Basic Electronics Day 5

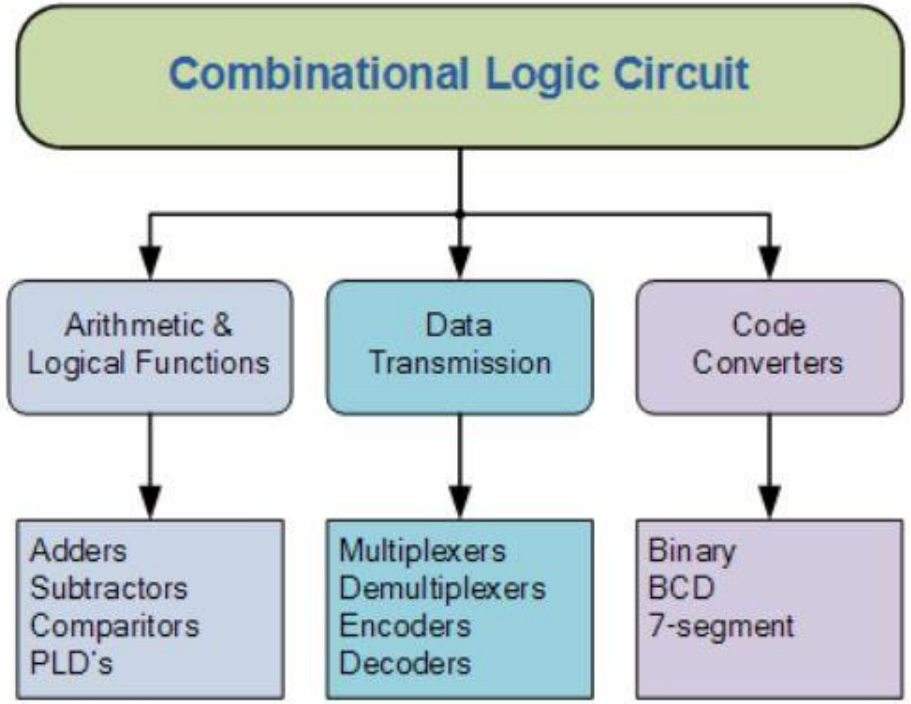
Adder
Subtractor
Multiplexer
De-multiplexers

1st Year of 4 year B.Tech.

Arithmetic Circuits

- Digital logic circuits are often known as switching circuits, because in digital circuits the voltage levels are assumed to be switched from one value to another value instantaneously. These circuits are termed as logic circuits, as their operation obeys a definite set of logic rules
- Digital system has usually two types of circuits:
    - Combinational Logic circuit
    - Sequential Logic circuit
- In combinational circuits, the output at any time depends only on the input values at that time
- In a sequential circuit, the output at any time depends on the present input values as well as the past output values
- Combinational digital logic circuits are basically made up of digital logic gates like AND gate, OR gate, NOT gate and universal gates (NAND gate and NOR gate)
- Combinational circuits are used in microprocessor and microcontroller for designing the hardware and software components of a computer

- Combinational digital logic circuits are classified into three major parts – arithmetic or logical functions, data transmission and code converter.
- The following chart will elaborate the further classifications of combinational digital logic circuits



Classification of combinational logic circuit

Half Adder:

- The half adder is a basic building block having two inputs and two outputs. The adder is used to perform OR operation of two single bit binary numbers
- The **carry** and **sum** are two output states of the half adder
- The half adder circuit is designed to add two single bit binary number A and B

The truth table of a half adder is given below:

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

Truth table

From the truth table the logic equations can be derived as:

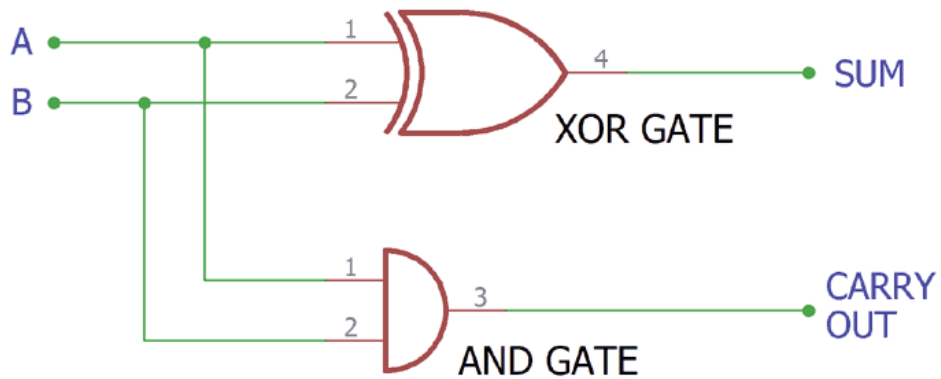
$$S = \bar{A}B + A\bar{B}$$

$$S = A \oplus B$$

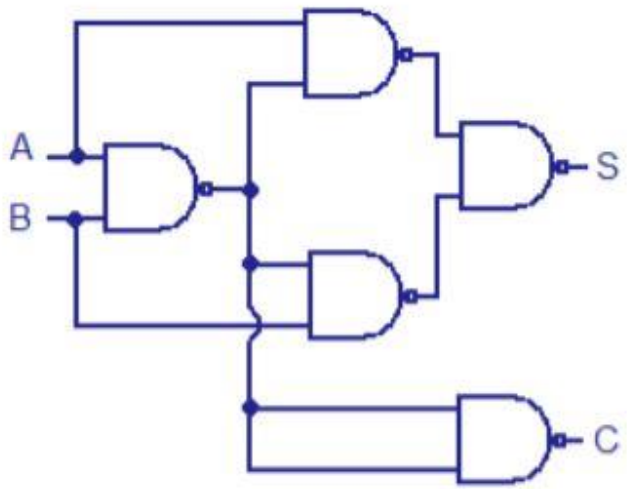
and,

$$C = A \cdot B$$

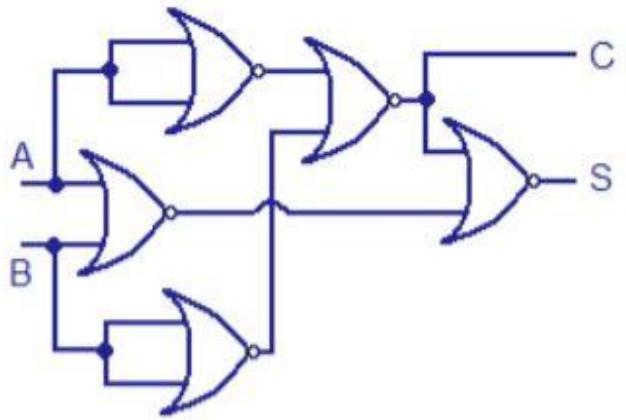
Hence, the circuit diagram for a half adder can be implemented as shown below:



