4. Combinational circuits

Combinational and sequential circuits

Digital circuit are divided into two broad categories (1) combinational circuits and (2) sequential circuits.

In combinational circuits, the outputs at any instant of time depend upon the inputs present at that instant of time. The output of a combinational circuit does not depend upon any past inputs or outputs ie; the outputs of the combinatinal circuits are not fed back to the input of the circuit. This means that there is no memory in these circuit. Moreover, in a combinational circuit, for a change in the input, the output appears immediately except for the propagation delay through the circuit. The block diagram of a combinational circuit with *m* inputs and *n* outputs is shown in the fig use.

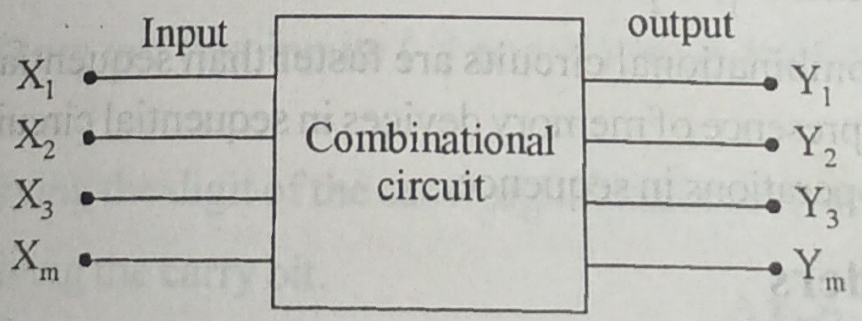

Fig. 102. Block diagram of combinational circuit

A sequential circuit is a logic circuit whose outputs at any instant of time depends on the present inputs as well as past inputs/outputs. i.e; there are elements (units) used to store past information. These elements are known as memory. The sequential logic system may have combinational logic sub-systems.

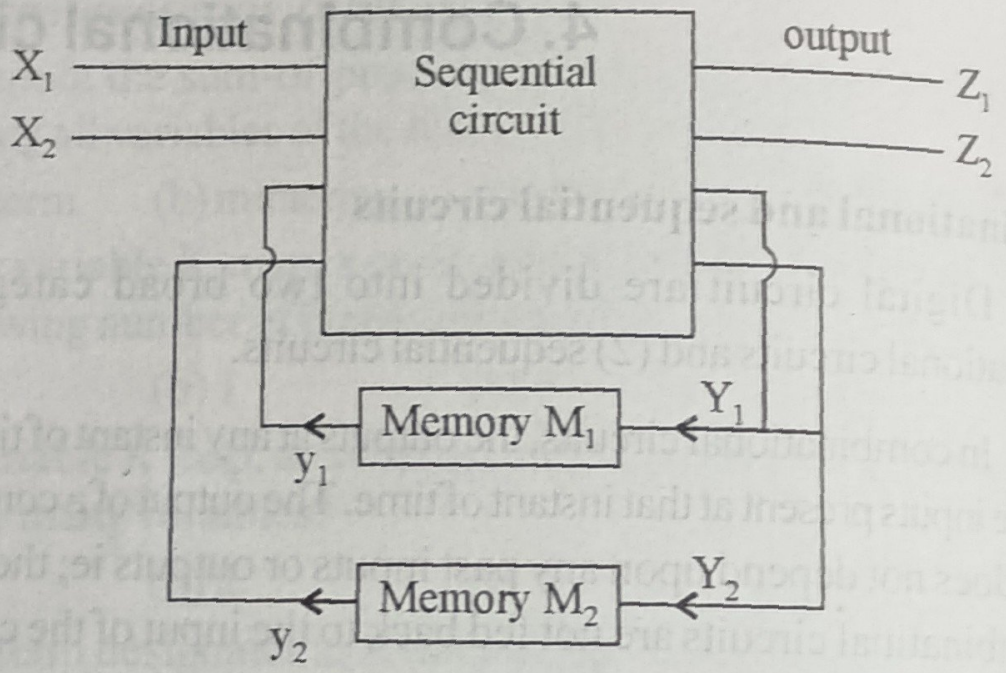

Fig. 103. *Block diagram of Sequential circuit.*

The circuit has *m* inputs (like x_1, x_2) and *n* number of outputs (like z_1, z_2). The signal value at some outputs of the combinational circuit are fed back to some inputs of the combinational circuits through the memory elements M_1, M_2 etc.

The combinational circuits are faster than sequential circuits because of the presence of memory devices in sequential circuits; which perform their operations in sequence.

