UNIT 4

COMBINATIONAL LOGIC

Combining a number of basic logic gates in a larger circuit to produce more complex logical operations is called combinational logic. Using such circuits, logical operations can be performed on any number of inputs whose logic state is either 1 or 0 and this technique is the basis of all digital electronics.

In combinational logic, the output is a pure function of the present input only. This is in contrast to sequential logic, in which the output depends not only on the present input but also on the history of the input. In other words, sequential logic has memory while combinational logic does not.

Combinational logic is used in computer circuits to perform Boolean algebra on input signals and on stored data. Practical computer circuits normally contain a mixture of combinational and sequential logic. For example, the part of an arithmetic logic unit, or ALU, that does mathematical calculations is constructed using combinational logic. Other circuits used in computers, such as half adders, full adders, half subtractors, full subtractors, multiplexers, demultiplexers, encoders and decoders are also made by using combinational logic.

Some of the characteristics of combinational circuits are following −

- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
- The combinational circuit do not use any memory element. The previous state of input does not have any effect on the present state of the circuit.
- A combinational circuit can have an n number of inputs and m number of outputs.

Design Procedure

1. The problem is stated (Verbal description).
2. Specify the number of inputs and required numbers of outputs.
3. The input and output variables are assigned letter symbols.
4. Construct the truth table to define relationship between inputs and outputs.

5. The simplified Boolean function for each output is obtained (using K-Map, Tabulation method and Boolean Algebra rules).
6. The logic diagram is drawn.

ADDER

An adder is a digital circuit that performs addition of numbers. In many computers and other kinds of processors adders are used in the arithmetic logic units or ALU. They are also used in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators and similar operations.

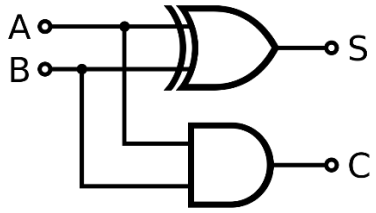
HALF ADDER

The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C). The carry signal represents an overflow into the next digit of a multi-digit addition. The value of the sum is A + B. The simplest half-adder design incorporates an XOR gate for S and an AND gate for C. The input variables of a half adder are called the augend and addend bits. The output variables are the sum and carry. The Boolean expression of half adder is

$S = A'B + AB'$

$C = AB.$

Logic Diagram



Truth Table

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

FULL ADDER

Full Adder is the adder which adds three inputs and produces two outputs. With the addition of an OR gate to combine their carry outputs, two half adders can be combined to make a full adder. So, a full adder can be constructed from two half adders by connecting A and B to the input of

one half adder, then taking its sum-output S as one of the inputs to the second half adder and $Carry_{in}$ as its other input, and finally the carry outputs from the two half-adders are connected to an OR gate. The sum-output from the second half adder is the final sum output (S) of the full adder and the output from the OR gate is the final carry output ($Carry_{out}$).

Truth Table

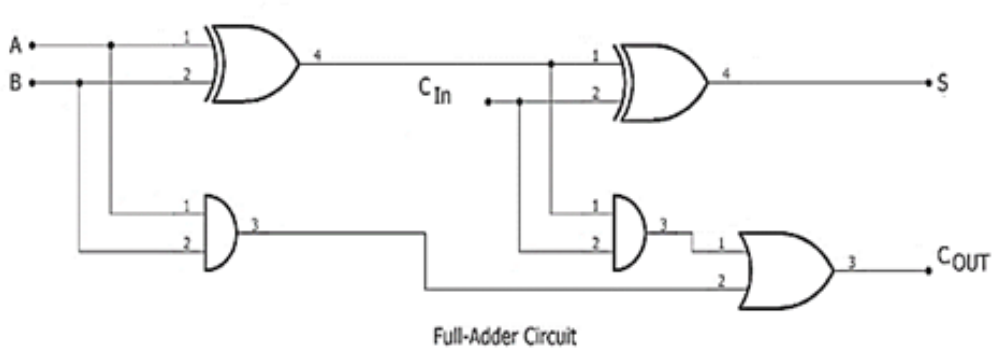
Inputs			Outputs	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The Boolean expression for full adder is:

$S = A \oplus B \oplus C_{in}$

$C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$.

Logic Diagram



Full-Adder Circuit

SUBTRACTOR

A subtractor is a digital circuit that performs subtraction of numbers. It is a combinational circuit that can perform subtraction of a given bit.

