UNIT 5 and 6
Combinational Logic

Prepared By

Er.Harendra Bikram Shah

(Lecturer)

Department of Computer Engineering

OUTLINES

- Introduction
- Adders and Subtractors
- CODE conversion(BCD to excess-3, 8 4 -2 -1 code to bcd, 2421 code to 8 4 -2 1 code)
- Decoder and Encoder
- Multiplexer and De-multiplexer
- BCD to 7 segment decoder
- ROM and PLA

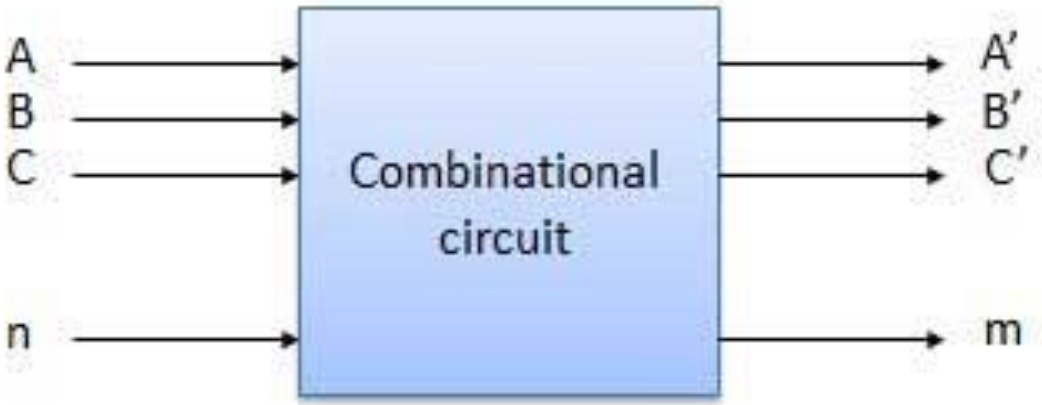
Introduction

- Combinational circuit/combinational logic is a type of logic circuit which is the combination of different gates in the circuit where the output is a pure function of the present input only.

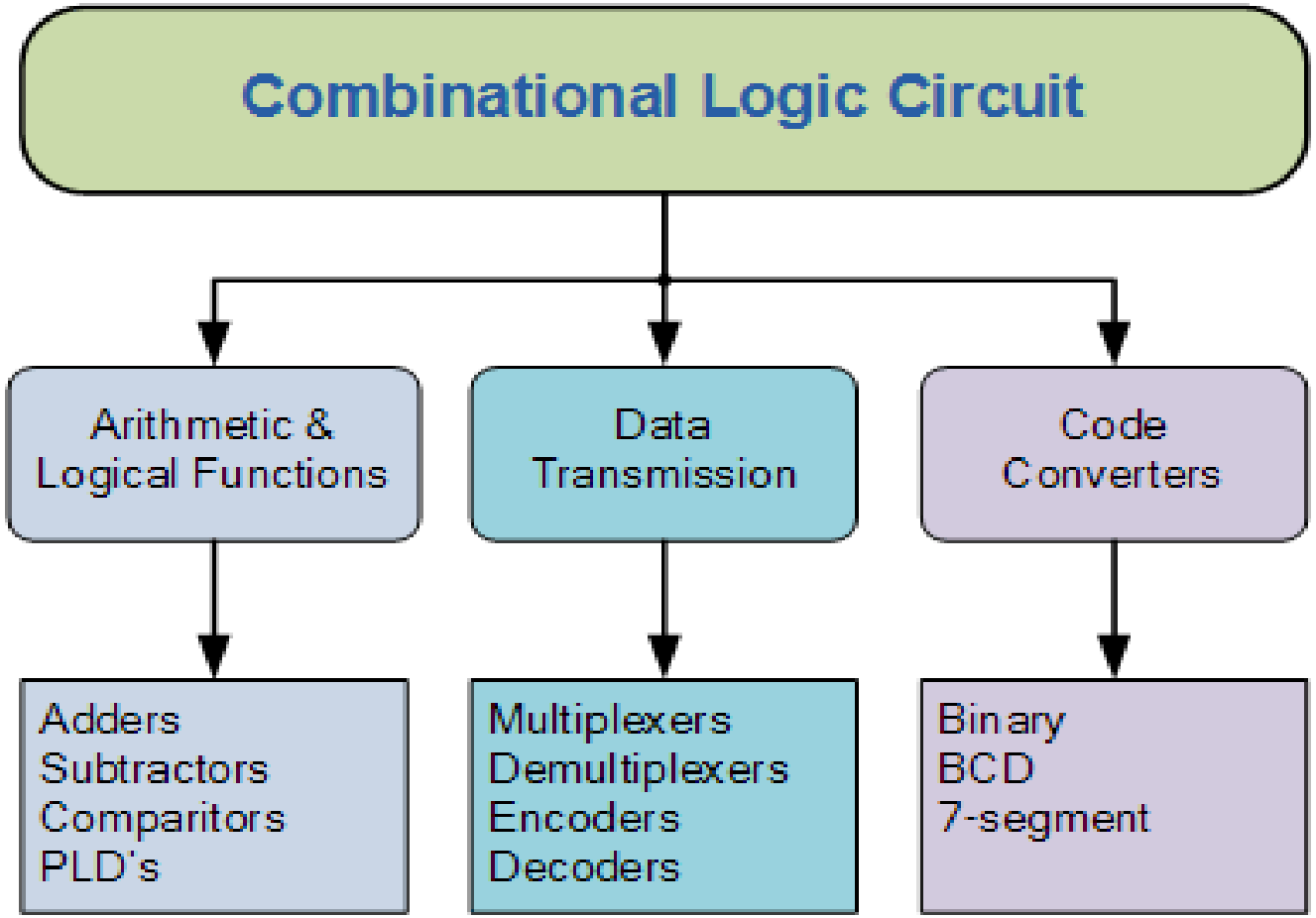
characteristics of combinational circuits are following :-

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

Block diagram



Classification of Combinational Logic



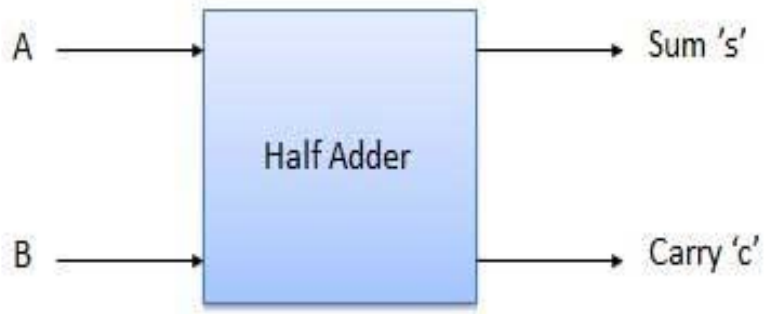
Adders

- An adder is a device that will add together two bits and give the result as the output.
- There are two kinds of adders - half adders and full adders

Half Adder

- Half adder is a combinational logic circuit with two inputs and two outputs.
- The half adder is able to add two single binary digits and provide the output plus a carry value
- This circuit has two outputs carry and sum.
- A half adder is a type of adder, an electronic circuit that performs the addition of numbers.
- The common representation uses a XOR logic gate and an AND logic gate.

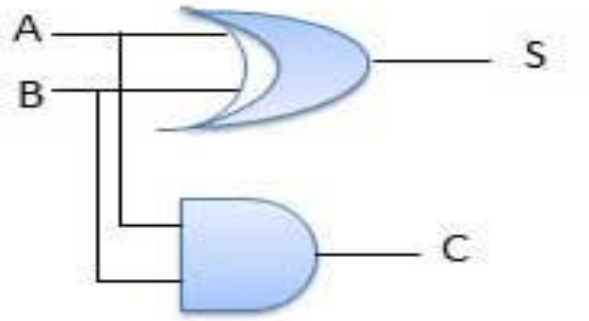
Block diagram



Truth Table

Inputs		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuit Diagram/combinational circuit diagram



Logic Expression

Sum= A XOR B

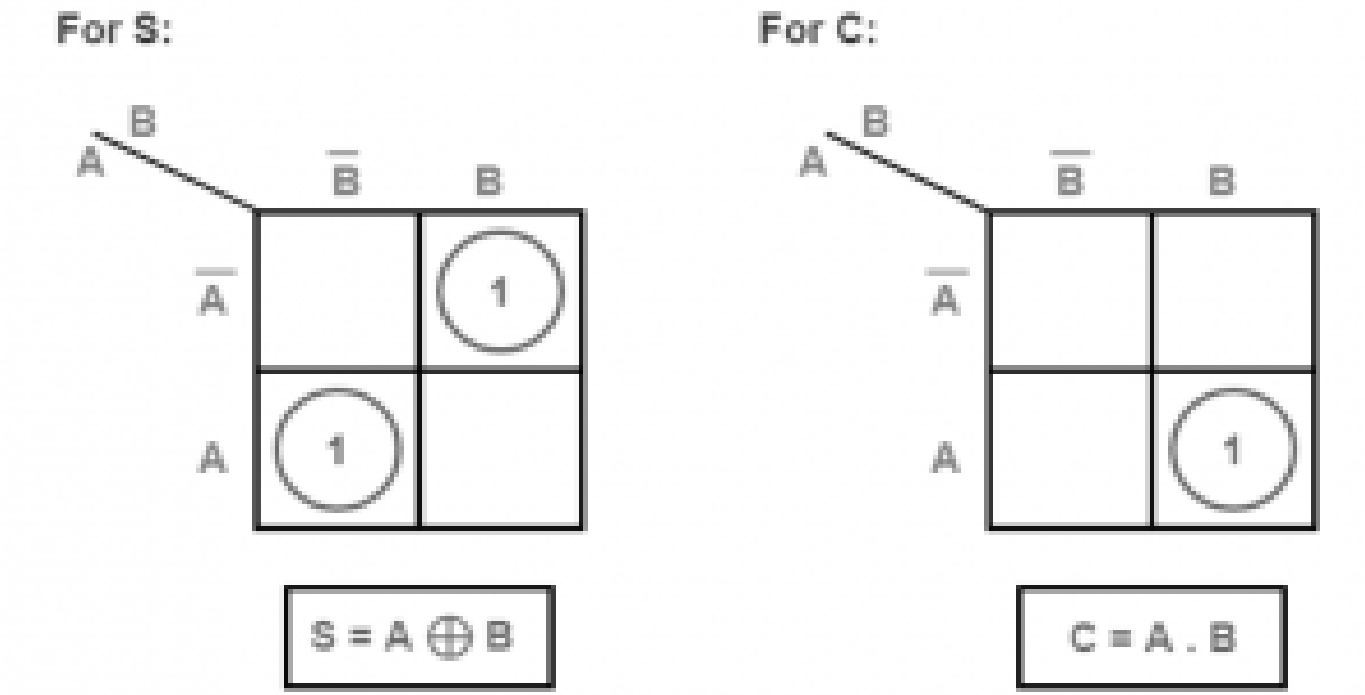
Carry = A AND B

Logic Expression

Sum= A XOR B

Carry = A AND B

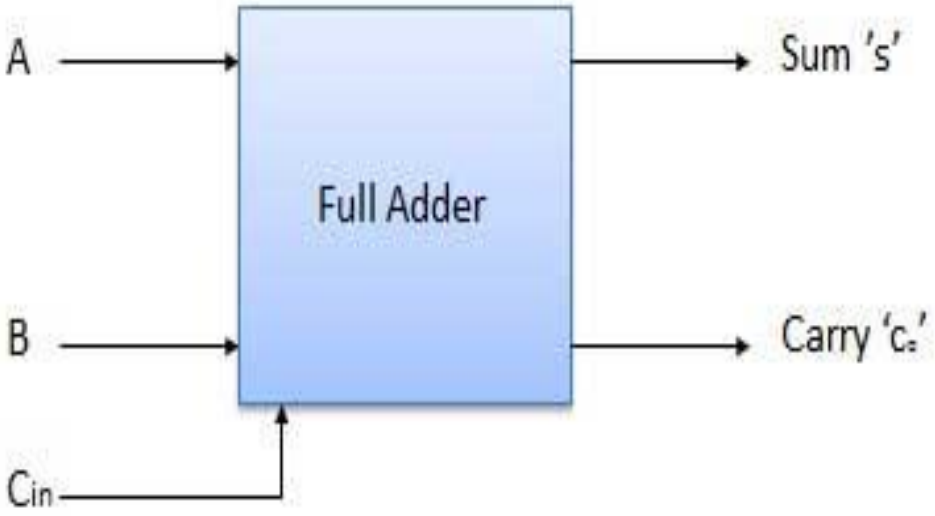
The half adder K-map is



Full Adders

- Full adder is developed to overcome the drawback of Half Adder circuit.
- Full adders is a combinational circuit that has 3 inputs and two outputs i.e. A,B ,Cin and Sum and Carry out
- It can add two one-bit numbers A and B, and carry c.

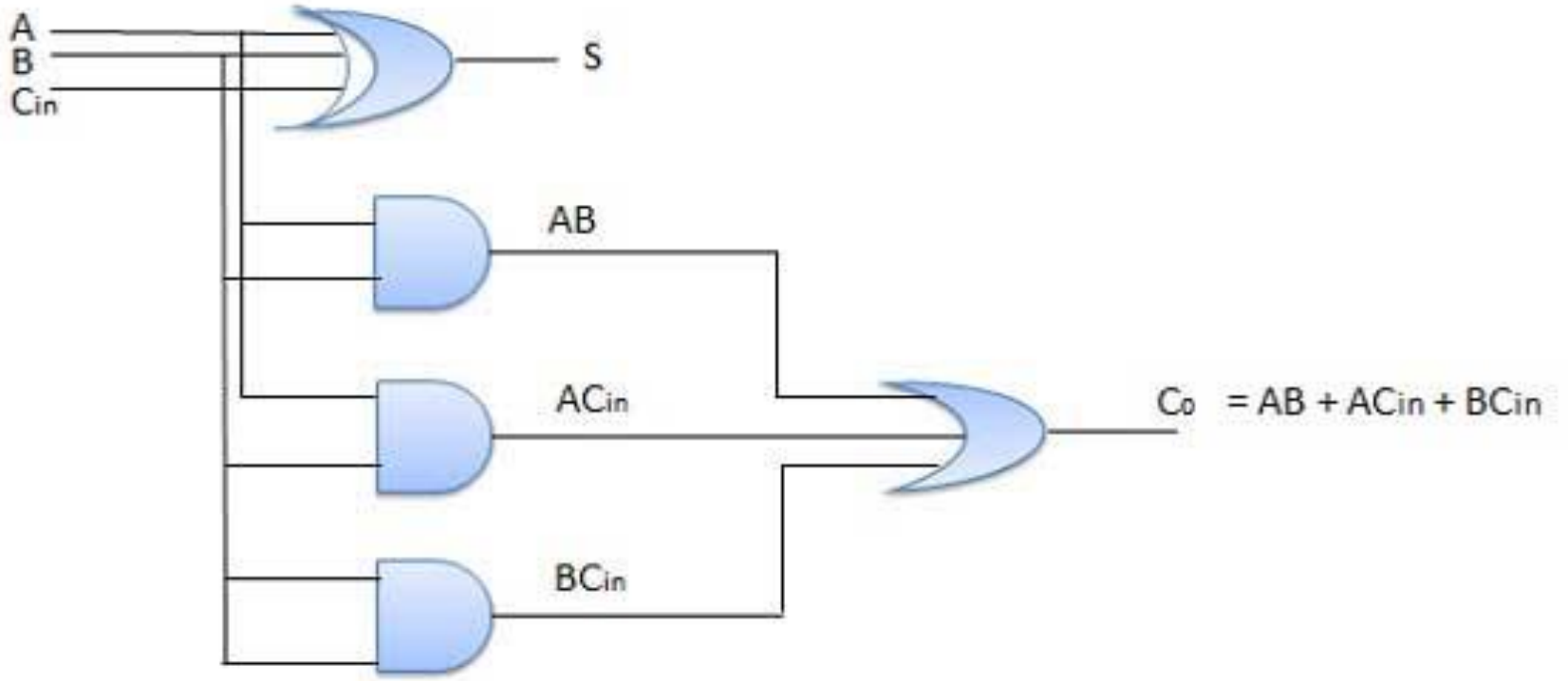
Block diagram



Truth Table

Inputs			Output	
A	B	Cin	S	Co
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

Circuit Diagram/combinational circuit diagram



Logic Expression

SUM = (A XOR B) XOR Cin = (A ⊕ B) ⊕ Cin

CARRY-OUT = A AND B OR Cin(A XOR B) = A.B + Cin(A ⊕ B)

Logic Expression

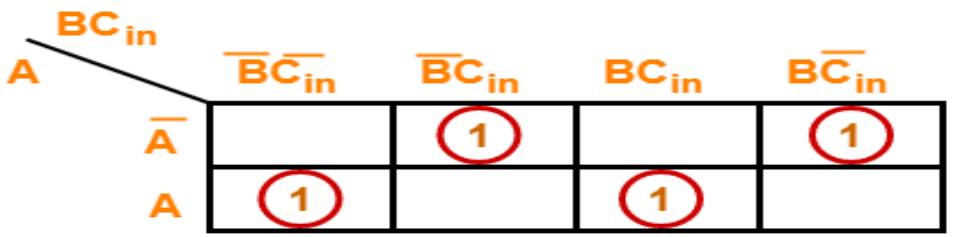
SUM = (A XOR B) XOR Cin = (A $\oplus$ B) $\oplus$ Cin

CARRY-OUT = A AND B OR Cin(A XOR B)

= A.B + Cin(A $\oplus$ B)

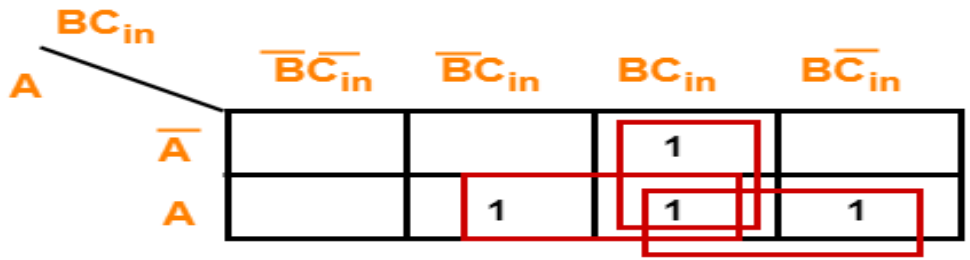
The full adder K-Map is

For S:



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

For C_{in}:



$$C_{out} = AB + BC_{in} + C_{in}A$$

Implementation of Full adder using two half adder and one OR gate

Full Adder Truth Table:

Inputs			Outputs	
A	B	C – IN	Sum	C - Out
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

Logical Expression for SUM:

= A’ B’ C-IN + A’ B C-IN’ + A B’ C-IN’ + A B C-IN

= C-IN (A’ B’ + A B) + C-IN’ (A’ B + A B’)

= C-IN XOR (A XOR B)

= (1,2,4,7)

Logical Expression for C-OUT:

= A’ B C-IN + A B’ C-IN + A B C-IN’ + A B C-IN

= A B + B C-IN + A C-IN

= (3,5,6,7)

Another form in which C-OUT can be implemented:

= A B + A C-IN + B C-IN (A + A’)

= A B C-IN + A B + A C-IN + A’ B C-IN

= A B (1 +C-IN) + A C-IN + A’ B C-IN

= A B + A C-IN + A’ B C-IN

= A B + A C-IN (B + B’) + A’ B C-IN

= A B C-IN + A B + A B’ C-IN + A’ B C-IN

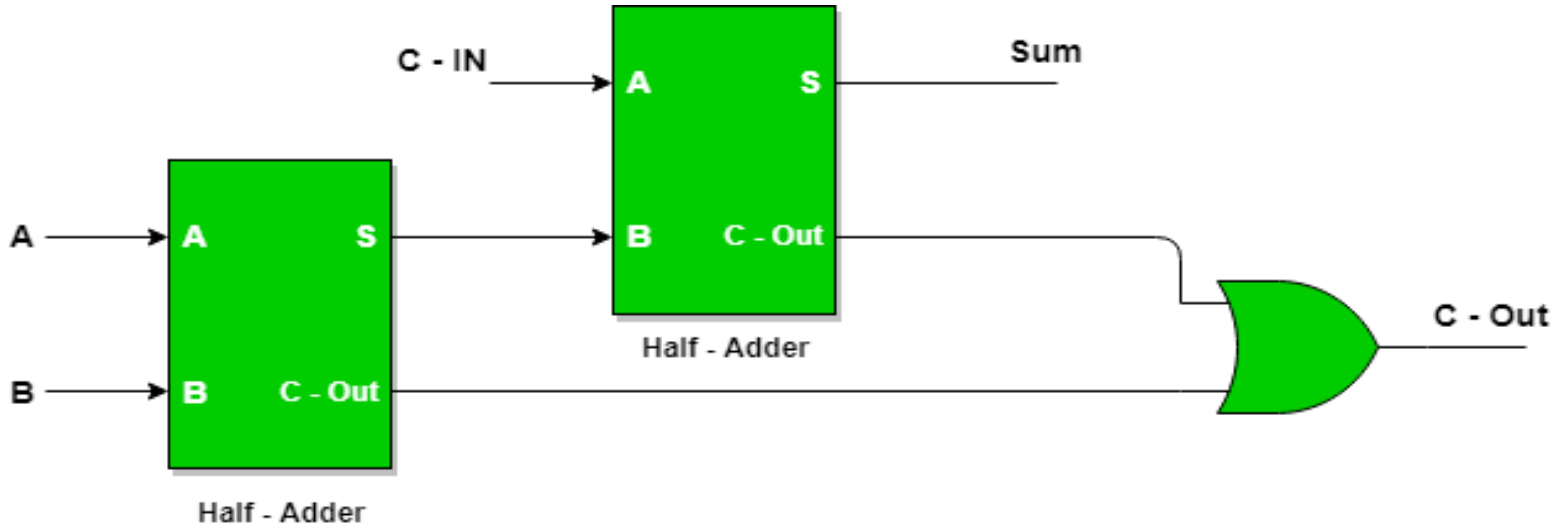
= A B (C-IN + 1) + A B’ C-IN + A’ B C-IN

= A B + A B’ C-IN + A’ B C-IN

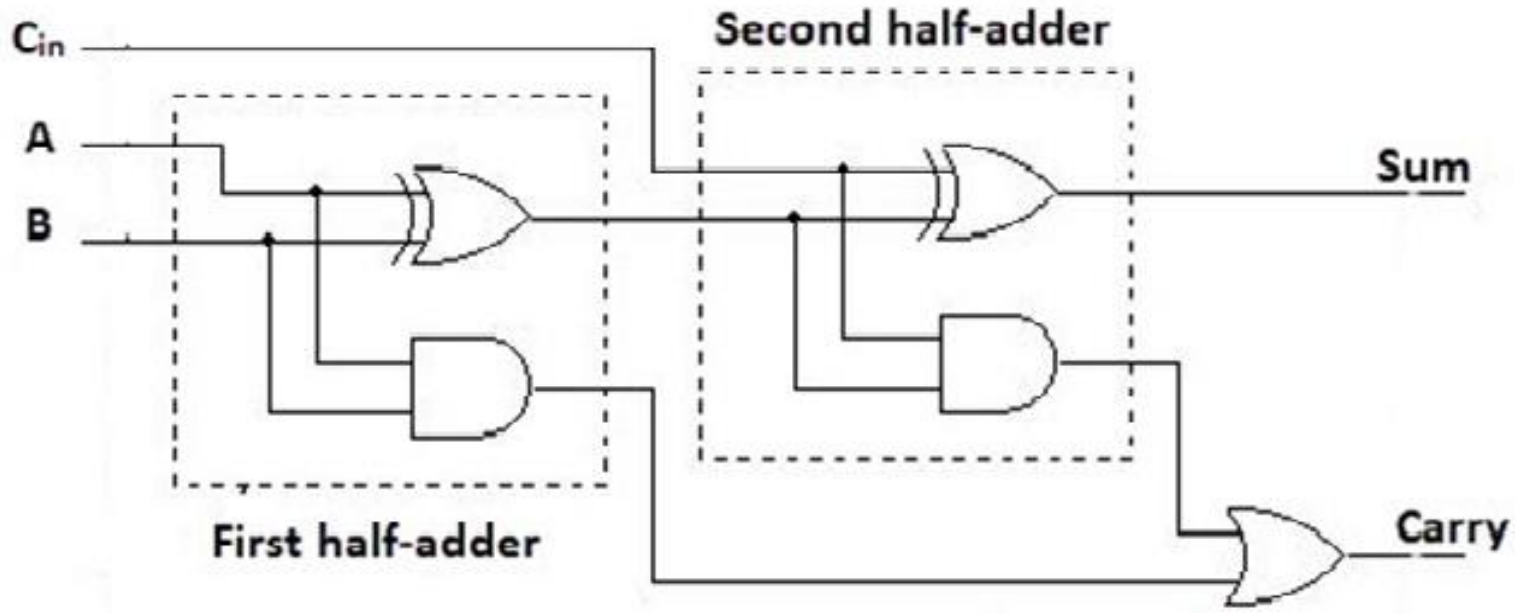
= AB + C-IN (A’ B + A B’)

Therefore COUT = AB + C-IN (A EX – OR B)

Block Diagram:



Combinational Circuit Diagram:

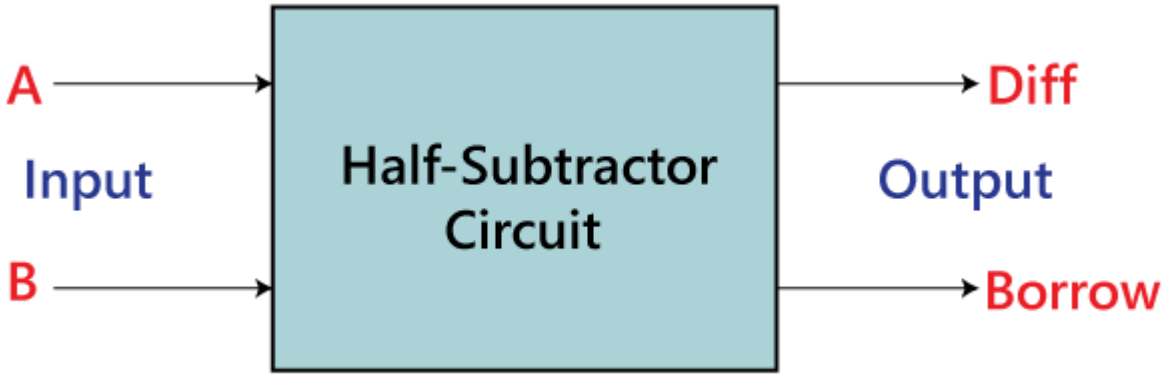


Subtractors

- An electronic Combinational logic circuit for calculating the difference between two binary numbers, the minuend and the number to be subtracted, the subtrahend.
- There are two types of subtractors ,Full and Half.

HALF SUBTRACTORS

- The half subtractor is also a building block for subtracting two binary numbers.
- It has two inputs and two outputs.
- This circuit is used to subtract two single bit binary numbers A and B.
- The 'diff' and 'borrow' are two output states of the half subtractor.