Binary Adders

A digital computer is a powerful mathematical tool. A basic computer needs to perform only a few operations such as fetching a word from storage,add,subtract, test and jump or put a word into storage. The arithmetic operations done by a computer are simply addition and subtraction. More complex operations such as multiplication, division, integration or taking a log or a power are done by repeated addition and

Multiplexer

Multiplex means 'many into one'

A digital multiplexer is a combinational circuit that selects *binary* information from one of several input lines and directs it to *one* output line for transmission to a common destination. The selection of a particular input line is controlled by a set of selection lines (data select).

The multiplexer (MUX) thus has several data input lines and a single output line. It also has data-select input that helps to choose any one of the inputs to be switched (connected) to the output line. The logic symbol for a four - input (4 to 1) multiplexer is shown in the figure 116.

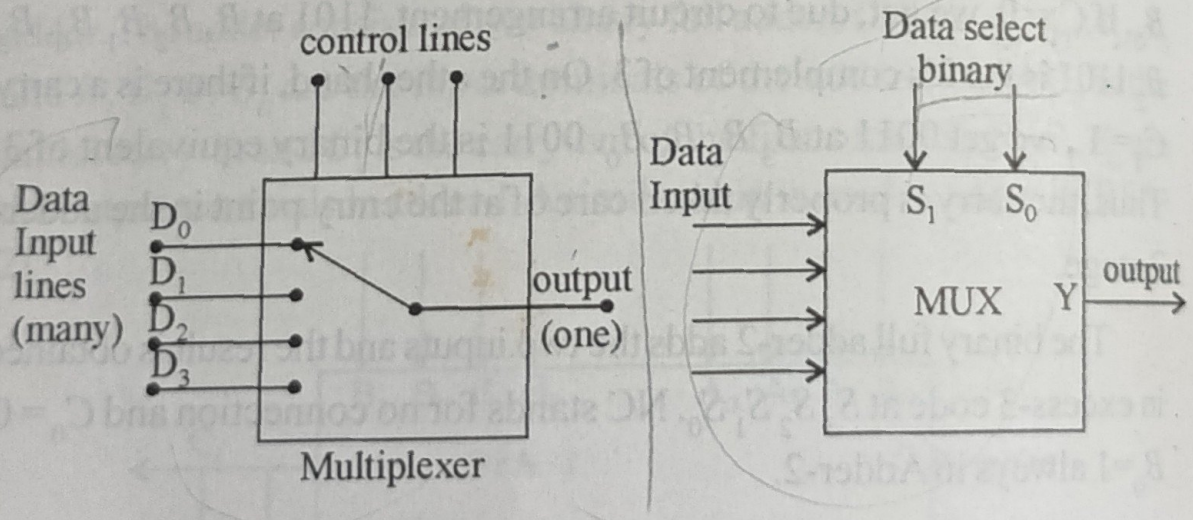

Fig. 116 *Multiplexer*

Data-select		Connection to
S_1	S_2	output Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

A two-bit binary number can be used as data - select. There are four possible combinations for two data-select S_1, S_0. The data input corresponding to this binary number is linked to the output line.

For example, if $S_1 = 0$, $S_0 = 0$ (binary 00) is applied to the data select lines, the data on the input D_0 will appear on the output line Y. If $S_1 = 0$ and $S_0 = 1$ (binary 01) is applied, the data on D_1 will appear on the output line Y. Similarly $S_1 = 1$ and, $S_0 = 0$ (binary 10) and $S_1 = 1$ and $S_0 = 1$ (binary 11) will connect respectively the D_2 and D_3 inputs to the out-put line. The summary of this action is given in the table above.

Thus, it is found that the data-output Y is equal to the data-input D_0 if and only if $S_1 = 0$ and $S_0 = 0$

$$\therefore Y = D_0 \overline{S}_1 \overline{S}_0$$

The data output is equal to D_1 if and only if

$S_1 = 0$ and $S_0 = 1$ $\quad\quad \therefore Y = D_1 \overline{S}_1 S_0$

The data output is equal to D_2 if and only if

$S_1 = 1$ and $S_0 = 0$ $\quad\quad \therefore Y = D_2 S_1 \overline{S}_0$

The data output is equal to D_3, if and only if

$S_1 = 1$ and $S_0 = 1$ $\quad\quad \therefore Y = D_3 S_1 S_0$

The logic expression for the data-output is obtained by ORing the above possible combinations.

