Digital CMOS Circuits

Dr. Yashika Gaidhani
Assistant Professor
YCCE Nagpur

Unit 6: VLSI System Components

- Data path VLSI System Components:
- Comparators,
- Barrel shifters,
- Multiplexers,
- Binary Decoders,
- Equality Detectors and
- Comparators,
- Priority Encoders,
- Shift and Rotation Operations,
- Bit Adder Circuits,
- Multipliers

Comparators

- Binary or digital comparator can be constructed using AND, NOR and NOT gates to compare digital signals present at their input terminals and produce an output depending upon condition of those inputs.
- There are two main types of **Digital Comparator** available and these are.
    1. Identity Comparator: is a digital comparator with only one output terminal for when A = B, either A = B = 1 (HIGH) or A = B = 0 (LOW)
    2. Magnitude Comparator: is a digital comparator which has three output terminals, as for equality i.e. A = B, greater than, A > B and less than A < B

- Purpose of a **Digital Comparator** is to compare a set of variables or unknown numbers, for example A (A1, A2, A3, …. An, etc) against that of a constant or unknown value such as B (B1, B2, B3, …. Bn, etc) and produce an output condition depending upon result of comparison.

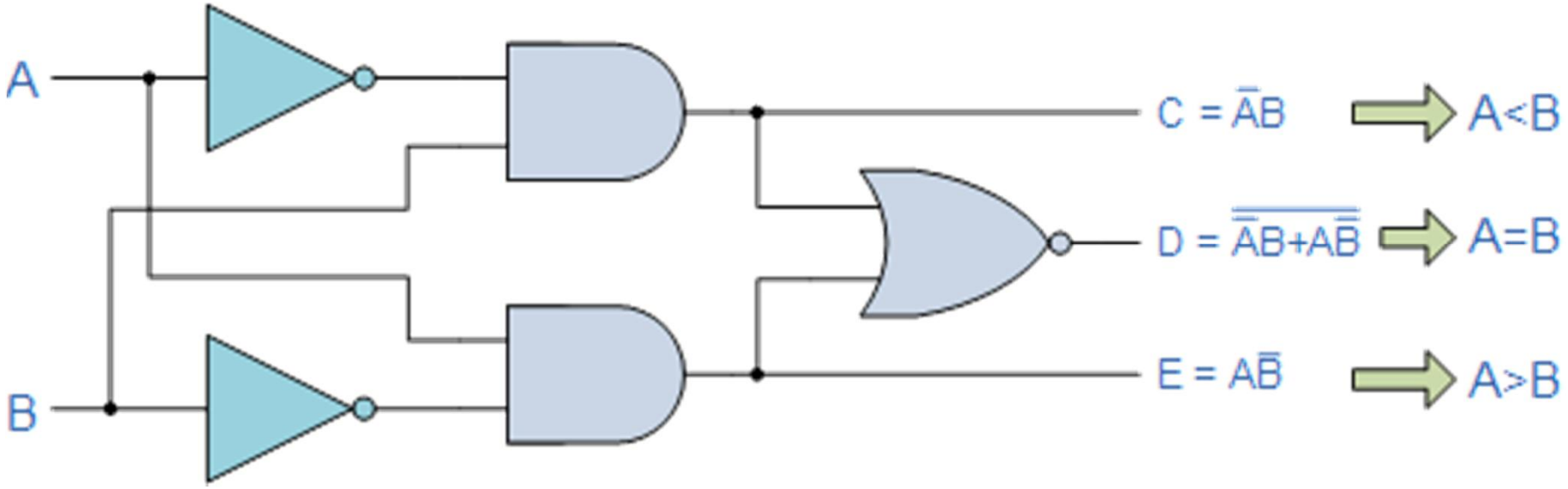
- For example, a magnitude comparator of two 1-bits, (A and B) inputs would produce following three output conditions when compared to each other.

A>B, A=B, A<B

•Which means: A is greater than B, A is equal to B, or A is less than B

•This is useful two variables are to be compared and want to produce an output when any of above three conditions are achieved.

1-bit Digital Comparator Circuit



- Operation of a 1-bit digital comparator is given in Truth Table.
- Two features about comparator from truth table are.
    1. Circuit does not distinguish between either two "0" or two "1"'s as an output A = B is produced when they are both equal, either A = B = "0" or A = B = "1".
    2. Output condition for A = B is same as logic gate, XNOR function as Q = A B

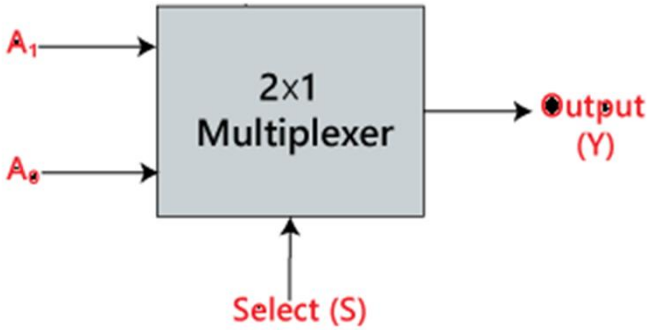
Truth Table

Inputs		Outputs		
B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Multiplexers

- Multiplexer acts as a multiple-input and single-output switch.
- It is a combinational circuit which have many data input (2^n) lines and single output line depending on control or select inputs (n).
- For N input lines, log n (base2) selection lines, for 2^n input lines, n selection lines are required.
- Multiplexers are also known as **"Data n selector, parallel to serial convertor, many to one circuit, universal logic circuit"**.

2×1 Multiplexer:

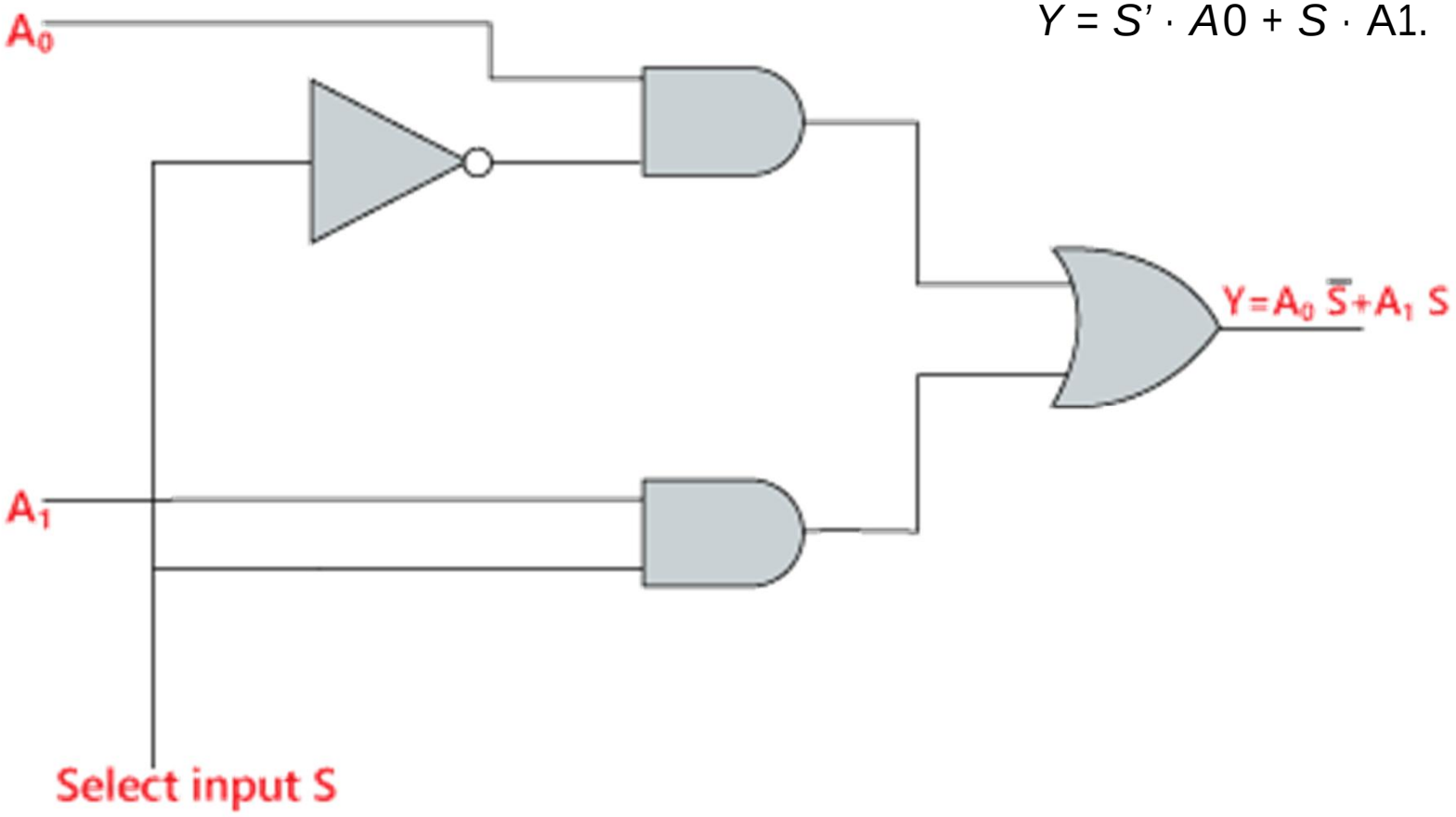


Truth Table:

INPUTS	Output
S_0	Y
0	A_0
1	A_1

- Multiplexers are key components in CMOS memory elements and data manipulation structures.
- A *multiplexer* chooses output from among several inputs based on a select signal.
- A 2-input, or 2:1 multiplexer, chooses input *A*0 when select is 0 and input *A*1 when select is 1.
- Logic function is $Y = S' \cdot A0 + S \cdot A1$.

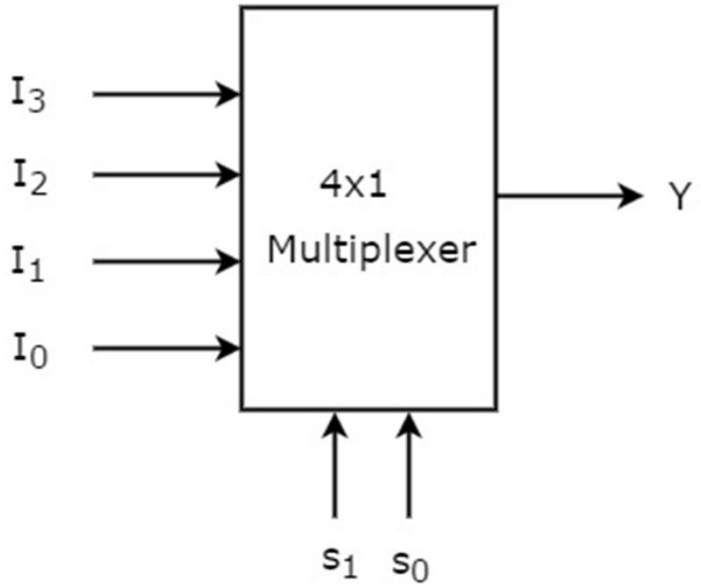
$Y = S' \cdot A0 + S \cdot A1.$



CMOS implementation of 2 : 1 MUX

4×1 Multiplexer

- 4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & s_0 and one output Y.
- **Block diagram** of 4x1 Multiplexer is shown in following figure.



