma m (1, 2, 4, 7, 8, 11, 12, 13, 14, 15)$

$f2 = \Sigma m (0, 1, 3, 5, 8, 9, 15)$

$f3 = \Sigma m (2, 3, 4, 7)$

$f4 = \Sigma m (0, 1, 3, 4, 7, 9)$

Q.11. (VTU 2022, 8 marks): Implement full subtractor using a decoder and two NAND gates and write its truth table.

Answer: Solved in this module.

MULTIPLEXERS

Q.12. (RTU 2018, 8 marks): How Mux is used for implement combinational logic? Implement y = A + BCD using a Mux.

Q.13. (PTU 2020, 5 marks): Discuss the addressed multiplexer configuration.

Q.14. (GTU 2021, 7 marks): Draw the logic circuit of 4-to-1 Multiplexer and explain its working with the help of truth-table.

Answer: Described in this module.

Q.15. (GTU 2022, 2023, 4 marks): Implement the 8×1 MUX using two 4×1 MUX.

Answer: Solved in this module.

Q.16. (AKTU 2020, 2021, 2 marks): Implement a 4:1 multiplexer using 2: 1 multiplexer.

Answer: Solved in this module.

Q.17. (RGPV 2022, 8 marks): Design and explain a 16:1 mux using 2:1 mux.

Answer: Solved in this module.

Q.18. (RTU 2018, 2 marks): Find the total no. of select line when a 8 x l Mux is implemented using 2 x l Mux.

Q.19. (BPUT 2020, 2 marks): Design a 4:1 multiplexors using smaller multiplexors.

Answer: Solved in this module.

Q.20. (UTU 2023, 5 marks): What is multiplexer? Design 4:1 multiplexer

Answer: Described in this module.

Q.21. (JNTUH 2022, 9 marks): Explain the differences between a MUX and a DEMUX. Realize 16-input multiplexer by cascading of two 8-input multiplexers.

Answer: A Multiplexer generates a single output for data and signals. A Demultiplexer generates multiple outputs for data and signals. It processes the digital data and info by collecting them from multiple sources and integrating them into a single source as the output.

Part (ii) is described in this module.

Q.22. (AKTU 2020, 10 marks): Implement a full adder by using 8:1 multiplexer.

Answer: Solved in this module.

Q.23. (JNTUH 2020, 7 marks): Design a 32:1 Multiplexer using two 16:1 and 2:1Multiplexers.

Answer: Solved in this module.

Q.24. (GTU 2023, 7 marks): Implement the following Boolean functions with a multiplexer and Decoder. $F(w, x, y, z) = \Sigma (2, 3, 5, 6, 11, 14, 15)$

Q.25. (AKTU 2023, 10 marks): Implement the function $F = \Sigma m (0, 1, 3, 4, 7, 8, 9, 11, 14, 15)$ using 8:1 mux.

Q.26. (BPUT 2022, 6 marks): Implement the following function with a MUX

$$F(A, B, C, D) = \Sigma (0, 1, 3, 4, 8, 9, 15)$$

Answer: Solved in this module.

Q.27. (GTU 2020, 4 marks): Implement the given function using 8 x 1 multiplexer.

$$F(A,B,C,D) = \Sigma m (0,1, 2, 3, 5, 8, 9, 11, 14).$$

Answer: A similar problem is solved in this module.

Q.28. (GTU 2023, 7 marks): Implement the following Boolean functions with a multiplexer and Decoder.

$$F(w, x, y, z) = \Sigma (2, 3, 5, 6, 11, 14, 15)$$

Answer: A similar problem is solved in this module.

Q.29. (UTU 2023, 10 marks): Implement $\bar{A}B\bar{C}+A\bar{B}\bar{C}D+\bar{A}CD$ using multiplexer.

Answer: A similar problem is solved in this module.

Q.30. (GTU 2023, 7 marks): Implement the switching function using F = Σm (0, 1, 3, 4, 12, 14, 15) using an 8 input MUX.

Answer: A similar problem is solved in this module.

Q.31. (UTU 2023, 5 marks): Describe 4 digit multiplexer display system.

Answer: Described in this module.

HAZARDS IN COMBINATIONAL CIRCUITS

Q.32. (UTU 2023, 10 marks): What is hazard and how to determine hazards in Combinational Circuits? Explain with example.

PLA

Q.33. (BPUT 2020, 8 marks): Write a detailed note on PLD(Programmable Logic Device).

Q.34. (RTU 2019, 15 marks): Write a short note on following :

(i) FPGA

(ii) PLA

(iii) CPLD

(iv) PAL

Q.35. (UTU 2023, 5 marks): Write a short notes on PAL and PLA.

Q.36. (GTU 2020, 2022, 2023, 4 marks): Compare ROM, PLA and PAL.

Q.37. (PTU 2020, 5 marks): What are the applications of ROM and PROM memories? Explain.

Q.38. (PTU 2020, 5 marks): Demonstrate the access time of PROM with a timing diagram

Q.39. (PTU 2020, 2 marks): "PAL has reprogrammable AND array, whereas GAL has programmable AND array". Comment.

Q.40. (GTU 2021, 7 marks): Draw and explain PLA block diagram. Also draw the Programmable Logic Array with three inputs, three product terms, and two outputs.

Q.41. (PTU 2020, 10 marks): List out the steps to be consider for PLA folding algorithm.

Q.42. (BPUT 2020, 6 marks): With the help of a proper diagram, write a note on Programmable Logic Arrays (PLA).

Q.43. (AKTU 2023, 10 marks): Define PLD’s. Implement the following function using PLA

(i) F1 = Σm (0,3,4,7)

(ii) F2 = Σm (1,2,5,7)

Q.44. (BPUT 2022, 8 marks): Implement the Boolean function with PLA

F1 (A, B, C) = Σ (0, 1, 2, 4)

Q.45. (AKTU 2023, 10 marks): Explain Decoder with neat diagram. Implement the logic expression Y= Σm (2, 4, 6, 7) using decoder as ROM.