category, the complexity is of more than 10,000 gates. For example, a microprocessor is a VLSI circuit which includes many SSI, MSI and LSI subsystems inside it.

(e) Programmable Logic Devices (PLD)

The term PLD is broadly applied to the integrated circuits that have a fixed underlying set of components such as logic gates (AND or OR) or some complex logic subsystems, but whose interconnections can be tailored by the circuit designer through external programming in an application specific way that makes them general purpose design components. The number of components in PLDs may be very large, so that it spans the range of complexity from MSI to VLSI. Under these PLDs we have Programmable Array Logic (PAL), Programmable Logic Array (PLA), Programmable Logic Element (PLE) and Gate Arrays etc.

In this chapter, we shall develop the design procedure of the combinational circuit using the logic components mainly of the MSI and LSI category. We shall see that the complete design of the circuit with MSI components needs some help of the logic gates of the SSI group also. MSI and LSI components are very much function specific and the design concept needs some new approach. As the MSI and LSI component integrates more digital subsystems in a single chip, the number of chip-count of the circuit and necessary wiring for the interconnections are reduced and hence the circuit becomes more reliable and cost effective.

Following is the list of the MSI and LSI components whose functional properties, logic structure and a few applications of each will be considered in this chapter :

Multiplexer, Demultiplexer, Magnitude Comparator, Decoder, Decoder Driver, Encoders, Code-Converters, Parity Generator and Checker, Logic Shifter and PLDs.

8.2 MULTIPLEXER

8.2.1 Introduction

Multiplexing means transmiting a large number of information units over a small number of channels or lines. The multiplexer is a combinational circuit which serves this purpose. A multiplexer circuit, in general, may have M data inputs and one output line. The selection of input to output transfer path is controlled by set of N select lines. The select lines are also called the address of the inputs. The relationship between these N address lines and the M data input lines is $M = 2^N$. Because, using N-bit binary inputs to the select lines we may generate 2^N address codes and each code addresses one data input line out of M. The multiplexer in short is called MUX. The MUX with M inputs and 1 output is mentioned in the literature as (M × 1) or as M-to-1 MUX. For example, a MUX with 4 data inputs and 1 output is called 4 × 1 or 4-to-1 MUX. Besides these inputs and output, usually, for all kinds multiplexers at least one input line is available which is known as Enable line. This enable line is either active low or active high type. If the enable input is kept inactive then the multiplexer circuit becomes inactive. Commercially available MUXs usually come with active low enable lines. This means that the multiplexer circuit will be active i.e. will

follow its functional behavior if the enable line is made logic low otherwise there will be no variations at the output though the different inputs are selected using select codes applied to the select inputs. Under disabled condition the output will show either a logic-0 or logic-1 irrespective of the select inputs. Whether the state of the output will be logic-0 or 1 under disabled condition of the muttiplexer depends on the internal circuit design of the multiplexer.

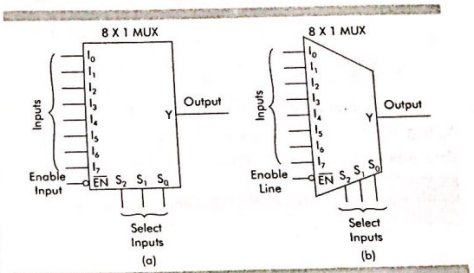


Fig. 8.1 Different block-diagrams representation of 8 X 1 MUX

The block diagram for a 8-to-1 MUX is shown in Fig. 8.1. The Fig. 8.1 (a) and (b) are two alternative logic symbols that are used for the 8-to-1 MUX. In the logic symbol we can see that there are 8 data input lines (I_0, I_1, I_2 , ..., I_7), three select lines (S_2, S_1, S_0), one active-low enable line ($\overline{EN}$) and one output line Y. Out of these 8 data inupts, any one input is selected at a time by the select code for receiving digital data and for transmission of the data from the selected input to the output Y. The selection of the particular input is done by setting the proper select codes to the select inputs S_2, S_1, and S_0. For example if $S_2S_1S_0 = 010$ then the I_2 input will be selected for receiving digital data i.e. the binary bit 1 or 0 through the input I_2 will be passed to be output Y and at this time all other input lines I_0, I_1, and I_3 to I_7 will remain deselected i.e. no digital signals will be able to pass through them to reach the output Y. That is, the selection of input is done one at a time. The function table (TABLE 8.1) explains the complete behavior of a 8-to-1 MUX. For this illustration, the enable input (EN) is assumed to be active-low and under disabled condition (EN=1) the output (Y) is assumed to be low.

Enable (EN)	Select Inputs S_2	S_1	S_0	Selected Input	Output Y will be same as Input
0	0	0	0	I_0	I_0
0	0	0	1	I_1	I_1
0	0	1	0	I_2	I_2
0	0	1	1	I_3	I_3
0	1	0	0	I_4	I_4
0	1	0	1	I_5	I_5
0	1	1	0	I_6	I_6
0	1	1	1	I_7	I_7
1	X	X	X	None	0

TABLE 8.1 Functional table of a 8-to-1 MUX

From this function table we may write down the logic equation for the output Y in terms of the inputs (I_0, I_1, ..., I_7) and the select inputs (S_2, S_1 and S_0) and the active-low enable

input $\overline{EN}$. The logic equation for this 8-to1 MUX using the TABLE 8.1 is

$$Y = (\bar{S}_2\bar{S}_1\bar{S}_0I_0 + \bar{S}_2\bar{S}_1S_0I_1 + \bar{S}_2S_1\bar{S}_0I_2 + \bar{S}_2S_1S_0I_3 + S_2\bar{S}_1\bar{S}_0I_4 + S_2\bar{S}_1S_0I_5 + S_2S_1\bar{S}_0I_6 + S_2S_1S_0I_7)\ \overline{EN}$$

$$= (\sum_0^7 m_iI_i)\,\overline{EN} \quad \text{.... (8.1)}$$

where m_i represents ith minterm of select variable S_2 S_1 and S_0 (e.g $\bar{S}_2\bar{S}_1\bar{S}_0 = m_0$, $\bar{S}_2\bar{S}_1S_0 = m_1$, $\bar{S}_2S_1\bar{S}_0 = m_2$ etc), I_i is the i^{th} data input line and $\overline{EN}$ is the active low enable input. In general the logic equation for a MUX with n number of select inputs and active low enable input may be written as

$$Y = (\sum_0^{2^n-1} m_iI_i)\,\overline{EN}. \quad \text{.... (8.2)}$$

From the above discussions we may view one MUX as a rotary mechanical swtich with select inputs determining the position of the switch arm to the appropriate input I_i. This is shown in Fig 8.2.

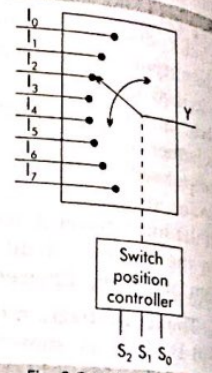


Fig. 8.2 Multiplexer as a rotary switch

8.2.2 Design of 4-to-1 Multiplexer

The Eqn 8.2 is for 2^n-to-1 MUX and this equation represents a sum of product form. Therefore, Eqn. 8.2 can be realised using a 2-level AND-OR logic circuit. For a 4-to-1. MUX

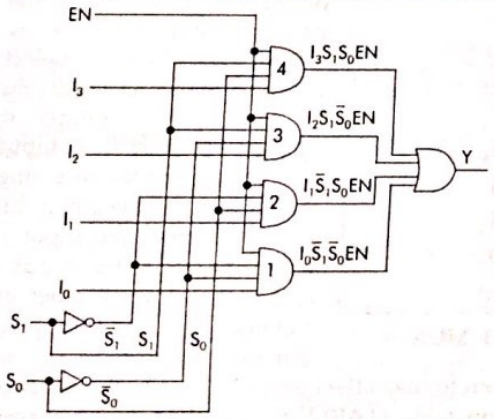


Fig. 8.3(a) Logic diagram of a 4-to-1 MUX with active high enable input (EN)

Enable EN	Select Inputs S_1	S_0	Output
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3
0	x	x	0

TABLE 8.2 (a) Function table of a 4-to-1 MUX with active high enable input.

there will be 4 data inputs, 1 output and optionally there may be one enable input. The enable input may be active low or active high type. Also, some MUXs give output polarity inverse of the selected input. Let us design a 4-to-1 MUX with active high enable (EN)

input and one output (Y). The fu
this table, the logic equation fo

$$Y = (\bar{S}_1\bar{S}_0I_0 + \bar{S}_1S_0I_1 + S_1\bar{S}$$

where S_1 and S_0 are the selec
4 data inputs and EN is the activ
enable input is made logic-1 the
4 rows of the functional table of
8.2(a). Otherwise for EN = 0 th
ignoring of select and data inpu
circuit for the Eqn (8.3) is show
diagram representation is show

From this logic circuit it is s
AND gates to generate the pr
inverters to get $\bar{S}_1$ and $\bar{S}_0$ from
4-input OR gate to get the final
and one enable line, we need 2^n
of inverters and one OR gate v

From the circuit of Fig. 8.3 (
= 0, S_0=0 and EN=1 the outputs

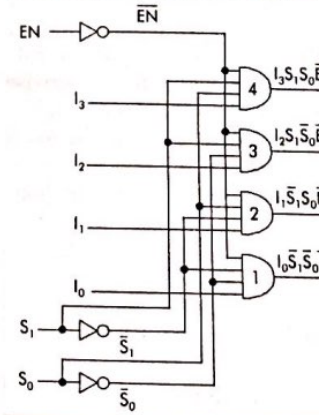


Fig. 8.3(c) Logic circuit of a 4-to- active low enable inpu

the output of AND gate-1. The
collects the outputs of all the
analysis it is easy to show that
because the AND gate-2 give

S_1 S_0 = 10 and EN = 1, the output Y is I_2 as the AND gate–3 gives an output I_2 with all other AND-gate outputs at logic-0; for S_1 S_0 = 11 and EN = 1, the output Y is I_3 as the AND gate-4 gives an output I_3 with all other AND gate outputs at logic-0. But, for EN = 0 all the AND gates will give logic-0 output irrespective of the different select inputs on S_3, S_2 and S_0. This leads to the logic-0 output at Y for all times so long as EN is low. This is indicated in the last row of the TABLE 8.2(a).