- One of these 4 inputs will be connected to output based on combination of inputs present at these two selection lines.
- **Truth table** of 4x1 Multiplexer is shown below.

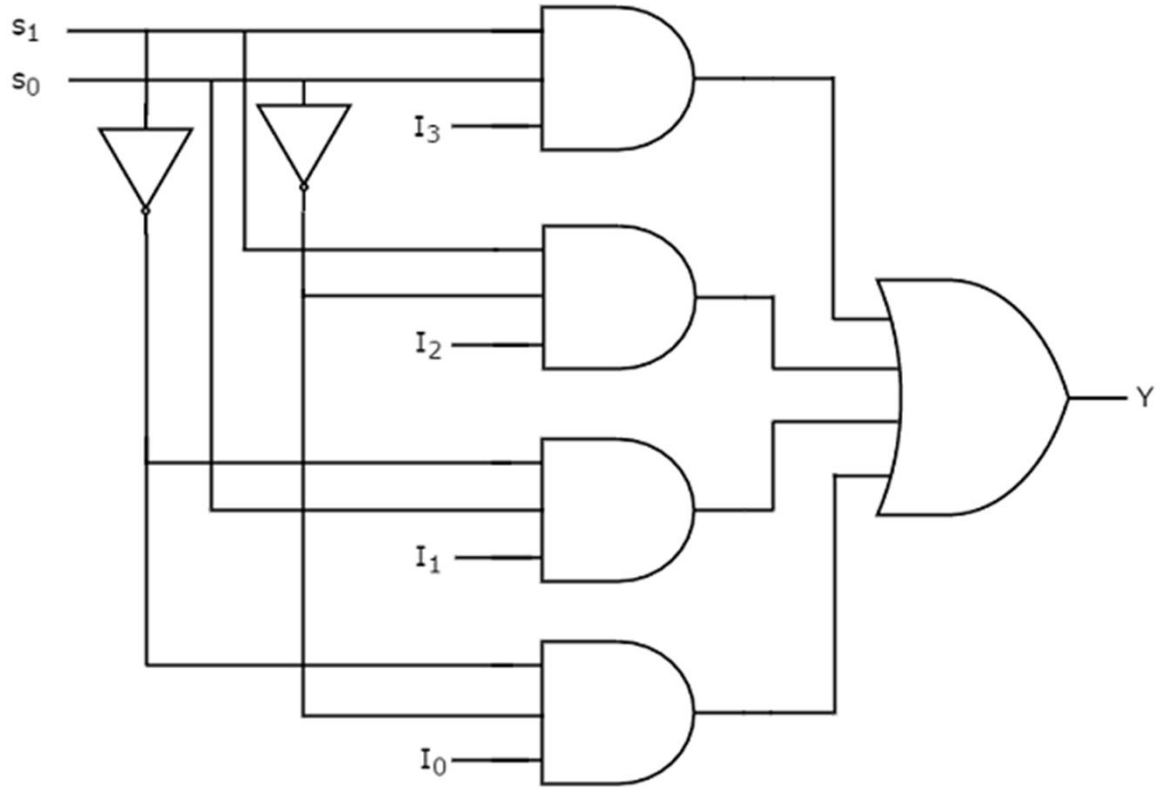
Truth Table

Selection Lines		Output
S1	**S0**	**Y**
0	0	I0
0	1	I1
1	0	I2
1	1	I3

- Logical expression from Truth table for term Y is as follows:

$$Y=S_1' S_0' I_0+S_1' S_0 I_1+S_1 S_0' I_2+S_1 S_0 I_3$$

- Implementation of this Boolean function using Inverters, AND gates & OR gate or **circuit diagram** of 4x1 multiplexer is shown below.

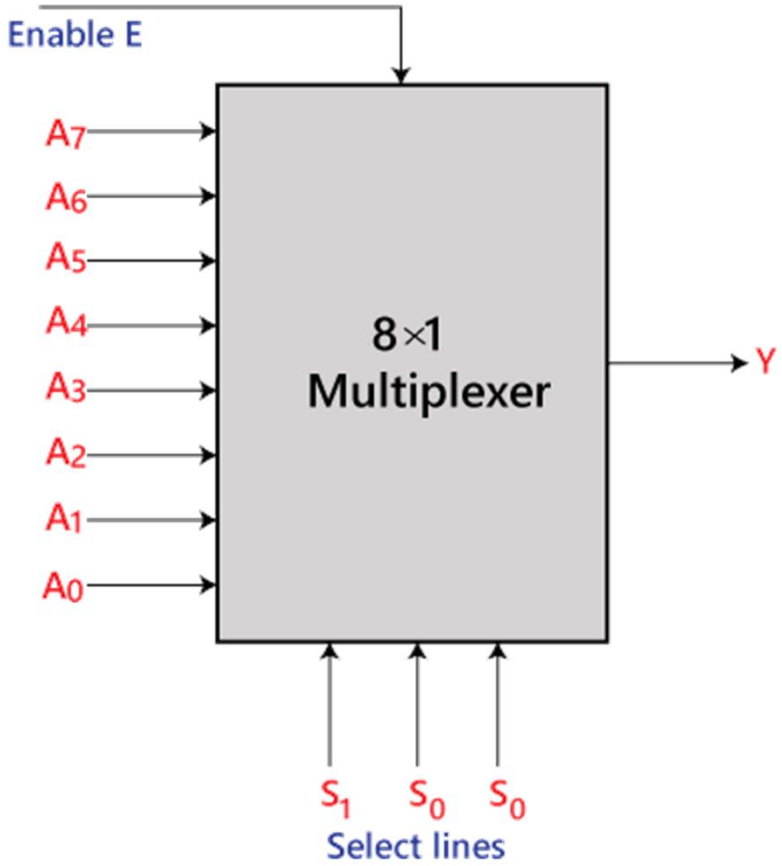


CMOS implementation of 4 : 1 MUX

8×1 Multiplexer

- Total eight inputs, i.e., A0, A1, A2, A3, A4, A5, A6, and A7,
- 3 selection lines, i.e., S0, S1and S2 and
- single output, i.e., Y.
- On basis of combination of inputs, present at selection lines S0, S1, and S2, one of these 8 inputs are connected to output.
- Block diagram and truth table are given as

Block Diagram:

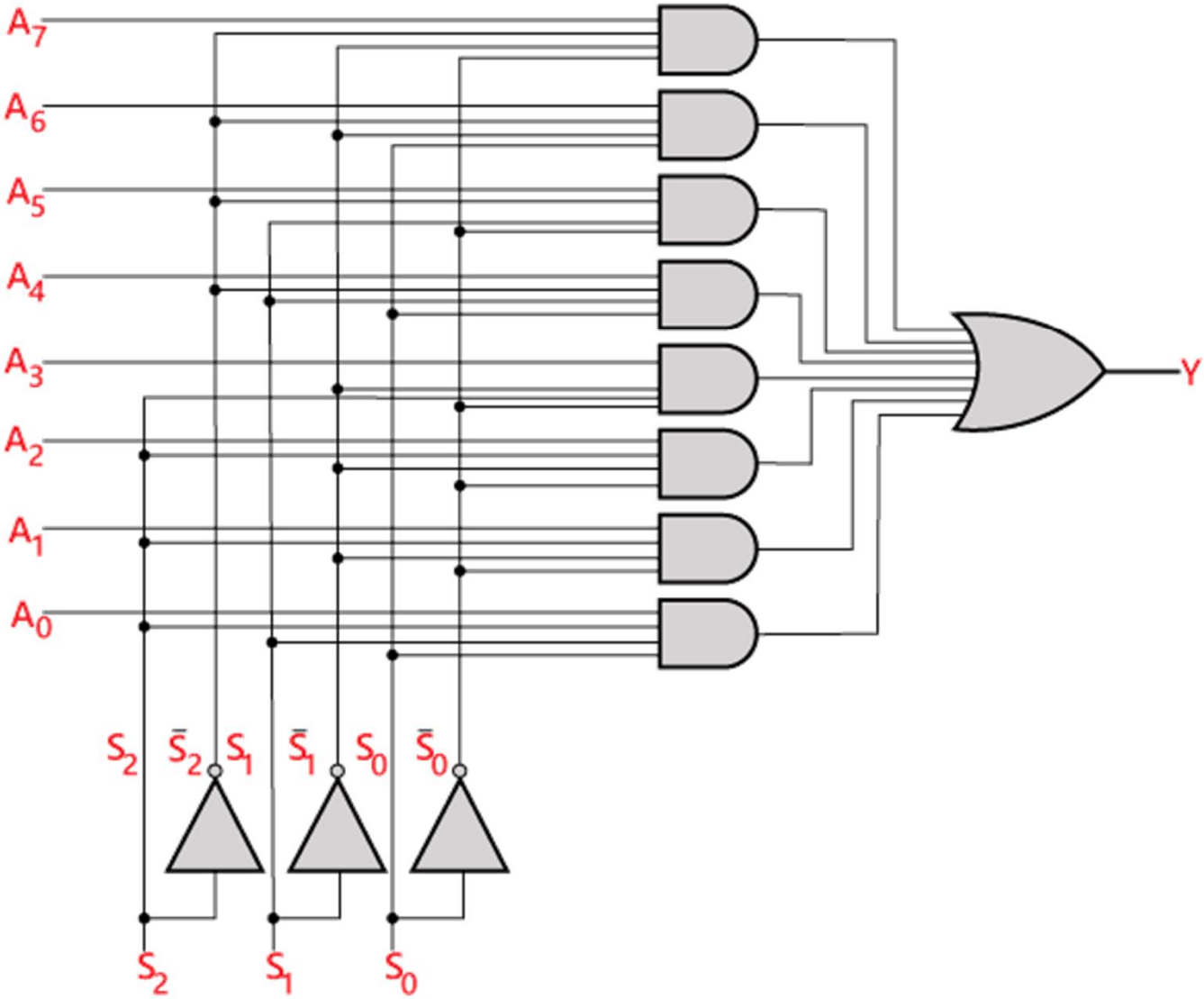


Truth Table:

INPUTS			Output
S_2	S_1	S_0	Y
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7

Logical expression for output Y is as follows:

Y=S0'.S1'.S2'.A0+S0.S1'.S2'.A1+S0'.S1.S2'.A2+S0.S1.S2'.A3+S0'.S1'.S2A4
+S0.S1'.S2 A5+S0'.S1.S2 .A6+S0.S1.S3.A7

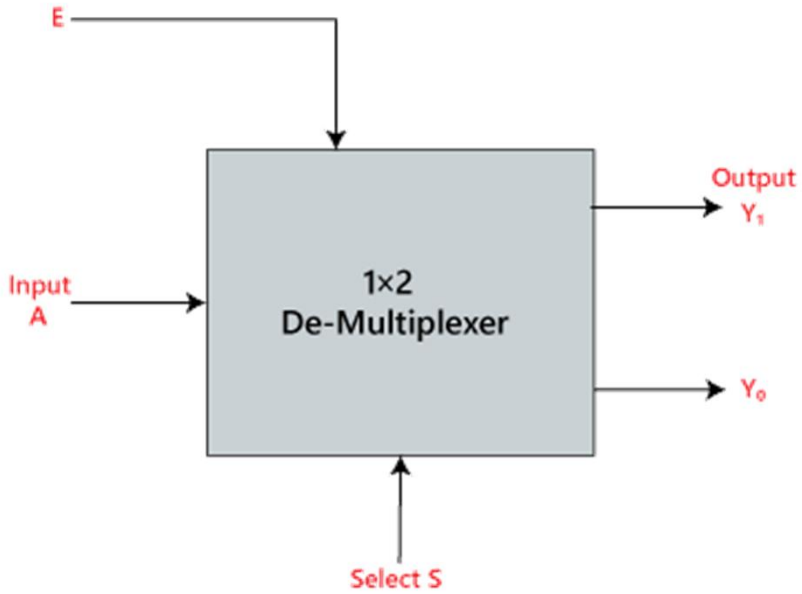


Demultiplexer

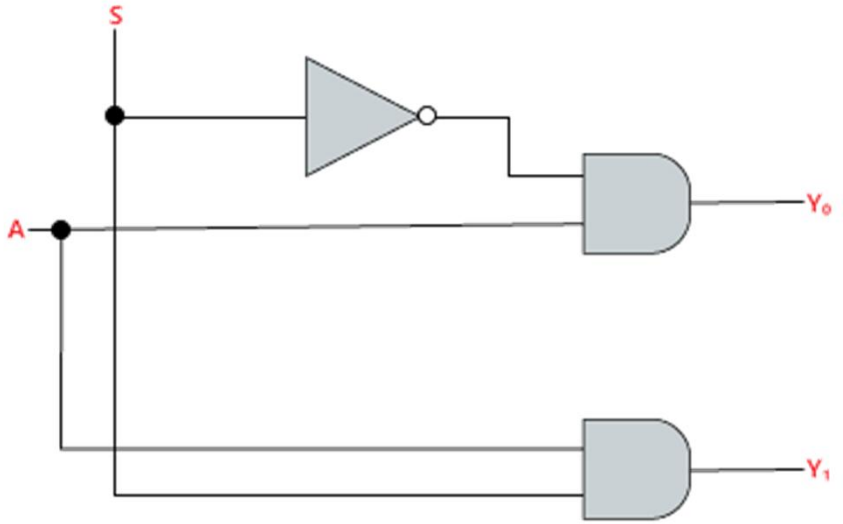
- A Demultiplexer is also called Demux or data distributor and its operation is quite opposite to a multiplexer because it is an inverse to multiplexer.
- Multiplexer is a many-to-one circuit whereas Demultiplexer is a one-to-many circuit.
- By using Demultiplexer, transmission of data can be done through one single input to a number of output data lines.
- A De-multiplexer is a combinational circuit that has only 1 input line and 2^N output lines.
- A demultiplexer, sometimes abbreviated as dmux, is a circuit that has one input and more than one output.
- It is used when a circuit intends to send a signal to one of many devices.
- Data which is obtained by a single input line can be transmitted to ‘n’ number of output lines.
- Multiplexers are called Data Selectors whereas
- Demultiplexers are Data Distributors because they transmit similar information which is obtained at the input to various outputs.

1×2 De-multiplexer

- In 1 to 2 De-multiplexer, there are only two outputs, i.e., Y_0, and Y_1, 1 selection lines, i.e., S_0, and single input, i.e., A.
- On basis of selection value, input will be connected to one of outputs.
- Block diagram and truth table of 1×2 multiplexer are given below.



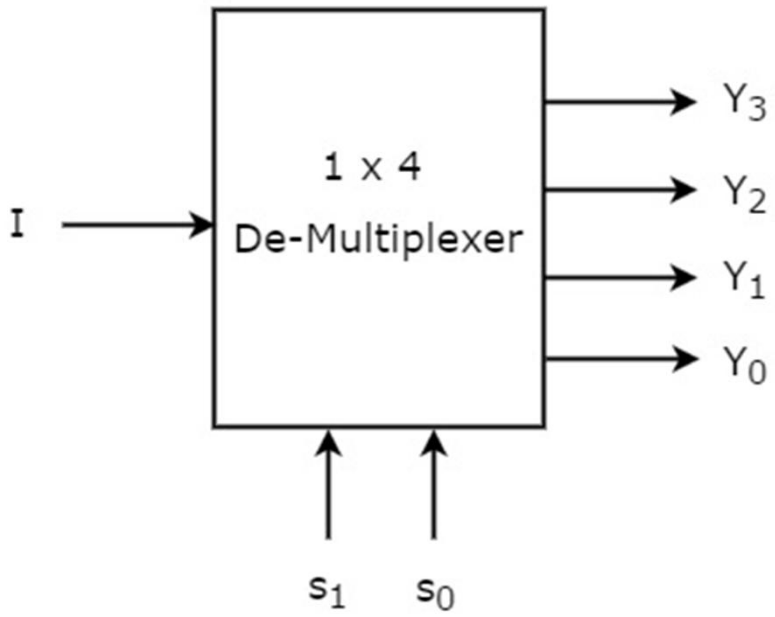
INPUTS	Output	
S_0	Y_1	Y_0
0	0	A
1	A	0



- Logical expression of term Y is as
  - $Y_0=S_0'.A$
  - $Y_1=S_0.A$
- Logical circuit of above expressions is as

1×4 De-multiplexer

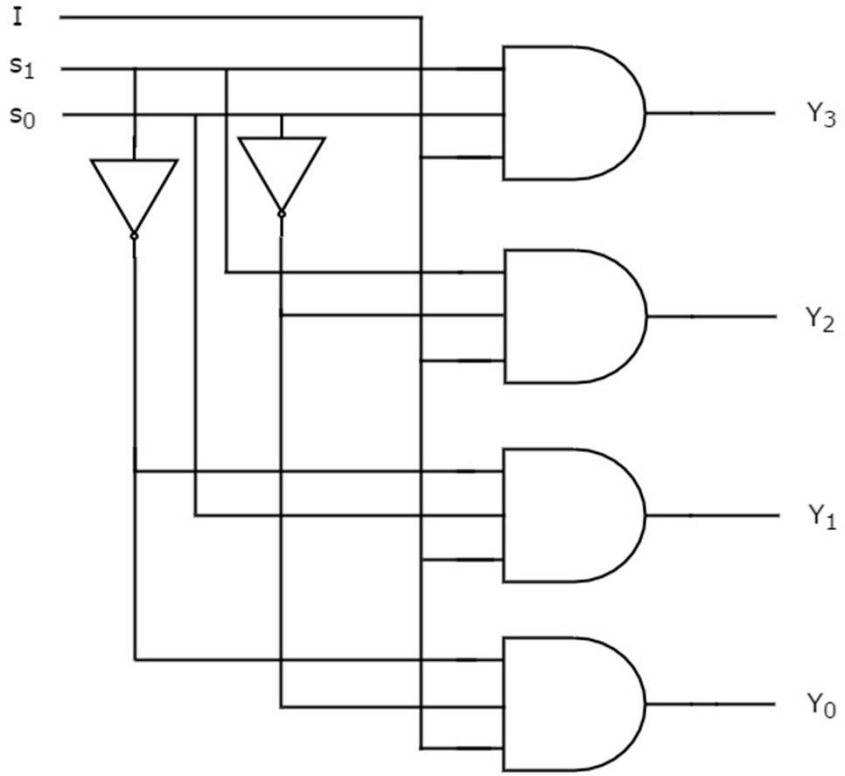
- 1x4 De-Multiplexer has one input I, two selection lines, s_1 & s_0 and four outputs Y_3, Y_2, Y_1 &Y_0.
- **Block diagram** of 1x4 De-Multiplexer is shown
- Single input 'I' will be connected to one of four outputs, Y_3 to Y_0 based on values of selection lines s_1 & s_0.
- **Truth table** of 1x4 De-Multiplexer is as.



Selection Inputs		Outputs			
S_1	**S_0**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	0	0	0	0	**I**
0	1	0	0	**I**	0
1	0	0	**I**	0	0
1	1	**I**	0	0	0

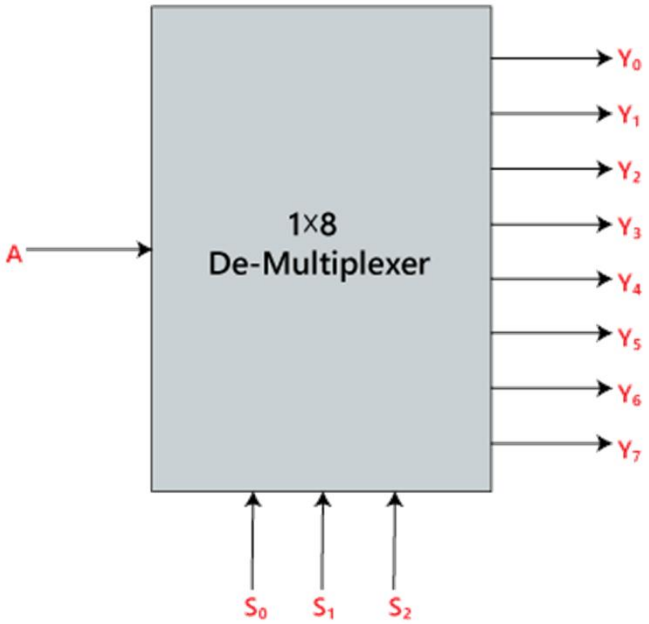
- Logical expression of the term Y is as follows:
  - $Y_0 = S_1' S_0' I$
  - $Y_1 = S_1' S_0 I$
  - $Y_2 = S_1 S_0' I$
  - $Y_3 = S_1 S_0 I$

- Implementation from these Boolean functions using Inverters & 3-input AND gates. The **circuit diagram** of 1x4 De-Multiplexer is shown in the following figure.



1×8 De-multiplexer

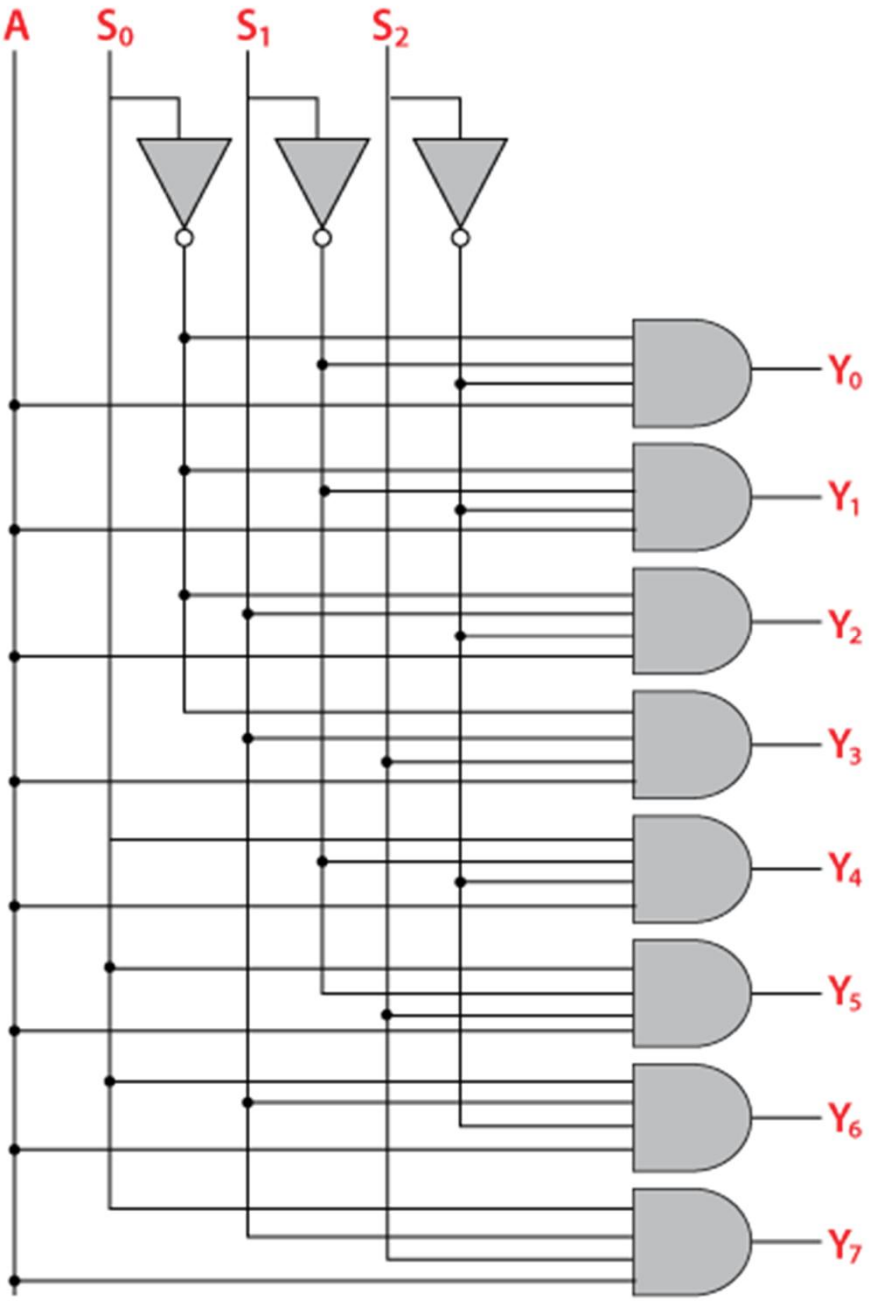
- Total eight outputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7, 3 selection lines, i.e., S_0, $S_1$and S_2 and single input, i.e., A.
- On basis of combination of inputs which are present at selection lines S_0, S_1 and S_2, input will be connected to one of these outputs.
- Block diagram and truth table of 1×8 de-multiplexer are given below.



INPUTS			Output							
S_2	S_1	S_0	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	0	A
0	0	1	0	0	0	0	0	0	A	0
0	1	0	0	0	0	0	0	A	0	0
0	1	1	0	0	0	0	A	0	0	0
1	0	0	0	0	0	A	0	0	0	0
1	0	1	0	0	A	0	0	0	0	0
1	1	0	0	A	0	0	0	0	0	0
1	1	1	A	0	0	0	0	0	0	0

- Logical expression of term Y is as follows:
  - $Y_0=S_0'.S_1'.S_2'.A$
  - $Y_1=S_0.S_1'.S_2'.A$
  - $Y_2=S_0'.S_1.S_2'.A$
  - $Y_3=S_0.S_1.S_2'.A$
  - $Y_4=S_0'.S_1'.S_2\ A$
  - $Y_5=S_0.S_1'.S_2\ A$
  - $Y_6=S_0'.S_1.S_2\ A$
  - $Y_7=S_0.S_1.S_3.A$

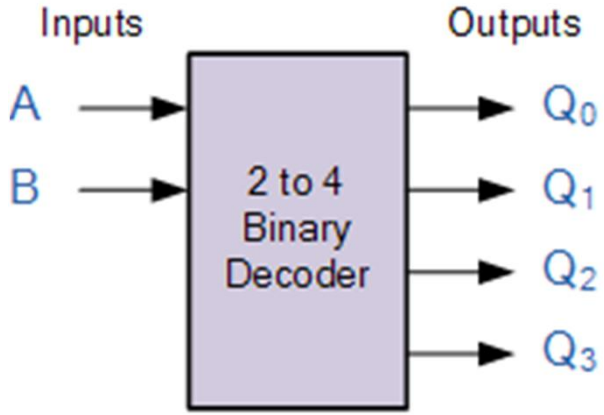
- Logical circuit of above expressions is given as



Binary Decoder

- Term “Decoder” means to translate or decode coded information from one format into another, so a binary decoder transforms “n” binary input signals into an equivalent code using 2^n outputs.
- A binary decoder (“decoder”) is a combinational logic circuit that converts binary information from n coded inputs to a maximum of 2^n unique outputs.
- Binary decoders are used in an extensive number of applications as data multiplexing and data demultiplexing and seven segment displays.

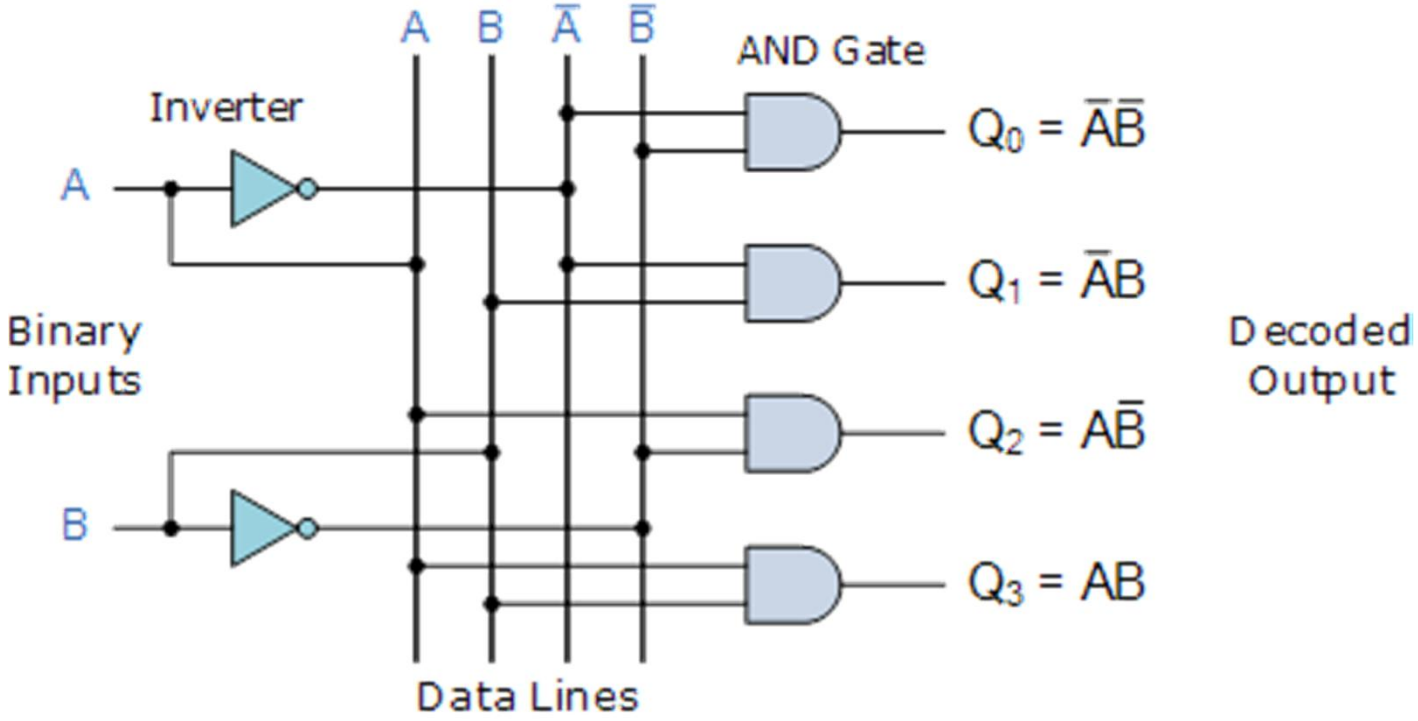
2-to-4 Binary Decoder



Truth Table

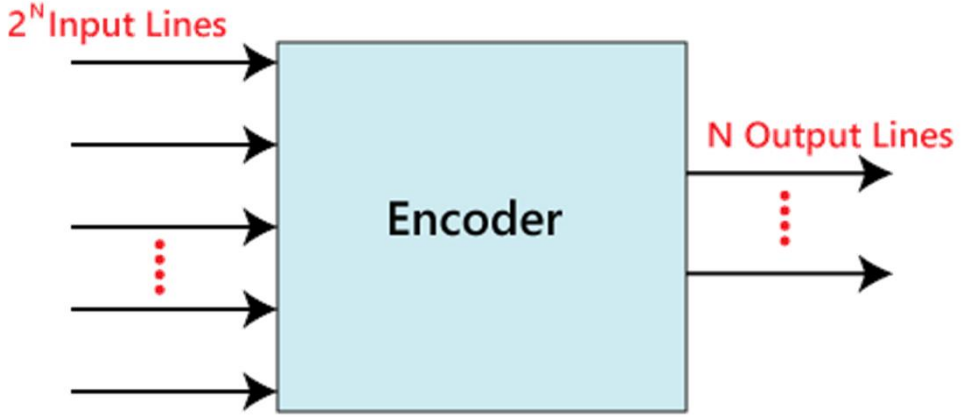
A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- 2-to-4 line binary decoder consists of an array of four AND gates.
- 2 binary inputs labeled as A and B are decoded into one of 4 outputs, hence description of 2-to-4 binary decoder.
- Each output represents one of minterms of 2 input variables,
- Binary inputs A and B determine which output line from Q0 to Q3 is “HIGH” at logic level “1” while remaining outputs are held “LOW” at logic “0” so only one output can be active (HIGH) at any one time.
- Therefore, whichever output line is “HIGH” identifies binary code present at input, in other words it “de-codes” binary input.



Encoder

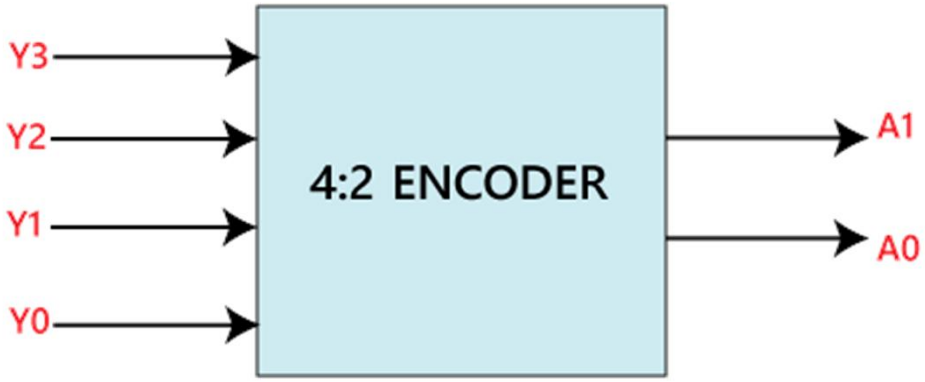
- An Encoder is a **combinational circuit** that performs reverse operation of Decoder.
- Combinational circuits that change binary information into N output lines are known as **Encoders**.
- It has maximum of **2^n input lines** and **‘n’ output lines**, hence it encodes information from 2^n inputs into an n-bit code.
- It will produce a binary code equivalent to input, which is active High.
- Therefore, encoder encodes 2^n input lines with ‘n’ bits.
- Output lines define N-bit code for binary information.
- At a time, only one input line is activated for simplicity.
- Produced N-bit output code is equivalent to binary information.



- Types of encoders which are as follows:

4 to 2 line Encoder

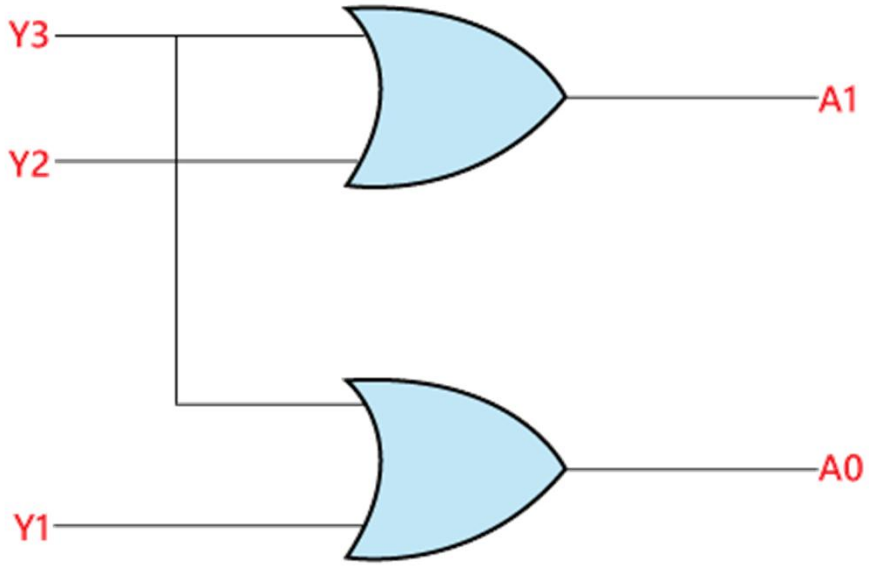
- In 4 to 2 line encoder, there are total of four inputs, i.e., Y_0, Y_1, Y_2, and Y_3, and two outputs, i.e., A_0 and A_1.
- In 4-input lines, one input-line is set to true at a time to get respective binary code in output side.
- Below are block diagram and truth table of 4 to 2 line encoder.



- Logical expressions for A0 and A1 are
  - $A_1 = Y_3 + Y_2$
  - $A_0 = Y_3 + Y_1$

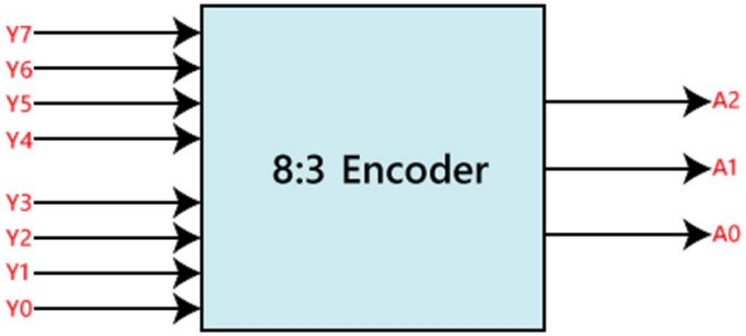
Inputs				Outputs	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- Logical expressions for A0 and A1 are
  - $A_1=Y_3+Y_2$
  - $A_0=Y_3+Y_1$
- Logical circuit of the above expressions is given below



8 to 3 line Encoder

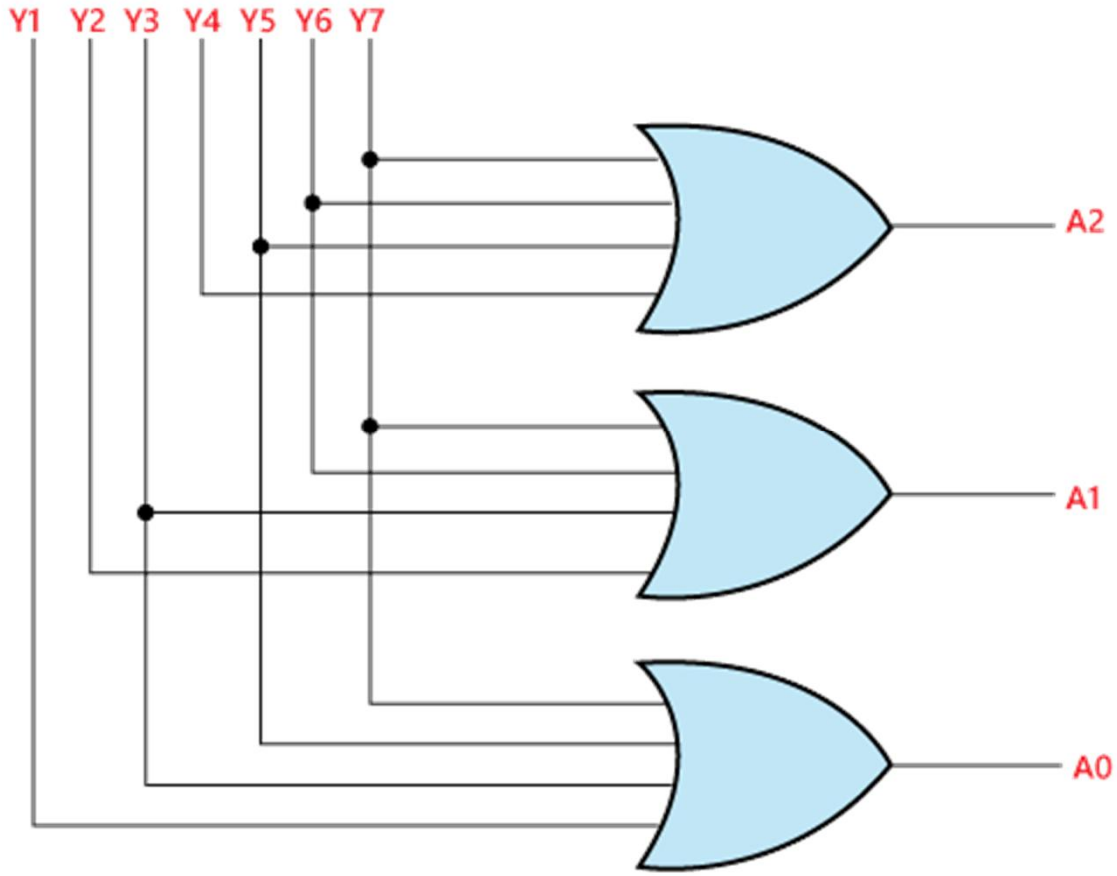
- 8 to 3 line Encoder is also known as **Octal to Binary Encoder**.
- In 8 to 3 line encoder, there is a total of eight inputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A1, and A_2.
- In 8-input lines, one input-line is set to true at a time to get respective binary code in output side.
- Below are block diagram and truth table of 8 to 3 line encoder.



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

- Logical expression of the term A0, A1, and A2 are as follows:
  - $A_2=Y_4+Y_5+Y_6+Y_7$
  - $A_1=Y_2+Y_3+Y_6+Y_7$
  - $A_0=Y_7+Y_5+Y_3+Y_1$

- Logical circuit of the above expressions is given below



Priority Encoder

4 to 2 line Priority Encoder:

- A 4 to 2 priority encoder has **4 inputs** : Y3, Y2, Y1 & Y0 and **2 outputs** : A1 & A0.
- Here, input, Y3 has **highest priority**, whereas input, Y0 has **lowest priority**.
- In this case, even if more than one input is '1' at same time, output will be (binary) code corresponding to input, which is having **higher priority**.
- Considered one more **output, V** in order to know, whether code available at outputs is valid or not.
    - If at least one input of encoder is '1', then code available at outputs is a valid one. In this case, output, V will be equal to 1.
    - If all inputs of encoder are '0', then code available at outputs is not a valid one. In this case, output, V will be equal to 0.
- Truth table for priority encoder is as :

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

- Logical expression of term A_0 and A_1 can be found using **K-map** as:

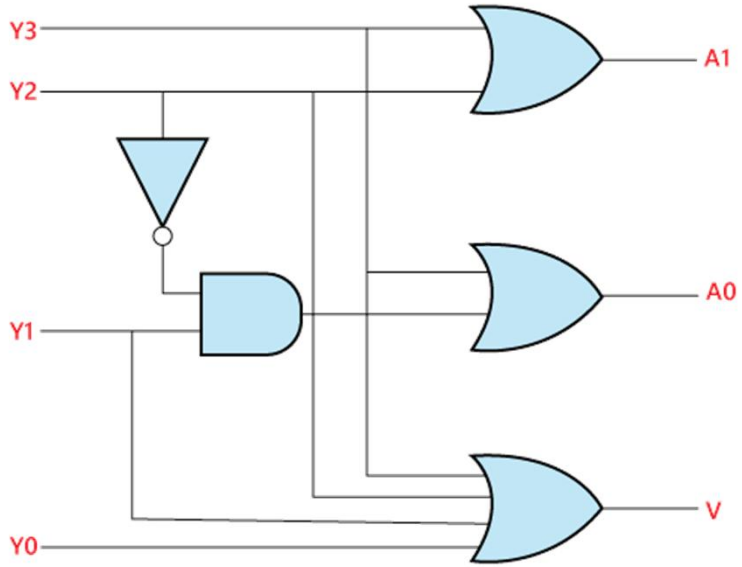
Y_3Y_2 \ Y_1Y_0	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$A_1 = Y_3 + Y_2$

Y_3Y_2 \ Y_1Y_0	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	x	x	x	x
10	1	1	1	1

$A_0 = Y_3 + Y_2' Y_1$

Logical circuit of the above expressions is given below:



Adder

- An adder circuit is one of important digital circuits used in computers, calculators, digital processing units, etc.
- There are two types adder circuits named **half-adder** and **full-adder**.
- Both half adder and full adder circuits are used to perform addition and also widely used for performing various arithmetic functions in digital circuits.

Half Adder

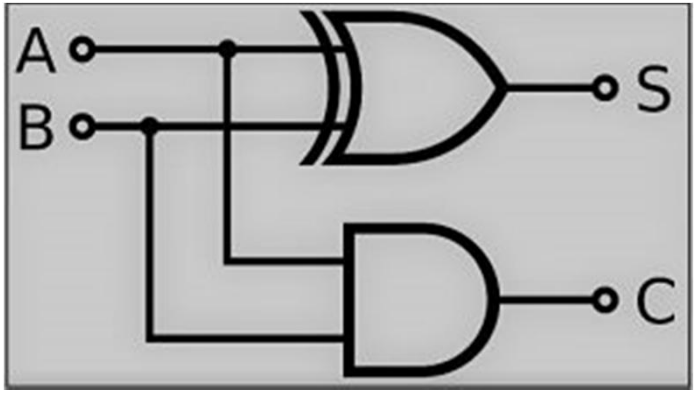
- A combinational logic circuit which is designed to add two binary digits is known as **half adder**.
- It provides output along with a carry value (if any).

Block diagram

Truth Table

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- It has two input terminals A and B
- Two output terminals for sum and carry.
- The carry output is 0 in case where both the inputs are not 1.
- From truth table Sum is X-OR of input A and B. Carry is AND of input A and B.
- So Half adder circuit is designed by connecting an EX-OR gate and one AND gate



- SOP form of the sum and carry are as follows:
- **Sum= A'B+AB'= A XOR B**
- **Carry = AB = A AND B**

Full Adder

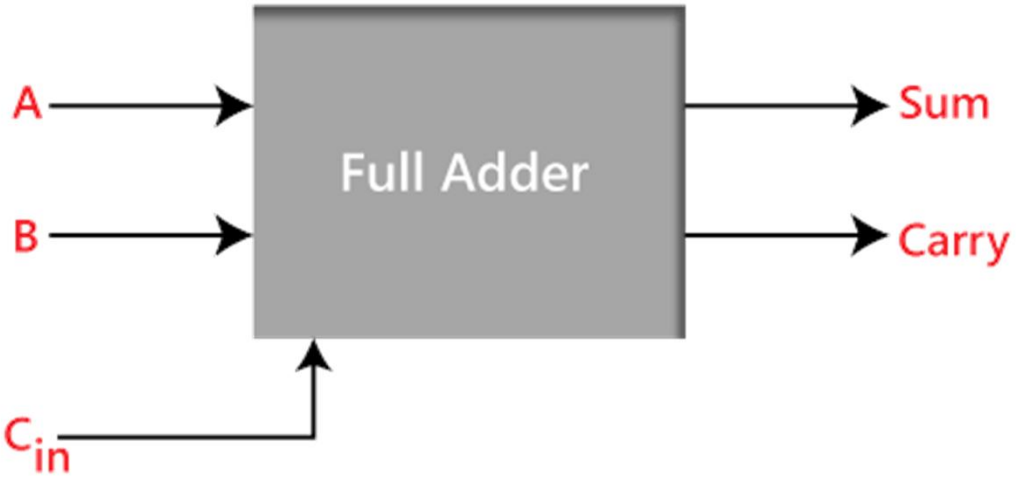
- The half adder is used to add only two numbers. To overcome this problem, the full adder was developed.

- A combinational circuit which is designed to add three binary digits and produce two outputs is known as full adder.

- The full adder is used to add three 1-bit binary numbers A, B, and carry C.

- The full adder has three inputs and two outputs i.e., sum and carry.

- **Full adder** circuit adds three binary digits, where two are inputs and one is carry forwarded from previous addition.
- Block diagram



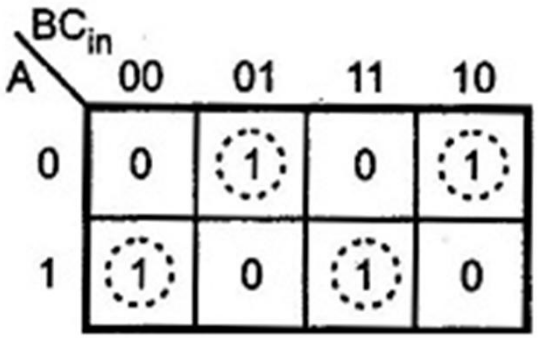
Truth table

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

In above table,

- 'A' and' B' are input variables. These variables represen two significant bits which are going to be added
- 'C_{in}' is third input which represents carry. From previous lower significant position, carry bit is fetched.
- 'Sum' and 'Carry' are output variables that define output values.

For sum = Óm(1,2,4,7)



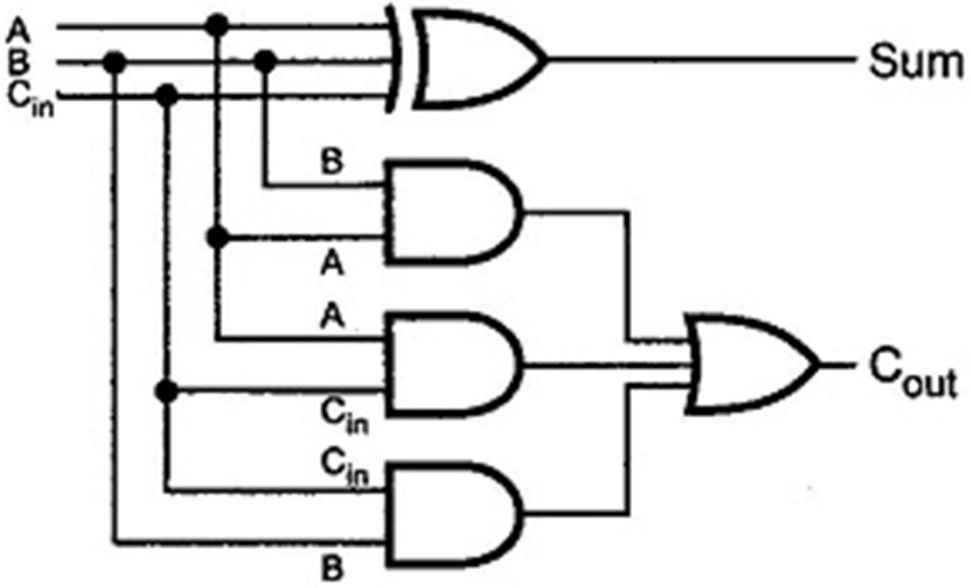
$$\begin{aligned}
\text{Sum} &= \overline{A}\,\overline{B}\,C_{in} + \overline{A}\,B\,\overline{C}_{in} + A\,\overline{B}\,\overline{C}_{in} + A\,B\,C_{in} \\
&= C_{in}\,(\overline{A}\,\overline{B} + AB) + \overline{C}_{in}\,(\overline{A}\,B + A\,\overline{B}) \\
&= C_{in}\,(A \odot B) + \overline{C}_{in}\,(A \oplus B) \\
&= C_{in}\,(\overline{A \oplus B}) + \overline{C}_{in}\,(A \oplus B) \\
&= C_{in} \oplus (A \oplus B)
\end{aligned}$$

For carry = Óm(3,5,6,7)

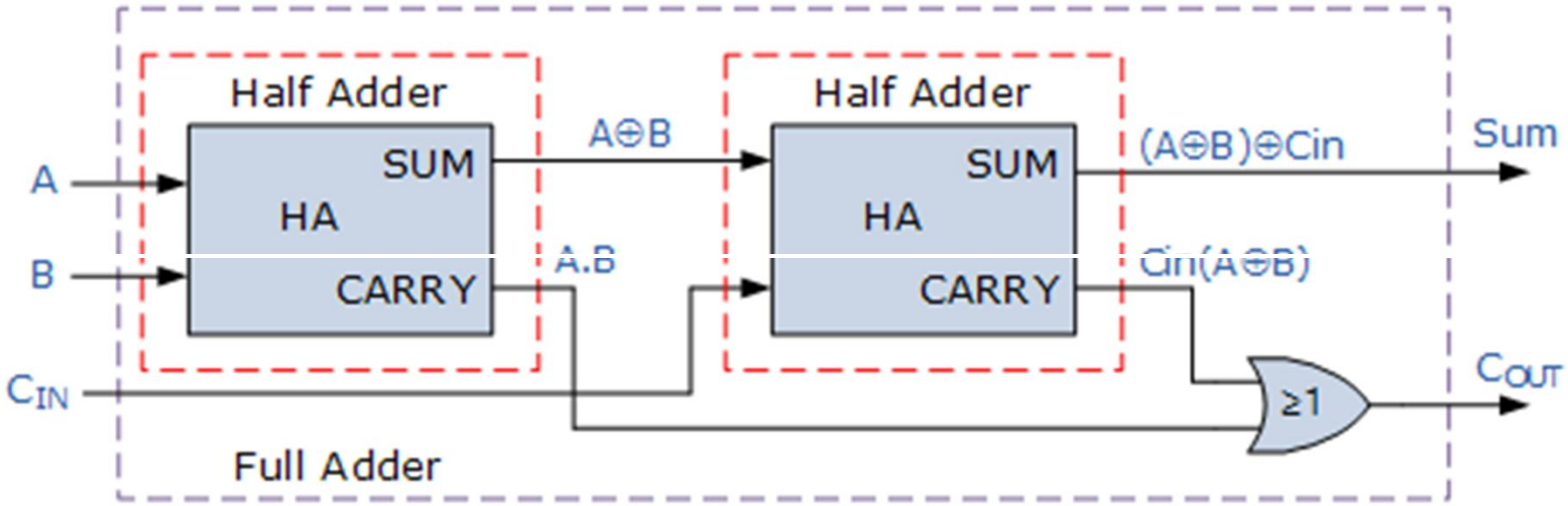
A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

- CARRY= $BC_{in} + AC_{in} + AB$

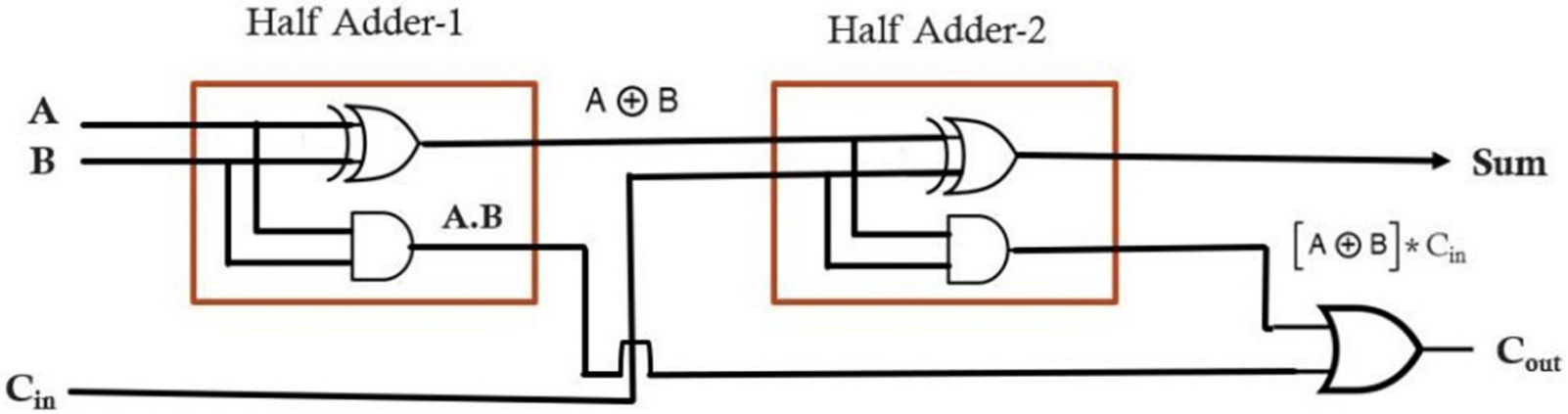
Gate level circuit from equations



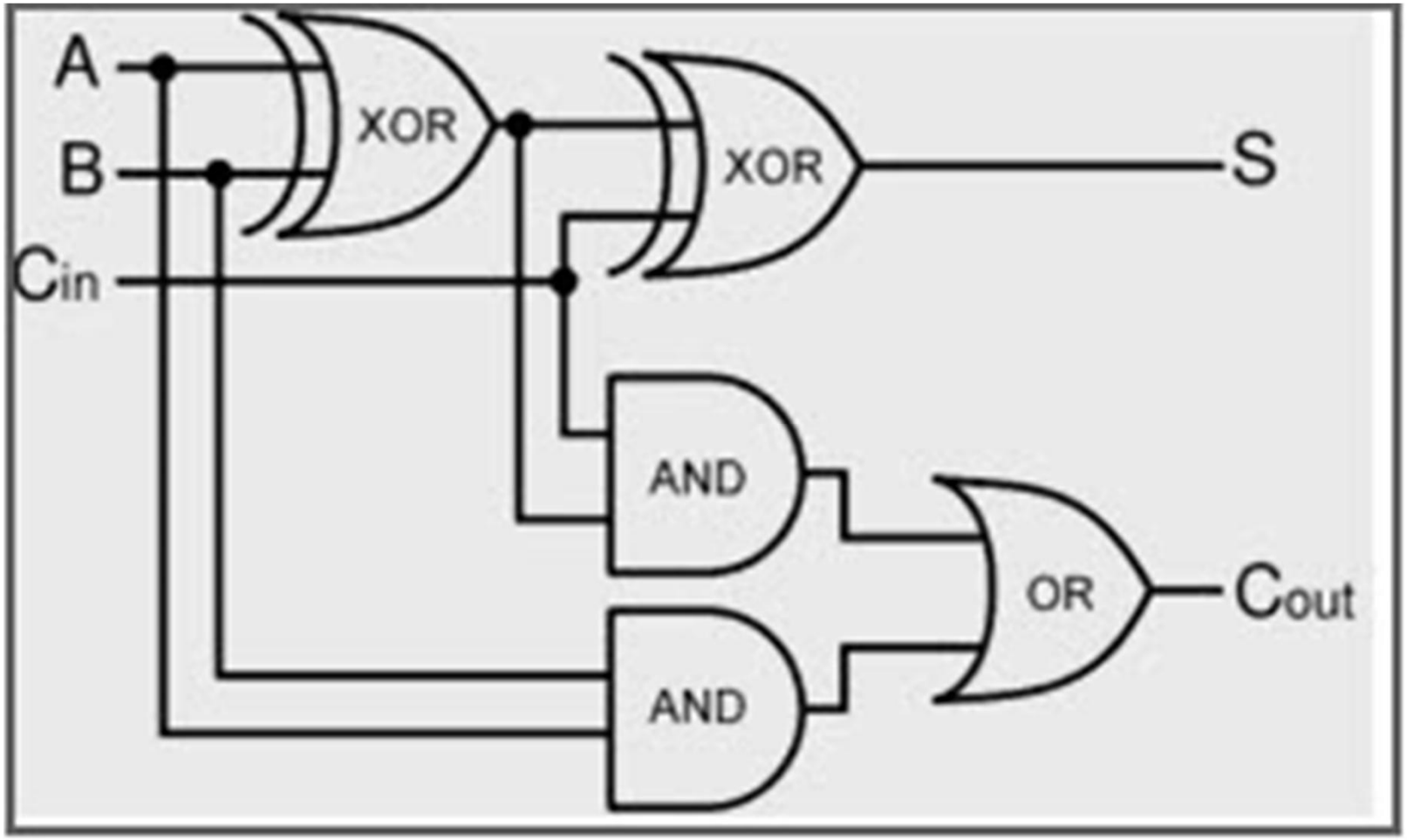
Implementation of Full Adder using Half Adders



- Circuit of the full adder consists of two EX-OR gates, two AND gates and one OR gate, which are connected together as shown in the full adder circuit.



•**Full adder circuit diagram** is shown below:



Subtractor

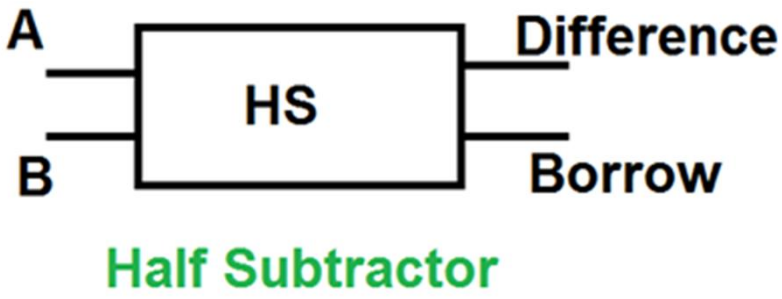
- A digital logic circuit that is widely used for subtraction of numbers is known as subtractor.
- Just like adders, subtractors are also used in processors that calculates addresses and similar activities.
- Subtractor Circuit is a combinational logic circuit that performs subtraction on binary numbers.
- As digits involved in Binary Notation are 0 and 1, subtraction of ‘0’ from a ‘0’ or ‘1’ does not change result.
- ‘1’ subtracted from ‘1’ results in ‘0’. Subtracting ‘1’ from ‘0’ requires borrow.

Subtractors are classified into two types –

- Half Subtractors
- Full Subtractors

Half Subtractor

- A half subtractor is a digital combinational logic circuit that performs binary subtraction of two single-bit binary numbers.
- It has two inputs (one bit each) termed as A and B that generates two outputs, DIFFERENCE(D) and BORROW(B).
- DIFFERENCE output is difference between two input bits, while BORROW output indicates whether borrowing was necessary during subtraction.
- Half subtractor is designed using three logic gates that is AND gate, NOT gate and XOR gate.
- Output of difference is obtained from XOR gate and output of borrow is obtained from AND gate.
- In subtraction (A-B), A is called a **Minuend bit** and B is called a **Subtrahend bit.**

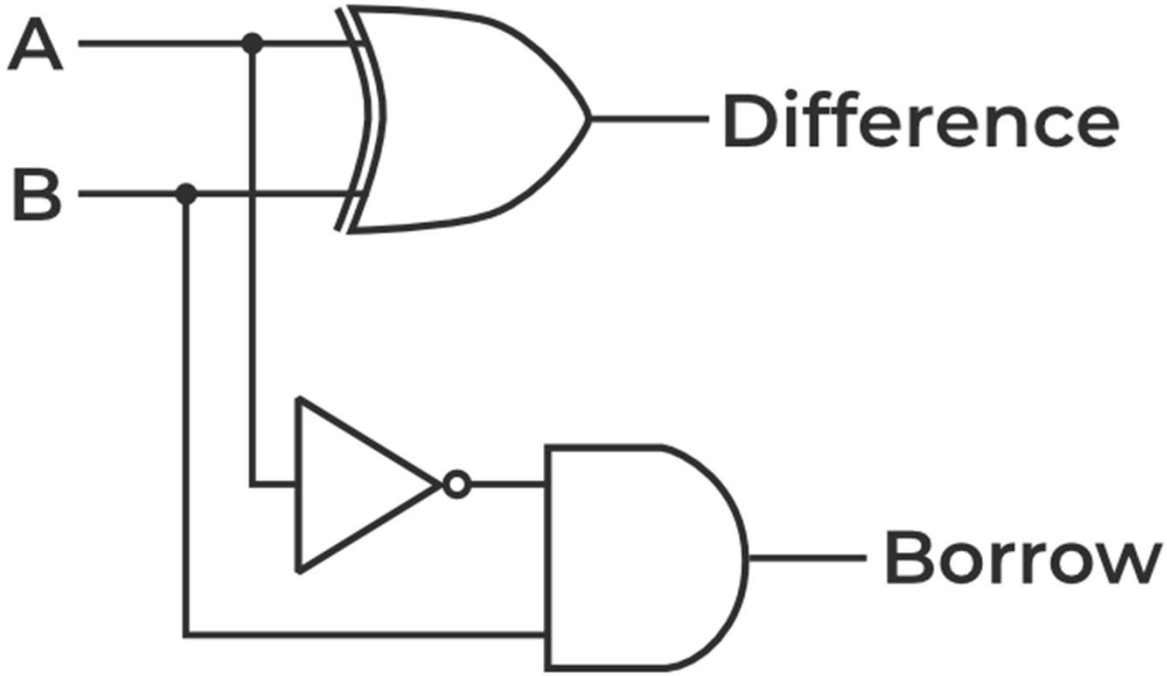


Half Subtractor

A	B	Difference(D)	Borrow(B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- Expressions for Difference and Borrow from truth table are as follows:
    - Difference = D = A'B+AB' = A⊕ B
    - Borrow = B = A'B

- **Implementation**

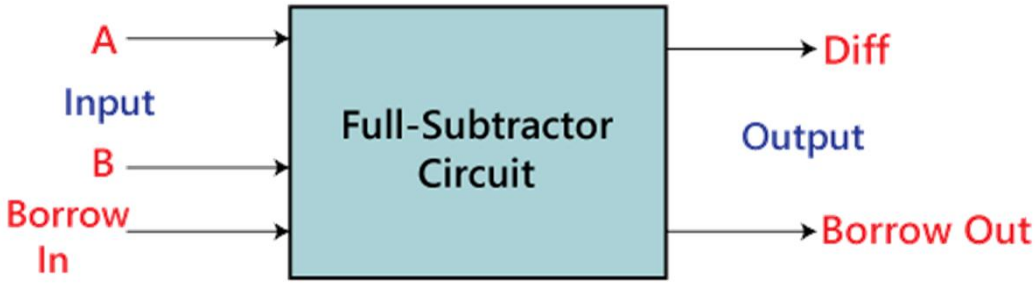


Full subtractor

- A full subtractor is a combinational circuit that performs subtraction involving three bits, A (minuend), B (subtrahend), and Bin (borrow-in).