NAND gates or NOR gates can be used for realizing the half adder in universal logic and the relevant circuit diagrams are shown in the figure below



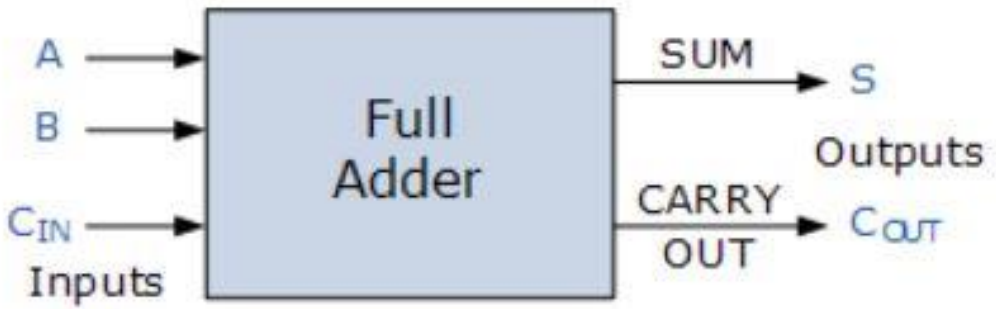
Half adder using NAND logic



Half adder using NOR logic

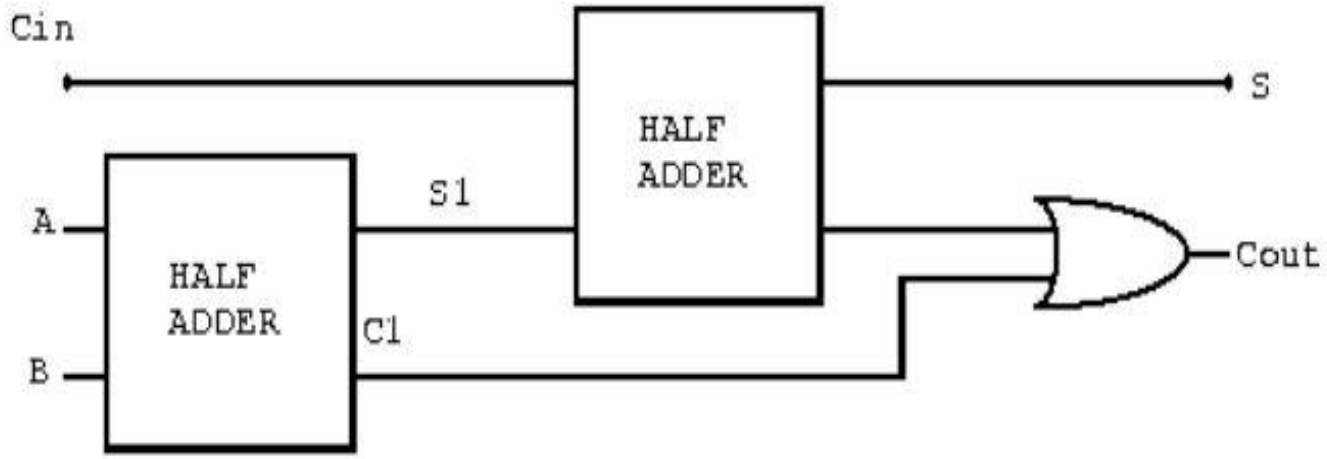
Full Adder:

- A half adder has only two bits and has no provision to add a carry coming from the lower order bits, when multibit addition is performed
- To fulfil this purpose, a full adder is designed
- A full adder performs the arithmetic sum of three input bits and produces a sum and a carry output
- The main difference between the Full Adder and the previous Half Adder is that a full adder has three inputs. The same two single bit data inputs A and B as before plus an additional Carry-in (C-in) input to receive the carry from a previous stage as shown below.



- Then the full adder is a logical circuit that performs an addition operation on three binary digits and just like the half adder, it also generates a carry out to the next addition column. Then a Carry-in is a possible carry from a less significant digit, while a Carry-out represents a carry to a more significant digit

- In many ways, the full adder can be thought of as two half adders connected together, with the first half adder passing its carry to the second half adder as shown



- As the full adder circuit above is basically two half adders connected together, the truth table for the full adder includes an additional column to take into account the Carry-in, C_{IN} input as well as the summed output, S and the Carry-out, C_{OUT} bit

Input			Output	
A	B	Cin	Sum	Cout
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

$$
\begin{aligned}
\textbf{SUM} &= \overline{A}\,\overline{B}\,C_{in} + \overline{A}\,B\,\overline{C}_{in} + A\,\overline{B}\,\overline{C}_{in} + A\,B\,C_{in} \\
&= C_{in}\,(\overline{A}\,\overline{B} + A\,B) + \overline{C}_{in}\,(A\,\overline{B} + \overline{A}\,B) \\
&= C_{in}\,\left[(\overline{A} + B)\cdot(A + \overline{B})\right] + \overline{C}_{in}\,(A\,\overline{B} + \overline{A}\,B) \\
&= C_{in}\,(\overline{A\,\overline{B}}\cdot\overline{\overline{A}\,B}) + \overline{C}_{in}\,(A\,\overline{B} + \overline{A}\,B) \\
&= C_{in}\,\overline{(A\,\overline{B} + \overline{A}\,B)} + \overline{C}_{in}\,(A\,\overline{B} + \overline{A}\,B) \\
&= C_{in} \oplus (A\,\overline{B} + \overline{A}\,B) \\
&= C_{in} \oplus (A \oplus B)
\end{aligned}
$$

For Carry (C_{out})