Block diagram

HALF SUBTRACTOR

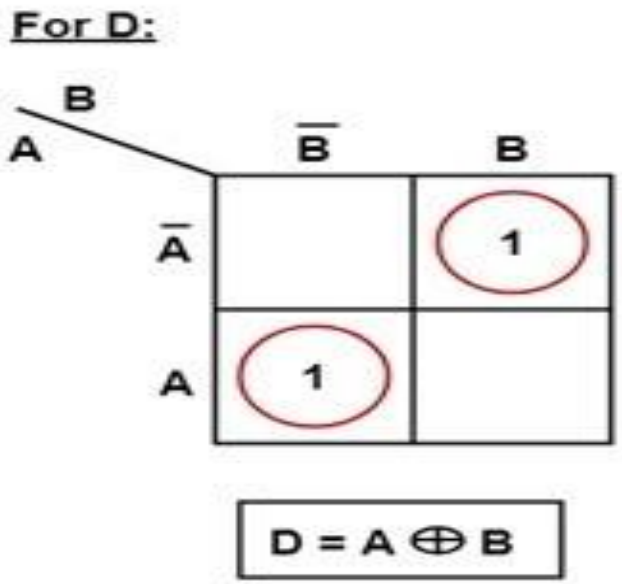
Truth Table

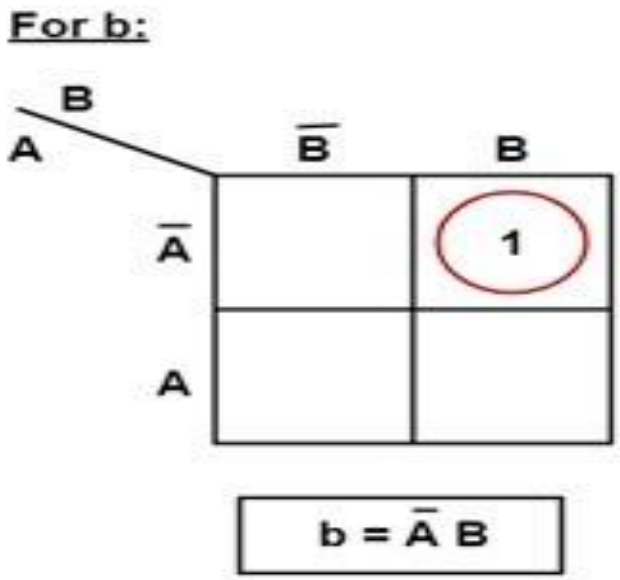
Inputs		Outputs	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

In above truth table

- A' and 'B' are the input variables whose values are going to be subtracted.
- The 'Diff' and 'Borrow' are the variables whose values define the subtraction result, i.e., difference and borrow.
- The first two rows and the last row, the difference is 1, but the 'Borrow' variable is 0.
- The third row is different from the remaining one. When we subtract the bit 1 from the bit 0, the borrow bit is produced.

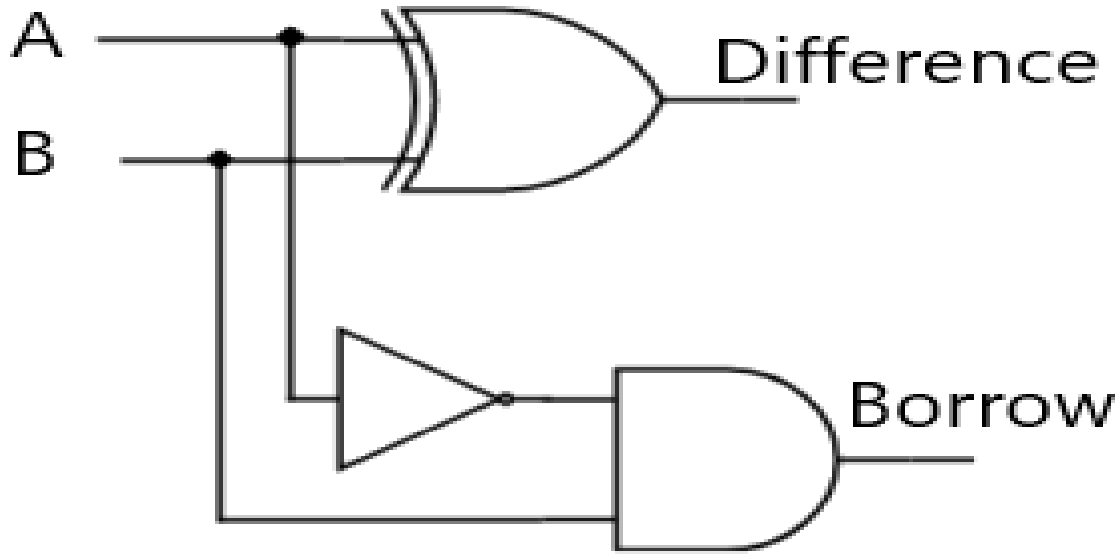
HALF SUBTRACTOR





Logical Expression

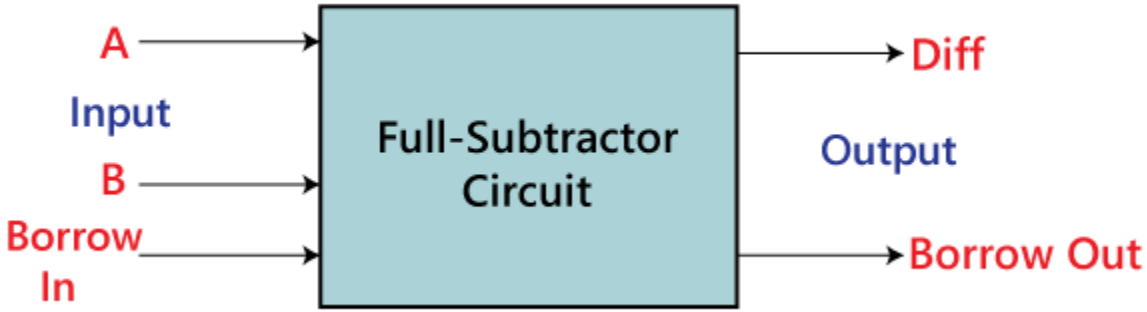
Difference = A XOR B
Borrow = A'B



Logic/Combinational Circuit of HALF SUBTRACTOR

FULL SUBTRACTOR

- A full subtractor is a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit.
- This circuit has three inputs and two outputs.
- The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively.
- The two outputs, D and Bout represent the difference and output borrow, respectively.



Block diagram

FULL SUBTRACTORS

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

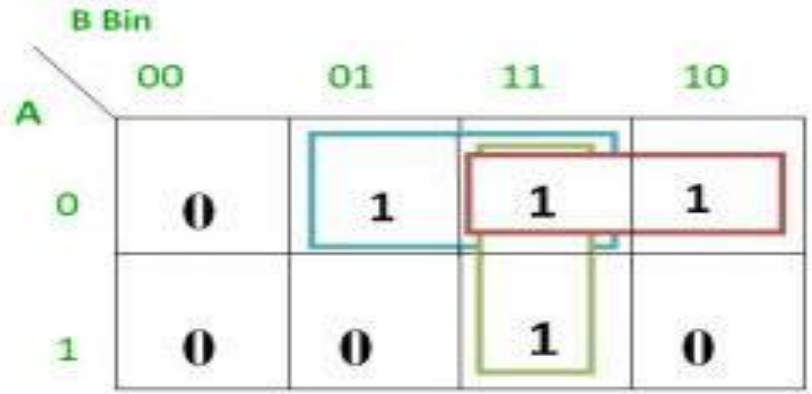
In the above table,

- 'A' and' B' are the input variables. These variables represent the two significant bits that are going to be subtracted.
- '$Borrow_{in}$' is the third input which represents borrow.
- The 'Diff' and 'Borrow' are the output variables that define the output values.
- The eight rows under the input variable designate all possible combinations of 0 and 1 that can occur in these variables.

K-map for Diff (D)

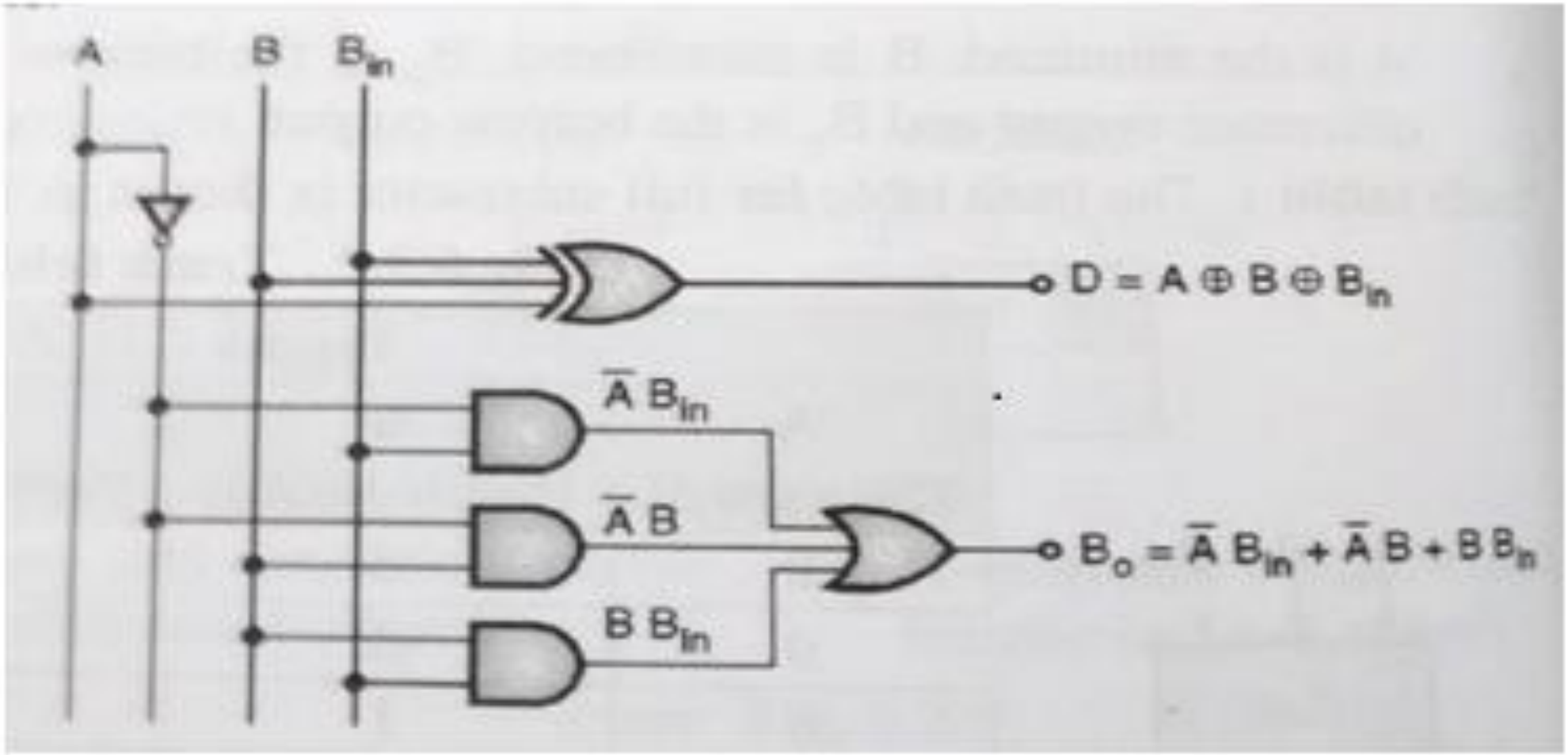
D=A'B'Bin + AB'Bin' + A'BBin' + ABBin

K-map for Borrow (Bout)



Bout=A'Bin + A'B + BBin

Logic/Combinational Circuit for Full Subtractor –



IMPLEMENTATION OF FULL SUBTRACTOR USING TWO HALF SUBTTRACTOR AND ONE OR GATE

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

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

IMPLEMENTATION OF FULL SUBTRACTOR USING TWO HALF SUBTTRACTOR AND ONE OR GATE

Logical expression for borrow –

Bout = A’B’Bin + A’BBin’ + A’BBin + ABBin

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

= A’Bin(B + B’) + A’B(Bin + Bin’) + BBin(A + A’)

= A’Bin + A’B + BBin

OR

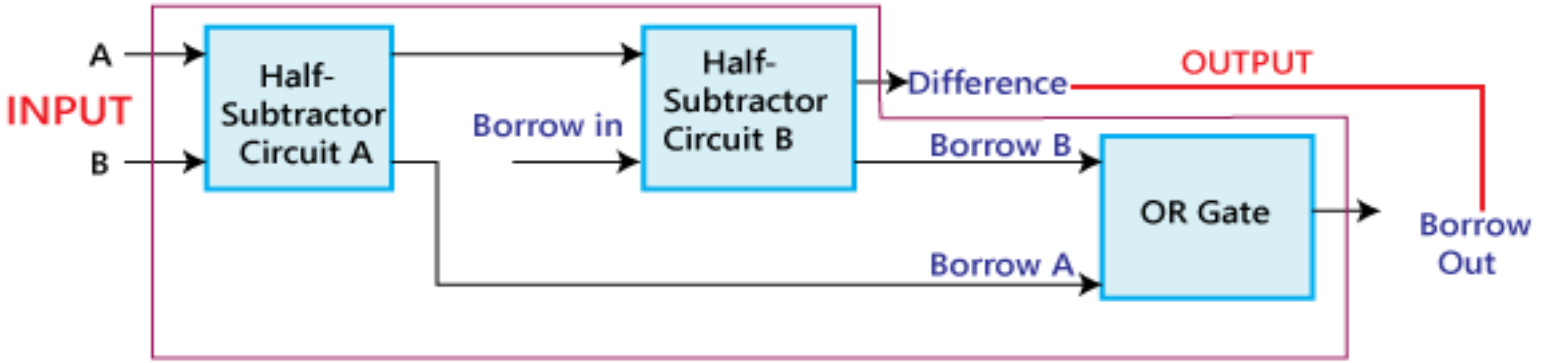
Bout = A’B’Bin + A’BBin’ + A’BBin + ABBin

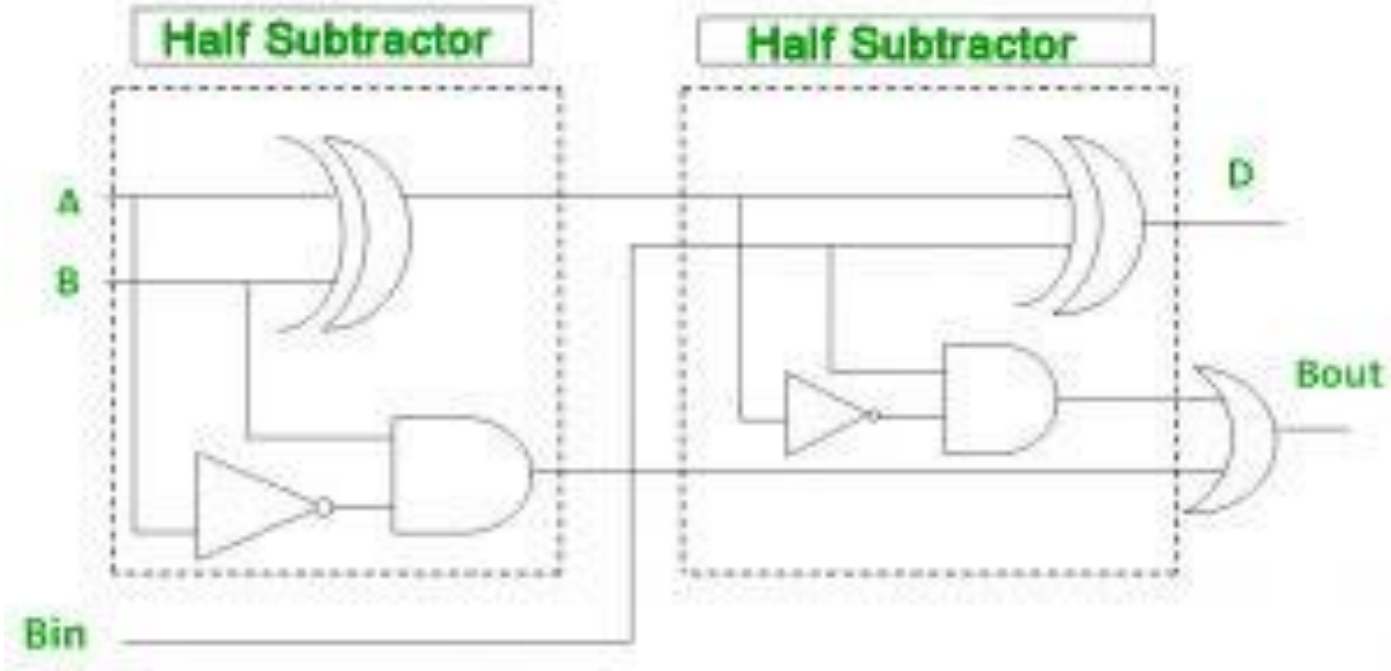
= Bin(AB + A’B’) + A’B(Bin + Bin’)

= Bin(A XNOR B) + A’B

= Bin (A XOR B)’ + A’B

IMPLEMENTATION OF FULL SUBTRACTOR USING TWO HALF SUBTTRACTOR AND ONE OR GATE



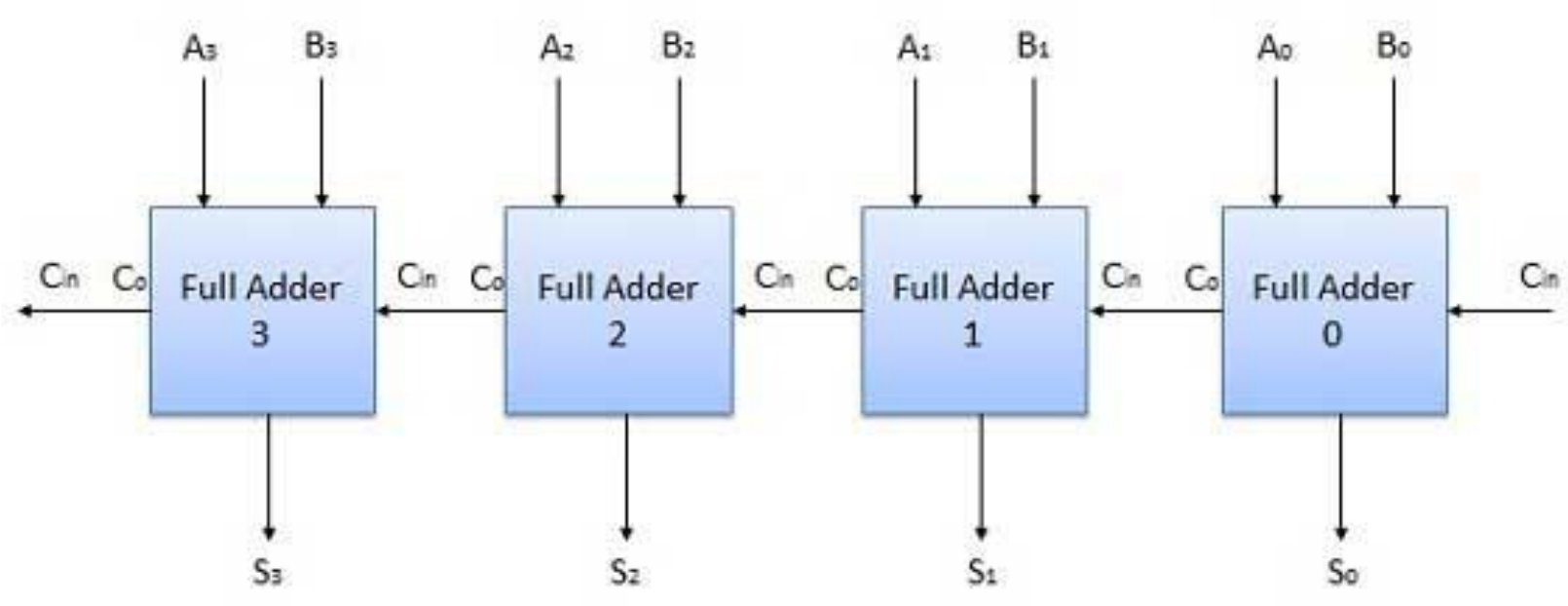


N-Bit Parallel Adder/Nibble Adder

The Full Adder is capable of adding only two single digit binary number along with a carry input.
But in practical we need to add binary numbers which are much longer than just one bit.
To add two n-bit binary numbers we need to use the n-bit parallel adder. It uses a number of full adders in cascade.
The carry output of the previous full adder is connected to carry input of the next full adder.

4 Bit Parallel Adder

In the block diagram, A0 and B0 represent the LSB of the four bit words A and B.
Hence Full Adder-0 is the lowest stage. Hence its Cin has been permanently made 0.
The rest of the connections are exactly same as those of n-bit parallel adder is shown in fig.
The four bit parallel adder is a very common logic circuit.
Block diagram



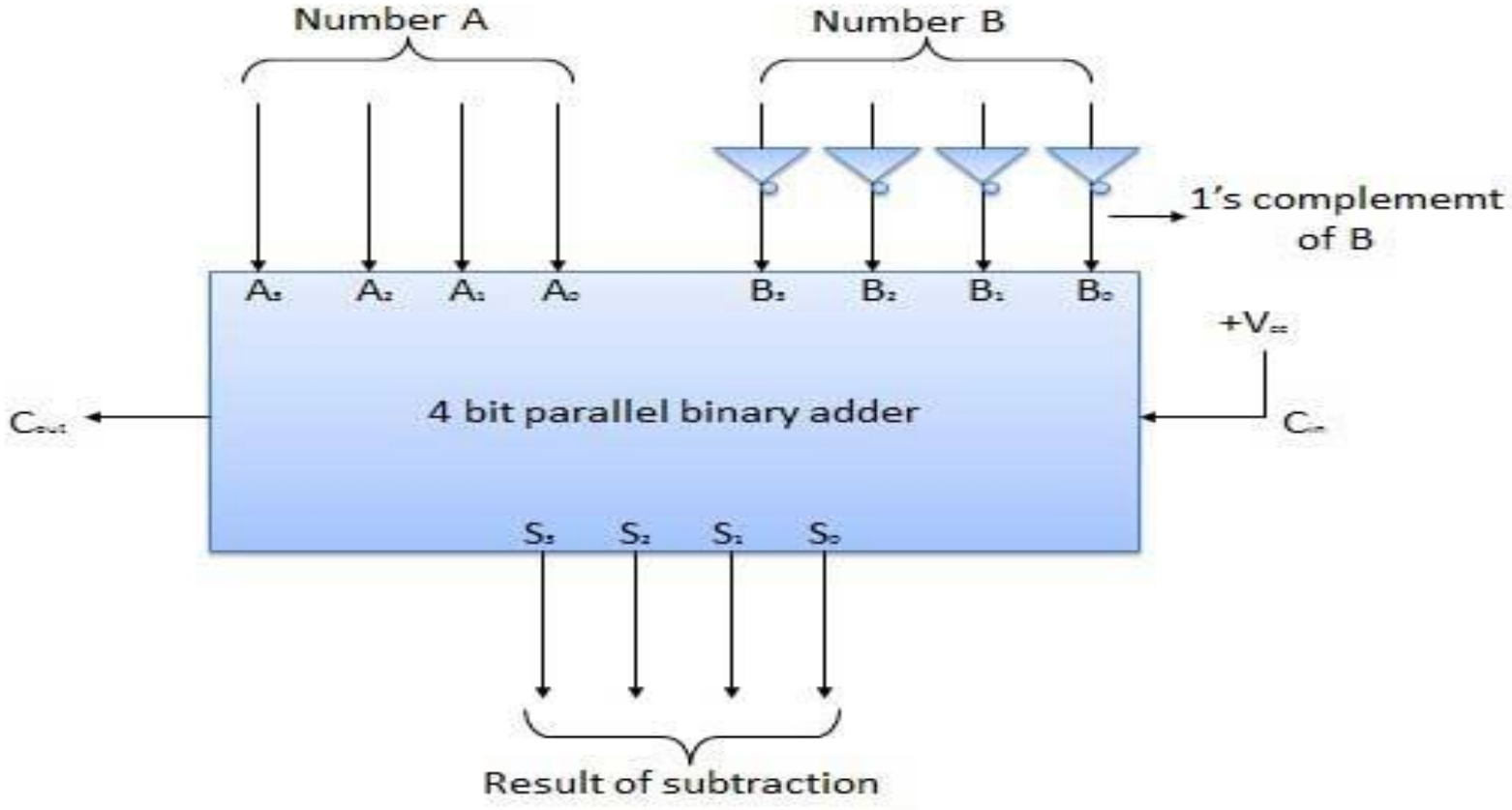
N-Bit Parallel Subtractor

The subtraction can be carried out by taking the 1's or 2's complement of the number to be subtracted.
For example we can perform the subtraction (A-B) by adding either 1's or 2's complement of B to A.
That means we can use a binary adder to perform the binary subtraction.

4 Bit Parallel Subtractor

The number to be subtracted (B) is first passed through inverters to obtain its 1's complement.
The 4-bit adder then adds A and 2's complement of B to produce the subtraction. S3 S2 S1 S0 represents the result of binary subtraction (A-B) and carry output Cout represents the polarity of the result.
If A > B then Cout = 0 and the result of binary form (A-B) then Cout = 1 and the result is in the 2's complement form.

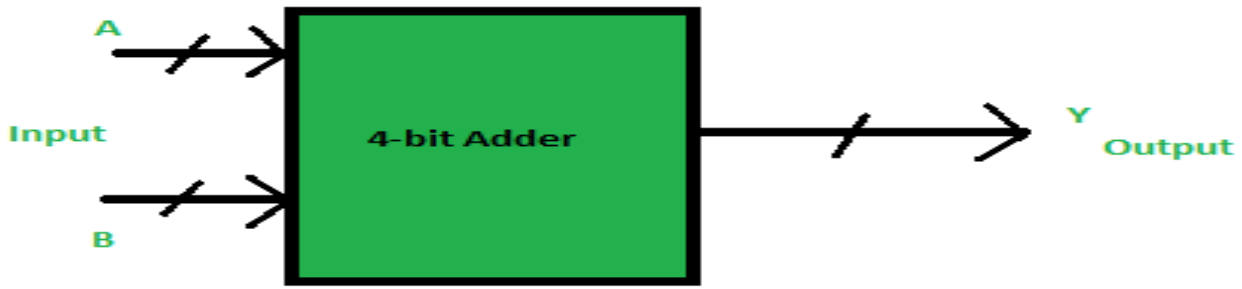
Block diagram



Decimal or BCD Adder

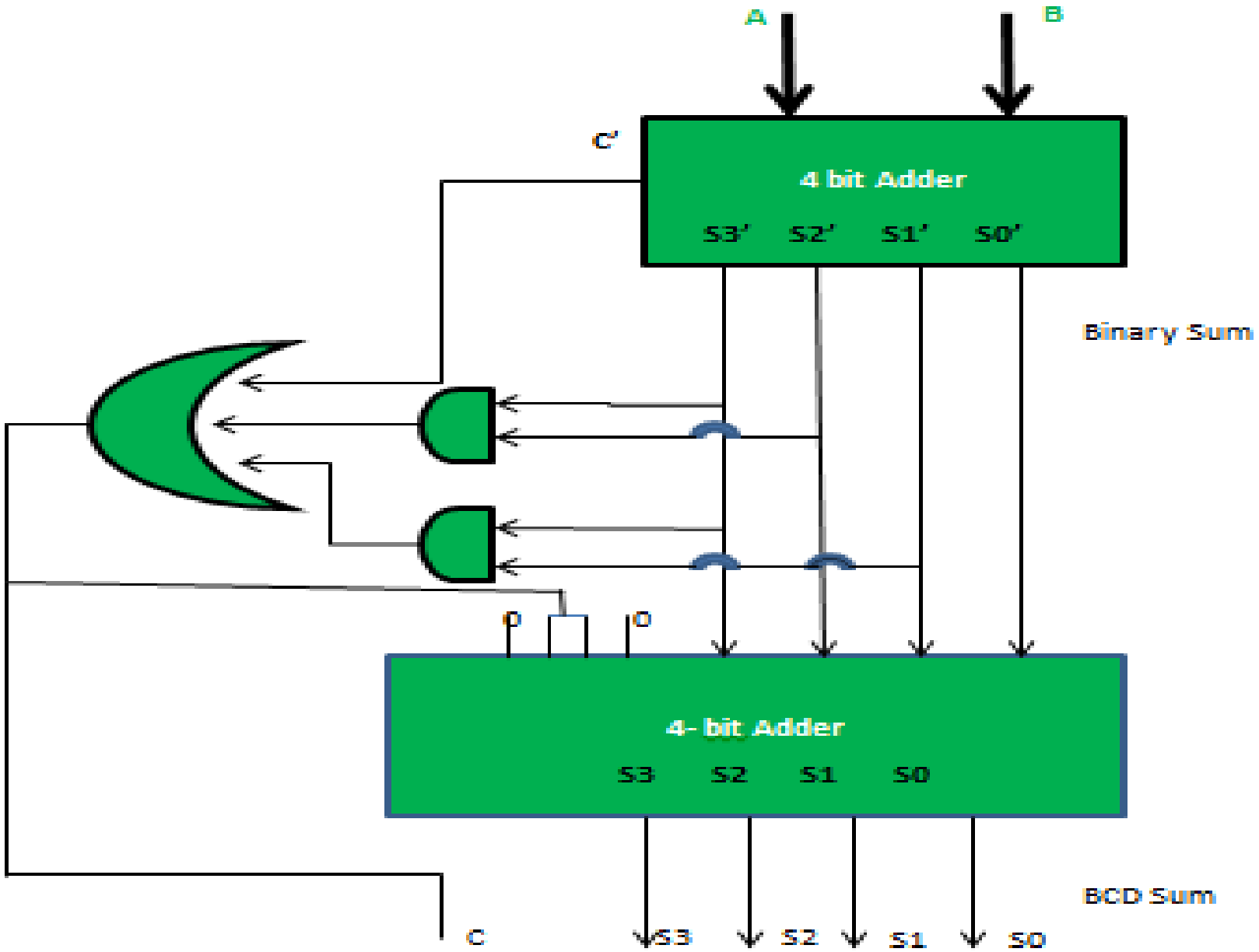
- The BCD-Adder is used in the computers and the calculators that perform arithmetic operation directly in the decimal number system.
- The BCD-Adder accepts the binary-coded form of decimal numbers.
- The Decimal-Adder requires a minimum of nine inputs and five outputs.