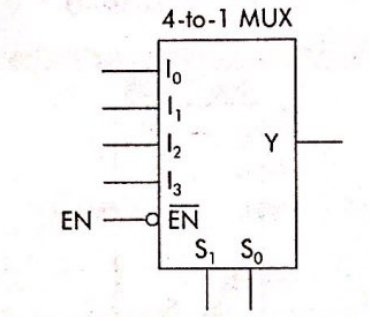


Fig. 8.3 (d) Block diagram of 4-to-1 MUX with active low enable input.

The logic circuit of a 4-to-1 MUX with active low enable input is shown in Fig. 8.3 (c) and its logic symbol is shown in Fig. 8.3 (d). The function table of the circuit in Fig. 8.3 (c) is shown in TABLE 8.2 (b).

The output Y of this MUX can be written from the TABLE 8.2(b) as $Y = (I_0\overline{S}_1\overline{S}_0 + I_1\overline{S}_1S_0 + I_2S_1\overline{S}_0 + I_3S_1S_0)\overline{EN}$.

8.2.3 Commercially Available 4-to-1 Multiplexer IC chips

TABLE 8.3 shows a list of currently available commercial TTL 4 to-1 MUX IC chips with their brief description. From the table it may be seen that each IC chip is dual 4-to-1 MUX which means that each chip contains two identical units of 4-to-1 MUX inside. The two units are not completely independent because their select inputs (S_1 and S_0) are common. From the last column of this table it is evident that the MUXs may produce three types of outputs **(i)** Same as inputs i.e. polarity of the output is same as that of the data inputs **(ii)** Inverse of the inputs; i.e. the polarity of the output is complement of the data inputs. **(iii)** same as that of the cases (i) and (ii) but the output is tristated. For the third category of multiplexers (IC 74253 and IC 74353) there is a control input called "**output control**" line which when made logic high the output goes into a high impedance state otherwise the IC 74253 and 74353 are similar to 74153 and 74352 respectively. The function tables of 74153, 74352, 74253, and 74353 are shown in TABLE 8.4 (a), (b), (c) and (d). The MUX with tristated outputs are usually used in a bus oriented circuits. For all these MUXs the enable inputs and the tristate-control inputs are active low. Brief description and the logic circuit of the different 4-to-1 MUXs are given below.

TTL IC Chip number	No. of Pins	Technology used	Functional description	Polarity of output
74 153	16	Std TTL, L, LS, S, HC	Dual 4-to-1 MUX	Same as Input
74 352	16	LS, S, ALS, AS, HC	"	complement of inputs
74 253	16	LS, S, ALS, AS, HC	"	Same as input but the output tristated
74 353	16	LS, ALS, AS, HC	"	complemented but the output tristated

TABLE 8.3 Commercially available 4-to-1 multiplexer IC chips

Control (2G)

Fig. 8.5 (a) Logic circuit of 74253.

Fig. 8.5 (b) Block diagram of 74253

8.2.4 Commercially Available 8-to-1 Multiplexer IC Chips

TTL IC Chip number	No. of Pins	Technology Used	Functional description	Polarity of outputs
74151	16	LS, S ALS, AS, HC	Single 8-to-1 MUX	Two outputs. Polarity of one is same as inputs and the other is inverse of inputs.
74152	16	LS, HC	"	One output, inverse of input.
74251	16	LS, S, ALS, HC	"	Two tristated outputs. One is same as input and the other is inverse of input.

system. ...multiplexes are used to interface with a single bus of a

8.3 Cascading of Multiplexers

The input capacity of multiplexer can be increased by cascading several multiplexers. By cascading **m** numbers of **n-to-1** MUXs we can have a multiplexing action equivalent to **(mxn)-to-1** MUX. Thus, the input capacity is increased from **n** to **(mxn)**. This cascading can be done in two ways- **(i)** Using **m** numbers of **n-to-1** MUXs for inputting data from **(mxn)** sources and one m-input OR-gate for outputting data. The select lines of all the **m** numbers multiplexers are made common and the multiplexers are enabled one after another by using some input logic circuit called decoding logic circuit. The OR gate used at the output stage needs **m** inputs because it collects the **m** outputs from **m** numbers of MUXs. **(ii)** Using **m** number of **n-to-1** MUX for inputting data from **(mxn)** sources and one **m-to-1** MUX for outputting data. All the multiplexers used in this method are kept always enabled. Following examples will explain the two modes of connections :

Example 2

Cascade two 4-to-1 MUX IC chips (IC 74153) to make an equivalent 8-to-1 MUX.

Solution

Method-I

The interconnections of the two 4-to-1 MUXs is shown in Fig. 8.10 (a). The select lines S_1 and S_0 of the MUX-1 are connected to the select lines S_1 and S_0 of MUX-2 respectively

i.e. the two MUXs have common select lines. The active low enable inputs of the MUXs are connected by an inverter as shown in the Fig. 8.10 (a) so that the MUX-1 remains enabled as long as the input C = 0 i.e. $EN_1 = 0$ and the output Y_1 of MUX-1 follows either D_0 or D_1 or D_2 or D_3 depending on whether BA = 00 or 01 or 10 or 11. Under this condition the MUX-2 remains in the disabled condition because EN_2=1 as C = 0 and hence $Y_2 = 0$. Hence, the output of the OR gate follows Y_1.

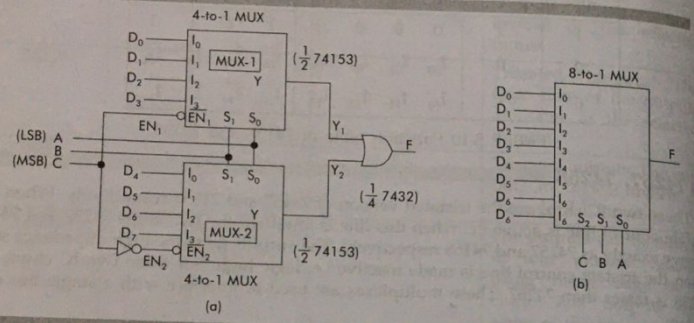


Fig.8.10 (a) Logic circuit for cascading two 4-to-1 MUXs, (b) Equivalent block diagram

Similarly, when C = 1 the MUX-2 is enabled as $EN_2 = 0$ and MUX-1 goes to the disabled condition as $EN_1 = 1$. For this situation Y_1 is always at logic-0 and the output Y_2 of MUX-2 follows either D_4 or D_5 or D_6 or D_7 for the input BA varying from 00 to 11. For this reason the output F of the OR gate follows Y_2. The functional table of the circuit of Fig. 8.10 (a) is shown in TABLE 8.11. Concentrating on the select inputs C, B and A and the output F of this table we may conclude that the two 4-to-1 MUXs cascaded as in Fig. 8.10 (a) is equivalent to one 8-to-1 MUX. The block diagram of the equivalent MUX is shown in Fig. 8.10 (b). For this cascaded multiplexer circuit the input C, B and A become the select inputs where C is MSB and A is the LSB.

Input					Output		
C	B	A	EN_1	EN_2	Y_1	Y_2	$F = Y_1 + Y_2$
0	0	0	0	1	D_0	0	D_0
0	0	1	0	1	D_1	0	D_1
0	1	0	0	1	D_2	0	D_2
0	1	1	0	1	D_3	0	D_3
1	0	0	1	0	0	D_4	D_4
1	0	1	1	0	0	D_5	D_5
1	1	0	1	0	0	D_6	D_6
1	1	1	1	0	0	D_7	D_7

TABLE 8.11 Function table of the circuit of Fig. 8.10

Method-II

The interconnection of the MUXs for this method is shown in Fig. 8.11 (a).

This logic circuit is constructed using only MUXs and all the MUXs are enabled permanently by connecting their enable inputs to ground. The select lines S and S_0 of MUX-

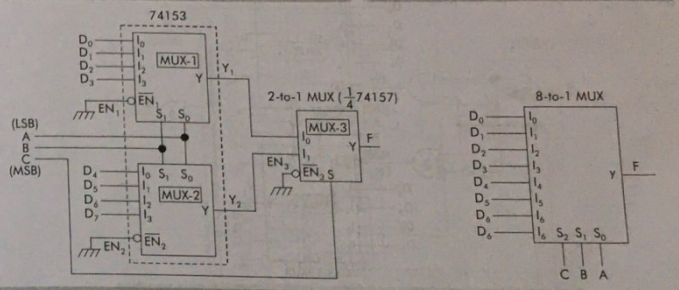


Fig 8.11 (a) Circuit for 8-to-1 MUX using two 4-to-1 MUXS (b) Block diagram of the equivalent to the circuit in Fig 8.11 (a)

1 and MUX-2 are common. So, for any input combination to B and A, data inputs of MUX-1and MUX-2 of same order are selected and the data are available at the outputs Y_1 and Y_2. For example, if BA = 10 then the input I_2 of both MUX-1 and MUX-2 will be selected for receiving data D_2 and D_6 but for C = 1 the select input S of MUX-3 is 1 and hence I_1 input of MUX-3 is selected. Hence, for C = 1 the Y_2 output of MUX-2 will be selected and the final output F will follow D_6. If C = 0 then the final output will be $F = Y_1 = D_2$ because for C = 0 the select input S of MUX3 is 0 and hence MUX3 selects only its I_0 input. The function table (TABLE 8.12) shows the final output F for different combinations of C, B and A. If we consider the input columns and the column for output F then we may say that the circuit of Fig. 8.11(a) behaves as an 8-to-1 MUX. The block diagram which is equivalent to this circuit is shown in the Fig. 8.11(b).

Inputs			Outputs		
C	B	A	Y_1	Y_2	F
0	0	0	D_0	D_4	D_0
0	0	1	D_1	D_5	D_1
0	1	0	D_2	D_6	D_2
0	1	1	D_3	D_7	D_3
1	0	0	D_0	D_4	D_4
1	0	1	D_1	D_5	D_5
1	1	0	D_2	D_6	D_6
1	1	1	D_3	D_7	D_7

TABLE 8.12 Functional table for the circuit in Fig. 8.11 (a).

