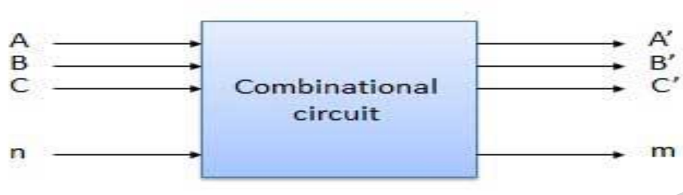
Module 1

Combinational Circuit(cc)

- A combinational circuit are circuit made up of different types of logic gates.That is, combination circuit is a connected arrangement of logic gates with a set of inputs and outputs
- A Logic gates is a basic building block of any electronic circuit
- At any given time, the binary values of the outputs are a function of the binary combination of the inputs. That is, the output of the combinational circuit depends on the values at the input at any given time.
- A block diagram of a combinational circuit are



- Then binary input variables come from an external source, The m binary output variable go to an external destination and in between these is an interconnection of logic gates.
- A combinational circuit transform binary information from the given input data to the required output data.
- combinational circuits are employed in digital computers for generating binary control decisions and for providing digital components required for data processing

Truth table

- A combinational circuit can be described by a truth table showing the binary relationship between the 'n' input variables and the 'n' output variables
- The truth table lists the corresponding output binary values for each of the second input combinations .
- A combination circuit can also be specified with **'m'** Boolean Functions, One for each output variable.
- Each output function is expressed in terms of the n input variables

The Adder

➢ An adder also called summer
➢ The most common Adders operate on binary numbers
➢ The digital circuits that generate the Arithmetic sum of binary numbers of any length is called a binary adder.
➢ It is mainly divided into two types

* HALF ADDER

* FULL ADDER

HALF ADDER

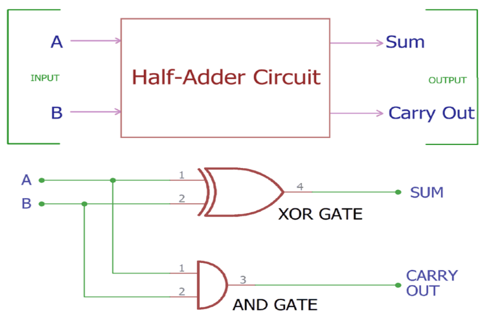
✓ The most basic digital arithmetic circuit is the addition of two binary digits.

✓ A combinational circuit that performs the arithmetic addition of two bits is called a half adder

That is, the two input A and B which add two input digits and generate 2 outputs (A carry and sum)

✓ The half adder adds two binary digits.
✓ The input variables of a half adder are called the augend and addend bits. the output variables are the sum and carry

Circuit Diagram

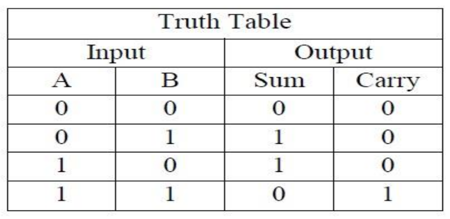


S=SUM

C=CARRY

The two-half adder needed to implement a full adder.

TRUTH TABLE



Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- in half adder, it is necessary to specify 2 output variables because the sum of 1+1 is binary 10, which has 2 digits

ie,

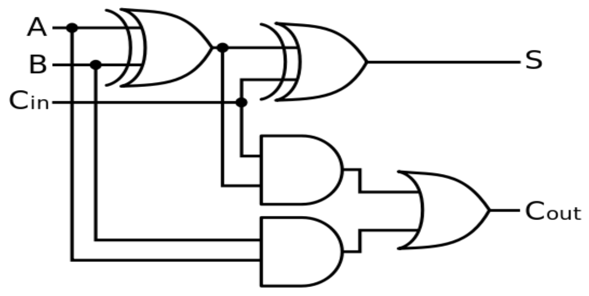
A	B	SUM
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**10**

Full adder

- It is a combinational circuit that performs the Arithmetic sum of three input bits.
- A combinational circuit that performs the arithmetic addition of 3 bits(2 significant bits and a previous carry) is called a full adder.

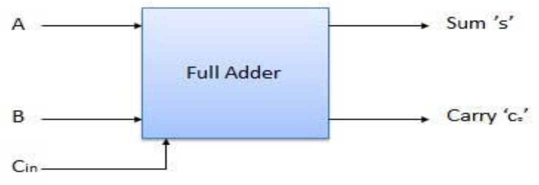
that is the three-input a, b and c which add the three input numbers and generate two outputs (a carry and sum)

circuit diagram



- The two-half adder to needed to implement full adder

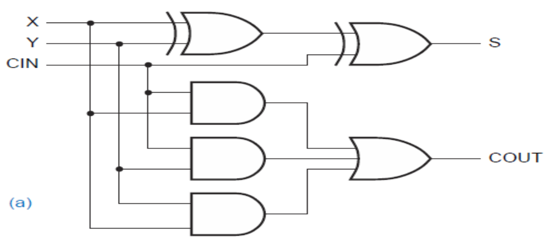
Block diagram



Truth table

INPUTS			OUTPUT	
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

Circuit Diagram



HALF ADDER

- the Boolean function for the two outputs can be obtained directly from the truth table

  S=X'Y+XY'

  =X (+) Y

  =XY
- In logic diagram, it is consisting of an exclusive-OR gate and an AND Gate .

FULL ADDER

- Two output are necessary because the arithmetic sum of three binary digits Ranges in value from 0 to3 and binary two or three needs two digits
- The two output are designated by the symbols S (for sums) and C(for carry).
- In truth table, the 8 Rows under the input variables designate all possible combinations that the binary variable may have
- The value of the output variable are determined from the Arithmetic sum of the input bits . when all inputs are zero, the 0 output is zero.the S output is equal to one when only one input is equal to 1 or when all three inputs are equal to 1
- The c output has a Carry of 1 if 2 or 3 input are equal to 1
- 1 possible expression for C is

C=XY+(X'Y+XY')Z

Realizing that x' y + x y'=x (+) y

S=x (+) y (+) z

C=x y+ (x (+) y)

- The full adder circuit consists of two half adder and OR gate

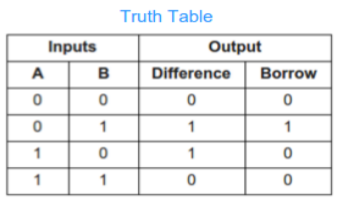
Substractor

- the most common substractor operate on binary numbers
- the Digital circuit that is generated The Arithmetic subtraction of binary numbers of any length is called a binary subs tractor
- it is divided into two types
  - half substractor
  - full substractor

Half substractor

- half subs tractor is used to perform two binary digit subs traction
- the substation circuit uses binary number (0,1) for the substations
- the circuit of half sub tractor can be built with two logic gates namely NAND and XOR gates
- this circuit give two elements such as the difference as well as the borrow
- In binary subs traction, the process of subtraction is similar to arithmetic subtractions. in Arithmetic subs traction the base 2 numbers system is used where as in binary subs traction
- Binary numbers are used for substations
- The resultant terms can be denoted with the difference and borrow

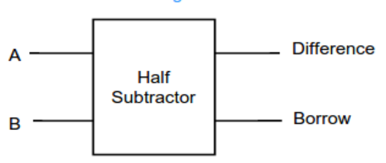
Truth table



Truth Table

Inputs		Output	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Block Diagram



Difference D=(X'Y+XY')

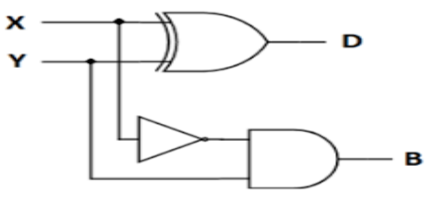
=X(+)Y

Borrow B=X'Y

APPLICATION OF HALF SUBSTRACTOR

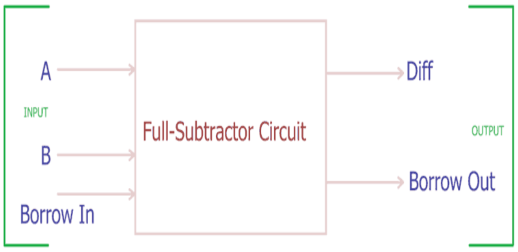
- HALF Subs tractor is used to reduce the force of radio or audio signals
- It can be used in amplifiers to reduce the sound distortion.
- half subs tractor is used in ALU of processor

logic circuit



Full subtraction

- Full subscription is used to perform 3 binary digits substation
- The subs tractor circuit used a binary number (0,1) for the substractions
- The circuit give three elements such as the different as used as the borrow



TRUTH TABLE

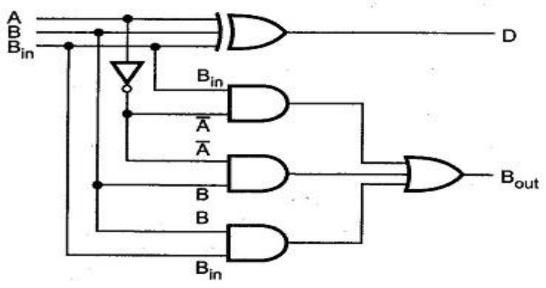
Input			Output	
A	B	C	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

