Combinational circuits

Combinational circuits are very well-known components in digital electronics which can provide output instantly based on the current input. Unlike sequential circuits, a combinational circuit listens for input signal and generates output no matter what is the past input or state as it has no feedback or memory component. It only cares about present input and state.

They are specially designed using multiple interconnected logic gates such that the output will be generated by computing the logical combinations of the present input only. No clock pulse is present here. Moreover, no previously stored value or state is taken into consideration here. The output is independent of previous states.

Features of Combinational Circuit

- In this output depends only upon present input.
- Its Speed is fast.
- Easy designed.
- There is no feedback between input and output.
- It is time independent.
- Elementary building blocks are Logic gates.
- Used for both arithmetic and Boolean operations.
- Combinational circuits don't have the capability to store any state.

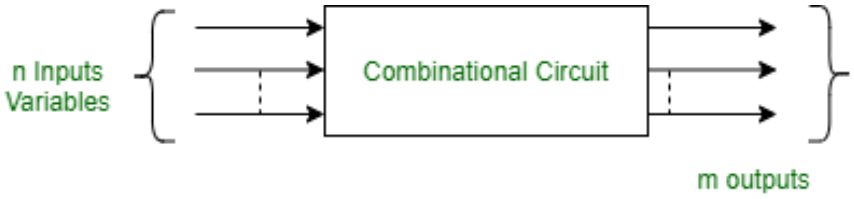


Figure - Block diagram of Combinational circuit

Examples of Combinational Circuits

- **Adders and Subtractors:** Combinational circuits are used to perform mathematical operations of binary numbers. **Adders and Subtractors** and Subtractor uses combinational circuit logic where in take two or more binary numbers as input and performs addition/subtraction to generate output.
- **Multiplexers and Demultiplexers:** Multiplexer is a special kind of combinational circuit where several numbers of inputs are present. From that one input can be selected and that will be

transmitted as an output based on selection signal. In the other hand, Demultiplexers also select one input signal but transmits it to one of the several outputs.

- **Encoders and Decoders:** Encoders also uses combinational circuits to convert a multiple set of input signal to smaller set of output signal but the resultant value is unchanged.
  Inversely, Decoders converts a small set of input signal to a large set of output signal without changing the resultant value. Basically, input of Encoder = output of Decoder and vice versa.

Applications of Combinational Circuit

In modern technologies, combinational circuits are widely used for its simple functionality and ability to give instant output. Some of the applications are discussed below:

- **Arithmetic and Logic Units (ALUs):** As combinational circuits are capable of performing various mathematical tasks like arithmetic and logical operations so, these circuits are used in calculators and processors as a fundamental component of **Arithmetic and Logic Units (ALUs)**.
- **Data Encryption and Decryption:** In information protection fields, combinational circuit are used for **Data Encryption** and decryption of data for secure communication. Encryption/decryption algorithms are also a complex mathematical formula which can be performed by combinational circuits.
- **Data Multiplexing and Demultiplexing:** It is just a practical implementation of multiplexer and demultiplexer. By using it we can optimize network bandwidth by effectively reducing network traffic as combinational circuit allows to transmit multiple data signals over a single communication channel.
- **Traffic Light Control:** In traffic lights control mechanism, combinational circuits are used to instantly determine the timing and sequence of traffic light changes based on the inputs of timers and sensors.

Advantages of Combinational Circuit

Some basic advantages of combinational circuits have made them very popular among modern technologies which are discussed below:

- **Simplicity:** Combinational circuits are very easy to implement and absence of memory or feedback element has made it more straightforward which is reducing complexity in digital systems and providing faster prototype building mechanism.

- **Real-time Operation:** Combinational circuits don't face any delay which is very much essential for quick-response applications like data transmission and signal processing.
- **Deterministic Behavior:** Combinational circuits generate output based on present input only which ensures predictable and repeatable results which is essential for applications in which consistency and reliability is required.

Disadvantages of Combinational Circuit

- **Limited Functionality:** As it provides output only based on present input and there is no way to store previous data so we can't use this circuit to any memory-based applications or where previous data is recalled for doing operations.
- **Lack of Flexibility:** Once the design of logic gates is done then for changing a small design required the entire redesigning of the circuits which is monotonous and time-consuming.
- **Increased Complexity for Large Designs:** For large designs the number of logic gates will increase gradually which means the management of input and output ports will become very complex. This may lead to higher production costs and increased design errors.

Adders and Subtractors in Digital Logic

Subtraction of two binary numbers can be accomplished by adding 2's complement of the subtrahend to the minuend and disregarding the final carry if any. If the MSB bit in the result of addition is a '0'. then the result of addition is the correct answer. If the MSB bit is a '1'. , this implies that the answer has a negative sign. The true magnitude, in this case, is given by 2's complement of the result of the addition.

Block Diagram of Combinational Logic Circuit:

Points to Remember on Combinational Logic Circuit:

1. Output depends upon the combination of inputs.

2. Output is a pure function of present inputs only i.e.; Previous State inputs won't have any effect on the output. Also, it doesn't use memory.
3. In other words,

OUTPUT=f(INPUT)

1. Inputs are called Excitation from circuits and outputs are called Responses of combinational logic circuits.

Classification of Combinational Logic Circuits:

1. Arithmetic:

- Adders
- Subtractors
- Multipliers
- Comparators

2. Data Handling:

- Multiplexers
- DeMultiplexers
- Encoders and Decoders

3. Code Converters:

- BCD to Excess-3 code and vice versa
- BCD to Gray code and vice versa
- Seven Segment

Design of Half Adders and Full Adders:

- A combinational logic circuit that performs the addition of two single bits is called Half Adder.
- A combinational logic circuit that performs the addition of three single bits is called Full Adder.

1. Half Adder:

- It is a arithmetic combinational logic circuit designed to perform addition of two single bits.
- It contains two inputs and produces two outputs.
- Inputs are called Augend and Added bits and Outputs are called Sum and Carry.

Let us observe the **addition of single bits**,

0+0=0

0+1=1

1+0=1

1+1=10

Since 1+1=10, the result must be two-bit output. So, Above can be rewritten as,

0+0=00

0+1=01

1+0=01

1+1=10

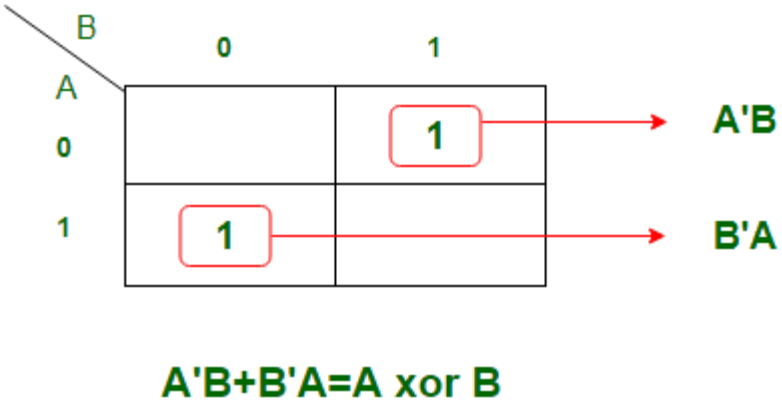
The result of 1+1 is 10, where '1' is carry-output (C_{out}) and '0' is Sum-output (Normal Output).

Truth Table of Half Adder:

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

Next Step is to draw the Logic Diagram. To draw Logic Diagram, we need Boolean Expression, which can be obtained using K-map (Karnaugh map). Since there are two output variables 'S' and 'C', we need to define K-map for each output variable.

K-map for output variable Sum 'S':



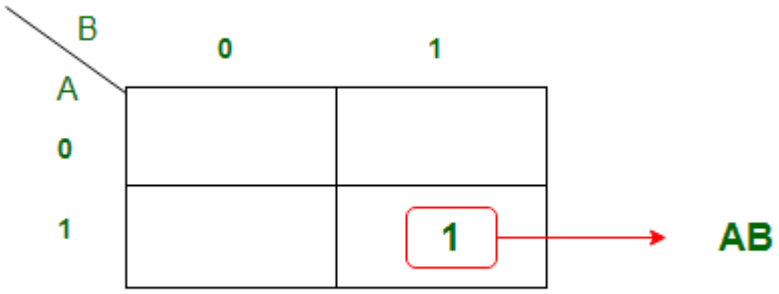
K-map is of **Sum of products** form. The equation obtained is

S = AB' + A'B

which can be logically written as,

S = A xor B

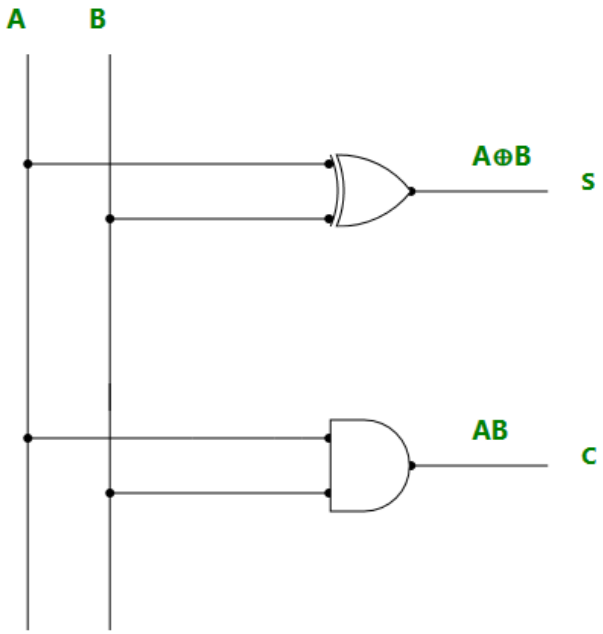
K-map for output variable Carry 'C':



The equation obtained from K-map is,

C = AB

Using the Boolean Expression, we can draw logic diagram as follows.



Limitations: Adding of Carry is not possible in Half adder.

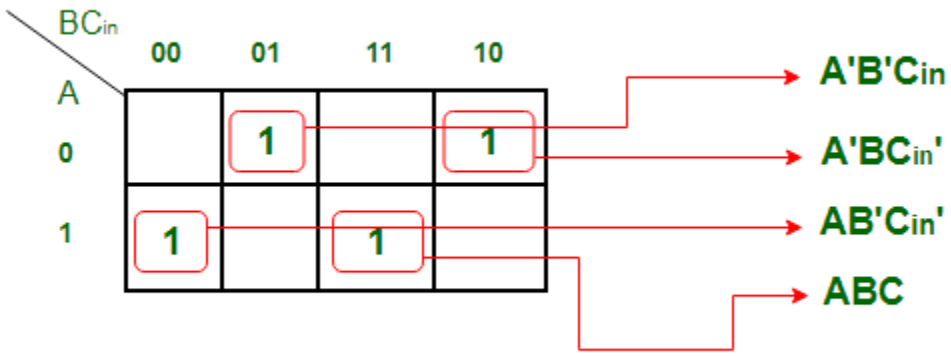
2. Full Adder:

- To overcome the above limitation faced with Half adders, Full Adders are implemented.
- It is a arithmetic combinational logic circuit that performs addition of three single bits.
- It contains three inputs (A, B, C_{in}) and produces two outputs (Sum and C_{out}).
- Where, C_{in} -> Carry In and C_{out} -> Carry Out

Truth table of Full Adder:

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

K-map Simplification for output variable Sum 'S':



The equation obtained is,

$S = A'B'C_{in} + AB'C_{in}' + ABC + A'BC_{in}'$

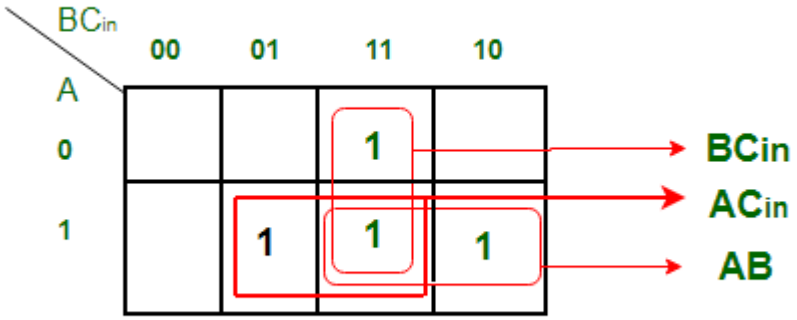
The equation can be simplified as,

$S = B'(A'C_{in}+AC_{in}') + B\ (AC + A'C_{in}')$

$S = B'(A \text{ xor } C_{in}) + B\ (A \text{ xor } C_{in})'$

$S = A \text{ xor } B \text{ xor } C_{in}$

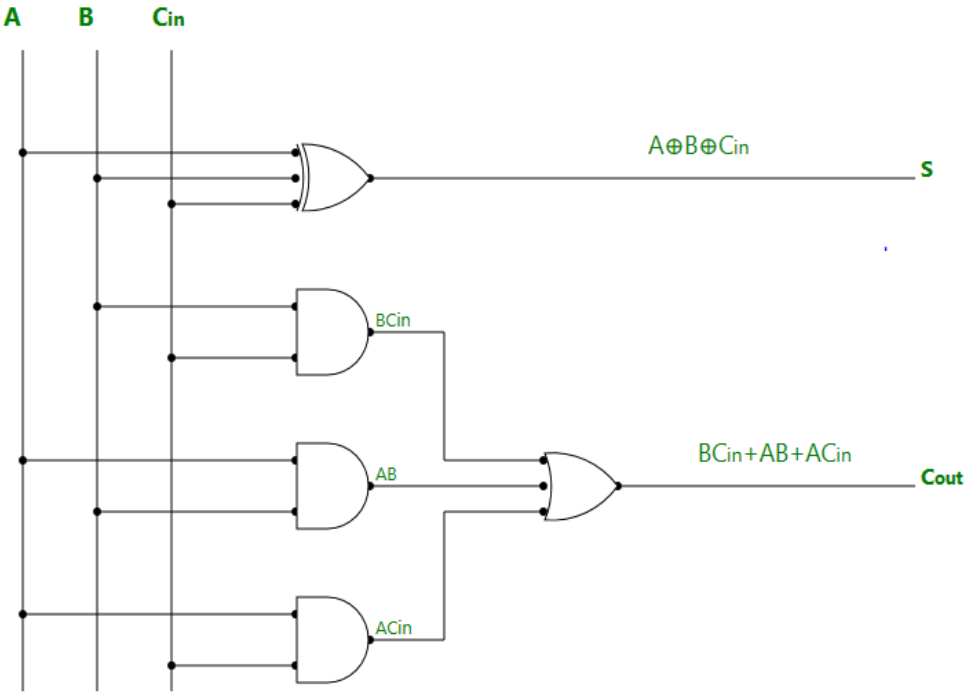
K-map Simplification for output variable 'C_{out}'



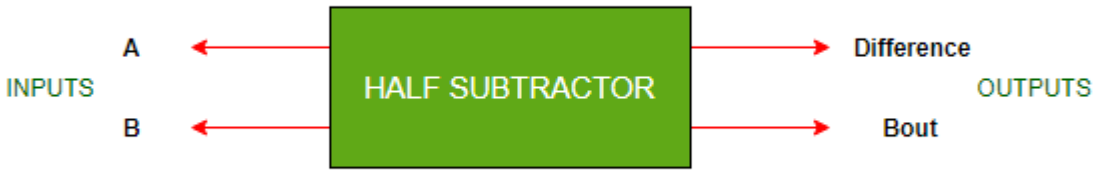
The equation obtained is,

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

Logic Diagram of Full Adder:



3. Half Subtractor:

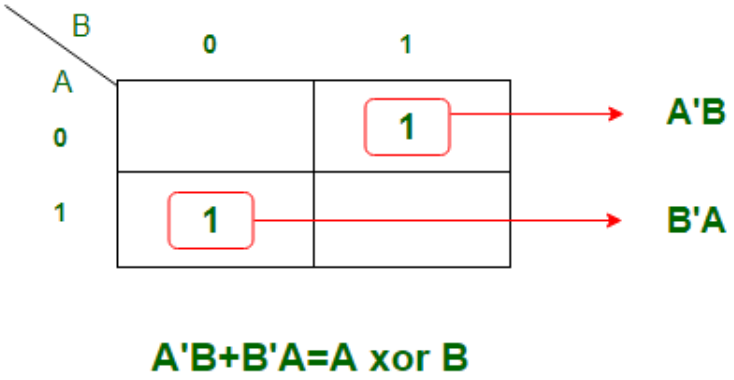


- It is a combinational logic circuit designed to perform the subtraction of two single bits.
- It contains two inputs (A and B) and produces two outputs (Difference and Borrow-output).

Truth Table of Half Subtractor:

Inputs		Outputs	
A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-map Simplification for output variable 'D':



A'B+B'A=A xor B

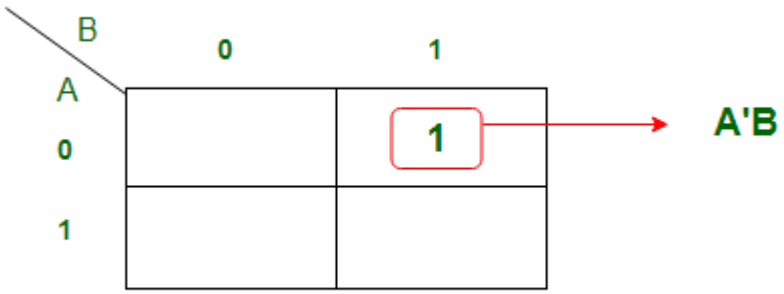
The equation obtained is,

D = A'B + AB'

which can be logically written as,

D = A xor B

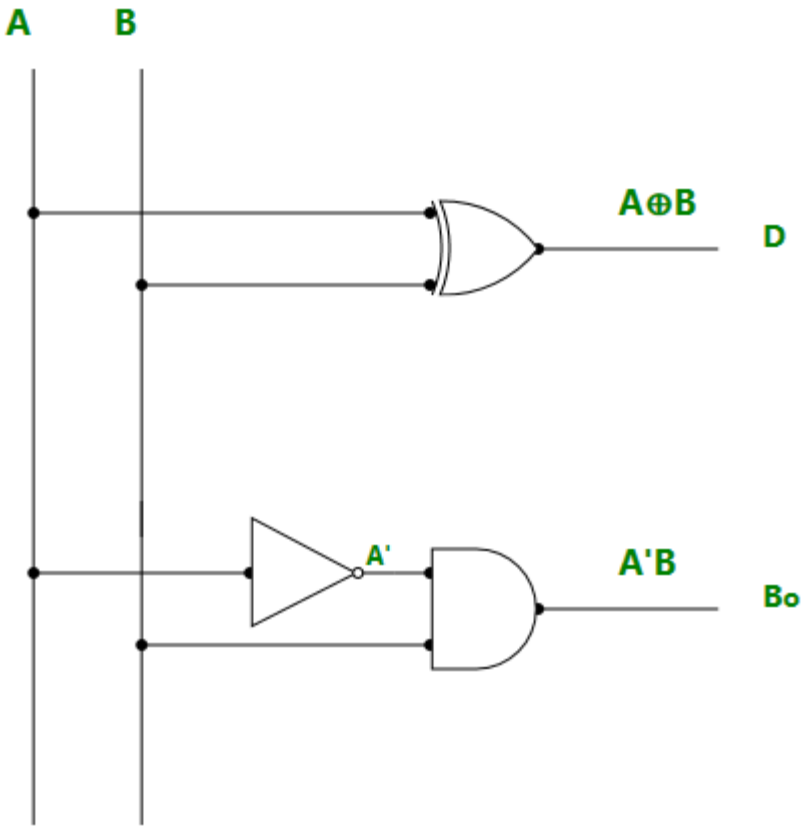
K-map Simplification for output variable 'B_{out}':



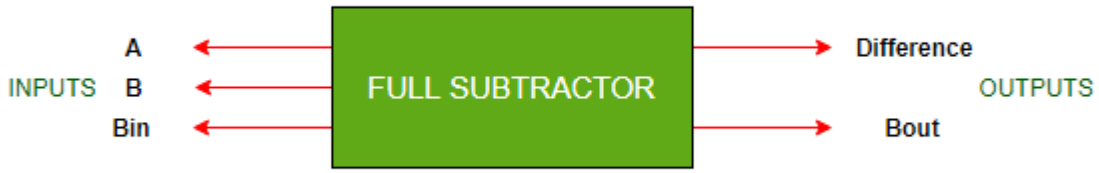
The equation obtained from above K-map is,

B_{out} = A'B

Logic Diagram of Half Subtractor:



4. Full Subtractor:

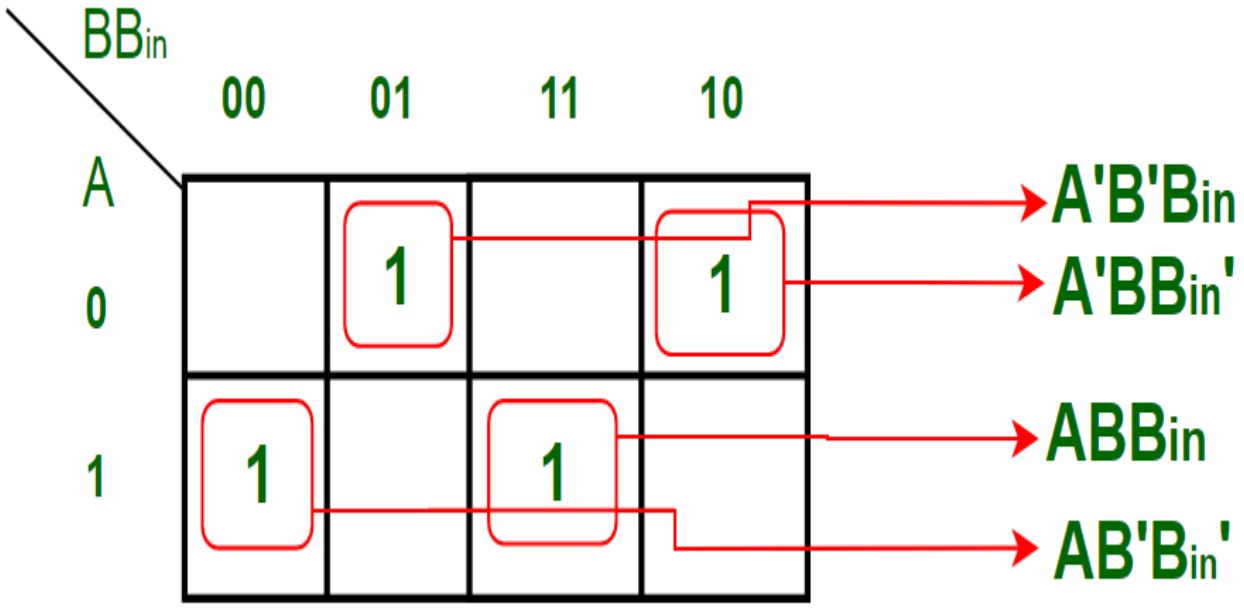


- It is a Combinational logic circuit designed to perform subtraction of three single bits.
- It contains three inputs (A, B, B_{in}) and produces two outputs (D, B_{out}).
- Where, A and B are called **Minuend** and **Subtrahend** bits.
- And, B_{in} -> Borrow-In and B_{out} -> Borrow-Out

Truth Table of Full Subtractor:

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
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

K-map Simplification for output variable 'D' :



The equation obtained from above K-map is,

$D = A'B'B_{in} + AB'B_{in}' + ABB_{in} + A'BB_{in}'$

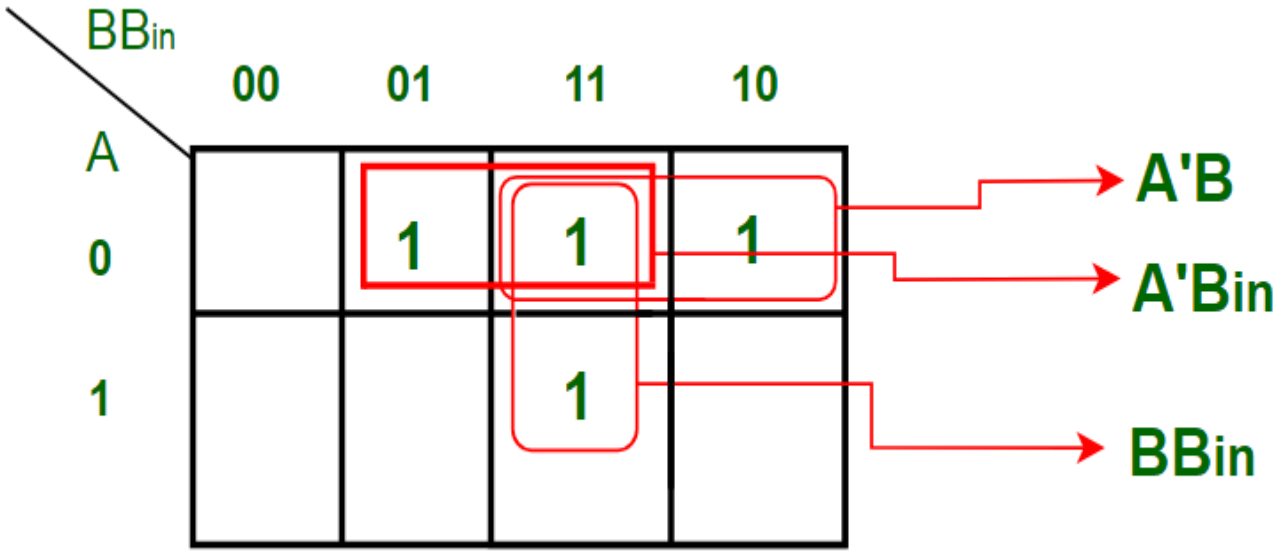
which can be simplified as,

$D = B'(A'B_{in} + AB_{in}') + B(AB_{in} + A'B_{in}')$

$D = B'(A \text{ xor } B_{in}) + B(A \text{ xor } B_{in})'$

$D = A \text{ xor } B \text{ xor } B_{in}$

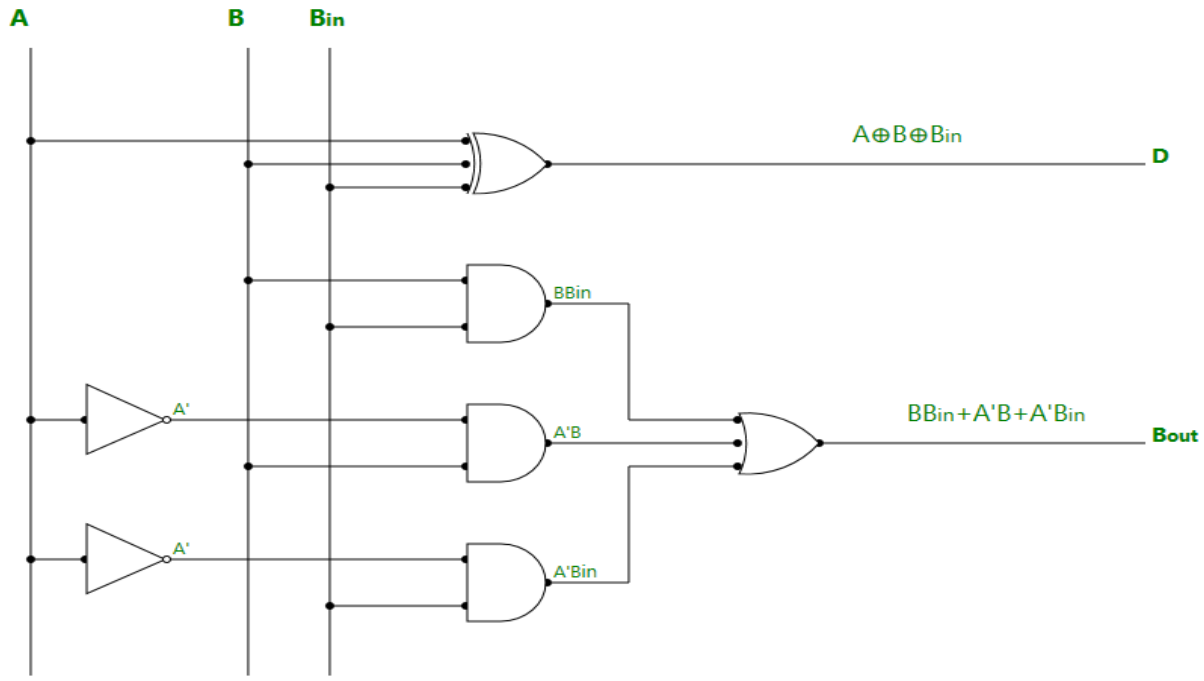
K-map Simplification for output variable 'B_{out}':



The equation obtained is,

$B_{out} = BB_{in} + A'B + A'B_{in}$

Logic Diagram of Full Subtractor:



Applications:

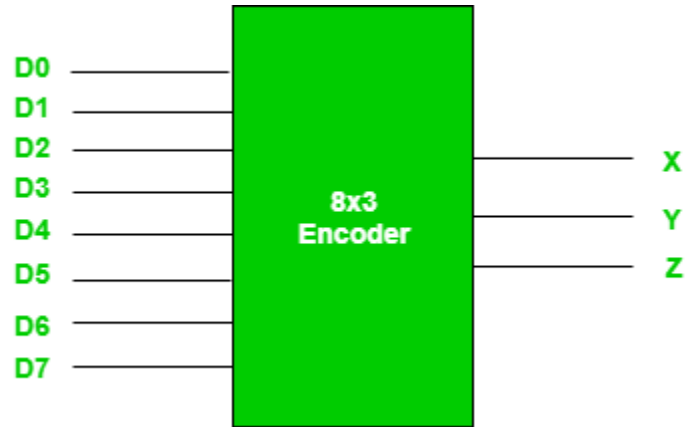
1. For performing arithmetic calculations in electronic calculators and other digital devices.
2. In Timers and Program Counters.
3. Useful in Digital Signal Processing.

Encoders and Decoders in Digital Logic

Binary code of N digits can be used to store 2N distinct elements of coded information. This is what encoders and decoders are used for.

Encoders convert 2N lines of input into a code of N bits and **Decoders** decode the N bits into 2N lines.

1. Encoders - An encoder is a combinational circuit that converts binary information in the form of a 2N input lines into N output lines, which represent N bit code for the input. For simple encoders, it is assumed that only one input line is active at a time. As an example, let's consider **Octal to Binary** encoder. As shown in the following figure, an octal-to-binary encoder takes 8 input lines and generates 3 output lines.



Truth Table -

D7	D6	D5	D4	D3	D2	D1	D0	X	Y	Z
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

As seen from the truth table, the output is 000 when D0 is active; 001 when D1 is active; 010 when D2 is active and so on.

Implementation

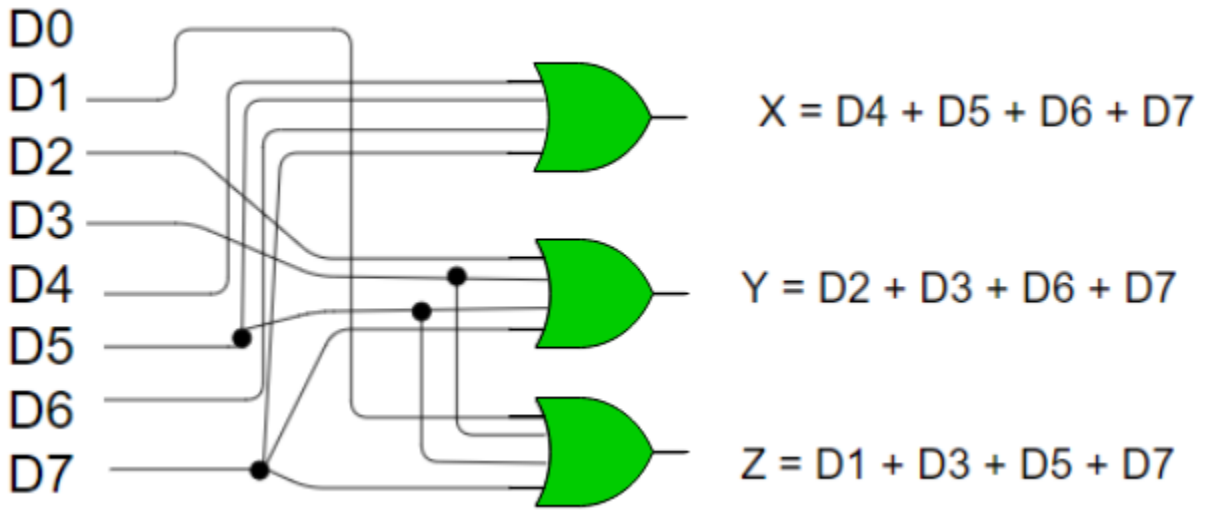
From the truth table, the output line Z is active when the input octal digit is 1, 3, 5 or 7. Similarly, Y is 1 when input octal digit is 2, 3, 6 or 7 and X is 1 for input octal digits 4, 5, 6 or 7. Hence, the Boolean functions would be:

X = D4 + D5 + D6 + D7

Y = D2 +D3 + D6 + D7

Z = D1 + D3 + D5 + D7

Hence, the encoder can be realized with OR gates as follows:



One limitation of this encoder is that only one input can be active at any given time. If more than one inputs are active, then the output is undefined. For example, if D6 and D3 are both active, then, our output would be 111 which is the output for D7. To overcome this, we use Priority Encoders. Another ambiguity arises when all inputs are 0. In this case, encoder outputs 000 which actually is the output for D0 active. In order to avoid this, an extra bit can be added to the output, called the valid bit which is 0 when all inputs are 0 and 1 otherwise.

Priority Encoder -

A priority encoder is an encoder circuit in which inputs are given priorities. When more than one inputs are active at the same time, the input with higher priority takes precedence and the output corresponding to that is generated. Let us consider the 4 to 2 priority encoders as an example. From the truth table, we see that when all inputs are 0, our V bit or the valid bit is zero and outputs are not used. The x's in the table show the don't care condition, i.e., it may either be 0 or 1. Here, D3 has highest priority, therefore,

whatever be the other inputs, when D3 is high, output has to be 11. And D0 has the lowest priority, therefore the output would be 00 only when D0 is high and the other input lines are low. Similarly, D2 has higher priority over D1 and D0 but lower than D3 therefore the output would be 010 only when D2 is high and D3 are low (D0 & D1 are don't care).

Truth Table -

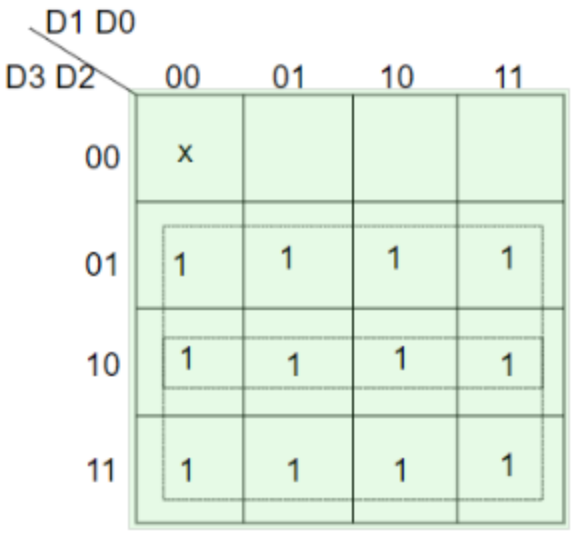
D3	D2	D1	D0	X	Y	V
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

Implementation -

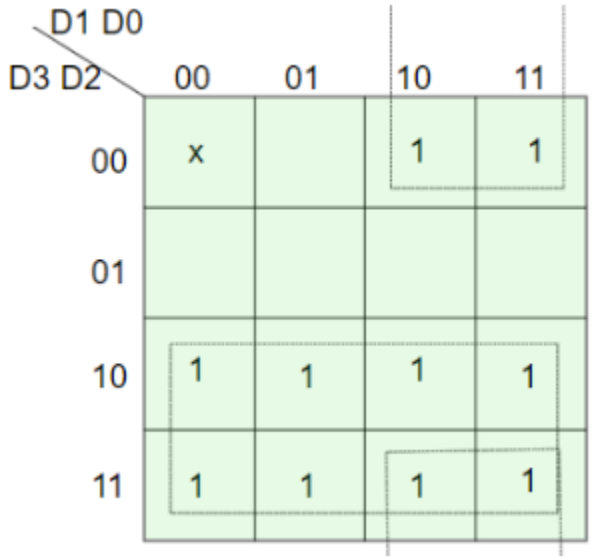
It can clearly be seen that the condition for valid bit to be 1 is that at least any one of the inputs should be high. Hence,

V = D0 + D1 + D2 + D3

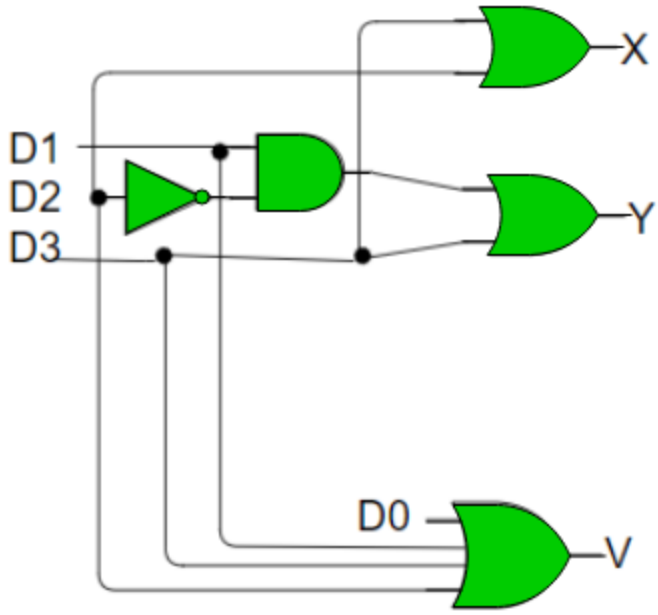
For X:



=> X = D2 + D3 For Y:



=> Y = D1 D2’ + D3 Hence, the priority 4-to-2 encoder can be realized as follows:



2. Decoders -

A decoder does the opposite job of an encoder. It is a combinational circuit that converts n lines of input into 2

n

lines of output. Let's take an example of 3-to-8-line decoder.

Truth Table -

X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0

X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Implementation -

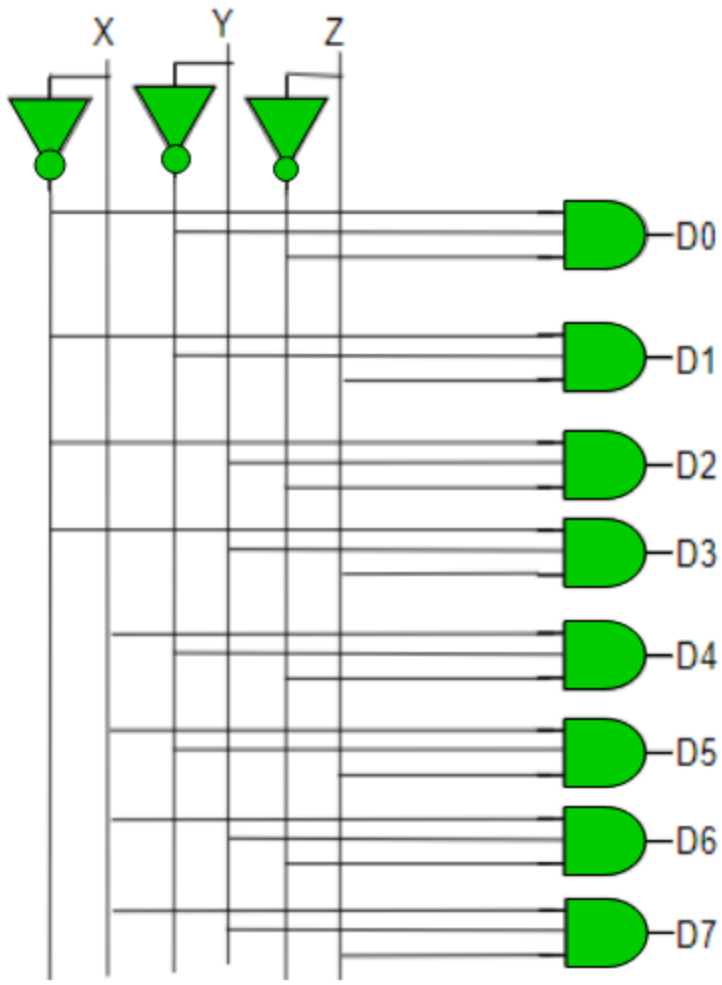
D0 is high when X = 0, Y = 0 and Z = 0. Hence,

D0 = X' Y' Z'

Similarly,

D1 = X' Y' Z
D2 = X' Y Z'
D3 = X' Y Z
D4 = X Y' Z'
D5 = X Y' Z
D6 = X Y Z'
D7 = X Y Z

Hence,



Multiplexer and Demultiplexer

A Multiplexer (MUX) and a Demultiplexer (DEMUX) are essential digital circuits in communication systems, performing opposite functions. A multiplexer combines multiple input signals into a single output, while a demultiplexer takes a single input signal and routes it to one of many output lines.

What is a Multiplexer?

A **multiplexer** is a combinational circuit with multiple data inputs and a single output, determined by control or select lines. Often referred to as **MUX**, it requires log⁡2(N)log2(*N*) selection lines for N*N* input lines, or equivalently, *n* selection lines for 2 power n input lines.

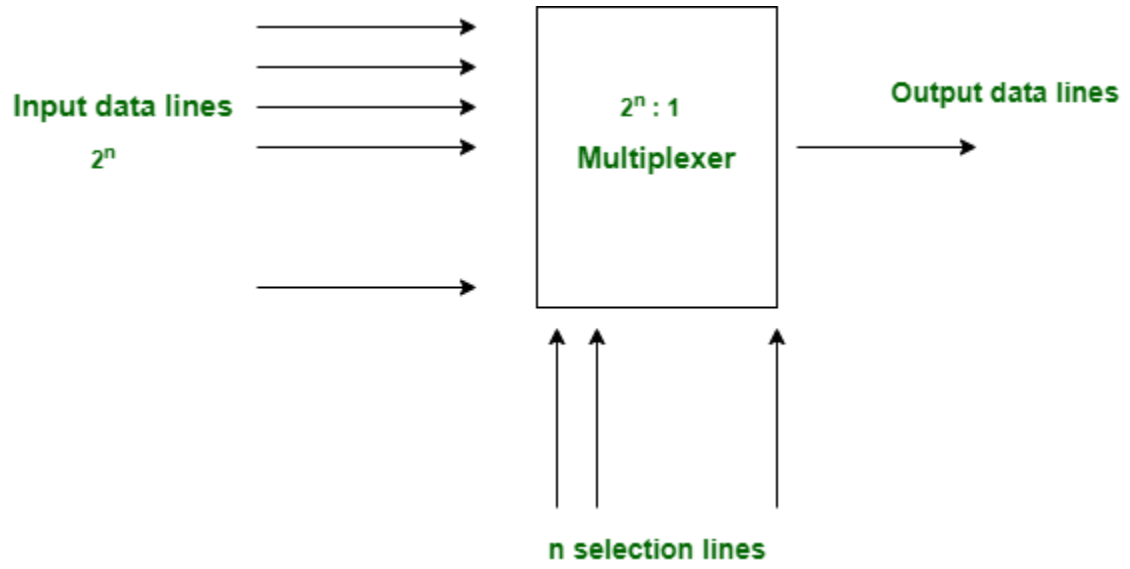
Multiplexers are also known as:

- N-to-1 selectors
- Parallel-to-serial converters

- Many-to-one circuits
- Universal logic circuits

They are mainly used to increase the amount of data that can be sent over a network within a certain amount of time and bandwidth

Below is the Block Diagram of the Multiplexer, It will have 2n Input lines and will select output based on the Select line.



Advantages of Multiplexer(MUX)

- **Reduces the number of data lines**: MUX allows multiple signals to share a single communication line, saving space and resources.
- **Simpler Design**: It simplifies the system by reducing the number of data channels needed.
- **Efficient Use of Resources**: MUX optimizes the use of communication channels or buses.
- **Flexible Design**: MUX can be used for different types of data and communication formats.
- **Compact Systems**: It helps make systems smaller and simpler by reducing the number of connections.
- **Reduces Errors**: Fewer data lines mean less chance of interference or noise.

Disadvantages of Multiplexer (MUX)

- **Complex Control**: It needs extra circuits to select which input is sent, making the design more complicated.
- **Limited Inputs**: The number of inputs depends on the number of control lines (e.g., 2 control lines allow only 4 inputs).
- **Propagation Delay**: Switching between inputs can introduce delays in high-speed systems.
- **Higher Power Use**: Larger multiplexers with more inputs can consume more power.

- **Signal Loss**: Some signal degradation may occur when combining multiple signals.

Multiplexer

Types of Mux

The Mux can be of different types based on input but in this article, we will go through two major types of mux which are

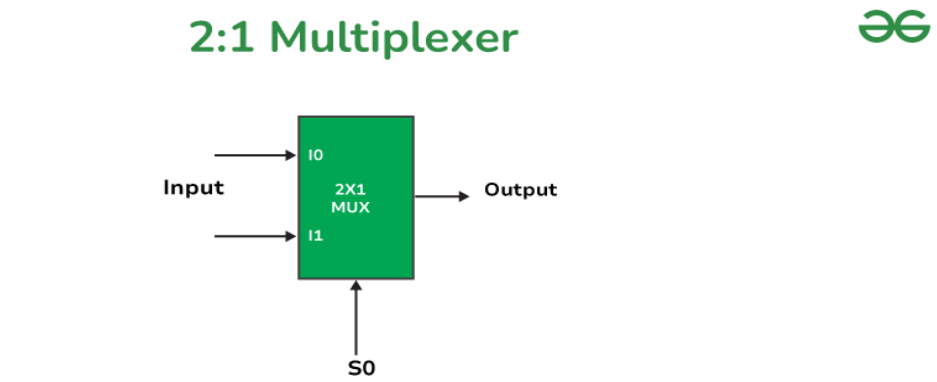
- **2x1 Mux**
- **4x1 Mux**

2x1 Multiplexer

The 2x1 is a fundamental circuit which is also known 2-to-1 multiplexer that are used to choose one signal from two inputs and transmits it to the output. The 2x1 mux has two input lines, one output line, and a single selection line. It has various applications in digital systems such as in microprocessor it is used to select between two different data sources or between two different instructions.

Block Diagram of 2:1 Multiplexer with Truth Table

Given Below is the Block Diagram and Truth Table of 2:1 Mux. In this Block Diagram where I0 and I1 are the input lines, Y is the output line and S0 is a single select line.



Truth Table

S_0	I_0	I_1	Y
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

Block Diagram of 2:1 Multiplexer with Truth Table

The output of the 2x1 Mux will depend on the selection line S0,

- When S is 0(low), the I0 is selected
- when S0 is 1(High), I1 is selected

Logical Expression of 2x1 Mux

Using the Truth Table ,the Logical Expression for Mux can be determined as

$Y=S0\bar{}.I0+S0.I1Y=S0.I0+S0.I1$

Circuit Diagram of 2x1 Multiplexers

Using truth table, the circuit diagram can be given as

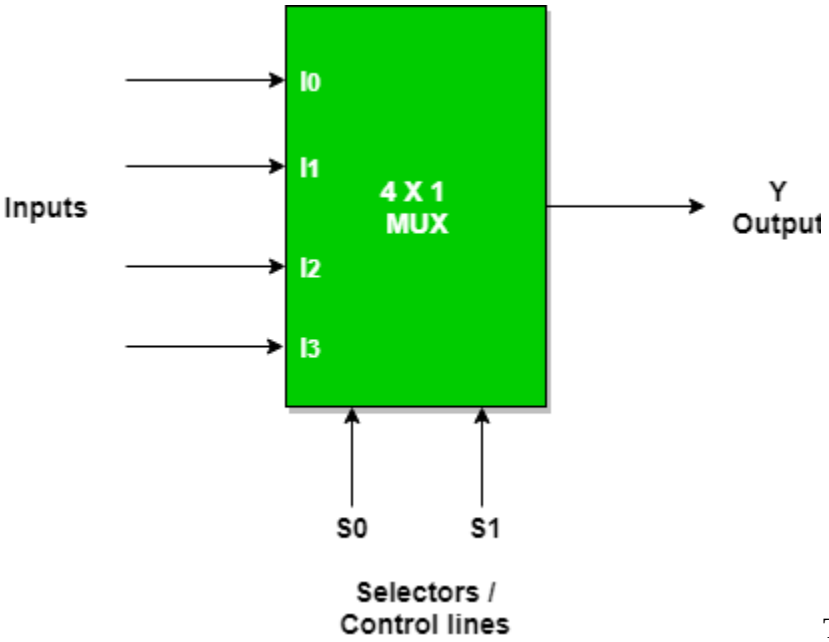
Circuit Diagram of 2x1 Mux

4×1 Multiplexer

The 4x1 Multiplexer which is also known as the 4-to-1 multiplexer. It is a multiplexer that has 4 inputs and a single output. The Output is selected as one of the 4 inputs which is based on the selection inputs. The number of the Selection lines will depend on the number of the input which is determined by the equation log2n*log*2*n*, In 4x1 Mux the selection lines can be determined as log4=2*log*4=2 , two selections are needed.

Block Diagram of 4×1 Multiplexer

In the Given Block Diagram I0, I1, I2, and I3 are the 4 inputs and Y is the Single output which is based on Select lines S0 and S1.



The output of the multiplexer is determined by the binary value of the selection lines

- When S1S0=00, the input I0 is selected.
- When S1S0=01, the input I1 is selected.

- When S1S0=10, the input I2 is selected.
- When S1S0=11, the input I3 is selected.

Truth Table of 4×1 Multiplexer

Given Below is the Truth Table of 4x1 Multiplexer

Truth Table

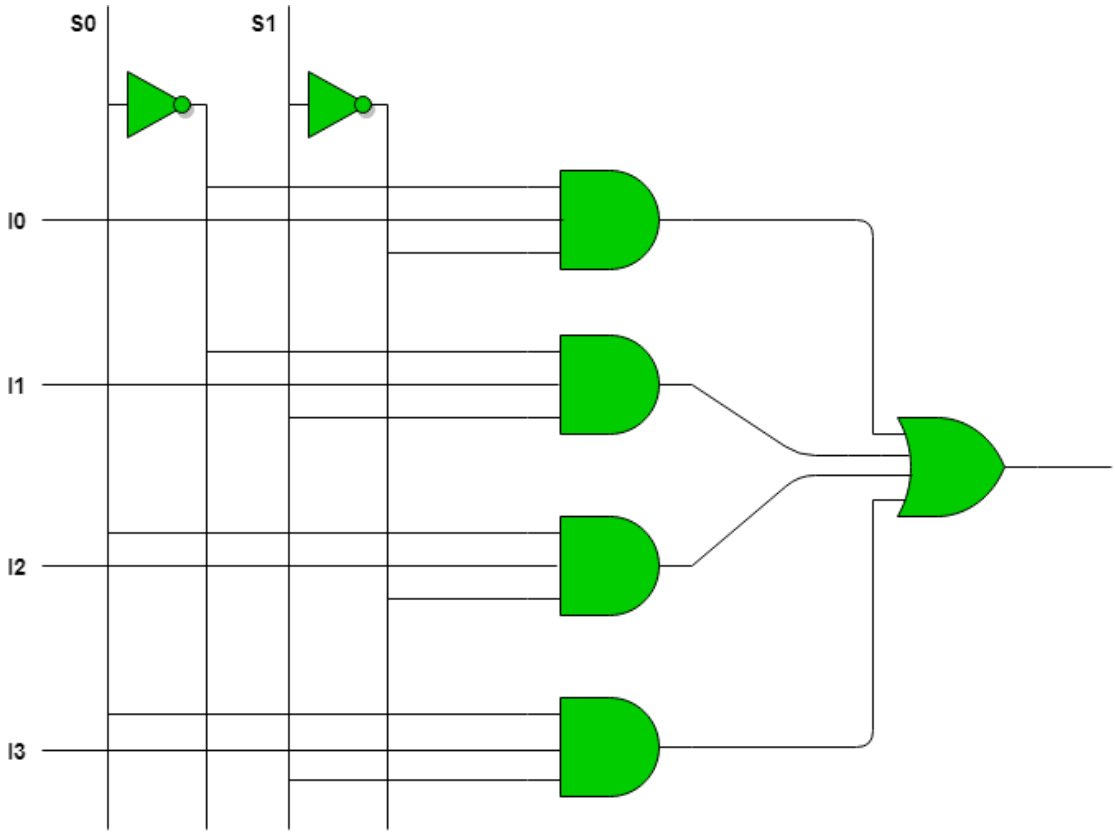
S0	S1	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

So, final equation,

$Y = S0'.S1'.I0 + S0'.S1.I1 + S0.S1'.I2 + S0.S1.I3$

Circuit Diagram of 4x1 Multiplexers

Using truth table, the circuit diagram can be given as



Multiplexer can act as universal combinational circuit. All the standard logic gates can be implemented with multiplexers.

Implementation of Different Gates with 2:1 Mux

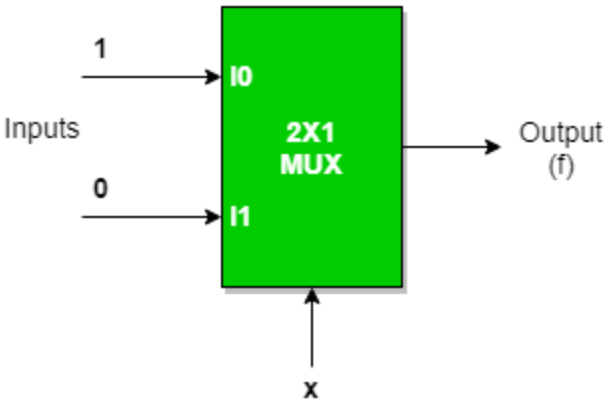
Given below are the Implementation of Different gate using 2:1 Mux

Implementation of NOT gate using 2: 1 Mux

The Not gate from 2:1 Mux can be obtained by

- Connect the input signal to one of the data input lines(I0).
- Then connect a line (0 or 1) to the other data input line(I1)
- Connect the same input line Select line S0 which is connected to D0.

Given Below is the Diagram for the Logical Representation of **NOT gate using 2: 1 Mux**



Truth Table

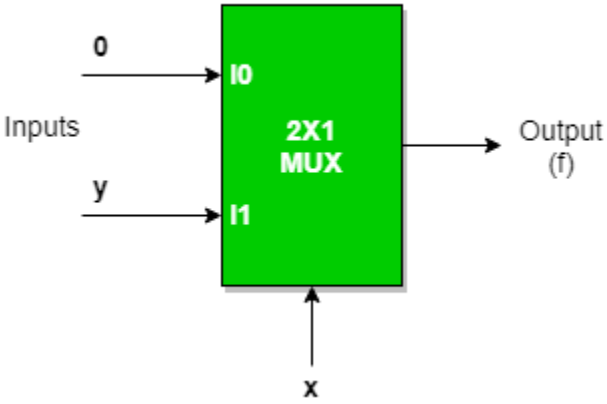
x	f
0	1
1	0

Implementation of AND gate using 2 : 1 Mux

The And gate from 2:1 Mux can be obtained by

- Connect the input Y to I1.
- Connect the input X to the selection line S0.
- Connect a line (0) to I0.

Given Below is the Diagram for the Logical Representation of **AND gate using 2 : 1 Mux**



Truth Table

x	y	f	f→y
0	0	0	f = 0
0	1	0	
1	0	0	f = y
1	1	1	

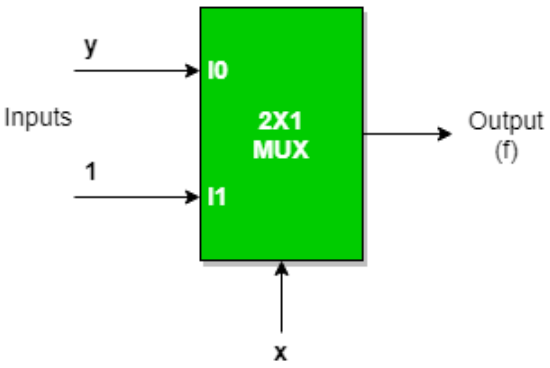
For further more on the **Implementation of AND gate using 2: 1 Mux**

Implementation of OR gate using 2: 1 Mux

The OR gate from 2:1 Mux can be obtained by

- Connect input X to the selection line S0.
- Connect input Y to I1.
- Connect Line(1) to I1.

Given Below is the Diagram for the Logical Representation of **OR gate using 2 : 1 Mux**



Truth Table

x	y	f	f→y
0	0	0	f = y
0	1	1	
1	0	1	f = 1
1	1	1	

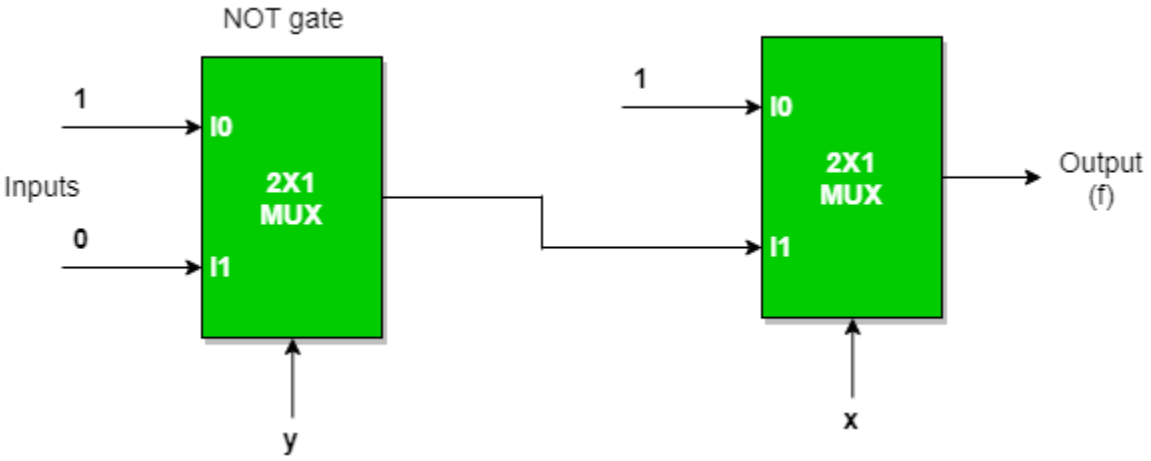
Implementation of NAND, NOR, XOR and XNOR gates requires two 2:1 Mux. First multiplexer will act as NOT gate which will provide complemented input to the second multiplexer.

Implementation of NAND gate using 2: 1 Mux

The NAND gate from 2:1 Mux can be obtained by

- In first mux take inputs and 1 and 0 and y as selection line.
- In Second MUX the Output from mux is connected to I1.
- line (1) is given to the I0.
- x is given as selection line for the second Mux.

Given Below is the Diagram for the Logical Representation of **NAND gate using 2 : 1 Mux**



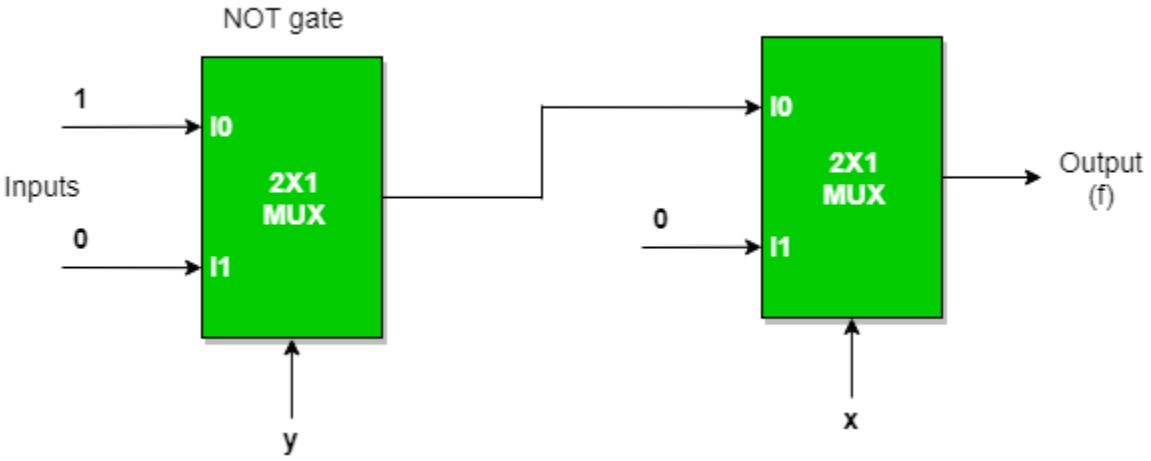
For further more on the **Implementation of NAND gate using 2 : 1 Mux**

Implementation of NOR gate using 2 : 1 Mux

The Nor gate from 2:1 Mux can be obtained by

- In first mux take inputs and 1 and 0 and y as selection line.
- In Second MUX the Output from mux is connected to I0.
- line(0) is given to the I1.
- x is given as selection line for the second Mux.

Given Below is the Diagram for the Logical Representation of **NOR gate using 2 : 1 Mux**



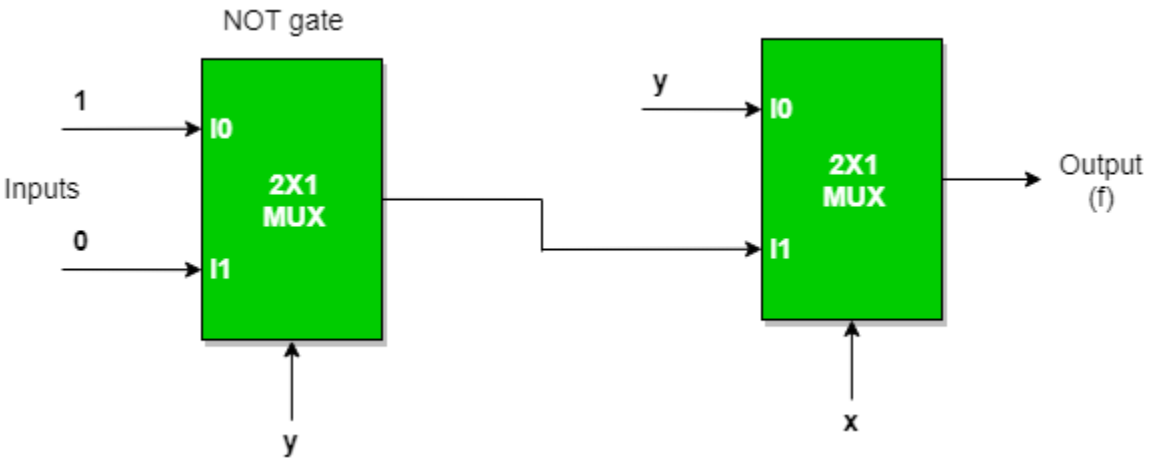
For further more on the **Implementation of NOR gate using 2 : 1 Mux**

Implementation of EX-OR gate using 2 : 1 Mux

The Nor gate from 2:1 Mux can be obtained by

- In first mux take inputs and 1 and 0 and y as selection line.
- In Second MUX the Output from mux is connected to I1.
- y is given to the I0.
- x is given as selection line for the second Mux.

Given Below is the Diagram for the Logical Representation of **EX-OR gate using 2 : 1 Mux**

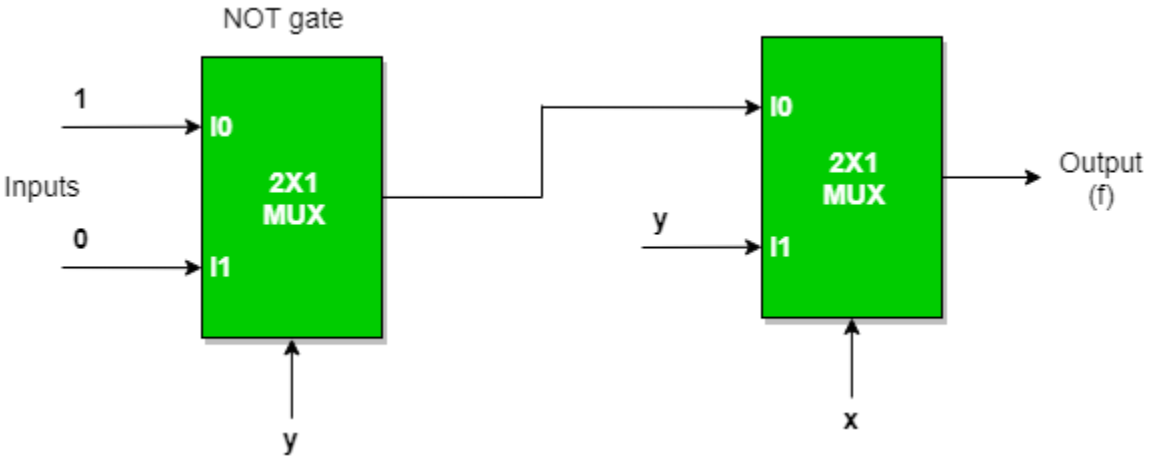


Implementation of EX-NOR gate using 2: 1 Mux

Given Below is the Diagram for the Logical Representation of **EX-OR gate using 2: 1 Mux**

The Nor gate from 2:1 Mux can be obtained by

- In first mux take inputs and 1 and 0 and y as selection line.
- In Second MUX the Output from mux is connected to I0.
- y is given to the I1.
- x is given as selection line for the second Mux.



Implementation of Higher Order MUX using Lower Order MUX

Given Below are the Implementation of Higher Order MUX Using Lower Order MUX

4: 1 MUX using 2 : 1 MUX

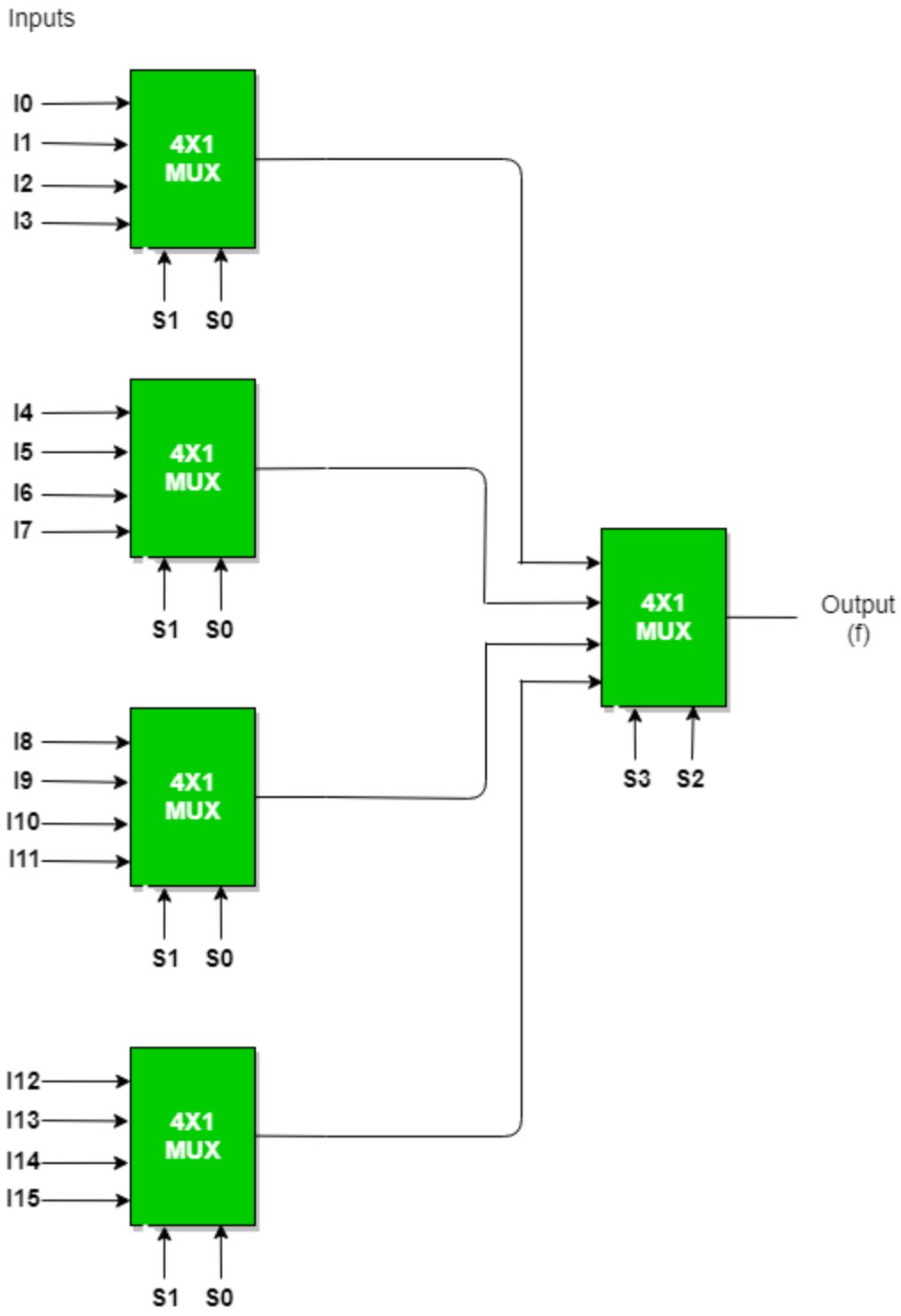
Three 2: 1 MUX are required to implement 4 : 1 MUX.4 : 1 MUX using 2 : 1 MUX

Similarly,

While an 8:1 MUX requires seven (7) 2:1 MUX, a 16:1 MUX requires fifteen (15) 2:1 MUX, and a 64:1 MUX requires sixty-three (63) 2:1 MUX. Hence, we can draw the conclusion that an 2n:12*n*:1 MUX requires sixty-three (63) 2:1 MUX. Hence, we can draw the conclusion that 2 *n* :1 MUX requires (2n−1)2:1 MUX (2 *n* −1)2:1 *MUX*.

16: 1 MUX using 4 : 1 MUX

Given Below is the logical Diagram of 16:1 Mux Using 4:1 Mux



In general, to implement B: 1 MUX using A: 1 MUX, one formula is used to implement the same.
B / A = K1,
K1/ A = K2,
K2/ A = K3

KN-1 / A = KN = 1 (till we obtain 1 count of MUX).

And then add all the numbers of MUXes = K1 + K2 + K3 + ... + K N .
To implement 64 : 1 MUX using 4 : 1 MUX
Using the above formula, we can obtain the same.
64 / 4 = 16
16 / 4 = 4
4 / 4 = 1 (till we obtain 1 count of MUX)
Hence, total number of 4 : 1 MUX are required to implement 64 : 1 MUX = 16 + 4 + 1 = 21.

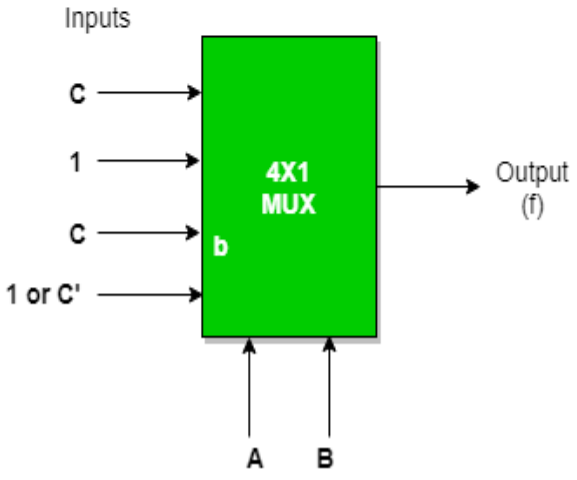
f(A, B, C) = ∑∑ (1, 2, 3, 5, 6) with don't care (7)

Using A and B as the select lines for 4: 1 MUX,

AB as select: Expanding the minterms to its boolean form and will see its 0 or 1 value in Cth place so that they can be placed in that manner.

A	B	C	f
0	0	0	C′
0	0	1	C
0	1	0	C′
0	1	1	C
1	0	0	C′
1	0	1	C
1	1	0	C′
1	1	1	C

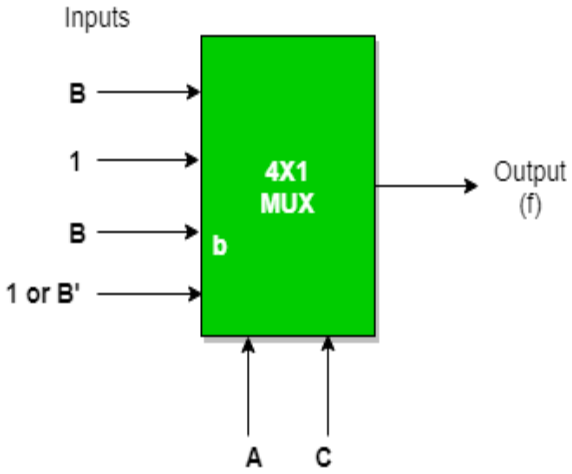
	I0	I1	I2	I3
C′	0	2	4	6
C	1	3	5	7 is don't care (can consider or not)
	C	1	C	1 (in case 7 is considered or C′)



AC as select: Expanding the midterms to its Boolean form and will see its 0 or 1 value in Both place so that they can be place in that manner.

A	B	C	f
0	0	0	B′
0	0	1	B′
0	1	0	B
0	1	1	B
1	0	0	B′
1	0	1	B′
1	1	0	B
1	1	1	B

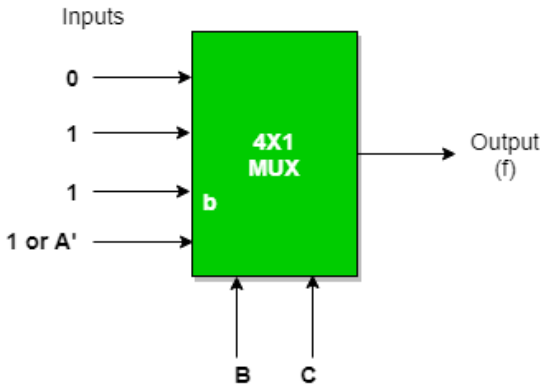
	I0	I1	I2	I3
B′	0	1	4	5
B	2	3	6	7 is don't care (can consider or not)
	B	1	B	1 (in case 7 is considered) or B′



BC as select: Expanding the midterms to its Boolean form and will see its 0 or 1 value in A place so that they can be place in that manner.

A	B	C	f
0	0	0	A’
0	0	1	A’
0	1	0	A’
0	1	1	A’
1	0	0	A
1	0	1	A
1	1	0	A
1	1	1	A

	I0	I1	I2	I3
A’	0	1	2	3
A	4	5	6	7 is don’t care (can considered or not)
	0	1	1	1 (in case 7 is considered) or A’



Advantages of MUX

- **Efficiency**: The Mux has good efficiency in routing multiple input signals to a single out signal based on control signals.
- **Optimization**: The Mux helps to conserve resources such as wires, pins and integrated circuit (IC).
- **Different Implementation:** The Mux can be used to implement different digital logic functions such AND, OR etc.
- **Flexibility:** Mux can be easily configuring according to the requirements and accommodate different data sources, enhancing system versatility.

Disadvantages of MUX

- **Limited number of data sources:** The number of inputs that can be taken by a multiplexer is restricted by the number of control lines, which can cause limitations in certain applications.
- **Delay:** Multiplexers can have some delay in the signal path, which can have impact on the performance of the circuit.
- **Complex control rationale:** The control logic for multiplexers can be complex, particularly for bigger multiplexers with a large number of inputs.
- **Power utilization:** Multiplexers can consume more power compared with other simple logic gate, particularly when they have a large number of inputs.

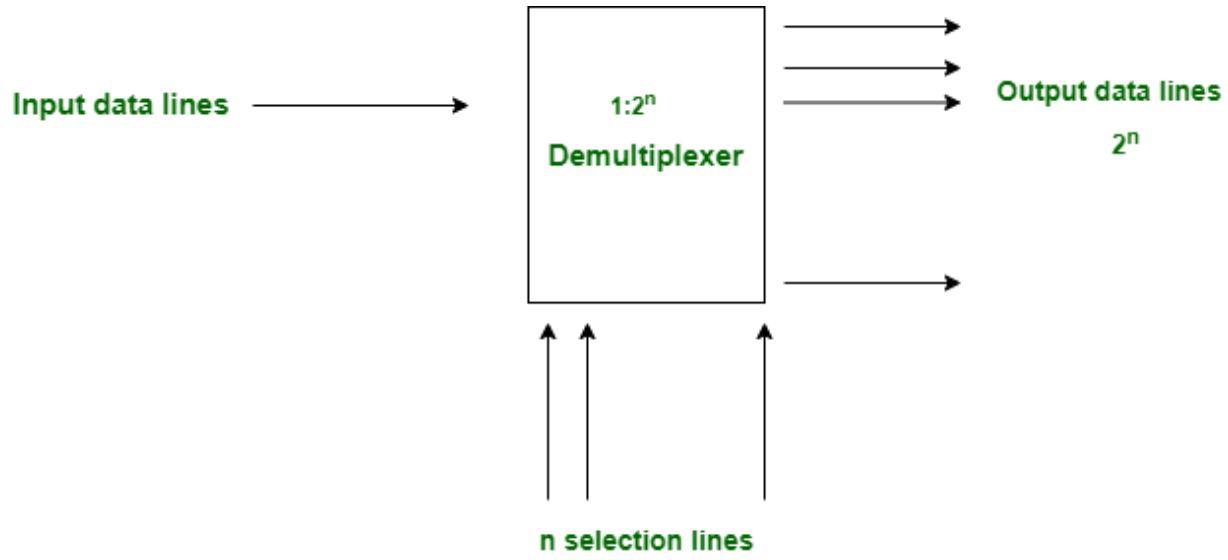
Applications of MUX

- **Data Routing** : The Mux is used for data routing in the digital system where they select one of the several data lines and re-route it the output.
- **Data Selection** : The Mux is used for data selection where they select data source according to the select lines.
- **Analog-to-Digital Conversion** : The Mux are used in ADC to select different analog input channels.
- **Address Decoding** : The Mux are used in Microprocessors or memory for address decoding.
- **Logic Function Implementation** : Mux can be used to implement various logic functions.

What is Demultiplexer (DEMUX)?

Demultiplexer is the opposite of multiplexer. It is also termed as DEMUX. It takes input from one source and also converts the data to transmit towards various sources. The demultiplexer has one data input line. The demultiplexer has several control lines (also known as select lines). These lines determine to which output the input data should be sent. The number of control lines determines the number of output lines.

Given below is the block diagram of the Demultiplexer, it will have one Input line and will give 2n output lines.



Advantages of Demultiplexers (DEMUX)

1. **Efficient Data Distribution**: Routes one input signal to multiple outputs effectively.
2. **Reduced Transmission Complexity**: Simplifies transmitter design by minimizing input lines.
3. **High-Speed Data Splitting**: Ideal for applications requiring rapid data distribution.
4. **Scalability**: Can handle more outputs by increasing control lines.

5. **Versatility**: Widely used in TV broadcasting, communication networks, and more.

Disadvantages of Demultiplexers (DEMUX)

1. **Control Complexity**: Additional circuits are needed for output selection.
2. **Limited Outputs**: The number of outputs depends on available control lines.
3. **Propagation Delay**: Routing input to outputs may introduce delays in larger systems.
4. **Signal Degradation**: Signal strength may reduce when split into multiple outputs.
5. **Higher Power Consumption**: Power usage increases with additional outputs.
6. **Noise Susceptibility**: Splitting signals can lead to interference

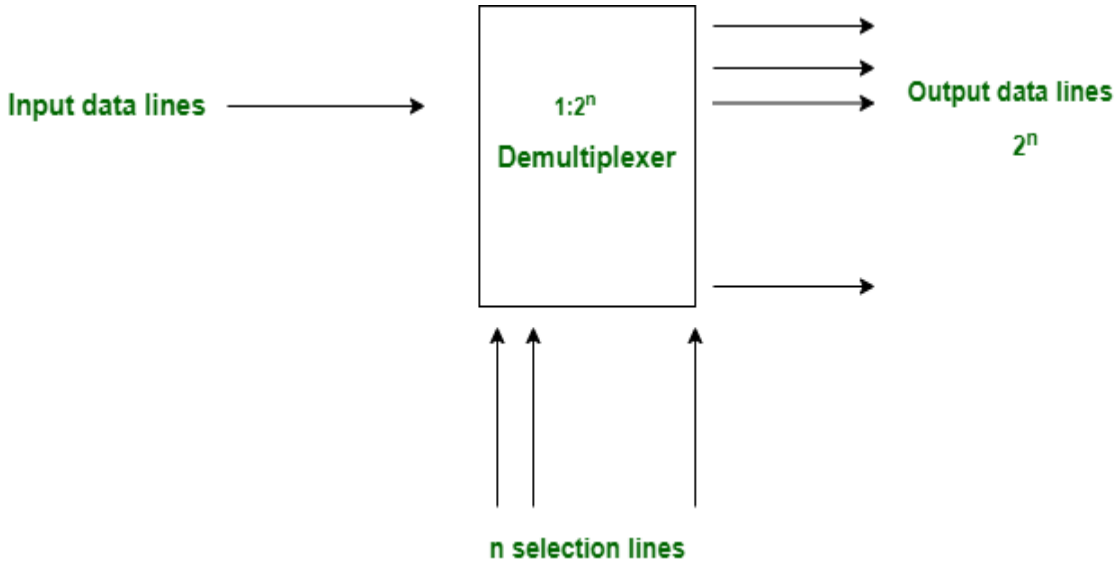
DEMUX

The DEMUX is a digital information processor. It takes input from one source and also converts the data to transmit towards various sources. The demultiplexer has one data input line. The demultiplexer has several control lines (also known as select lines). These lines determine to which output the input data should be sent. The number of control lines determines the number of output lines.

Let us discuss the DEMUX with the attached information.

General Block Diagram Of A DEMUX

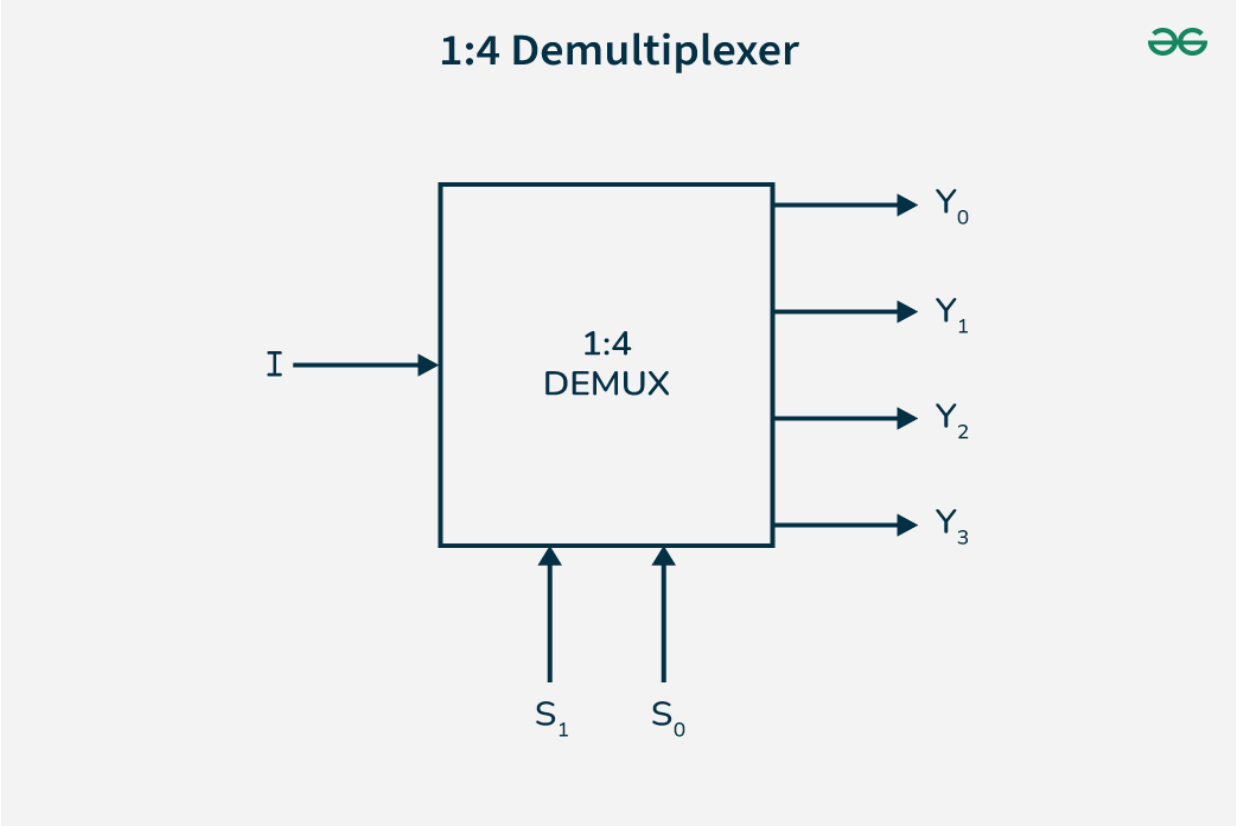
Here is the basic block diagram of a DEMUX as mentioned below.



DEMUX

Truth Table of A 1X4 DEMUX

A 1x4 DEMUX has only one input which is denoted as I. There are two selection lines i.e. S1 and S0. At last, the DEMUX has output lines including Y3, Y2, Y1 &Y0. Here is the 1x4 DEMUX with diagram as mentioned below**.**



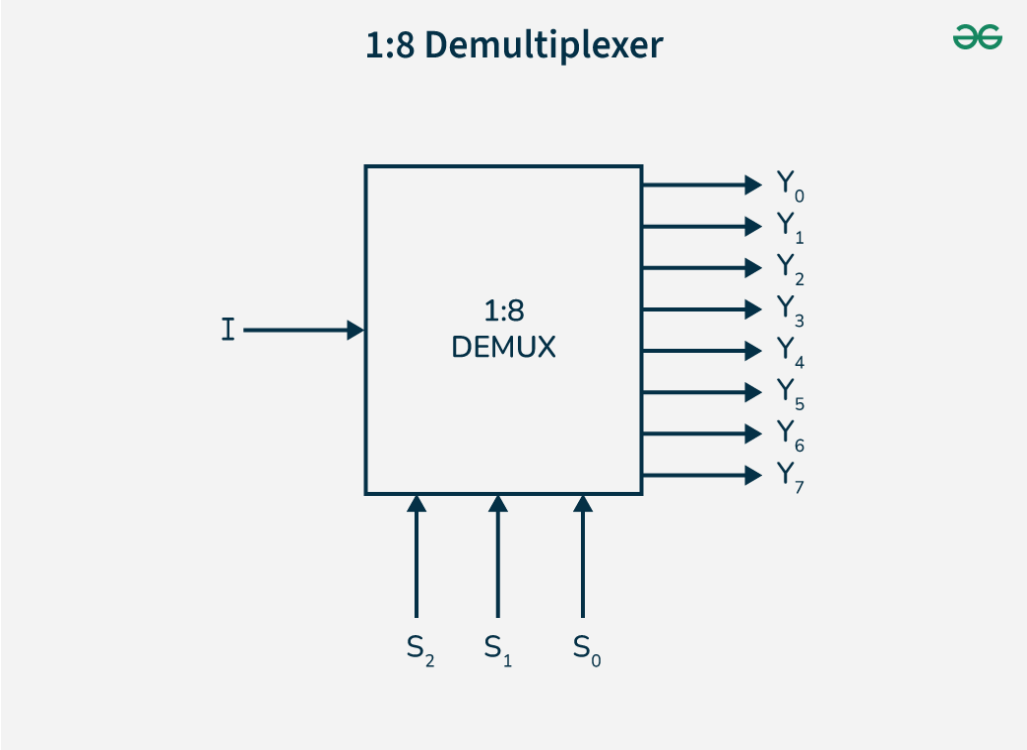
1:4 DEMUX

The truth table of the 1x4 DEMUX as mentioned below.

Selection Inputs		Outputs			
S1	**S0**	**Y3**	**Y2**	**Y1**	**Y0**
0	**0**	**0**	**0**	**0**	**I**
0	**1**	**0**	**0**	**I**	**0**
1	**0**	**0**	**I**	**0**	**0**
1	**1**	**I**	**0**	**0**	**0**

Truth Table of A 1x8 De-Multiplexer

The 1x8 DEMUX was designed by using two DEMUX. They are the two 1x4 DEMUX and one 1x2 DEMUX. The 1x8 DEMUX contains two input lines with four outputs. Let us see the block diagram of the 1x8 DEMUX as mentioned below.



1:8 DEMUX

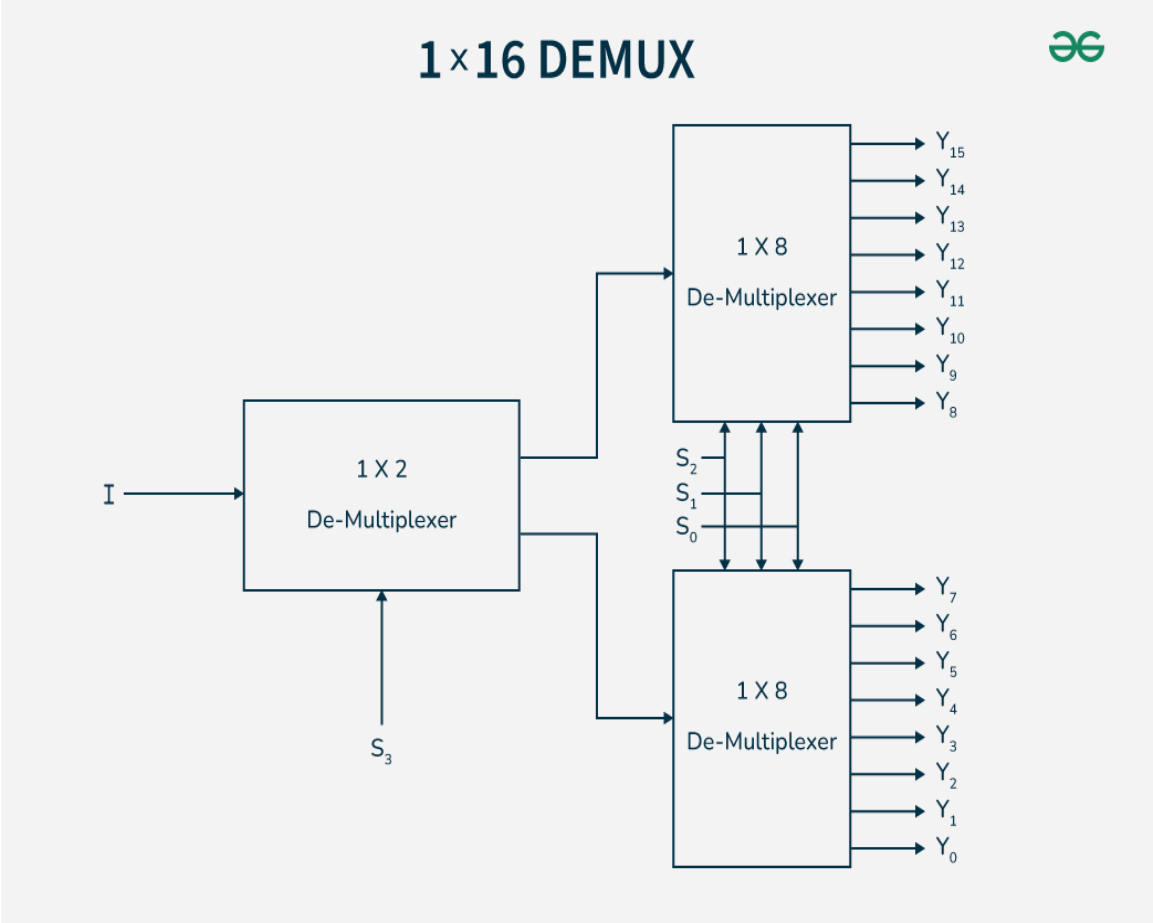
Here is the 1x8 DEMUX truth table as mentioned below.

Selection Inputs			Outputs							
S2	S1	S0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0

Selection Inputs			Outputs							
S2	S1	S0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

Truth Table Of A 1x16 De-Multiplexer

The 1x16 DEMUX was designed by using the two 1x8 DEMUX and one 1x2 DEMUX. The 1x16 DEMUX have two input lines. It has eight outputs. Let us see the block diagram of the 1x16 DEMUX as mentioned below.



1:16 DEMUX

Try yourself to make truth table

Advantages and Disadvantages of the DEMUX

Now, we are going to discuss the advantages and disadvantages of the DEMUX as mentioned below.

Advantages of the DEMUX

- The DEMUX increases the efficiency of the particular communication system as it takes data from a specific input source and distributes it to different sources.
- The DEMUX helps to separate the different signals from the mixed data sources. Then it distributes these data to different sources.
- DEMUX can decode the signal outputs of the multiplexer, as the system works in a reverse way of the MUX.

Disadvantages of the DEMUX

- The DEMUX may cause a wastage of bandwidth as it distributes the refined data in different channels. These channels can overlap with each other which leads to the loss of signal.
- The DEMUX may cause problems in the synchronization of signals. The data channels can overlap with each other which leads to the delay in the whole process.

Applications of DEMUX

- DEMUXs are majorly used to design the ALU circuits and parallel data segments.
- The DEMUXs convert the output of the MUX into the actual input. On the receiver end, the DEMUX can be used to verify the original form of the data to carry out the entire communication process.
- DEMUX helps to save the output. The output generally saved to the ALU. The output will be saved in the registers and the various storage units of the system.
- Each DEMUX has a connection with multiple registers which helps to store the processed data into it.
- The counterpart of the DEMUX regulates the data signal at the output stage of the DEMUX operation. These data can be retrieved later to read out parallelly.
- In the audio or video system, the DEMUX distributes them in different channels. In broadcasting, the DEMUX separates the composite signal into an individual one.

Difference Between of Multiplexer and Demultiplexer

Multiplexer(MUX)	**Demultiplexer(DEMUX)**
Multiplexer processes the digital information from various sources into a single source.	Demultiplexer receives digital information from a single source and converts it into several sources
It is known as Data Selector	It is known as Data Distributor
Multiplexer is a digital switch	Demultiplexer is a digital circuit
It follows combinational logic type	It also follows combinational logic type
It has 2n input data lines	It has single input line

Multiplexer(MUX)	Demultiplexer(DEMUX)
It has a single output data line.	It has 2n output data lines
Efficiently uses bandwidth by combining many signals into a single line for transmission.	Divides a single signal into several parts, so bandwidth is less efficiently used.
Needs control lines to select which input signal to send to the output.	Needs control lines to determine which output line should receive the input signal.
It works on many to one operational principle	It works on one to many operational principles.
May consume more power due to the need for control logic and multiple input connections.	Typically uses less power, especially when splitting a single signal to multiple destinations.
In time division Multiplexing, multiplexer is used at the transmitter end.	In time division Multiplexing, demultiplexer is used at the receiver end.

Programmable Logic Devices

Programmable Logic Devices (PLDs) are a collection of integrated circuits which are configured to perform various logical functions. PLDs play an important role in the field of engineering and technology, as they form the basis of innovation and support engineers to develop automated digital systems to improve process flexibility and efficiency. Here, "programmable" means defining a function that can be performed multiple times without human intervention.

Programmable Logic Devices (PLDs) are the integrated circuits. They contain an array of AND gates & another array of OR gates. There are three kinds of PLDs based on the type of array(s), which has programmable feature.

- Programmable Read Only Memory
- Programmable Array Logic

- Programmable Logic Array

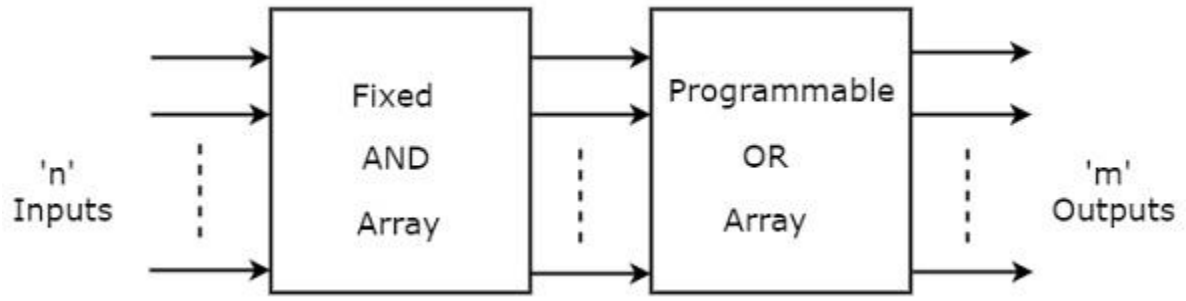
The process of entering the information into these devices is known as programming. Basically, users can program these devices or ICs electrically in order to implement the Boolean functions based on the requirement. Here, the term programming refers to hardware programming but not software programming.

In this chapter, we will explain the basic concepts of programmable logic devices, their types, advantages, limitations, and applications.

Programmable Read Only Memory (PROM)

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 is shown in the following figure.



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.

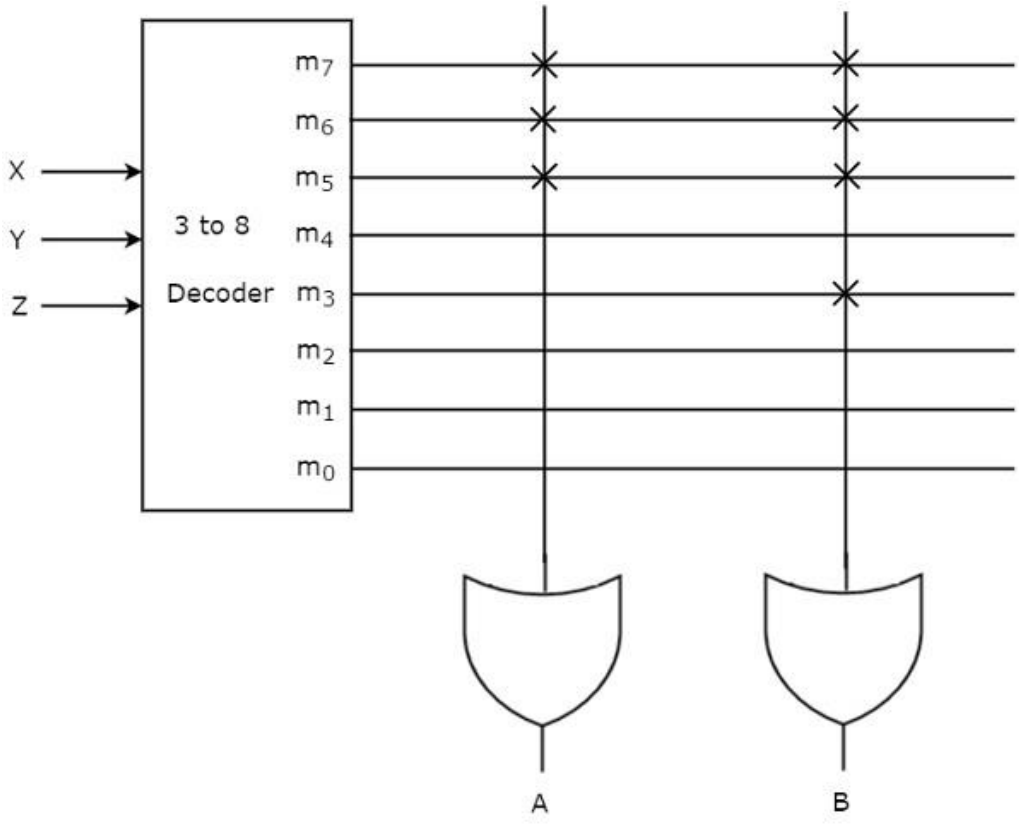
Example

Let us implement the following Boolean functions using PROM.

A (X, Y, Z) =∑m (5,6,7) A (X, Y, Z) =∑m (5,6,7)

B (X, Y, Z) =∑m (3,5,6,7) B (X, Y, Z) =∑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.

Code Converters - BCD (8421) to/from Excess-3

Prerequisite

Excess-3 binary code is an **unweighted self-complementary** BCD code. Self-Complementary property means that the 1's complement of an excess-3 number is the excess-3 code of the 9's complement of the corresponding decimal number. This property is useful since a decimal number can be nines complemented (for subtraction) as easily as a binary number can be ones complemented; just by inverting all bits.

For example, the excess-3 code for 3(0011) is 0110, and to find the excess-3 code of the complement of 3, we just need to find the 1's complement of 0110 -> 1001, which is also the excess-3 code for the 9's complement of 3 -> (9-3) = 6.

What is BCD?

BCD is a class of binary encodings of decimal numbers in which each decimal digit is represented by its own binary sequence. In the most common BCD encoding, the 8421 code, each decimal

digit is represented by a 4-bit binary number. In the most common BCD encoding, the 8421 code, each decimal digit is represented by a 4-bit binary number:

0 is 0000

1 is 0001

2 is 0010

3 is 0011

4 is 0100

5 is 0101

6 is 0110

7 is 0111

8 is 1000

9 is 1001

BCD is used in systems that require the digital manipulation of decimal numbers as it is a technique that can be used for the arithmetic of BCD numbers. It gives an easy method of passing from one base to the other and saves much time in calculations where the accuracy of decimal is necessary.

What is Excess-3 Code?

Excess-3 code is one of the BCD code where every single decimal number is represented by a four-bit binary number which is three in excess of BCD value. For instance, the decimal digit 0 will be represented in Excess-3 code as 0010 as since the scheme adds 3 to the original numbers. It is an unweighted self-complementary code which is employed in the digital systems with the intention of having it facile in taking complements and in arithmetic subtraction.

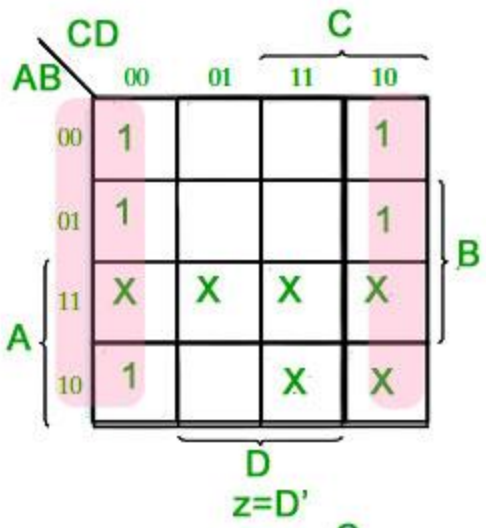
The self-complementary aspect to Excess-3 is that the 1's complement of an Excess-3 number is equal to the Excess-3 code of the 9's complement of the decimal form of the given numeral. Great care has been taken here to ensure that this feature can be used to compute the 9's complement by just flipping all the bits.

Converting BCD (8421) to Excess-3

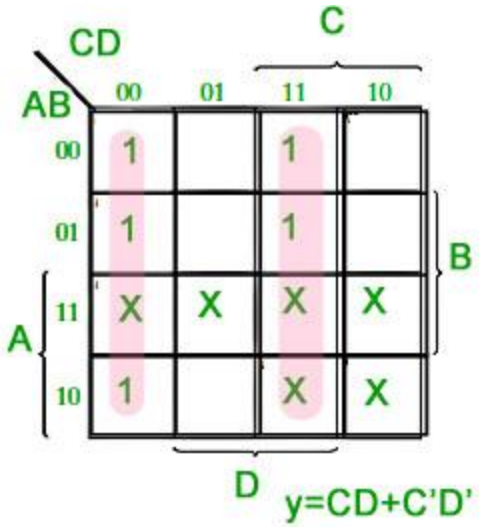
As is clear by the name, a BCD digit can be converted to its corresponding Excess-3 code by simply adding 3 to it. Since we have only 10 digits(0 to 9) in decimal, we don't care about the rest and marked them with a cross(X).
Let A,B,C,andD $A,B,C,andD$ be the bits representing the binary numbers, where D D is the LSB and A A is the MSB, and
Let w,x,y,andz $w,x,y,andz$ be the bits representing the gray code of the binary numbers, where z z is the LSB and w w is the MSB.
The truth table for the conversion is given below. The X's mark is don't care condition.

BCD(8421)				Excess-3			
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

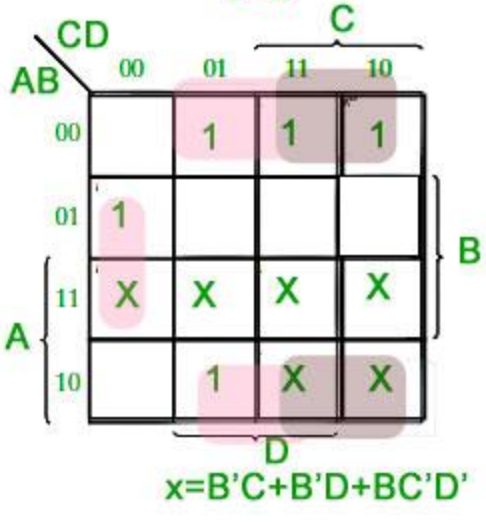
To find the corresponding digital circuit, we will use the K-Map technique for each of the Excess-3 code bits as output with all of the bits of the BCD number as input.



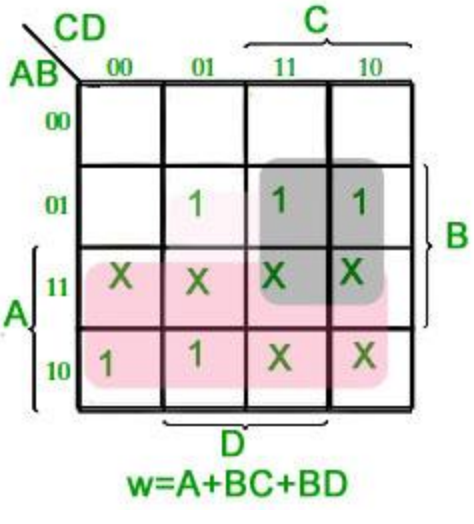
z=D'



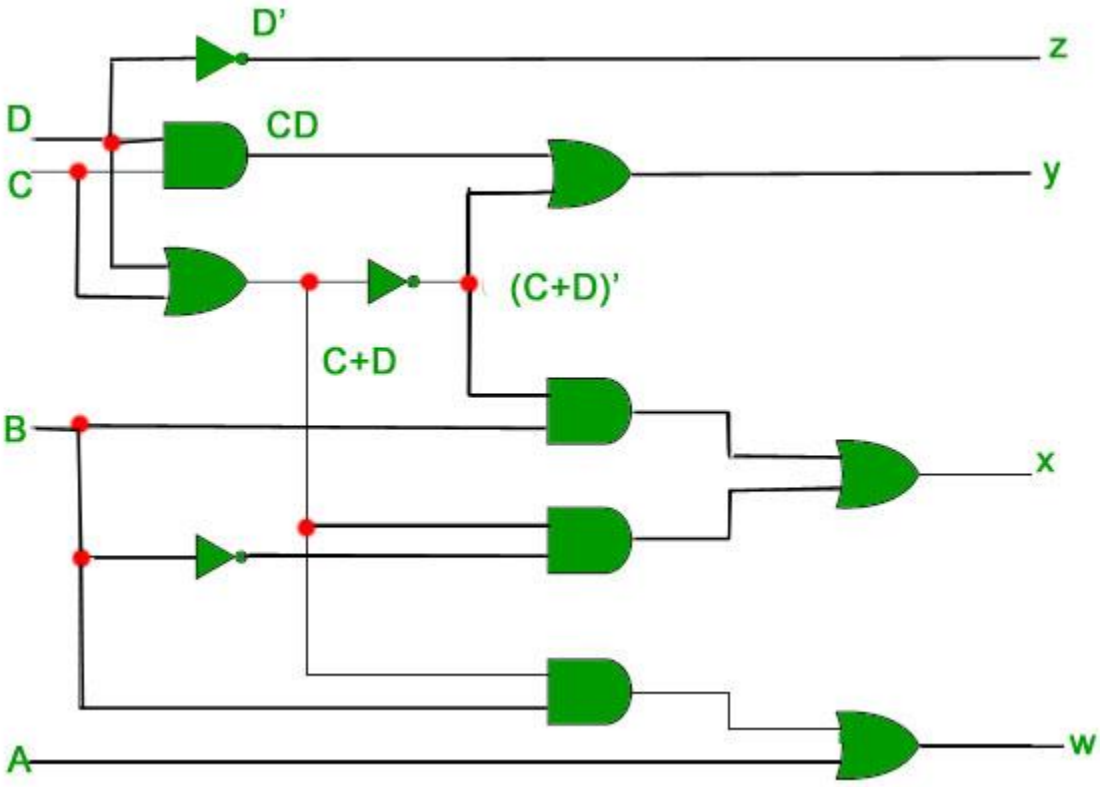
y=CD+C'D'



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



w=A+BC+BD

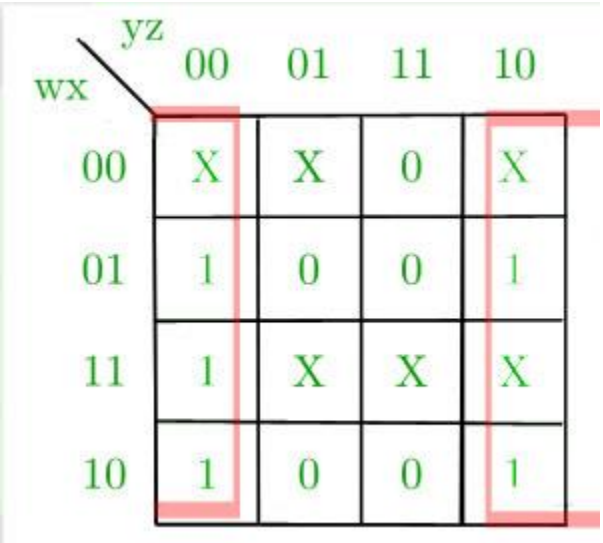


Converting Excess-3 to BCD(8421)

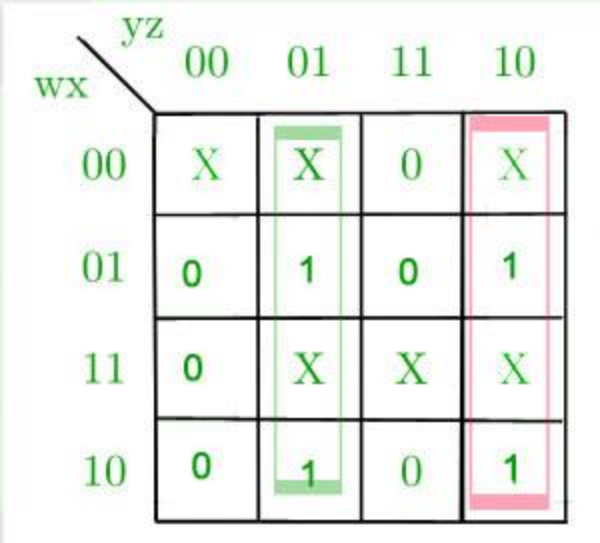
Excess-3 code can be converted back to BCD in the same manner. Let A,B,C,andD *A,B,C,andD* be the bits representing the binary numbers, where D . *D* is the LSB and A *A* is the MSB, and Let w,x,y,andz *w,x,y,andz* be the bits representing the gray code of the binary numbers, where z *z* is the LSB and w *w* is the MSB. The truth table for the conversion is given below. The X's mark is don't care condition.

Excess-3				BCD			
w	x	y	z	A	B	C	D
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

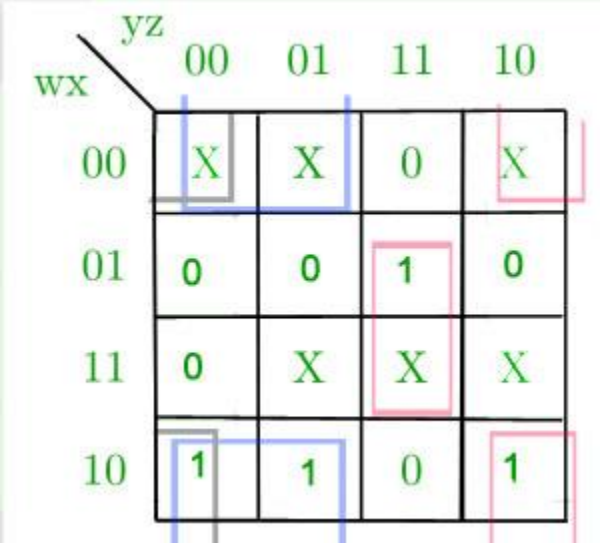
K-Map for D



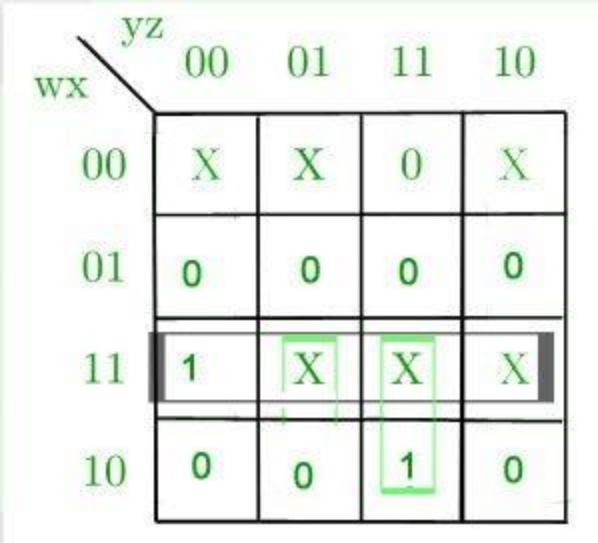
K-Map for C



K-Map for B

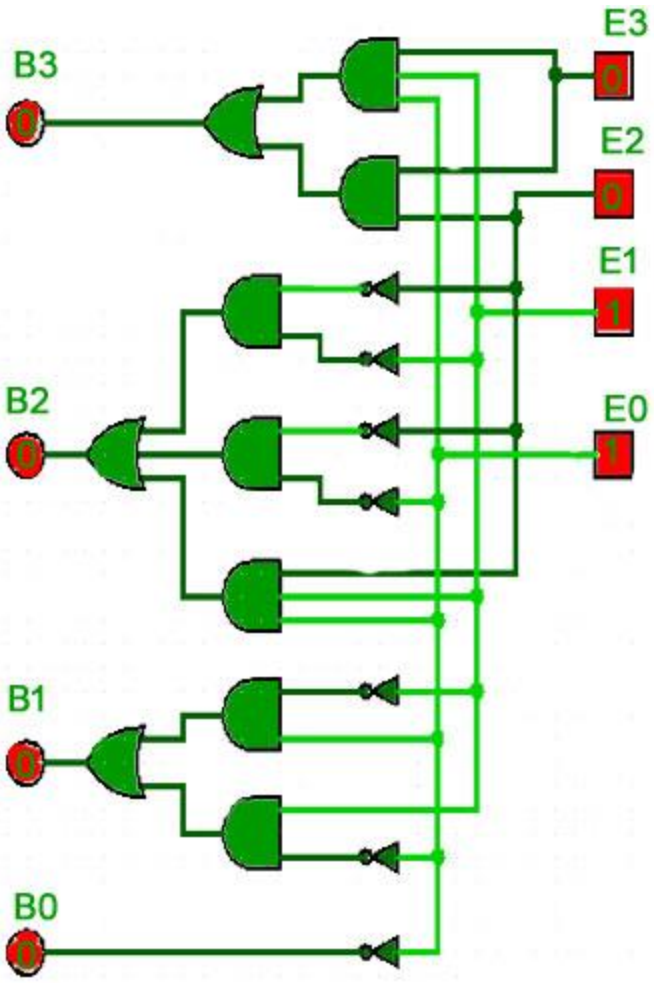


K-Map for A



Corresponding minimized Boolean expressions for Excess-3 code bits - A=wx+wyzB=x′y′+x′z′+xyzC=y′z+yz′D=z′ *A=wx+wyzB=x′y′+x′z′+xyzC=y′z+yz′D=z′*

The corresponding digital circuit - Here E3,E2,E1,andE0 *E*3,*E*2,*E*1,*andE*0 correspond to w,x,y,andz *w,x,y,andz* and B3,B2,B1,andB0 *B*3,*B*2,*B*1,*andB*0 correspond to A,B,C,andD *A,B,C,andD* .



Difference Between BCD and Excess-3

Feature	BCD	Excess-3
Coding Method	Represents directly each decimal digit with its binary equivalent.	Represents each decimal digit with its binary equivalent plus 3.
Value range	Encodes decimal digits 0-9.	Encodes decimal digits 3-12 (with 0-9 wrapped around).
Arithmetic Operations	Additional logic needed for arithmetic operations.	Arithmetic operations are simplified by self- complementary.

Feature	BCD	Excess-3
Error Detection	Not very effective in detecting and correcting errors.	It is better in the error detection capability since it applies an encoding technique.
Complement Operation	The operation is slightly harder in implementing complex operations.	Complement operation in a more straightforward way due to self-complementing nature.
Binary Representation	Uses 4-bit binary to represent each decimal digit.	Uses 4-bit binary, but added 3 to the binary representation of each decimal digit.
Conversion Complexity	Direct conversion to/from decimal.	For conversion, add or subtract 3 for the binary value.

THE END