HALF SUBTRACTOR

The half subtractor is a combinational circuit which is used to perform subtraction of two bits. Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit. In electronics, a subtractor can be designed using the same approach as that of an adder. The boolean expression is as follows:

$Diff = A'B + AB'$

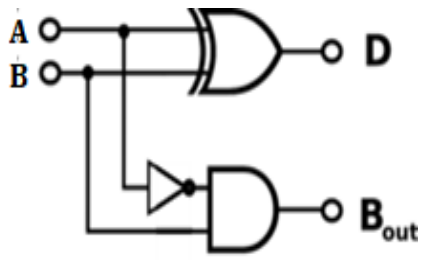
$B_{out} = A'B$

'A' and 'B' are the input variables whose values are going to be subtracted. The 'Diff' and 'Borrow' are the variables whose values define the subtraction result, i.e., difference and borrow.

The first two rows and the last row, the difference is 1, but the 'Borrow' variable is 0.

The third row is different from the remaining one. When we subtract the bit 1 from the bit 0, the borrow bit is produced.

Logic Diagram



Truth Table

Inputs		Outputs	
A	B	D	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

FULL SUBTRACTOR

The Half Subtractor is used to subtract only two numbers. To overcome this problem, a full subtractor was designed. The full subtractor is used to subtract three 1-bit numbers A, B, and C (borrow in), which are minuend, subtrahend, and borrow, respectively. The full subtractor has three input states and two output states i.e., diff and borrow.

The actual logic circuit of the full subtractor is shown in the above diagram. The full subtractor circuit construction can also be represented in a Boolean expression.

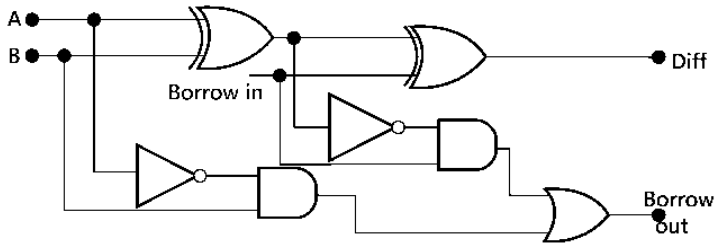
Diff = (A ⊕ B) ⊕ $Borrow_{in}$

Borrow = A'.B + (A ⊕ B)'. $Borrow_{in}$

Truth Table

Inputs			Outputs	
A	B	$Borrow_{in}$	Diff	Borrow
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

Logic Diagram



In the above table, 'A' and' B' are the input variables. These variables represent the two significant bits that are going to be subtracted. 'Borrowin' is the third input which represents borrow. The 'Diff' and 'Borrow' are the output variables that define the output values.

The eight rows under the input variable designate all possible combinations of 0 and 1 that can occur in these variables.

MULTIPLEXER

Multiplexer is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

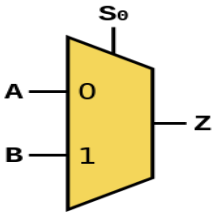
A multiplexer of inputs has select lines, which are used to select which input line to send to the output. Multiplexers are mainly used to increase the amount of data that can be sent over

the network within a certain amount of time and bandwidth. Multiplexers can also be used to implement Boolean functions of multiple variables.

An electronic multiplexer makes it possible for several signals to share one device or resource, for example, one analog-to-digital converter or one communications transmission medium, instead of having one device per input signal.

The basic function of a multiplexer is combining multiple inputs into a single data stream. On the receiving side, a demultiplexer splits the single data stream into the original multiple signals.

In digital circuit design, the selector wires are of digital value. In the case of a 2-to-1 multiplexer, a logic value of 0 would connect to the output while a logic value of 1 would connect to the output. A 2-to-1 multiplexer has a boolean equation where A and B are the two inputs, S_0 is the selector input, and Z is the output:



S_0	Z
0	A
1	B

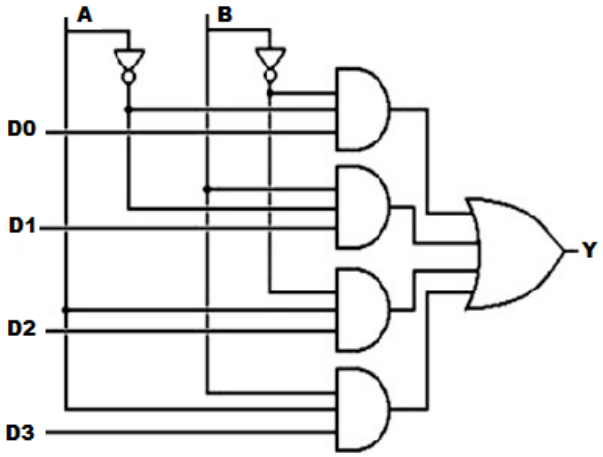
4:1 MULTIPLEXER

The 4-to-1 multiplexer has 4 input bit, 2 control bits, and 1 output bit. The number of control input depends upon the number of inputs. For 2^n as input, n is the control input. The four input bits are D0, D1, D2 and D3. only one of this is transmitted to the output Y The output depends on the value of AB which is the control input. The control input determines which of the input data bit is transmitted to the output. For instance, as shown in fig. when AB = 00, the upper AND gate is enabled while all other AND gates are disabled. Therefore, data bit D0 is transmitted to the output, giving Y = Do. If the control input is changed to AB =11, all gates are disabled except the bottom AND gate. In this case, D3 is transmitted to the output and Y = D3. When A=0, B=1, D1 is the output and when A=1, B=0, D2 is the output. So, the 4:1 multiplexer select any of the input lines (D0, D1, D2, D3) depending upon the select input (A, B).

Logic Expression

$Y=A'B'D_0 + A'BD_1 + AB'D_2 + ABD_3$

Logic Diagram



Truth Table

Select Input		Output
A	B	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

8:1 MULTIPLEXER

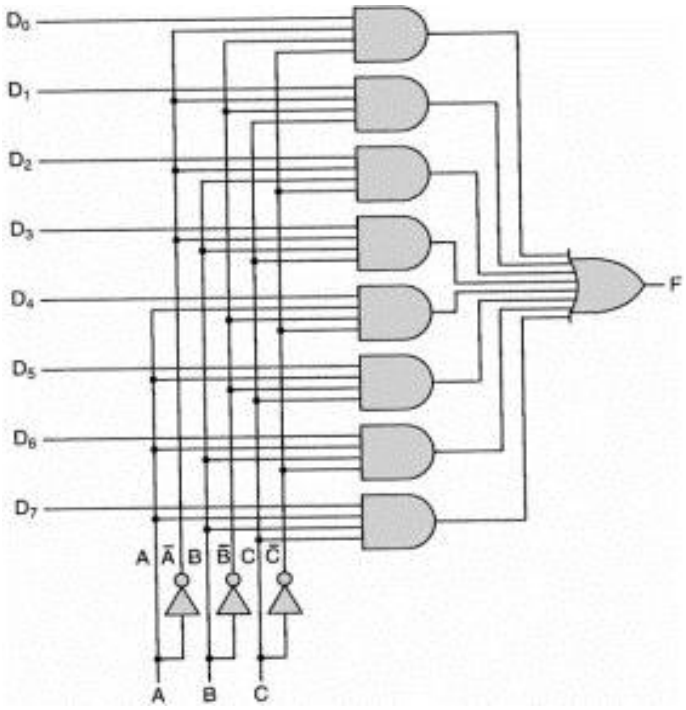
The 8-to-1 multiplexer has 8 input bit, 3 control bits, and 1 output bit. The eight input bits are D0, D1, D2,…, D7. Only one of this is transmitted to the output F. The output depends on the value of ABC which is the control input. The control input determines which of the input data bit is transmitted to the output.

When ABC = 000, the upper AND gate D0 is enabled while all other AND gates are disabled. Therefore, data bit D0 is transmitted to the output, giving F = Do. If the control input is changed to AB =111, all gates are disabled except the bottom AND gate. In this case, D7 is transmitted to the output and F = D7. When A=0, B=0, C=1, D1 is the output and when A=0, B=1, C=0 D2 is the output and so on. So, the 8:1 multiplexer select any of the input lines (D0,D1,D2,…D7) depending upon the select input (A, B, C). The logic diagram for 8:1 multiplexer is drawn below.

Logic Expression

$F=A'B'C'D_0 + A'B'CD_1 + A'BC'D_2 + A'BCD_3 + AB'C'D_4 + AB'CD_3 + ABC'D_3 + ABCD_3$

Logic diagram



Truth Table

Select Input			Output
A	B	C	F
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

DEMULTIPLEXERS

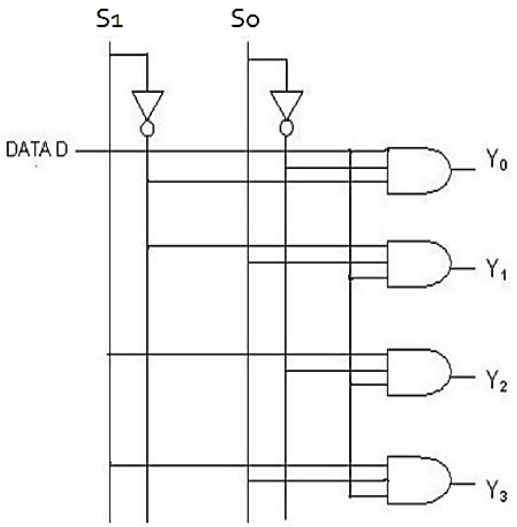
The demultiplexer is a combinational logic circuit designed to switch one common input line to one of several separate output lines. Demultiplexers take one data input and a number of selection inputs, and they have several outputs. They forward the data input to one of the outputs depending on the values of the selection inputs. Demultiplexers are sometimes convenient for designing general-purpose logic because if the demultiplexer's input is always true, the demultiplexer acts as a binary decoder. This means that any function of the selection bits can be constructed by logically OR-ing the correct set of outputs.