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

The above logic equation can be implemented by using two inverters, four 3 input AND gates and one four input OR gate as shown in figure117. The circuit is also known as data selector, since data can be selected from any of the input lines.

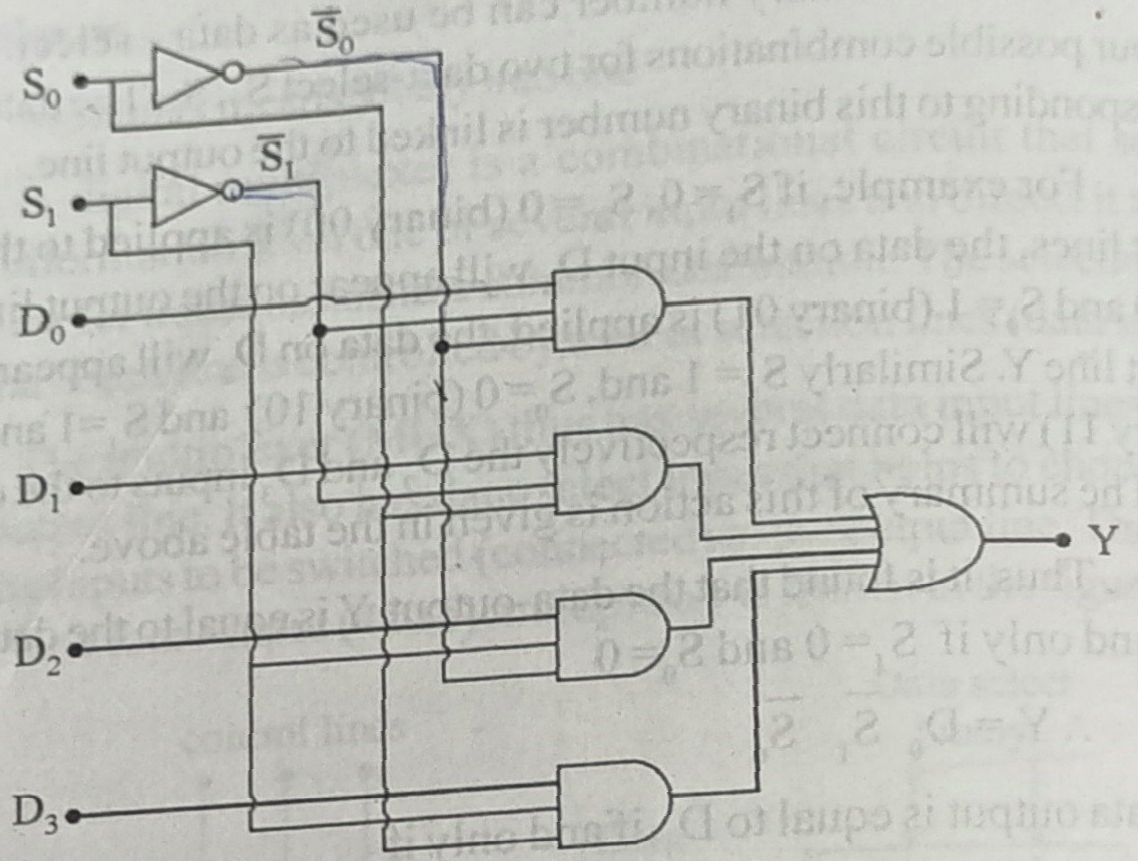

Fig. 117 *A 4 to 1 multiplexer.*

The circuit can be expanded to include more input lines. With n data-select line, 2^n input lines can be connected to the single output in a multiplexer.