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

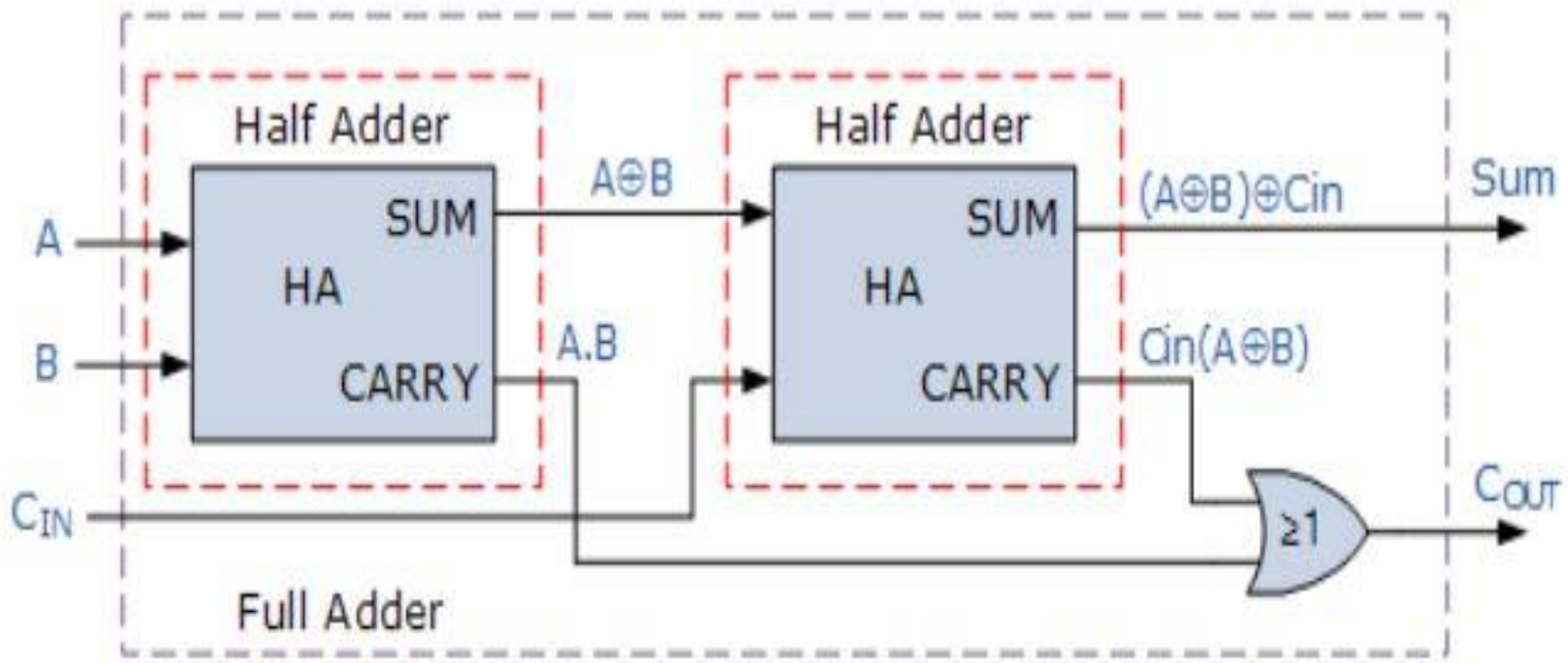
$C_{out} = AB + A\,C_{in} + B\,C_{in}$

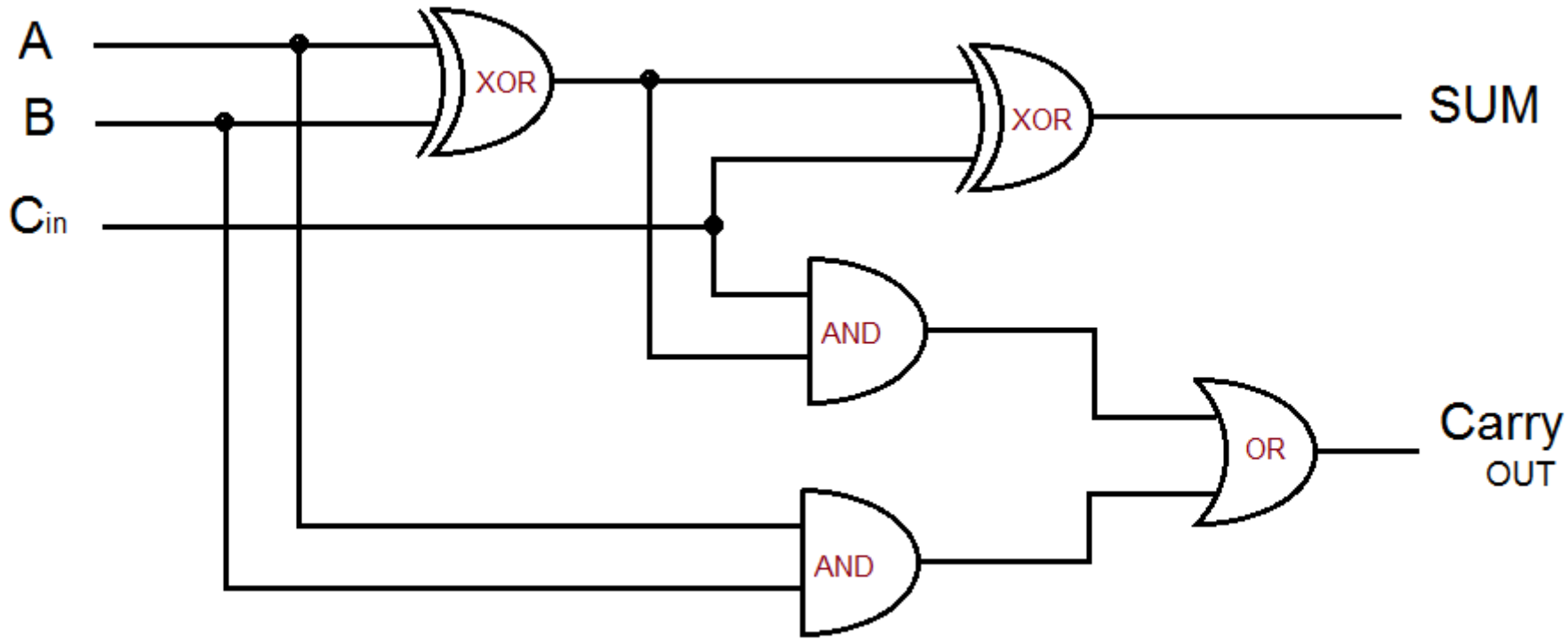
For Sum

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

$Sum = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in}$

- Full adder implemented using two Half adders:

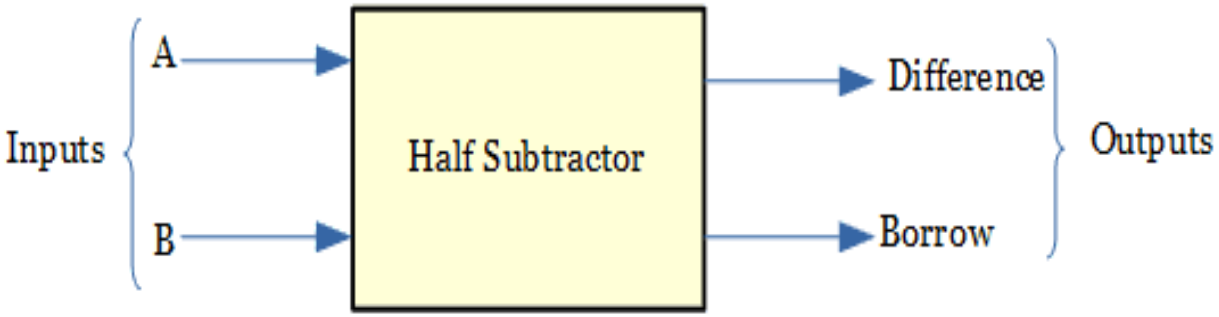




Try Full adder using only basic gates!!!

Half Subtractor

- A half subtractor is a combinational circuit which is used to perform subtraction of two bits
- It has two inputs – minuend and subtrahend and two outputs-difference and borrow out
- The truth table for a half subtractor is shown below-



Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

0 - 0 = 0

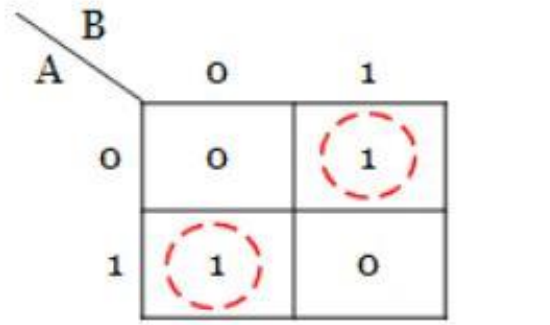
0 - 1 = 1 with 1 borrow

1 - 0 = 1

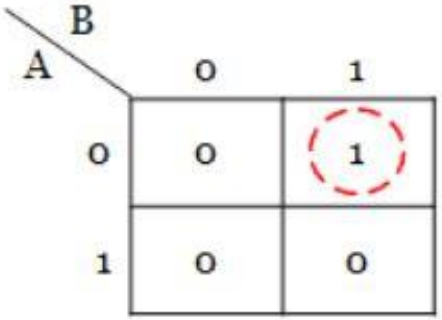
1 - 1 = 0

Minuend subtrahend

- For difference and borrow outputs, a boolean expression has to be derived using Karnaugh map. Since it has only two input variables, 4-cells k-map is used to simplify.

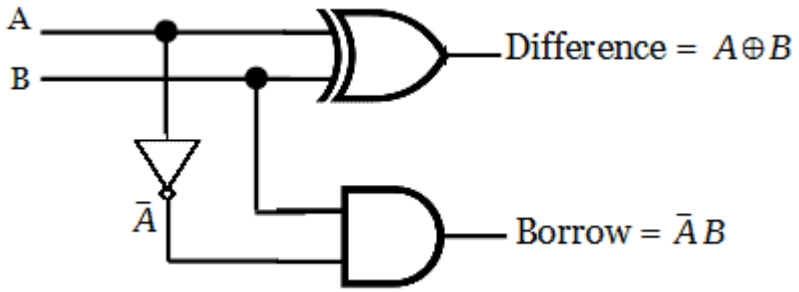


Difference = $A\overline{B} + \overline{A}B = A \oplus B$



Borrow = $\overline{A}B$

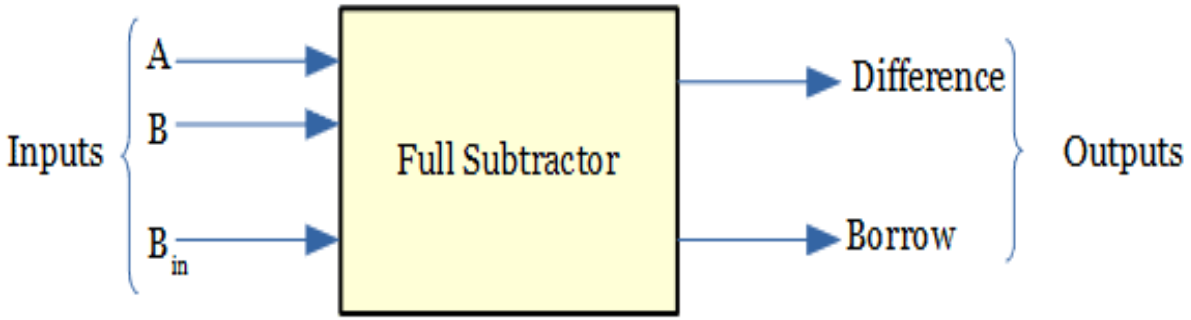
The obtained boolean expression for difference output is an Ex-OR gate output. Hence, the Logic circuit diagram for a half subtractor circuit is given below.



Half subtractor is limited to subtraction of two bits without borrow. But when performing multi digit operations, the subtraction is to be performed with the borrow from the previous digit subtraction. This computation is not possible with half subtractor. Hence full subtractor is used for such operations.