It is used to perform the addition of BCD numbers.
A BCD digit can have any of ten possible four-bit representations.
Suppose, we have two 4-bit numbers A and B.
The value of A and B can vary from 0(0000 in binary) to 9(1001 in binary) because we are considering decimal numbers.
The output will vary from 0 to 18 if we are not considering the carry from the previous sum.
But if we are considering the carry, then the maximum value of output will be 19 (i.e. 9+9+1 = 19).
When we are simply adding A and B, then we get the binary sum. Here, to get the output in BCD form, we will use BCD Adder.



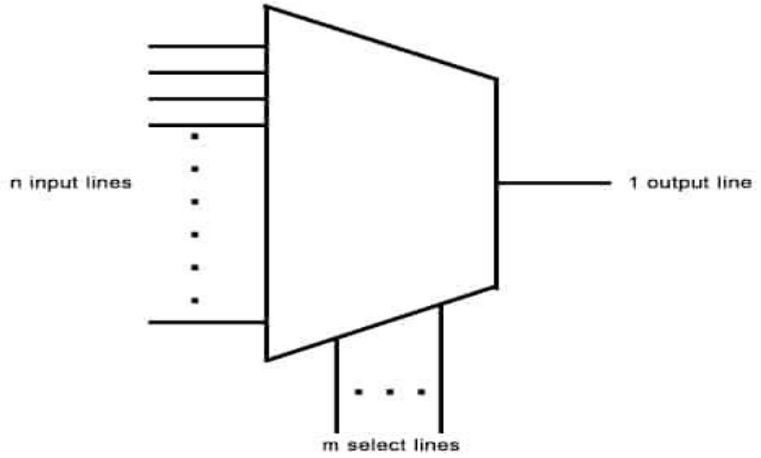
Decimal	Binary Sum					BCD Sum				
	C'	S3'	S2'	S1'	S0'	C	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

We are adding “0110” (=6) only to the second half of the table. The conditions are:
1.If C’ = 1 (Satisfies 16-19)
2.If S3′.S2′ = 1 (Satisfies 12-15)
3.If S3′.S1′ = 1 (Satisfies 10 and 11)
So, our logic is
C' + S3'.S2' + S3'.S1' = 1


A
B
C'
4 bit Adder
S3' S2' S1' S0'
Binary Sum
0
0
4- bit Adder
S3 S2 S1 S0
BCD Sum
C
S3
S2
S1
S0

Multiplexer

- A multiplexer is a combinational circuit that has 2^n input lines and a single output line.
- Simply, the multiplexer is a multi-input and single-output combinational circuit.
- The binary information is received from the input lines and directed to the output line.
- A multiplexer of 2^n inputs has n select lines that will be used to select input line to send to the output.
- Multiplexer is abbreviated as Mux.



There are various types of the multiplexer which are as follows:
They are
2 :1
4:1
8:1
16:1

Advantages and Disadvantages of Multiplexer

The advantages of multiplexer include the following.

- In multiplexer, the usage of a number of wires can be decreased
- It reduces the cost as well as the complexity of the circuit
- The implementation of a number of combination circuits can be possible by using a multiplexer
- Mux doesn't require K-maps & simplification
- The multiplexer can make the transmission circuit less complex & economical
- The multiplexer ability can be extended to switch audio signals, video signals, etc.
- The digital system reliability can be improved using a MUX as it decreases the number of exterior wired connections.
- MUX is used to implement several combinational circuits

The disadvantages of multiplexer include the following.

- Additional delays required within switching ports & I/O signals which propagate throughout the multiplexer.
- The ports which can be utilized at the same time have limitations
- Switching ports can be handled by adding the complexity of firmware
- The controlling of multiplexer can be done by using additional I/O ports.

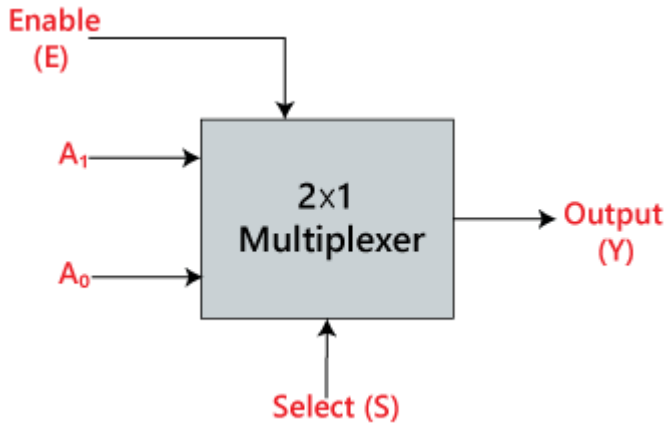
Applications of Multiplexers

- Communication System
- Computer Memory
- Transmission from the Computer System of a Satellite

2 X 1 /(2:1) MUX

2×1 Multiplexer:

- In 2×1 multiplexer, there are only two inputs, i.e., A_0 and A_1, 1 selection line, i.e., S_0 and single outputs, i.e., Y.
- On the basis of the combination of inputs which are present at the selection line S^0, 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.



Block Diagram

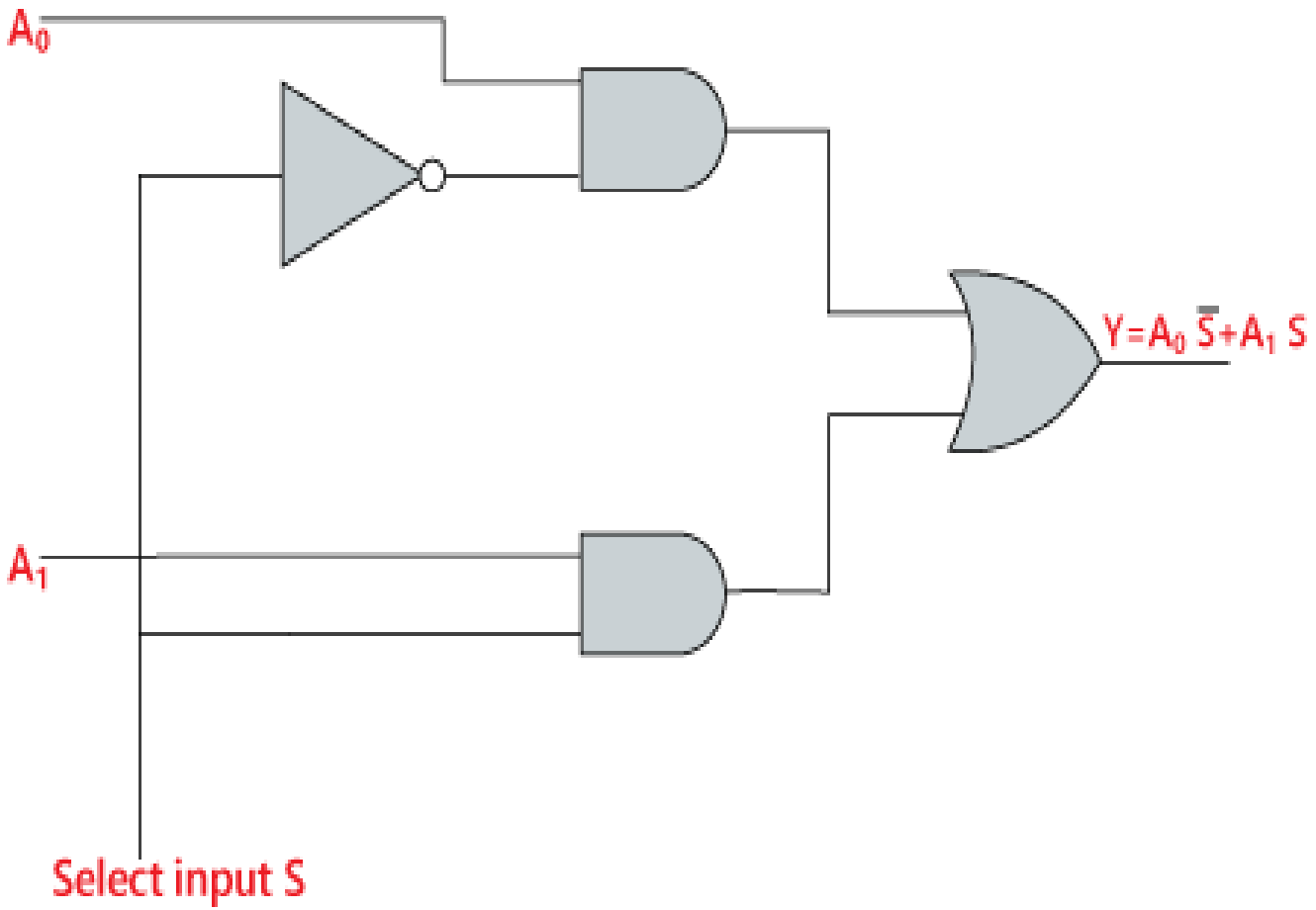
Truth Table:

INPUTS	Output
S_0	Y
0	A_0
1	A_1

The logical expression of the term Y is as follows:

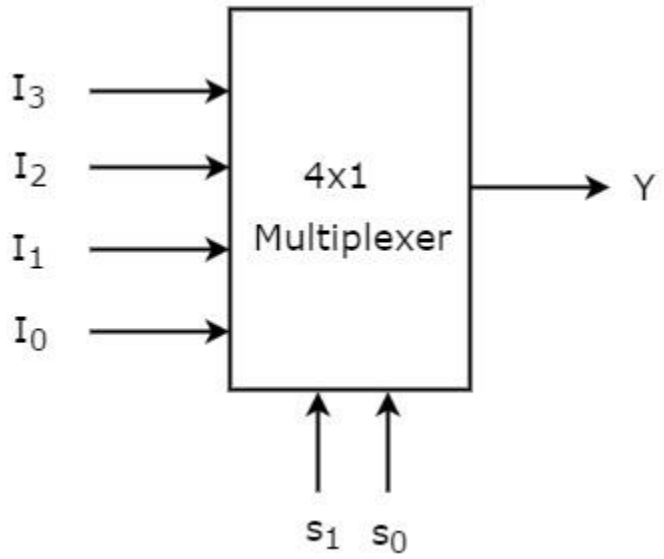
Y=S0'.A0+S0.A1

Logical circuit of the above expression is given below:



4:1/(4X1) MUX

- 4x1 Multiplexer has four data inputs I3, I2, I1 & I0, two selection lines s1 & s0 and one output Y.
- The block diagram of 4x1 Multiplexer is shown in the following figure.
- One of these 4 inputs will be connected to the output based on the combination of inputs present at these
- two selection lines.



Block Diagram

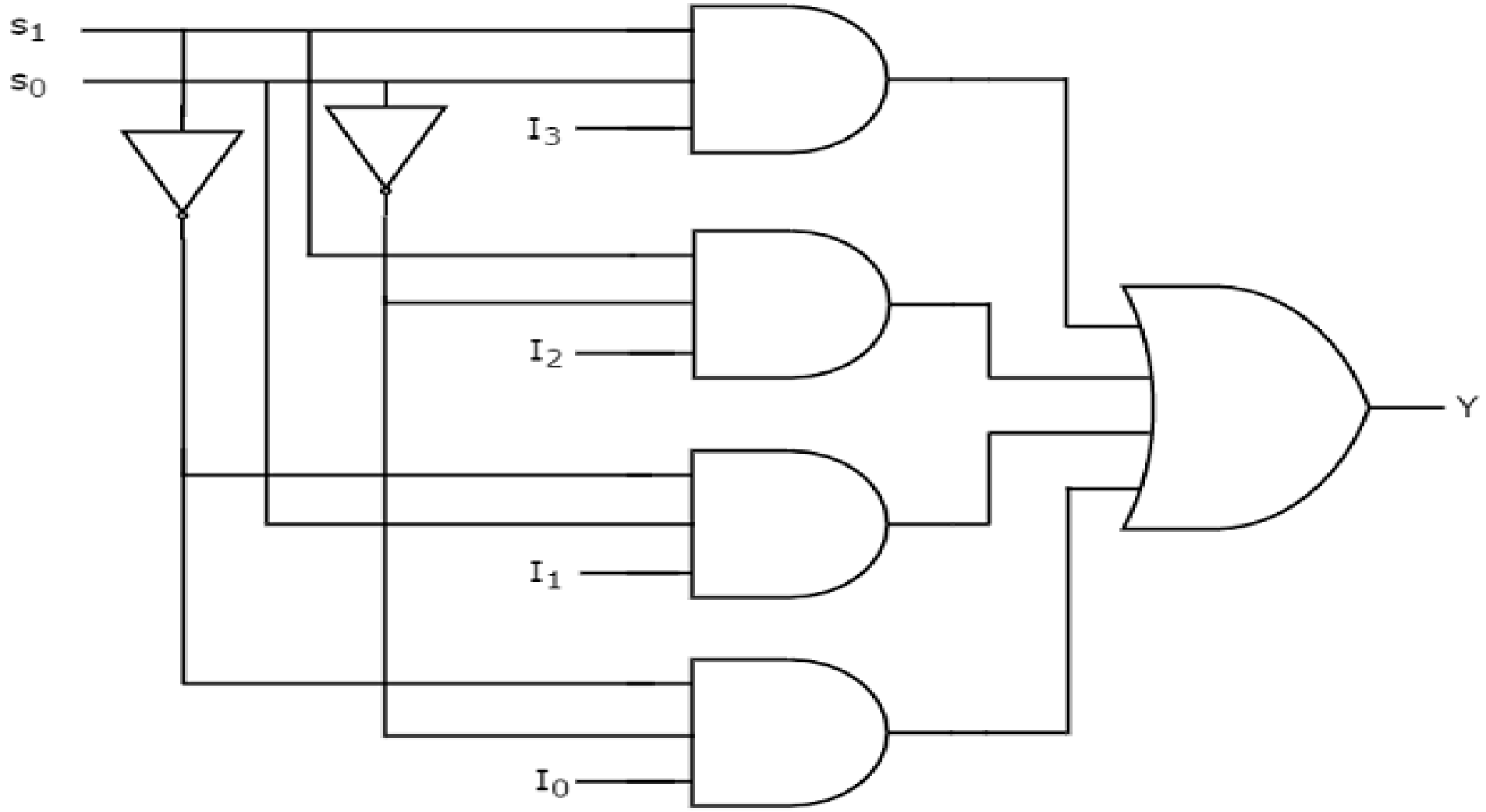
Truth table of 4x1 Multiplexer is shown below.

Selection Lines		Output
S_1	**S_0**	**Y**
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

From Truth table, we can directly write the Boolean function for output, Y as

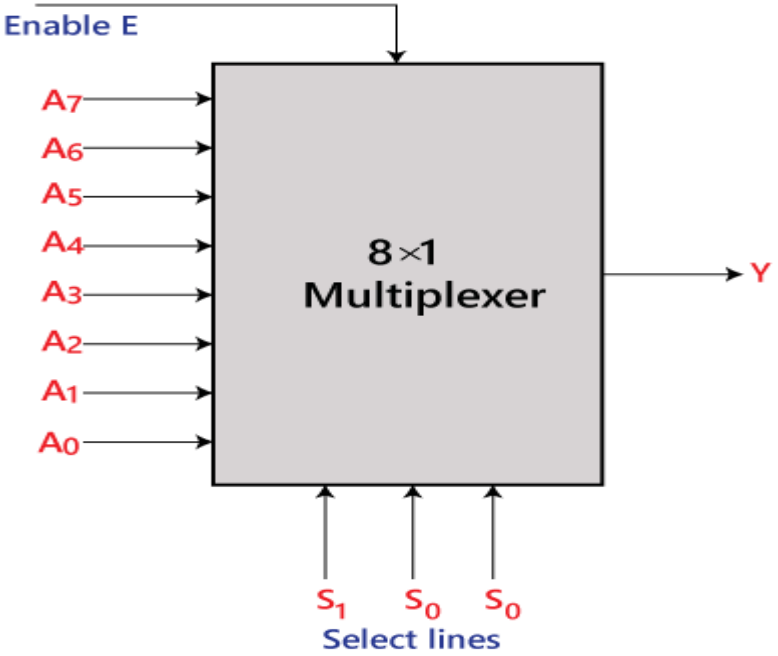
Y=S1'S0'I0+S1'S0I1+S1S0'I2+S1S0I3

The circuit diagram of 4x1 multiplexer is shown in the following figure.



8X1 / (8:1) MUX

- In the 8 to 1 multiplexer, there are total eight inputs, i.e., A_0, A_1, A_2, A_3, A_4, A_5, A_6, and A_7, 3 selection lines, i.e., S_0, $S_1$and S_2 and single output, i.e., Y.
- On the basis of the combination of inputs that are present at the selection lines S^0, $S^{1,}$ and S_2, one of these 8 inputs are connected to the output.



Block Diagram

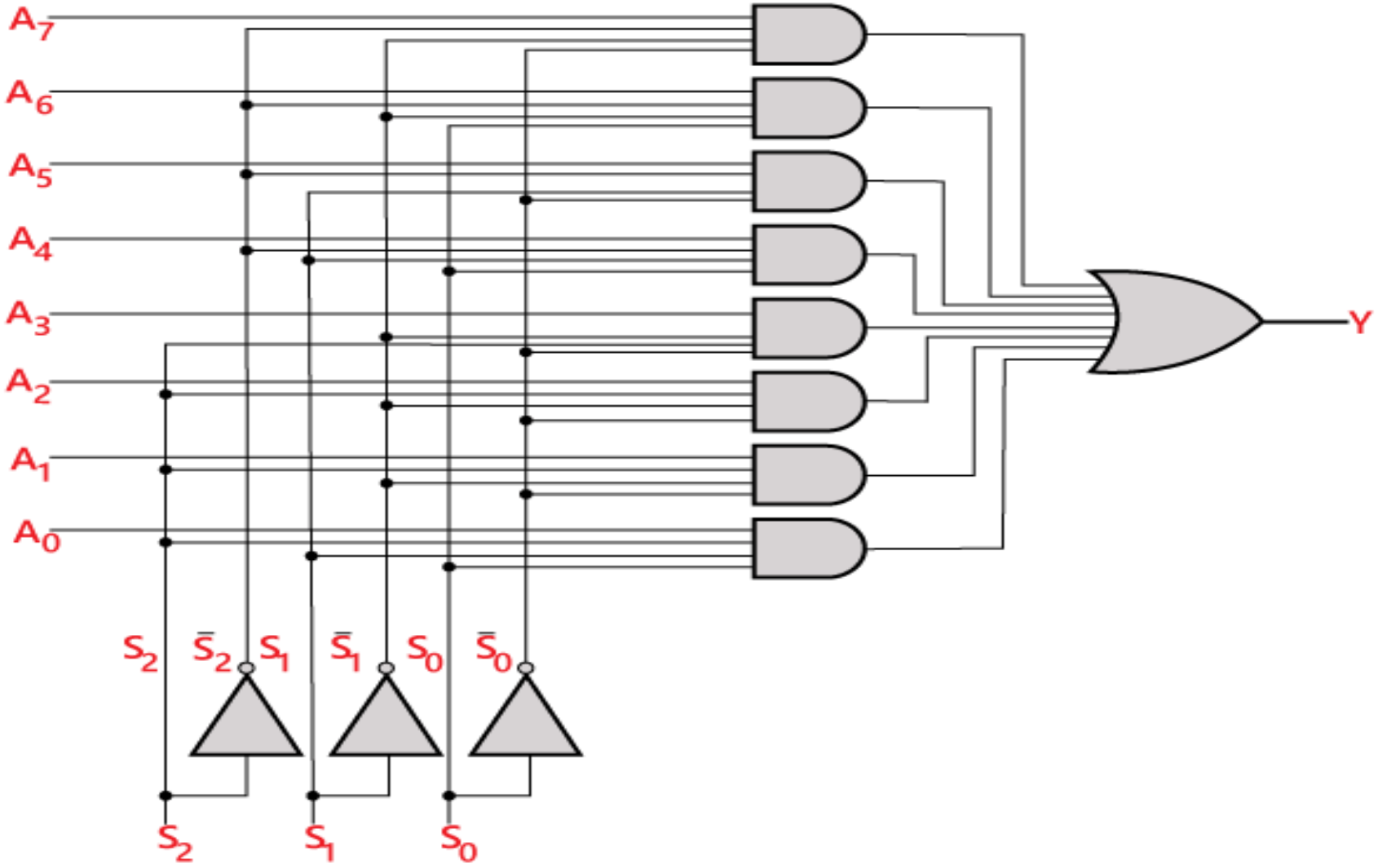
Truth table of 8x1 Multiplexer is shown below.

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

The logical expression of the term Y is as follows:

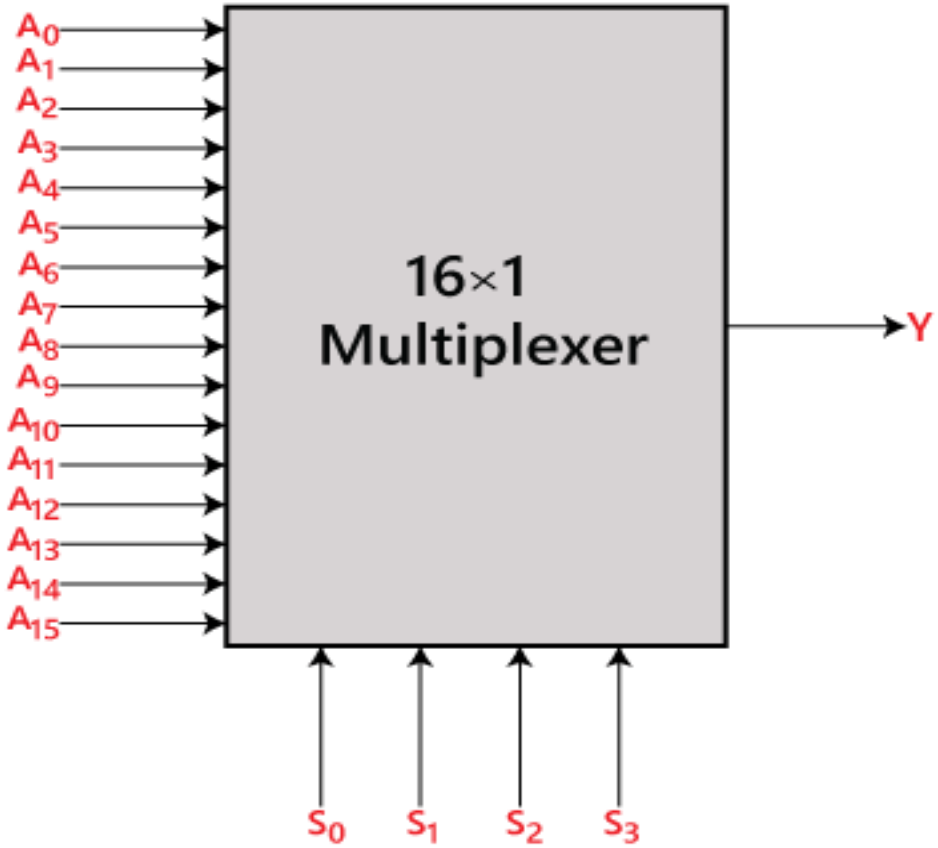
Y=S0'.S1'.S2'.A0+S0.S1'.S2'.A1+S0'.S1.S2'.A2+S0.S1.S2'.A3+S0'.S1'.S2 A4+S0.S1'.S2 A5+S0'.S1.S2 .A6+ S0.S1.S3.A7

Logical circuit of the above expression is given below:



16X1 / (16:1) MUX

- In the 16 to 1 multiplexer, there are total of 16 inputs, i.e., A0, A1, …, A16, 4 selection lines, i.e., S0, S1, S2, and S3 and single output ,i.e., Y. On the basis of the combination of inputs that are present at the selection lines S0, S1,S2 and S3, one of these 16 inputs will be connected to the output.



