Unit 4: Combinational Logic

Combinational Logic Circuits

Logic circuit in the digital system may be combinational or sequential.
A Combinational circuit consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regards to previous inputs. A sequential circuit employ the memory elements in addition to logic gates. Their output are the function of the inputs and the state of the memory elements.

Combinational Logic Circuits are memory less digital logic circuits whose output at any instant in time depends only on the combination of its inputs. The outputs of Combinational Logic Circuits are only determined by the logical function of their current input state, logic "0" or logic "1", at any given instant in time.

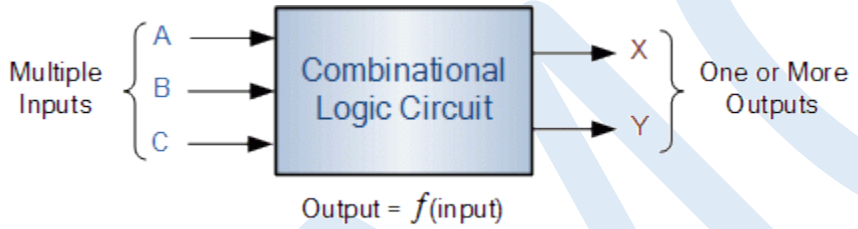


Figure 1: Block Diagram of Combinational Circuit

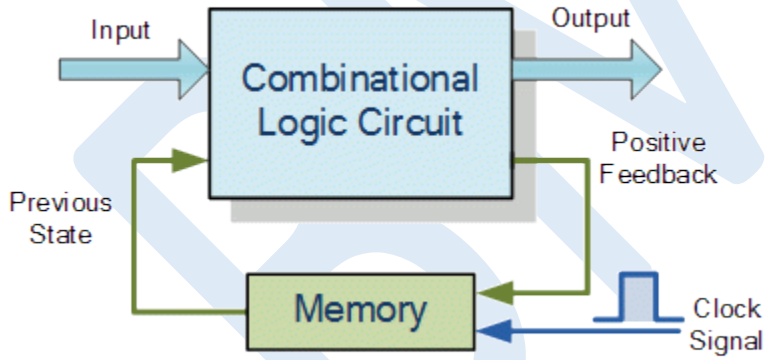


Figure 2Block diagram of sequential logic circuit

Combinational circuit consist of input variables, logic gates and out variables. The logic gates are combined in such a way that the output state depends entirely on the input states. Combinational logic circuits have "no memory", "timing" or "feedback loops", there operation is immediate.Examples of common combinational logic circuits include: half adders, full adders, multiplexers, demultiplexers, encoders and decoders.

Design Procedure of Combinational logic circuit

1. The problem is stated
2. The number of available input variables and required output variables is determined.
3. The input and output variables are assigned letter symbols
4. The truth table that defines the required relationship between inputs and output is derived.
5. The simplified Boolean function for each output is obtained.

6. The logic diagram is drawn.

Adders and Subtractors

Adder:

An Adder is a device that can add two binary digits.It is a type of digital circuit that performs the operation of additions of two numbers. It is mainly designed for the addition of binary number, but they can be used in various other applications like binary code decimal, address decoding, table index calculation, etc. There are two types of Adder.

1. Half Adder

Half Adder is a combinational circuit that performs the addition of two bit is called half- adder. There are two inputs and two outputs in a Half Adder. Inputs are named as A and B, and the outputs are named as Sum (S) and Carry (C).

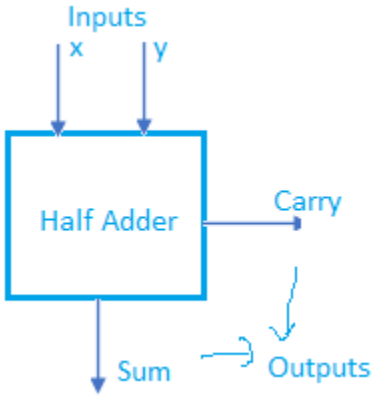


Figure: Block Diagram of Half Adder

The truth table of the half adder is shown below.

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

The simplified Boolean function for the two outputs can be obtained from the above truth table as:

Carry (c) = AB

Sum (S) = A'B+AB' | (i.e S=A⊕B)

Logical diagram for half adder from above Boolean function can be:

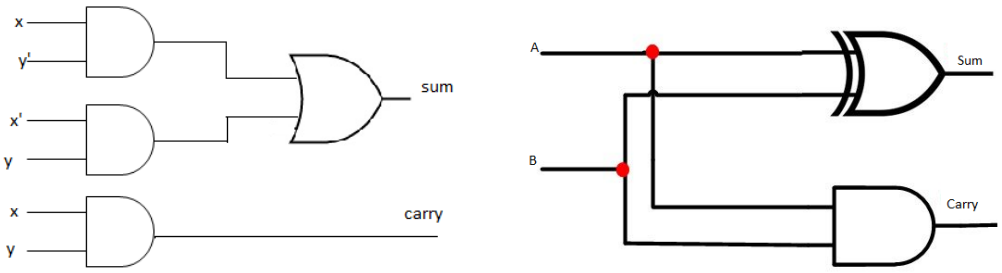


Figure 3: Half Adder

The main disadvantage of this circuit is that it can only add two inputs. To overcome this difficulty Full Adder is designed. While performing complex addition, there may be cases when you have to add two 8 bit bytes together. This can be done with the help of Full Adder.

2. Full Adder

A full adder is a combinational circuit that performs the addition of three input bits. It consists of three inputs and two outputs. The two of the input variables, denoted by x and y, represents the two significant bits to be added. The third input, z , represents the carry from the previous lower significant position. The two outputs are arithmetic sum (S) and Carry (C).

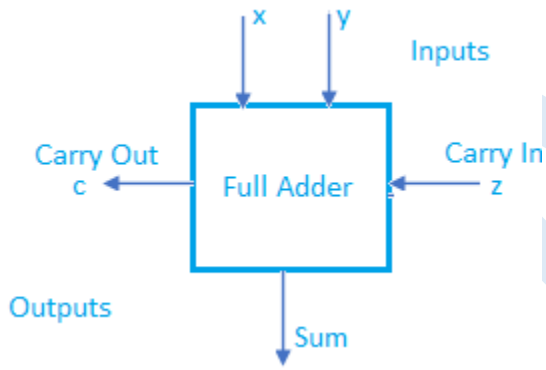


Figure: Block Diagram of Full Adder

The truth table for full-adder is given below:

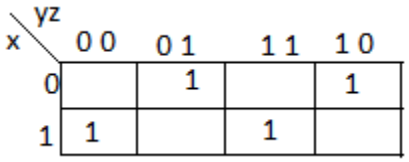
Inputs			Outputs	
x	Y	z	Carry(C)	Sum(S)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

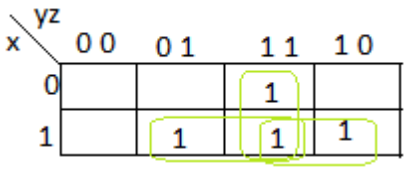
Boolean function for output are:

S= x'y'z + x'yz' + xy'z' + xyz

C=x'yz + xy'z + xyz' + xyz

Now, simplify the following Boolean function using k-map

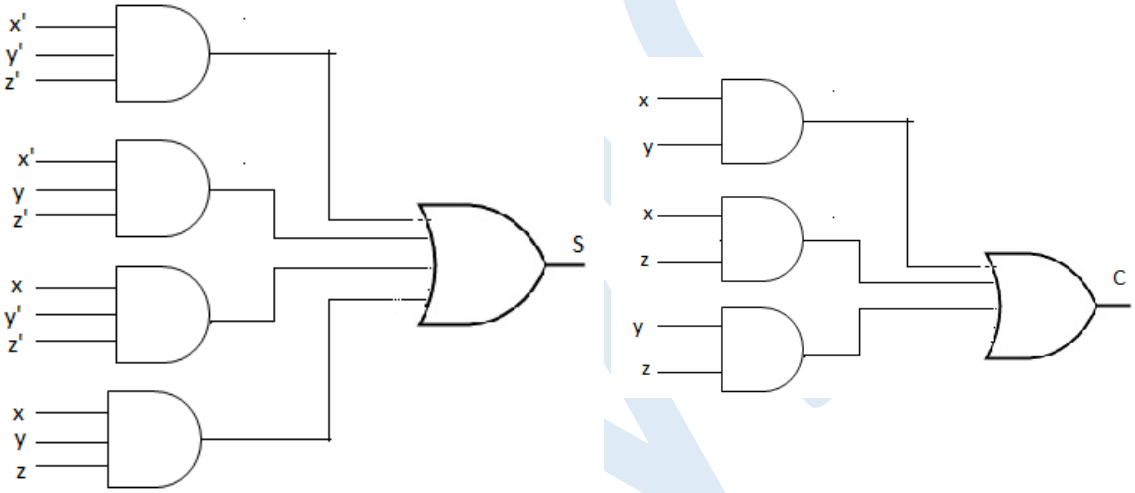




S= x'y'z + x'yz' + xy'z' + xyz

C= xy + xz + yz

Logical diagram for full adder.