D=X’ Y’Bin + X’ Y Bin’+ X Y’Bin+ XYBin

The full substractor is a combinational circuit with three input x, y,Bin and two output D, Bout

Here x is the mainued
Y is the subs trahend
B is the borrow

Circuit diagram



Binary subtraction

Binary subtraction is also similar to that of normal subs traction except the fault that when 1 is subtracted from zero, it is borrowed from next higher order bit and that bit is reduced by one

A-B	**SUBS TRACT**	**BORROW**
0-0	**0**	**0**
0-1	**1**	**1**
1-0	**1**	**0**
1-1	**0**	**0**

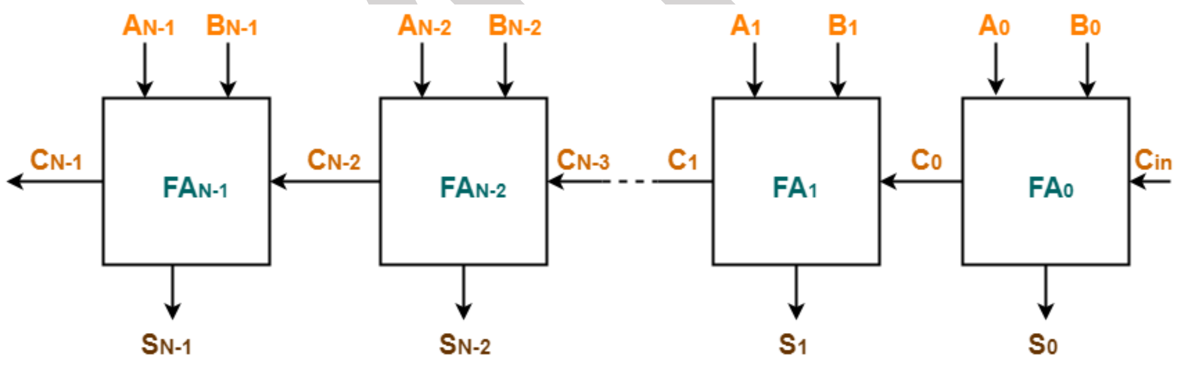
Binary addition

binary addition is similar to that of normal method of addition except the fault that it carries a value of two instead of 10 . this is because in binary number system two is represented as 10

A+B	SUM	CARRY
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1

Ripple carry Adder

- Ripple carry adder is a digital circuit that produce the Arithmetic sum of two binary numbers
- It can be constructed with full adders connected in cascaded with the carry output from cache full adder connected to the carry input of the next full adder in the chain
- example of the interconnection of a 4-full adder (FA)CIRCUIT to provide a ripple carry adder



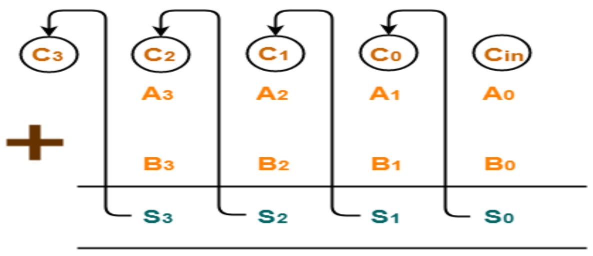
Here so to s3 represent the sum of ripple carry adder

- A ripple carry adder also known as n-bit parallel adder it is combinational logic circuit used for the purpose of adding two n-bit binary numbers and requires 'n' full adders in the circuit

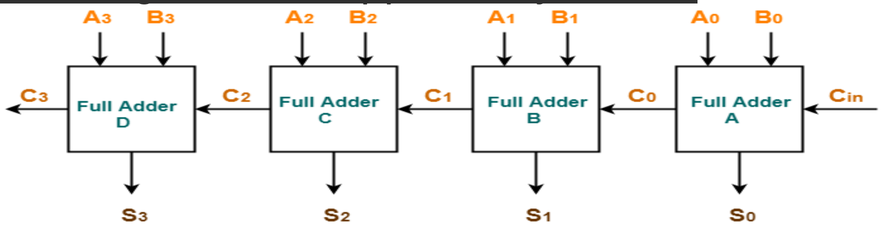
4-Bit Ripple Carry

- 4-bit ripple carry adder used for the purpose of adding two 4 bit binary numbers.
- in Mathematics, any two 4 bit binary numbers A3,A2,A1,A0 and B3,B2,B1,B0 will be added as

Logic Diagram



Working Of 4-bit Ripple Carry Adder-



4-bit Ripple Carry Adder

Suppose we want to add 4 bit binary numbers 0101 (A3A2A1A0) and 1010 (B3B2B1B0) using a ripple carry adder, this addition will be carried out as,

explained in the following stages-

Stage-01:

- When C_{in} will be fed as input to the full Adder A, it activates the full adder A.
- Then,at full adder A, $A_0 = 1$, $B_0 = 0$, $C_{in} = 0$.

The sum bit and carry bit produced as output by full adder A will be calculated by full adder A as

Calculation of S_0–

$S_0 = A_0 \oplus B_0 \oplus C_{in}$

$S_0 = 1 \oplus 0 \oplus 0$

S0=1

Calculation of C_0–

$C_0 = A_0B_0 \oplus B_0C_{in} \oplus C_{in}A_0$

$C_0 = 1.0 \oplus 0.0 \oplus 0.1$

$C_0 = 0 \oplus 0 \oplus 0$

C0=0

Stage-02:

- When C_0 will be fed as input to the full adder B by full adder A, it activates the full adder B.
- Then at full adder B, $A_1 = 0$, $B_1 = 1$, $C_0 = 0$.

Full adder B computes the sum bit and carry bit as-

Calculation of S_1–

$S_1 = A_1 \oplus B_1 \oplus C_0$

$S_1 = 0 \oplus 1 \oplus 0$

S1=1

Calculation of C_1–

$C_1 = A_1B_1 \oplus B_1C_0 \oplus C_0A_1$

$C_1 = 0.1 \oplus 1.0 \oplus 0.0$

$C_1 = 0 \oplus 0 \oplus 0$

$C_1 = 0$

C1=0

Stage-03:

- When C_1 will be fed as input to the full adder C by full adder B, it activates the full adder C.
- Then at full adder C, $A_2 = 1$, $B_2 = 0$, $C_1 = 0$.

Full adder C computes the sum bit and carry bit as-

Calculation of S_2–

$S_2 = A_2 \oplus B_2 \oplus C_1$

$S_2 = 1 \oplus 0 \oplus 0$

$S_2 = 1$

$\boxed{S2=1}$

Calculation of C_2–

$C_2 = A_2B_2 \oplus B_2C_1 \oplus C_1A_2$

$C_2 = 1.0 \oplus 0.0 \oplus 0.1$

$C_2 = 0 \oplus 0 \oplus 0$

$C_2 = 0$

Stage-04:

- When C_2 will be fed as input to the full adder D by full adder C, it activates the full adder D.
- Then at full adder D, $A_3 = 0$, $B_3 = 1$, $C_2 = 0$.

Full adder D computes the sum bit and carry bit as-

Calculation of S_3–

$S_3 = A_3 \oplus B_3 \oplus C_2$

$S_3 = 0 \oplus 1 \oplus 0$

$S_3 = 1$ $\boxed{S3=1}$

Calculation of C_3–

$C_3 = A_3B_3 \oplus B_3C_2 \oplus C_2A_3$

$C_3 = 0.1 \oplus 1.0 \oplus 0.0$

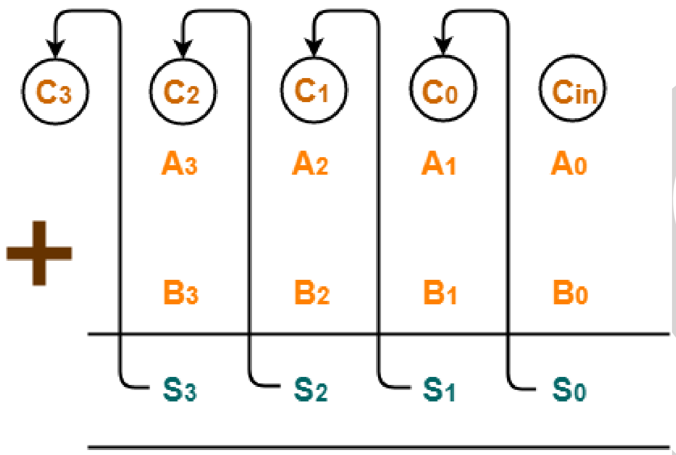
$C_3 = 0 \oplus 0 \oplus 0$

$C_3 = 0$

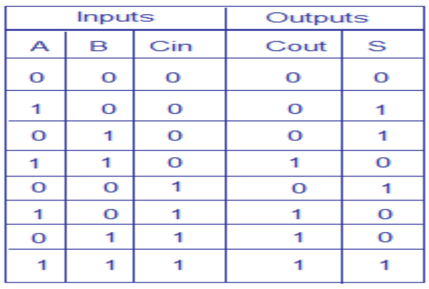
C3=0

Thus finally,

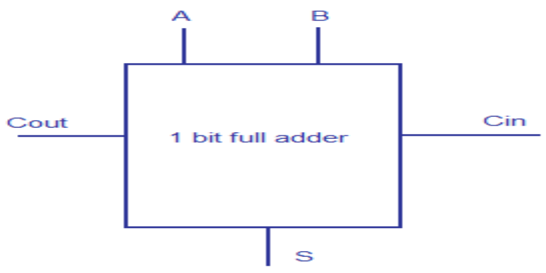
- Output Sum = $S_3S_2S_1S_0$ = 1111
- Output Carry = C_3 = 0



Adding two 4-bit Numbers



Inputs			Outputs	
A	B	Cin	Cout	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1



1 bit full adder truth table & schematic

MSB LSB
A=0 1 0 1

X1X2X3X4
B3 B2 B1 B0

Carry=A0B0+BCin+ACin
Carry=A1B1+B1C0+A1C0Ĺ

why Ripple carry adder is called so?