Block Diagram

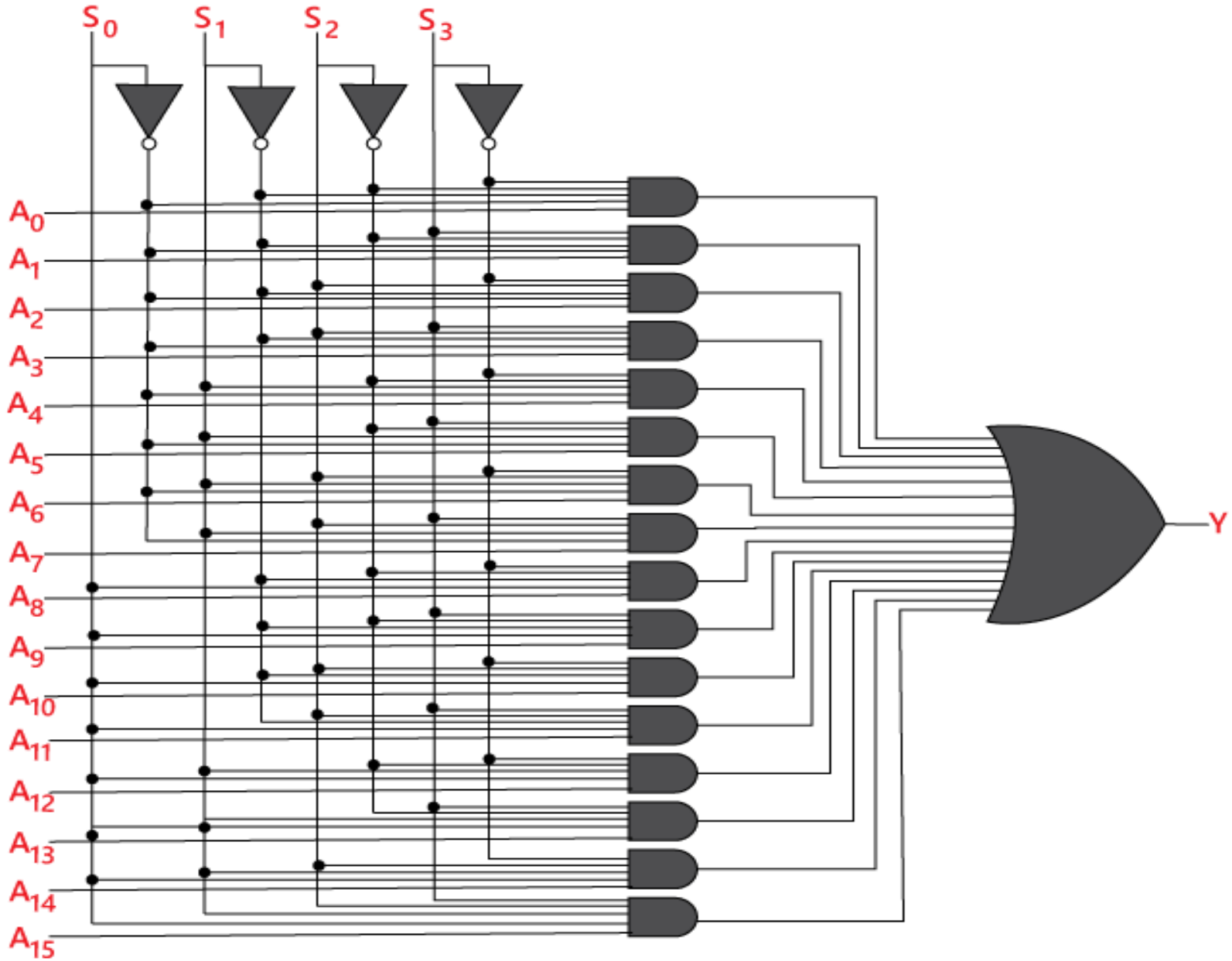
Truth Table:

INPUTS				Output
S_0	S_1	S_2	S_3	Y
0	0	0	0	A_0
0	0	0	1	A_1
0	0	1	0	A_2
0	0	1	1	A_3
0	1	0	0	A_4
0	1	0	1	A_5
0	1	1	0	A_6
0	1	1	1	A_7
1	0	0	0	A_8
1	0	0	1	A_9
1	0	1	0	A_{10}
1	0	1	1	A_{11}
1	1	0	0	A_{12}
1	1	0	1	A_{13}
1	1	1	0	A_{14}
1	1	1	1	A_{15}

The logical expression of the term Y is as follows:

$$Y=A_0.S_0'.S_1'.S_2'.S_3'+A_1.S_0'.S_1'.S_2'.S_3+A_2.S_0'.S_1'.S_2.S_3'+A_3.S_0'.S_1'.S_2.S_3+A_4.S_0'.S_1.S_2'.S_3'+A_5.S_0'.S_1.S_2'.S_3+A_6.S_1.S_2.S_3'+A_7.S_0'.S_1.S_2.S_3+A_8.S_0.S_1'.S_2'.S_3'+A_9.S_0.S_1'.S_2'.S_3+Y_10.S_0.S_1'.S_2.S_3'+A_11.S_0.S_1'.S_2.S_3+A_12\ S_0.S_1.S_2'.S_3'+A_13.S_0.S_1.S_2'.S_3+A_14\ S_0.S_1.S_2.S_3'+A_15.S_0.S_1.S_2.S_3$$

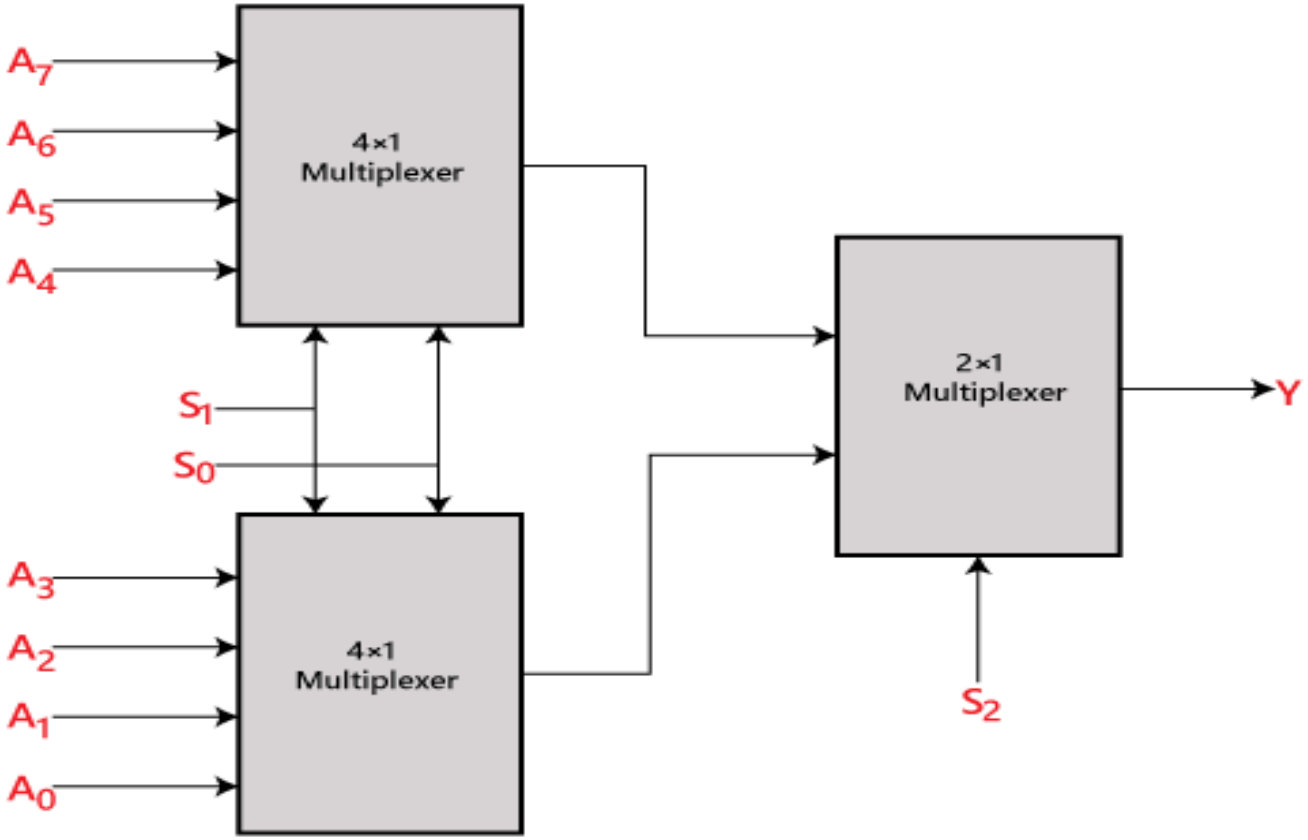
Logical circuit of the above expression is given below:



8 ×1 multiplexer using 4×1 and 2×1 multiplexer

- We can implement the 8×1 multiplexer using a lower order multiplexer.
- To implement the 8×1 multiplexer, we need two 4×1 multiplexers and one 2×1 multiplexer.
- The 4×1 multiplexer has 2 selection lines, 4 inputs, and 1 output. The 2×1 multiplexer has only 1 selection line.
- For getting 8 data inputs, we need two 4×1 multiplexers.
- The 4×1 multiplexer produces one output.
- So, in order to get the final output, we need a 2×1 multiplexer.

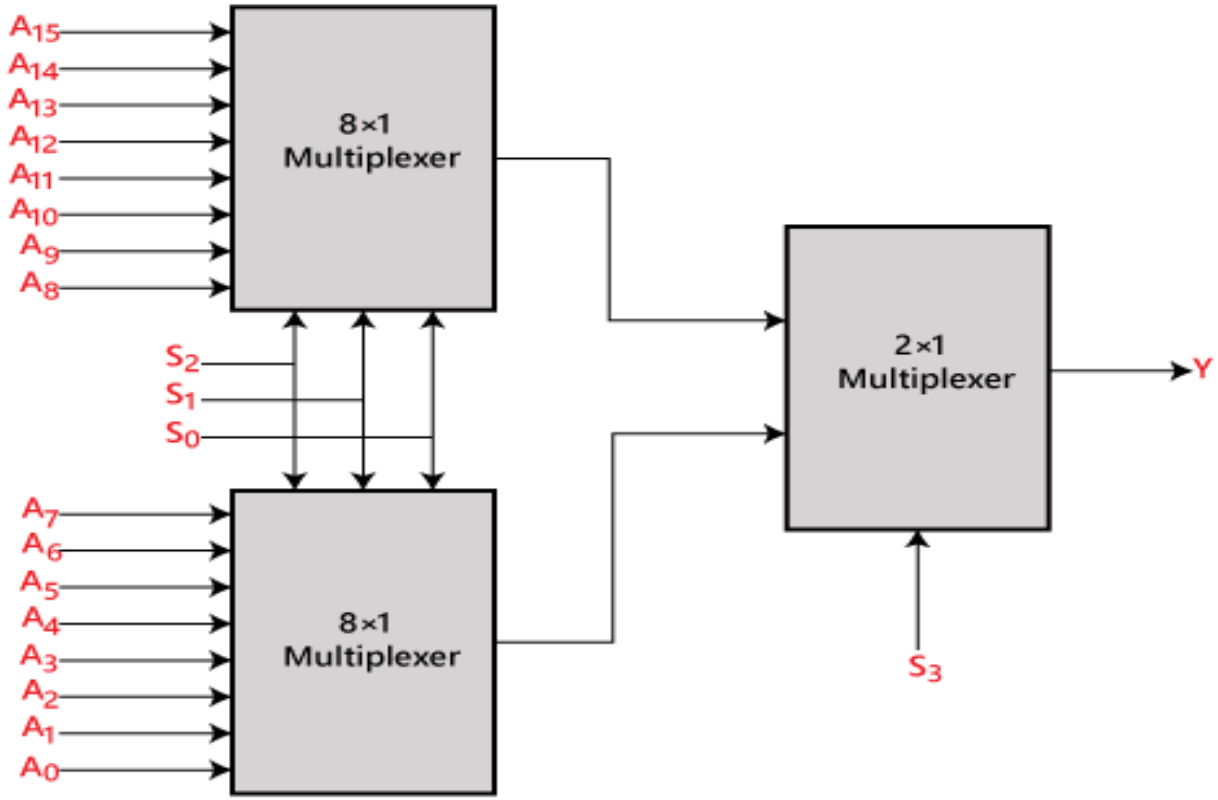
The block diagram of 8×1 multiplexer using 4×1 and 2×1 multiplexer is given below.



16×1 multiplexer using 8×1 and 2×1 multiplexer

- We can implement the 16×1 multiplexer using a lower order multiplexer.
- To implement the 8×1 multiplexer, we need two 8×1 multiplexers and one 2×1 multiplexer.
- The 8×1 multiplexer has 3 selection lines, 4 inputs, and 1 output.
- The 2×1 multiplexer has only 1 selection line.
- For getting 16 data inputs, we need two 8 ×1 multiplexers.
- The 8×1 multiplexer produces one output.
- So, in order to get the final output, we need a 2×1 multiplexer.

The block diagram of 16×1 multiplexer using 8×1 and 2×1 multiplexer is given below.



Demultiplexer

- 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.
- Select Lines is based on N output lines.
- De-multiplexer is also treated as De-mux.

Applications of Demultiplexers

The applications include:

•Demultiplexers are used in clock data recovery solutions.

•Demultiplexer along with multiplexer is necessary for any communication system for data transmission.

•Demultiplexers are used in ATM packets broadcasting.

•The output of Arithmetic Logic Unit is stored in respective registers using Demultiplexers.

•They act as Serial to Parallel converter.

•They are also used in Wavelength routers.

Advantages of Demultiplexers

The advantages include:

- Transmission of Audio/Video signals requires combination of Multiplexers and Demultiplexers.
- They are also used as decoders in security systems like banking sectors.
- Combination of De-muxes with Muxes increases the efficiency of the communication system.

Disadvantages of Demultiplexers

The disadvantages include:

- Wastage of Bandwidth might occur.
- Delays might occur due to synchronization of signals.

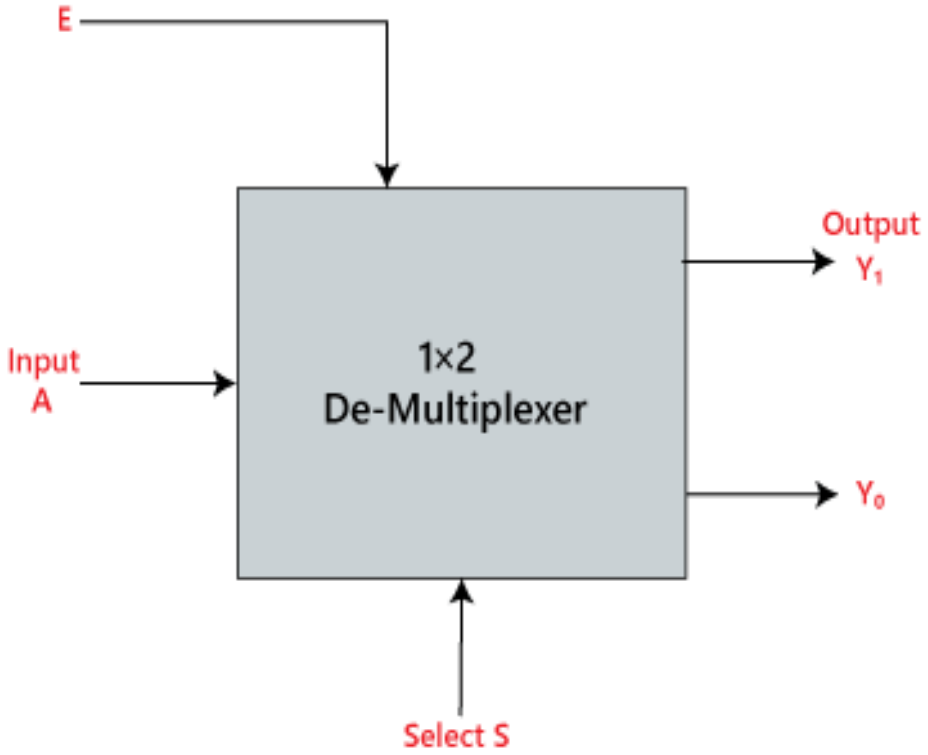
Types of Demultiplexer:

1) 1:2 De-mux
2) 1:4 De-mux
3) 1:8 De-mux
4) 1:16 De-mux

1×2 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.



Block Diagram

Truth Table:

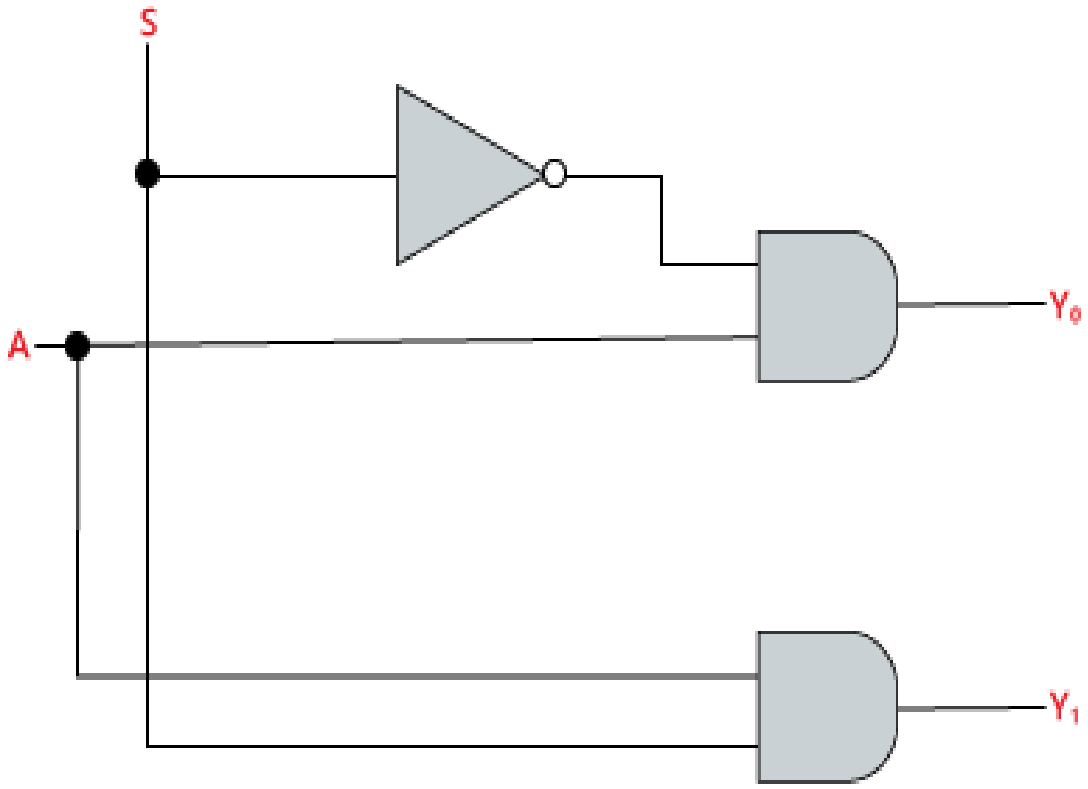
INPUTS	Output	
S_0	Y_1	Y_0
0	0	A
1	A	0

The logical expression of the term Y is as follows:

Y0=S0'.A

Y1=S0.A

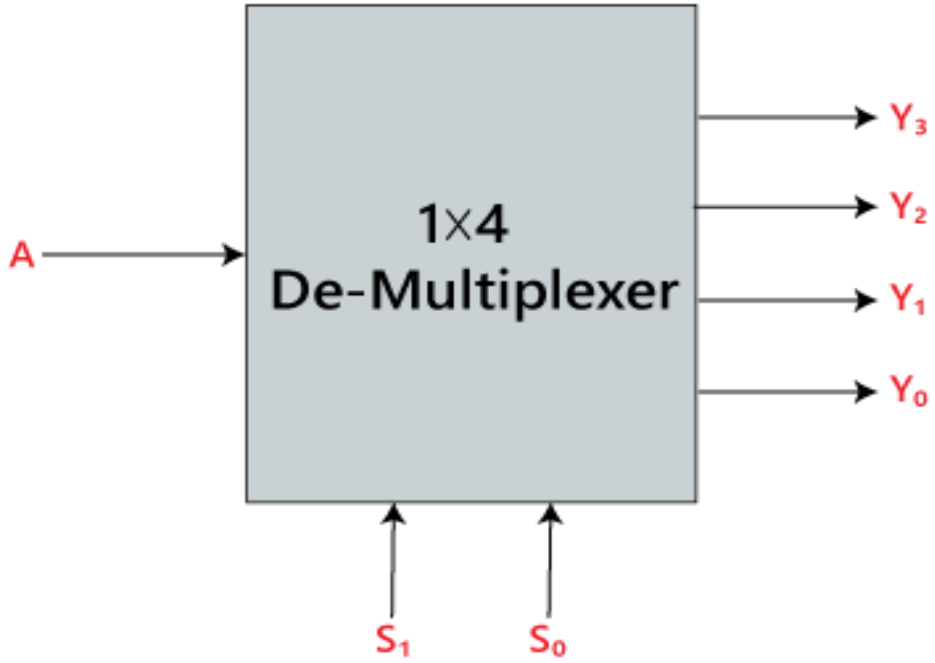
Logical circuit of the above expressions is given below:



1×4 De-multiplexer:

- In 1 to 4 De-multiplexer, there are total of four outputs, i.e., Y0, Y1, Y2, and Y3, 2 selection lines, i.e., S0 and S1 and single input, i.e., A.
- On the basis of the combination of inputs which are present at the selection lines S0 and S1, the input be connected to one of the outputs.

The block diagram and the truth table of the 1×4 multiplexer



Block Diagram

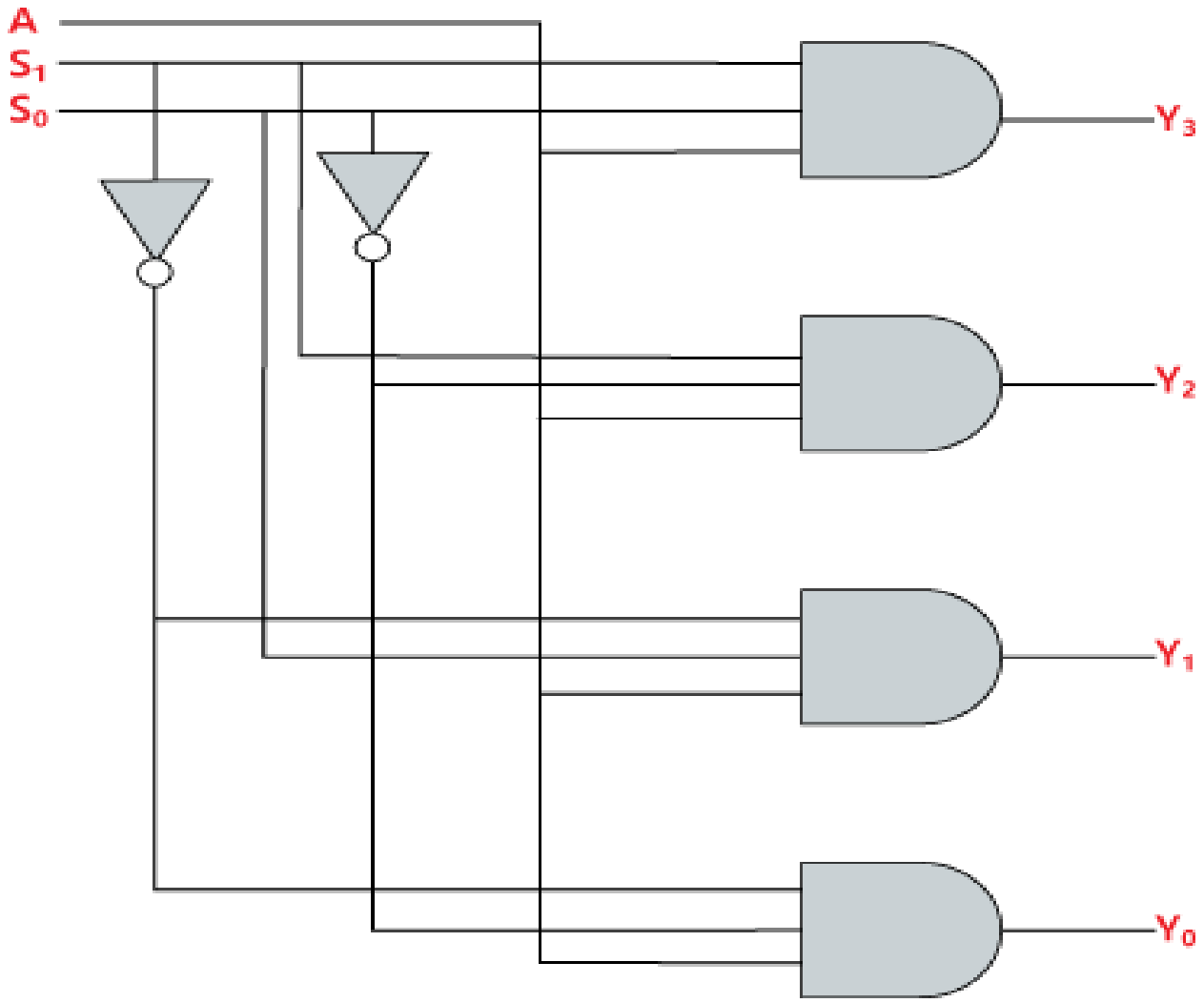
Truth Table:

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:

Y0=S1' S0' A
y1=S1' S0 A
y2=S1 S0' A
y3=S1 S0 A

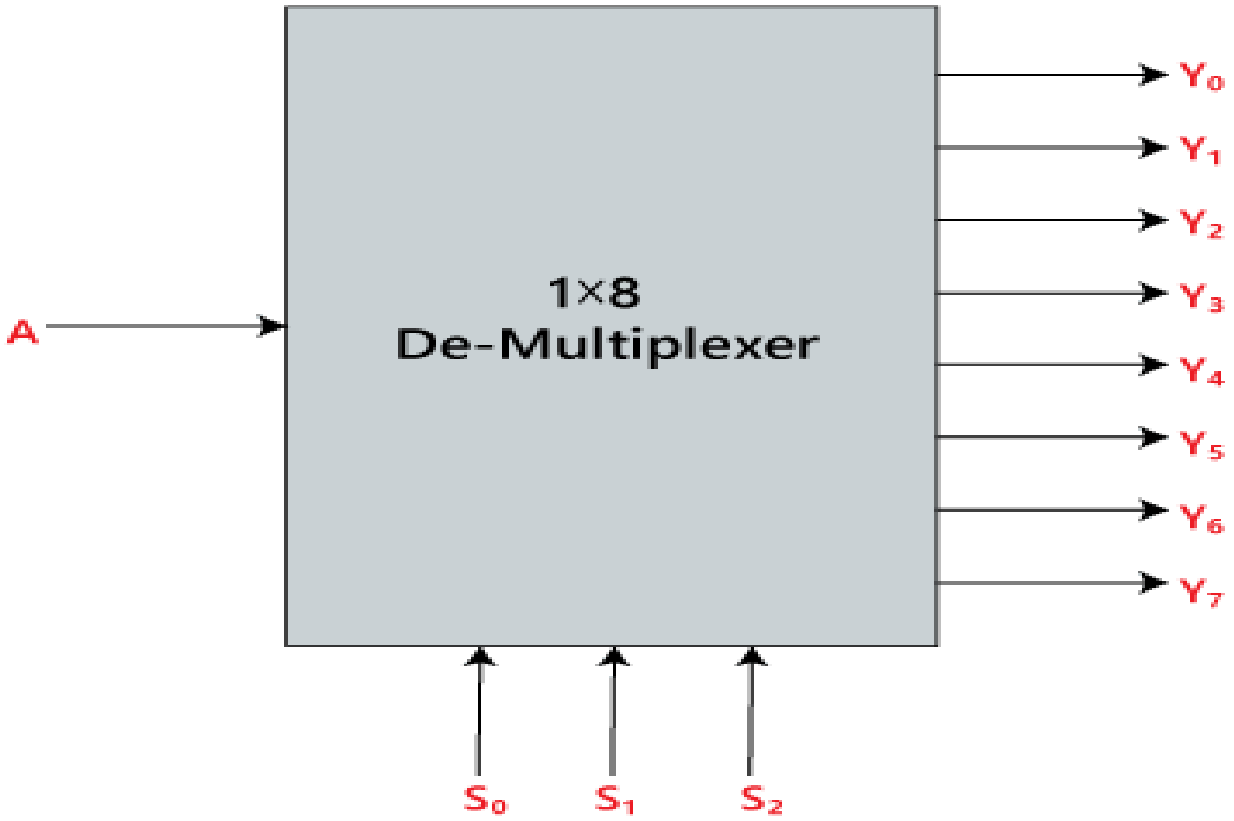
Logical circuit of the above expressions is given below:



1×8 De-multiplexer:

- In 1 to 8 De-multiplexer, there are total of eight outputs, i.e., Y0, Y1, Y2, Y3, Y4, Y5, Y6, and Y7, 3 selection lines, i.e., S0, S1and S2 and single input, i.e., A.
- On the basis of the combination of inputs which are present at the selection lines S0, S1 and S2, the input will be connected to one of these outputs.

The block diagram and the truth table of the 1×8 de-multiplexer are given below.



Block Diagram

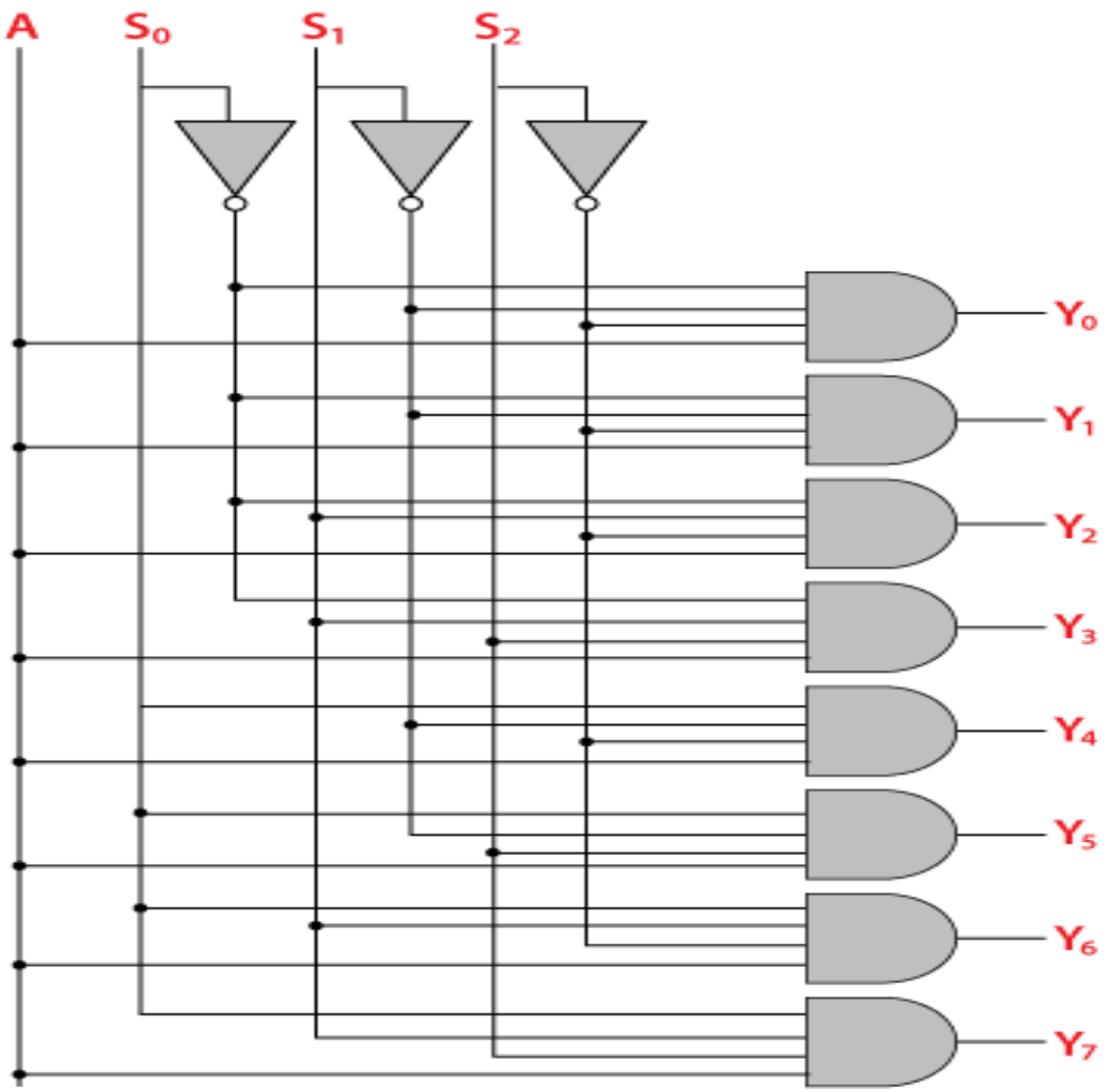
Truth Table:

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

The logical expression of the term Y is as follows:

Y0=S0'.S1'.S2'.A
Y1=S0.S1'.S2'.A
Y2=S0'.S1.S2'.A
Y3=S0.S1.S2'.A
Y4=S0'.S1'.S2 A
Y5=S0.S1'.S2 A
Y6=S0'.S1.S2 A
Y7=S0.S1.S3.A

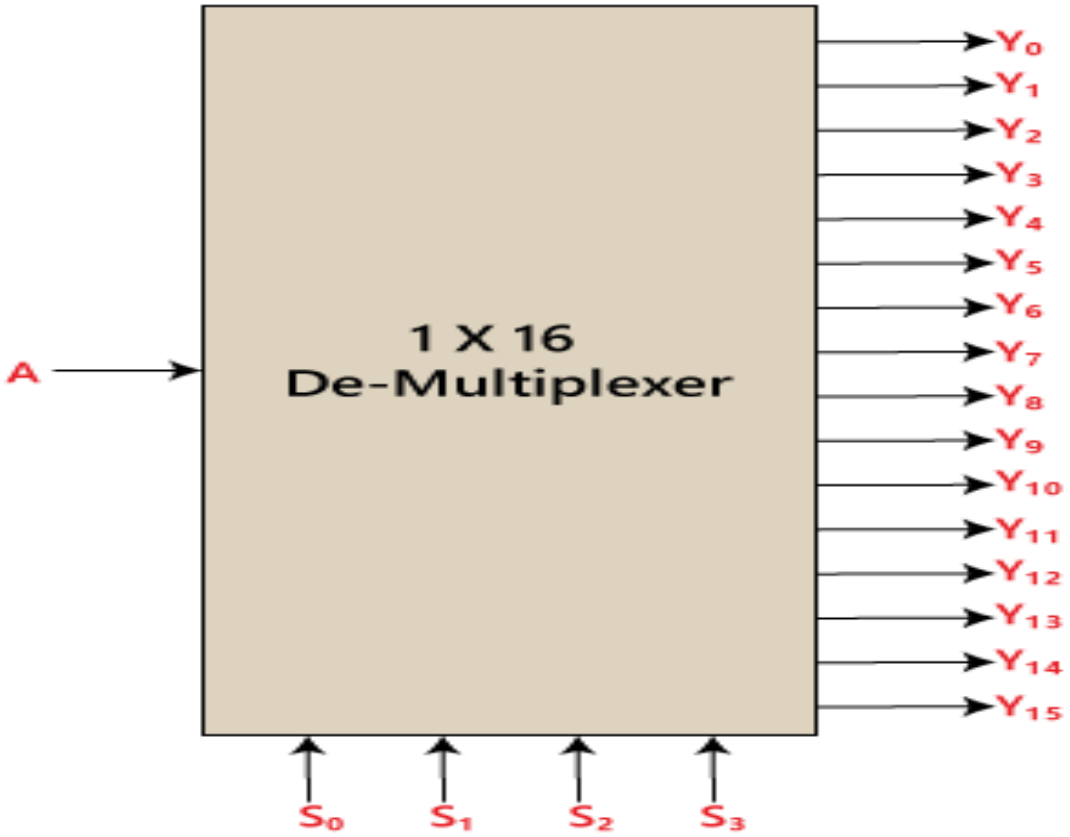
Logical circuit of the above expressions is given below:



1 x 16 De-multiplexer

- In 1×16 de-multiplexer, there are total of 16 outputs, i.e., Y_0, Y_1, …, Y_{16}, 4 selection lines, i.e., S_0, S_1, S_2, and S_3 and single input, i.e., A.
- On the basis of the combination of inputs which are present at the selection lines S^0, S^1, and S_2, the input will be connected to one of these outputs.

The block diagram and the truth table of the 1×16 de-multiplexer are given below.



Block Diagram

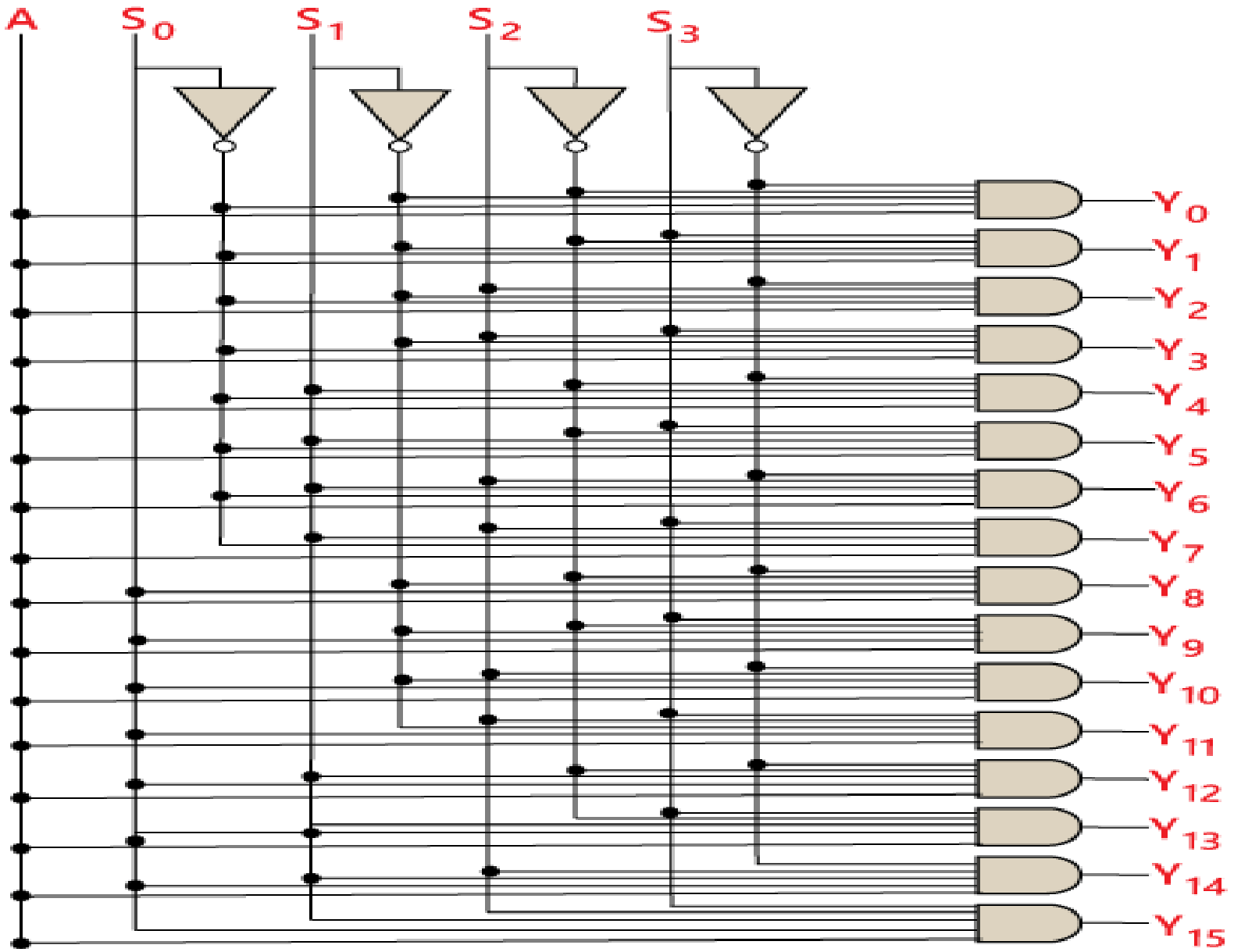
Truth Table:

INPUTS				OUTPUTS															
S_3	S_2	S_1	S_0	Y_{15}	Y_{14}	Y_{13}	Y_{12}	Y_{11}	Y_{10}	Y_9	Y_8	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	0	0	0	0	0	0	0	0	0	A
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	A	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	A	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	A	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	A	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	A	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	A	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	A	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	A	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	A	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	A	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The logical expression of the term Y is as follows:

$$Y_0=A.S_0'.S_1'.S_2'.S_3'$$
$$Y_1=A.S_0'.S_1'.S_2'.S_3$$
$$Y_2=A.S_0'.S_1'.S_2.S_3'$$
$$Y_3=A.S_0'.S_1'.S_2.S_3$$
$$Y_4=A.S_0'.S_1.S_2'.S_3'$$
$$Y_5=A.S_0'.S_1.S_2'.S_3$$
$$Y_6=A.S_0'.S_1.S_2.S_3'$$
$$Y_7=A.S_0'.S_1.S_2.S_3$$
$$Y_8=A.S_0.S_1'.S_2'.S_3'$$
$$Y_9=A.S_0.S_1'.S_2'.S_3$$
$$Y_{10}=A.S_0.S_1'.S_2.S_3'$$
$$Y_{11}=A.S_0.S_1'.S_2.S_3$$
$$Y_{12}=A.S_0.S_1.S_2'.S_3'$$
$$Y_{13}=A.S_0.S_1.S_2'.S_3$$
$$Y_{14}=A.S_0.S_1.S_2.S_3’$$
$$Y_{15}=A.S_0.S_1.S_2'.S_3$$

Logical circuit of the above expressions is given below:



CODE conversion(BCD to excess-3, 8 4 -2 -1 code to bcd, 2421 code to 8 4 -2 1 code)

BCD(8 4 2 1) to excess-3

- The Excess-3 binary code is an example of a self-complementary BCD code.
- A self-complementary binary code is a code which is always complimented in itself.
- By replacing the bit 0 to 1 and 1 to 0 of a number, we find the 1's complement of the number.
- The sum of the 1'st complement and the binary number of a decimal is equal to the binary number of decimal 9.

The process of converting BCD to Excess-3 is quite simple from other conversions.

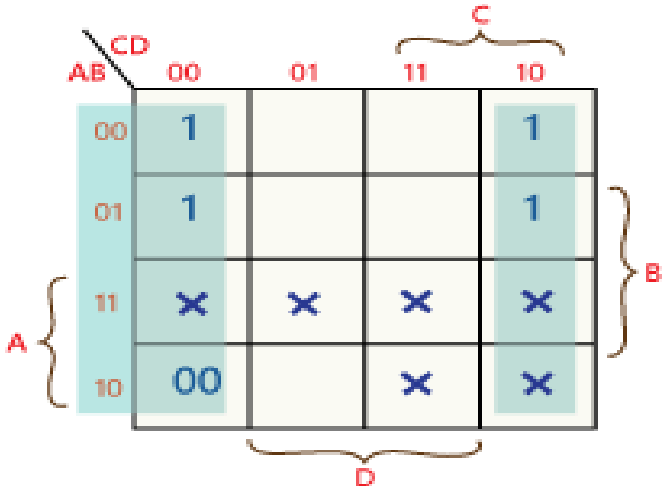
- The Excess-3 code can be calculated by adding 3, i.e., 0011 to each four-digit BCD code. Below is the truth table for the conversion of BCD to Excess-3 code.
- In the below table, the variables A, B, C, and D represent the bits of the binary numbers.
- The variable 'D' represents the LSB, and the variable 'A' represents the MSB.
- In the same way, the variables w, x, y, and z represent the bits of the Excess-3 code.
- The variable 'z' represents the LSB, and the variable 'w' represents the MSB.
- The 'don't care conditions' is expressed by the variable 'X'.

TRUTH TABLE:

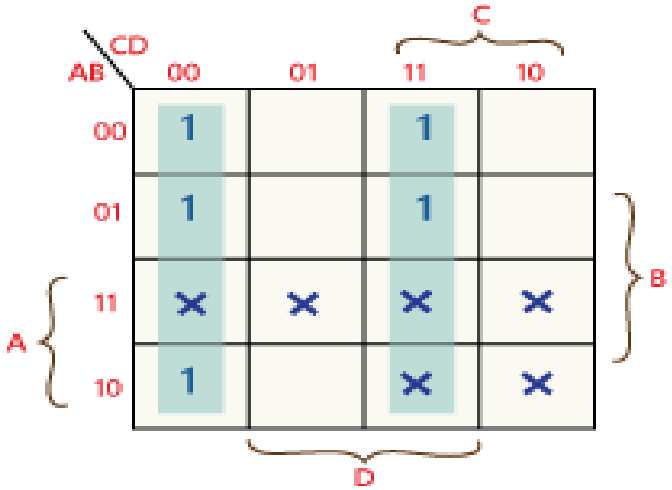
Decimal Number	BCD Code				Excess-3 Code			
	A	B	C	D	W	x	y	z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

Now, we will use the K-map method to design the logical circuit for the conversion of BCD to Excess-3 code as:

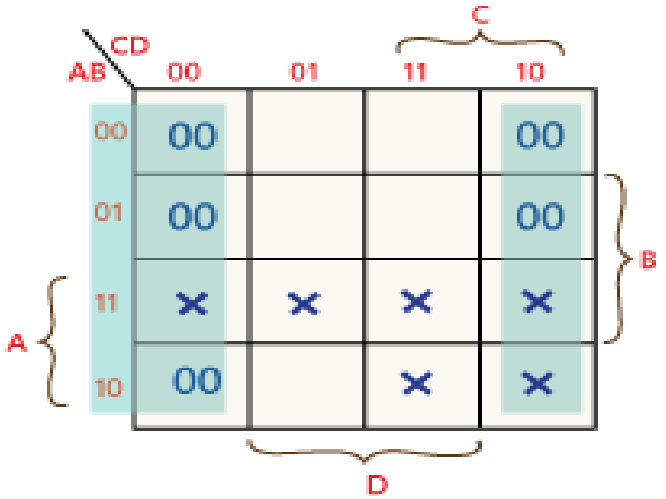
K-Map



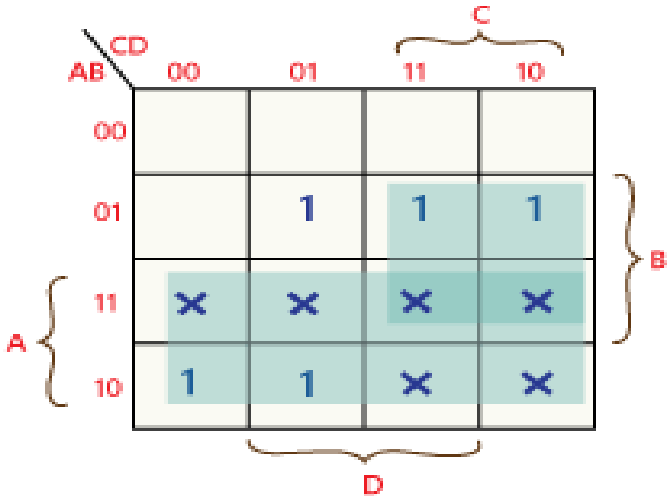
z=D′



y=CD+C′D′



x=B′C+B′D+BC′D′



w=A+BC+BD

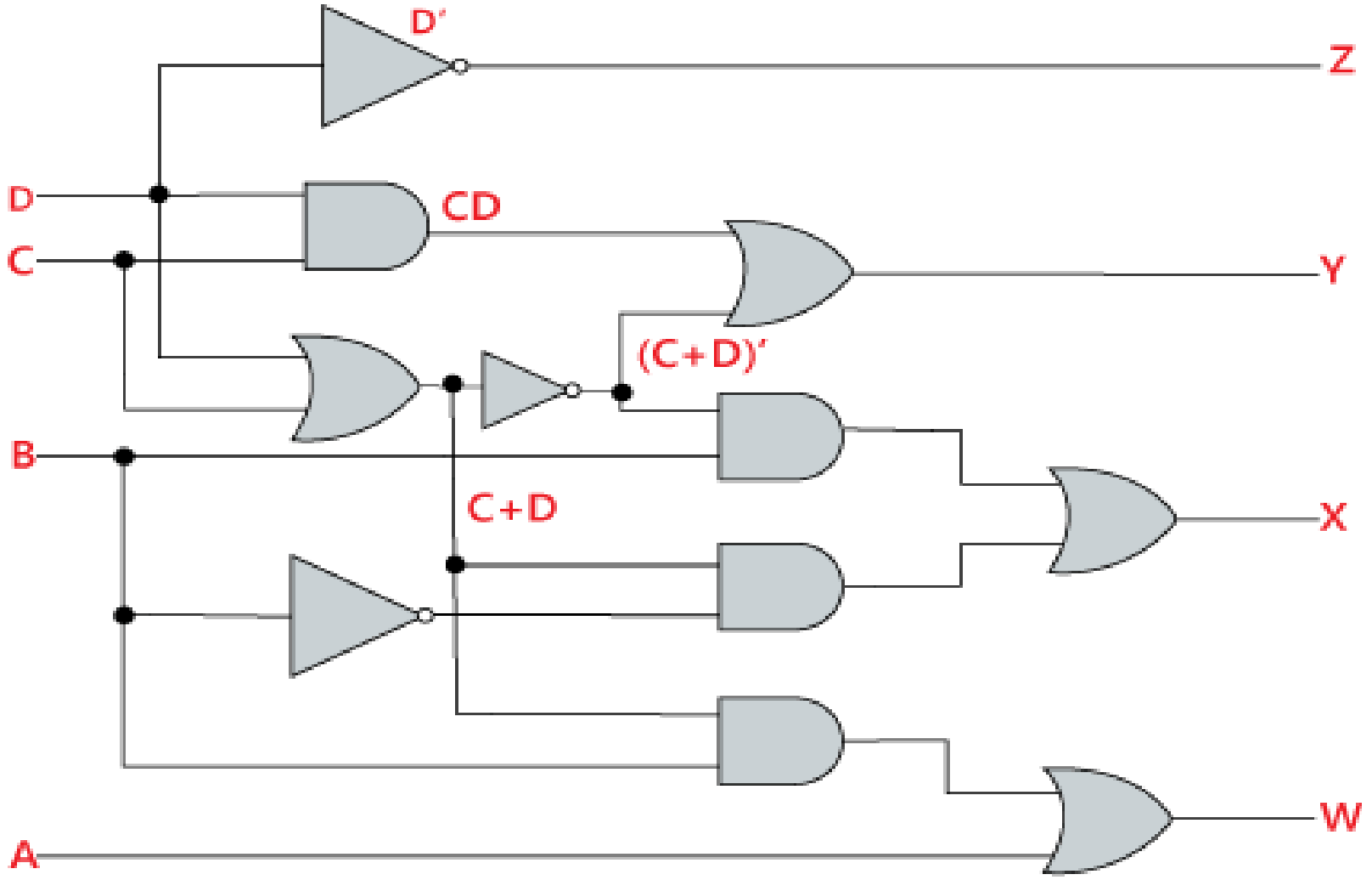
Logical Expression

w=A+BC+BD

x=B' C+B' D+BC' D'

y=CD+C'D'

z=D'



Excess-3 to BCD(8 4 2 1) conversion

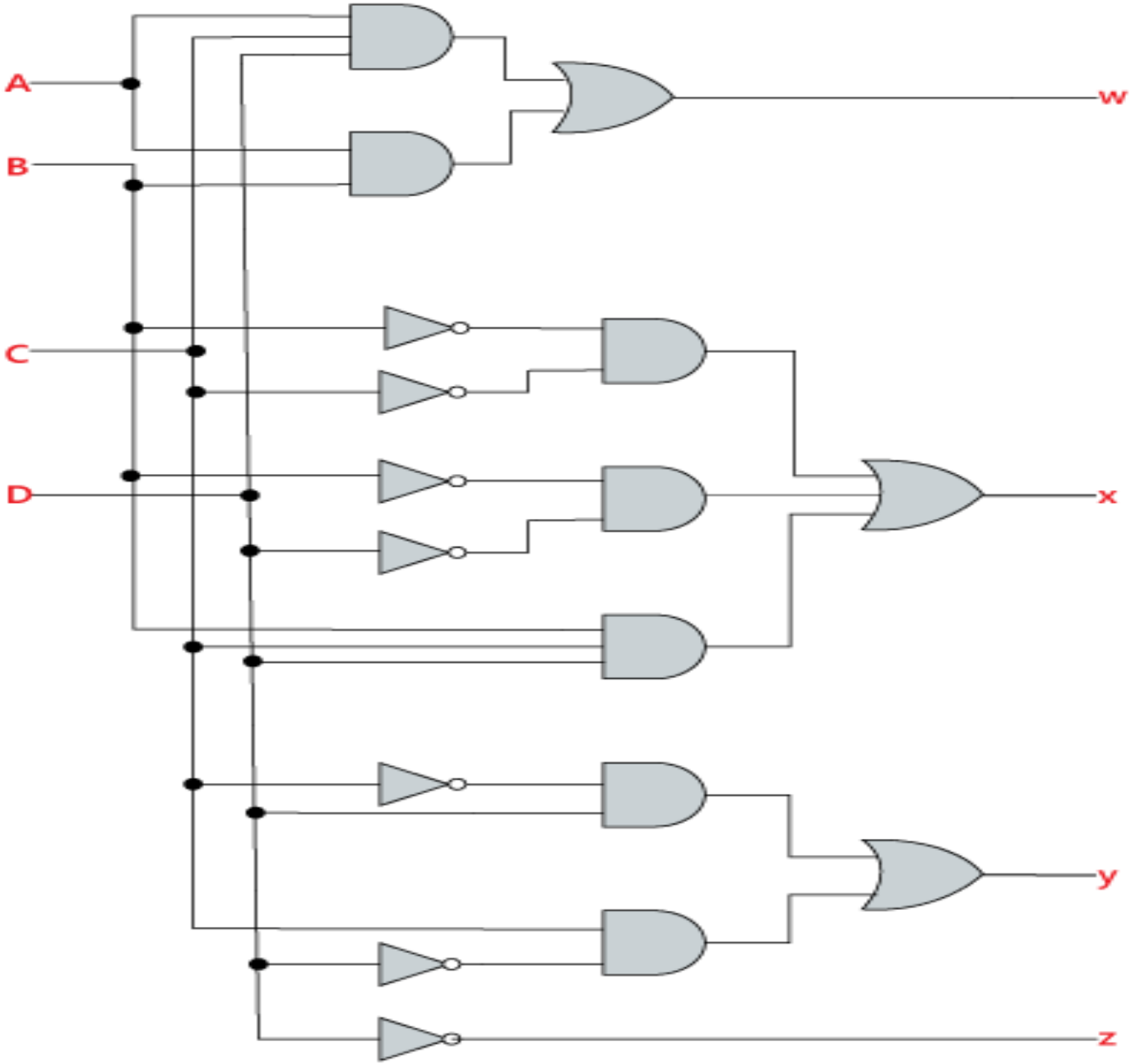
- The process of converting Excess-3 to BCD is opposite to the process of converting BCD to Excess-3.
- The BCD code can be calculated by subtracting 3, i.e., 0011 from each four-digit Excess-3 code.
- Below is the truth table for the conversion of Excess-3 code to BCD.
- The 'don't care conditions' is defined by the variable 'X'.

Decimal Number	Excess-3 Code				BCD Code			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	X	X	X	X
1	0	0	0	1	X	X	X	X
2	0	0	1	0	X	X	X	X
3	0	0	1	1	0	0	0	0
4	0	1	0	0	0	0	0	1
5	0	1	0	1	0	0	1	0
6	0	1	1	0	0	0	1	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	0	1	0	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	1	1
11	1	0	1	1	1	0	0	0
12	1	1	0	0	1	0	0	1
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

Now, we will use the K-map method to design the logical circuit for the conversion of Excess-3 code to

After K-maps BCD as:

W=AB+ACD
X=A’ B’+A’ D’+ACD
Y=C’ D+CD’
Z=D'



Binary to Gray Conversion:

- It is a logical circuit that is used to convert the binary code into its equivalent Gray code.

Decimal Number	4-bit Binary Code	4-bit Gray Code
	ABCD	**$G_1G_2G_3G_4$**
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

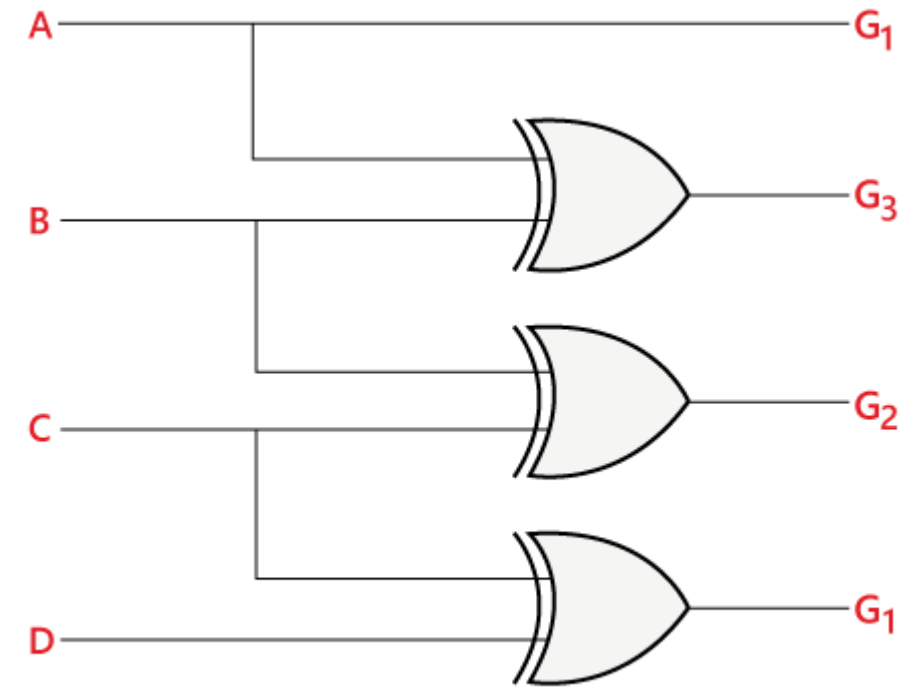
b3,b2 \ b1,b0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

b3,b2 \ b1,b0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

b3,b2 \ b1,b0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

b3,b2 \ b1,b0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

g3=b3=1
g2 = b3 XOR b2
g1= b2 XOR b1
g0= b1 XOR b0

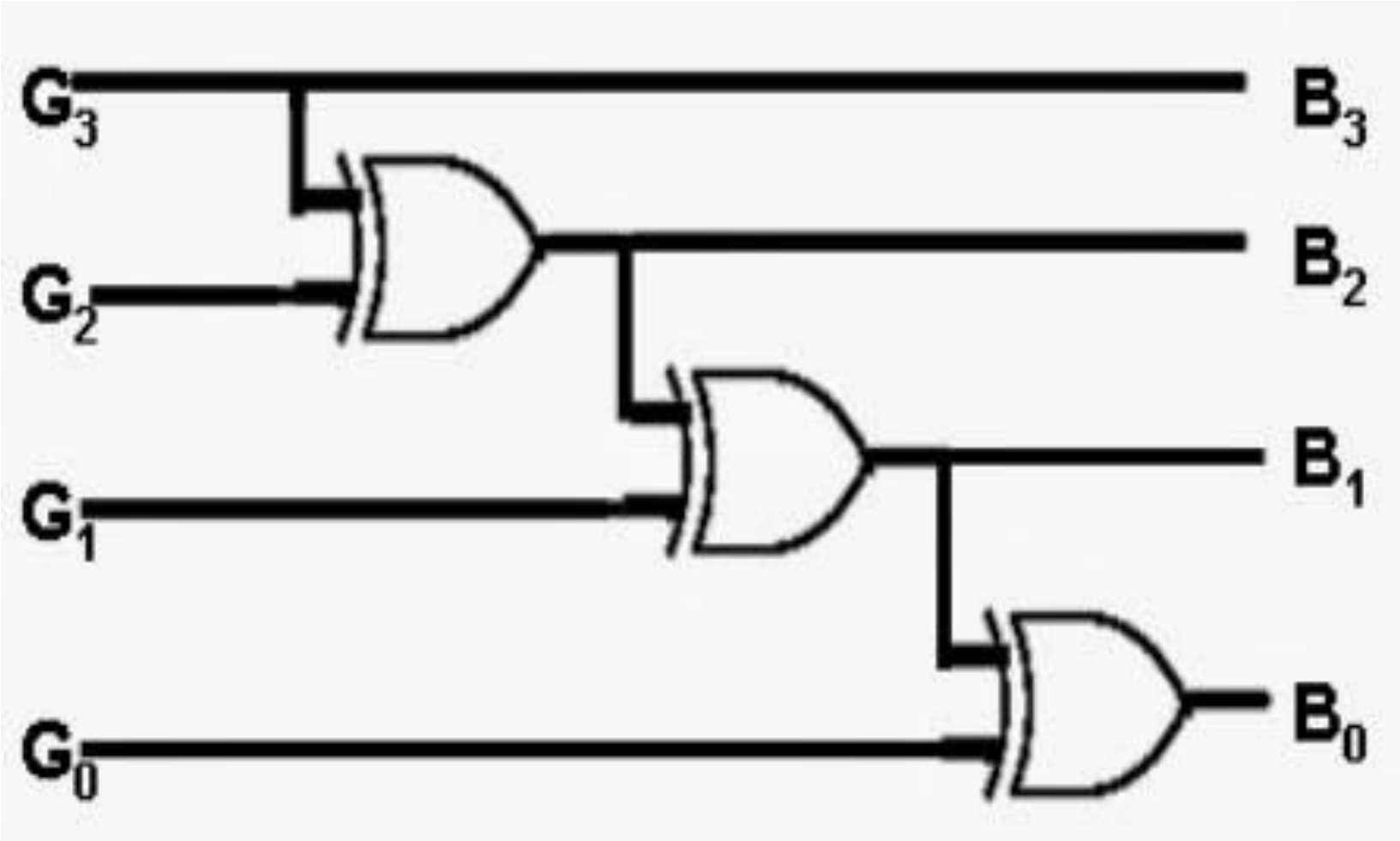


Logic Circuit for Binary to Gray Code Converter

Gray To Binary Conversion:

Gray Code Input				Binary Code Output			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

Table1


G3
G2
G1
G0
B3
B2
B1
B0

8 4 -2 -1 code to BCD Converter

Decimal Digit	84-2-1 Code	8421 Code
	A B C D	P Q R S
0	0 0 0 0	0 0 0 0
1	0 1 1 1	0 0 0 1
2	0 1 1 0	0 0 1 0
3	0 1 0 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	1 0 1 1	0 1 0 1
6	1 0 1 0	0 1 1 0
7	1 0 0 1	0 1 1 1
8	1 0 0 0	1 0 0 0
9	1 1 1 1	1 0 0 1

BCD(8 4 2 1) to 8 4 -2 -1 code Converter

Decimal Digit	8421(BCD) Code	84-2-1 Code
	P Q R S	A B C D
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 1 1 1
2	0 0 1 0	0 1 1 0
3	0 0 1 1	0 1 0 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 0 1 0
7	0 1 1 1	1 0 0 1
8	1 0 0 0	1 0 0 0
9	1 0 0 1	1 1 1 1

2 4 2 1 to 8 4 2 1(BCD) code Converter

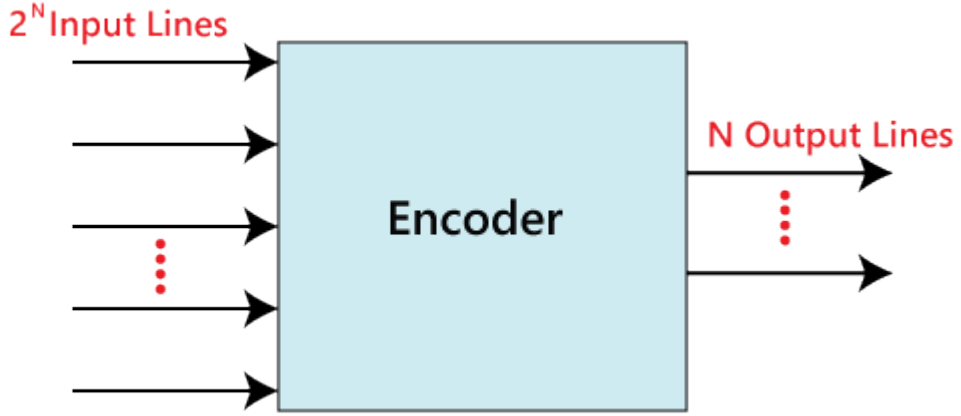
Decimal Digit	2421 Code	8421(BCD) Code
	A B C D	P Q R S
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	1 0 1 1	0 1 0 1
6	1 1 0 0	0 1 1 0
7	1 1 0 1	0 1 1 1
8	1 1 1 0	1 0 0 0
9	1 1 1 1	1 0 0 1

8 4 2 1(BCD) to 2 4 2 1 code Converter

Decimal Digit	8421(BCD) Code	2421 Code
	P Q R S	A B C D
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 1 0 0
7	0 1 1 1	1 1 0 1
8	1 0 0 0	1 1 1 0
9	1 0 0 1	1 1 1 1

Encoder

- An encoder is a combinational circuit that converts binary information in the form of a 2^N input lines into N output lines, which represent N bit code for the input.
- The combinational circuits that change the binary information into N output lines are known as **Encoders**.
- The binary information is passed in the form of 2^N input lines. The output lines define the N-bit code for the binary information.
- At a time, only one input line is activated for simplicity. The produced N-bit output code is equivalent to the binary information.



Applications of the Encoder

- Speed synchronization of multiple motors in industries
- War field flying robot with a night vision flying camera
- Robotic vehicle with the metal detector
- RF based home automation system
- Automatic health monitoring systems
- Encoders are used to convert a decimal number into the binary number.

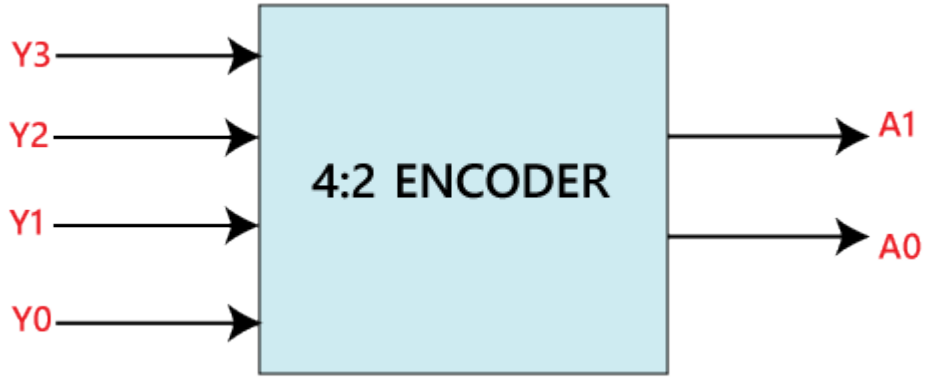
Types of Encoder

4 to 2 line Encoder:

- In 4 to 2 line encoder, there are total of four inputs, i.e., Y0, Y1, Y2, and Y3, and two outputs, i.e., A0 and A1.
- In 4-input lines, one input-line is set to true at a time to get the respective binary code in the output side.

Below are the block diagram and the truth table of the 4 to 2 line encoder.

Block Diagram:

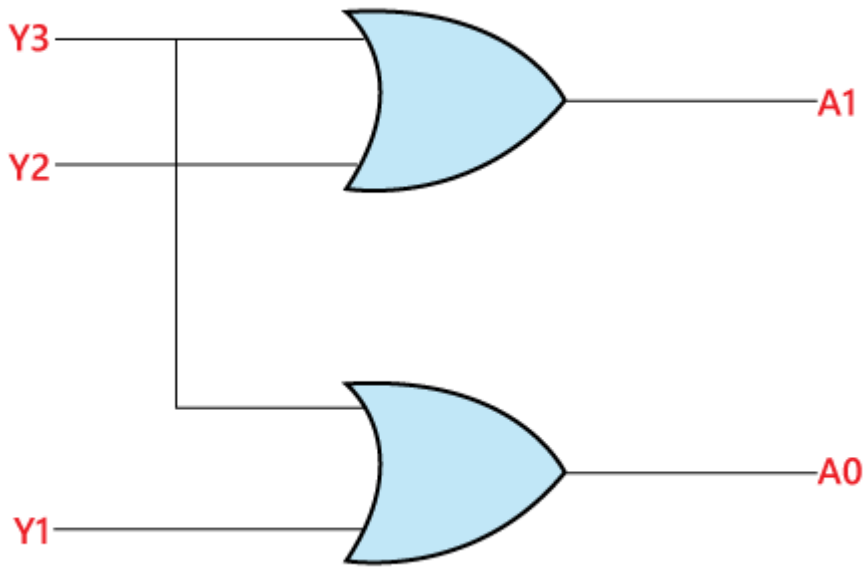


Truth Table:

INPUTS				OUTPUTS	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

The logical expression of the term A0 and A1 is as follows:

A1=Y3+Y2
A0=Y3+Y1
Logical circuit/Combinational Circuit of the above expressions is given below:

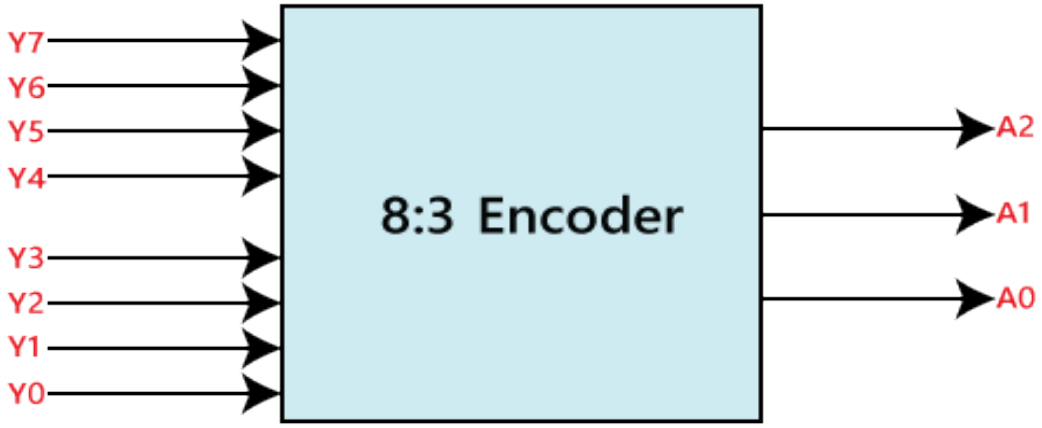


8 to 3 line Encoder/Octal to Binary

- The 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., Y0, Y1, Y2, Y3, Y4, Y5, Y6, and Y7 and three outputs, i.e., A0, A1, and A2.
- In 8-input lines, one input-line is set to true at a time to get the respective binary code in the output side.

Below are the block diagram and the truth table of the 8 to 3 line encoder.

Block Diagram:



Truth Table:

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

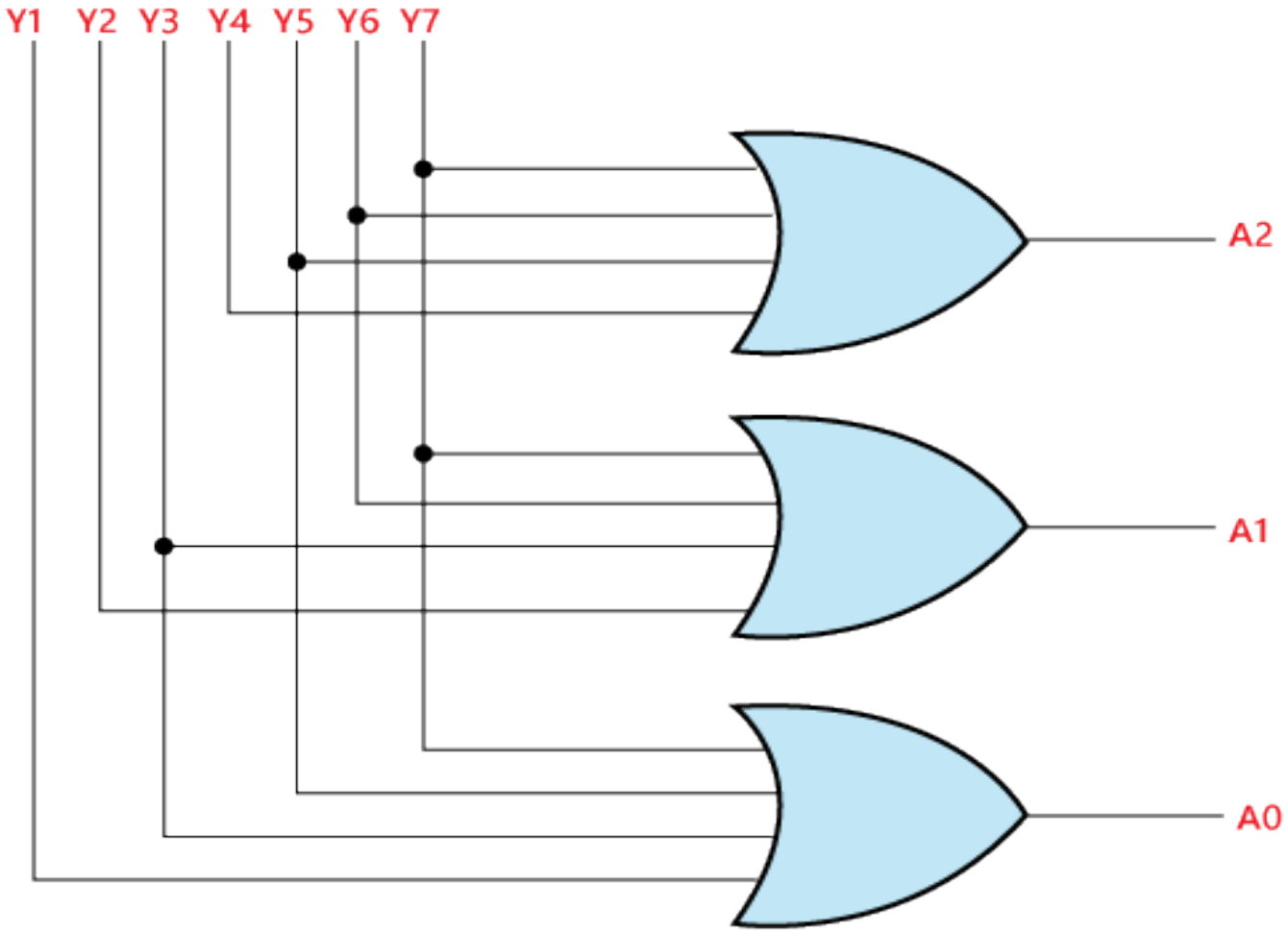
The logical expression of the term A0, A1, and A2 are as follows:

A2=Y4+Y5+Y6+Y7

A1=Y2+Y3+Y6+Y7

A0=Y7+Y5+Y3+Y1

Logical circuit of the above expressions is given below:

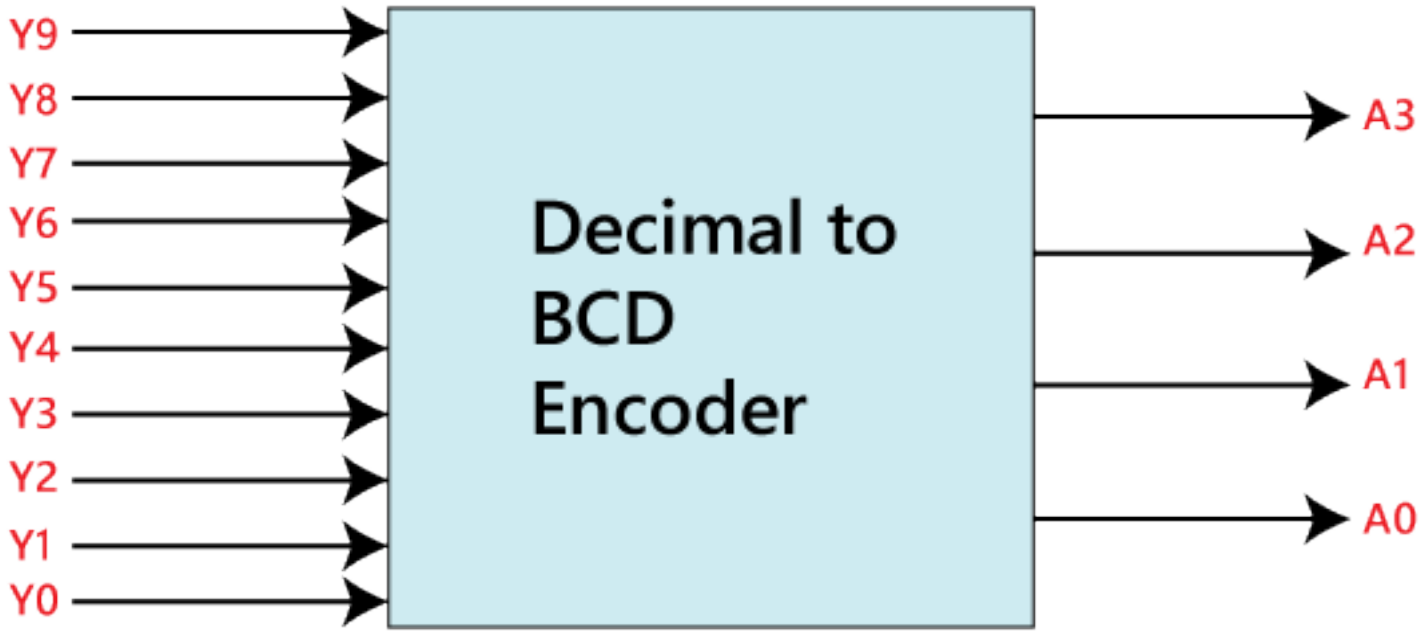


Decimal to BCD Line Encoder

- The Octal to Binary Encoder is also known as 10 to 4 line Encoder.
- In 10 to 4 line encoder, there are total of ten inputs, i.e., Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, and Y9 and four outputs., A0, A1, A2, and A3. In 10-input lines, one input-line is set to true at a time to get the respective BCD code in the
  output side.

The block diagram and the truth table of the decimal to BCD encoder are given below.

Block Diagram:



Truth Table:

INPUTS										OUTPUTS			
Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_3	A_2	A_1	A_0
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

The logical expression of the term A0, A1, A2, and A3 is as follows:

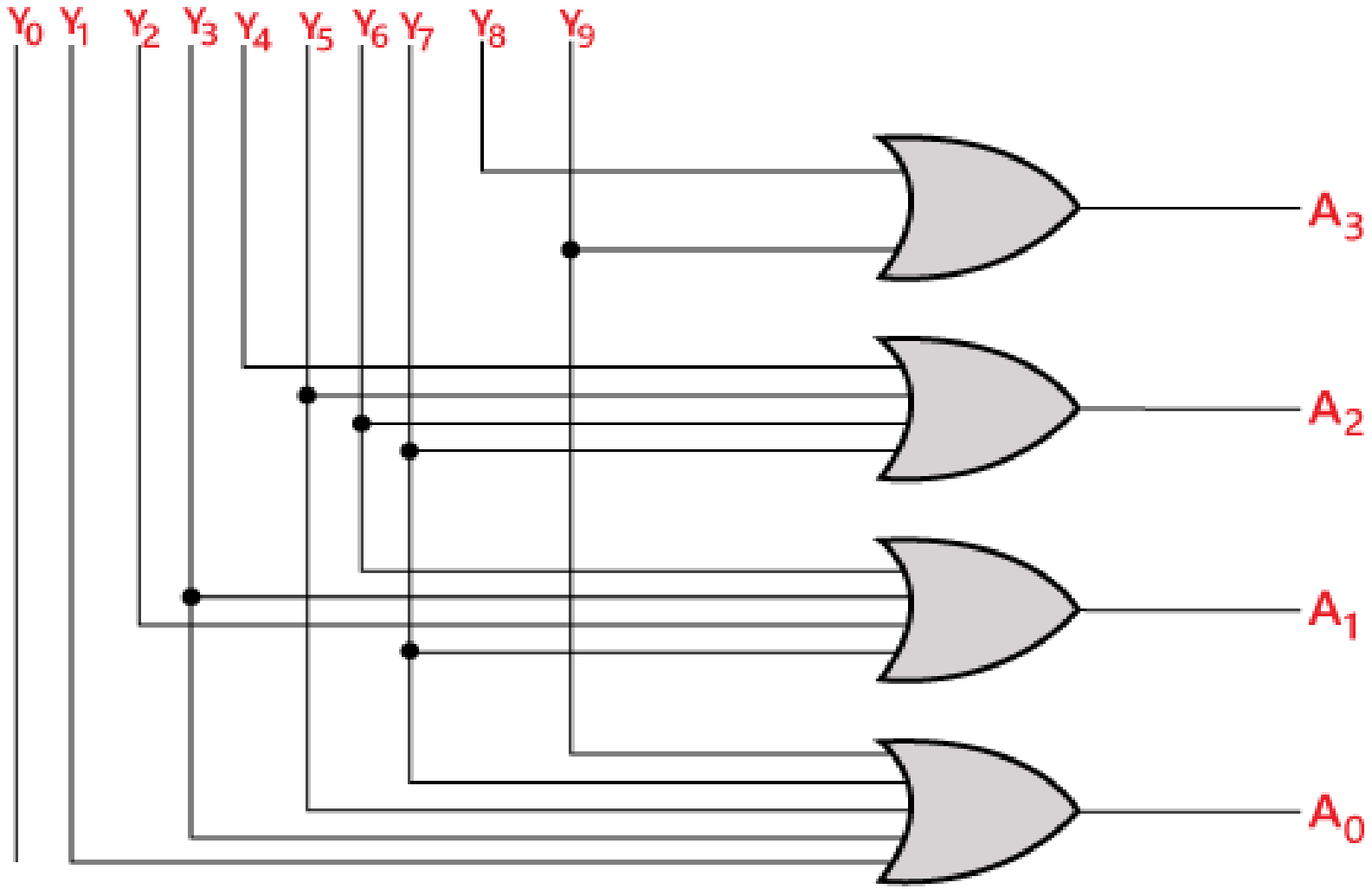
A3=Y9+Y8

A2=Y7+Y6+Y5+Y4

A1=Y7+Y6+Y3+Y2

A0 = Y9 + Y7 +Y5 +Y3 + Y1

Logical circuit of the above expressions is given below:



Priority Encoder

- A 4 to 2 priority encoder has four inputs Y3, Y2, Y1 & Y0 and two outputs A1 & A0.
- Here, the input, Y3 has the highest priority, whereas the input, Y0 has the lowest priority.
- In this case, even if more than one input is ‘1’ at the same time, the output will be the binary code corresponding to the input, which is having higher priority.
- We considered one more output, V in order to know, whether the code available at outputs is valid or not.

•If at least one input of the encoder is ‘1’, 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 ‘0’, then the code available at outputs is not a valid one. In this case, the output, V will be equal to 0.

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

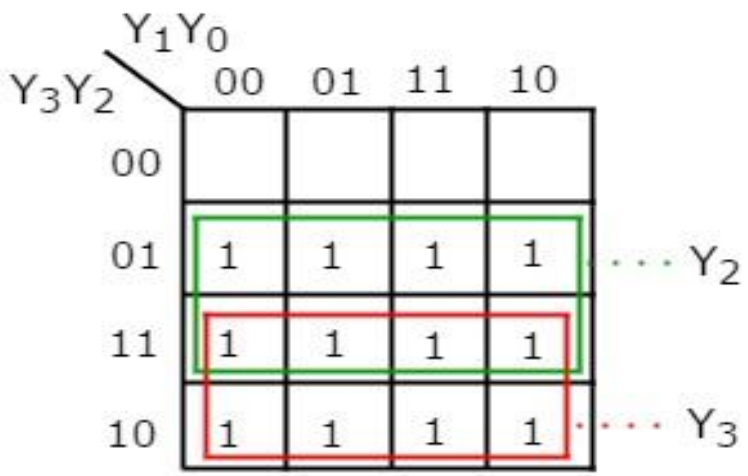
.

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

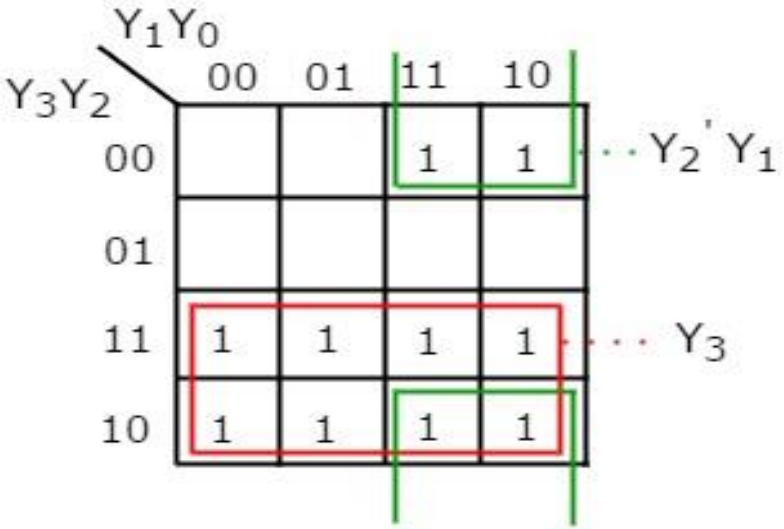
Use 4 variable K-maps for getting simplified expressions for each output

Use 4 variable K-maps for getting simplified expressions for each output.