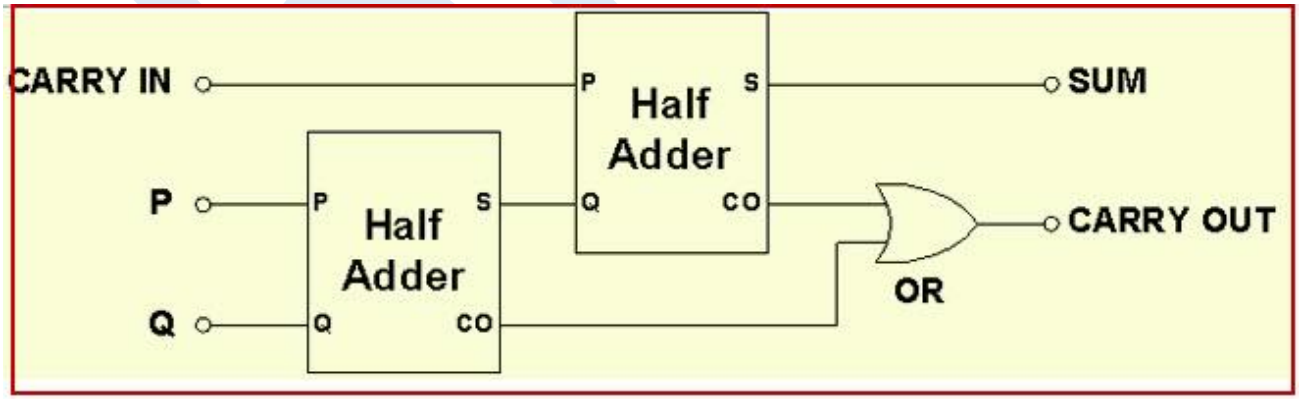


Figure 4Full-Adder by half-adder

Full adder can be implemented with two half-adder and one OR gate as shown below:

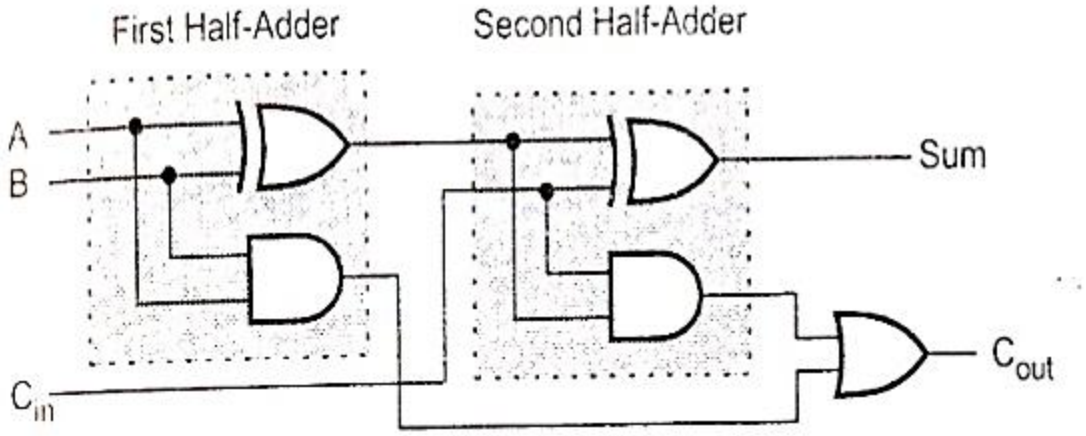


Figure: implementation of full adder with two half adder and one OR gate

The S output from the first half adder is the exclusive OR of Z gives:

$S = z \oplus (x \oplus y)$

=

Subtractors:

The Binary Subtractor is another type of combinational circuit that produces an output which is the subtraction of two binary numbers. Unlike the Binary Adder which produces a SUM and a CARRY bit when two binary numbers are added together, the binary subtractor produces a DIFFERENCE, D by using a BORROW bit, B from the previous column.

1. Half-Subtractor

A half subtractor is a logical circuit that performs a subtraction operation on two binary digits. The half subtractor produces a sum and a borrow bit for the next stage.

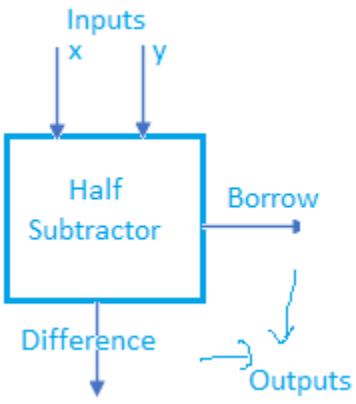


Figure: Block Diagram of Half Subtractor

Truth table

Inputs		Outputs	
X	Y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

The Boolean function for Borrow (B) and Difference (D) is:

B=X'Y

D=X'Y+XY'

Logic diagram for half-subtractor is

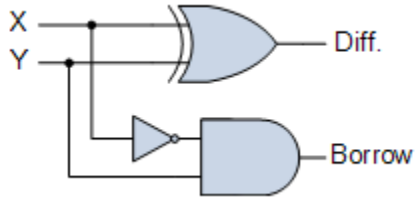


Figure 5: Half-Subtractor

2. Full- Subtractor

A full-subtractor is a combinational logic circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage. This circuit has three inputs and two outputs. The three inputs x,y and z denote the minuend, subtrahend, and previous borrow respectively. The two outputs, D and B, represents the difference and borrow, respectively.

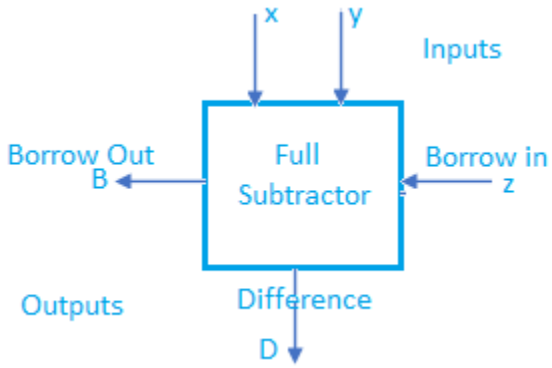


Figure: Block Diagram of full Subtractor

Truth table of full-subtractor is given below:

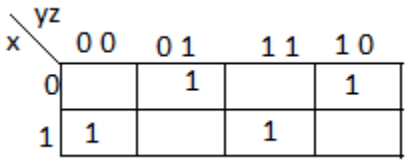
Inputs			Outputs	
x	Y	z	Borrow (B)	Difference (D)
0	0	0	0	0

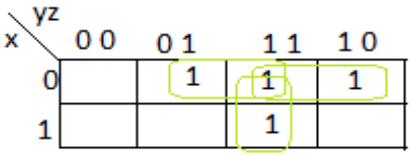
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

D= x'y'z + x'yz' + xy'z' + xyz

B= x'y'z+x'yz'+x'yz+xyz

Simplified Boolean expression using k-map is





D= x'y'z + x'yz' + xy'z' + xyz

B= x'y + x'z + yz

Logical Diagram:

Diagram required

Parallel Binary Adders

A parallel binary adder is a digital function that produces the arithmetic sum of two binary numbers in parallel. It consists of full adders connected in cascade, with the output carry from one full-adder connected to input carry of the next full-adder. An "n" bit parallel adder requires "n" full-adders. That is: for two-bit number, two adders are needed while for four bit number, four adders are needed and so on.

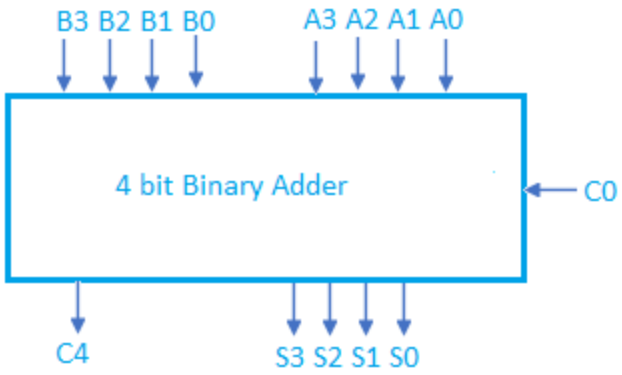


Figure: Block Diagram of 4 bit binary Adder

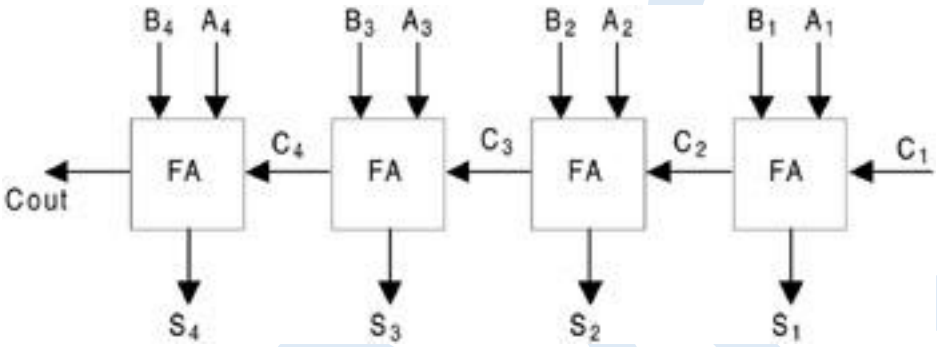


Figure: 4 bit full-adders

Consider the example that two 4-bit binary numbers $B_4\,B_3\,B_2\,B_1$ and $A_4A_3A_2A_1$ are to be added with a carry input C_1. This can be done by cascading four full adder circuits as shown in Figure. The least significant bits A_1, B_1, and C_1 are added to the produce sum output S_1 and carry output C_2. Carry output C_2 is then added to the next significant bits A_2 and B_2 producing sum output S_2 and carry output C_3. C_3 is then added to A_3 and B_3 and so on. Thus finally producing the four-bit sum output $S_4S_3S_2S_1$ and final carry output C_{out}.

Multiplexers

Multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of particular input line is controlled by a set of selection lines. Normally, there are 2^n input lines and *n* selection lines whose bit combination determines which input is selected.

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

4x1 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. The block diagram of 4x1 Multiplexer is shown in the following figure.

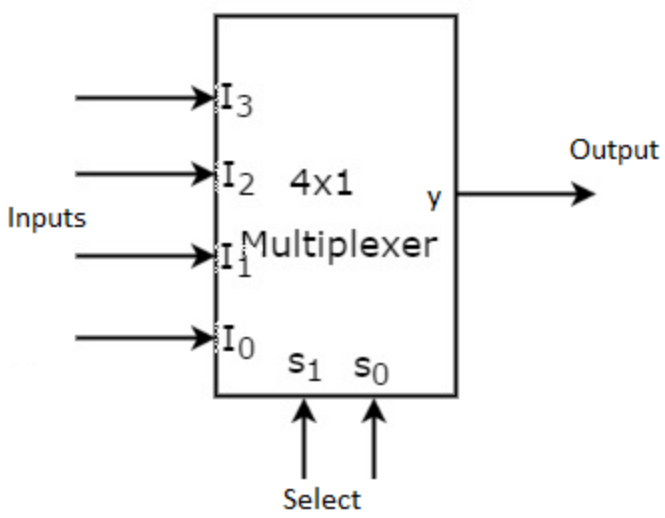


Figure 6: Block Diagram of 4×1 Mux

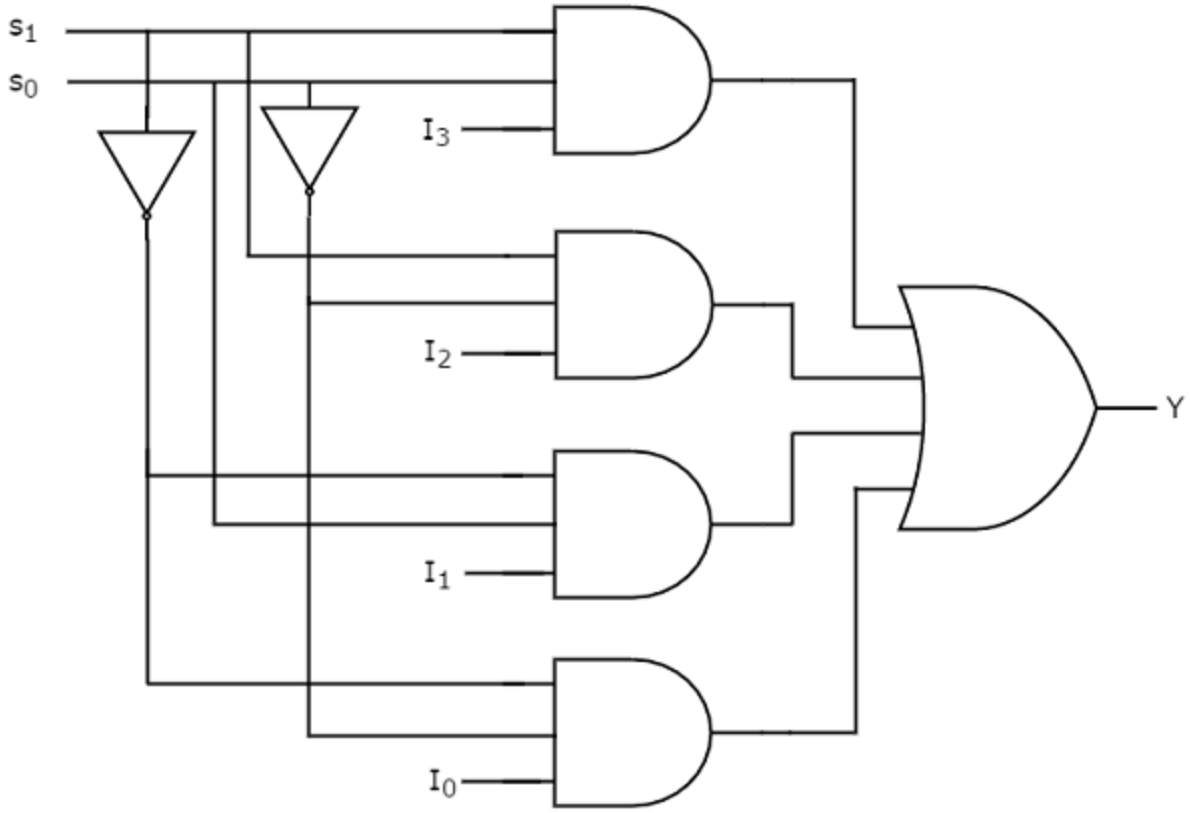
One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines. **Truth table** of 4x1 Multiplexer is shown below.

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

$Y=S_1'S_0'I_0+S_1'S_0I_1+S_1S_0'I_2+S_1S_0I_3$

The circuit diagram of 4x1 mux is given below:



Question: Construct 8×1 Mux

Truth table:

Selection Inputs			Output
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

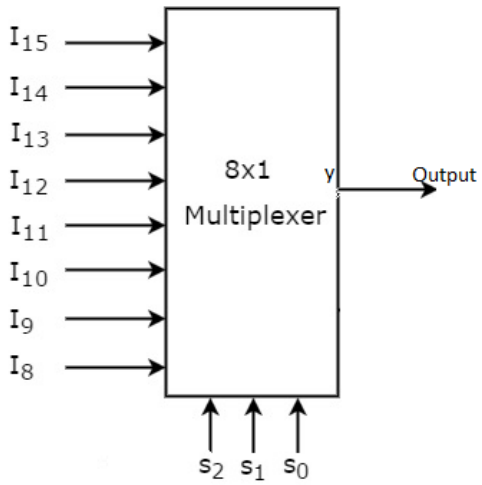
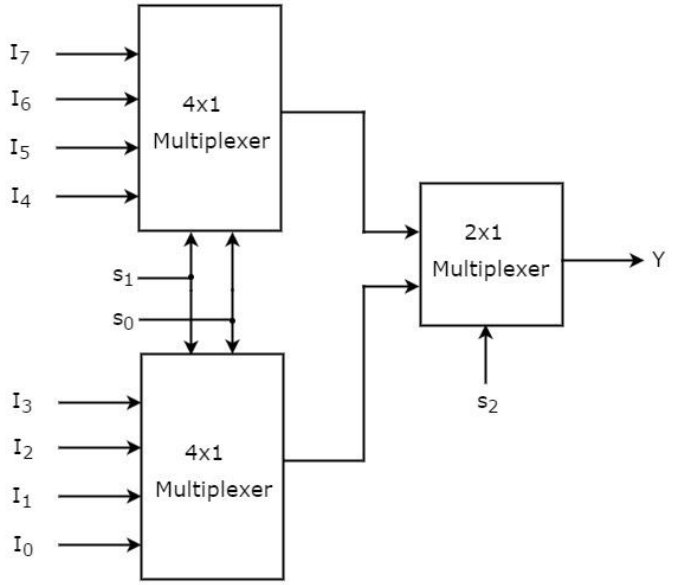


Figure 7: Block Diagram of 8×1 Mux

The circuit diagram of 8x1 mux is given below:

Implementation of 8×1 mux using two 4×1 mux and one 2×1 mux

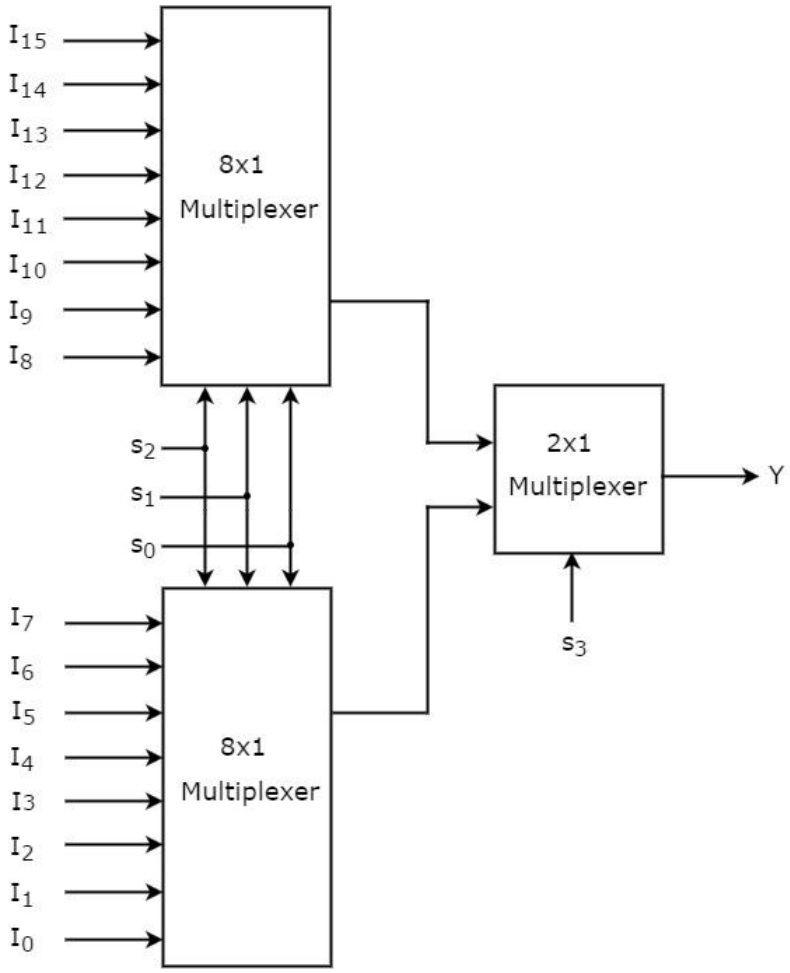


The outputs of first stage 4x1 Multiplexers are applied as inputs of 2x1 Multiplexer that is present in second stage. The other selection line, s_2 is applied to 2x1 Multiplexer.

- ⊕ If s_2 is zero, then the output of 2x1 Multiplexer will be one of the 4 inputs I_3 to I_0 based on the values of selection lines s_1& s_0.
- ⊕ If s_2 is one, then the output of 2x1 Multiplexer will be one of the 4 inputs I_7 to I_4 based on the values of selection lines s_1& s_0.

Therefore, the overall combination of two 4x1 Multiplexers and one 2x1 Multiplexer performs as one 8x1 Multiplexer.

Implementation of 16×1 mux using two 8×1 mux and one 2×1 mux



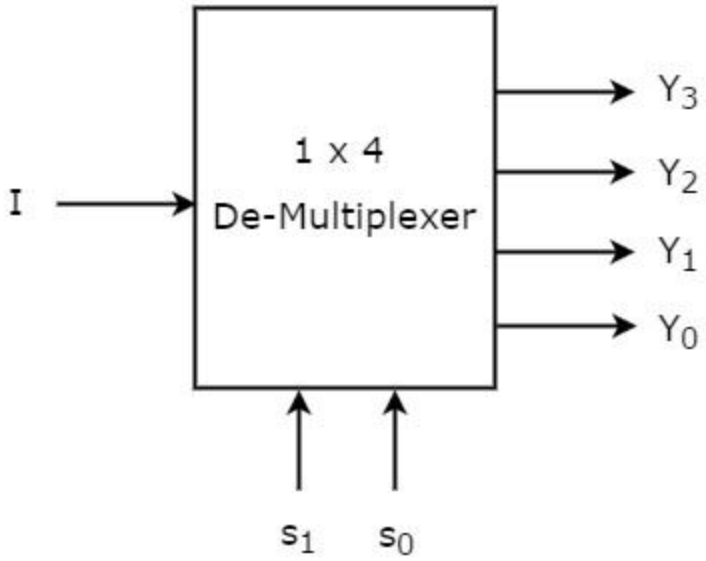
Demultiplexers

De-Multiplexer is a combinational circuit that performs the reverse operation of Multiplexer. It has single input, 'n' selection lines and maximum of 2^n outputs. The input will be connected to one of these outputs based on the values of selection lines.

Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination can select only one output. De-Multiplexer is also called as De-Mux.

1x4 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. The block diagram of 1x4 De-Multiplexer is shown in the following figure.



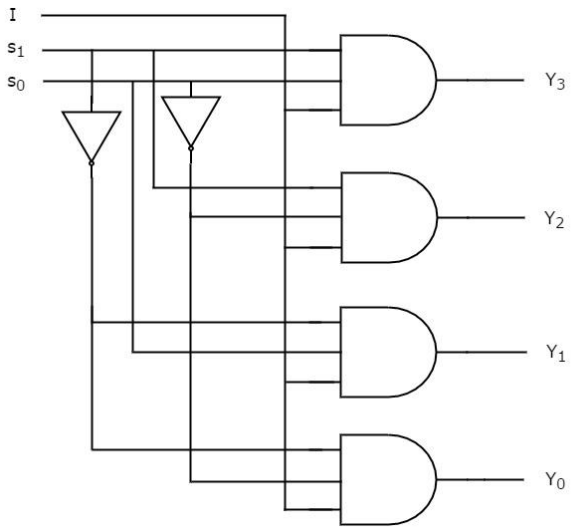
The Truth table of 1x4 De-Multiplexer is shown below.

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

From the above Truth table, the **Boolean functions** for each output is obtained as:

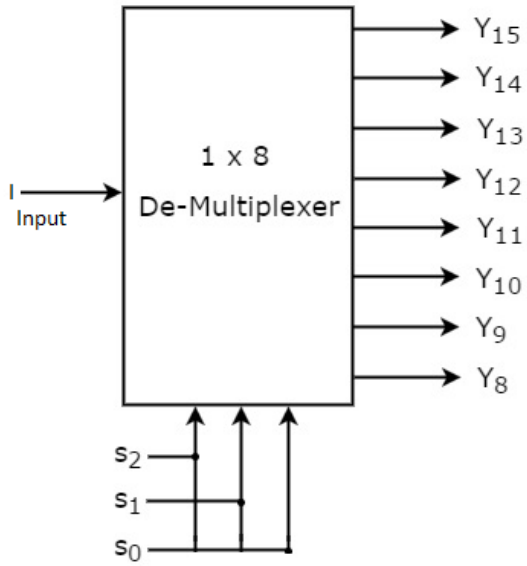
$Y_3=s_1s_0I$
$Y_2=s_1s_0'I$
$Y_1=s_1's_0I$
$Y_0=s_1's_0'I$

The circuit diagram of 1x4 De-Multiplexer is shown in the following figure.



1x8 De-Multiplexer

Block Diagram



Truth Table

Selection Inputs				Outputs						
s_2	s_1	s_0		Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0		0	0	0	0	0	0	**I**
0	0	1		0	0	0	0	0	**I**	0
0	1	0		0	0	0	0	**I**	0	0
0	1	1		0	0	0	**I**	0	0	0
1	0	0		0	0	**I**	0	0	0	0
1	0	1		0	**I**	0	0	0	0	0
1	1	0		**I**	0	0	0	0	0	0

1	1	1		0	0	0	0	0	0	0

The **Boolean functions** for each output is obtained as:

Y0=

Y1=

Y2=

Y3=

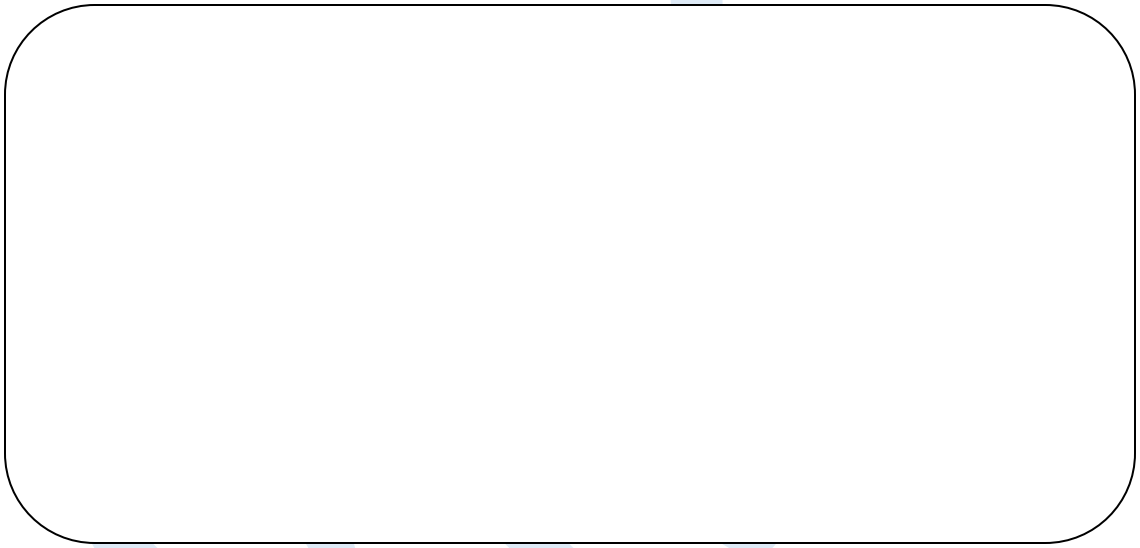
Y4=

Y5=

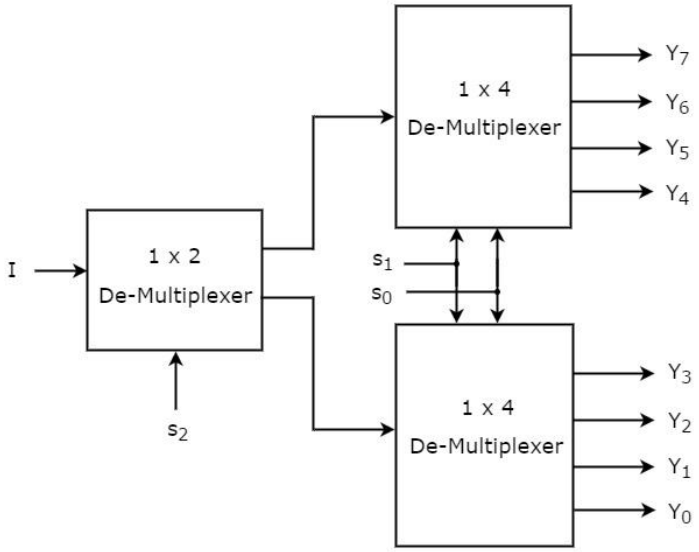
Y6=

Y7=

The circuit diagram of 1x8 De-Multiplexer is shown in the following figure.

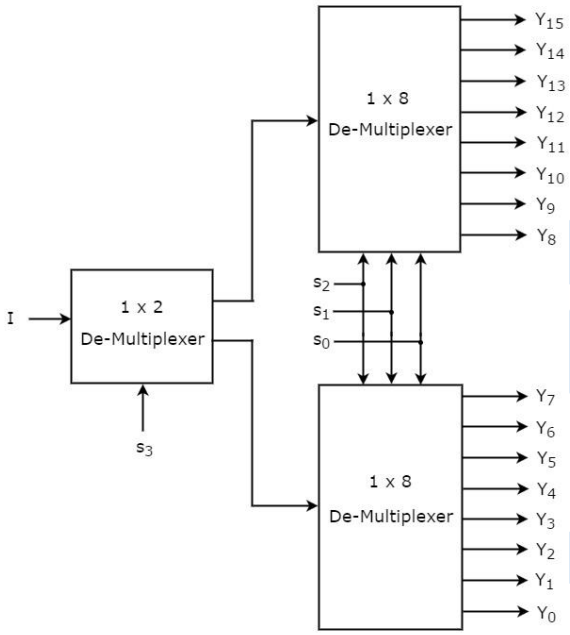


<u>Implementation of 1×8 De-mux using one 1×2 de-mux and two 1×4 De-mux.</u>



The common **selection lines, s_1 & s_0** are applied to both 1x4 De-Multiplexers. The outputs of upper 1x4 De-Multiplexer are Y_7 to Y_4 and the outputs of lower 1x4 De-Multiplexer are Y_3 to Y_0.

The other **selection line, s_2** is applied to 1x2 De-Multiplexer. If s_2 is zero, then one of the four outputs of lower 1x4 De-Multiplexer will be equal to input, I based on the values of selection lines s_1 & s_0. Similarly, if s_2 is one, then one of the four outputs of upper 1x4 DeMultiplexer will be equal to input, I based on the values of selection lines s_1 & s_0.



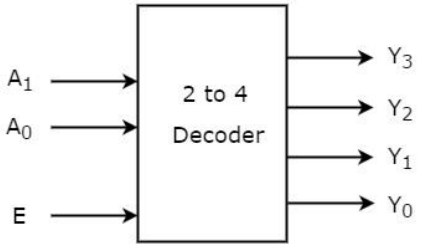
Encoders and decoders

Decoder

Decoder is a combinational circuit that has 'n' input lines and maximum of 2^n 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.

2 to 4 Decoder

Let 2 to 4 Decoder has two inputs A_1 & A_0 and four outputs Y_3, Y_2, Y_1 & Y_0. 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.

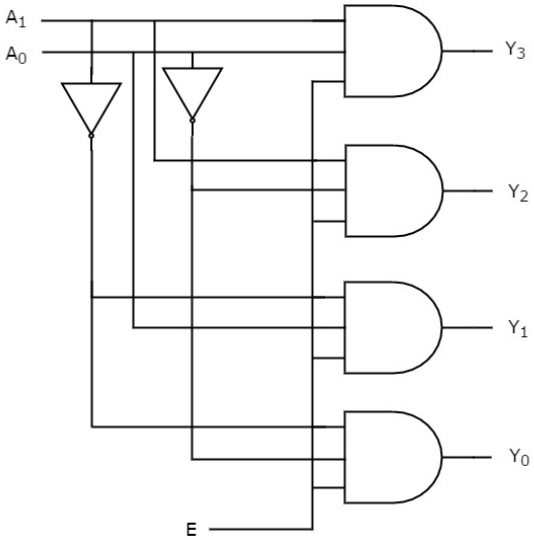
Enable	Inputs		Outputs		
E	A_1	A_0	Y_2	Y_1	Y_0
0	x	x	0	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0

"X" are don't care conditions

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

$Y_3=E.A_1.A_0$
$Y_2=E.A_1.A_0'$
$Y_1=E.A_1'.A_0$
$Y_0=E.A_1'.A_0'$

Each output is having one product term. So, there are four product terms in total. The circuit diagram of 2 to 4 decoder is shown in the following figure.



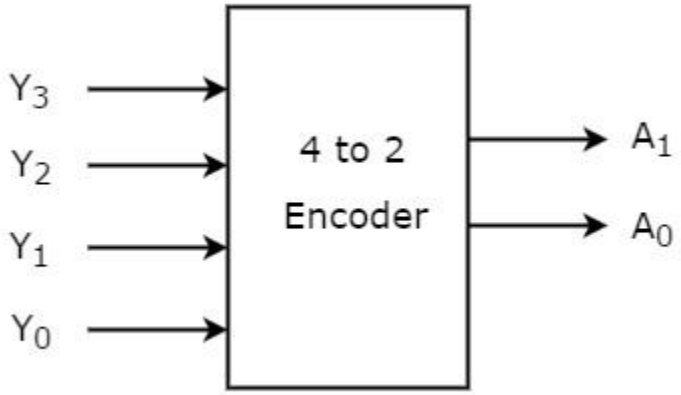
Therefore, the outputs of 2 to 4 decoder are nothing but the min terms of two input variables A1 & A0, when enable (E is equal to 1). If E is zero, then all the outputs of decoder will be equal to zero.

Encoder

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

4 to 2 Encoder

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



The Truth table of 4 to 2 encoder

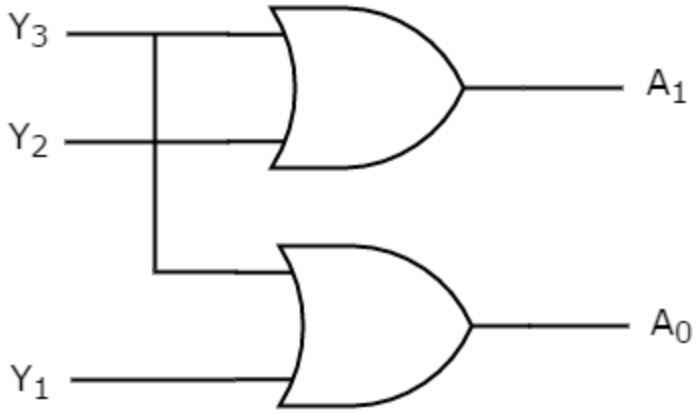
Inputs				Outputs	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Boolean functions for each output as

A1=Y3+Y2

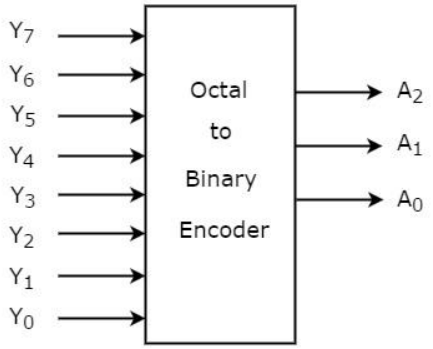
A0=Y3+Y1

The circuit diagram of 4 to 2 encoder is shown



Octal to Binary Encoder (8 to 3 encoder)

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

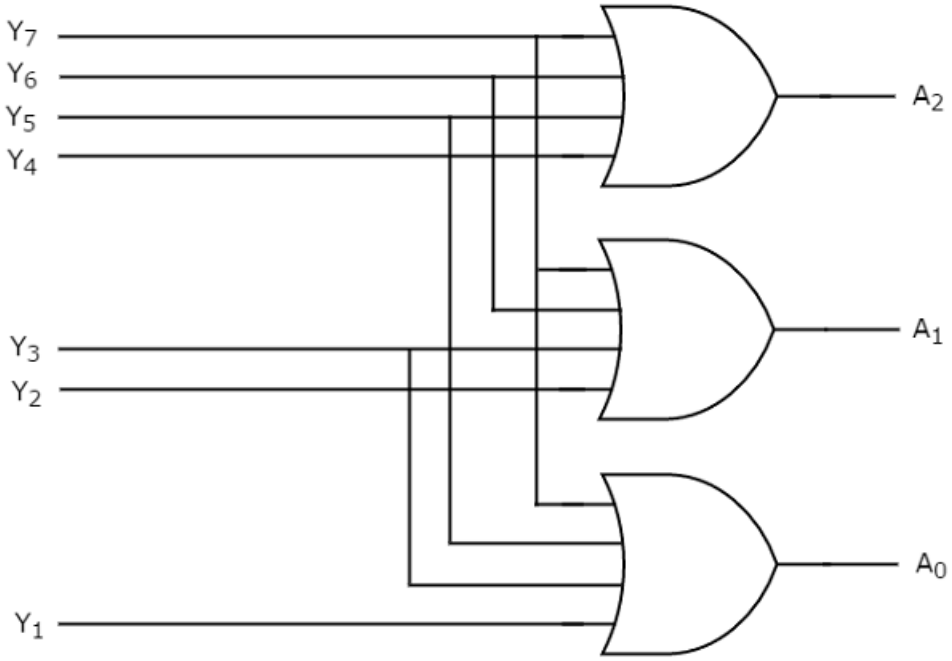


At any time, only one of these eight inputs can be '1' in order to get the respective binary code. The Truth table of octal to binary encoder is shown below.

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

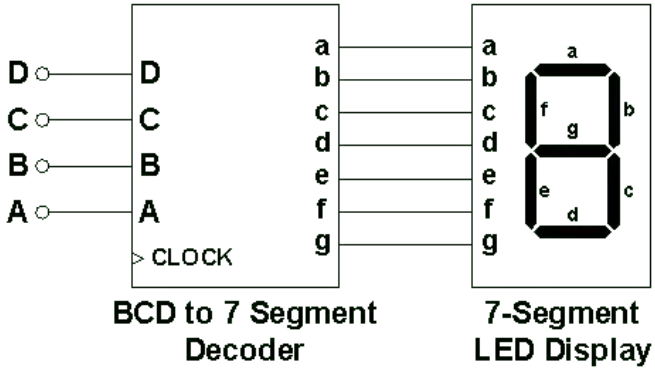
A2=Y7+Y6+Y5+Y4

A1=Y7+Y6+Y3+Y2
A0=Y7+Y5+Y3+Y1
The circuit diagram of octal to binary encoder is shown in the following figure.



Seven Segment Decoder

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.

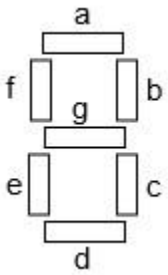


The number 8 is displayed when the power is given to all the segments and if you disconnect the power for 'g', then it displays number 0. In seven segment display, power (or voltage) at different pins can be applied at the same time, so we can form combinations of

display numerical from 0 to 9. Since seven segment display cannot form alphabet like X and Z, so it cannot be used for alphabet and it can be used only for displaying decimal numbers. However, seven segment displays can form alphabets A, B, C, D, E, and F, so it can also use for representing hexadecimal digits.

We can produce a truth table for each decimal digit.

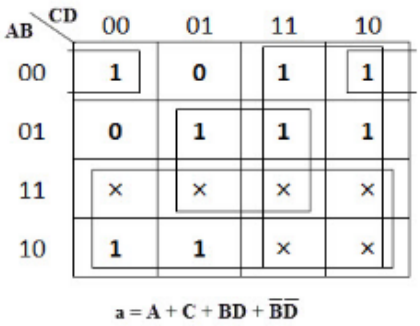
Decimal Digit	Individual Segments Illuminated						
	A	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	0	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	0	0	1	1



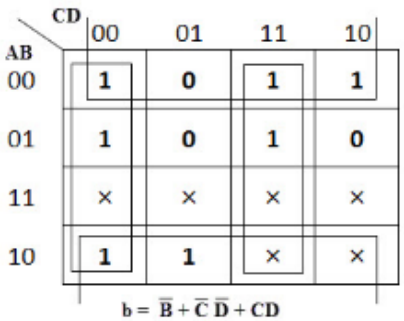
From the above truth table, the Boolean expressions of each output functions can be written as:

a = F1 (A, B, C, D) = ∑m (0, 2, 3, 5, 7, 8, 9)
b = F2 (A, B, C, D) = ∑m (0, 1, 2, 3, 4, 7, 8, 9)
c = F3 (A, B, C, D) = ∑m (0, 1, 3, 4, 5, 6, 7, 8, 9)
d = F4 (A, B, C, D) = ∑m (0, 2, 3, 5, 6, 8)
e = F5 (A, B, C, D) = ∑m (0, 2, 6, 8)
f = F6 (A, B, C, D) = ∑m (0, 4, 5, 6, 8, 9)
g = F7 (A, B, C, D) = ∑m (2, 3, 4, 5, 6, 8, 9)

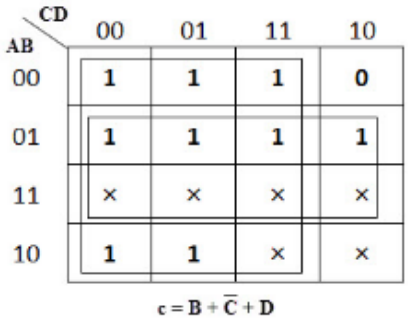
Now, simplify the given functions using k-map



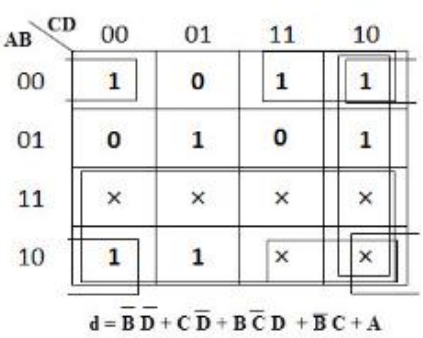
$a = A + C + BD + \overline{B}\overline{D}$



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

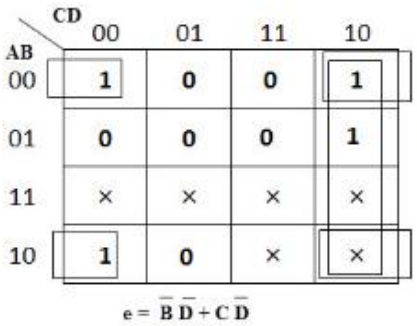


$c = B + \overline{C} + D$



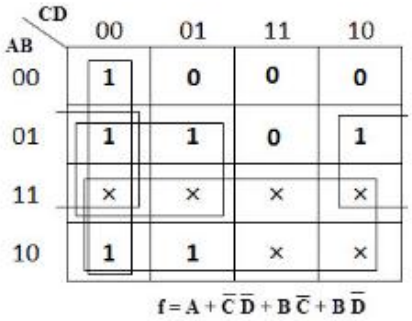
AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	×	×	×	×
10	1	1	×	×

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



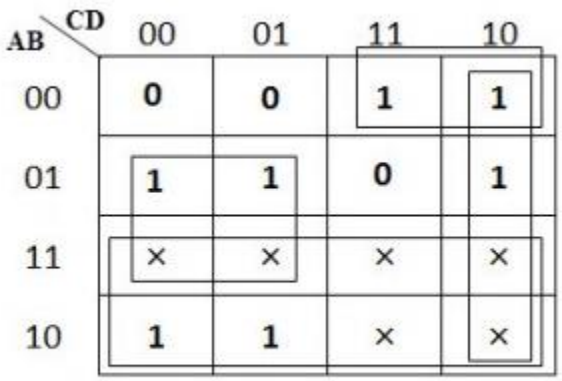
AB \ CD	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	×	×	×	×
10	1	0	×	×

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



AB \ CD	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	×	×	×	×
10	1	1	×	×

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



AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	×	×	×	×
10	1	1	×	×

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

From the above simplification, we get the simplified function as:

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

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

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

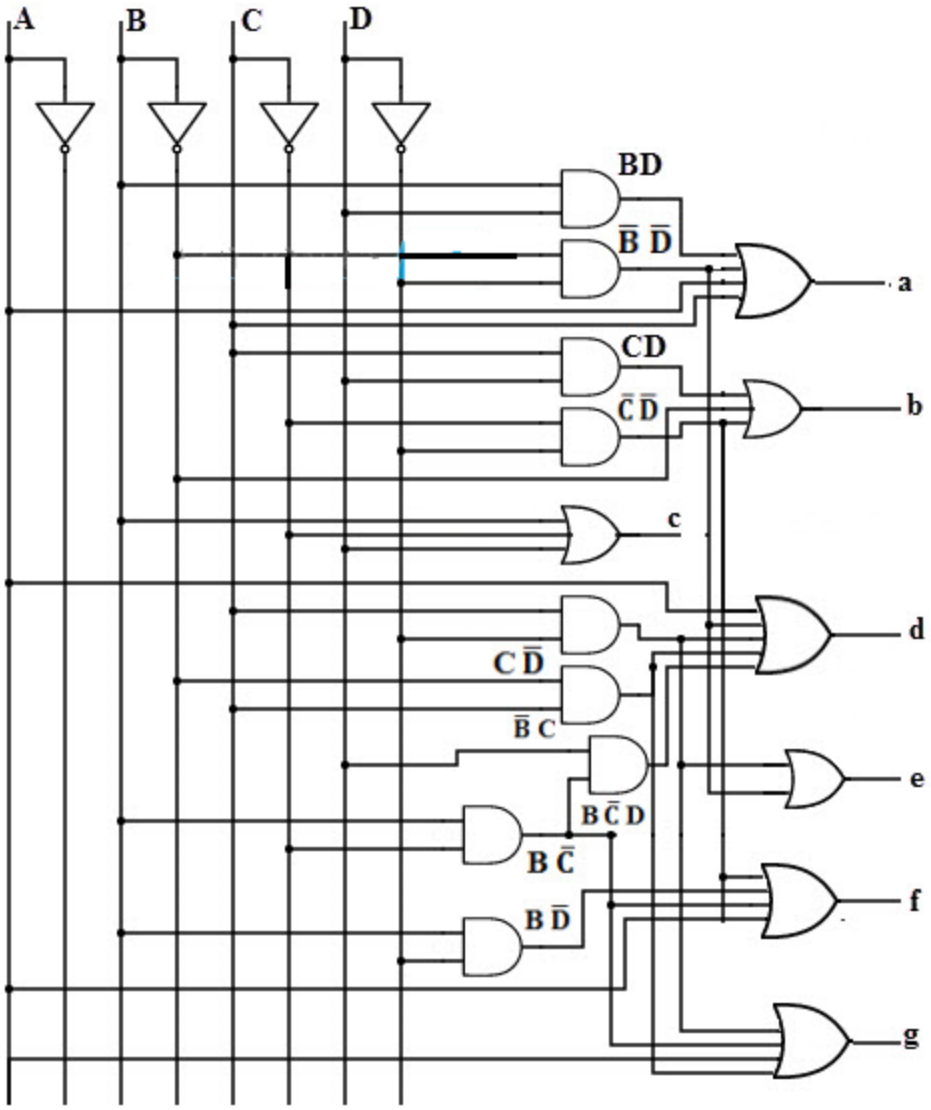
$d = \overline{B}\,\overline{D} + C\,\overline{D} + B\,\overline{C}\,D + \overline{B}\,C + A$

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

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

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

Logic circuit



Code Converters

It is sometime necessary to use the output of one system as the input of another system of different digital logic. A conversion circuit must be inserted between the two systems if each used different codes for the same information. Thus, a code converter is a circuit that makes the two system compatible even though each uses a different binary codes. This code conversion can be illustrated by the following examples:

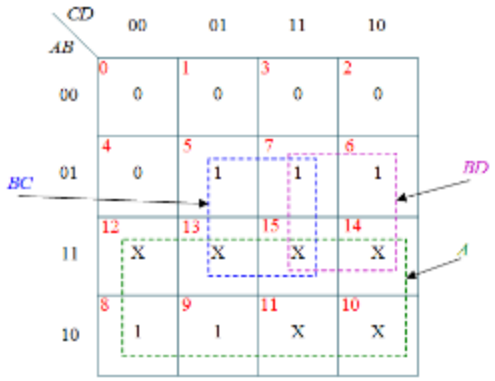
a) BCD to excess-3
b) Binary to gray code
c) Gray to binary

BCD to Excess-3

BCD digit can be converted to its corresponding Excess-3 code by simply adding 3 to it.

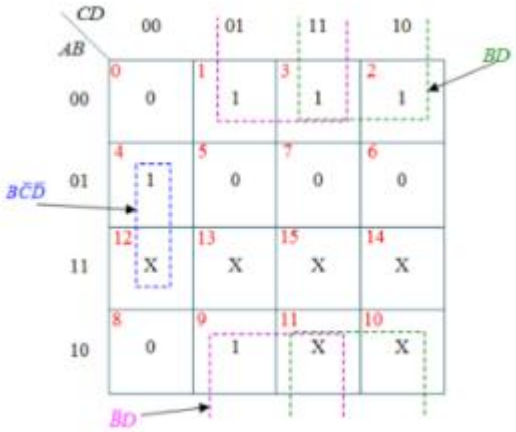
The truth table for the conversion is given below. The don't care conditions are represented by (X) marks.

BCD Code (Input)				Excess-3 Code (Output)			
A	*B*	*C*	*D*	*W*	*X*	*Y*	*Z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



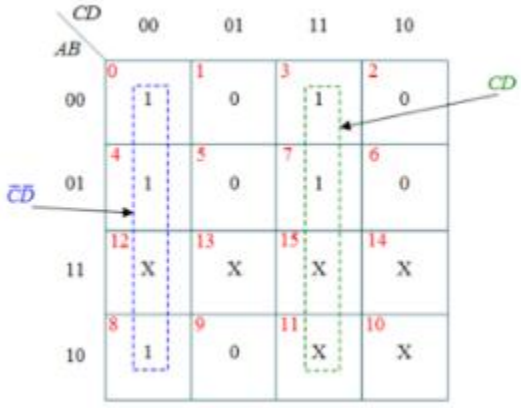
Kmap for w

W= A + BC + BD



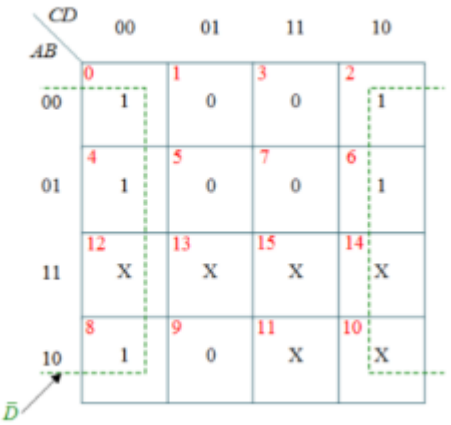
K map for x

X= B'C + B'D + VC'D'



K map for Y

Y= CD + C'D'



K map for z

Z= D'

Logic circuit diagram based on the minimized expression for each output

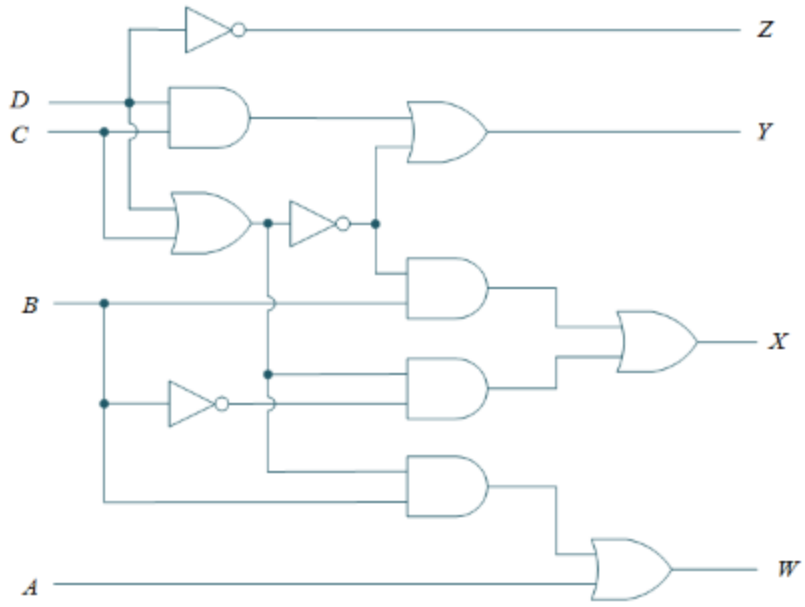
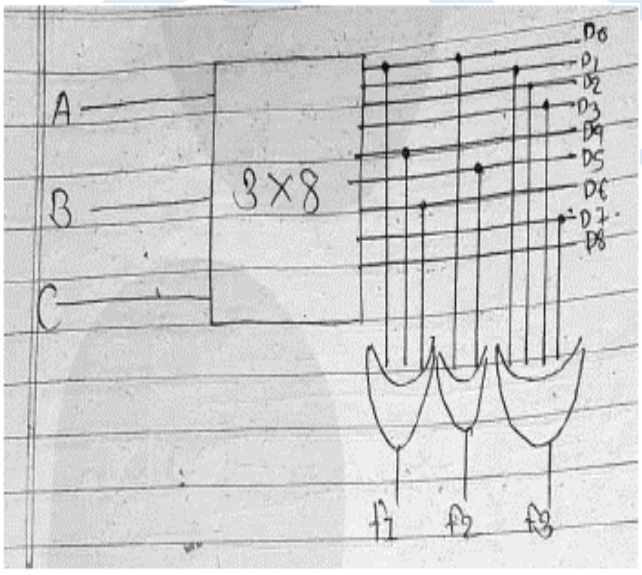


Figure 2: *BCD to Excess-3 Code Converter logic diagram.*

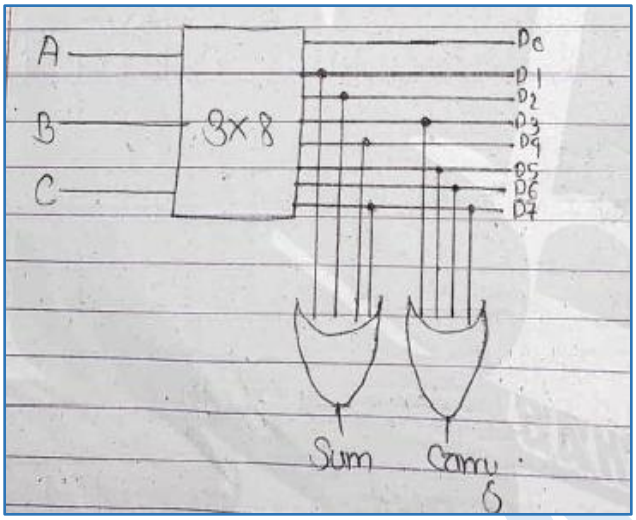
Implementation of Boolean function using Decoder

F= $\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + AB\bar{C}$ i.e F=$\sum(0,4,6)$



Implementation of full adder using Decoder

Boolean functions for output are:
S= x'y'z + x'yz' + xy'z' + xyz (i.e (1,2,4,7))
C= x'yz + xy'z + xyz' + xyz (i.e (3, 5, 6, 7))



Magnitude Comparator:

A magnitude digital Comparator is a combinational circuit that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition. This is useful if we want to compare two variables and want to produce an output when any of the above three conditions are achieved.

The circuit for comparing two n-bit numbers has 2^{2n} entries in the truth table and become too cumbersome even with n=3.

Applications of Comparators:

- ❒ Comparators are used in central processing units (CPUs) and microcontrollers (MCUs).
- ❒ These are used in control applications in which the binary numbers representing physical variables such as temperature, pressure,position, etc. are compared with a reference value.
- ❒ Comparators are also used as process controllers and for Servo motor control.
- ❒ Used in password verification and biometric applications.

1-Bit Magnitude Comparator:

A comparator used to compare two bits is called a single bit comparator. It consists of two inputs each for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers. The truth table for a 1-bit comparator is given below:

Inputs		Outputs		
A	B	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

From the above truth table logical expressions for each output can be expressed as follows:

A>B: AB'
A<B: A'B
A=B: A'B' + AB

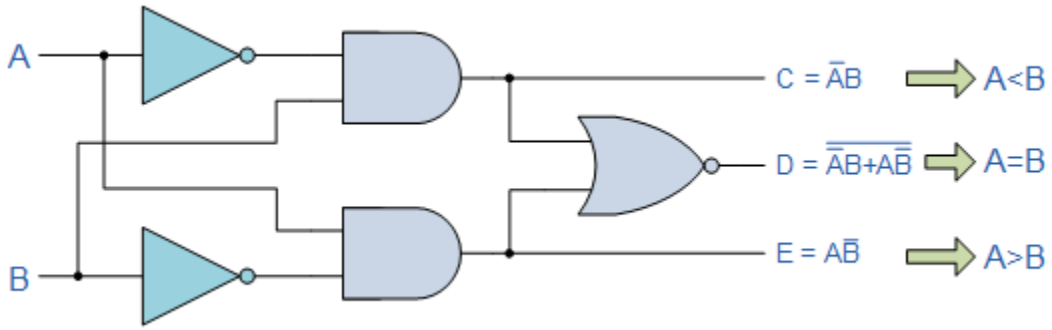


Fig: 1 bit digital comparator circuit

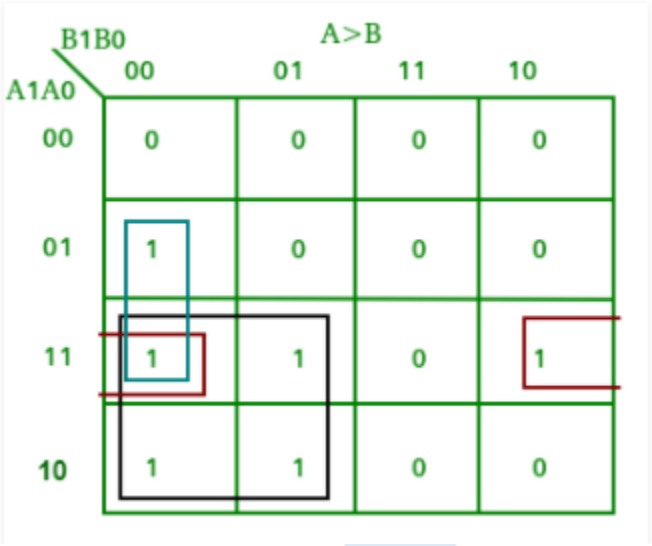
2-bit Magnitude Comparator

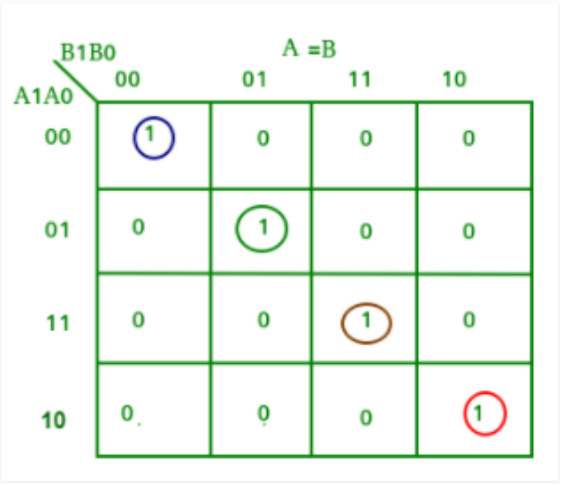
A comparator that compares two binary numbers (each number having 2 bits) and produces three outputs based on the relative magnitudes of given binary bits is called a 2-bit magnitude comparator.

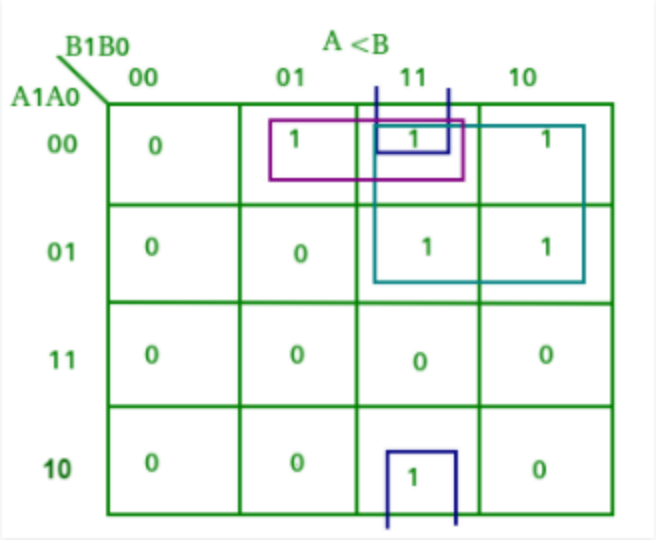
Truth Table

Inputs	Outputs

A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0







From the above K-maps logical expressions for each output can be expressed as follows:

A>B: A1B1’ + A0B1’B0’ + A1A0B0’

A=B: A1’A0’B1’B0’ + A1’A0B1’B0 + A1A0B1B0 + A1A0’B1B0’

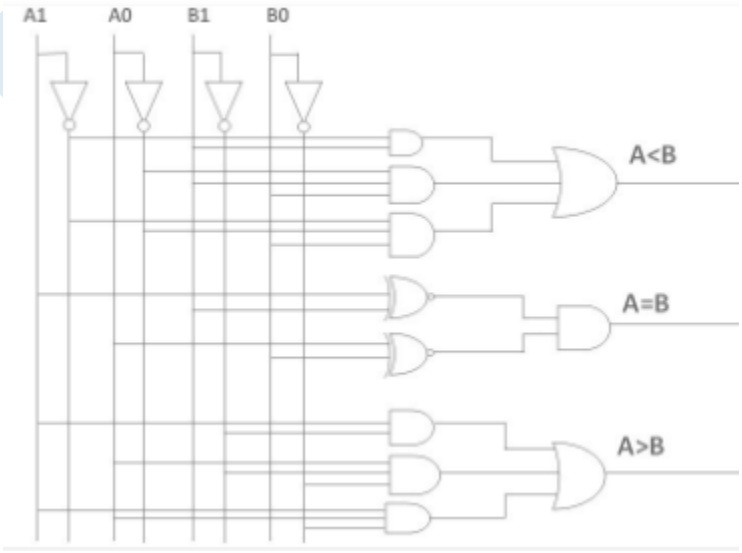
: A1’B1’ (A0’B0’ + A0B0) + A1B1 (A0B0 + A0’B0’)

: (A0B0 + A0’B0’) (A1B1 + A1’B1’)

: (A0 Ex-Nor B0) (A1 Ex-Nor B1)

A<B: A1’B1 + A0’B1B0 + A1’A0’B0

With these expressions, the Circuit diagram can be as follows



- End of Unit 4

Exercises:

1. What is combinational logic circuit? Write the steps for designing combinational logic circuit.
2. What is adder? Describe half adder with its truth table and logical diagram.
3. What is full adder? Explain
4. Implement a full adder with two half adder and one OR gate.
5. Describe different types of subtractor with block diagram, truth table and logic diagram.
6. Describe 4 bit parallel binary adder with diagram.
7. What is multiplexer? Design 4×1 mux.
8. Design 8×1 mux.
9. What is demux? Design circuit diagram of 1×8 demux.
10. How to implement 1×16 De-mux using one 1×2 de-mux and two 1×8 De-mux.
11. What is decoder? Design 2 to 4 decoder.
12. Design Octal to binary encoder.
13. What is seven segment display decoder? Design and draw a logic circuit diagram of seven segment display decoder.
14. What is comparator? Design 2 bit magnitude comparator.
15. Design a logic circuit that has three inputs, A, B, and C, and whose output will be HIGH only when a majority of the inputs are HIGH.