UNIT 3:
Combinational logical circuits

Half Adder Vs. Full Adder: What is the Difference Between Half Adder and Full Adder?

Both Half and Full adders are combinational logic circuits, and they both differ from each other in the aspect of input processing. Any combinational circuit is devoid of memory elements- they only comprise the logic gates. There is a primary difference between half adder and full adder. Half adder only adds the current inputs as 1-bit numbers and does not focus on the previous inputs. On the other hand, Full Adder can easily carry the current inputs as well as the output from the previous additions.

Before we let us look into more difference between half adder and full adder, let us know a bit more about them individually.

What is a Half Adder?

It is a combinational logic circuit. You can design it by connecting one AND gate and one EX-OR gate. A half-adder circuit consists of two input terminals- namely A and B. Both of these add two input digits (one-bit numbers) and generate the output in the form of a carry and a sum. Thus, there are two output terminals.

The output that one obtains from the EX-OR gate is the sum of both the one-bit numbers. The output obtained from the AND gate is called the carry. But you cannot forward the carry that you obtain in one addition into another addition. It is because of the absence of any logic gate to process it. Thus, it’s called the Half Adder circuit.

We can write the equation of output for both the gates in the form of a logical operation that the logic gates perform. Here, we write the carry equation in the form of AND operation and the sum equation in the form of EX-OR operation.

Logical Expression of Half Adder

Sum (S) = $A \oplus B$

Carry (C) = $A . B$

Truth Table

Here is a truth table representing the possible outputs obtained from the possible inputs in a Half Adder:

Input		**Output**	
A	B	CARRY	SUM
0	0	0	0
1	1	1	0
0	1	0	1
1	0	0	1

What is Full Adder?

A full adder is a circuit that has two AND gates, two EX-OR gates, and one OR gate. The full adder adds three binary digits. Among all the three, one is the carry that we obtain from the previous addition as C-IN, and the two are inputs A and B. It designates the input carry as the C-OUT and the normal output as S (or SUM).

Just like the Half Adder, the Full Ladder is a combinational type of logic circuit- meaning, it has no storage element. But it has additional logic gates. Thus, it adds the previous carry to generate the complete output. Thus, it is called the Full Adder.

One can also designate a Full Adder using one OR gate and two Half Adders. The OR gate here generates a carry that it obtains after the addition. We obtain the sum of these digits in the form of output from the second Half Adder.

The equation for the output that you can obtain by the EX-OR gate is the sum of all the binary digits. Here, the output that you obtain from the AND gate is the carry that you obtain by addition. This equation is in the form of a logical operation.

Logical Expression of Full Adder

CARRY-OUT = AB + BCin + ACin

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

Truth Table

A truth table represents the possible outputs obtained from the possible inputs. A truth table for the Full Adder is as follows:

Input			**Output**	
A	B	C	SUM	CARRY OUT
0	0	0	0	0
1	1	1	1	1
0	1	1	0	1
1	0	1	0	1
0	0	1	1	0

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

Difference Between Half Adder and Full Adder

Parameter	**Half Adder**	**Full Adder**
Basics	The Half Adder is a type of combinational logic circuit that adds two of the 1-bit binary digits. It generates carry and sum of both the inputs.	The Full Adder is also a type of combinational logic that adds three of the 1-bit binary digits for performing an addition operation. It generates a sum of all three inputs along with a carry value.
Adding the Previous Carry	The Half Adder does not add the carry obtained from the previous addition to the next one.	The Full Adder, along with its current inputs A and B, also adds the previous carry.
Hardware Architecture	A Half Adder consists of only one AND gate and EX-OR gate.	A Full Adder consists of one OR gate and two EX-OR and AND gates.
Total Inputs	There are two inputs in a Half Adder- A and B.	There are a total of three inputs in a Full Adder- A. B. C-in.
Usage	The Half Adder is good for digital measuring devices, computers, calculators, and many more.	The Full Adder comes into play in various digital processors, the addition of multiple bits, and many more.
Logical Expression	Here is the logical expression of Half Adder:	Here is the logical expression of Full Adder: $Cout = (AB) + CinA \oplus CinB$

	$C = A * B$ $S = A \oplus B$	$S = A \oplus B \oplus Cin$

Addition is a fundamental operation in digital electronics, and is used in a wide range of applications such as arithmetic, data processing, and control systems. There are two main types of adders used in digital circuits: Serial Adder and Parallel Adder. Understanding the differences between these two types of adders is essential to designing and implementing efficient and effective digital systems.

1. Serial Adder: A serial adder is used to add two binary numbers in serial form. The two binary numbers to be added serially are stored in two shift registers. The circuit adds one pair at a time with the help of one full adder. The carry output from the full adder is applied to a D flip-flop, the output of which is then used as a carry input for the next pair of significant bits. However, the sum bit S from the output of the full adder can be transferred into a third shift register.
2. Parallel Adder: A parallel adder is a combinational digital circuit that adds two binary numbers in parallel form. It consists of full adders connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder.

Similarities:

- Both Serial Adder and Parallel Adder are used for adding binary numbers in digital circuits.
- They both use full adders as the basic building blocks for performing addition operations.
- Both types of adders have inputs for two binary numbers and a carry-in bit, and outputs for a sum and a carry-out bit.
- The operation of both types of adders is based on the principles of binary addition, where a carry-out bit is generated if the sum of the two bits is greater than or equal to two.
- They can both be used for various applications in digital electronics, such as arithmetic, data processing, and control systems.

Difference between Serial Adder and Parallel Adder:

Parameters	**Serial Adder**	**Parallel Adder**
Addition manner	It is used to add two binary numbers in serial form.	It is used to add two binary numbers in parallel form.

Parameters	**Serial Adder**	**Parallel Adder**
Type of Registers	A serial adder uses shift registers.	A parallel adder uses registers with parallel loads.
Requirement	It requires a single full adder.	It requires multiple full adders.
Usage of	A carry flip-flop is used in the serial adder.	Ripple carry adder is used in the parallel adder.
Circuit Type	A serial adder is a sequential circuit.	A parallel adder is a combinational circuit.
Propagation Delay	In serial adder, propagation delay is less.	In parallel adder, propagation delay is present from input carry to output carry.
Speed	The serial adder has a slow speed as compared to the parallel adder.	The parallel adder has fast speed as compared to the serial adder.
Addition process	The addition process is carried out bit by bit. Therefore, addition time relies on bit count.	The addition process is carried out simultaneously. That implies all bits sum up simultaneously. Therefore, time does not rely on bit count.
Requirement of Components	It necessitates fewer components.	It necessitates more components because of design complexity.

Parameters	**Serial Adder**	**Parallel Adder**
Number of Full Adders	The number of required full adders is fixed i.e. one.	The number of required full adders is equal to the number of bits in the binary number.

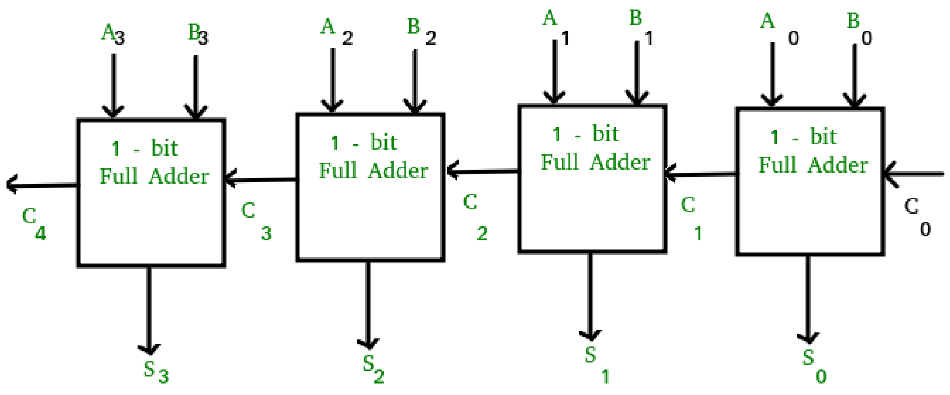
Carry Look-Ahead Adder

The adder produce carry propagation delay while performing other arithmetic operations like multiplication and divisions as it uses several additions or subtraction steps. This is a major problem for the adder and hence improving the speed of addition will improve the speed of all other arithmetic operations. Hence reducing the carry propagation delay of adders is of great importance. There are different logic design approaches that have been employed to overcome the carry propagation problem. One widely used approach is to employ a carry look-ahead which solves this problem by calculating the carry signals in advance, based on the input signals. This type of adder circuit is called a carry look-ahead adder.

Here a carry signal will be generated in two cases:

1. Input bits A and B are 1
2. When one of the two bits is 1 and the carry-in is 1.

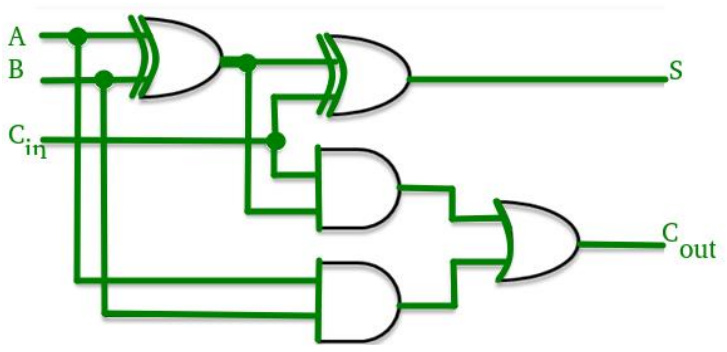
In ripple carry adders, for each adder block, the two bits that are to be added are available instantly. However, each adder block waits for the carry to arrive from its previous block. So, it is not possible to generate the sum and carry of any block until the input carry is known. The block waits for the block to produce its carry. So there will be a considerable time delay which is carry propagation delay.

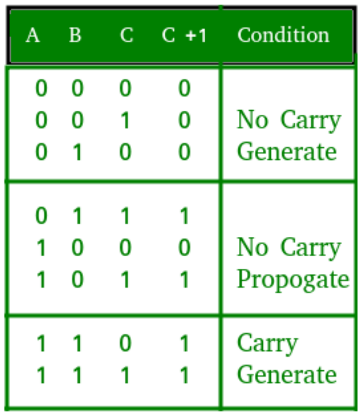


Consider the above 4-bit ripple carry adder. The sum is produced by the corresponding full adder as soon as the input signals are applied to it. But the carry input is not available on its final steady-state value until carry is available at its steady-state value. Similarly depends on and on . Therefore, though the carry must propagate to all the stages in order that output and carry settle their final steady-state value.

The propagation time is equal to the propagation delay of each adder block, multiplied by the number of adder blocks in the circuit. For example, if each full adder stage has a propagation delay of 20 nanoseconds, then will reach its final correct value after 60 (20 × 3) nanoseconds. The situation gets worse, if we extend the number of stages for adding more number of bits.

Carry Look-ahead Adder :
A carry look-ahead adder reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic. Let us discuss the design in detail.





A	B	C	C +1	Condition
0	0	0	0	No Carry Generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No Carry Propogate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry Generate
1	1	1	1	

Consider the full adder circuit shown above with corresponding truth table. We define two variables as **'carry generate'** and **'carry propagate'** then,

The sum output and carry output can be expressed in terms of carry generate and carry propagate as

where produces the carry when both , are 1 regardless of the input carry. is associated with the propagation of carry from to .

The carry output Boolean function of each stage in a 4 stage carry look-ahead adder can be expressed as

C_1 = Go+ PoC$_{in}$

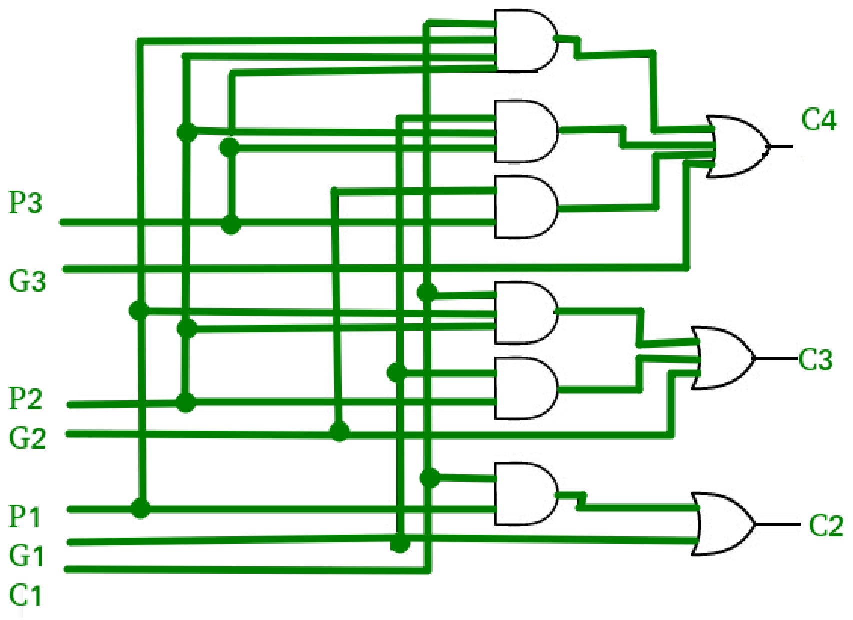
C_2 G1+ PIC1 G1+ PIGO+ Pi PoCin

Ca G_2+ P_2C2 G_2+P2G1 + P2P Go + P2P PoCin

C_1 = G3+ P3C3 G3+ P3G2+ P3P2G1 + P3P2P Go+ P3P2P PoCin =

From the above Boolean equations we can observe that does not have to wait for and to propagate but actually is propagated at the same time as and . Since the Boolean expression for each carry output is the sum of products so these can be implemented with one level of AND gates followed by an OR gate.

The implementation of three Boolean functions for each carry output (, and) for a carry look-ahead carry generator shown in below figure.



Time Complexity Analysis :
We could think of a carry look-ahead adder as made up of two "parts"

1. The part that computes the carry for each bit.
2. The part that adds the input bits and the carry for each bit position.

The complexity arises from the part that generates the carry, not the circuit that adds the bits.

Now, for the generation of the carry bit, we need to perform a AND between (n+1) inputs. The complexity of the adder comes down to how we perform this AND operation. If we have AND gates, each with a fan-in (number of inputs accepted) of k, then we can find the AND of all the bits in time. This is represented in asymptotic notation as .

Advantages and Disadvantages of Carry Look-Ahead Adder :
Advantages –

- The propagation delay is reduced.
- It provides the fastest addition logic.

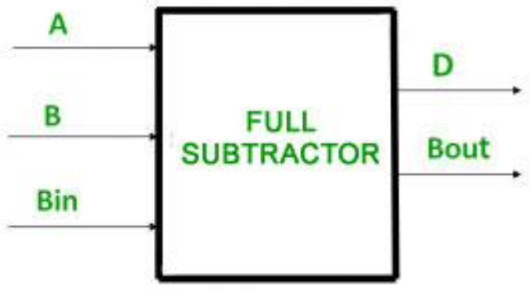
Disadvantages –

- The Carry Look-ahead adder circuit gets complicated as the number of variables increase.
- The circuit is costlier as it involves more number of hardware.

NOTE :
For n-bit carry lookahead adder to evaluate all the carry bits it requires [n(n + 1)]/2 AND gates and n OR gates.

Full Subtractor

A full subtractor is a **combinational circuit** that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit. This circuit **has three inputs and two outputs**. The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and Bout represent the difference and output borrow, respectively. Although subtraction is usually achieved by adding the complement of subtrahend to the minuend, it is of academic interest to work out the Truth Table and logic realisation of a full subtractor; x is the minuend; y is the subtrahend; z is the input borrow; D is the difference; and B denotes the output borrow. The corresponding maps for logic functions for outputs of the full subtractor namely difference and borrow.

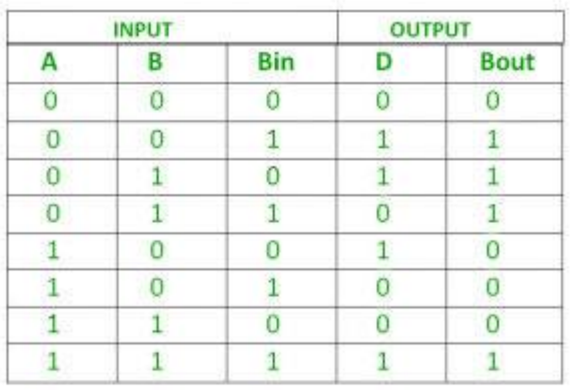


Here's how a full subtractor works:
1. First, we need to convert the binary numbers to their two's complement form if we are subtracting a negative number.
2. Next, we compare the bits in the minuend and subtrahend at the corresponding positions. If the subtrahend bit is greater than or equal to the

minuend bit, we need to borrow from the previous stage (if there is one) to subtract the subtrahend bit from th**e** minuend bit.
3. We subtract the two bits along with the borrow-in to get the difference bit. If the minuend bit is greater than or equal to the subtrahend bit along with the borrow-in, then the difference bit is 1, otherwise it is 0.
4. We then calculate the borrow-out bit by comparing the minuend and subtrahend bits. If the minuend bit is less than the subtrahend bit along with the borrow-in, then we need to borrow for the next stage, so the borrow-out bit is 1, otherwise it is 0.
The circuit diagram for a full subtractor usually consists of two half-subtractors and an additional OR gate to calculate the borrow-out bit. The inputs and outputs of the full subtractor are as follows:

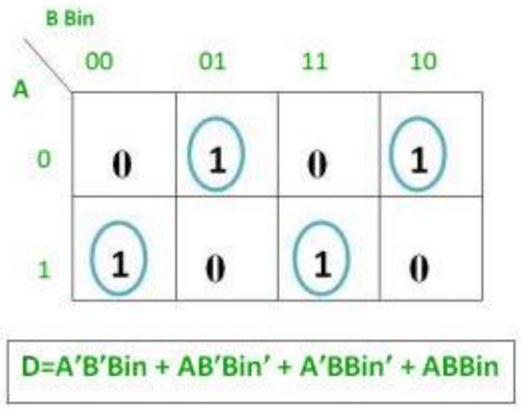
Inputs:
A: minuend bit
B: subtrahend bit
Bin: borrow-in bit from the previous stage
Outputs:
Diff: difference bit
Bout: borrow-out bit for the next stage

Truth Table –

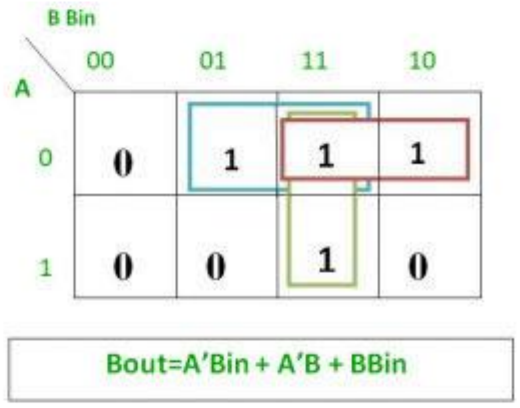


INPUT			OUTPUT	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

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



“borrow”.



Logical expression for difference –

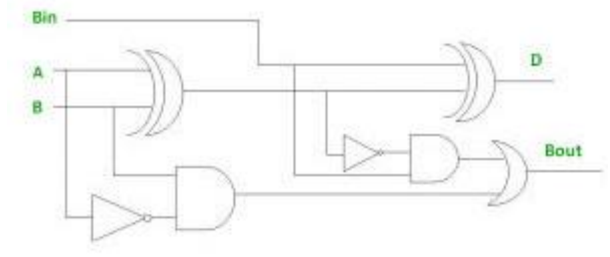
```
D   = A’B’Bin + A’BBin’ + AB’Bin’ + ABBin

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

    = Bin( A XNOR B) + Bin’(A XOR B)

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

    = Bin XOR (A XOR B)

    = (A XOR B) XOR Bin
```

Logical expression for borrow –

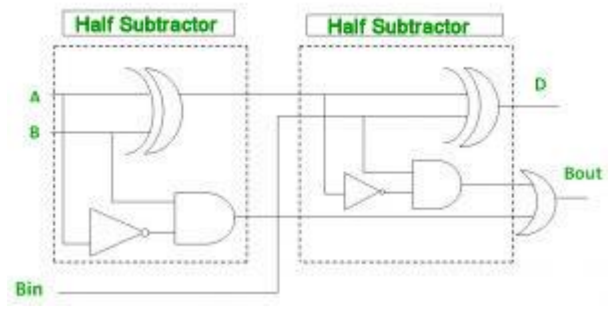
```
Bout = A’B’Bin + A’BBin’ + A’BBin + ABBin

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

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

     = A’Bin + A’B + BBin
```

```
OR


Bout = A’B’Bin + A’BBin’ + A’BBin + ABBin
     = Bin(AB + A’B’) + A’B(Bin + Bin’)
     = Bin( A XNOR B) + A’B
     = Bin (A XOR B)’ + A’B
```

Logic Circuit for Full Subtractor –



Implementation of Full Subtractor using Half Subtractors – 2 Half Subtractors and an OR gate is required to implement a Full Subtractor.



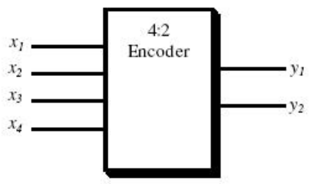
CODE CONVERTERS

Decoder and encoder circuits are used to convert from one type of input encoding to a different output encoding. For example, a 2: 4 decoder converts a 2-bit binary number input to a one-hot encoding sequence (see Section 9.6) at the output. Similarly, a 4: 2 binary encoder performs the opposite conversion. There are many other possible types of code converters known as BCD-to-seven-segment code converter, BCD-to-Gray code converter, BCD-to-excess-3 code converters, and so on.

Truth Table

x_4	x_3	x_2	x_1	y_2	y_1
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Block Diagram



Logic Circuit Implementation

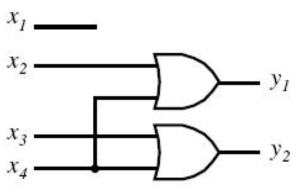
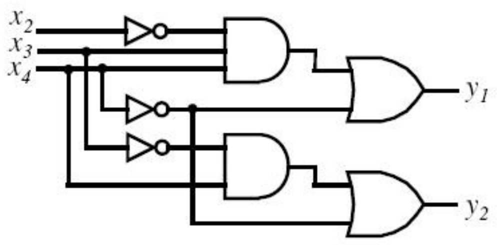


Figure 7.21 4 : 2 Binary Encoder

Truth Table

x_4	x_3	x_2	x_1	y_2	y_1
0	0	0	1	0	0
1	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Logic Implementation

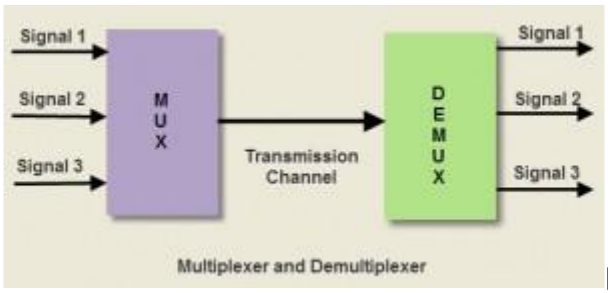


Multiplexer and Demultiplexer : Types and Their Differences

In large scale digital systems, a single line is required to carry on two or more digital signals – and of course! at a time, one signal can be placed on one line. But, what is required is a device that will allow us to select; and, the signal we wish to place on a common line, such a circuit is referred to as a multiplexer. The function of a multiplexer is to select the input of any 'n' input lines and feed that to one output line. The function of a demultiplexer is to inverse the function of the multiplexer. The shortcut forms of the multiplexer and demultiplexers are mux and demux. Some multiplexers perform both multiplexing and demultiplexing operations. The main function of the multiplexer is that it combines input signals, allows data compression, and shares a single transmission channel. This article gives an overview of multiplexer and demultiplexer.

What are Multiplexer and Demultiplexer?

In-network transmission, both the multiplexer and demultiplexer are combinational circuits. A multiplexer selects an input from several inputs then it is transmitted in the form of a single line. An alternative name of the multiplexer is MUX or data selector. A demultiplexer uses one input signal and generates many. So it is known as Demux or data distributor.

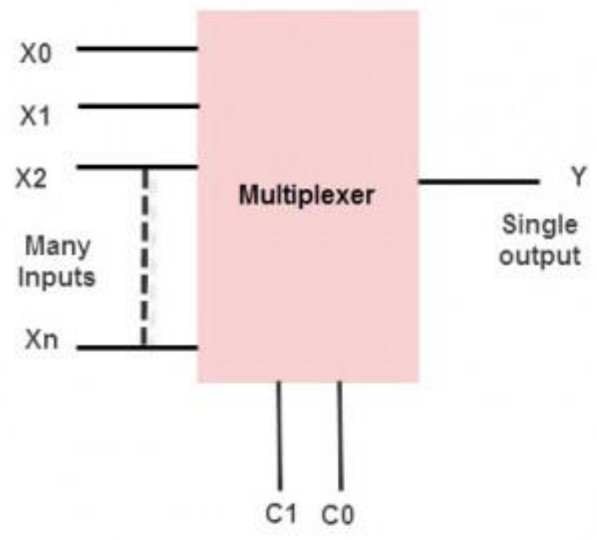


Multiplexer and Demultiplexer

What is a Multiplexer?

The multiplexer is a device that has multiple inputs and single line output. The select lines determine which input is connected to the output, and also increase the amount of data that can be sent over a network within a certain time. It is also called a data selector.

The single-pole multi-position switch is a simple example of a non-electronic circuit of the multiplexer, and it is widely used in many electronic circuits. The multiplexer is used to perform high-speed switching and is constructed by electronic components.



Multiplexer

Multiplexers are capable of handling both analog and digital applications. In analog applications, multiplexers are made up of relays and transistor switches, whereas in digital applications, the multiplexers are built from standard logic gates. When the multiplexer is used for digital applications, it is called a digital multiplexer.

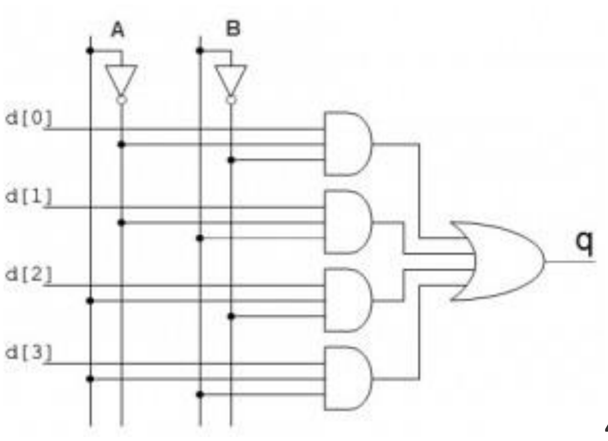
Multiplexer Types

Multiplexers are classified into four types:

- 2-1 multiplexer (1select line)
- 4-1 multiplexer (2 select lines)
- 8-1 multiplexer(3 select lines)
- 16-1 multiplexer (4 select lines)

4-to-1 Multiplexer

The 4X1 multiplexer comprises 4-input bits, 1- output bit, and 2- control bits. The four input bits are namely 0, D1, D2, and D3, respectively; only one of the input bits is transmitted to the output. The o/p ‘q’ depends on the value of control input AB. The control bit AB decides which of the i/p data bit should transmit the output. The following figure shows the 4X1 multiplexer circuit diagram using AND gates. For example, when the control bits AB =00, then the higher AND gates are allowed while remaining AND gates are restricted. Thus, data input D0 is transmitted to the output ‘q”

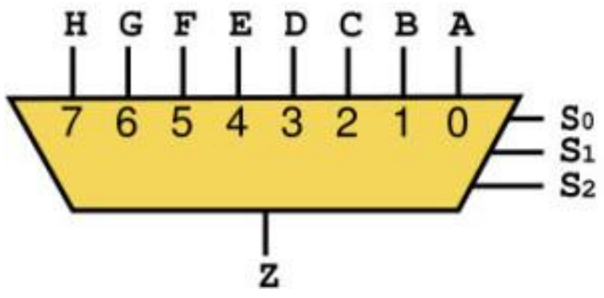


4X1 Mux

If the control input is changed to 11, then all gates are restricted except the bottom AND gate. In this case, D3 is transmitted to the output, and q=D0. If the control input is changed to AB =11, all gates are disabled except the bottom AND gate. In this case, D3 is transmitted to the output, and q = D3. The best example of a 4X1 multiplexer is IC 74153. In this IC, the o/p is the same as the i/p. Another example of a 4X1 multiplexer is IC 45352. In this IC, the o/p is the compliment of the i/p

8-to-1 Multiplexer

The 8-to-1 multiplexer consists of 8 input lines, one output line, and 3 selection lines.

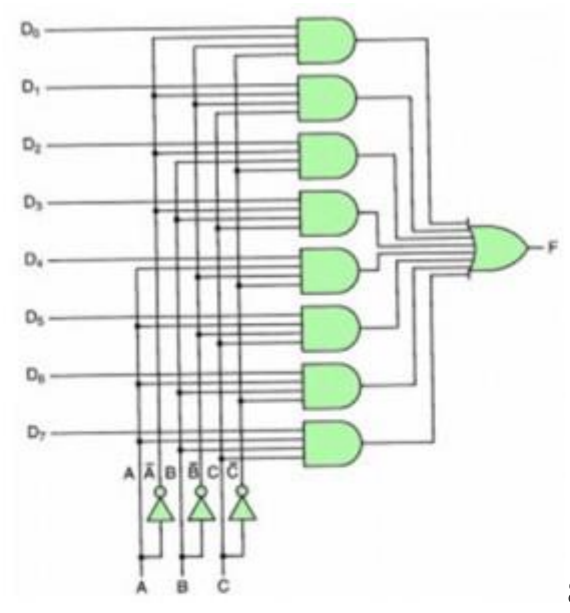


8-to-1 Mux

8-1 Multiplexer Circuit

For the combination of a selection input, the data line is connected to the output line. The circuit shown below is an 8*1 multiplexer. The 8-to-1 multiplexer requires 8 AND gates, one OR gate, and 3 selection lines. As an input, the combination of selection inputs is giving to the AND gate with the corresponding input data lines.

In a similar fashion, all the AND gates are given connection. In this 8*1 multiplexer, for any selection line input, one AND gate gives a value of 1 and the remaining all AND gates give 0. And, finally, by using OR gates, all the AND gates are added; and, this will be equal to the selected value.



8-to-1 Mux Circuit

Advantages and Disadvantages of Multiplexer

The **advantages of multiplexer** include the following.

- In multiplexer, the usage of a number of wires can be decreased

- It reduces the cost as well as the complexity of the circuit
- The implementation of a number of combination circuits can be possible by using a multiplexer
- Mux doesn’t require K-maps & simplification
- The multiplexer can make the transmission circuit less complex & economical
- The dissipation of heat is less because of the analog switching current which ranges from 10mA to 20mA.
- The multiplexer ability can be extended to switch audio signals, video signals, etc.
- The digital system reliability can be improved using a MUX as it decreases the number of exterior wired connections.
- MUX is used to implement several combinational circuits
- The logic design can be simplified through MUX

The **disadvantages of multiplexer** include the following.

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

Applications of Multiplexers

Multiplexers are used in various applications wherein multiple-data need to be transmitted by using a single line.

Communication System

A communication system has both a communication network and a transmission system. By using a multiplexer, the efficiency of the communication system can be increased by allowing the transmission of data, such as audio and video data from different channels through single lines or cables.

Computer Memory

Multiplexers are used in computer memory to maintain a huge amount of memory in the computers, and also to reduce the number of copper lines required to connect the memory to other parts of the computer.

Telephone Network

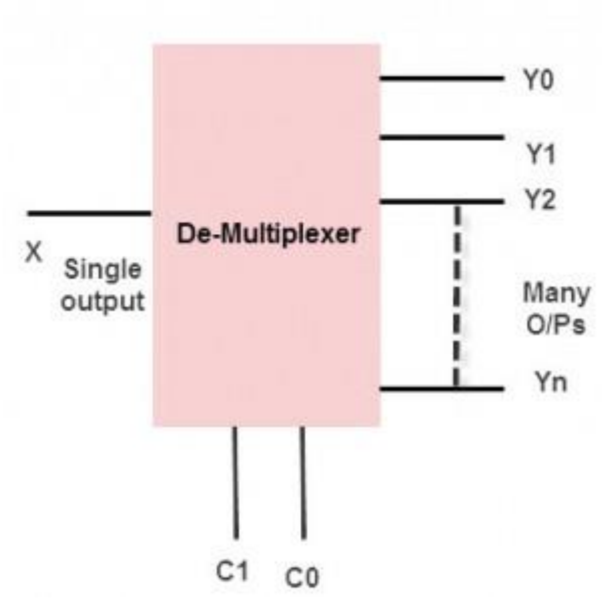
In telephone networks, multiple audio signals are integrated on a single line of transmission with the help of a multiplexer.

Transmission from the Computer System of a Satellite

The multiplexer is used to transmit the data signals from the computer system of a spacecraft or a satellite to the ground system by using a GSM satellite.

What is Demultiplexer?

De-multiplexer is also a device with one input and multiple output lines. It is used to send a signal to one of the many devices. The main difference between a multiplexer and a de-multiplexer is that a multiplexer takes two or more signals and encodes them on a wire, whereas a de-multiplexer does reverse to what the multiplexer does.



Demultiplexer

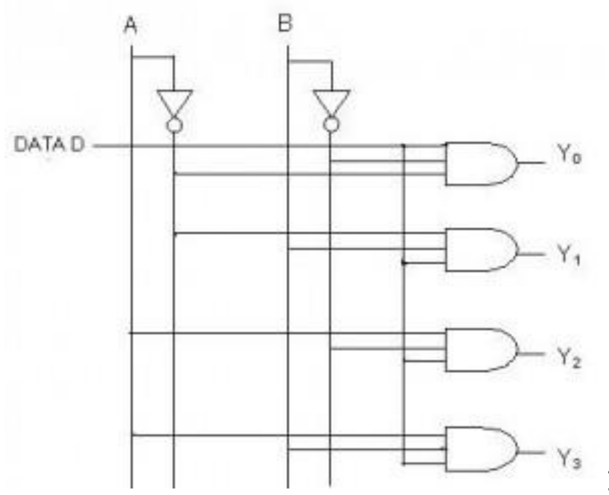
Types of Demultiplexer

Demultiplexers are classified into four types

- 1-2 demultiplexer (1 select line)
- 1-4 demultiplexer (2 select lines)
- 1-8 demultiplexer (3 select lines)
- 1-16 demultiplexer (4 select lines)

1-4 Demultiplexer

The 1-to-4 demultiplexer comprises 1- input bit, 4-output bits, and control bits. The 1X4 demultiplexer circuit diagram is shown below.



1X4 Demux

The i/p bit is considered as Data D. This data bit is transmitted to the data bit of the o/p lines, which depends on the AB value and the control i/p.

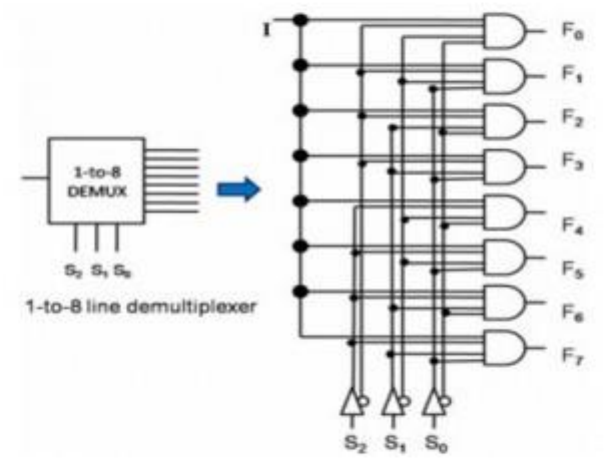
When the control i/p AB = 01, the upper second AND gate is permitted while the remaining AND gates are restricted. Thus, only data bit D is transmitted to the output, and Y1 = Data.

If the data bit D is low, the output Y1 is low. IF data bit D is high, the output Y1 is high. The value of the output Y1 depends upon the value of data bit D, the remaining outputs are in a low state.

If the control input changes to AB = 10, then all the gates are restricted except the third AND gate from the top. Then, data bit D is transmitted only to the output Y2; and, Y2 = Data. . The best example of 1X4 demultiplexer is IC 74155.

1-8 Demultiplexer

The demultiplexer is also called a data distributor as it requires one input, 3 selected lines, and 8 outputs. De-multiplexer takes one single input data line and then switches it to any one of the output lines. The 1-to-8 demultiplexer circuit diagram is shown below; it uses 8 AND gates for achieving the operation.



1-8 Demux Circuit

The input bit is considered as data D and it is transmitted to the output lines. This depends on the control input value of the AB. When AB = 01, the upper second gate F1 is enabled, while the remaining AND gates are disabled, and the data bit is transmitted to the output giving F1= data. If D is low, the F1 is low, and if D is high, the F1 is high. So the value of the F1 depends on the value of D, and the remaining outputs are in the low state.

Advantages and Disadvantages of Demultiplexer

The **advantages of demultiplexe**r include the following.

- A demultiplexer or Demux is used to divide the mutual signals back into separate streams.
- The function of Demux is quite opposite to MUX.
- The Audio or Video signals transmission needs a combination of Mux and Demux.
- Demux is used as a decoder within the security systems of banking sectors.
- The communication system efficiency can be enhanced through the combination of Mux & Demux.

The **disadvantages of demultiplexer** include the following.

- Bandwidth wastage might happen
- Because of the synchronization of the signals, delays might take place

Applications of Demultiplexer

Demultiplexers are used to connect a single source to multiple destinations. These applications include the following:

Communication System

Mux and demux both are used in communication systems to carry out the process of data transmission. A De-multiplexer receives the output signals from the multiplexer and at the receiver end, it converts them back to the original form.

Arithmetic Logic Unit
The output of the ALU is fed as an input to the De-multiplexer, and the output of the demultiplexer is connected to multiple registers. The output of the ALU can be stored in multiple registers.

Serial to Parallel Converter
This converter is used to reconstruct parallel data. In this technique, serial data is given as an input to the De-multiplexer at a regular interval, and a counter is attached to the demultiplexer at the control input to detect the data signal at the output of the demultiplexer. When all data signals are stored, the output of the demux can be read out in parallel.

Difference between Multiplexer and Demultiplexer

The main difference between multiplexer and demultiplexer is discussed below.

Multiplexer	**Demultiplexer**
A multiplexer (Mux) is a combinational circuit that uses several data inputs to generate a single output.	A demultiplexer (Demux) is also a combinational circuit that uses single input that can be directed throughout several outputs.
Multiplexer includes several inputs and the single output	Demultiplexer includes single input and several outputs
A multiplexer is a data selector	The demultiplexer is a data distributor
It is a digital switch	It is a digital circuit
It works on the principle of many to one	It works on the principle of one-to-many

The parallel to serial conversion is used in the multiplexer	The serial to parallel conversion is used in Demultiplexer
The multiplexer used in TDM (Time Division Multiplexing is at the end of the transmitter	The demultiplexer used in TDM (Time Division Multiplexing is at the end of the receiver
The multiplexer is called MUX	The demultiplexer is called Demux
It doesn’t use any extra gates while designing	In this, additional gates are necessary while designing demux
In Multiplexer, control signals are used to choose the specific input that has to be sent at the output.	Demultiplexer uses the control signal to permit us to include several outputs.
The multiplexer is used to improve the efficiency of the communication system using transmission data like transmission of audio as well as video.	Demultiplexer gets the o/p signals from the Mux & changed them to the unique form at the end of the receiver.
The different types of multiplexers are 8-1 MUX, 16-1 MUX, and 32-1 MUX.	The different types of demultiplexers are 1-8 Demux, 1-16 Demux, 1-32 Demux.
In multiplexer, the set of selection lines are used to control the specific input	In demultiplexer, the selection of output line can be controlled through n-selection lines bit values.

Key Difference between Multiplexer and Demultiplexer

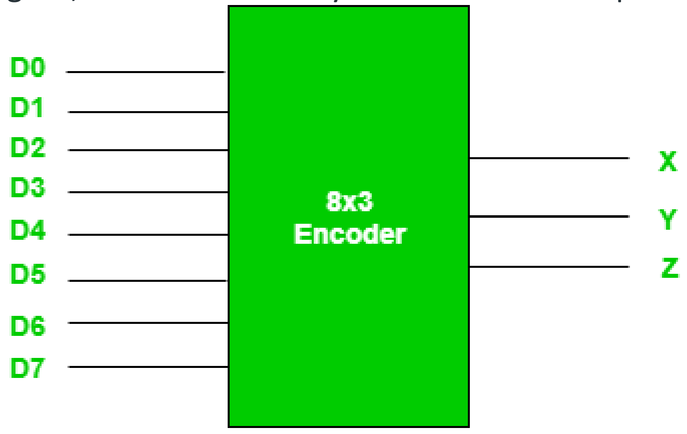
The key differences between multiplexer and demultiplexer are discussed below.

- The combinational logic circuits like multiplexer and demultiplexer are used within communication systems however their function is accurately opposite to each other because one works on multiple inputs whereas the other works on only input.
- Multiplexer or Mux is an N-to-1 device whereas demultiplexer is a 1-to-N device.

- A multiplexer is used to convert several analog or digital signals into a single o/p signal through different control lines. These control lines can be determined by using this formula like 2n=r where ‘r’ is the no of i/p signals & ‘n’ is the no of required control lines.
- The data conversion method used in MUX is parallel to serial and it is not difficult to understand because it uses different inputs. However, DEMUX works quite reverse to MUX like a serial to parallel conversion. So, the number of outputs can be achieved in this case.
- A demultiplexer is used to convert one i/p signal to several. The number of control signals can be determined by using the same formula of MUX.
- Both the Mux and Demux are used to transmit the data over a network in less bandwidth. But multiplexer is used at the transmitter end whereas the Demux is used at the receiver end.

Encoders and Decoders

Binary code of N digits can be used to store 2^N distinct elements of coded information. This is what encoders and decoders are used for. **Encoders** convert 2^N lines of input into a code of N bits and **Decoders** decode the N bits into 2^N lines.
1. Encoders –
An encoder is a combinational circuit that converts binary information in the form of a 2^N input lines into N output lines, which represent N bit code for the input. 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.

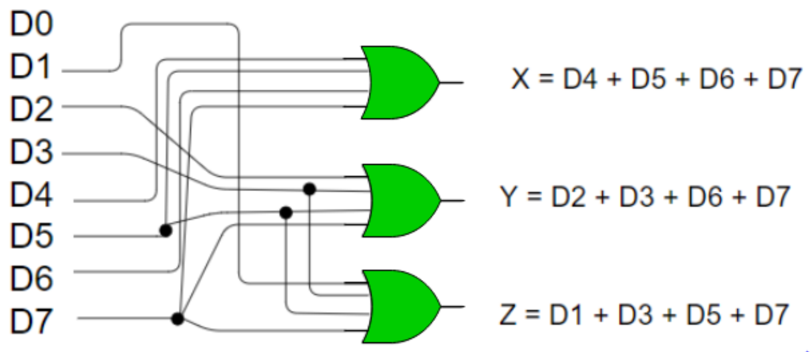
Implementation –
From the truth table, the output line Z is active when the input octal digit is 1, 3, 5 or 7. Similarly, Y is 1 when input octal digit is 2, 3, 6 or 7 and X is 1 for input octal digits 4, 5, 6 or 7. Hence, the Boolean functions would be:

$X = D4 + D5 + D6 + D7$

$Y = D2 + D3 + D6 + D7$

$Z = D1 + D3 + D5 + D7$

Hence, the encoder can be realised 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 encoder 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).

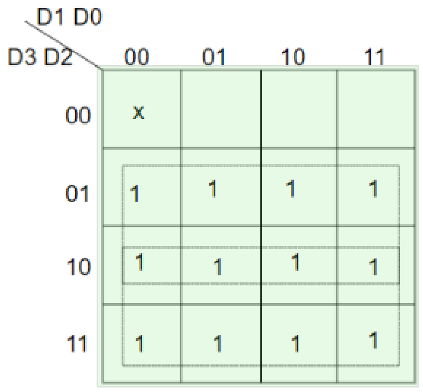
Truth Table –

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

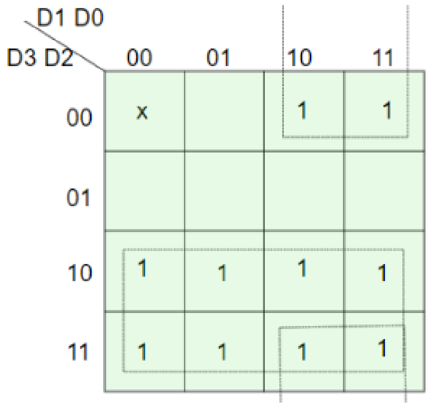
D3	D2	D1	D0	X	Y	V
0	1	x	x	1	0	1
1	x	x	x	1	1	1

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:

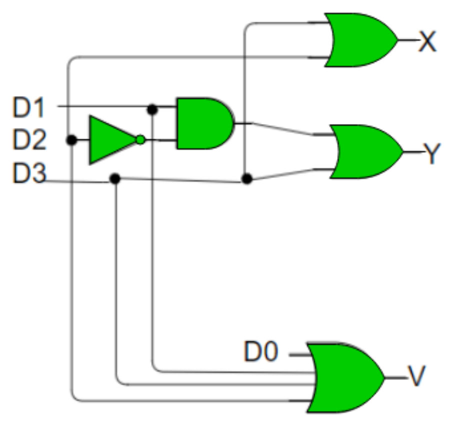


=> X = D2 + D3
For Y:



=> 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
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

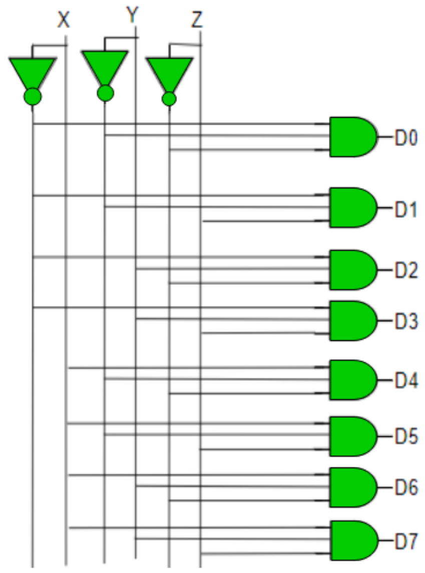
D4 = X Y' Z'

D5 = X Y' Z

D6 = X Y Z'

D7 = X Y Z

Hence,



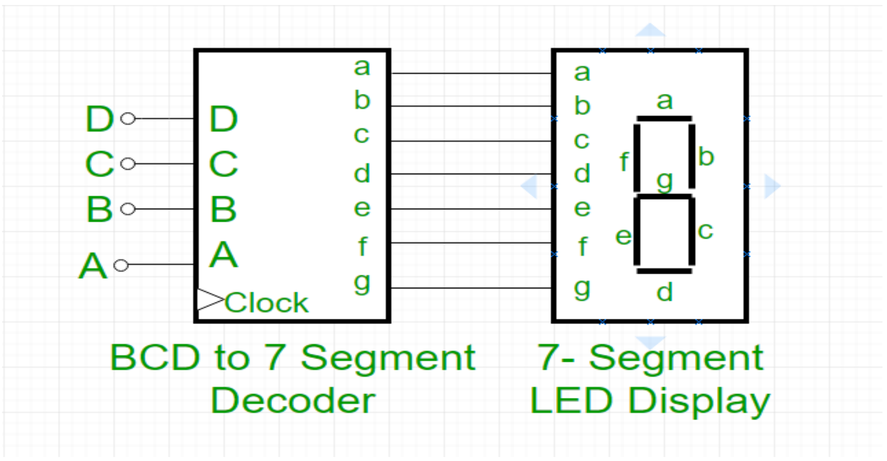
Seven segment decoder

In **Binary Coded Decimal (BCD)** encoding scheme each of the decimal numbers(0-9) is represented by its equivalent binary pattern(which is generally of 4-bits).
Whereas, **Seven segment** display is an electronic device which consists of seven Light Emitting Diodes (LEDs) arranged in a some definite pattern (common cathode or common anode type), which is used to display Hexadecimal numerals(in this case decimal numbers,as input is BCD i.e., 0-9).
Two types of seven segment LED display:

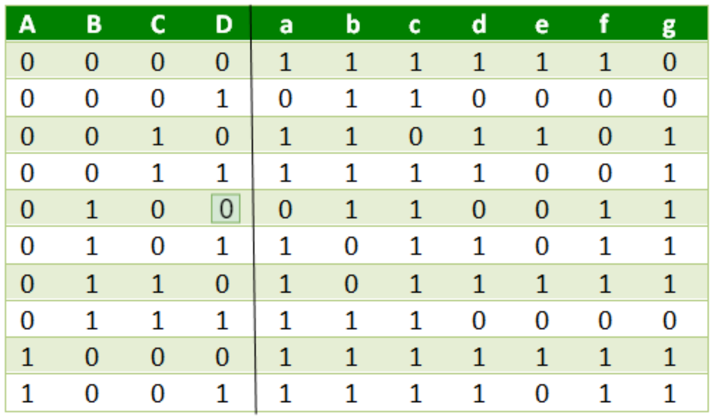
1. **Common Cathode Type:** In this type of display all cathodes of the seven LEDs are connected together to the ground or -Vcc(hence,common cathode) and LED displays digits when some 'HIGH' signal is supplied to the individual anodes.
2. **Common Anode Type:** In this type of display all the anodes of the seven LEDs are connected to battery or +Vcc and LED displays digits when some 'LOW' signal is supplied to the individual cathodes.

But, seven segment display does not work by directly supplying voltage to different segments of LEDs. First, our decimal number is changed to its BCD equivalent signal then BCD to seven segment decoder converts that signals to the form which is fed to seven segment display.

This BCD to seven segment decoder has four input lines (A, B, C and D) and 7 output lines (a, b, c, d, e, f and g), this output is given to seven segment LED display which displays the decimal number depending upon inputs.



Truth Table – For common cathode type BCD to seven segment decoder:

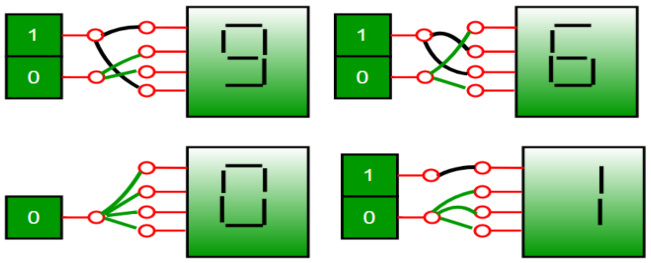


A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Note –

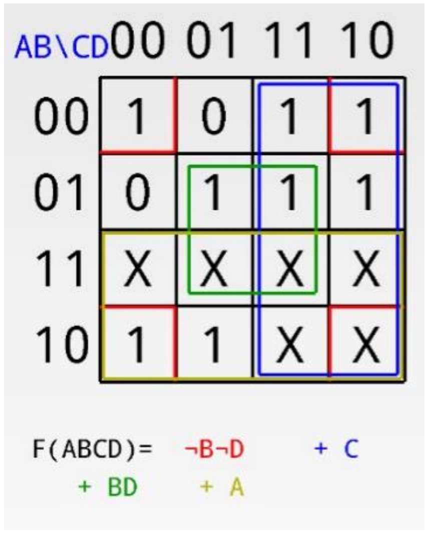
- For Common Anode type seven segment LED display, we only have to interchange all ‘0s’ and ‘1s’ in the output side i.e., (for a, b, c, d, e, f, and g replace all ‘1’ by ‘0’ and vice versa) and solve using K-map.
- Output for first combination of inputs (A, B, C and D) in Truth Table corresponds to ‘0’ and last combination corresponds to ‘9’. Similarly rest corresponds from 2 to 8 from top to bottom.
- BCD numbers only range from 0 to 9,thus rest inputs from 10-F are invalid inputs.

Example –



Explanation –
For combination where all the inputs (A, B, C and D) are zero (see Truth Table), our output lines are a = 1, b = 1, c = 1, d = 1, e = 1, f = 1 and g = 0. So 7 segment display shows 'zero' as output.
Similarly, for combination where one of the input is one (D = 1) and rest are zero,our output lines are a = 0, b = 1, c = 1, d = 0, e = 0, f = 0 and g = 0. So only LEDs 'b' and 'c' (see diagram above) will glow and 7 segment display shows 'one' as output.

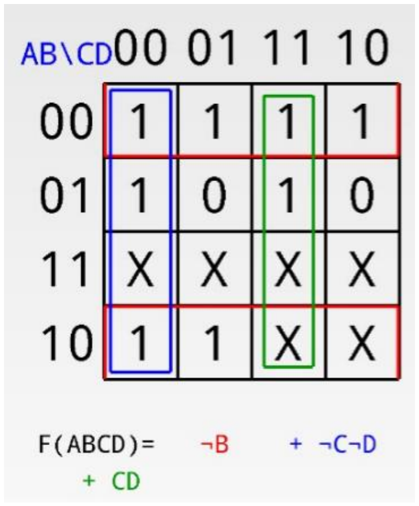
K-Maps:
#for a:



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

F(ABCD)= ¬B¬D + C + BD + A

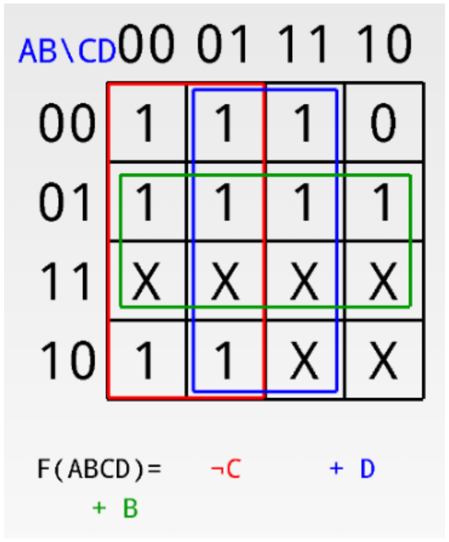
#for b:



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

F(ABCD)= ¬B + ¬C¬D + CD

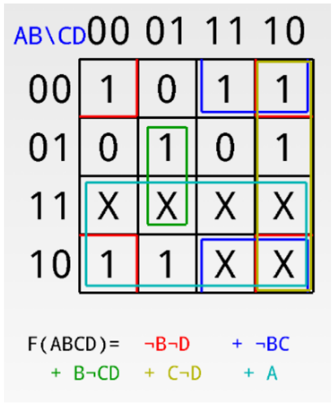
#for c:



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

F(ABCD)= ¬C + D + B

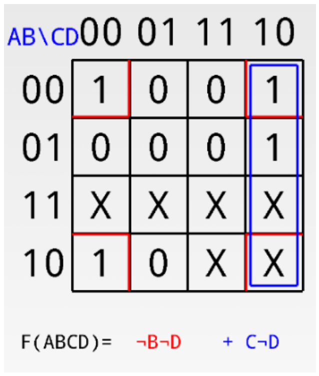
#for d:



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

F(ABCD)= ¬B¬D + ¬BC + B¬CD + C¬D + A

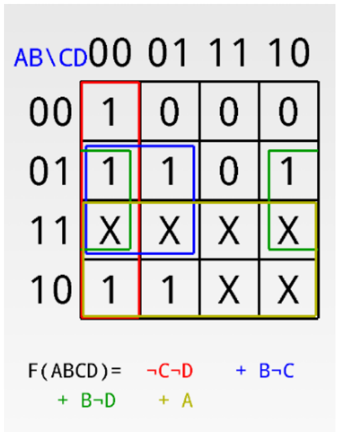
#for e:



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

F(ABCD)= ¬B¬D + C¬D

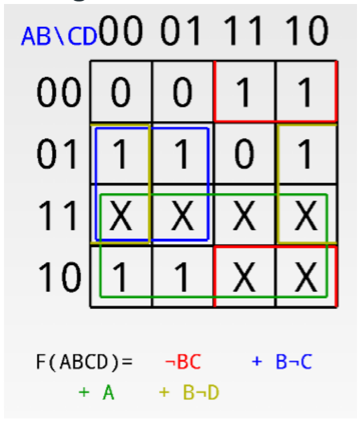
#for f:



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

F(ABCD)= ¬C¬D + B¬C + B¬D + A

#for g:



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

F(ABCD)= ¬BC + B¬C + A + B¬D

Applications –
Seven-segment displays are used to display the digits in calculators, clocks, various measuring instruments, digital watches and digital counters.