Demultiplexer

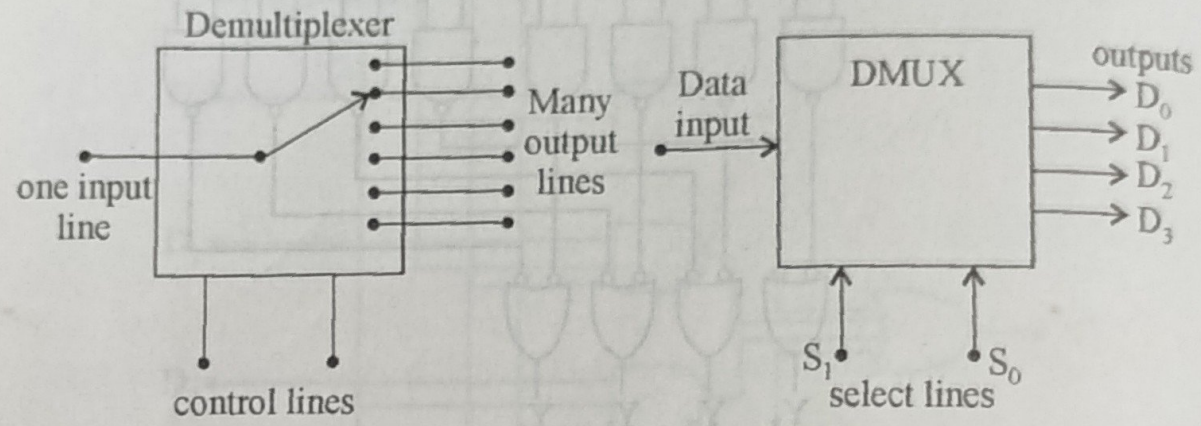

Fig. 118 *Demultiplexer*

A demultiplexer (DMUX) does the reverse action of a multiplexer. It takes data from one line (single input) and distributes it to a given number of output lines. Thus, The DMUX has one input and many outputs. The demultiplexer is a circuit that is controlled to switch a single input through it to one of several outputs.The logic symbol for one input - 4 output DMUX is shown in figure 118.

Figure 119 shows a 1-line to 4-line demultiplexer circuit. The input data line goes to all the AND gates. The select lines will enable only one gate at a time. When a particular AND gate is enabled the corresponding output will be alive.

For example, with $E=1$ and $S_1=0, S_0=0$ (binary 00), the output line D_0 will be enabled i.e., output D_0 will get connected to input E. For the select line $S_1=0, S_0=1$, the input line E will be linked to the output line D_1. In a similar way, D_2 and D_3 lines are excited for $S_1=1, S_0=0$ and $S_1=1$ and $S_0=1$ respectively. A summary of this functioning is given in the table. The logic expressions for the separate outputs are as below.

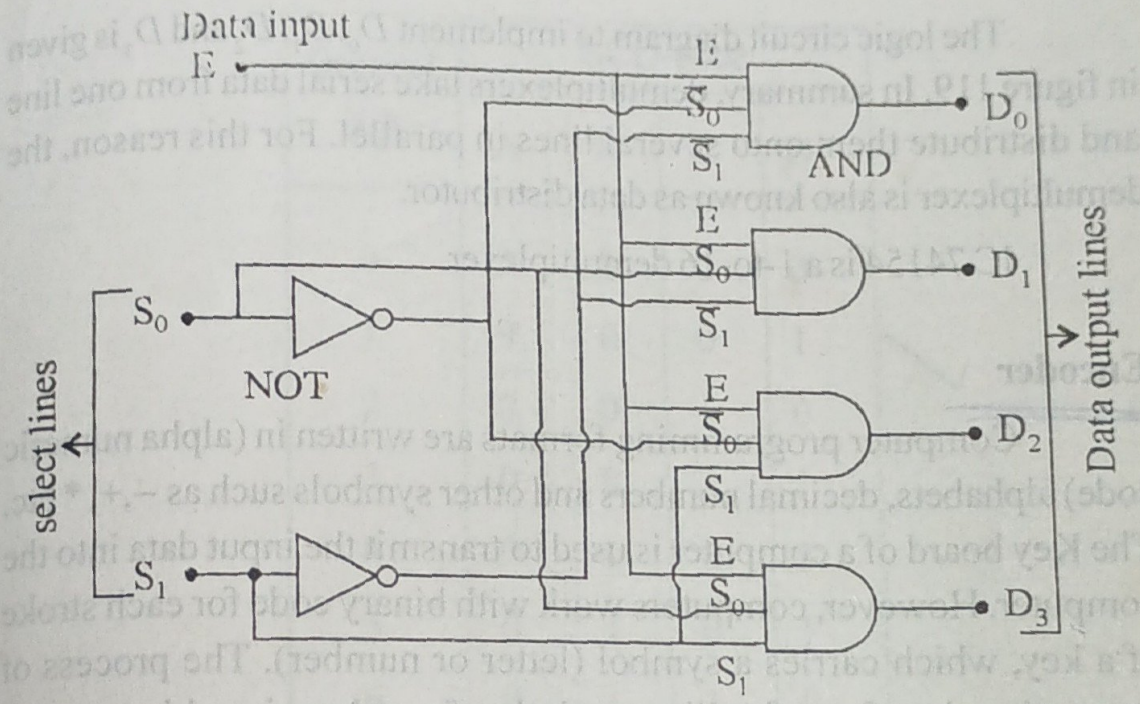

Fig. 119 *One to four line demultiplexer.*

The outputs are

$$D_0 = E\,\bar{S}_1\bar{S}_0 \qquad D_1 = E\,\bar{S}_1 S_0$$

$$D_2 = E\,S_1\bar{S}_0 \qquad D_3 = E\,S_1 S_0$$

Data Input	Data-select		Connection to output Y
	S_1	S_2	
E	0	0	$D_0 = E\,\bar{S}_1\bar{S}_0$
E	0	1	$D_1 = E\,\bar{S}_1 S_0$
E	1	0	$D_2 = E\,S_1\bar{S}_0$
E	1	1	$D_3 = E\,S_1 S_0$

The logic circuit diagram to implement D_0, D_1, D_2 and D_3 is given in figure 119. In summary, demultiplexers take serial data from one line and distribute them onto several lines in parallel. For this reason, the demultiplexer is also known as data distributor.

IC 74154 is a 1-to-16 demultiplexer.

Encoder

Computer programming formats are written in (alpha numeric code) alphabets, decimal numbers and other symbols such as –,+, * etc. The Key board of a computer is used to transmit the input data into the computer. However, computers work with binary code for each stroke of a key, which carries a symbol (letter or number). The process of converting data from familiar symbols of numbers into binary code (or codes format) is necessary and it is called encoding. Thus the key board of computer is equipped with an encoder.

Decimal-to-BCD Encoder

The BCD code for the decimal numbers from 0 to 9 is listed in the Table. The most significant digit is A. The decimal terms for which A appears as 1 (high) are 8 or 9.

$$A = 8 + 9$$

The bit B is 1 (high) for 4 or 5 or 6 or 7

$$\therefore B = 4 + 5 + 6 + 7$$

Similarly, $C = 2 + 3 + 6 + 7$ and $D = 1 + 3 + 5 + 7 + 9$. Since $D = 1+3+5+7+9$, the lines 1,3,5,7,9 are to be connected as inputs to an OR gate to get D output.

Decimal digit	BCD-code			
	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

We can encode each of the decimal digits 0 to 9 to the corresponding binary equivalent using four OR gates and a system of push-button switches like those of a pocket calculator.

When a particular button is pressed, the corresponding binary number is available at the output terminals (the output of the OR gates). The output is ABCD. For example, when the button 3 is pressed, the OR gates C and D have high inputs.

Therefore, the output is ABCD = 0011.

When the button 5 is pressed, the output becomes ABCD = 0101. Thus the BCD encoder produces binary codes for the given decimal inputs. The codes produced are stored in the register in a computer for further processing.

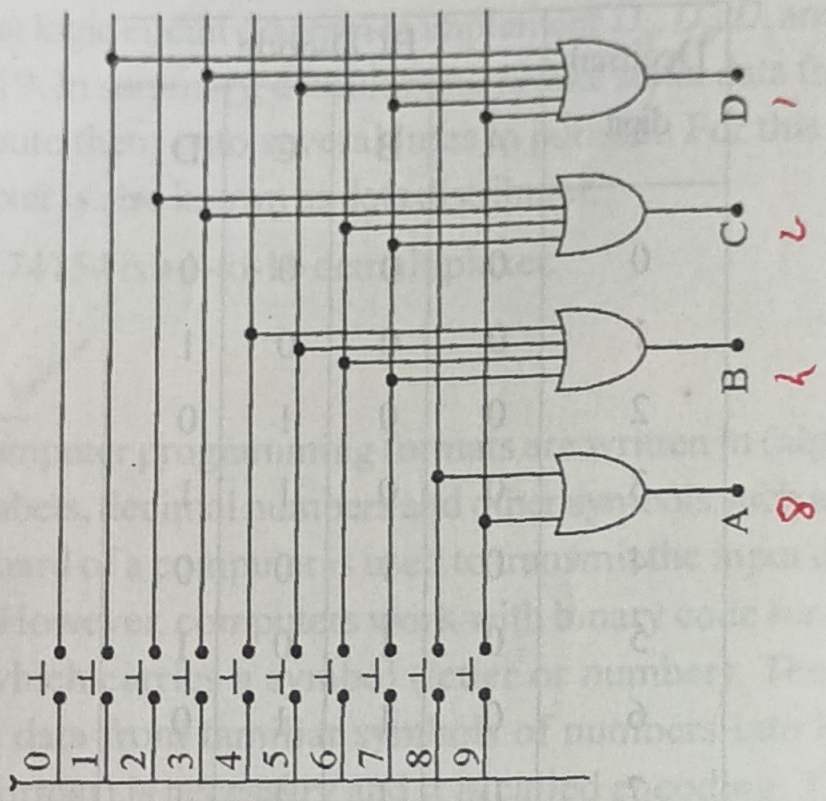

Fig. 120. Decimal to Binary encoder

The logic symbol of an encoder is given in figure 121.

A computer works with binary numbers. when a particular operation is over, it must produce the result in the form of alphanumeric character on the display screen (output). The process of converting the results from binary code into alphanumeric character is known as decoding.

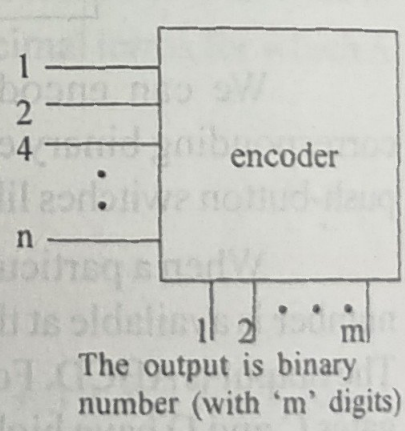

Fig. 121. Encoder logic symbol

Decoders

Decoders do the reverse function of encoders. *i.e.*, they must recognize a multiple binary input signal combination and convert it into a single output signal for producing a decimal format. The decoder is used in the output part of the computer.

Decoder is a circuit that responds to one specific input word or code, while rejecting all others. A decoder is designed to detect specific codes.

BCD-to-decimal decoder

It converts the BCD code into its decimal equivalent number. Consider the decimal equivalent of the binary codes listed in the Table below.

Suppose, we want to decode a BCD code representing one decimal digit-say 6. This operation may be carried out with a 4-input AND gate, excited by the four BCD bits. The output of the AND gate is 1 if and only if the BCD inputs are

$$A = 0,\ B = 1;\ C = 1;\ D = 0$$

i.e., 0110 input must produce the output 6.

A	B	C	D	Decimal number
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6 *
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Since this combination corresponds to 6, the output is labelled 'line 6'. When line-6 is alive, the corresponding nixi filaments in a bulb or LEDs in the seven-segment display are energized. Then if the input to the decoder is 0110, the number 6 will glow in the display connected to the output.

The logic circuit for implementation of a decimal number in the decoder can be obtained by forming a combination of high bits (bits with 1). For example, 7 can be realised from $ABCD = (0111)$, which is obtained from the Table above.

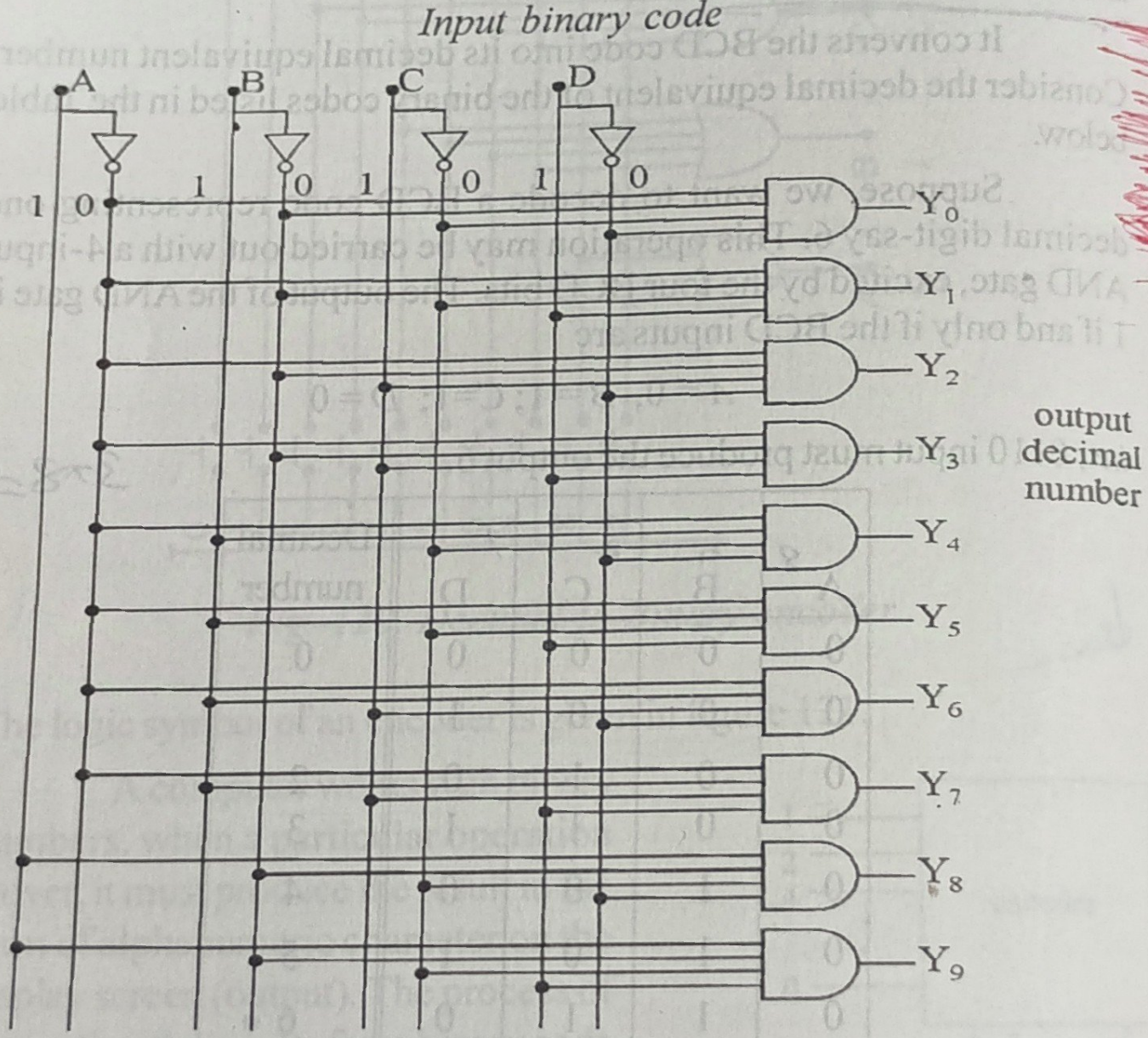

Fig. 122 BCD - to - decimal Decoder

Parity generator and parity checker

Parity

When an n-bit binary number has even number of 1 digits, it is said to have even parity. For example 110011 has four 1s. So, the number has even parity. When a given binary number has odd number of 1s, it is said to have odd parity. Parity of a binary number is dertermined by the number of 1s it has.

Parity checker

The parity of a binary number can be checked using Ex-OR gates. An

Ex-OR gate produces an output 1 when the input has an odd number of 1s. A word with

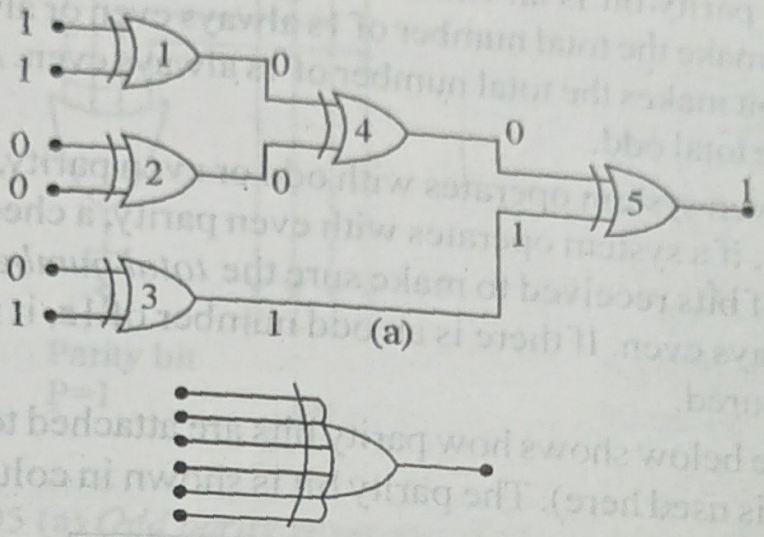

Fig. 94

even parity produces 0 - output. Figure shows a six input XOR gate. The binary word 110001 is fed to the six-input XOR gate.

The output of Ex-OR -1 is 0. The output of Ex-OR-2 is 0. ∴ The output of gate 4 is 0. The output of Ex-OR -3 is 1. Finally the output of gate 5-is 1. Since the output is 1, the input word has odd parity. This is found to be true on inspecting the pairty of the input word. Similarly if the final output is 0, it means that the input word has even parity. In this way Ex-OR gates can be used as parity checker.

Parity checker is a circiut that tests the pairty bit that has been added to a binary word so that a pairty check can be performed (made).

Pairty generator

When digital codes are being tranmitted from one system to another system or when the codes are being tranferred from one point to another point in a digital system, errors can occour. The errors may be due to component malfunction or electrical noise. The errors may take the form of unnecessary changes in the bits of the words handled. i.e., a 1 may change to 0 or 0 may change to 1 in the word.

As a means of detecting an error in a bit, many systems employ a pairty bit. A parity bit is an extra bit included with a binary message (number) to make the total number of 1s always even or always odd. An even parity bit makes the total number of 1s always even. An odd parity bit makes the total odd.

A given system operates with odd or even parity, but not both. For example, if a system operates with even parity, a check is made on each group of bits received to make sure the *total number* of 1s in that group is always even. If there is an odd number of 1s, it means that an error has occured.

Table below shows how parity bits are attached to a code word. (B C D code is used here). The parity bit is shown in column *P*.

Even parity		Odd parity	
P	Word	*P*	Word
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

Fig. 95 (a)

The arrangement for odd parity generation is shown in the figure 95 (a). A four-bit code x_3, x_2, x_1, x_0 is applied as the input word. (Example data bit 0110). The input bits are applied to the parity bit generator which is a four input XOR gate, followed by a NOT gate. The output of the NOT is a 1. With this 1 and 0110 in the input word, the total number of 1s is odd. Thus a five - bit number with odd pairty is generated as 10110.

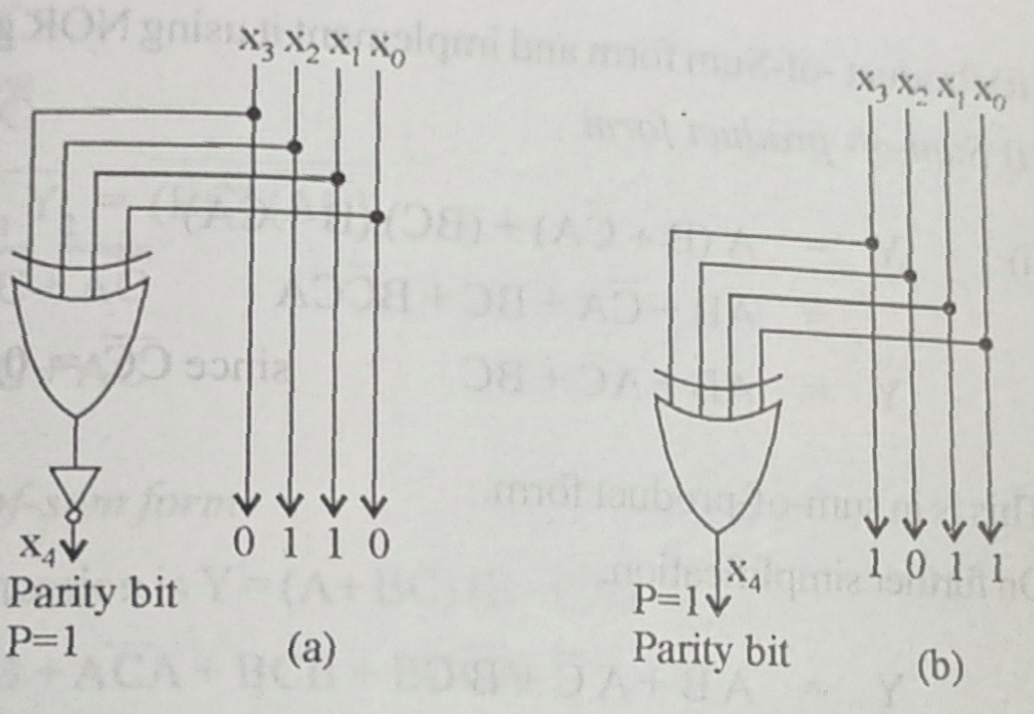

Fig. 95 (a) *Odd parity generator (b) Even parity generator.*

Similarly when there is odd number of 1s in the input word the output in the NOT gate will be 0. In this way a 0 parity bit can be produced.

To get even parity generator the same arrengement can be used, but without the inverter. (Fig 95 (b). Let this input word to the even parity generator be 1011. This is fed as x_3, x_2, x_1, x_0 to the four input XOR gate. The output of XOR is 1 when there are odd number of 1s in the input. So, the parity bit generated is 1. Including the parity bit, the word is 11011, which has even parity. Pairty generator is a circuit that produces either an odd or even parity bit to go along with the data (word).