- In ripple carry adder, the carry out produced by each full adder as output serves as the carry in input for its next most significant full adder
- Since in ripple carry adder each carry bit ripple or waves into the next stage that's why it is called by the name ripple carry adder/

Limitations of Ripple carry adder

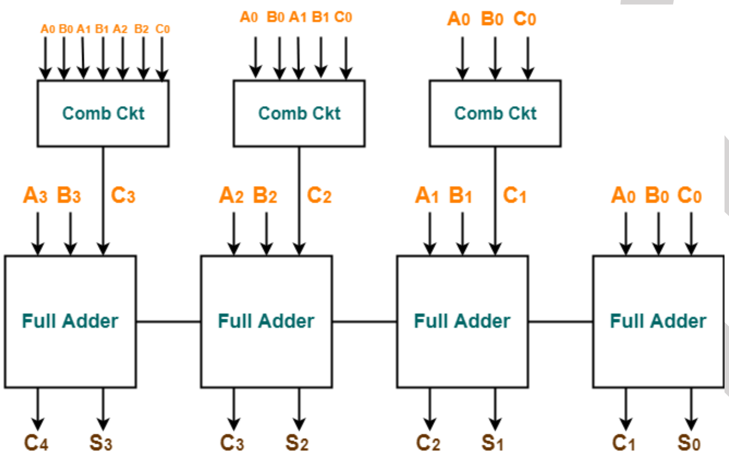
- ✓ Ripple carry adder does not allow all full adders to be used simultaneously and each full adder has to necessarily wait till the carry bit becomes available from it is adjacent less significant full adder
- ✓ This increases the propagation time and due to this reason ripple carry adder becomes extremely slow which is considered to be the biggest disadvantage of using ripple carry adder

Carry look Ahead adder

- In Ripple carry adder each full adder has to wait for its carry-in from its previous stage full adder to start its operation which causes unnecessary delay

- carry look ahead adder is an improved version of the ripple carry adder which generate the carry-in of each full adder simultaneously without causing any delay

Logic diagram for carry look ahead adder



Working of carry look Ahead adder

- The working of carry look ahead adder is based on the principle that the carry-in of any stage full adder is independent of the Carry bits generated during intermediate stages and only dependent of the following two parameters
  1. bits being added in the previous stages

  2. Carry provided in the beginning

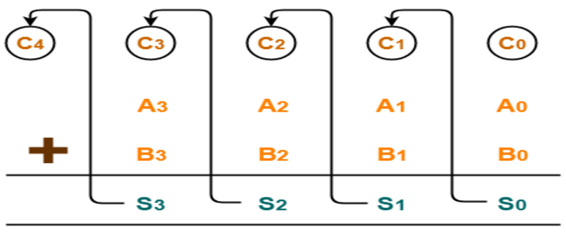
- Since, the above two parameters are always known, the Cin of any stage full adder can be evaluated at any instant of time
  Thus, a full adder need not wait until its carry-in is generated by it's previous stage full adder

Example:

4-Bit Carry Look Ahead Adder-

Suppose we want to add two 4-bit binary numbers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are to be added.

Mathematically, the two numbers will be added as-



Adding two 4-bit Numbers

From here, we have-

$C_1 = C_0 (A_0 \oplus B_0) + A_0B_0$

$C_2 = C_1 (A_1 \oplus B_1) + A_1B_1$

$C_3 = C_2 (A_2 \oplus B_2) + A_2B_2$

$C_4 = C_3 (A_3 \oplus B_3) + A_3B_3$

For simplicity, Let-

- $G_i = A_iB_i$ where G is called carry generator
- $P_i = A_i \oplus B_i$ where P is called carry propagator

Then, re-writing the above equations, we have-

$C_1 = C_0P_0 + G_0$ ………….. (1)

$C_2 = C_1P_1 + G_1$ ………….. (2)

$C_3 = C_2P_2 + G_2$ ………….. (3)

$C_4 = C_3P_3 + G_3$ ………….. (4)

Now,

- Clearly, C1, C2 and C3 are intermediate carry bits.
- So, let's remove C_1, C_2 and C_3 from RHS of every equation.
- Substituting (1) in (2), we get C_2 in terms of C_0.
- Then, substituting (2) in (3), we get C_3 in terms of C_0 and so on.

Finally, we have the following equations-

- $C_1 = C_0P_0 + G_0$
- $C_2 = C_0P_0P_1 + G_0P_1 + G_1$
- $C_3 = C_0P_0P_1P_2 + G_0P_1P_2 + G_1P_2 + G_2$
- $C_4 = C_0P_0P_1P_2P3 + G_0P_1P_2P_3 + G_1P_2P_3 + G_2P_3 + G_3$

These equations are important to remember.

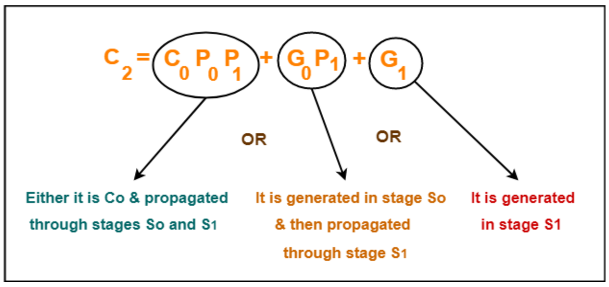
These equations show that the carry-in of any stage full adder depends only on-

- Bits being added in the previous stages
- Carry bit which was provided in the beginning

Trick To Memorize Above Equations-

As an example, let us consider the equation for generating carry bit C_2.

There are three possible reasons for generation of C_2 as depicted in the following picture-


$C_2 = C_0P_0P_1 + G_0P_1 + G_1$
OR
OR
Either it is Co & propagated through stages So and S1
It is generated in stage So & then propagated through stage S1
It is generated in stage S1

In the similar manner, we can write other equations as well very easily.

Implementation Of Carry Generator Circuits-

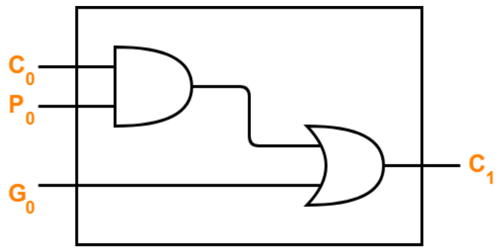
The above carry generator circuits are usually implemented as-

- Two level combinational circuits.
- Using AND and OR gates where gates are assumed to have any number of inputs.

Implementation Of C_1–

- The carry generator circuit for C1 is implemented as shown below.
- It requires 1 AND gate and 1 OR gate.

$C_1 = C_0P_0 + G_0$



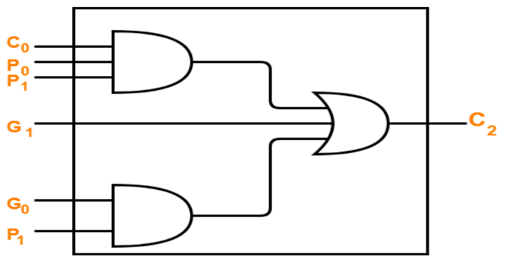
Implementation of C1

Implementation Of C_2–

- The carry generator circuit for C2 is implemented as shown below.

- It requires 2 AND gates and 1 OR gate.

$C_2 = C_0P_0P_1 + G_0P_1 + G_1$



Implementation of C2

<u>Implementation Of C_3 & C_4–</u>

Similarly, we implement C_3 and C_4.

- Implementation of C_3 uses 3 AND gates and 1 OR gate.
- Implementation of C_4 uses 4 AND gates and 1 OR gate.

Total number of gates required to implement carry generators (provided carry propagators P_i and carry generators G_i) are-

- Total number of AND gates required for addition of 4-bit numbers = 1 + 2 + 3 + 4 = 10.
- Total number of OR gates required for addition of 4-bit numbers = 1 + 1 + 1 + 1 = 4.

Advantages of Carry Look Ahead Adder-

The advantages of carry look ahead adder are-

- It generates the carry-in for each full adder simultaneously.
- It reduces the propagation delay.

Disadvantages of Carry Look Ahead Adder-

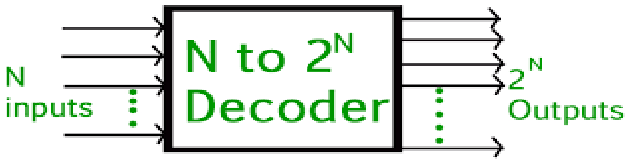
The disadvantages of carry look ahead adder are-

- It involves complex hardware
- It is costier since it involves complex hardware
- It gets more complicated as the number of bits increases

Decoders

- The name “Decoders” means to translate or Decode or coded information from one format into another, So digital decoder transforms a set of digital input signals into an equivalent decimal code at its output.

- A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines



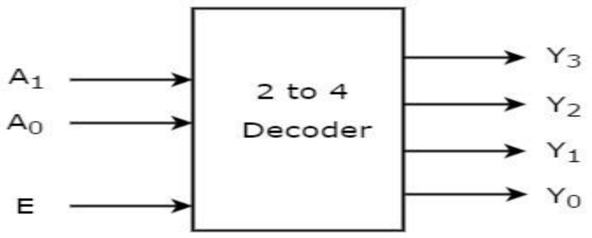
Binary Decoder

- Binary decoders are another type of Digital Logic device that has input of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines so a decoder that has a set of two or more bits will be defined as having an n-bit code and therefore it will be possible to represent 2^n possible values
- A binary Decoder converts coded inputs into coded outputs where the input and output codes are different and decoders are available to decode either a binary or BCD input pattern to typically a decimal output code
- Practical "binary Decoder' circuit include 2-to-4,3-to-8,and 4-to-16 line configurations
- Some binary Decoder have an additional input pin labelled enable that controls the outputs from the device this extra input allows the decoders output to be tuned "ON" OR 'OFF' as required output is only generated when the enable input has value 1 otherwise all the outputs are zero

2 to 4 Decoder

Let 2 to 4 Decoder has two input A1 and A0 and 4 outputs Y3 Y2 Y1 Y0

- The block diagram of 2 to 4 Decoder are



One of these four output will be '1' for each combination of inputs when enable ,E is '1'.

- The truth table of 2-to-4 decoder is shown below.

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

From truth table, we can't write the Boolean function for each output

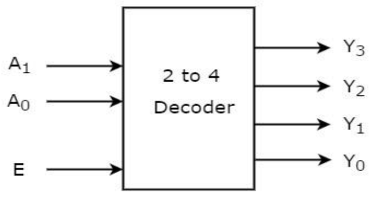
Y2=E.A1.A0

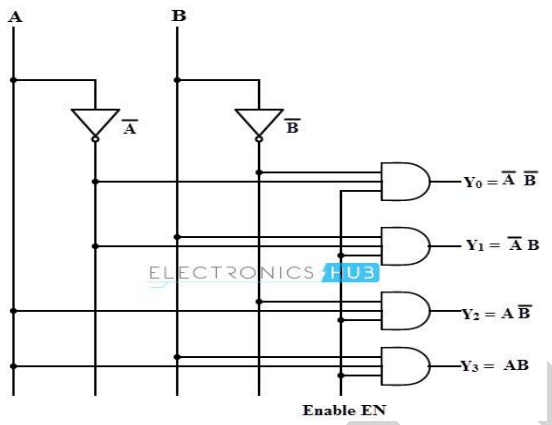
Y2=E.A1.A0’
Y1=E.A1’.A0
Y0=E.A1’.A0’

- Here each output is having one product term. so, there are4 product term in total

We can implement these 4product terms by using 4 AND gates having 3 inputs each and 2 inverters.

- The circuit diagram of 2-to-4 decoders is shown in the following figure:





- There for the outputs of 2-to-4 decoder are nothing but the min terms of 2 input variables A1 and A0 .when enable E is equal to 1
- If enable E is 0 then all the output of decoders will be equal to zero
- Similarly 3-to-8 decoders produce 8 min terms of 3 input variables A2, A1 and A0 and 4-TO-16 decoder produces 16 min term of 4 input variables A3, A2,A1,and

Implementation of higher order decoder

Now let as implement the following to higher order Decoder using lower order decoders
3 to 8 Decoder
4 to16 Decoder

3-to 8 decoders

- In this section let as implement 3 to 8 Decoder using 2 to 4 Decoder we know that 2 to 4 Decoder as 2 inputs A1 and A0 and four outputs Y3 and Y0.Whereas 3 to 8 Decoder has 3 input A2,A1, A0 and 8 outputs Y7-to-Y0
- we can find the number of lower order decoders required for implementing higher order Decoder using the following formula

required number of LOWER order

Decoders = M1/ M1

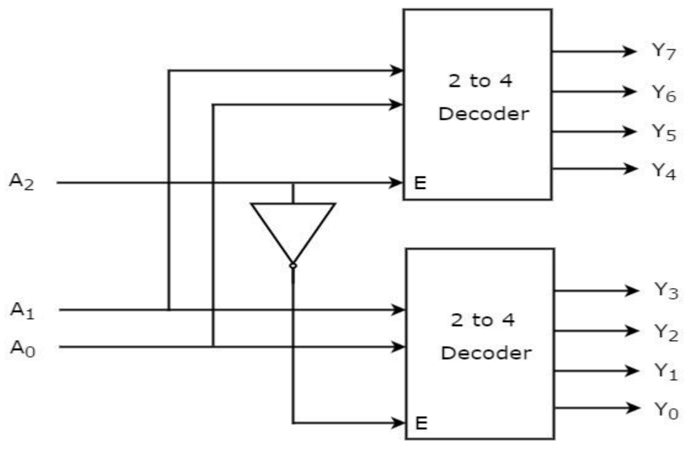
where,

M1-The number of outputs of the the lower order decoders
M2- is equal to the number of outputs of higher order decoder

Here, M1-4 and M2=8 substituted these two values in the above formulas
That is, required number of 2-to-4 decoder

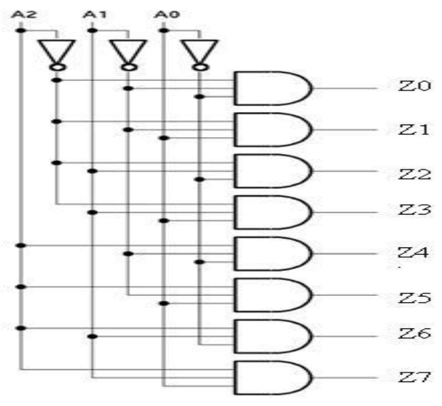
= 8/4
= 2
==

There for, we require to 2-to-4 decoder for implementing 3-to-8 de-coder using 2-to-4 decoder is,



- The parallel inputs A1 and A0 are applied to each 2 to 4 Decoder the complement of input A2 is connected to enable, E of lower 2-to-4 decoders in order to get the output Y3 to Y0. these are the lower for min terms. the input A2 is directly connected to enables E of upper 2 to 4 Decoder in order to get the output Y7 to y4 these are the higher 4 min terms

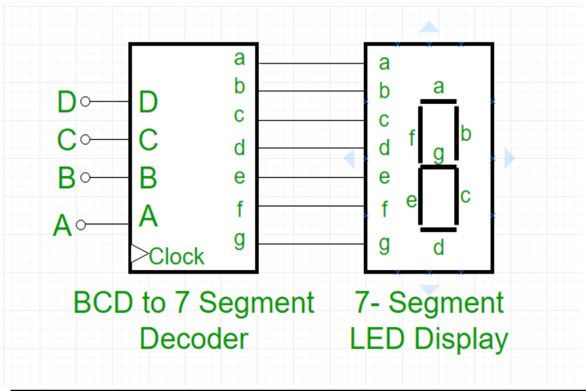
without using lower order 3 to 8-line decoder

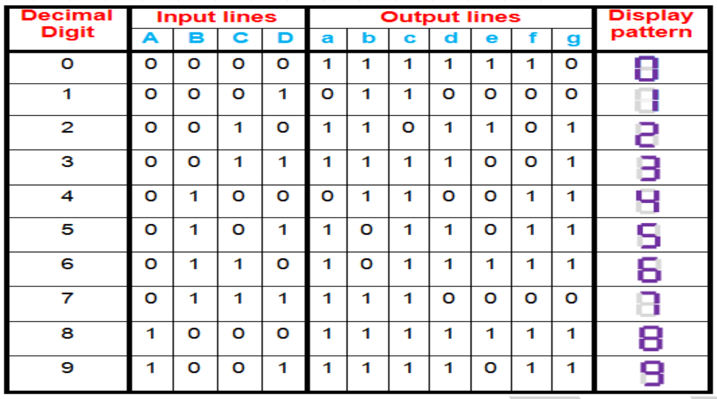


- the logic diagram of 3-to-8-line Decoder are above three data input A1 A0 and A2 are decoded into eight outputs, each output representing one of the combinations of the three binary input variables.
- the 3 inverters provide the complement of the inputs and each of the 8 AND Gates generate one of the binary combination.

a input variable to represent binary number and the outputs represent the 8 digits of the octal number system however a 3 to 8 line Decoder can be used for decoding any three bit code to provide 8 outputs one for each combination of the binary code

BCD to 7 segment decoder





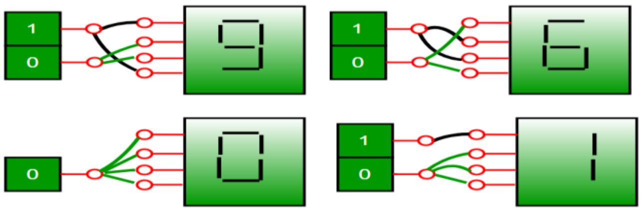
Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9

- A BCD to 7 segment Decoder is a combinational logic circuits that accept a decimal digit in BCD input and generate appropriate outputs for the segment to display the input decimal digit
- That is, in binary coded decimal BCD encoding scheme each of the decimal numbers (0-9) is represented by its equivalent binary pattern (which is generally of 4 bits)

Whereas seven segment display is an electronic device which consists of 7 LCD arranged in some definite pattern which is used to display hexadecimal number

seven segment display does not work by directly supplying voltage to different segment of LED's first, our decimal number is changes to its BCD equivalent signal then BCD to 7 segment Decoder convert that signal to the form which is led to 7 segment display

This BCD to 7 segment Decoder has 4 input lines a b c and d and 7 output lines A B C D E F G this output is given to 7 segment LED display which displays the decimal number depending upon inputs



Application to study in seven segments

- digits in calculators
- clock
- various measuring instruments
- digital watches
- digital counters

Digital logic

- digital logic is the manipulation of binary values through printed circuit board technology that uses circuits and logic gates to construct the implementation of computer operations

- digital logic is common part of electrical engineering and design course
- the main component of digital logic consists of five different logic gates
  - AND
  - OR
  - XOR
  - NAND
  - NOR

Positive and negative logic

There are two types of representations used in digital systems. the positive logic and the negative logic representation

In positive logic representation bit one represents logic high and bit 0 representation logic low as shown in figure

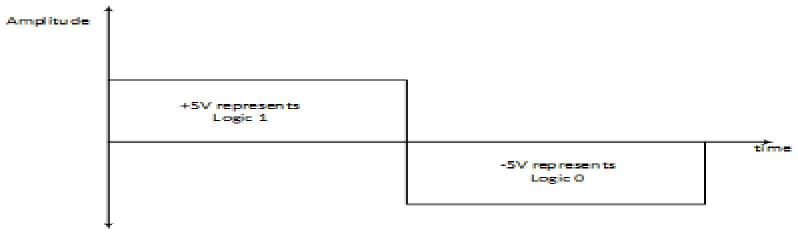


Fig. 2 a. Positive logic representation

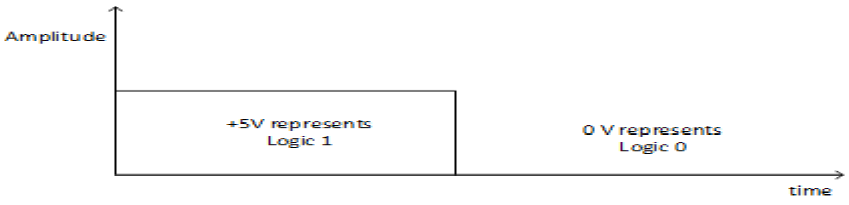


Fig. 2 b. Positive logic representation

positive logic representation

high is represented by + 5 volts and low is represented by - 5 volts Or 0 volts

In negative logic representation bit 1 represents logic row and bit 0 represents logic high as show in figure

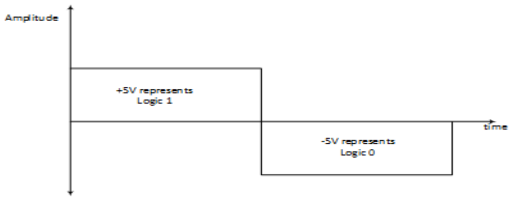


Fig. 2 a. Positive logic representation

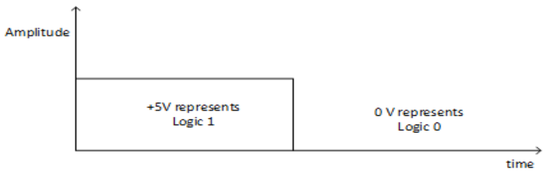


Fig. 2 b. Positive logic representation

In terms of voltage level,bit can be represented as +5 vplt and bit 0 can be represented as 0v or -5volt

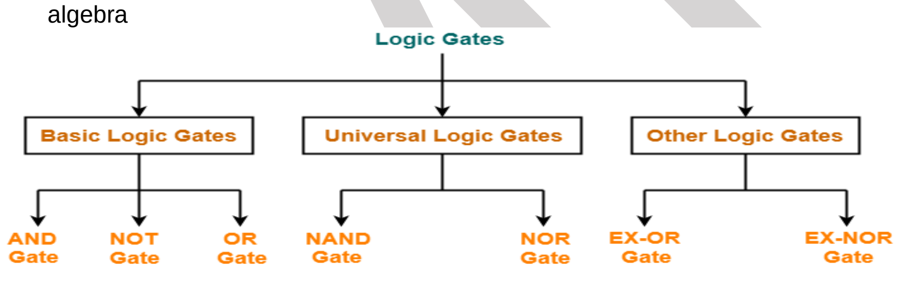
Logic Gates

binary logical deals with binary variables with operations that assume a logical meaning

- it is used to describe in Algebra or tabular from that is the manipulation and processing of binary information
- the manipulation of binary information is done by logic circuit is called Gates
- A variety of logic gates are commonly used in digital computer systems
- Each gate has a distinct graphic symbol and its operation can be described by means of an algebraic expression
- The input output relationship of the binary variables for each Gate can be represented in tabular form by a truth tables
- At any given moment, every terminal is one of two binary conditional law (0) OR high (1) represented by different voltage levels
- there are seven basic logic gates
  AND
  OR
  XOR
  NOT
  NAND
  NOR
  XNOR

Basic logic gates

- A gate is simply on electronic circuit which operates on one or more signal to produce an output signal
- Gates are digital (two state) circuit because the input and output signals are that low voltage (denotes 0) or High Voltage (denotes 1) gates are often called logical circuit because they can be analyzed with Boolean algebra



Types of Logic Gates

there are three types of logical Gates
1 inverters (not gate)
2 OR gate
3 And Gate

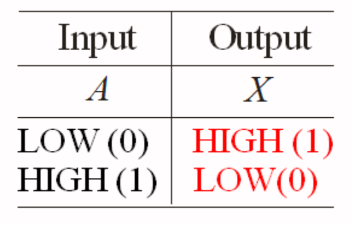
Boolean Algebra

A Boolean algebra is an algebra non-empty set empty that is
set, operations, elements set A to Z, 0 to 9
operation: a: b,a+b,……
elements:0,1………
(consists of less than or equal to 2)

inverter (not gate)

- an inverter is a gate with only one input signal and one output signals the output state is always the opposite of the input state
- an inverter is also called a not gate because the output is not the same as the input output is sometimes called the complement of the input

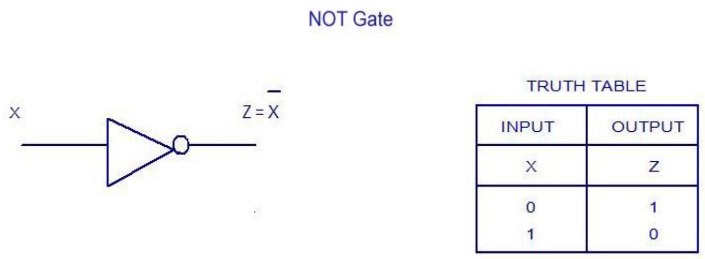
Truth table for not gate



Input	Output
A	*X*
LOW (0)	HIGH (1)
HIGH (1)	LOW(0)

- A low input that is zero produces high output that is one and vice versa the symbol for inverter is given in figure

alternative to table for not gate

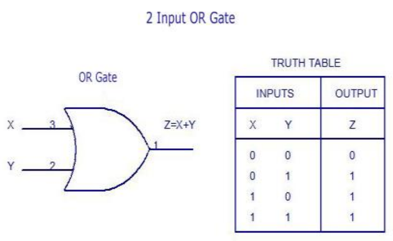


INPUT	OUTPUT
X	Z
0	1
1	0

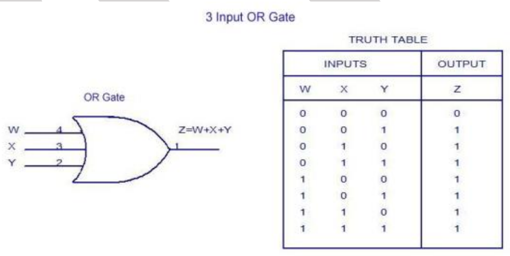
not gate symbol

OR gate

- The OR gate has two or more input signals but only one output signals if any of the input signal is 1(high) the output signal is one(high)
- If all inputs are zero then output is also zero if one or more inputs are 1, the output is 1, an OR Gates can have as many input as desired no matter how many inputs are there the action of OR is a same one or more once (high)inputs produce output as one



INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1



INPUTS			OUTPUT
W	X	Y	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

The symbol for August is given above

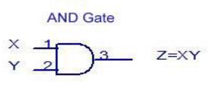
- Two input or gate
- Three input OR GATE
- FOUR input OR GATE

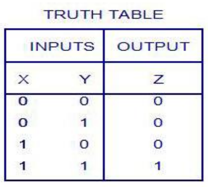
AND gate

- The AND gate can have two or more than two input signals and produce an output signal when all the input is 1 otherwise output is 0 only

- If any of the input is 0 the output is 0, to obtain output as 1,all the input must be 1.an AND gate can have as many inputs as desired.

2 Input AND Gate

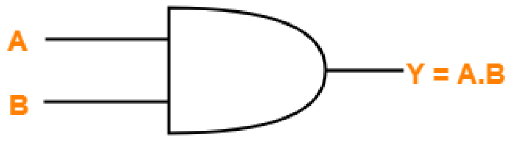




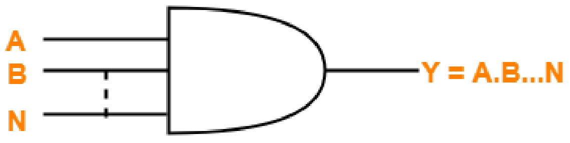
TRUTH TABLE

INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

The symbol for AND GATE



2-Input AND Gate



N-Input AND Gate

a) 2-input AND GATE
b) 3-input AND GATE
c) 4-input AND GATE

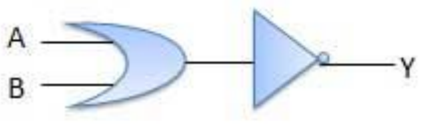
More about logic gates

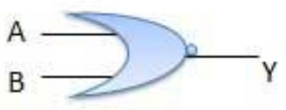
we have covered three basic logic gates NOT, OR,AND so far, But there are some more logic gates also which are derived from three basic Gates that is(AND,OR and NOT) these Gates are more popular than NOT,OR ,AND, and are widely used in industry. these sections introduce NOR, NAND, XOR, XNOR GATES

NOR Gate

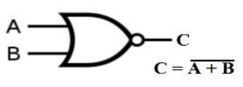
The NOR gate has two or more input signals but only one output signals if all the inputs are 0 then output will be 0 NOR gate is nothing but inverted OR gate. The NOR GATE can have as many inputs as desired. No matter how many inputs are there, the action of nor gate is the same All (0) zero input produce output as one (1)

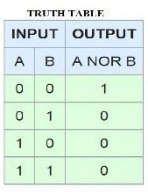
a) Logic meaning of NOR GATE
b) 2 -input nor get
c) 3- input Nor gate
d) 4- input NOr gate





NOR GATE



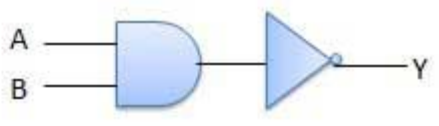


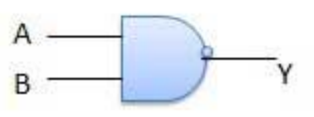
TRUTH TABLE

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

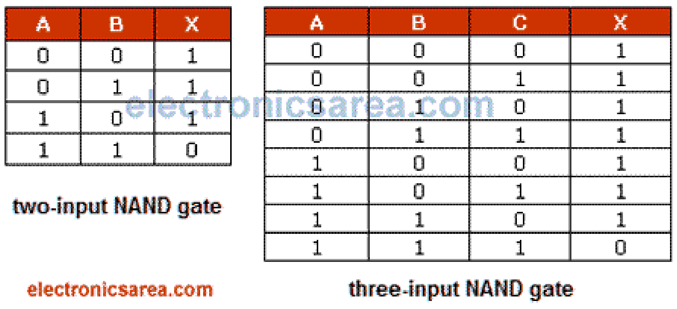
ProjectIoT123.com

NAND Gate





- The NAND gate has two or more input signals but only one output signal if all of the inputs are 1 then the output produced is 0(low)
- NAND gate is invented AND Gate for all one inputs it produces zero output otherwise for any other input combination is produces a 1 output. NAND Gate can also have as many inputs desired NAND action is illustrated in following Truth table



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

two-input NAND gate

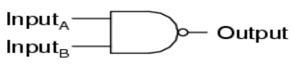
electronicsarea.com

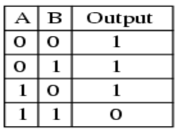
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

three-input NAND gate

Logical meaning of NAND gate

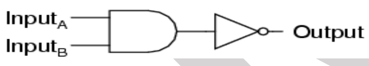
2-input NAND gate





A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

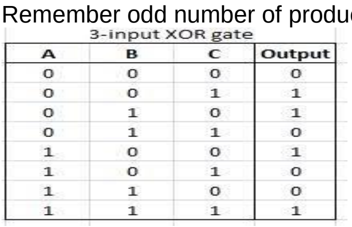
Equivalent gate circuit



XOR Gate (exclusive OR gate)

The XOR gate can also have two or more inputs but produces one output signals exclusive OR gate is different from OR gate. OR gate produces output 1 for any input combination having one or more is. but XOR Gate produces output 1 or only those input combinations that have odd number is
In Boolean Algebra (+)SIGN stands for XOR operation this A XOR B can be written as A(+)B

Remember odd number of produces output 1

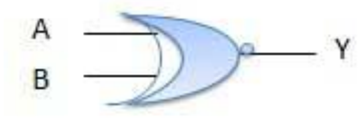


3-input XOR gate

A	B	C	Output
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

2 input XOR gate
3 input XOR gate
4input XOR gate
XOR addition can be summarized as,

XNOR Gate (Exclusive Nor gate)



The ex-nor gate is logically equivalent to an inverted XOR Gate
That is XOR gate followed by a not gate thus XNOR produces one output when the input combination has even number of ones

the bubble small circle on the output of NAND NOR XNOR gates represents complementation

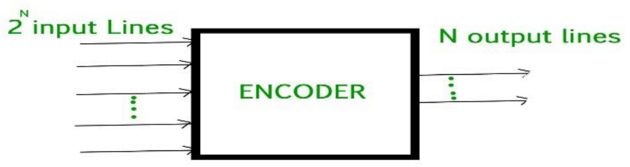
Now that we are familiar with logic gates we can use them in designing logic circuit solution the given Boolean expression can also be written as follows if ABC is equal to a into B + a + b into a boring to see

INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	EXOR	EXNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

ENCODER

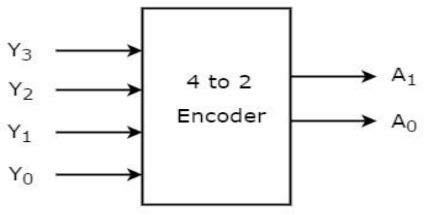
- An encoder is a combinational circuit that perform the reverse operation of decoders.
- It has maximum of 2^n input lines and n output lines.
- It will produce a binary code equivalent to the input, which is active high there for the encoder

2^n input lines with n bits
it is optional to represent the enable signal in encoders



4-to-2 encoders

- Let 4 to 2 encoder has 4 input Y3, Y2, Y1, Y 0 and 2 outputs A1&AO
- The block diagram of 4-to-2 encoder are
- At any time, only one of these four inputs can be one in order to get the respective binary code at the outputs
- The truth table of 4-to-2 encoder is



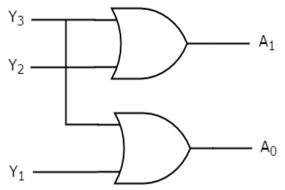
INPUTS				OUTPUTS	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

INPUTS										OUTPUTS			
Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	A3	A2	A1	A0
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

- From truth table, we can write the Boolean function for each output as

A1=Y3+Y2
A0=Y3+Y1

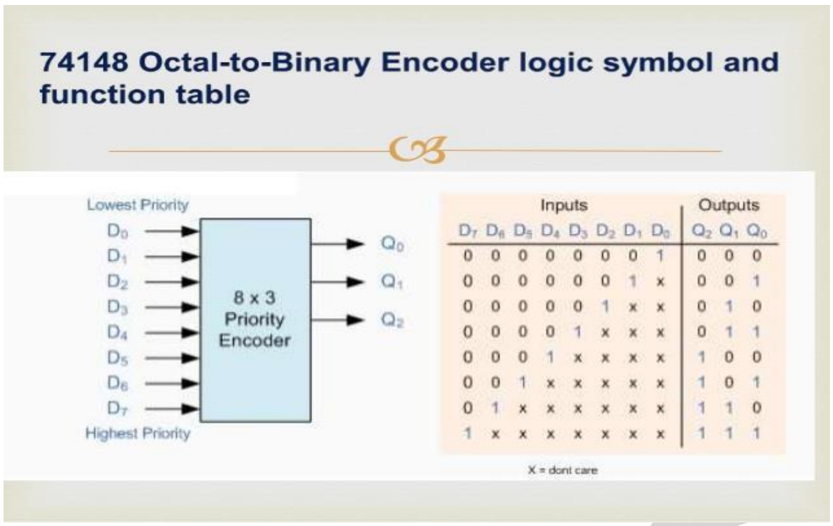
- we can implement the above 2 Boolean functions by using 2 input OR gates the circuit diagram of 4-to-2 encoder is



- The above circuit diagram contains two OR gate. These OR gates encode the four inputs with two bits

Octal to binary encoders(8-to-3)

- octal to binary encoder has 8 input Y7 -to-Y0 and three outputs A2,A1&A0
- octal to binary encoder is nothing but 8 to 3 encoders
- the block diagram of 8-to-3 encoder are


Lowest Priority
D0
D1
D2
D3
D4
D5
D6
D7
Highest Priority
8 x 3 Priority Encoder
Q0
Q1
Q2

74148 Octal-to-Binary Encoder logic symbol and function table

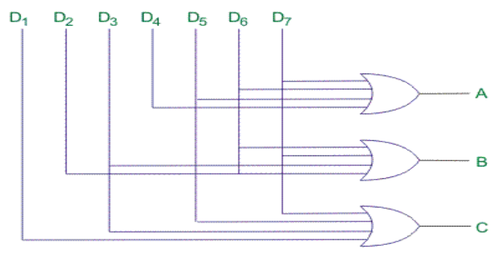
Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = dont care

- At any time, only one of these 8 input be ‘1’ in order to get the respective binary code
- From truth table, we can write the Boolean function for each output as

  Q2=D7+D6+D5+D4
  Q1=D7+D6+D3+D2
  Q0=D7+D5+D3+D1

- we can implement the above Boolean functions by using for input or Gates the circuit diagram of 8-to-3 binary encoder are



- The above circuit diagram contains three four input OR gate these OR gates encode the 8 input with 3 bits

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 1 or when all inputs are 0
- If more than one input is active high, then the encoder producer an output, which many not be the correct codes

For example

- If both y3 and Y6 are one, then the encoder produces 111 at the output this is neither equivalent code corresponding to y3, when it is one nor the equivalent code corresponding to y6, when it is one (1)

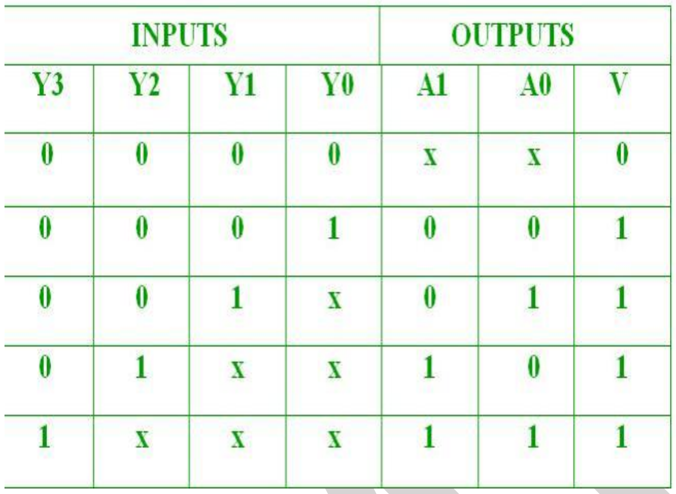
So, to overcome these difficulties, we should assign priorities to each input of encoder. then, the output of encoder will be the binary code corresponding to the active high input which has higher priority. This encoder is called as priority encoder

Priority encoder

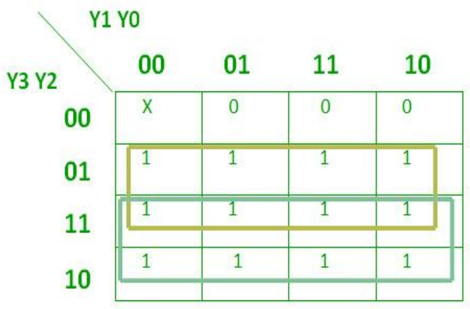
- A 4-to-2 priority encoder has four inputs Y3, Y2,Y1&Y0 and 2 outputs A1&A0.

Here, the input Y3 has the higher priority where are the input y0 has the lowest priority

- In this case, even if more than one input is one at the same time, the output will be the binary code corresponding to the input, which is having higher priority
- we considered one or more input Vin order to know, whether the code available at output is valid or not
- If at least one input of the encoder is once, then the code available at outputs is a valid one in this case the output, V will be equal to 1
- If all the inputs of encoder are zero, then the code available at outputs is not a valid one. in this case the output V will be equal to zero
- the truth table of 4-to-2 priority encoder are

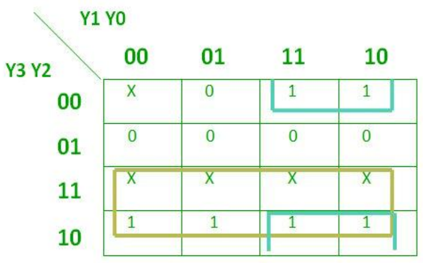


INPUTS				OUTPUTS		
Y3	Y2	Y1	Y0	A1	A0	V
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1



Y3 Y2 \ Y1 Y0	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

A1=Y3 + Y2



Y3 Y2 \ Y1 Y0	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	X	X	X	X
10	1	1	1	1

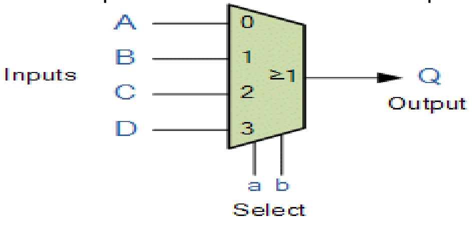
A0=Y3 + Y2’ Y1

MULTIPLEXERS

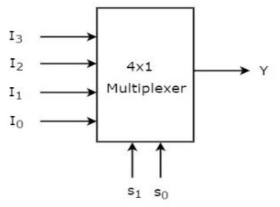
- The multiplexers shortened to ‘MUX’ or” MPX” is a combination logical circuit designed to switch one of the several input lines through to a single common output line by the application of a control signal

That is, A multiplexer is a combinational circuit / device that receive binary information from one of 2^n input data lines and direct it into a single output line.

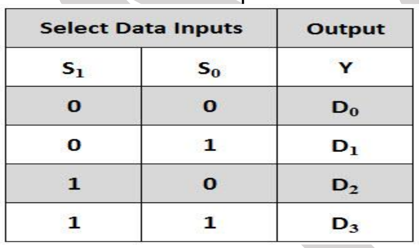
- The selection of a particular input data line for the output is determined by a set of selection inputs
- A 2^n-to-1 multiplexer has 2^n input data lines and “n” input selection lines, which are used to select which input lines to send it to the output
- Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth
- A Multiplexers is also called a data selector / “MUX”,
- Multiplexers can also be used to implement Boolean functions of multiple variables



4*1 multiplexer

- 4*1 Multiplexers has four data input I3,I2,I1 & I0 , 2 selection line S1&S0 and one output Y
- the block diagram of 4*1 multiplexer is
- 

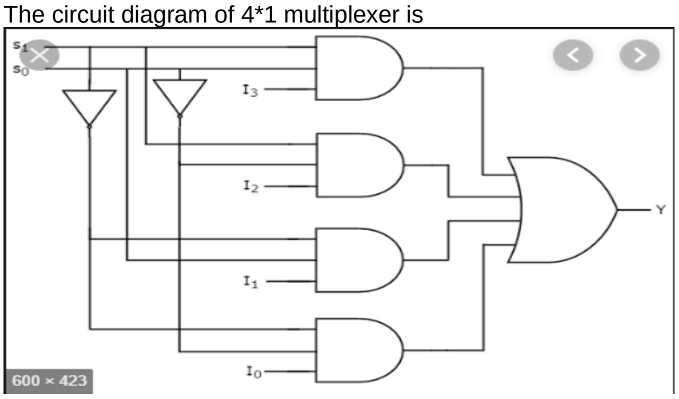
- one of these four input will be connected to the output based on the combination of input present at these 2 selection lines
- Truth table of 4*1 multiplexer is



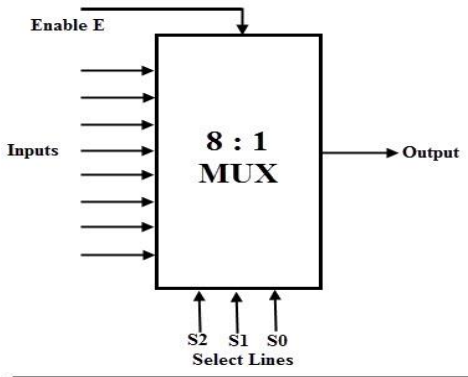
Select Data Inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

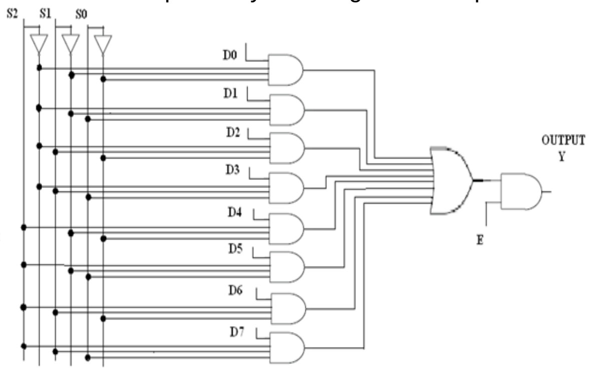
- From truth table, we can directly write the Boolean functions for output Y as Y=S1’S0’D0+S1’S0D1+S1S0’D2+S1S0D3
- We can implement this Boolean function using Inverters, AND gates &OR gates.