Full Subtractor

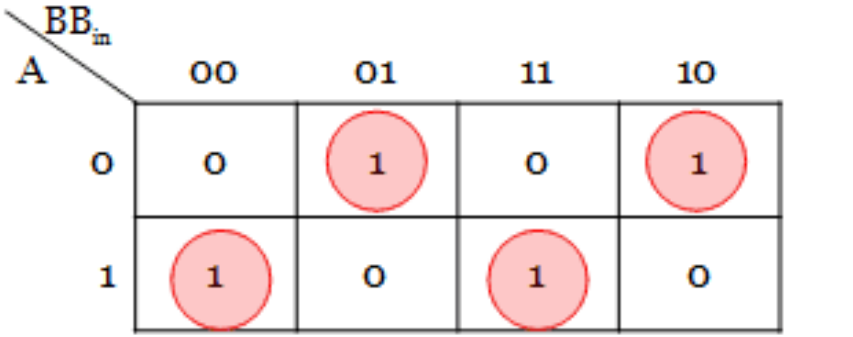
- A full subtractor has three binary inputs(subtrahend, minuend and one borrow bit from the previous subtraction operation). It produces two binary outputs(difference and borrow) using a combination of logic gates. The borrow given to the previous subtraction operation is denoted as B_{in}



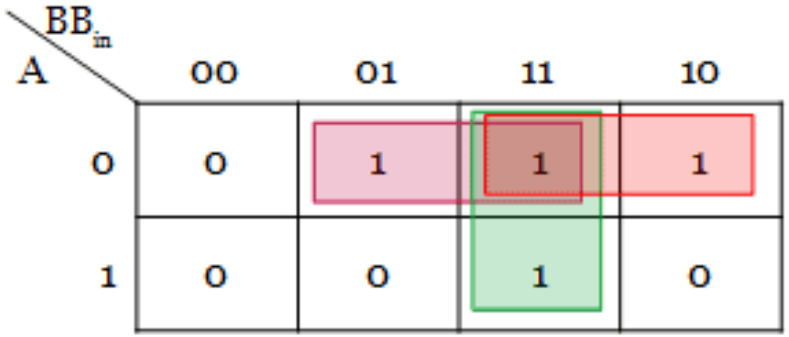
In full subtractor, eight possible operations are possible with three inputs and produces eight, two digit outputs. The operation is shown in the truth table below.

Inputs			Outputs	
A	B	B_{in}	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

- For difference and borrow outputs, boolean expression has to be derived using Karnaugh map. Since it has three input variables, 8-cells k-map is used to simplify the expression.



Difference = $\overline{A}\overline{B}B_{in} + \overline{A}B\overline{B}_{in} + A\overline{B}\overline{B}_{in} + ABB_{in}$



Borrow = $\overline{A}B + \overline{A}B_{in} + BB_{in}$

The boolean expression for difference output can further be simplified as -

$$\text{Difference} = \bar{A}\,\bar{B}\,B_{in} + \bar{A}\,B\,\bar{B}_{in} + A\,\bar{B}\,\bar{B}_{in} + A\,B\,B_{in}$$

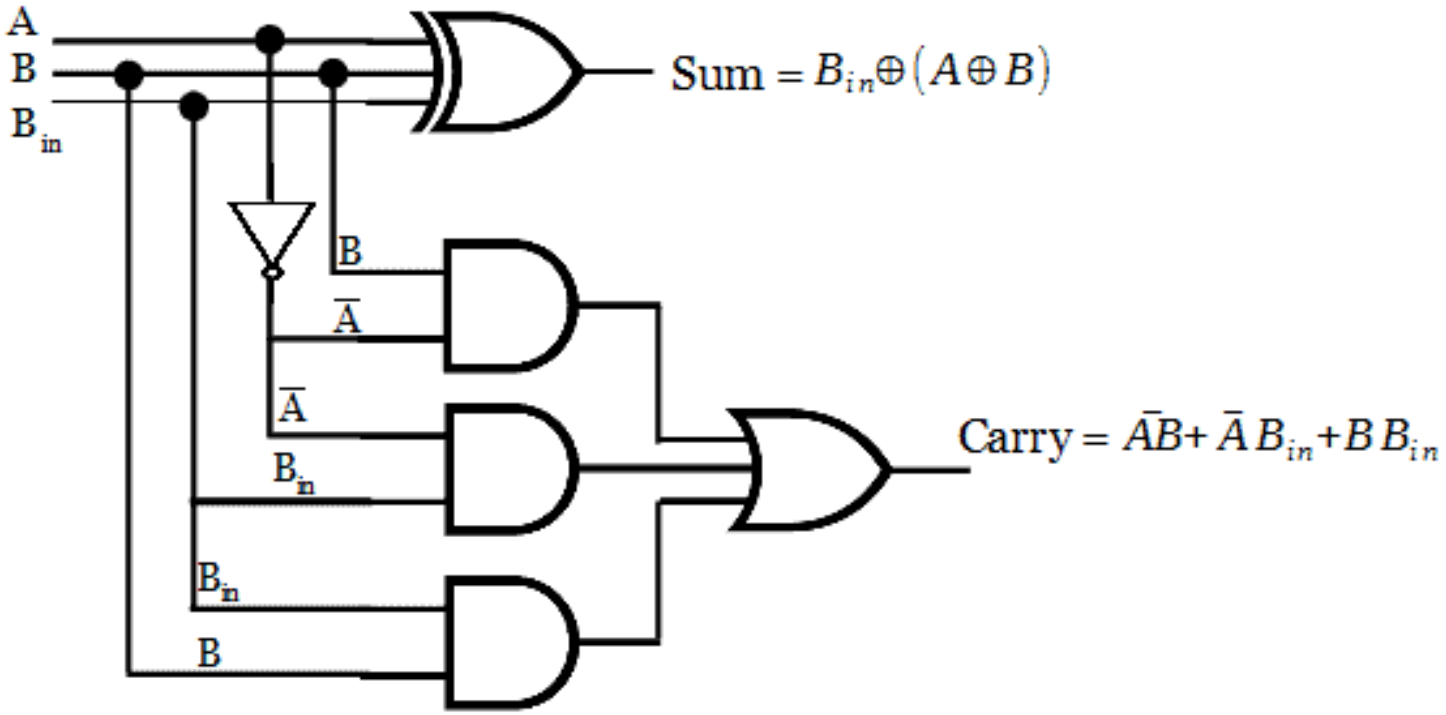
$$= B_{in}(\bar{A}\,\bar{B} + A\,B) + \bar{B}_{in}(\bar{A}\,B + A\,\bar{B})$$

$$= B_{in}(A \odot B) + \bar{B}_{in}(A \oplus B)$$

$$= B_{in}(\overline{A \oplus B}) + \bar{B}_{in}(A \oplus B)$$

$$= B_{in} \oplus (A \oplus B)$$

We have two boolean expressions for difference and borrow output. With those expressions, the combinational circuit for full subtractor is implemented.