1:4 DEMULTIPLEXER

The 1 to 4 demultiplexer consists of one input, four outputs, and two control lines to make selections. The input bit is Data D with two select lines A and B. The input bit D is transmitted to four output bits Y0, Y1, Y2, and Y3. When AB is 01 The upper second AND gate is enabled while the other AND gate is disabled. Thus, only one data is transmitted at Y1. If D is low, then Y1 is low and if D is high, Y1 is high. The value of Y1 depends on the value of D.

If the control input changes to AB=10 all the gates are disabled except the third AND gate from the top. Then D is transmitted to output Y2.

The diagram below shows the circuit of 1 to 4 demultiplexer.



Truth Table

Input	Select Input		Output Lines			
D	S1	So	Yo	Y1	Y2	Y3
D	0	0	D	0	0	0
D	0	1	0	D	0	0
D	1	0	0	0	D	0
D	1	1	0	0	0	D

Boolean Expression:

$Y0 = S_1'S_0'D$

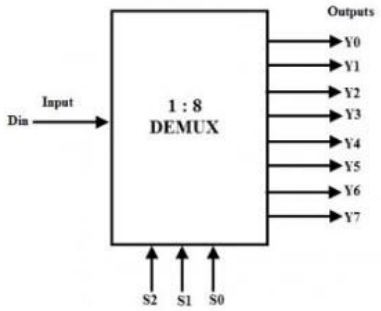
$Y1 = S_1' S_0 D$

$Y2 = S_1 S_0' D$

$Y3 = S_1 S_0 D$

1:8 DEMULTIPLEXER

A 1 to 8 demultiplexer consists of one input line, 8 output lines and 3 select lines. Let the input be D, S1 and S2 are two select lines and eight outputs from Y0 to Y7. It is also called as 3 to 8 demux because of the 3 selection lines. Below is the block diagram of 1 to 8 demultiplexer.

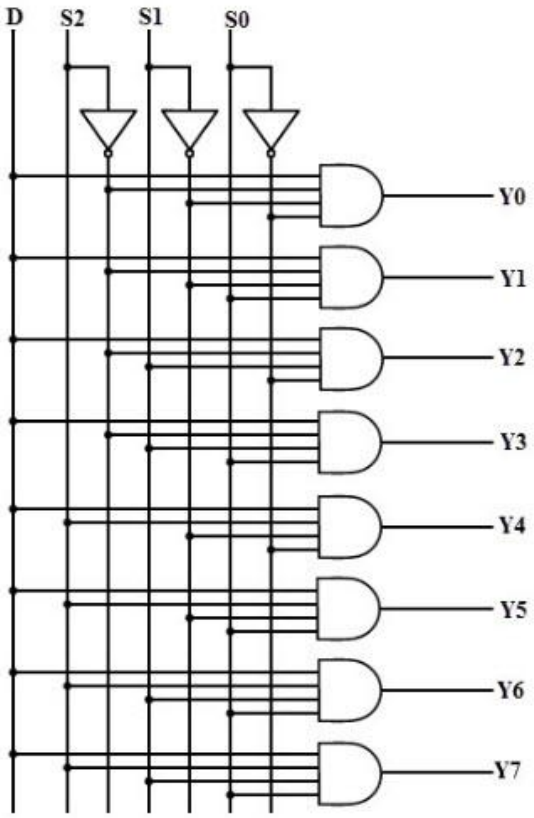


Truth Table

The below is the truth table for 1 to 8 demultiplexer. It tells the functionality of the demultiplexer, like, if $S_2S_1S_0$=000, then the output is seen at Y_0 and so on.

Data Input	Select Inputs			Outputs							
D	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

Using the above truth table, the logic diagram of the demultiplexer is implemented using eight AND and three NOT gates. The different combinations of the select lines select one AND gate at given time, such that data input will be seen at a particular output.



Boolean Expression

$Y0 = S_2' S_1' S_0' D$

$Y1 = S_2' S_1' S_0 D$

$Y2 = S_2' S_1 S_0' D$

$Y3 = S_2' S_1 S_0 D$

$Y4 = S_2 S_1' S_0' D$

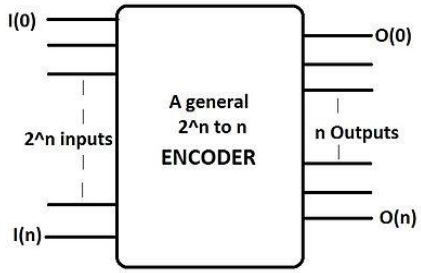
$Y5 = S_2 S_1' S_0 D$

$Y6 = S_2 S_1 S_0' D$

$Y7 = S_2 S_1 S_0 D$

ENCODER

An **Encoder** is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and 'n' output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with 'n' bits. It is optional to represent the enable signal in encoders.



A General encoder's block diagram.

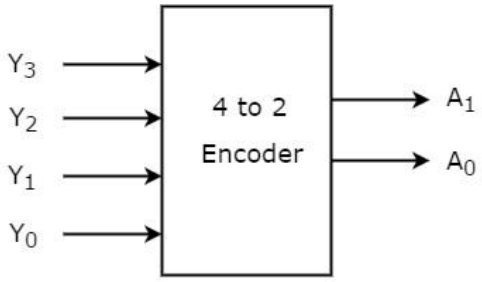
TYPES OF ENCODER

A 2^n-to-n encoder has n number of outputs in correspondence to the 2^n number of inputs. It thus reduces the number of transmission lines and can be compared to a multiplexer. Only one of the inputs become "high" (logic state "1") at a time.

Some typical examples would be 4:2 encoder, 8:3 encoder, 16:4 encoders etc.

4 TO 2 ENCODER

Let 4 to 2 Encoder has four inputs Y_3, Y_2, Y_1 & Y_0 and two outputs A_1 & A_0. The **block diagram** of 4 to 2 Encoder is shown in the following figure.



At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output.

The **Truth table** of 4 to 2 encoder is shown below.

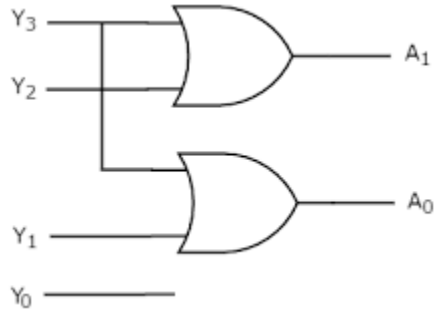
Inputs				**Outputs**	
Y_3	**Y_2**	**Y_1**	**Y_0**	**A_1**	**A_0**
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

From Truth table, we can write the **Boolean functions** for each output as

A1=Y3+Y2

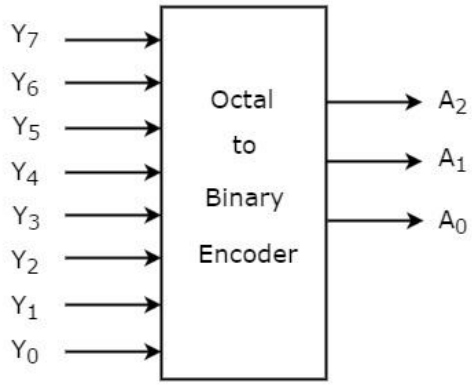
A0=Y3+Y1

We can implement the above two Boolean functions by using two input OR gates. These OR gates encode the four inputs with two bits. The **circuit diagram** of 4 to 2 encoder is shown in the following figure.



8 TO 3 ENCODER

Octal to binary (8 to 3) Encoder has eight inputs, Y_7 to Y_0 and three outputs A_2, A_1 & A_0. Octal to binary encoder is nothing but 8 to 3 encoders. The **block diagram** of octal to binary Encoder is shown in the following figure.



At any time, only one of these eight inputs can be '1' in order to get the respective binary code. The **Truth table** of octal to binary encoder is shown below.

Inputs								**Outputs**		
Y_7	**Y_6**	**Y_5**	**Y_4**	**Y_3**	**Y_2**	**Y_1**	**Y_0**	**A_2**	**A_1**	**A_0**
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

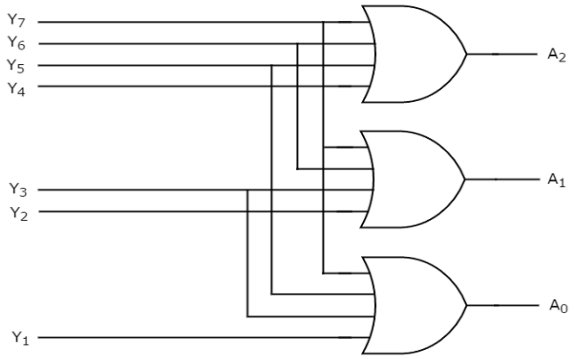
From Truth table, we can write the **Boolean functions** for each output as

$A2=Y7+Y6+Y5+Y4$

$A1=Y7+Y6+Y3+Y2$

$A0=Y7+Y5+Y3+Y1$

We can implement the above Boolean functions by using four input OR gates. These OR gates encode the eight inputs with three bits. The **circuit diagram** of octal to binary encoder is shown in the following figure.



Drawbacks of Encoder

- There is an ambiguity, when all outputs of encoder are equal to zero. Because, it could be the code corresponding to the inputs, when only least significant input is one or when all inputs are zero.

- If more than one input is active High, then the encoder produces an output, which may not be the correct code. For **example**, if both Y_3 and Y_6 are '1', then the encoder produces 111 at the output. This is neither equivalent code corresponding to Y_3, when it is '1' nor the equivalent code corresponding to Y_6,

DECODER

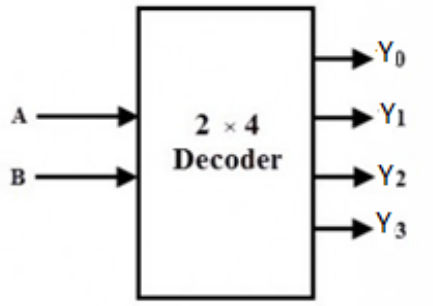
In digital electronics, a binary decoder is a combinational logic circuit that converts binary information from the n coded inputs to a maximum of 2^n unique outputs. They are used in a wide variety of applications, including data multiplexing and data demultiplexing, seven segment displays, and as address decoders for memory and port-mapped I/O.

There are several types of binary decoders, but in all cases a decoder is an electronic circuit with multiple input and multiple output signals, which converts every unique combination of input state to a specific combination of output states. In addition to integer data inputs, some decoders also have one or more "enable" inputs. When the enable input is negated (disabled), all decoder outputs are forced to their inactive states.

Depending on its function, a binary decoder will convert binary information from n input signals to as many as 2^n unique output signals. Some decoders have less than 2^n output lines; in such cases, at least one output pattern may be repeated for different input values.

2-TO-4 BINARY DECODER

In a 2-to-4 binary decoder, two inputs are decoded into four outputs hence it consists of two input lines and 4 output lines. Only one output is active at any time while the other outputs are maintained at logic 0 and the output which is held active or high is determined the two binary inputs A and B.



The figure below shows the truth table for a 2-to-4 decoder. For a given input, the outputs Y_0 through Y_3 are active high if enable input EN is active high (EN = 1). When both inputs A and B are low (or A= B= 0), the output Y_0 will be active or High and all other outputs will be low.

When A = 0 and B = 1, the output Y_1 will be active and when A = 1 and B = 0, then the output Y_2 will be active. When both the inputs are high, then the output Y_3 will be high. If the enable bit is zero then all the outputs will be set to zero. This relationship between the inputs and outputs are illustrated in below truth table clearly.

Truth Table

Inputs			Outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	×	×	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Boolean Expression

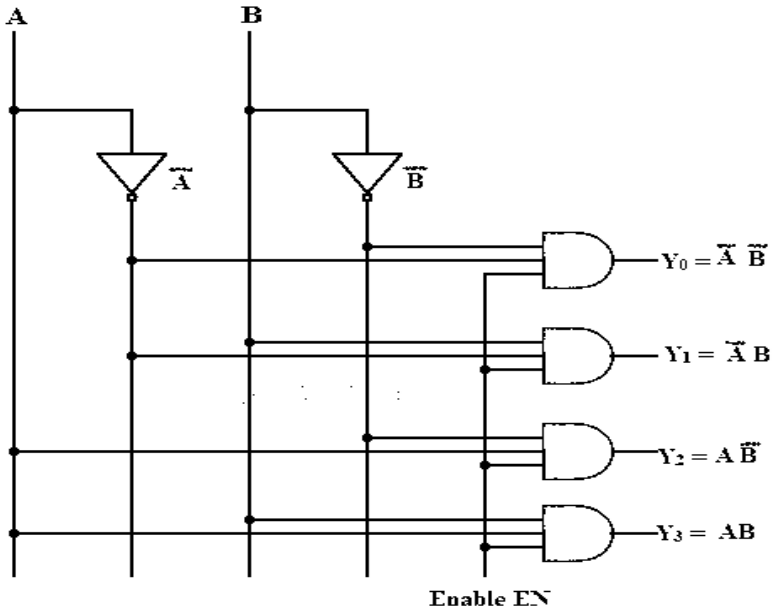
$Y0 = \overline{A}\,\overline{B}$

$Y1 = \overline{A}\,B$

$Y2 = A\,\overline{B}$

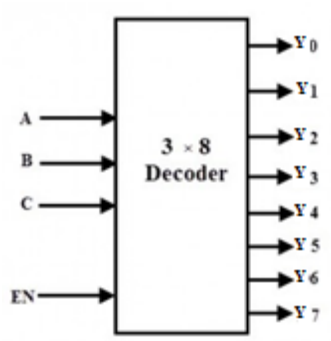
$Y3 = A\,B$

These expressions can be implemented by using basic logic gates. Thus, the logic circuit design of the 2-to-4 line decoder is given below which is implemented by using NOT and AND gates. Two NOT gates or inverters provide the complement of inputs. A common enable line is connected to each AND gate such that when EN= 0 all the outputs are zero and if EN=1, depends on the inputs A and B, outputs are produced. Each output represents one of the minterms of the 2 input variables.



3-TO-8 DECODER

In a 3-to-8 decoder, three inputs are decoded into eight outputs. It has three inputs as A, B, and C and eight output from D0 through D7. Based on the combinations of the three inputs, only one of the eight outputs is selected.



The figure below shows the truth table of a 3-to-8 decoder. Enable input is provided to activate the decoded output depends on the input combinations A, B and C. Suppose if A = B=1 and C= 0, then the output Y6 is 1 and all other outputs are zero. So from the truth table, minterms represents the each output equation and are given as:

Truth Table

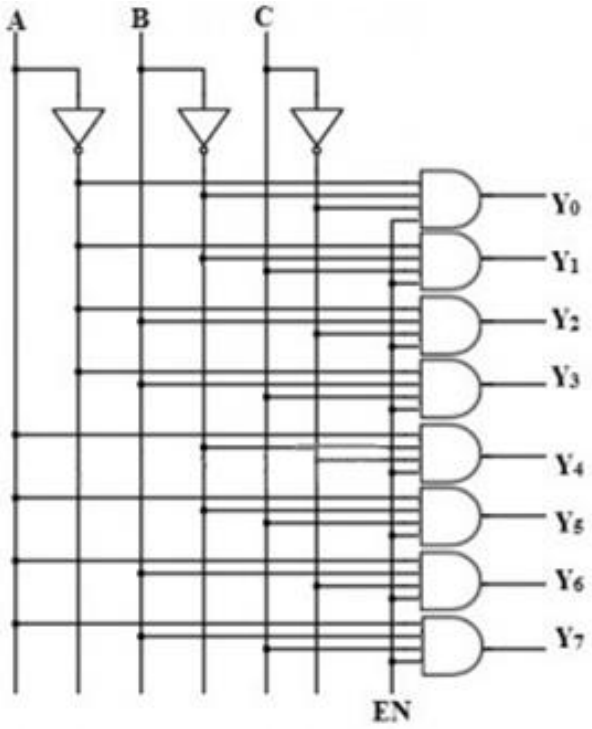
Inputs				Outputs							
EN	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	×	×	×	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Boolean Expression

$Y0 = \bar{A}\,\bar{B}\,\bar{C}$
$Y1 = \bar{A}\,\bar{B}\,C$
$Y2 = \bar{A}\,B\,\bar{C}$
$Y3 = \bar{A}\,B\,C$
$Y4 = A\,\bar{B}\,\bar{C}$
$Y5 = A\,\bar{B}\,C$
$Y6 = A\,B\,\bar{C}$
$Y7 = A\,B\,C$

Using the above min term expressions for each output, the circuit of 3-to-8 decoder is can be implemented by using three NOT gates and eight AND gates. Each NOT gate provides the complement of the input and AND gates generates one of the minterms.

Also enable input activate the decoded output depends on the input data. The logic diagram of this decoder is shown below. Only one of eight outputs is high at a given time for a particular input combination, that why this decoder is also called as 1-of-8 decoder. Suppose, when ABC = 011, then only AND gate 4 has all inputs high, thus Y3 is high. Also, 3-bit binary numbers at the input is converted to eight digits at the output (which is equivalent to octal number system), that's how; it is also called as a binary-to-octal decoder.



PROGRAMMABLE LOGIC ARRAY

Programmable Logic Array (PLA) is a programmable device used to implement combinational logic circuits. The PLA has a set of programmable AND, which link to a set of programmable OR planes, which can then be conditionally complemented to produce an output. This layout allows for a large number of logic functions to be synthesized in the sum of products canonical forms.

A programmable logic array (PLA) has a programmable AND array at the inputs and programmable OR array at the outputs. The PLA has a programmable AND array instead of

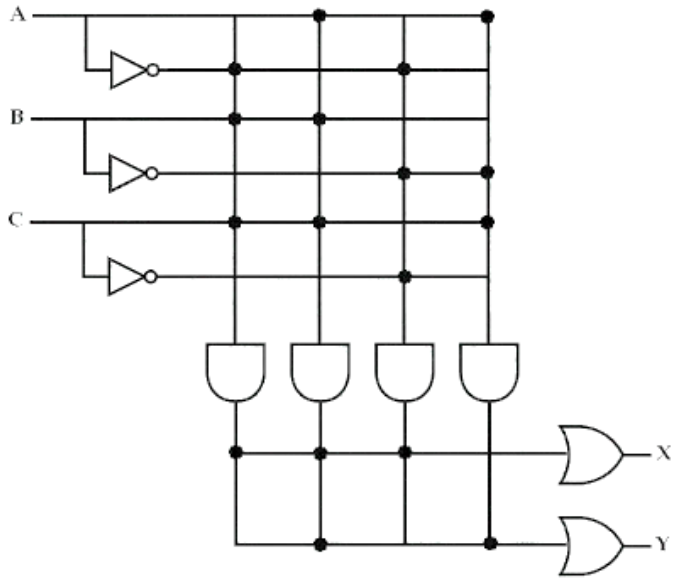
hard-wired AND array. The number of AND gates in the programmable AND array are usually much less and the number of inputs of each of the OR gates equal to the number of AND gates. The OR gate generates an arbitrary Boolean function of minterms equal to the number of AND gates. Figure below shows the PLA architecture with four input lines, a programmable array of eight AND gates at the input and a programmable array of two OR gates at the output.

Suppose we need to implement the functions:

$X = A'BC + ABC + A'B'C'$ and

$Y = ABC + AB'C$.

The following figures shows how PLA is configured. The big dots in the diagram are connections. For the first AND gate (left most), *A* complement, *B*, and *C* are connected, which is first minterm of function *X*. For second AND gate (from left), *A*, *B*, and *C* are connected, which forms *ABC*. Similarly for *A'B'C'*, and *AB'C*. Once the minterms are implemented. Now we have to combine them using OR gates to the functions *X*, and *Y*.



One application of a PLA is to implement the control over a data path. It defines various states in an instruction set, and produces the next state (by conditional branching). Note that the use of the word "Programmable" does not indicate that all PLAs are field-programmable; in fact many are mask-programmed during manufacture in the same manner as a ROM. This is particularly true of PLAs that are embedded in more complex and numerous

integrated circuits such as microprocessors. PLAs that can be programmed after manufacture are called FPLA (Field-programmable logic array).

Solved Example 1:

Realize the Boolean expression

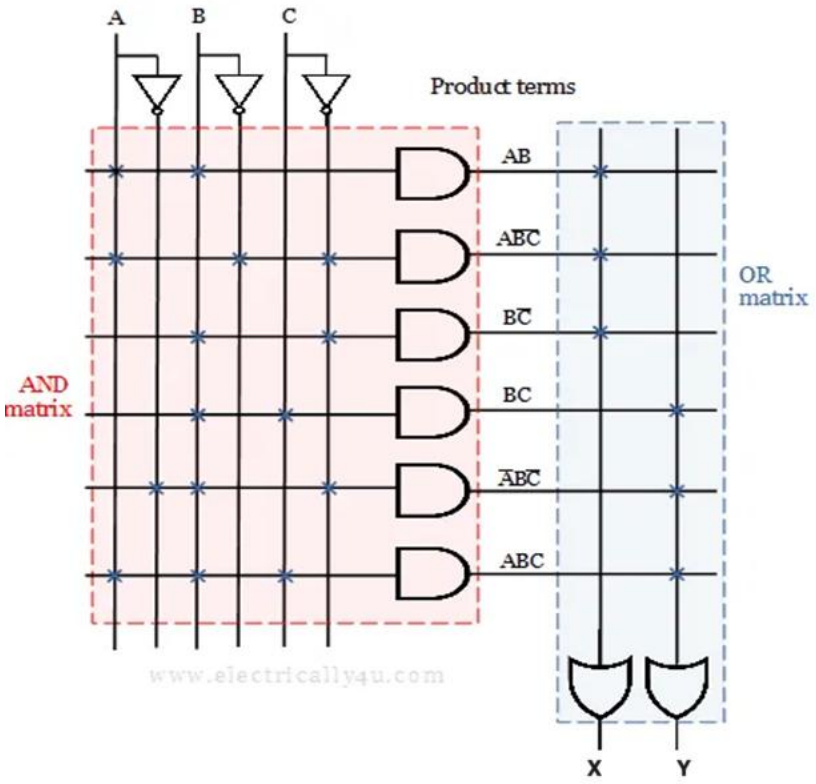
X = AB + AB'C' + BC' and

Y = BC + A'BC' + ABC

using Programmable Logic Array

Ans:

- For the given problem, there are three inputs(A, B, C) and two outputs(X, Y). The complement of three inputs are obtained through NOT gates. Thus the realization has six input lines(input with its complement).
- The given expression has six product terms
- Two OR gate arrays are used at the output to realize the the two functions. The fuses are placed at the corresponding product terms for each Boolean function.



SolvedExample 2:

Realize a boolean functions

F1(A, B, C) = ∑ m(1, 3, 6, 7) and

F2(A, B, C) = ∑ m(0, 2, 4, 5)

using PLA.

A \ BC	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$F_1 = \bar{A}C + AB$

A \ BC	00	01	11	10
0	1	0	0	1
1	1	1	0	0

$F_2 = \overline{AC} + A\bar{B}$