K-Map for A_1



K-Map for A_0



The simplified Boolean functions are

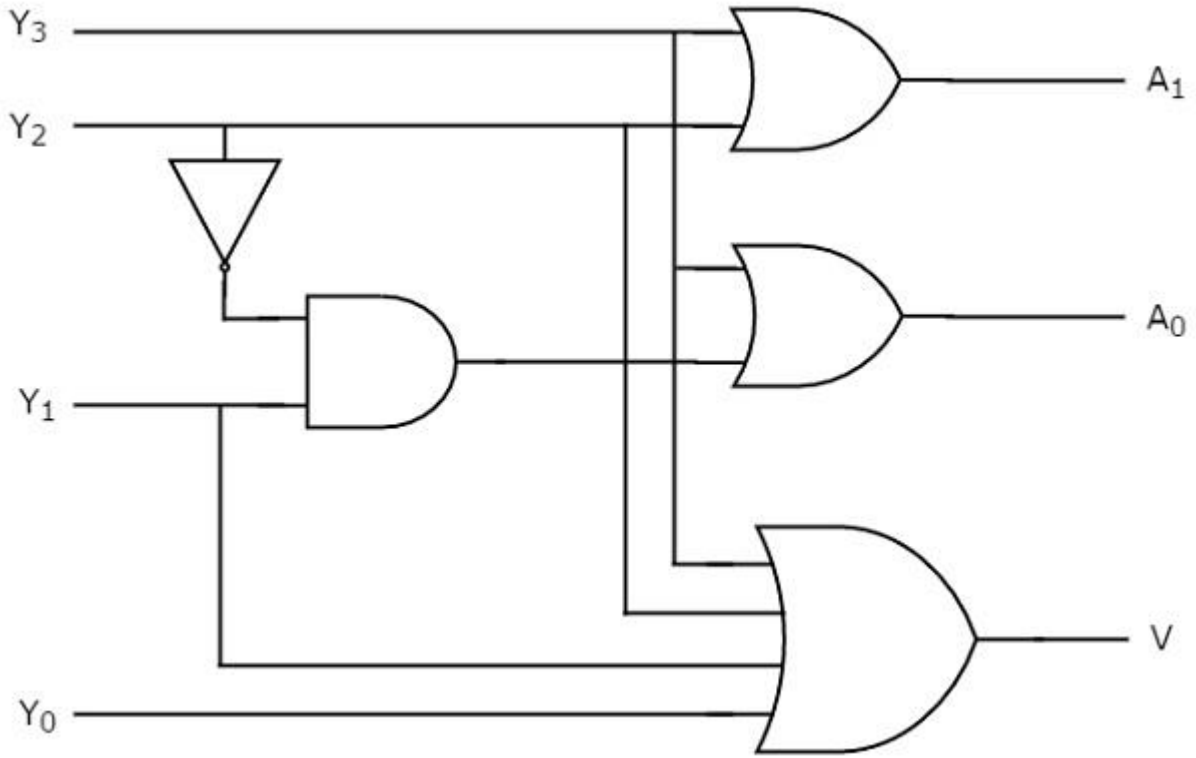
A1=Y3+Y2A1=Y3+Y2

A0=Y3+Y2'Y1A0=Y3+Y2'Y1

Similarly, we will get the Boolean function of output, V as

V=Y3+Y2+Y1+Y0

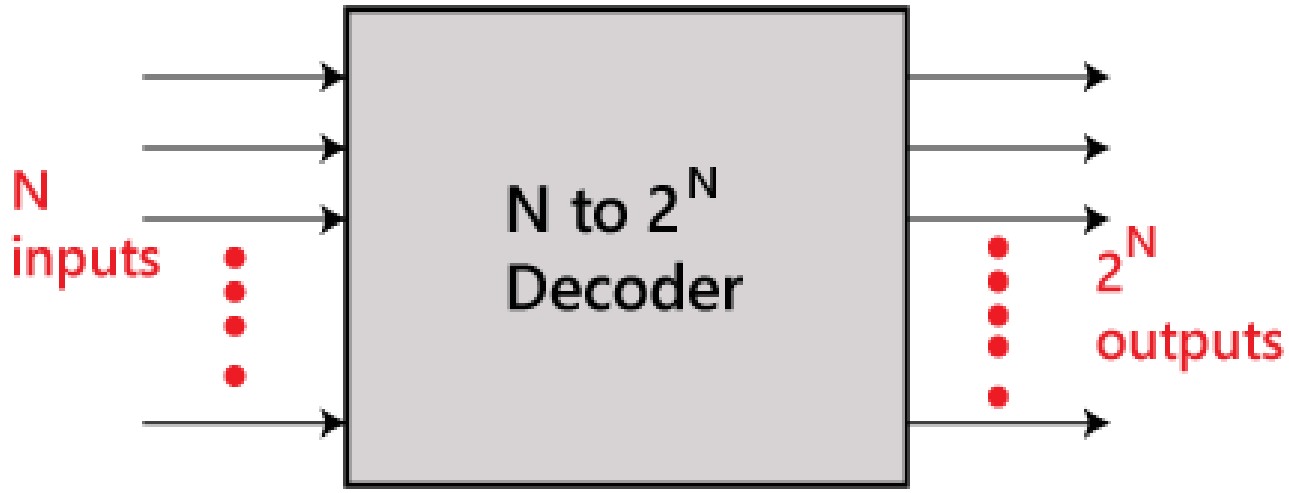
We can implement the above Boolean functions using logic gates.
The circuit diagram of 4 to 2 priority encoder is shown in the following figure.



Decoder

- Decoder is a combinational circuit that has ‘n’ input lines and maximum of 2n output lines.
- One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled.
- That means decoder detects a particular code.
- The outputs of the decoder are nothing but the min terms of ‘n’ input variables lines, when it is enabled.

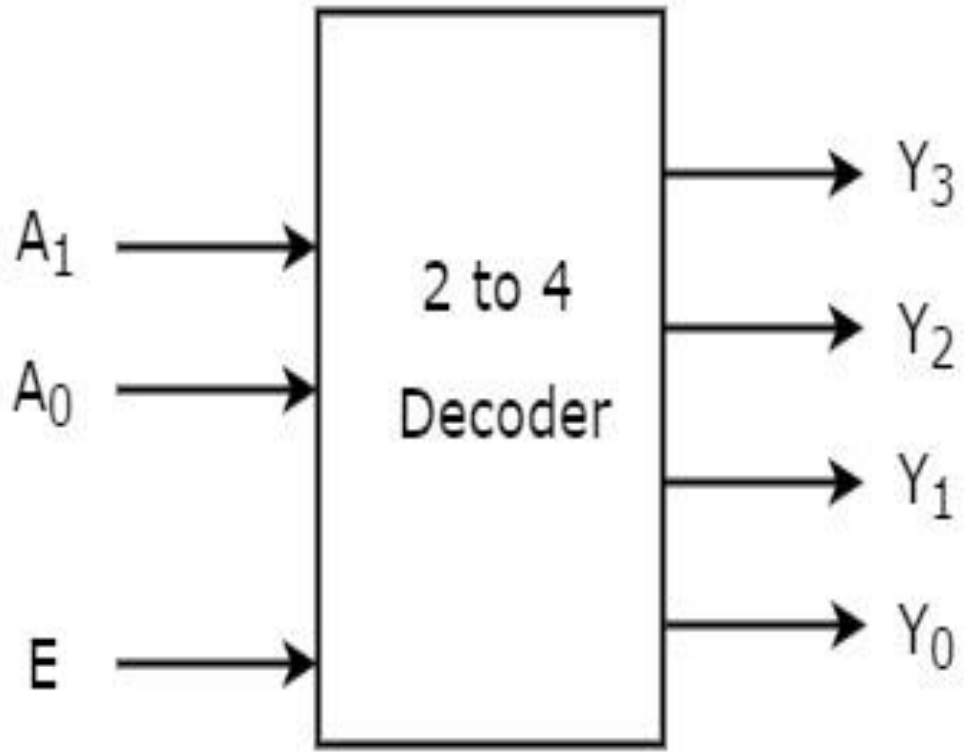
The produced 2N-bit output code is equivalent to the binary information.



There are various types of decoders which are as follows:

2 to 4 Decoder

- Let 2 to 4 Decoder has two inputs A1 & A0 and four outputs Y3, Y2, Y1 & Y0.
- The block diagram of 2 to 4 decoder is shown in the following figure.



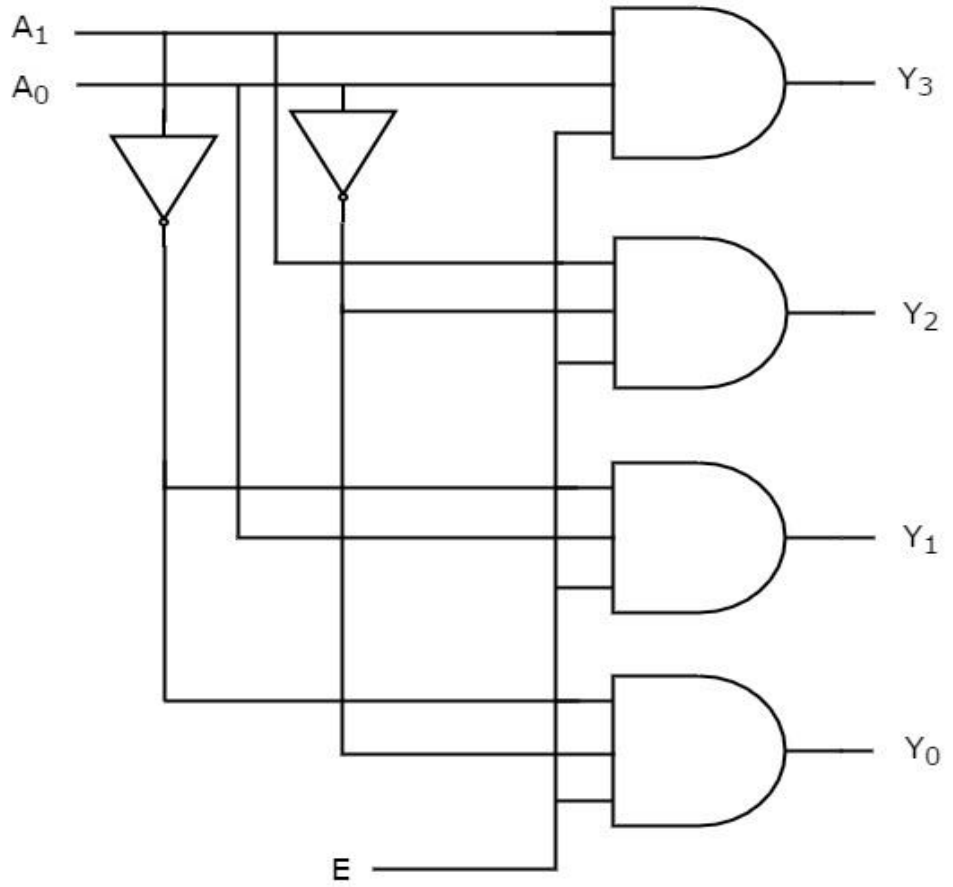
- One of these four outputs will be ‘1’ for each combination of inputs when enable, E is ‘1’.

The Truth table of 2 to 4 decoder is shown below.

Enable	Inputs		Outputs			
E	**A_1**	**A_0**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	x	x	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 write the Boolean functions for each output as
Y3=E.A1.A0
Y2=E.A1.A0’
Y1=E.A1’.A0
Y0=E.A1’.A0’

- Each output is having one product term.
- So, there are four product terms in total.
- We can implement these four product terms by using four AND gates having three inputs each & two inverters.
- The circuit diagram of 2 to 4 decoder is shown in the following figure.



3 to 8 line decoder:

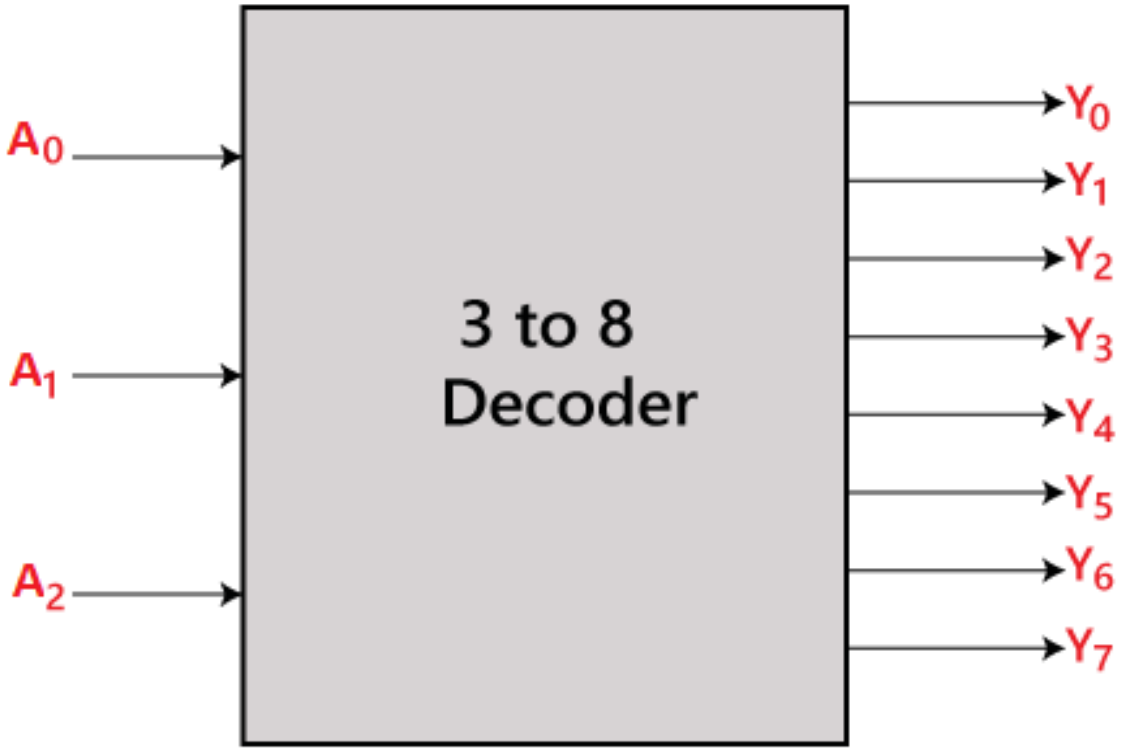
The 3 to 8 line decoder is also known as Binary to Octal Decoder.

In a 3 to 8 line decoder, there is a total of eight outputs, i.e., Y0, Y1, Y2, Y3, Y4, Y5, Y6, and Y7 and three outputs, i.e., A0, A1, and A2.

This circuit has an enable input 'E'. Just like 2 to 4 line decoder, when enable 'E' is set to 1, one of these four outputs will be 1.

The block diagram and the truth table of the 3 to 8 line encoder are given below.

Block Diagram:



Truth Table:

Enable	INPUTS			Outputs							
E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

The logical expression of the term Y0, Y1, Y2, Y3, Y4, Y5, Y6, and Y7 is as follows:

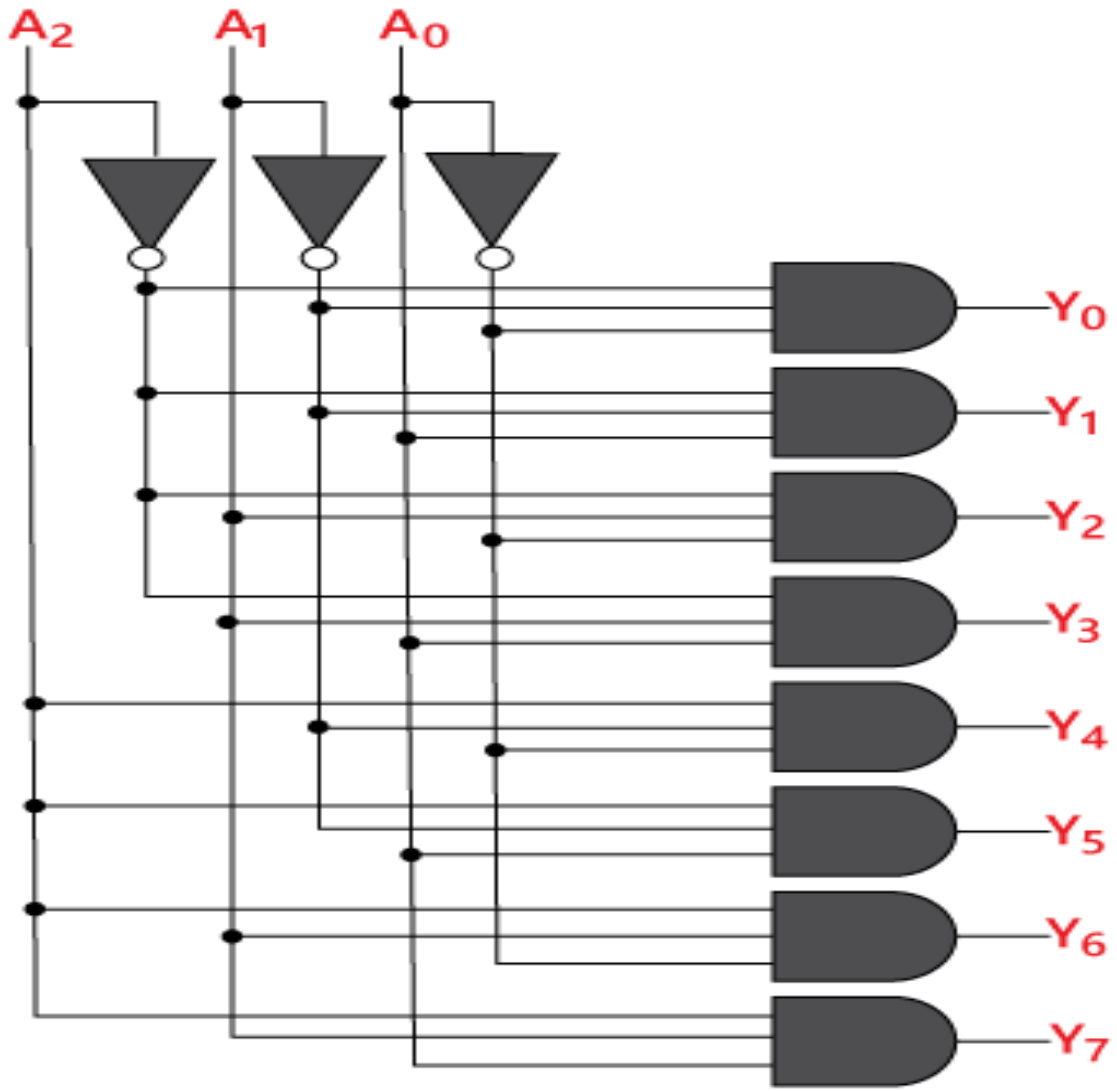
Y0=A0'.A1'.A2' Y4=A0'.A1'.A2

Y1=A0.A1'.A2' Y5=A0.A1'.A2

Y2=A0'.A1.A2' Y6=A0'.A1.A2

Y3=A0.A1.A2' Y7=A0.A1.A2

Logical circuit of the above expressions is given below:



4 to 16 line Decoder:

In the 4 to 16 line decoder, there is a total of 16 outputs, i.e., Y_0, Y_1, Y_2,……, Y_{16} and four inputs, i.e., A_0, A1, A_2, and A_3.

INPUTS				OUTPUTS															
A_3	A_2	A_1	A_0	Y_{15}	Y_{14}	Y_{13}	Y_{12}	Y_{11}	Y_{10}	Y_9	Y_8	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	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	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	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCD to 7 – Segment Decoder

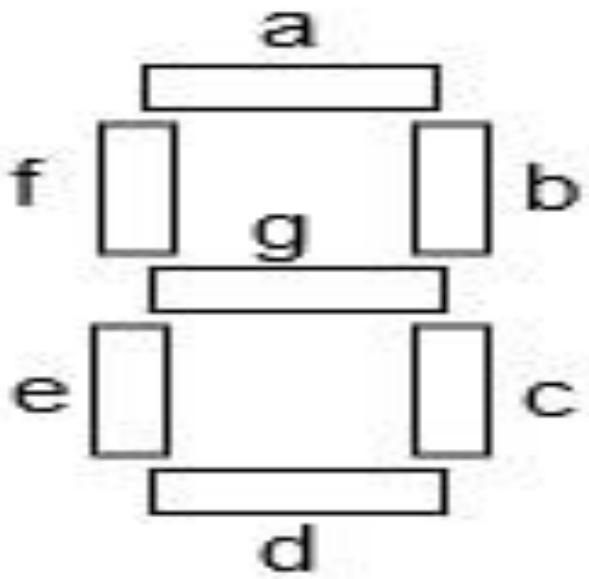
- Light Emitting Diode is most widely used semiconductor which emit either visible light or invisible infrared light when forward biased.
- A Light emitting diodes (LED) is an optical electrical energy into light energy when voltage is applied.

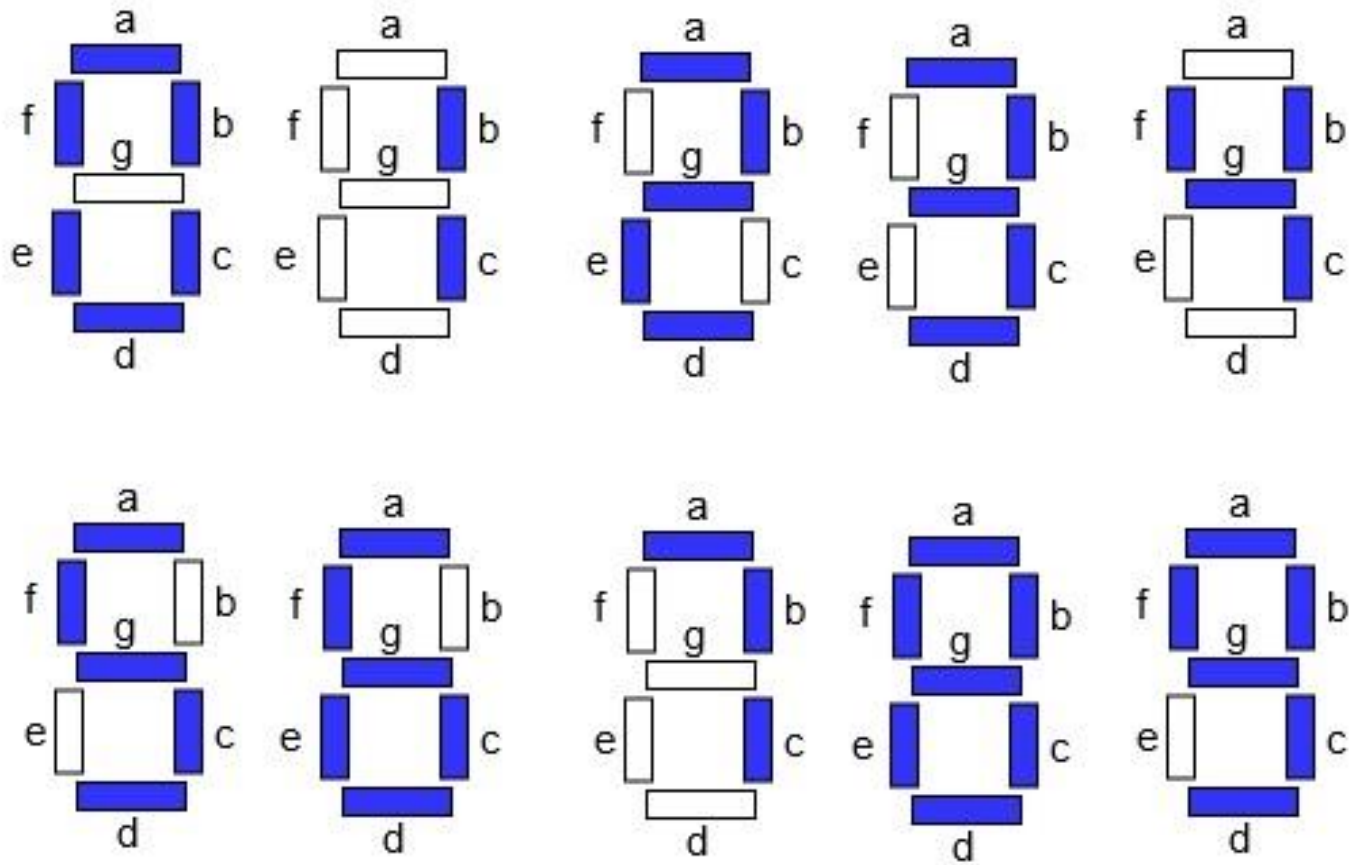
Seven Segment Displays

- Seven segment displays are the output display device that provide a way to display information in the form of image or text or decimal numbers which is an alternative to the more complex dot matrix displays.
- It is widely used in digital clocks, basic calculators, electronic meters, and other electronic device that display numerical information.

It consists seven segments of light emitting diodes (LEDs) which is assembled like numerical 8.

Working of Seven Segment Displays



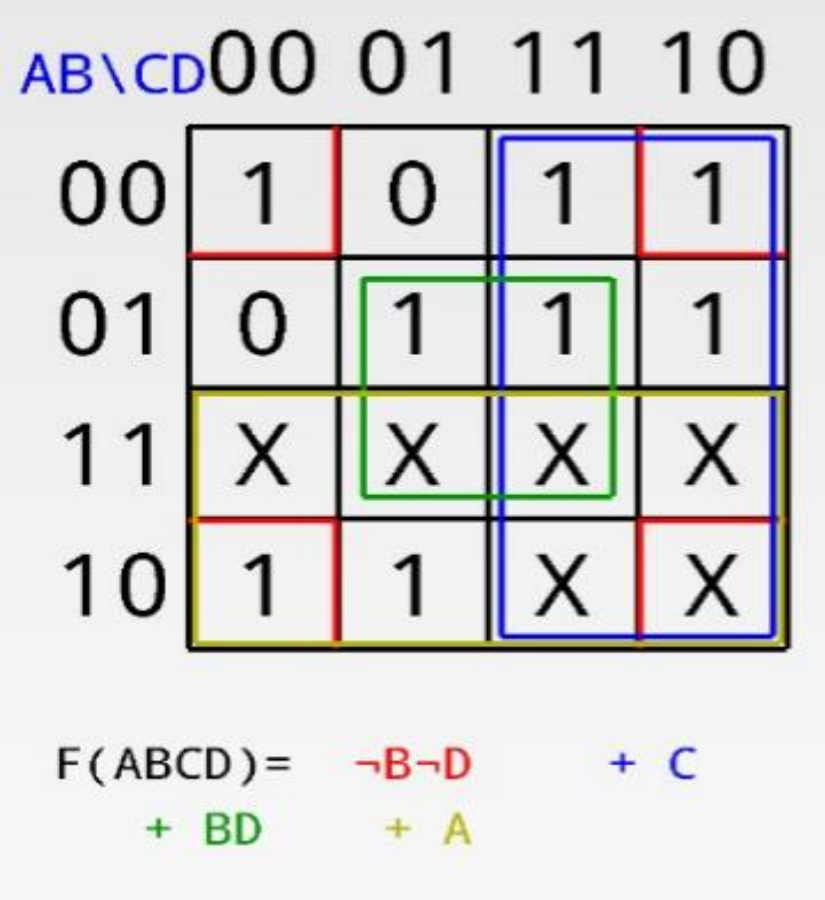

a
f
b
g
e
c
d

Truth Table BCD to seven 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	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

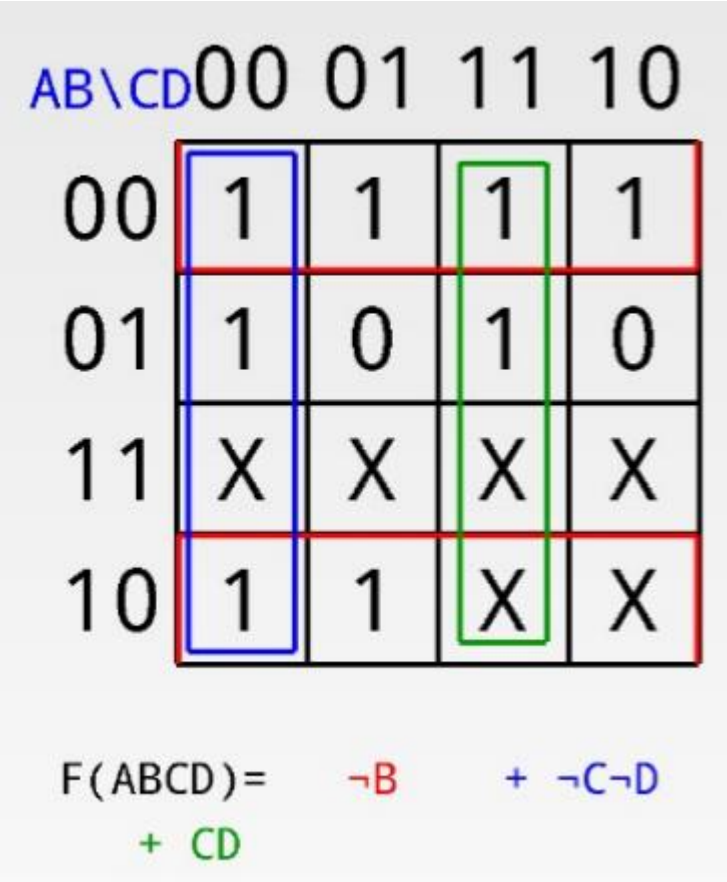
K-Maps:

#for a:

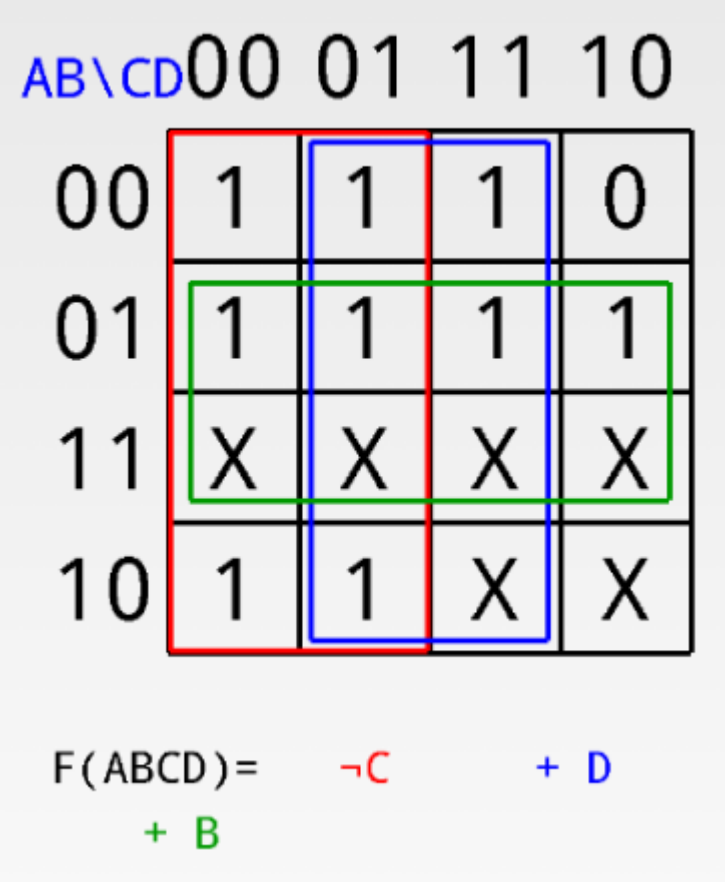


K-Maps:

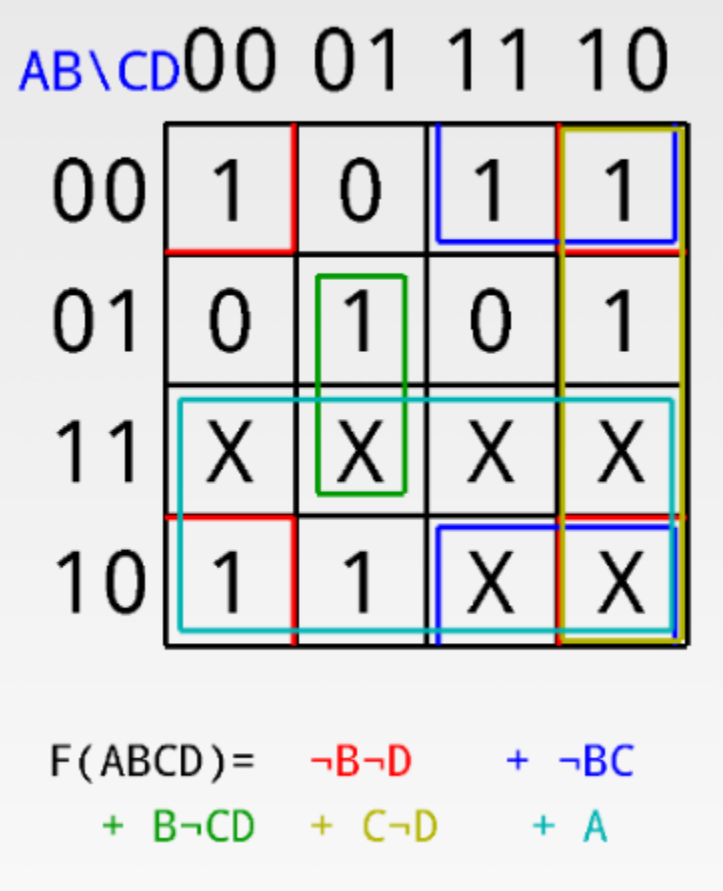
#for b:



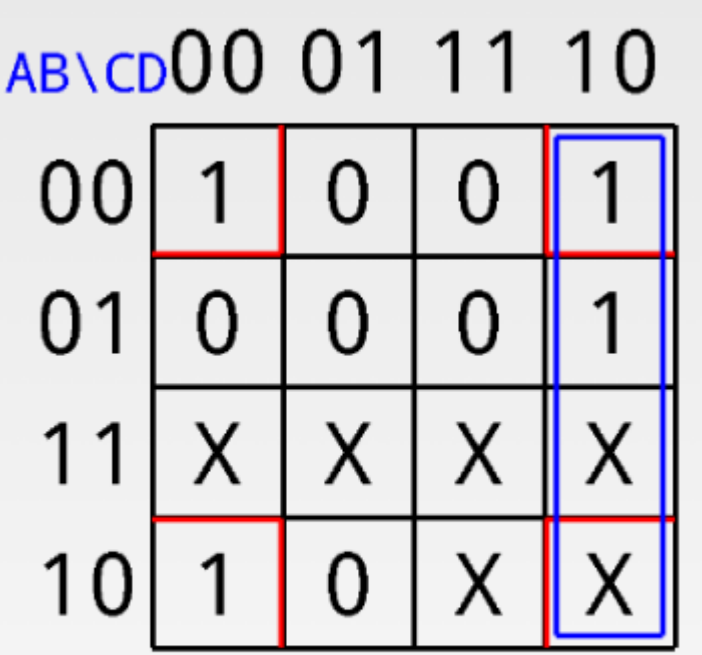
K-Maps:
#for c:



K-Maps:
#for d:

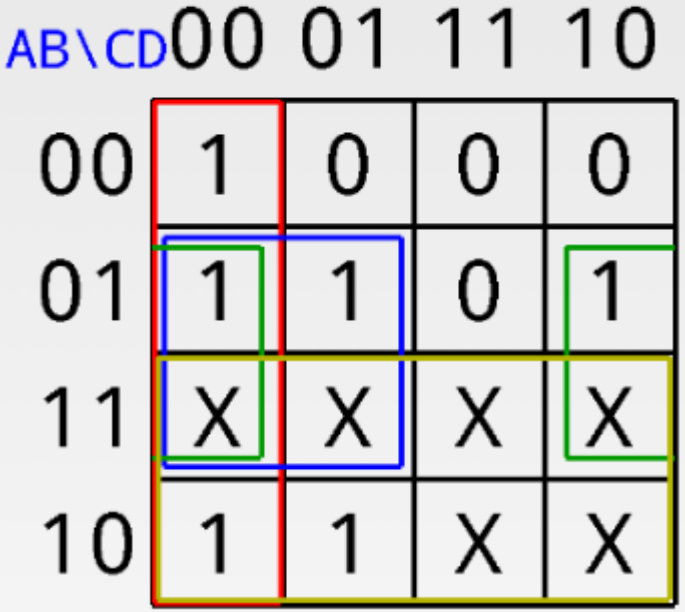


K-Maps:
#for e:



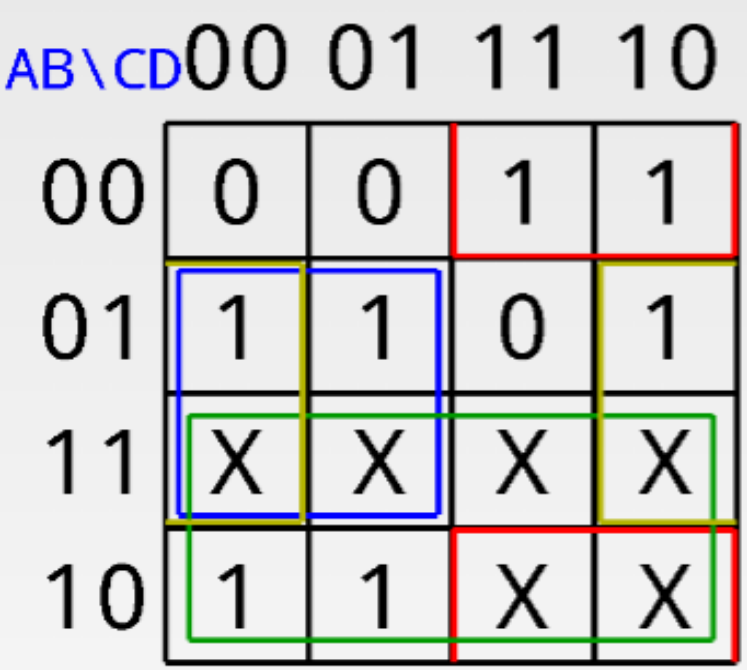
F(ABCD)= ¬B¬D + C¬D

K-Maps:
#for f:



F(ABCD)= ¬C¬D + B¬C
+ B¬D + A

K-Maps:
#for g:



F(ABCD)= ¬BC + B¬C
+ A + B¬D

From the above simplification, we get the output values as

$$a = A + C + BD + \overline{B}\,\overline{D}$$

$$b = \overline{B} + \overline{C}\,\overline{D} + CD$$

$$c = B + \overline{C} + D$$

$$d = \overline{B}\,\overline{D} + C\,\overline{D} + B\,\overline{C}\,D + \overline{B}\,C + A$$

$$e = \overline{B}\,\overline{D} + C\,\overline{D}$$

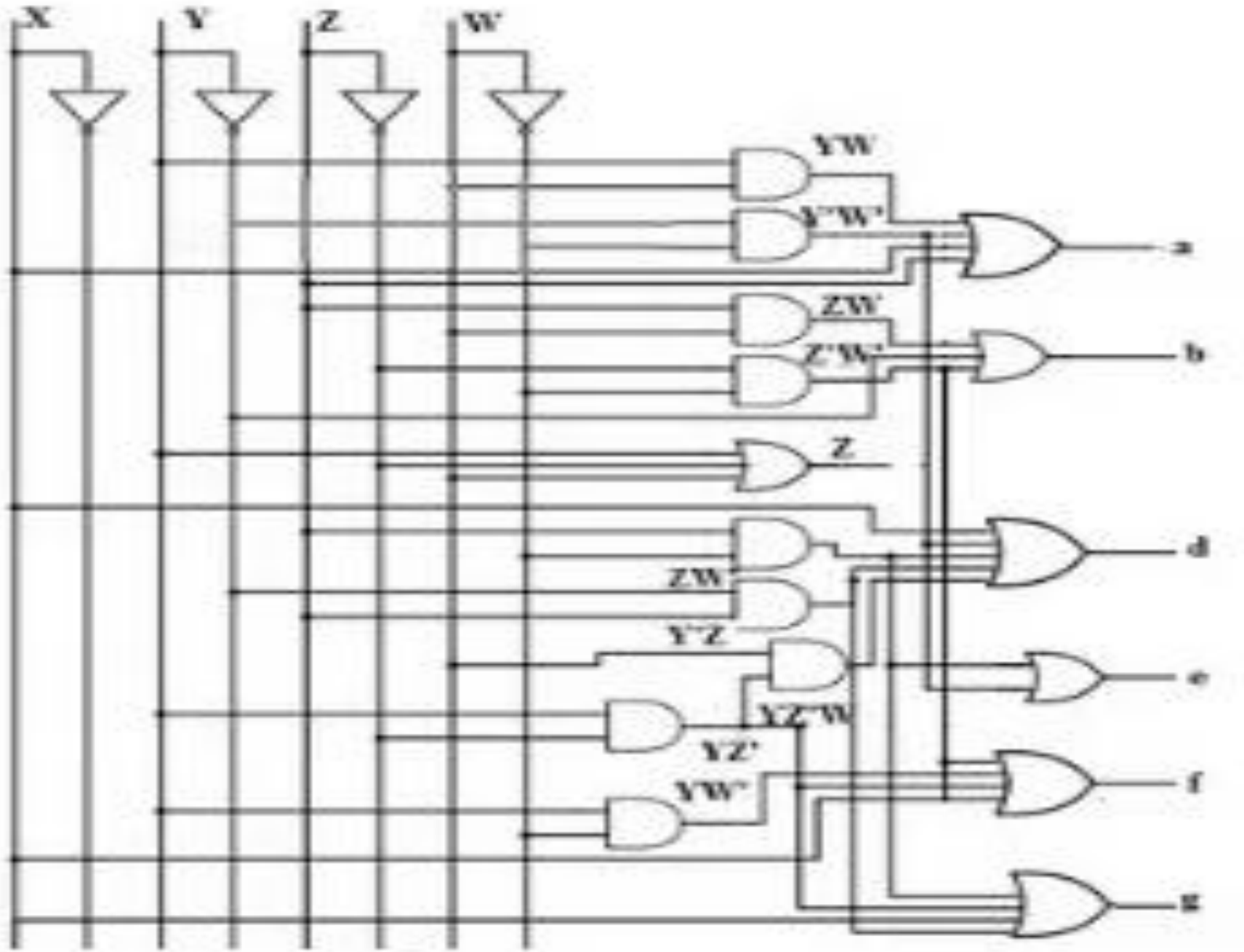
$$f = A + \overline{C}\,\overline{D} + B\,\overline{C} + B\,\overline{D}$$

$$g = A + B\,\overline{C} + \overline{B}\,C + C\,\overline{D}$$

Applications of Seven Segment Displays

Common applications of seven segment displays are in:

- Digital clocks
- Clock radios
- Calculators
- Wristwatchers
- Speedometers
- Motor-vehicle odometers
- Radio frequency indicators


X
Y
Z
W
YW
Y'W'
a
ZW
Z'W'
b
Z
d
ZW
Y'Z
e
XZ'W
YZ'
YW'
f
g

Programmable Logic Devices PLDs

PLDs are the integrated circuits that contain an array of AND gates & another array of OR gates.

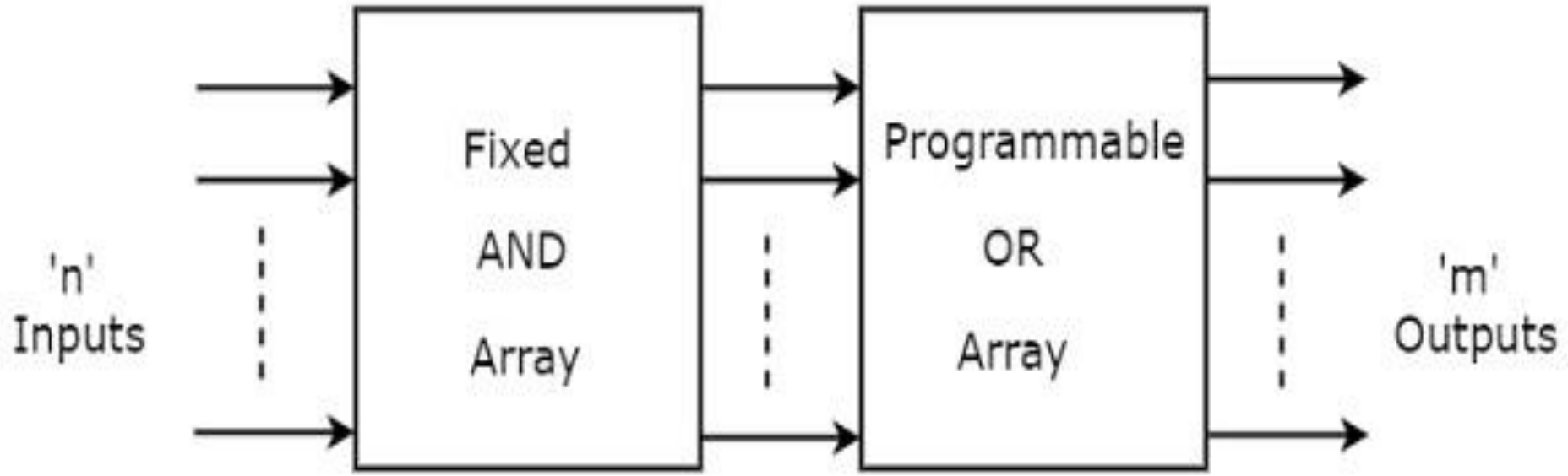
There are three kinds of PLDs based on the type of arrays, which has programmable feature.

- **Programmable Read Only Memory**
- **Programmable Array Logic**
- **Programmable Logic Array**

Programmable Read Only Memory PROM/ROM

- Read Only Memory ROM is a memory device, which stores the binary information permanently.
- That means, we can't change that stored information by any means later.
- If the ROM has programmable feature, then it is called as Programmable ROM PROM.
- The user has the flexibility to program the binary information electrically once by using PROM programmer.

- PROM is a programmable logic device that has fixed AND array & Programmable OR array.



The block diagram of PROM

- Here, the inputs of AND gates are not of programmable type.
- So, we have to generate 2^n product terms by using 2^n AND gates having n inputs each.
- We can implement these product terms by using $nx2^n$ decoder. So, this decoder generates 'n' **min terms**.

- Here, the inputs of OR gates are programmable.
- That means, we can program any number of required product terms, since all the outputs of AND gates are applied as inputs to each OR gate.
- Therefore, the outputs of PROM will be in the form of **sum of min terms**.

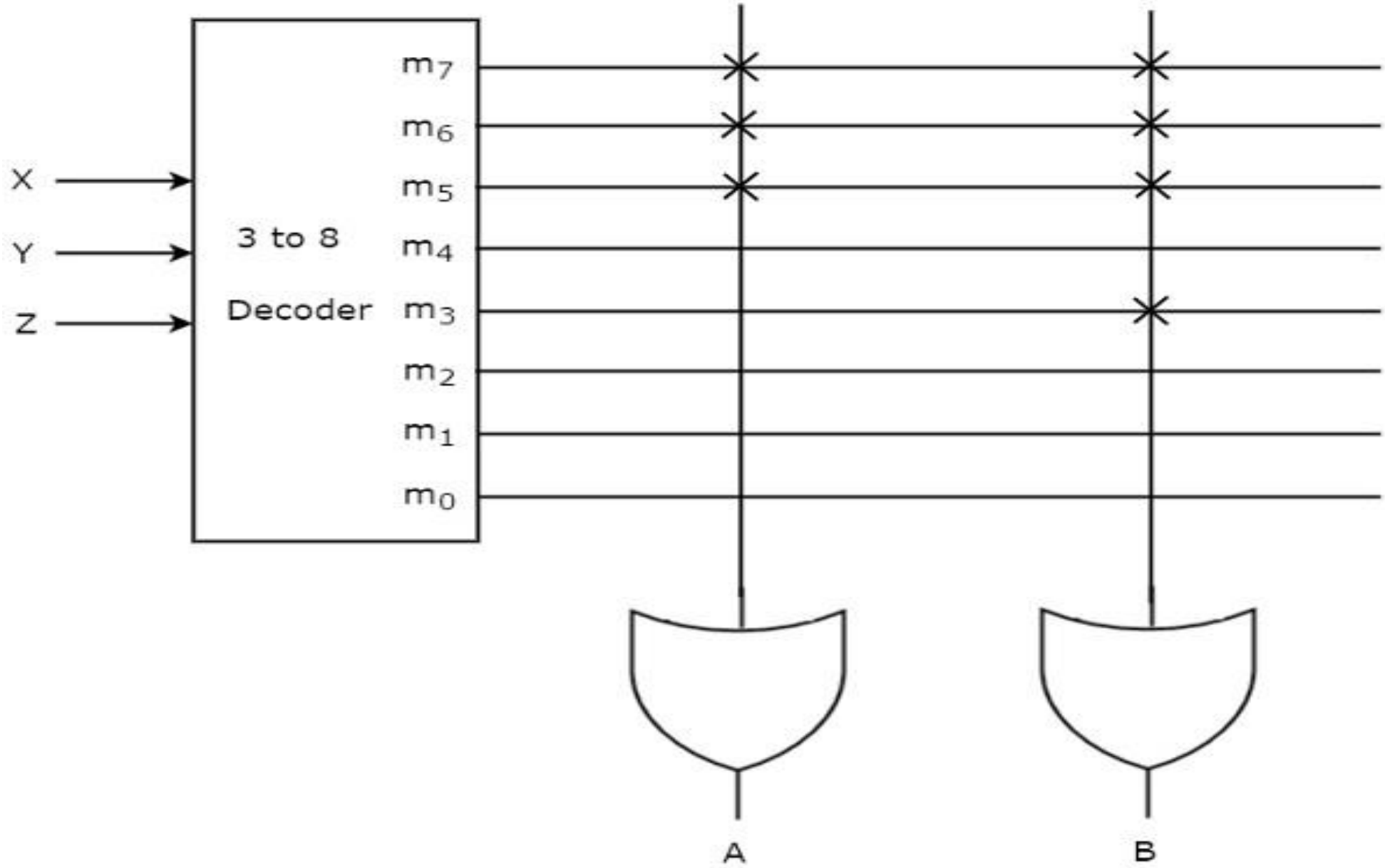
Example

Let us implement the following Boolean functions using PROM.

$A(X,Y,Z)=\sum m(5,6,7)A(X,Y,Z)=\sum m(5,6,7)$

$B(X,Y,Z)=\sum m(3,5,6,7)$

- The given two functions are in sum of min terms form and each function is having three variables X, Y & Z.
- So, we require a 3 to 8 decoder and two programmable OR gates for producing these two functions.
- The corresponding **PROM** is shown in the following figure.



Here, 3 to 8 decoder generates eight min terms. The two programmable OR gates have the access of all these min terms. But, only the required min terms are programmed in order to produce the respective Boolean functions by each OR gate. The symbol ‘X’ is used for programmable connections.

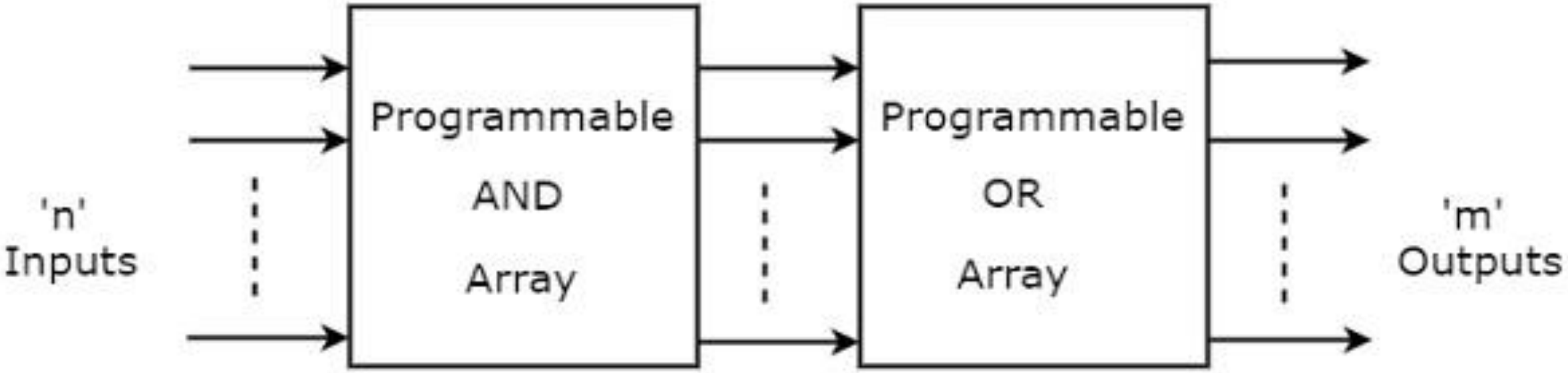
Programming Logic Array

- Programmable Logic Array(PLA) is a fixed architecture logic device with programmable AND gates followed by programmable OR gates.
- PLA is basically a type of programmable logic device used to build a reconfigurable digital circuit.
- PLDs have an undefined function at the time of manufacturing, but they are programmed before made into use.
- PLA is a combination of memory and logic.

Applications:

•PLA is used to provide control over Datapath.
•PLA is used as a counter.
•PLA is used as a decoder.
•PLA is used as a BUS interface in programmed I/O.

The block diagram of PLA is shown in the following figure.



- Here, the inputs of AND gates are programmable.
- That means each AND gate has both normal and complemented inputs of variables.
- So, based on the requirement, we can program any of those inputs.
- So, we can generate only the required **product terms** by using these AND gates.

- Here, the inputs of OR gates are also programmable.
- So, we can program any number of required product terms, since all the outputs of AND gates are applied as inputs to each OR gate.
- Therefore, the outputs of PAL will be in the form of **sum of products form**.

Example

Let us implement the following Boolean functions using PLA.

A=XY+XZ′A=XY+XZ′

B=XY′+YZ+XZ′B=XY′+YZ+XZ′

The given two functions are in sum of products form.

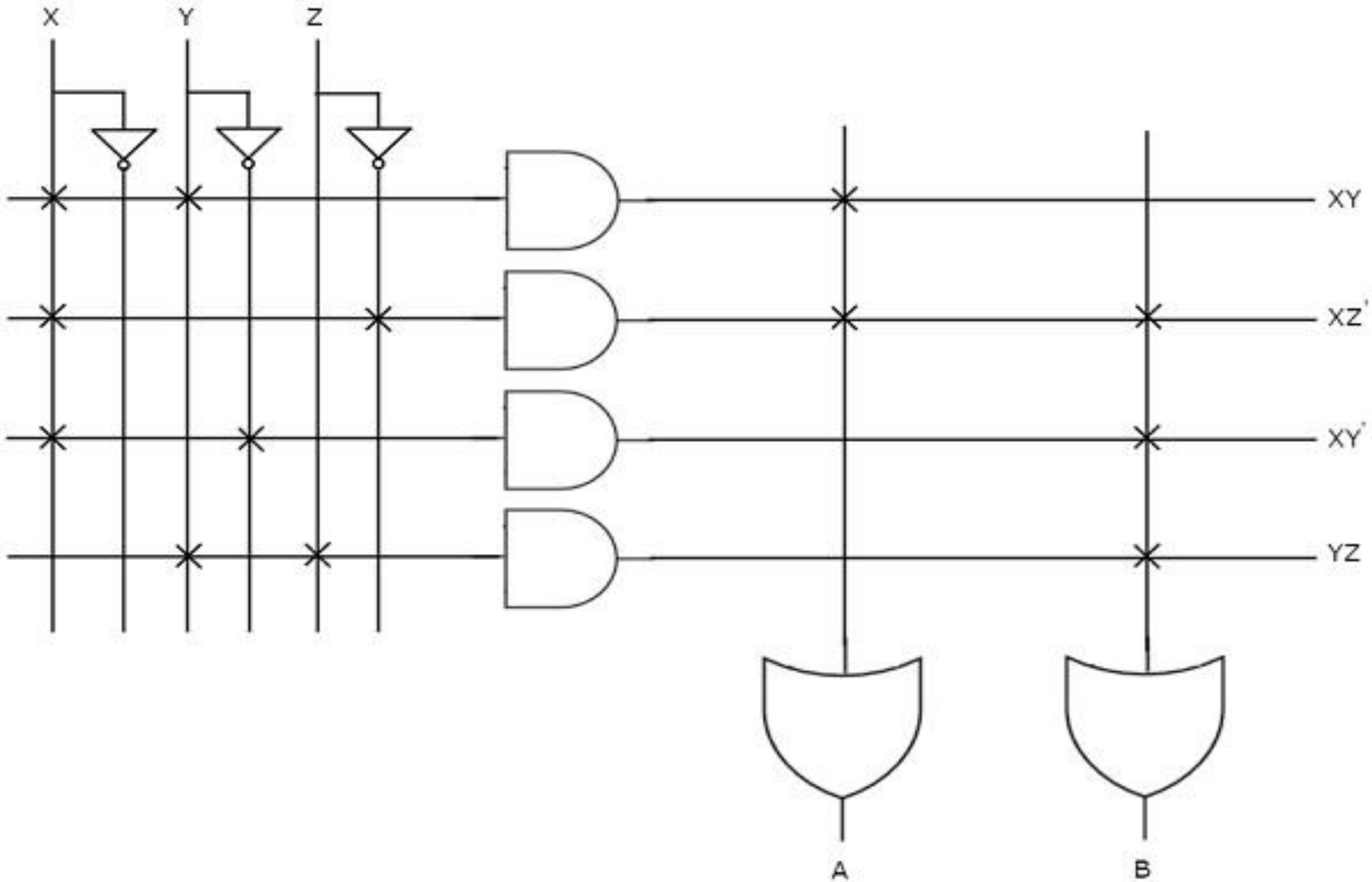
The number of product terms present in the given Boolean functions A & B are two and three respectively.

One product term, Z′XZ′X is common in each function.

So, we require four programmable AND gates & two programmable OR gates for producing those two functions.

The corresponding PLA is shown in the following figure.

Combinational Circuit of PLA



END of UNITS