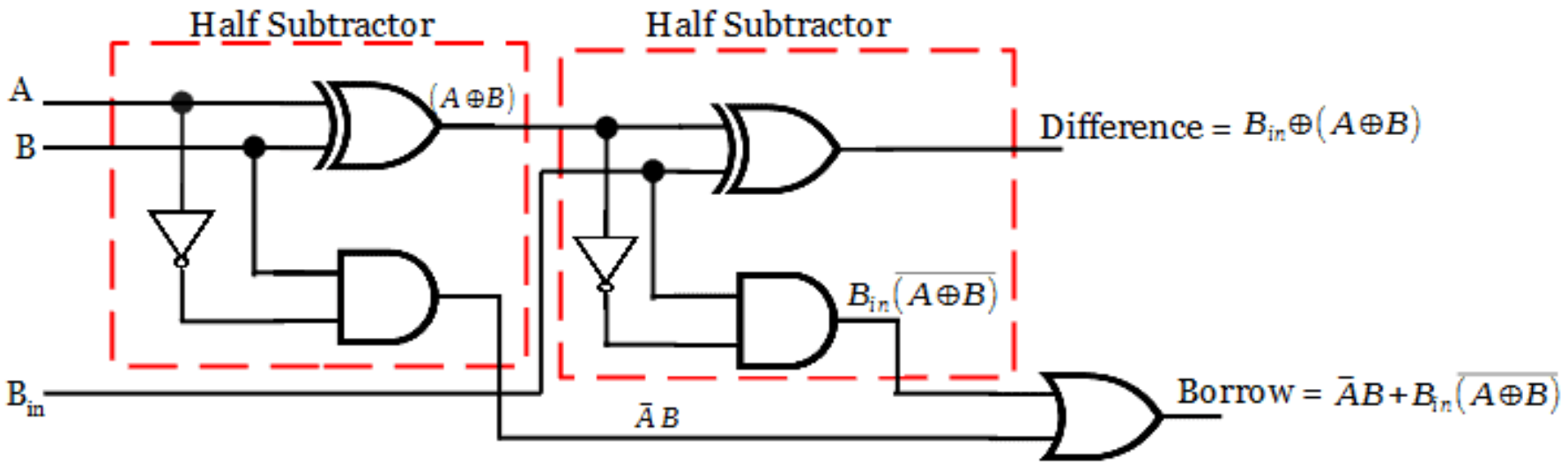


The boolean expression for borrow output is further simplified as below

$$
\begin{aligned}
\text{Borrow} &= \bar{A}\,B + \bar{A}\,B_{in} + B\,B_{in} \\
&= \bar{A}\,B + \bar{A}\,B_{in}(B+\bar{B}) + B\,B_{in}(A+\bar{A}) \\
&= \bar{A}\,B + \bar{A}\,B\,B_{in} + \bar{A}\,\bar{B}\,B_{in} + A\,B\,B_{in} + \bar{A}\,B\,B_{in} \\
&= \bar{A}\,B(1+B_{in}+B_{in}) + \bar{A}\,\bar{B}B_{in} + A\,B\,B_{in} \\
&= \bar{A}\,B + \bar{A}\,\bar{B}\,B_{in} + A\,B\,B_{in} \\
&= \bar{A}\,B + B_{in}(\bar{A}\,\bar{B} + A\,B) \\
&= \bar{A}\,B + B_{in}(A \odot B) \\
&= \bar{A}\,B + B_{in}\overline{(A \oplus B)}
\end{aligned}
$$

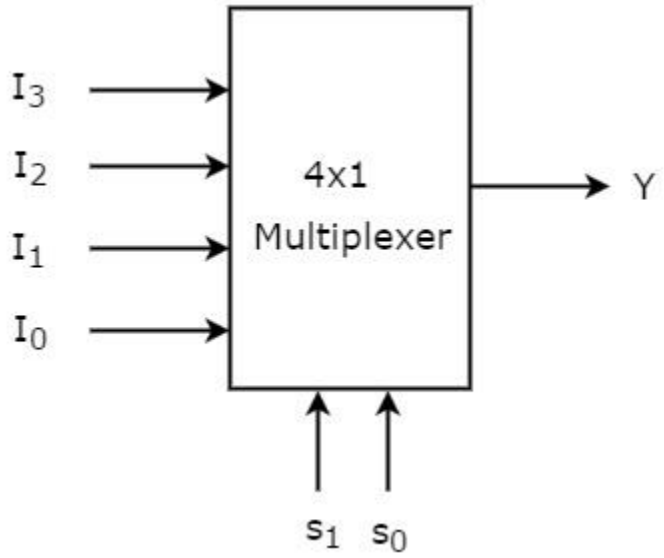
$A+A=A$

$A+1=1$



Combinational Circuits

- Multiplexers(Data Selectors):
  - The term ‘multiplex’ means ‘many to one’
  - Process of transmitting a large number of information over a single line
  - Multiplexers have several input lines and one output line
  - Selection of a particular input line is controlled by a set of selection lines



Block diagram of a multiplexer

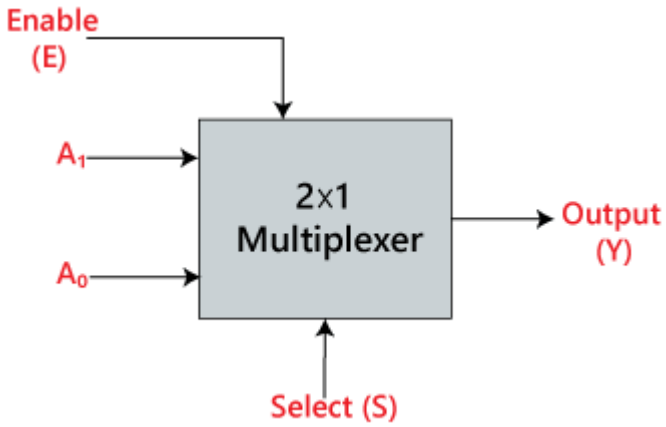
This is a block diagram of a multiplexer with 4 input lines, one output line and 2 select lines