- It accepts three inputs: A (minuend), B (subtrahend) and a Bin (borrow bit) and it produces two outputs: D (difference) and Bout (borrow out).

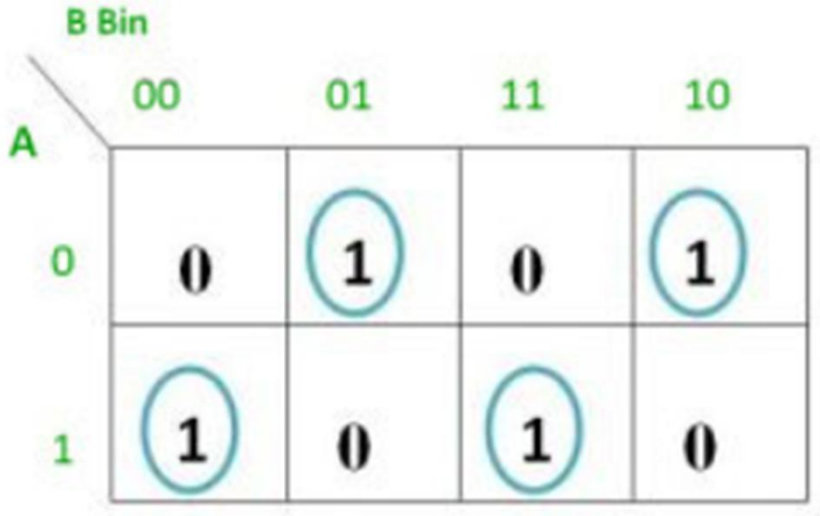
- Logic symbol and truth table are shown below.



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

From above table we can draw the K-Map as shown for “difference”



Logical expression for difference –

D = A’B’Bin + A’BBin’ + AB’Bin’ + ABBin

= Bin(A’B’ + AB) + Bin’(AB’ + A’B)

= Bin(A XNOR B) + Bin’(A XOR B)

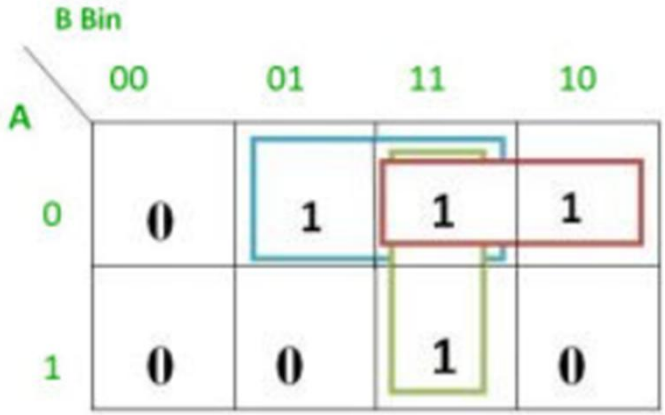
= Bin (A XOR B)’ + Bin’(A XOR B)

= Bin XOR (A XOR B)

= (A XOR B) XOR Bin

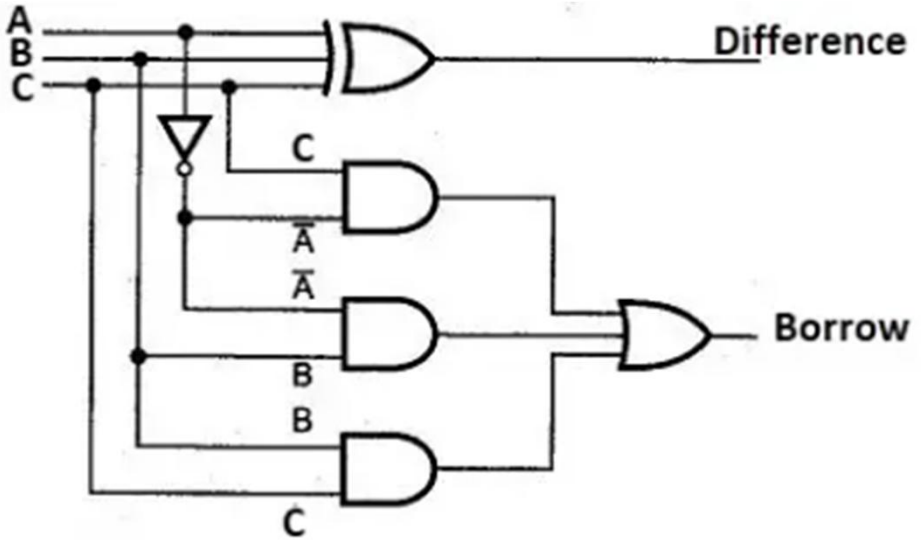
= A B Bin

From above table we can draw the K-Map as shown for “borrow”.

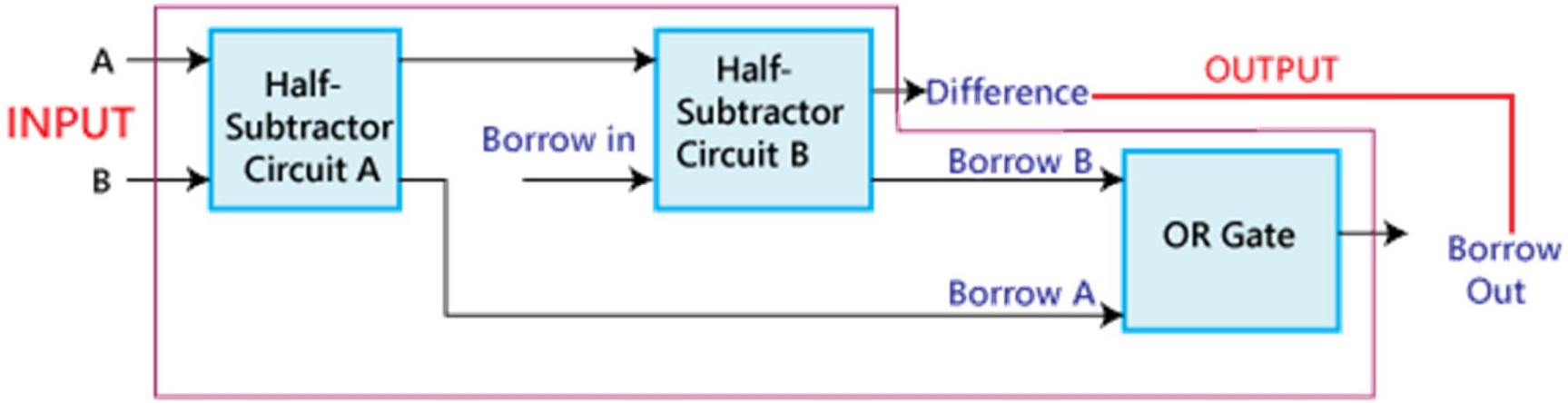


Logical expression for borrow
Bout = A' Bin + A' B + B Bin

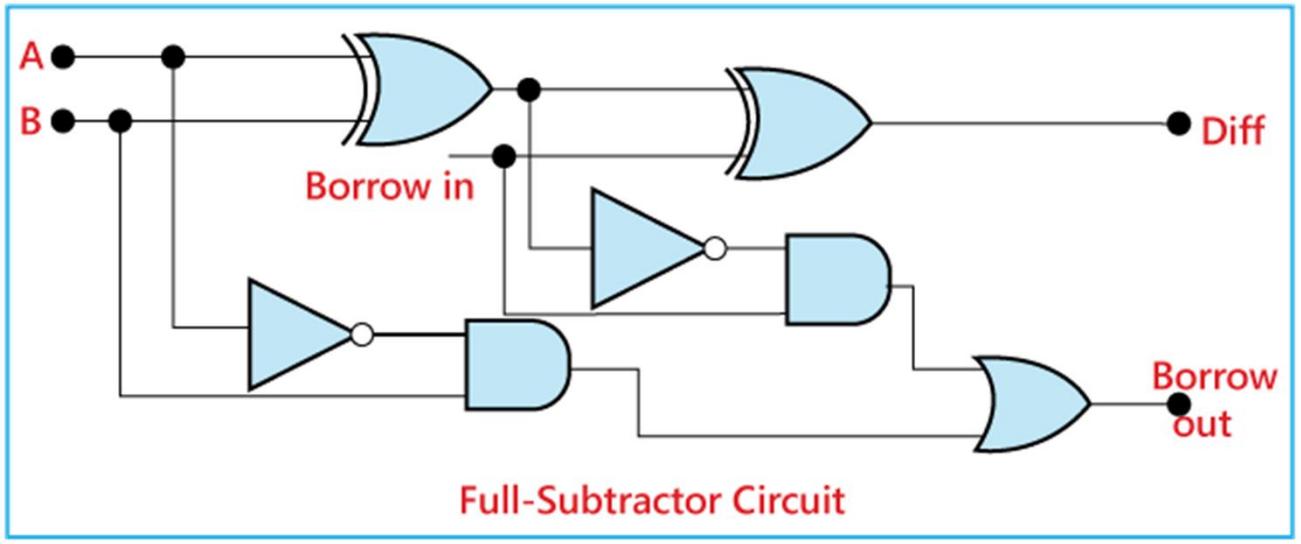
Gate level circuit from equations



Construction of Full Subtractor Circuit using half subtractors



- Full subtractor logic circuit can be constructed using the **'AND', 'XOR', and NOT gate** with an **OR gate**.



- Actual logic circuit of full subtractor is shown in above diagram.

- Full subtractor circuit construction can also be represented in a Boolean expression.

  - Difference:

- Perform XOR operation of input A and B.
- Perform XOR operation of outcome with 'Borrow'.
- So, difference is (A XOR B) XOR '$Borrow_{in}$' which is also represented as:

  (A B) '$Borrow_{in}$'

  - Borrow:

- Perform 'AND' operation of inverted input A and B.
- Perform 'XOR' operation of input A and B.
- Perform 'OR' operations of both outputs that come from previous two steps.
- So 'Borrow' can be represented as:

  A'.B + (A B)'

A	B	C	D	Br	D	Br
0	0	0	0	0	–	–
0	0	1	1	1	A’B’C	A’B’C
0	1	0	1	1	A’BC’	A’BC’
0	1	1	0	1	–	A’BC
1	0	0	1	0	AB’C’	–
1	0	1	0	0	–	–
1	1	0	0	0	–	–
1	1	1	1	1	ABC	ABC

- Expression for Full Subtractor
- Logic expression for full subtractor is given as
  - **Difference (D) = A’B’C + A’BC’ + AB’C’ + ABC**
  - **Borrow (Br) = A’B’C + A’BC’ + A’BC + ABC**
- **Final expression**
  - **Difference (D) = A B C**
  - **Borrow = A’B + BC + A’C**