From the final circuit of Fig. 8.10 (a) and 8.11 (it is seen that all the enable lines of the MUXs have been used and so there is no enal input left for controlling the equivalent 8-to-1 MUXs. To have the enable action in Fig. 8 (a) we may connect the enable inputs EN_1, EN_2 and EN_3 of the MUXs together inste of connecting them to the ground permanently. The common point can be used as an ac

low enable input. For the circuit of Fig. 8.10 (a) we may use two 2-input OR gate at the enable input of the MUX-1 and MUX-2 as shown in Fig. 8.12. In this circuit if the input EN is 0 then the logic state of the line C will reach the enable of input of MUX-1 and the

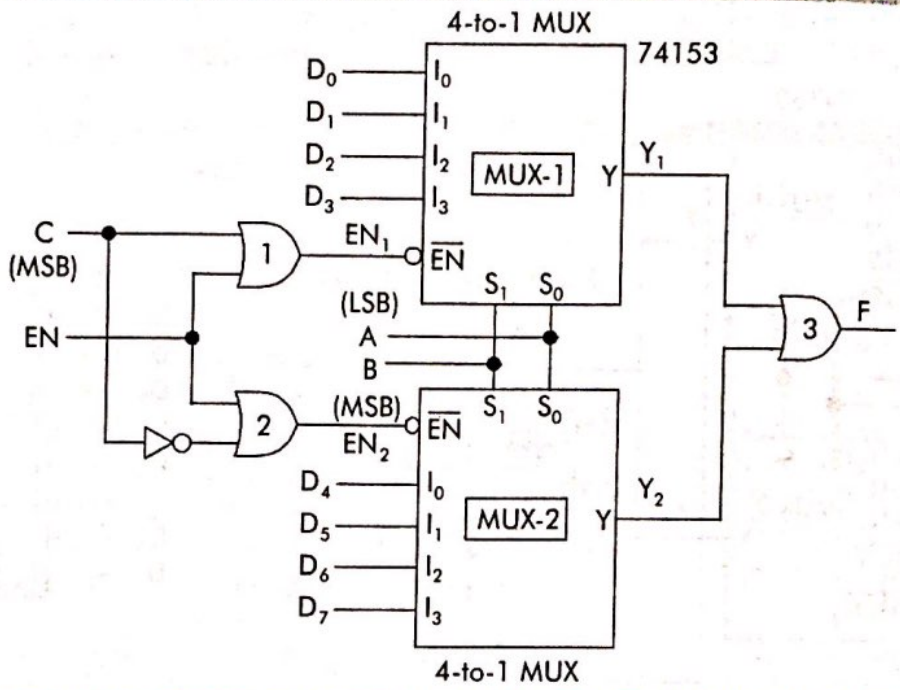


Fig. 8.12

inverse of C i.e. $\overline{C}$ will reach the enable input of MUX-2. For this reason for C = 0, the MUX = 1 will be enabled and the MUX-2 will be disabled. But for C = 1 and EN = 0 the MUX-2 will be enabled and MUX-1 will go into disabled condition. When EN = 1 the output of the OR gates 1 and 2 will always be 1 and the two MUXs will be in the disabled condition producing $Y_1 = Y_2 = 0$ and hence F = 0.

Example 3

(i) Design a 16-to-1 MUX using all 4-to-1 MUXs.

(ii) Design a 16-to-1 MUX using 8-to-1 and other suitable MUX.

(iii) Design a 16-to-1 MUX using all 8-to-1 MUXs.

(iv) Design a 16-to-1 MUX using two 8-to-1 MUXs and NAND gate.

8.4 Multiplexer as Logic Function Generator

It is already known to us that the output equation (Equn. 8.1 as an example) of a multiplexer can be expressed as cannonical SOP form i.e. as a sum of a fixed number of minterms. The number of minterms depends on the number of select inputs. For example, if there are N select inputs then there will be 2^N number of minterms. Each minterm contains (N + 1) number of literals. Out of these (N + 1) literals the N literals come from the select variables and one literal is due to the input data variable. For example, for a 4-to-1 MUX with select variables S_1 and S_0 and the input data variables D_0, D_1, D_2 and D_3 the output equation of this MUX is

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

The enable input is not considered in this equation.

To realise any function using a MUX we have to express the function in a standard canonical form and then we have to compare each minterm with the terms of the output equation of the MUX to get the idea for using the function variables as the select and the data variable of the given MUX. The absent minterms for the function can be realised by applying logic-0 to the data input lines. Following are some procedures which can be used to realise a logic function using multiplexers

(i) Algebraic Approach, (ii) Using Truth Table, (iii) Using K-Map, (iv) Using Implementation Table, (v) Using MUX Trees.

8.4.2 Using Truth Table

The circuit using multiplexers can also be realised starting from the truth table of the given function. In this method the rows of the truth table of n-variable function are partitioned in such a way that any m number of variables (m < n) in each partition have the same logic values After such partitioning, one should find out the functional dependance of the output F on the remaining variables (n – m) of each partition. These functional dependance from each partition gives the multiplexer inputs and the m variables whose logic values do not change in a partition become the select inputs of the multiplexer.

Let us consider the truth table [TABLE 8.17(a)] of 3 variables A, B, C,. This table is partitioned with respect to two variables, A and B such that in partition-I, AB = 00; in partition-II, AB = 01; in partition-III, AB = 10 and in partition-IV, AB = 11. The partitioning is shown in TABLE 8.17(b). It is seen from the partition-I that for AB = 00, the output F is same as C Therefore, we may write that F = C so long as A = B = 0. Similarly, when AB = 01 the output F = $\overline{C}$; when AB = 10, the output F = $\overline{C}$ and when AB = 11 the output F = C. Thus, the TABLE 8.17(b) may be described in a more compact form as shown in TABLE 8.17 (c).

Input			Output
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

TABLE 8.17 (a)

	Input			Output
	A	B	C	F
Partition-I	0	0	0	0
	0	0	1	1
Partition-II	0	1	0	1
	0	1	1	0
Partition-III	1	0	0	1
	1	0	1	0
Partition-IV	1	1	0	0
	1	1	1	1

TABLE 8.17 (b)

A	B	F
0	0	C(=I_0)
0	1	$\overline{C}$(=I_1)
1	0	$\overline{C}$(=I_2)
1	1	C(=I_3)

TABLE 8.17 (c)

The last column of this compact table describes the inputs to the 4-to-1 multiplexer with A and B as the select inputs (S_1, S_0). The inputs I_0, I_1, I_2 and I_3 have been indicated in this table. The connections of C, A and B to the 4-to-1 MUX is shown in Fig. 8.25(a).

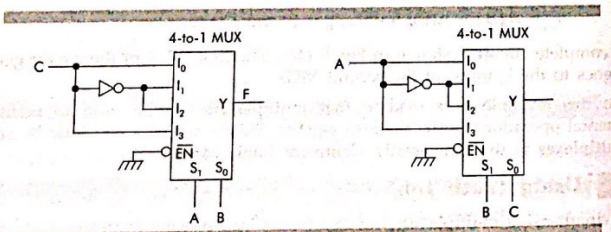


Fig. 8.25(a) Fig. 8.25(b)

If we decide to use B and C as the select variables (i.e. S_1 = B, S_0 = C) then we have to rearrange the given truth table so that the rows of the 1st, 2nd, 3rd and 4th partition, have the values BC = 00, 01, 10 and 11 respectively. The rearranged truth table is shown in TABLE 8.17(d). The compact table and the corresponding circuit with select inputs

S_1 = B and S_0 = C of a 4-to-1 MUX is shown in TABLE 8.17(e) and Fig. 8.25(b) respectively. Similarly, we might have used A and C as the select variables.

	Input			Output
	B	C	A	F
Partition-I	0	0	0	0
	0	0	1	1
Partition-II	0	1	0	1
	0	1	1	0
Partition-III	1	0	0	1
	1	0	1	0
Partition-IV	1	1	0	0
	1	1	1	1

TABLE 8.17 (d)

B	C	F
0	0	A(=I_0)
0	1	$\overline{A}$(=I_1)
1	0	$\overline{A}$(=I_2)
1	1	A(=I_3)

TABLE 8.17 (e)

Example 13

Implement circuit $F_1(A, B, C) = \sum m(1, 2, 4, 7)$ and $F_2(A, B, C) = \sum m(3, 5, 6, 7)$ using 4-to-1 multiplexers and inverter. Use truth table approach.

Solution The truth table of this example is shown in the TABLE 8.18(a). The table is partitioned with respect to the two variables A and B. The compact truth table is shown

	Input			Output	
	A	B	C	F_1	F_2
Partition-I	0	0	0	0	0
	0	0	1	1	0
Partition-II	0	1	0	1	0
	0	1	1	0	1
Partition-III	1	0	0	1	0
	1	0	1	0	1
Partition-IV	1	1	0	0	1
	1	1	1	1	1

TABLE 8.18 (a)

A	B	F_1	F_2
0	0	C(=I_0)	0(=I_0)
0	1	$\overline{C}$(=I_1)	C(=I_1)
1	0	$\overline{C}$(=I_2)	C(=I_2)
1	1	C(=I_3)	1(=I_3)

TABLE 8.18 (b)

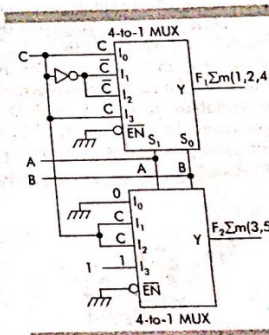


Fig. 8.26

in TABLE 8.18 (b). The two columns F_1 and F_2 of 8.18(b) are used as the inputs I_0, I_1, I_2 and I_3 to two s 4-to-1 MUXs with A and B as the common select for both. The implemented circuit is shown in Fi

Example 14

Implement the function F(A, B, C, D) = $\sum$m(0, 2, 4, 6, 8, 10, 14) using one 4-to-1 MUX and necessary logic gates.

Solution The truth table for this example is shown in TABLE 8.19 (a). Since we have to design the circuit using 4-to-1 MUX which needs two select inputs, therefore, this table is partitioned with respect to two variables A and B assuming that these two variables are to be used as the select inputs S_1 and S_0 respectively. This choice of the two select variables is arbitrary. Any two variables out of A, B, C and D could have been used for the select variables. For the partition-I, partition-II, partition-III, and partition-IV the values of the variables A and B are 00, 01, 10 and 11 respectively. The output F as a function of the remaining two variables C and D of this truth table is shown in the TABLE 8.19 (b). This is the compact truth table of TABLE 8.19 (a). The expression for F(C, D) for the 4 inputs to the 4-to-1multiplexer can easily be obtained using K-maps drawn separately for each partition. For this problem we may use four 2-variable K-maps, one for each partition where the map variables will be C and D. For simple cases the function F(C, D) can be obtained by inspection of the columns of C and D. For example, in this problem for the three partitions I, II and III we can directly find that the output function F is inverse of the column D so we may write F = $\overline{D}$ but for partition-IV we may use a K-map for F using C and D as the map variables, to get the functional dependance of F on C and D (though it is not very difficult to obtain the expression F(C, D) = C$\overline{D}$ by observation). The K-map for this partition is shown in FIg. 8.27(a) which gives F(C, D) = C$\overline{D}$. The compact table TABLE 8.19 (b) gives the four multiplexer inputs I_0, I_1, I_2 and I_3 for different select inputs AB = 00, 01, 10 and 11. The circuit for this example is shown in Fig. 8.27(b).

	Input				Output
	A	B	C	D	F
Partition-I	0	0	0	0	1
	0	0	0	1	0
	0	0	1	0	1
	0	0	1	1	0
Partition-II	0	1	0	0	1
	0	1	0	1	0
	0	1	1	0	1
	0	1	1	1	0
Partition-III	1	0	0	0	1
	1	0	0	1	0
	1	0	1	0	1
	1	0	1	1	0
Partition-IV	1	1	0	0	0
	1	1	0	1	0
	1	1	1	0	1
	1	1	1	1	0

TABLE 8.19 (a)

A	B	F (C,D)
0	0	$\overline{D}$ (=I_0)
0	1	$\overline{D}$ (=I_1)
1	0	$\overline{D}$ (=I_2)
1	1	C$\overline{D}$ (=I_3)

TABLE 8.19 (b)

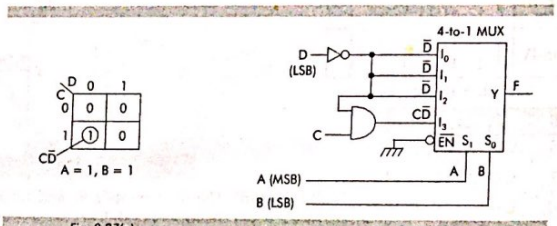


Fig. 8.27(a) Fig. 8.27(b)

8.4.3 Using Karnaugh Map

If the functional behavior of a circuit is described using a K-map then the implementation of the circuit using a multiplexer can easily be done starting from this map. In the n-variable K-map, if we decide to use m number of variables as the select variables then the groupings of the remaining (n – m) variables gives us the multiplexer inputs I_0, I_1, ... I_{2^m-1}. Let us consider the following illustrations to explain the procedure for obtaining the multiplexer inputs.

Let us consider a K-map of a 3-variable function F(A, B, C) as plotted in Fig. 8.28(a). Out of these 3 map variables A, B and C we may use any two variables as the select inputs of a 4-to-1 multiplexer. The easiest way is to choose B and C as the select variables since the columns of K-map in Fig 8.28(a) are already coded according to the four 2-bit binary select codes of a 4-to-1 MUX, i.e. BC = 00, 01, 10, 11.

The K-map is divided into four groups as shown in Fig. 8.28(b) where in each group the logic values of B and C do not change but that of the variable A changes. For the first column of the K-map, BC = 00 and the function F in this group is

F = 0 when A = 0 and BC = 00

F = 1 when A = 1 and BC = 00

Therefore, we may write F = A when BC = 00.

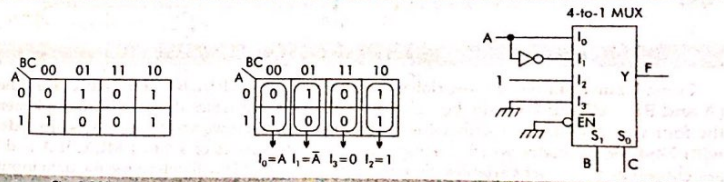


Fig. 8.28(a) Fig. 8.28(b) Fig. 8.28(c)

Therefore, it may be said that if B and C are used as the select inputs of a 4-to-1 MUX then the input I_0 of this MUX is to be connected to A to satisfy F = A with BC = 00.

001, 010, 011, 100, 101, 110 and 111 are shown in Fig. 8.28(o) and also the inputs I_0 to I_7 are indicated in general. For our specific example we may follow the grouping techniques of Fig. 8.28(o) to get $I_0 = 1$, $I_1 = 0$, $I_2 = 1$, $I_3 = D$, $I_4 = D$, $I_5 = \overline{D}$, $I_6 = 0$ and $I_7 = 1$ from the Fig. 8.28(p). The circuit with these eight inputs is shown in Fig. 8.28(q).

8.4.4 Using Implementation Table

This is a tabular method for finding the inputs of the multiplexer when the minterm numbers for the given function to be implemented are known to us. The implementation table has two rows and M number of columns where the number M is half of the maximum number of minterms possible with N–variable function. For N–variable function the maximum number of minterms is 2^N. So, the value of M is $\frac{2^N}{2} = 2^{N-1}$. If we check 2^N minterms of any Boolean function having N variables then we find that any variable appears in the uncomplemented form in exactly one half of the total number of minterms, while other half contains complemented form of that variable. In an implementation table the top row is labeled with the complement of a variable (i.e. say $\overline{A}$). of the function F(A, B, C, D) and the bottom row is labeled with the true form of the same variable (i.e. A) of the function. The top row lists all minterms which are associated with the variable in complemented form and the second row lists all minterms with that variable in the uncomplemented form. Usually, either the most significant or the least significant variable is used for labelling the rows. For example, Fig.8.29(a) and Fig.8.29(b) show the structure

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15

Fig. 8.29(a)

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{D}$	0	2	4	6	8	10	12	14
D	1	3	5	7	9	11	13	15

Fig. 8.29(b)

of the implementation table for a four variable function F (A, B, C, D). In Fig 8.29(a) the most significant variable A is used for row heading and the corresponding decimal equivalent number of the minterms are inserted in the boxes which are associated with $\overline{A}$ and A. Similarly, in Fig.8.29(b) the least significant variable D is used for heading the rows. In these two implementation tables there are 8 columns and each column is headed by the multiplexer inputs I_0, I_1, I_2, ..., I_7. As there are 8 inputs so we need an 8-to-1 MUX to realise the function F (A, B, C, D). In general, for an N-variable function we need a MUX having (N – 1) select inputs and 2^{N-1} data input lines.

Clearly, the entries in each column in Fig. 8.29(a) are determined by the bit combination of the variables A, B, C and D. For example, in the column I_3 there are two numbers, $3(\equiv \overline{A}\,\overline{B}CD)$ and $11(\equiv A\overline{B}CD)$ with row heading $\overline{A}$ and A respectively. The last three literals $\overline{B}$, C and D constitutes the select code of MUX input I_3. For the Fig.8.29(a), the numbers for the two boxes under any column I_i can be determined by writing the row-heading literal corresponding to that row in the most significant position followed by the

select code of I_i. Similarly, in case of Fig.8.29(b) the row-heading variable are written as the least significant variable (in this case they are $\overline{D}$ or D) after the select code of I_i to get the numbers under the column I_i. In this case the select codes are obtained using the variables A, B, C For example, in Fig. 8.29(b) the decimal number 12 and 13 are written in the top and bottom row under the column of I_6 because select code to select MUX input of I_6 using select variables A, B and C is $AB\overline{C}$ and the row heading literal for the top row in $\overline{D}$ and for the bottom row it is D. Here D is the least significant variable for the function F (A,B,C,D). Therefore, the top and bottom boxes under I_6 mean the decimal number of $(AB\overline{C})\overline{D}$ =12 and $(AB\overline{C})D$ = 13.

Fig. 8.29(a) and Fig. 8.29(b) show the boxes for the 16 minterms of a 4-variable function. The numbers 0 to 15 within the boxes are the decimal equivalent numbers of the 16 minterms of a 4-variable function. The variable A and D are the most significant variables in Fig. 8.29(a) and least significant variable in Fig. 8.29(b) respectively. Therefore, there is a difference in numbering of the boxes in the two tables. But all the 16 minterms may not be present in the given function. The minterms which are actually present in the function are circled in the implementation table.

Thus, the decimal number of the table indicates the decimal equivalent number of the minterm of the function.

Let us now consider an example of a function F (A, B, C, D) = $\sum$m(0, 1, 3, 6, 7, 8, 11, 12, 14). To implement this function with MUX taking the help of an implementation table we consider the format of the implementation table as shown in Fig. 8.29(a) in which the minterms which are actually present in the function F are marked with circle. The table after circling the present minterm is shown in Fig. 8.29(c). Let us now develop the following rules to determine the values of the MUX inputs I_0, ..., I_7 directly from the table simply from the observation of the circled minterms in a column.

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{A}$	⓪	①	2	③	4	5	⑥	⑦
A	⑧	9	10	⑪	⑫	13	⑭	15

Fig. 8.29(c)

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{A}$	⓪	①	2	③	4	5	⑥	⑦
A	⑧	9	10	⑪	⑫	13	⑭	15
MUX Input	1	$\overline{A}$	0	1	A	0	1	$\overline{A}$

Fig. 8.29(d)

The given function F (A, B, C, D) = $\sum$m(0, 1, 3, 6, 7, 8, 11, 12, 14) is written as

$$= \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}CD + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + AB\overline{C}\,\overline{D} + ABC\overline{D}.$$

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

$$= 1.(\overline{B}\,\overline{C}\,\overline{D}) + \overline{A}.(\overline{B}\,\overline{C}D) + 1.(\overline{B}CD) + A.(B\overline{C}\,\overline{D}) + 1.(BC\overline{D}) + \overline{A}.(BCD).$$

Comparing this with multiplexer input equation of an 8-to-1 MUX with select inputs S_2 = B, S_1 = C and S_0 = D we get

$I_0 = 1, I_1 = \overline{A}, I_2 = 0, I_3 = 1, I_4 = A, I_5 = 0, I_6 = 1, I_7 = \overline{A}$. From this result we may develop the following rules

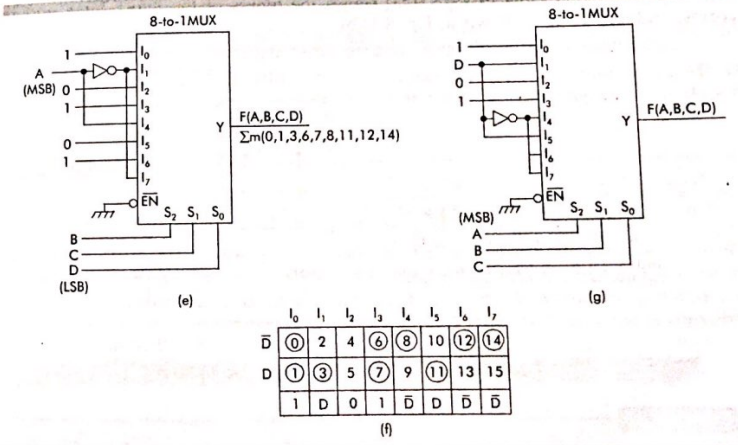
Rule 1 : If two circled minterms of a F(A, B, C, D) in an implementation table are present in a column I_i then the input I_i of the 8-to-1 MUX will be 1 if B, C and D are the select variables of the MUX. In this example, we have $I_0 = I_3 = I_6 = 1$.

Rule 2 : If one circled minterm is present in a column I_i and in a row with row heading $\overline{A}$ then the input I_i of the 8-to-1 MUX will be $\overline{A}$ if B, C and D are the select variables of the MUX. For example, in this illustration $I_1 = I_7 = \overline{A}$.

Rule 3 : If one circled minterm is present in a column I_i and in a row with rowheading A then the input I_i of the 8-to-1 MUX will be A if B, C and D are the select variables of the multiplexer. In this example, I_4 = A.

Rule 4 : If no circled minterm are present in a column I_i then the input I_i of 8-to-1 MUX will be equal to 0. In this example, $I_2 = I_5 = 0$.

The implemented circuit for the function F = $\sum$m(0, 1, 3, 6, 7, 8, 11, 12, 14) is shown in Fig. 8.29(e).



	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{D}$	⓪	2	4	⑥	⑧	10	⑫	⑭
D	①	③	5	⑦	9	⑪	13	15
	1	D	0	1	$\overline{D}$	D	$\overline{D}$	$\overline{D}$

(f)

Fig. 8.29

In the above illustration the most significant variable A of F (A, B, C, D) and its complement $\overline{A}$ have been used for heading the two rows of this table and B, C, and D have

been used as the select variables. If the least significant variable D of the function F(A, B, C, D) and its complement are used for labelling the rows and A, B, and C as the select variable, then the format of the table in Fig.8.29(b) will be the starting table. By circling the minterms present in the function and applying the 4 rules as stated we can obtain the multiplexer's inputs and the required circuit. These are shown in Fig 8.29(f) and Fig.8.29(g).

This implementation table has the advantage over the algebraic and K-map method as it gives us a more systemetic procedure to implement a Boolean function using multiplexer. In fact when the number of variables of the function increases then this implementation method becomes more useful.

But, it is not an efficient procedure because using this method it is possible to implement a function of N variables with MUX having 2^{N-1} inputs rather than with a MUX with samller number of inputs. Following are some examples for using Implementation table.

Example 15

Implement the function F(A, B, C, D) = $\sum$m(0, 2, 3, 4, 8, 10, 14) using MUX and other necessary logic gates. Use implementation table for the design.

Solution This 4-variable function needs one 8-to-1 MUX. Choose $\overline{A}$, and A for labelling the rows of the Implementation table and B, C and D as select variables of an 8-to-1 MUX. The implementation table in Fig.8.30(a) shows the inputs of the 8-to-1 MUX and the corresponding circuit is shown in Fig. 8.30(b).

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{A}$	⓪	1	②	③	④	5	6	7
A	⑧	9	⑩	11	12	13	⑭	15
MUX Inputs	1	0	1	$\overline{A}$	$\overline{A}$	0	A	0

Fig. 8.30(a)

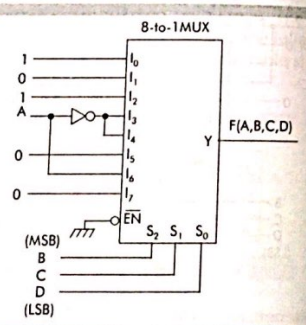


Fig. 8.30(b)

Example 16

Implement a circuit having three outputs F_1, F_2 and F_3 using all 4-to-1 MUXs. The three outputs are F_1(A, B, C) = $\sum$m(0, 1, 4, 5), F_2(A, B, C) = $\sum$m(1, 2, 3, 6) and F_3(A, B, C) = $\sum$m(0, 2, 4, 5, 6, 7). Use implementation table and assume that the complement of the most significant variable (A) is available.

Solution For this problem we need three separate Implementation tables which are shown in Fig.8.31(a). Variable A is the row heading variable. The variables, B and C are

	I_0	I_1	I_2	I_3
$\overline{A}$	⓪	①	2	3
A	④	⑤	6	7
MUX-1 Inputs	1	1	0	0

	I_0	I_1	I_2	I_3
$\overline{A}$	0	①	②	③
A	4	5	⑥	7
MUX-2 Inputs	0	$\overline{A}$	1	$\overline{A}$

	I_0	I_1	I_2	I_3
$\overline{A}$	⓪	1	②	3
A	④	⑤	⑥	⑦
MUX-3 Inputs	1	A	$\overline{1}$	A

Fig. 8.31 (a)

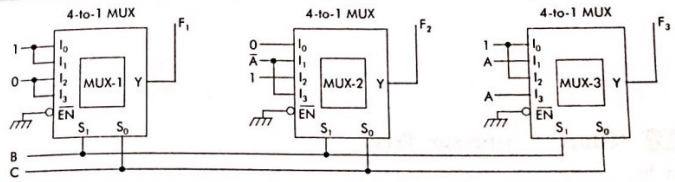


Fig. 8.31 (b)

used as the select variables for all the 4-to-1 multiplexers. The data inputs of the MUXs are determined from these implementation tables and the corresponding circuit is shown in Fig. 8.31(b).

Example 17

Implement the function F(A, B, C, D) = $\overline{A}BD + \overline{B}CD + \overline{C}\,\overline{D}$ using MUX with the help of Implementation table. Use the least significant variable D as the input data variable for the MUX.

Solution This is a 4-variable function. Out of these four variables we have to use the least significant variable D as the data input variable leaving the variables A, B and C as the select variables. Therefore, we need one 8-to-1 MUX. The given function is converted into a canonical SOP form as follows

$$F(A, B, C, D) = \overline{A}BD + \overline{B}CD + \overline{C}\,\overline{D}$$

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

$$= \overline{A}BCD + \overline{A}B\overline{C}D + A\overline{B}CD + \overline{A}\,\overline{B}CD + AB\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

$$= \sum m(0, 3, 4, 5, 7, 8, 11, 12)$$

These minterms are plotted in the implementation table [Fig. 8.32 (a)] to get the 8 i of the 8-to-1 MUX. The corresponding circuit is shown in Fig. 8.32 (b).

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{D}$	⓪	2	④	6	⑧	10	⑫	14
D	1	③	⑤	⑦	9	⑪	13	15
MUX inputs	$\overline{D}$	D	1	D	$\overline{D}$	D	$\overline{D}$	0

Fig. 8.32(a)

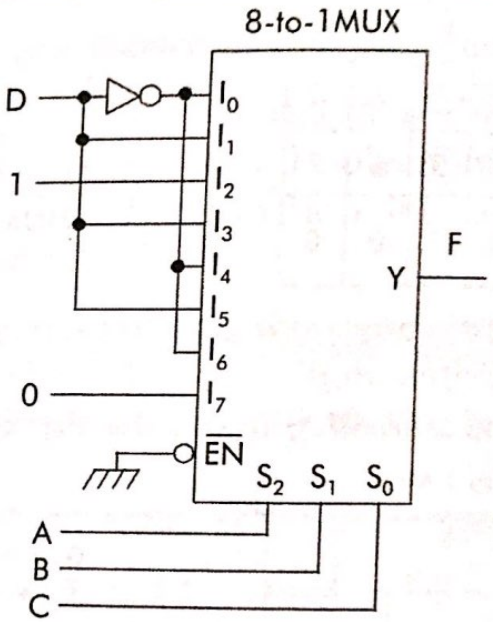


Fig. 8.32(b)

8.5 Decoder

8.5.1 Introduction

A decoder is a combinational circuit with n-Bit input and m number of mutually exclusive outputs. **The mutually exclusive output means that the decoder circuit makes only one output active (active low or active high) among the m possible outputs and this output is unique** and depends on the input code applied to the n-Bit input. We know that using n-bits we can have maximum 2^n codes, so the maximum number of mutually exclusive outputs may be $m = 2^n$. The function tables shown in TABLE 8.20(a) and TABLE 8.20(b) are the examples of a 2-input 4-output decoder circuit. In TABLE 8.20(a) one of four outputs (Y_0, Y_1, Y_2 and Y_3) is high for each 2-Bit input code ; all the other outputs are low for each 2-Bit code input. But in TABLE 8.20(b) one of four outputs is low ; all other three outputs are high for the same 2-Bit code input. These decoders having 2 inputs and 4 outputs are called 2-to-4 decoder. In general a decoder with n inputs and m outputs is called n-to-m decoder. The inputs B and A are also called the address inputs of the decoder.

Inputs		outputs			
B	A	Y_0	Y_1	Y_2	Y_3
0	0	**1**	0	0	0
0	1	0	**1**	0	0
1	0	0	0	**1**	0
1	1	0	0	0	**1**

(a)

Inputs		outputs			
B	A	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$
0	0	**0**	1	1	1
0	1	1	**0**	1	1
1	0	1	1	**0**	1
1	1	1	1	1	**0**

(b)

TABLE 8.20 Function table of (a) 2-to-4 decoder with active high output and (b) 2-to-4 decoder with active low output.

The block diagram representation of 2-to-4 decoder with active high and active low outputs are shown in Fig. 8.36(a) and Fig 8.36(b) respectively. Note the bubbles attached

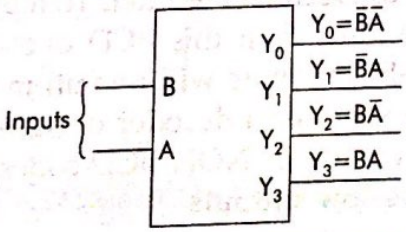


Fig. 8.36 (a) Block diagram representation of a 2-to-4 decoder with active high outputs.

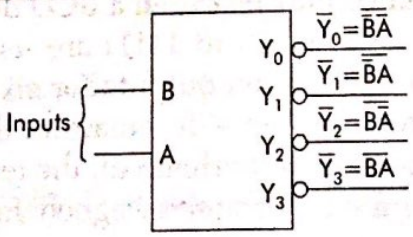


Fig. 8.36 (b) Block diagram representation of a 2-to-4 decoder with active low outputs.

to each output in the Fig 8.36(b). This indicates that all the outputs are active low type. Also, it may be noted from the function tables that any active high or low output of this 2-to-4 decoder circuit corresponds to a 2-Bit minterm code of the input B and A.

For example,

If BA = 00 = 2-Bit minterm code of m_0 at the input then only $Y_0 = 1$ (for active high output), $\overline{Y}_0 = 0$ (for active low output).

BA = 01 = 2-Bit minterm code of m_1 at the input then only $Y_1 = 1$ (for active high output), $\overline{Y}_1 = 0$ (for active low output).

BA = 10 = 2-Bit minterm code of m_2 at the input then only $Y_2 = 1$ (for active high output), $\overline{Y}_2 = 0$ (for active low output).

BA = 11 = 2-Bit minterm code of m_3 at the input then only $Y_3 = 1$ (for active high output), $\overline{Y}_3 = 0$ (for active low output).

Thus, by looking at the logic state and the position of the outputs we may at once understand the presence of the minterm code at the input. Hence, we may write the functional relationship of input and output as

$Y_i = m_i$ for a decoder with active high output,

and $\overline{Y}_i = \overline{m}_i$ for a decoder with active low output where $m_i = m(B, A)$.

As each output Y_i of a decoder represents a minterm m_i or the inverse of the minterm so the decoder circuit may be called a **Minterm Generator** or **Minterm Indicator.**

A decoder circuit with N inputs and 2^N outputs is called a **Binary Decoder** because this type of decoder generates all the minterms which are possible with the N bit binary inputs. The 2-to-4, 3-to-8 and 4-to-16 decoders are the examples of the binary decoders. But the decoders can be designed and are also commercially available for which the number of outputs is less than 2^N with N-Bit code as the input. For these type of decoders the number of outputs depends on the number of possible valid minterm codes at the input. For example, if BCD codes are used as the input then the maximum number of outputs will be ten to indicate ten valid input minterm codes of the BCD though the total number possible input codes are 16 with 4 input bits. This 4-Bit input would generate 16 possible outputs if all the combinations of 4-Bits are considered. This particular decoder is called as **4-to-10 decoder** or **may be called a BCD decoder or Decimal Decoder.** To a BCD decoder if six non-BCD codes (1010 to 1111) are used as inputs then this BCD decoder will not respond to them. All the ten outputs for six NON-BCD inputs will remain in the inactive state i.e. all the ten outputs will remain in the low state for a decoder designed for active high outputs and on the otherhand all the ten outputs for six NON-BCD codes will remain in the high state for a decoder designed for active low outputs.

Besides the N-Bit input, the commercial decoder IC chips are provided with one or more additional inputs called enable input. The enable inputs may be all active low or may be partly active low and partly active high. It may be mentioned here that these **enable inputs help to cascade many decoders to increase the input and output capacity or to use a decoder as a demultiplexer.**

8.5.2 Decoder Design

(a) 2-to-4 Decoder : The function table of 2-to-4 decoder is shown in TABLE 8.20. Two tables show the logic state of the decoder outputs as a function of the two inputs B and A. Using these tables we may write down the logic equations for the outputs as follows :

i) $Y_0 = \overline{B}\,\overline{A}$, $Y_1 = \overline{B}A$, $Y_2 = B\overline{A}$ and $Y_3 = BA$; for a 2-to -4 decoder with active high outputs. This follows from the TABLE 8.20(a).

ii) $Y_0 = \overline{\overline{B}\,\overline{A}}$, $Y_1 = \overline{\overline{B}A}$, $Y_2 = \overline{B\overline{A}}$ and $Y_3 = \overline{BA}$; for a 2-to-4 decoder with active low outputs. This follows from the TABLE 8.20(b).

If the enable input is included in the decoder circuit then the above equations are modified to

A) $Y_0 = \overline{B}\,\overline{A}.\overline{EN}$, $Y_1 = \overline{B}A.\overline{EN}$, $Y_2 = B\overline{A}.\overline{EN}$ and $Y_3 = BA.\overline{EN}$ for a 2- to-4 decoder with four active high outputs and active low enable input, $\overline{EN}$.

B) $\overline{Y}_0 = \overline{\overline{B}\,\overline{A}.\overline{EN}}$, $\overline{Y}_1 = \overline{\overline{B}A.\overline{EN}}$, $\overline{Y}_2 = \overline{B\overline{A}.\overline{EN}}$ and $\overline{Y}_3 = \overline{BA.\overline{EN}}$ for 2-to-4 decoder with four active low outputs and one active low enable input, $\overline{EN}$.

Using the above equations for the outputs we may easily draw the logic circuits for the decoders. These circuits are shown in Fig. 8.37. In these two decoder circuits when the

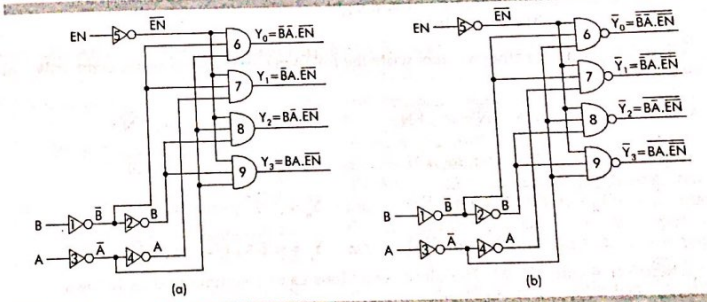


Fig. 8.37 Logic circuit of 2-to-4 decoder (a) with active high outputs (b) with active low outputs

enable input is made low i.e. active then and only then the four outputs of the decoder circuits follow the function table (Table 8.20) otherwise for logic high input to the enable input all the outputs will go to inactive state (all the outputs will be low for a decoder having active high outputs, [Fig 8.37 (a)]; all the outputs will be high for a decoder having active low outputs, [Fig. 8.37 (b)]). Two inverters in series for each decoder input B or A have been used to reduce loading to the external circuit driving the B and A inputs.

(b) 3-to-8 Decoder : The logic circuit for this decoder can be implemented using all discrete logic gates following the approach of the design of the 2-to-4 decoder. The function table of 3-to-8 decoder is shown in TABLE 8.21 with one active low enable input (EN), three active high address inputs (C,B,A) and eight active low outputs ($\overline{Y}_0$ through $\overline{Y}_7$). The block diagram representation of the 3-to-8 decoder is shown in Fig. 8.38.

Inputs				Outputs							
EN	C	B	A	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
0	0	0	0	**0**	1	1	1	1	1	1	1
0	0	0	1	1	**0**	1	1	1	1	1	1
0	0	1	0	1	1	**0**	1	1	1	1	1
0	0	1	1	1	1	1	**0**	1	1	1	1
0	1	0	0	1	1	1	1	**0**	1	1	1
0	1	0	1	1	1	1	1	1	**0**	1	1
0	1	1	0	1	1	1	1	1	1	**0**	1
0	1	1	1	1	1	1	1	1	1	1	**0**
1	x	x	x	1	1	1	1	1	1	1	1

TABLE 8.21 Function table of 3-to-8 decoder with active low enable input and eight active low outputs.

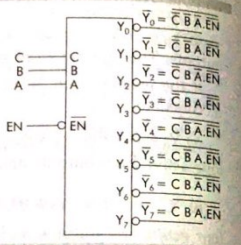


Fig. 8.38 Block diagram representation of 3-to-8 decoder

From the function table we may write the following logic equations for eight active low outputs.

$$\overline{Y}_0 = (\overline{C}\,\overline{B}\,\overline{A}).\overline{EN} = \overline{m_0.EN} \qquad \overline{Y}_4 = (C\overline{B}\,\overline{A}).\overline{EN} = \overline{m_4.EN}$$

$$\overline{Y}_1 = (\overline{C}\,\overline{B}A).\overline{EN} = \overline{m_1.EN} \qquad \overline{Y}_5 = (C\overline{B}A).\overline{EN} = \overline{m_5.EN}$$

$$\overline{Y}_2 = (\overline{C}B\overline{A}).\overline{EN} = \overline{m_2.EN} \qquad \overline{Y}_6 = (CB\overline{A}).\overline{EN} = \overline{m_6.EN}$$

$$\overline{Y}_3 = (\overline{C}BA).\overline{EN} = \overline{m_3.EN} \qquad \overline{Y}_7 = (CBA).\overline{EN} = \overline{m_7.EN}$$

where $m_i = m(C, B, A)$. The above equations can be summarised as follows :

When the 3-bit code of the minterm m_i is present at the input (C, B and A) and if the circuit is enabled (EN = 0) then we may write for the outputs as follows

$\overline{Y}_K = 0$ for K = i where i is the suffix of the input minterm code of m_i

$= 1$ for $K \neq i$ where i, K = 0, 1, ..., 7

and $\overline{Y}_K$ is always 1 (K = 0 to 7) for all possible 3-Bit minterm codes when the enable input is kept inactive i.e. the input EN in the above equation is not low.

The logic circuit based on the above logic equations is shown in Fig. 8.39. In this logic circuit six inverters (No.1 to 6) have been used instead of three to generate the true and

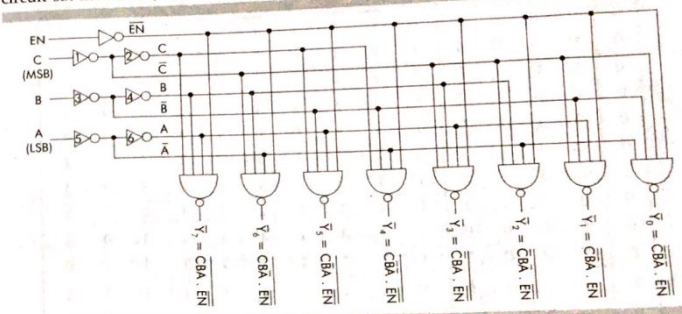


Fig. 8.39 Logic circuit of 3-to-8 decoder with active low outputs and active low enable input

complement form of the inputs address C, B, and A (i.e. C, $\overline{C}$, B, $\overline{B}$, A and $\overline{A}$) which are used as the NAND gate inputs. This has been done to reduce the effect of loading to the external circuit driving the decoder inputs C, B, and A. All the inverters in this circuit are buffer type inverters.

(c) 4-to-10 Decoder (BCD-to-Decimal decoder) : The function table of a 4-to-10 decoder with 10 active low outputs ($\overline{Y}_0$ through $\overline{Y}_9$), 4-Bit BCD input (D, C, B, A) and one active low enable input ($\overline{EN}$) is shown in TABLE 8.22 and the block diagram representation is shown in Fig. 8.40. This table shows that this decoder responds to the ten DCBA inputs (DCBA = 0000 to 1001) producing ten mutually exclusive active low outputs ($\overline{Y}_0$ through $\overline{Y}_9$) provided the enable input is active (EN = 0). But for EN = 1 all the ten outputs go to inactive state i.e., all the ten outputs become High irrespective of the inputs to D, C, B and A. One important point to note in TABLE 8.22 is that for non-BCD codes under EN = 0 the ten outputs are taken to be don't care types. This means that the circuit will not output a single 0 on $\overline{Y}_0$ through $\overline{Y}_9$ like ten valid BCD codes while the six non-BCD codes (DCBA = 1010, 1011, 1100, 1101, 1110 and 1111) are used as inputs

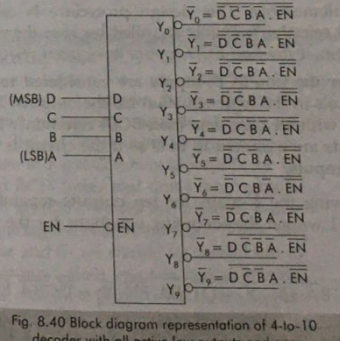


Fig. 8.40 Block diagram representation of 4-to-10 decoder with all active low outputs and one active low enable input

Decoder Inputs					Decoder Outputs									
EN	D	C	B	A	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	**0**	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	**0**	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	**0**	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	**0**	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	**0**	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	**0**	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1	**0**	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	**0**	1	1
0	1	0	0	0	1	1	1	1	1	1	1	1	**0**	1
0	1	0	0	1	1	1	1	1	1	1	1	1	1	**0**
1	ϕ	ϕ	ϕ	ϕ	1	1	1	1	1	1	1	1	1	1
0	non-BCD code				ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

TABLE 8.22 Function table of a 4-to-10 decoder with active low outputs and one active low enable input

To find out the logic equations for the outputs, the k-map has to be drawn separately for each output. While drawing these k-maps we may consider the don't care outputs in the following two ways.

(a) All the don't care outputs are considered to be 1. This means that for any non-BCD input all the ten outputs become 1 simultaneously. This design procedure is called **"Decoder design with false-data (non-BCD) rejection".** This is so called because the non-BCD inputs have no effects on the outputs as if the non-BCD inputs have been rejected.

(b) Some selected don't care outputs for the non-BCD inputs are considered to be 0. This is done to reduce the number of literals in the logic equation for the outputs. This design procedure is called **"Decoder design without false-data (non-BCD) rejection".** This is so called because, some don't care outputs are assumed to be 0 as if the circuit is not rejecting completely the effect of non-BCD inputs.

Some commercial decoder ICs have more than one enable input. Out of the multiple enable

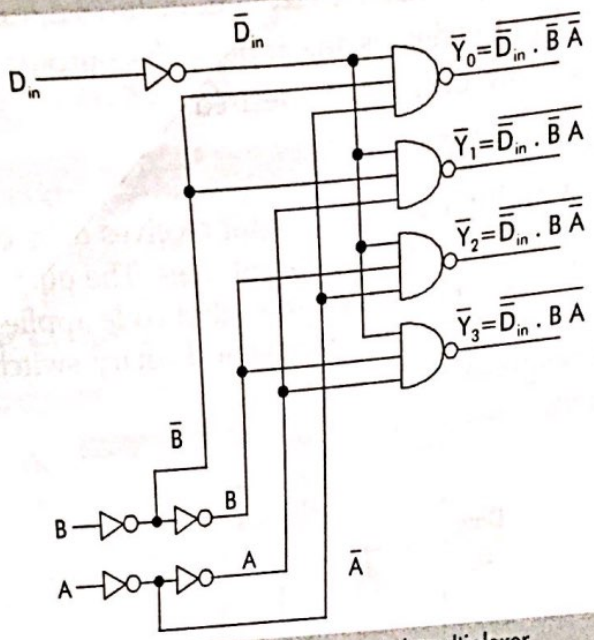


Fig. 8.45 Logic circuit of 1-to-4 demultiplexer

Data Input	Select code		Outputs			
D_{in}	B	A	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$
0	0	0	0	1	1	1
1	0	0	1	1	1	1
0	0	1	1	0	1	1
1	0	1	1	1	1	1
0	1	0	1	1	0	1
1	1	0	1	1	1	1
0	1	1	1	1	1	0
1	1	1	1	1	1	1

TABLE 8.24 Function table of 1-to-4 demultiplexer

inputs a few may be active high or active low. Active high enable input of a decoder can be used as the data input but the routed data at the output will be complemented.

8.5.4 Commercially Available Decoder/Demultiplexers

Following are the descriptions and discussions regarding the input/output features of a few commercially available decoder / demultiplexer.

2-to-4 Decoder / Demultiplexers

Chip No.	No of Pins	Technology	Functional Description
74139	16	LS,S,AS,HC	Contains 2 fully independent 2-to-4 Decoder/ Demultiplexers. Each unit has one seperate active low enable input, four active low outputs and two active high address inputs.
74155	16	Std. TTL, LS	Contains 2 units of 2-to-4 decoder/demultiplexers. The two units have two common address inputs. One unit has one active low and one active high enable input. The two enable inputs for the other unit are both active low. The four outputs of each unit are active-low type.
74156	16	Std.TTL, LS	Same as 74155 but the outputs are open collector type.
74239	16	HC	Same as 74139 but the outputs are active high type.

TABLE 8.25 Brief description of commercially available 2-to-4 Decoder/Demultiplexer IC chips.

of elements of inputs. For n elements of information to be uniquely encoded, the output code of width m must satisfy the following relation $2^m \geq n$. For example, to encode 10 decimal digits distinctly we need a group of 4 binary bits. Here m = 4 and n = 10. The distinct codes may be arbitrarily assigned to the inputs. In case of simple encoder the drawback is that this circuit cannot encode properly if more than one input are active simultaneously. To prevent this odd situation a new type of encoder circuit is designed which is called "**Priority encoder.**" A Priority encoder responds only to one input when more than one input are simultaneously active, in accordance with some priority. The most common priority system is based on the relative magnitude of the inputs. For example, a decimal-to-BCD priority encoder will encode the input for 7 if two inputs 5 and 7 are used simultaneously to this type of encoder. This happens because 7 has the higher priority than 5.

Example 34

Design an encoder circuit which has 4 active high inputs I_3, I_2, I_1 and I_0. This circuit needs to produce 6-bit distinct code for each input as shown in the TABLE-8.39. The restriction regarding the inputs is that only one input may be active at a time.

Solution The block diagram for the required circuit is shown Fig. 8.71 (a). In this diagram there are 4 inputs. For any single active high input the circuit outputs is a 6-bit code on Y_0 through Y_5 output lines.

Input				Output					
I_3	I_2	I_1	I_0	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
1	0	0	0	1	0	1	1	0	1
0	1	0	0	1	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	0	1	0	1	1	0	1	1

TABLE 8.39

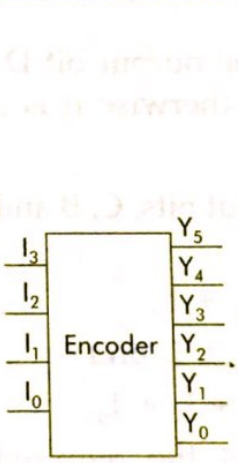


Fig. 8.71(a)

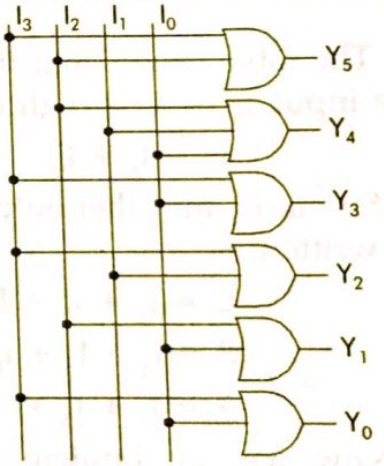


Fig. 8.71(b) Encoder Circuit for the example 34.

This encoder can be designed by looking at the output columns for Is in TABLE 8.39. We may write $Y_5 = I_3 + I_2$, $Y_4 = I_2 + I_1 + I_0$, $Y_3 = I_3 + I_0$, $Y_2 = I_3 + I_1$, $Y_1 = I_2 + I_0$ and $Y_0 = I_3 + I_0$. The logic circuit based on these 6 equations can be drawn as shown in Fig. 8.71 (b).

8.6.1 Decimal-to-BCD Encoder

The logic symbol of this encoder is shown in Fig. 8.72.

In this encoder there are 10 inputs, I_0, I_1, ..., I_9 each representing an input for a decimal digit. The output of this encoder has 4 parallel outputs (D, C, B and A). Each 4-Bit output corresponds to BCD code of the input line. For example, when only I_5 input is active out-

of ten inputs then the BCD output of this circuit will give BCD code of 5 i.e. 0101. Similarly for input I_8 the output will be 1000. We assume all inputs are active high. The input/output relationship of this encoder is shown in TABLE 8.40.

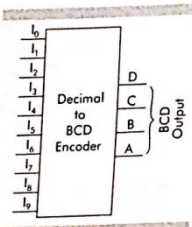


Fig. 8.72 Logic Symbol for a Decimal-to-BCD encoder

Inputs										BCD output			
I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	D	C	B	A
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

TABLE 8.40 Function table of the Decimal-to-BCD encoder

The table shows that the output bit D of the output BCD code becomes 1 only when the input I_8 or I_9 is high otherwise it is always 0. So, we may write.

$$D = I_8 + I_9$$

Similarly, the other output bits, C, B and A can be written as

$$C = I_4 + I_5 + I_6 + I_7,$$
$$B = I_2 + I_3 + I_6 + I_7 \text{ and}$$
$$A = I_1 + I_3 + I_5 + I_7 + I_9.$$

Now we can implement the required logic circuit for encoding each decimal digit to a BCD code by using the above logic expressions. We need to OR the appropriate inputs to form BCD output. The logic circuit for decimal to BCD is shown in Fig. 8.73. The inputs to all the OR gate are normally low but when a HIGH input appears then the outputs of the OR gates become High depending on the input For example if I_7 is HIGH the outputs are D = 0, C = 1, B = 1and A=1. The I_0 input is not needed because the BCD output bits are all 0 which is same when no inputs are used.

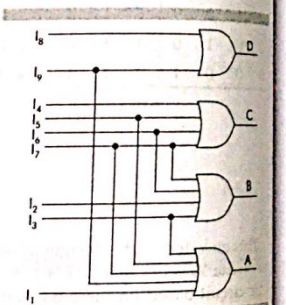


Fig. 8.73 Logic circuit of Decimal-to-BCD encoder.

8.6.2 Octal-to-Binary Encoder

Using the same approach followed in the case of Decimal-to-BCD encoder we can design an Octal-to-Binary encoder. In the octal encoder there are 8 inputs. To encode these 8 inputs in an unique way we need at least 3-bit output. In this design we take 3-bit output. The inputs and the outputs of an encoder may be either active high or low. For this particular case we consider all active low inputs and three active high outputs. Block diagram representation of such an encoder is shown in Fig. 8.74(a). The TABLE 8.41 describes the function of such an encoder. From this table it is seen that when no inputs are used (first row of TABLE 8.41) or when the input line 7 is made active then the output code becomes CBA = 111. But for the inputs other than the first row of this table we get the octal code at the output corresponding to the input used. Also, it may be noted that the inputs are active-low one at a time.

I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	C	B	A
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	1	0	0	1
1	1	1	1	1	0	1	1	0	1	0
1	1	1	1	0	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1

TABLE 8.41

From this table we may write the following logic equations for the outputs considering 0s of the inputs.

$$\overline{A} = \bar{I}_0 + \bar{I}_2 + \bar{I}_4 + \bar{I}_6$$

$$\text{or, } A = \overline{\bar{I}_0 + \bar{I}_2 + \bar{I}_4 + \bar{I}_6}$$
$$= I_0 \cdot I_2 \cdot I_4 \cdot I_6$$

$$\text{or, } \overline{B} = \bar{I}_0 + \bar{I}_1 + \bar{I}_4 + \bar{I}_5$$

$$\text{or, } B = \overline{\bar{I}_0 + \bar{I}_1 + \bar{I}_4 + \bar{I}_5}$$
$$= I_0 \cdot I_1 \cdot I_4 \cdot I_5$$

$$\text{or, } \overline{C} = \bar{I}_0 + \bar{I}_1 + \bar{I}_2 + \bar{I}_3$$

$$\text{or, } C = \overline{\bar{I}_0 + \bar{I}_1 + \bar{I}_2 + \bar{I}_3}$$
$$= I_0 \cdot I_1 \cdot I_2 \cdot I_3$$

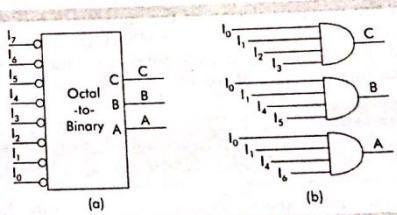


Fig. 8.74

The logic circuit for the Octal-to-Binary encoder is shown in Fig.8.74(b). In this case the input I_7 is not used.

Example 35

Design a Decimal-to-XS3 encoder with active high inputs and outputs.

Solution The function table of such an encoder is shown in TABLE 8.42. From this table we may develop the following logic equations describing the inputs and outputs

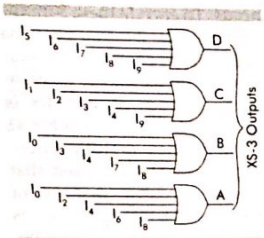


Fig. 8.75 Logic circuit for the Decimal-to-XS-3 encoder

I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	D	C	B	A
0	0	0	0	0	0	0	0	0	1	0	0	1	1
0	0	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0	0	0	1	0	1	0
0	1	0	0	0	0	0	0	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

TABLE 8.42

$A = I_0 + I_2 + I_4 + I_6 + I_8$

$B = I_0 + I_3 + I_4 + I_7 + I_8$

$C = I_1 + I_2 + I_3 + I_4 + I_9$

$D = I_5 + I_6 + I_7 + I_8 + I_9$

The logic circuit based on these equations are shown in Fig. 8.75.

8.6.3 Priority Encoder

In the previous design of the encoders it has always been mentioned that the input should be active (may be either HIGH or LOW) one at a time otherwise the code which will be produced may or may not correspond to the input. This statement may be verified considering any one of the encoders designed so far.

Example 36

Consider a Decimal-to-BCD encoder (not priority encoder) with all active HIGH inputs. What will be the output code if the input lines I_5 and I_6 are used simultaneously?

Solution From the TABLE 8.40 we may write the following :

For I_5 the code is DCBA = 0101

For I_6 the code is DCBA = 0110, if they are used separately. But these inputs are used simultaneously then the code at the output will be DCBA = 0111 which is a code of input I_7 and not a code for either I_5 or I_6.

From this example it is noted that a faulty code is generated at the output if more than one input are active at the same time. A modified version of this type of encoder circuit avoiding this drawback is the **Priority Encoder** that ensures that when two or more inputs are used, the output code will correspond to the highest numbered input. For example, in Example-36 the output code will be for I_6 only, though the two inputs I_5 and I_6 are used, in case of the priority encoder because the input I_6 has higher priority over I_5.

Example 37

Design a four input priority encoder such that when two inputs, I_i and I_j are high simultaneously, I_i has priority over I_j when $i > j$. The encoder should produce the following 2-bit code for the four inputs I_0, I_1, I_2 and I_3 :

$I_0 \rightarrow 00$, $I_1 \rightarrow 01$, $I_2 \rightarrow 10$ and $I_3 \rightarrow 11$.

Solution From the statement of the problem we may develop the function table (TABLE 8.43) for this encoder. This table shows that Y_1 is true when $I_2 = 1$, $I_3 = 0$, $I_1 = I_0 = \phi$ or when $I_3 = 1$, $I_2 = I_1 = I_0 = \phi$. The logic equation of Y_1 as a function of I_3, I_2, I_1 and I_0 can be obtained using the K-map as shown in Fig. 8.76(a). Similarly, Y_0 is true when $I_1 = 1$, $I_2 = I_3 = 0$, $I_0 = \phi$ or when $I_3 = 1$, $I_2 = I_1 = I_0 = \phi$. The output Y_0 can be expressed in terms of I_3, I_2, I_1 and I_0 from the K-map [Fig 8.76(b)]. From these two K-maps we get $Y_1 = I_3 + I_2$, $Y_0 = I_3 + \bar{I}_2 I_1$.

I_0	I_1	I_2	I_3	Y_1	Y_0
1	0	0	0	0	0
ϕ	1	0	0	0	1
ϕ	ϕ	1	0	1	0
ϕ	ϕ	ϕ	1	1	1

TABLE 8.43

The logic circuit for this encoder is shown in Fig. 8.76(c).

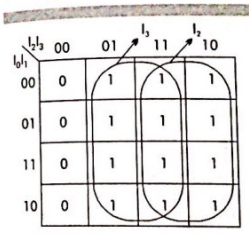


Fig. 8.76 (a) K-map for Y_1

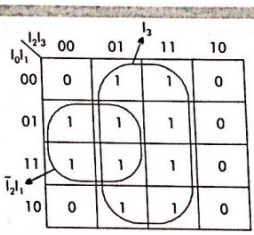


Fig. 8.76 (b) K-map for Y_0

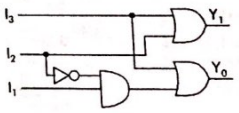


Fig. 8.76 (c) Logic circuit for the Example 37

Example 38

(a) Develope a truth table for a priority encoder with 3 active high inputs (A, B and C) such that the input C with least priority produces an output code 01, input B with next higher priority produces a code 10 and the input A with highest priority input produces a code 11. This circuit should have another active low output called 'INACTIVE' output which indicates that no inputs have been used i.e., A = B = C = 0.

(b) Implement the encoder circuit with a 3-to-8 decoder (74138) and other necessary logic gates.

(c) Implement the same encoder circuit using a 4-to-1 MUX and other necessary logic gates.

(d) Compare the economy of the two designs.