The selection lines decide the number of input lines of a particular multiplexer

If number of select lines is m, the number of input lines is 2^m

• 2-to-1 MUX

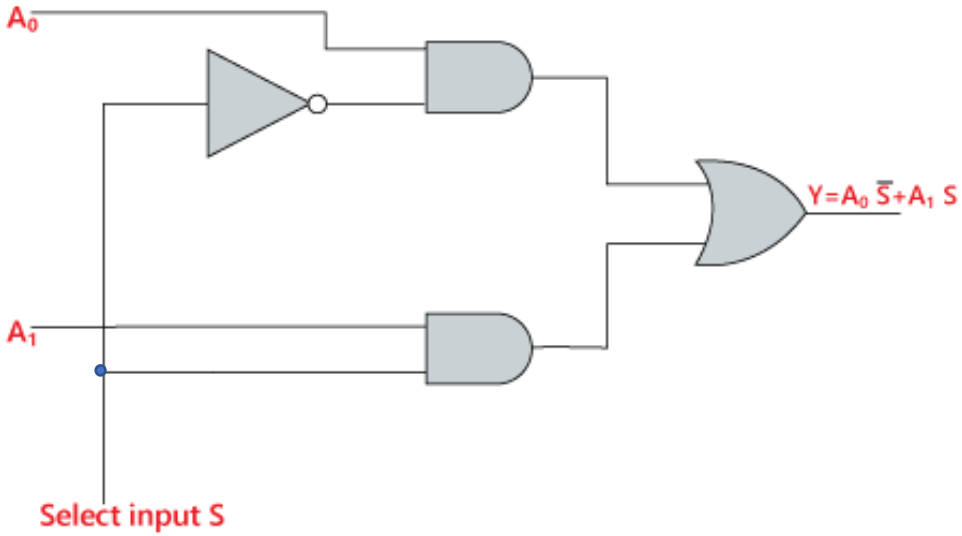
In 2×1 multiplexer, there are only two inputs, i.e., A0 and A1, 1 selection line, i.e., S0 and single outputs, i.e., Y. On the basis of the combination of inputs which are present at the selection line S0, one of these 2 inputs will be connected to the output. The block diagram and the truth table of the 2×1 multiplexer are given below.



INPUTS	Output
S_0	Y
0	A_0
1	A_1

The logical expression of the term Y is:

$$Y = \overline{S_0}.A_0 + S_0.A_1$$



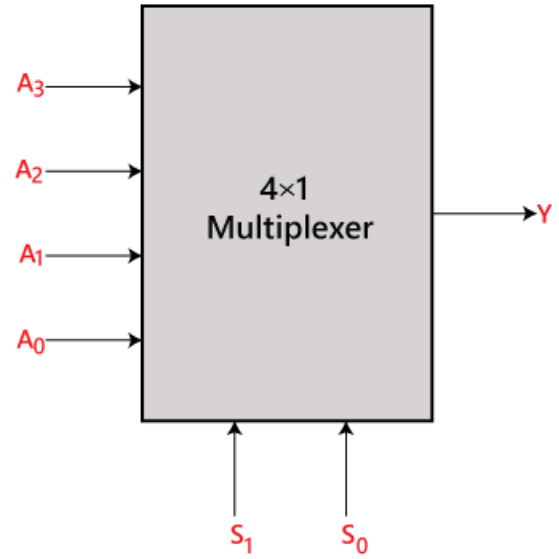
- 4-to-1 MUX

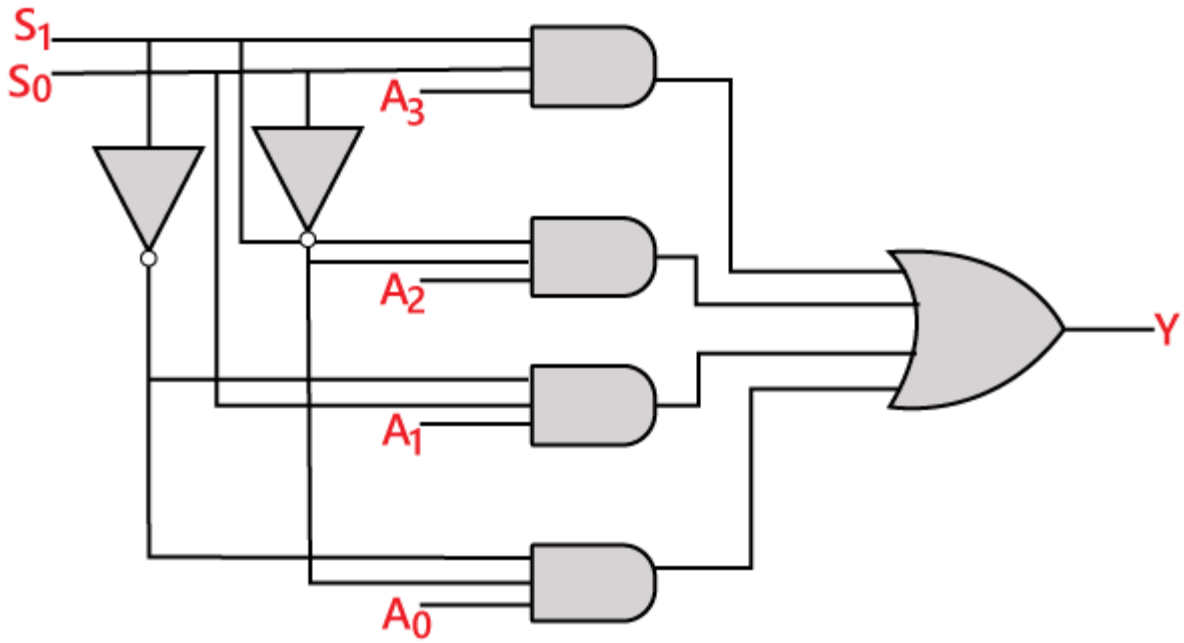
- In the 4×1 multiplexer, there is a total of four inputs, i.e., A_0, A_1, A_2, and A_3, 2 selection lines, i.e., S_0 and S_1 and single output, i.e., Y. On the basis of the combination of inputs that are present at the selection lines S^0 and S_1, one of these 4 inputs are connected to the output.

INPUTS		Output
S_1	S_0	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

The logical expression of the term Y is as follows:

$$Y=\overline{S}_1.\overline{S}_0\ A_0+\overline{S}_1\ S_0\ A_1+S_1\ \overline{S}_0\ A_2+S_1\ S_0\ A_3$$





To demonstrate the operation of the circuit-

- Let us consider the case when S1S0=**00**, they are applied to the select lines
- The AND gate associated with D0 will have two of its inputs equal to 1 and the third input connected to D_0.
- The other three AND gates have 0 atleast in one of their inputs, which makes their output **0**
- Hence, the OR output Y is equal to the value of D_0
- D_0 appears on the data output line Y
- If S1S0 = **01** are applied to the select lines, the data on the input line D1 appears on the data output line
- If S1S0 = **10**, the data on the input line D2 appears on the output line
- Similarly, S1S0= **11**, data on D2 is switched to the output line

De-multiplexer

A De-multiplexer is a combinational circuit that has only 1 input line and 2^N output lines.