The circuit diagram of 4*1 multiplexer is

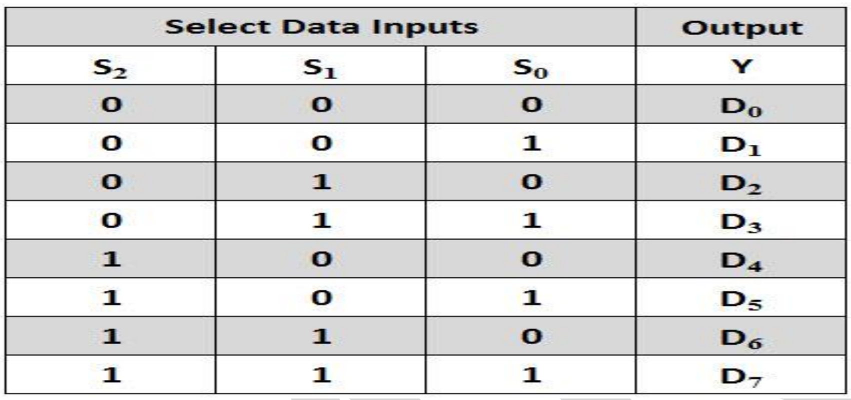


- Similarly, we can implement 8 *1 Multiplexer and 16*1 multiplexer by following the same procedure





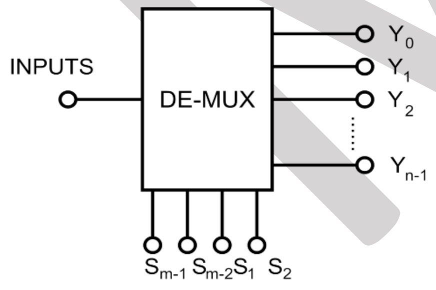
$$Y = E.[\overline{S2}\,\overline{S1}\,\overline{S0}\,D0 + \overline{S2}\,\overline{S1}\,S0\,D1 + \overline{S2}\,S1\,\overline{S0}\,D2 + \overline{S2}\,S1\,S0\,D3 + S2\,\overline{S1}\,\overline{S0}\,D4 + S2\,\overline{S1}\,S0\,D5 + S2\,S1\,\overline{S0}\,D6 + S2\,S1\,S0\,D7]$$



Select Data Inputs			Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

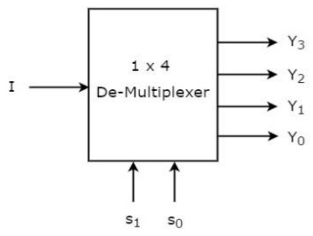
De-multiplexer

- De-multiplexer is a combination circuit that perform the reverse operation of multiplexers
- It has single input 'n' selection lines and maximum of 2^n outputs
- The input will be connected to one of these outputs based on the value of selection lines.
  since there are n selection lines there will be 2^n possible combinations of zeros and ones so each combination can select only one output
- De-multiplexer is also called as De-MUX



1*4 De-multiplexer

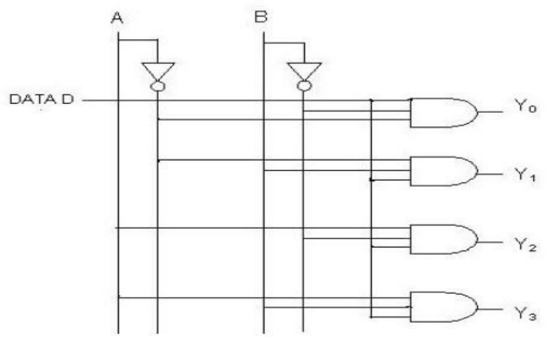
- 1*4 De-multiplexers has one input I, 2 selection lines s1&s0 and 4 outputs Y3,Y2,Y1&Y0
- The block diagram of 1*4 De-multiplexers is



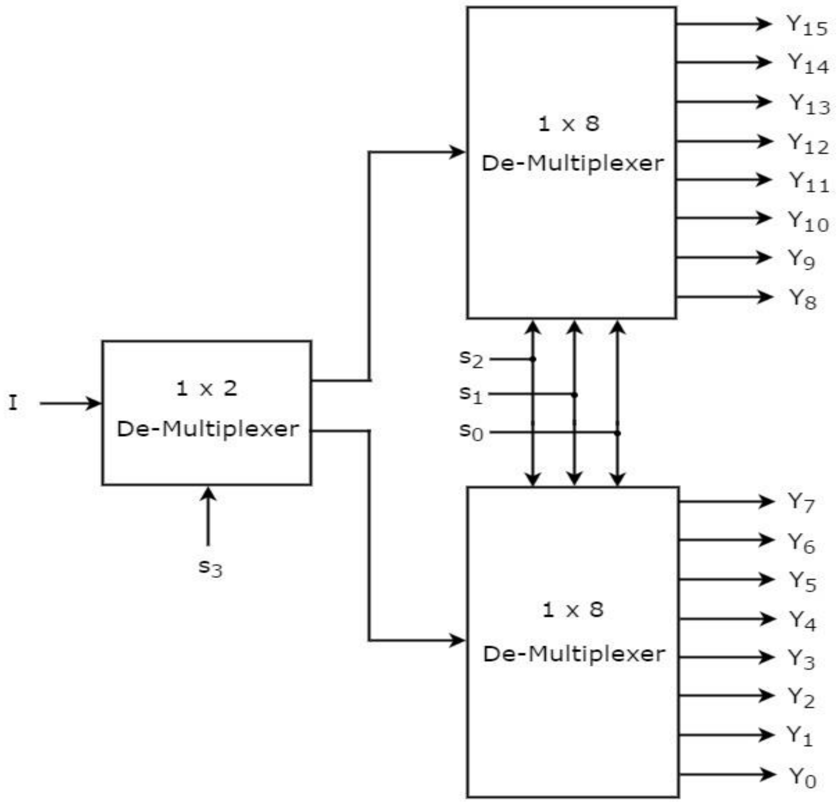
- The single input 'I' will be connected to one of the 4 outputs Y3 to Y0 based on the values of selection lines S1&S0
- The truth table of 1*4 De-MUX is

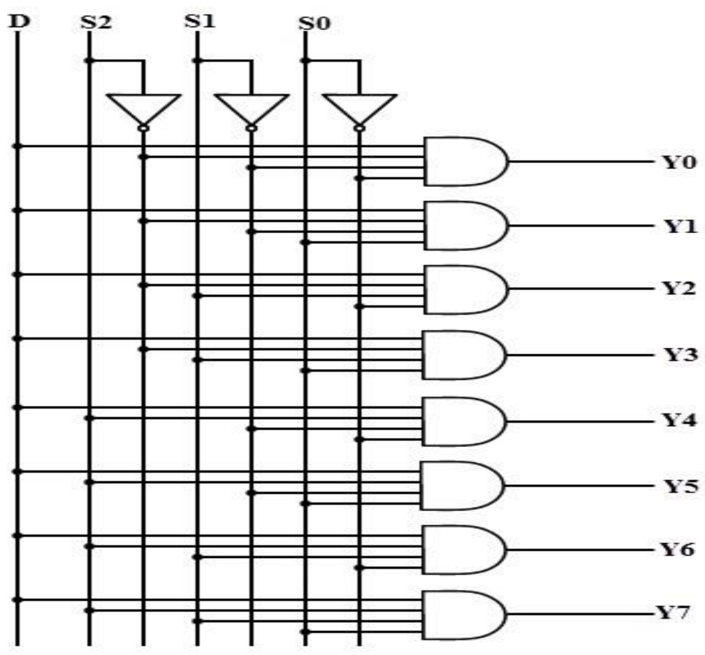
Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

- From the above truth table ,we can directly write the Boolean function for each output as,
  Y3=S1S0D
  Y2=S1S0'D
  Y1=S1'S0D
  Y0=S1'S0'D
- We can implement these Boolean functions using inverters and 3-inputs AND gates
- The circuit diagram of 1*4 De-MUX is



- Similarly, we can Implement 1*8 De-MUX and 1*16 D-MUX by following the same procedure




D
S2
S1
S0
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7