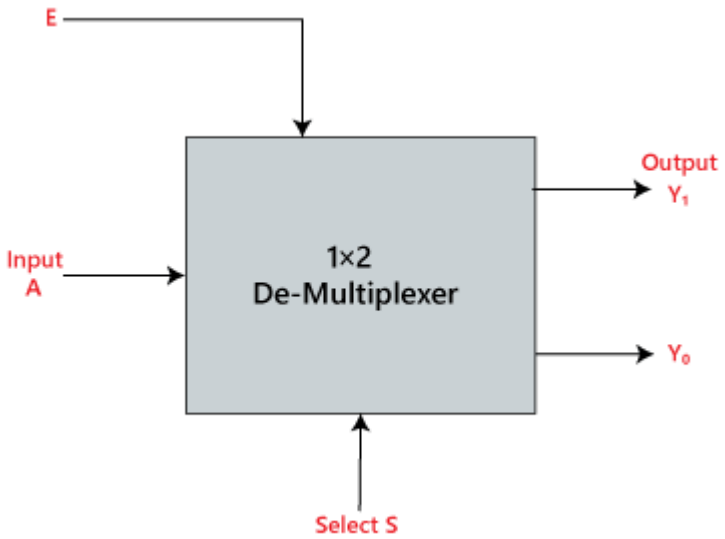
Simply, the de-multiplexer is a single-input and multi-output combinational circuit.

The information is received from the single input lines and directed to the output line. On the basis of the values of the selection lines, the input will be connected to one of these outputs.

De-multiplexer is opposite to the multiplexer

1x2 De-multiplexer:

In the 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 the basis of the selection value, the input will be connected to one of the outputs. The block diagram and the truth table of the 1×2 multiplexer are given below.

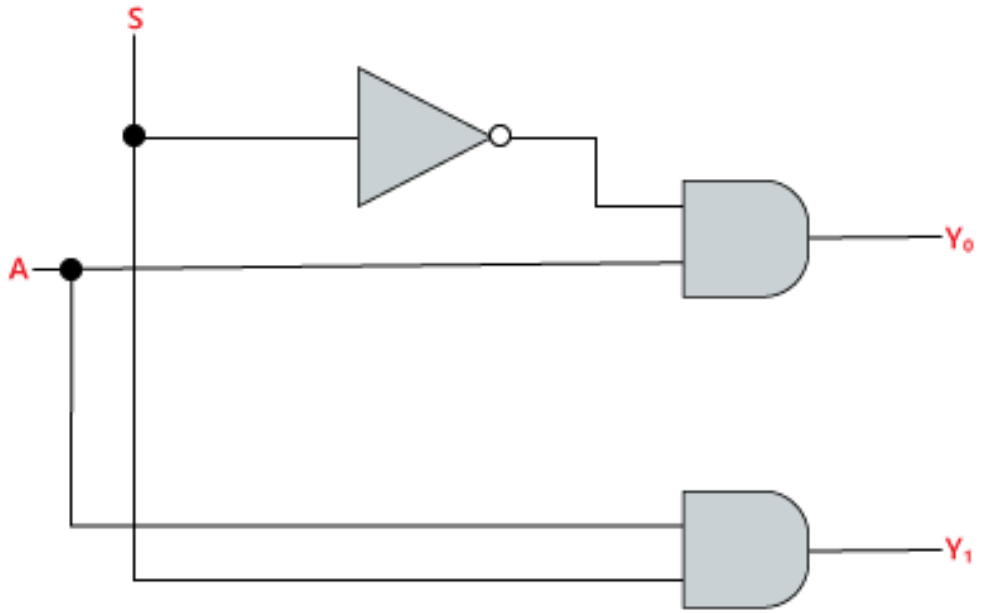


INPUTS	Output	
S_0	Y_1	Y_0
0	0	A
1	A	0

The logical expression of the term Y is as follows:

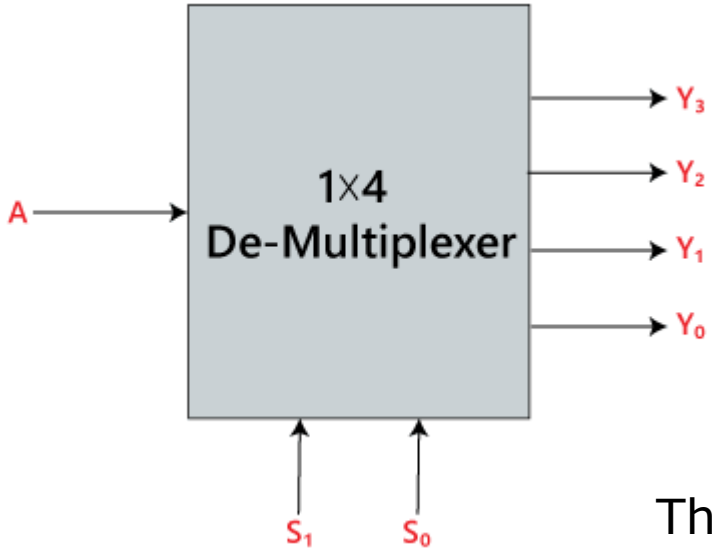
$Y_0 = \bar{S}_0.A$ and $Y_1 = S_0.A$

Logical circuit of the above expressions is given below:



• 1-to-4 Demultiplexer

In 1 to 4 De-multiplexer, there are total of four outputs, i.e., Y_0, Y_1, Y_2, and Y_3, 2 selection lines, i.e., S_0 and S_1 and single input, i.e., A. On the basis of the combination of inputs which are present at the selection lines S_0 and S_1, the input be connected to one of the outputs. The block diagram and the truth table of the 1×4 de-multiplexer are given below.



INPUTS		Output			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	A
0	1	0	0	A	0
1	0	0	A	0	0
1	1	A	0	0	0

The logical expression of the term Y is as follows:

$$Y_0 = \bar{S}_1 \bar{S}_0 A \qquad Y_1 = \bar{S}_1 S_0 A$$

$$Y_2 = S_1 \bar{S}_0 A \qquad Y_3 = S_1 S_0 A$$

Logical circuit of the above expressions is given below: