Digital Design and Computer Organization

Module 2:

Data Processing Circuits: Multiplexers, Demultiplexers,1of,16 Decoder, BDC to decimal Decoders, Seven segment Decoders, Encoders, Exclusive OR Gates, Parity Generators and Checkers.

Arithmetic Circuits: Binary Addition, Binary Subtraction, 2'S Complement Representation, 2'S Complement Arithmetic, Arithmetic Building Blocks.

Introduction to Sequential Circuits – Flip Flops, operation and excitation tables, Triggering of FF, Analysis and design of clocked sequential circuits

Data Processing Circuit

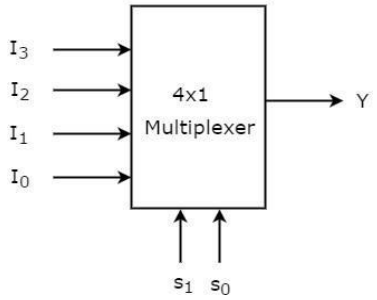
Multiplexer

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

Since there are 'n' selection lines, there will be 2^n possible combinations of zeros andones. So, each combination will select only one data input. Multiplexer is also calledas **Mux**.

4x1 Multiplexer

4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & s_0 and oneoutput Y. The **block diagram** of 4x1 Multiplexer is shown in the following figure.



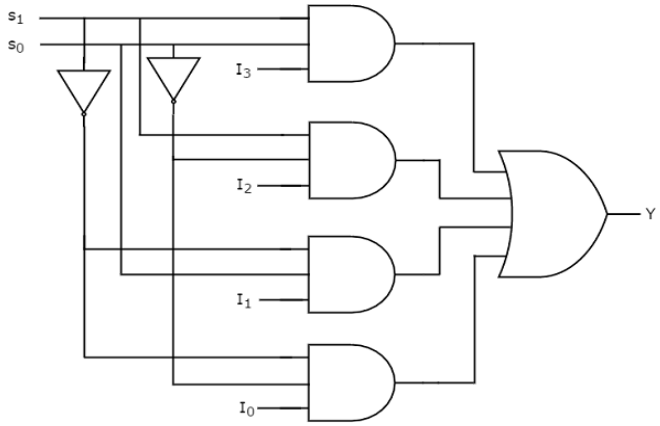
One of these 4 inputs will be connected to the output based on the combination of inputspresent at these two selection lines. **Truth table** of 4x1 Multiplexer is shown below.

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

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

Y={S{1}}'{S{0}}'I{0}+{S{1}}'S{0}I{1}+S{1}{S{0}}'I{2}+S{1}S{0}I{3}

We can implement this Boolean function using Inverters, AND gates & OR gate.The **circuit diagram** of 4x1 multiplexer is shown in the following figure.



We can easily understand the operation of the above circuit. Similarly, you can implement8x1 Multiplexer and 16x1 multiplexer by following the same procedure.

Implementation of Higher-order Multiplexers.

Now, let us implement the following two higher-order Multiplexers using lower-orderMultiplexers.

- 8x1 Multiplexer
- 16x1 Multiplexer

8x1 Multiplexer

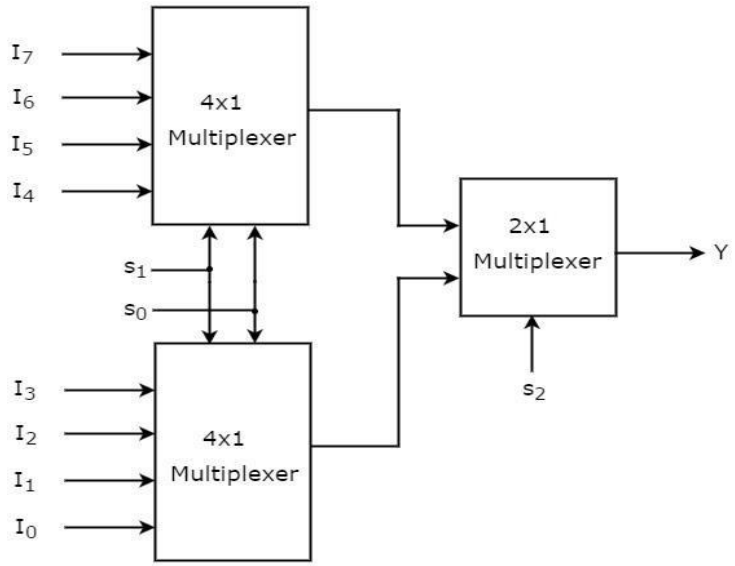
In this section, let us implement 8x1 Multiplexer using 4x1 Multiplexers and 2x1 Multiplexer. We know that 4x1 Multiplexer has 4 data inputs, 2 selection lines and one output. Whereas, 8x1 Multiplexer has 8 data inputs, 3 selection lines and one output.

So, we require two **4x1 Multiplexers** in first stage in order to get the 8 data inputs. Since, each 4x1 Multiplexer produces one output, we require a **2x1 Multiplexer** in second stage byconsidering the outputs of first stage as inputs and to produce the final output.

Let the 8x1 Multiplexer has eight data inputs I_7 to I_0, three selection lines s_2, s_1 & s0 and one output Y. The **Truth table** of 8x1 Multiplexer is shown below.

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

We can implement 8x1 Multiplexer using lower order Multiplexers easily by considering theabove Truth table. The **block diagram** of 8x1 Multiplexer is shown in the following figure.



The same **selection lines, s_1 & s_0** are applied to both 4x1 Multiplexers. The data inputs of upper 4x1 Multiplexer are I_7 to I_4 and the data inputs of lower 4x1 Multiplexer are I_3 to I_0. Therefore, each 4x1 Multiplexer produces an output based on the values of selection lines, s_1 & s_0.

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

- If s_2 is zero, then the output of 2x1 Multiplexer will be one of the 4 inputs I_3 toI_0 based on the values of selection lines s_1 & s_0.
- If s_2 is one, then the output of 2x1 Multiplexer will be one of the 4 inputs I_7 toI_4 based on the values of selection lines s_1 & s_0.

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

16x1 Multiplexer

In this section, let us implement 16x1 Multiplexer using 8x1 Multiplexers and 2x1 Multiplexer. We know that 8x1 Multiplexer has 8 data inputs, 3 selection lines and one output. Whereas, 16x1 Multiplexer has 16 data inputs, 4 selection lines and one output.

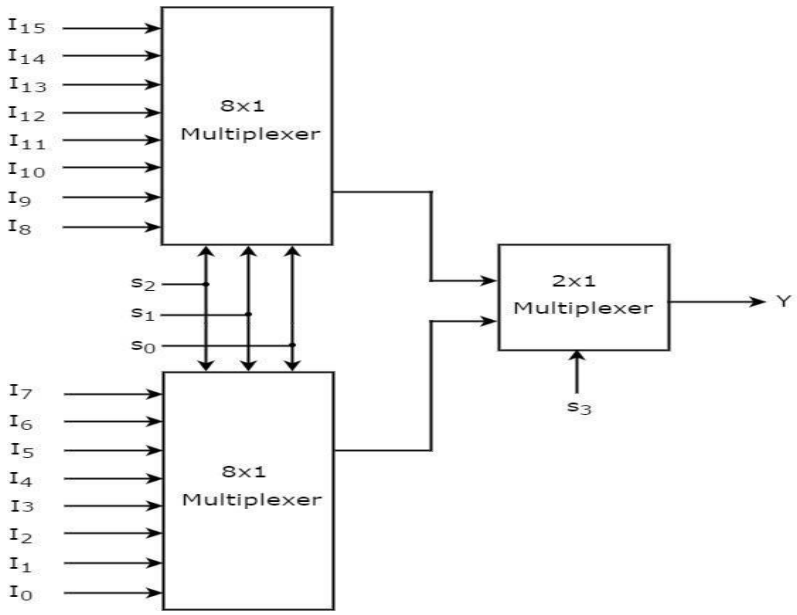
So, we require two **8x1 Multiplexers** in first stage in order to get the 16 data inputs. Since, each 8x1 Multiplexer produces one output, we require a 2x1 Multiplexer in second stage by considering the outputs of first stage as inputs and to produce the final output.

Let the 16x1 Multiplexer has sixteen data inputs I_{15} to I_0, four selection lines s_3 to s_0 and one output Y. The **Truth table** of 16x1 Multiplexer is shown below.

We can implement 16x1 Multiplexer using lower order Multiplexers easily by considering the below Truth table.

Selection Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}
1	1	0	1	I_{13}
1	1	0	1	I_{14}
1	1	1	1	I_{15}

The **block diagram** of 16x1 Multiplexer is shown in the following figure.



The **same selection lines, s_2, s_1 & s_0** are applied to both 8x1 Multiplexers. The data inputs ofupper 8x1 Multiplexer are I_{15} to I_8 and the data inputs of lower 8x1 Multiplexer are I_7 to I_0. Therefore, each 8x1

Multiplexer produces an output based on the values of selection lines, s_2, s_1 & s_0.

The outputs of first stage 8x1 Multiplexers are applied as inputs of 2x1 Multiplexer that is present in second stage. The other **selection line, s_3** is applied to 2x1 Multiplexer.

- If s_3 is zero, then the output of 2x1 Multiplexer will be one of the 8 inputs Is_7 toI_0 based on the values of selection lines s_2, s_1 & s_0.
- If s_3 is one, then the output of 2x1 Multiplexer will be one of the 8 inputs I_{15} toI_8 based on the values of selection lines s_2, s_1 & s_0.

Therefore, the overall combination of two 8x1 Multiplexers and one 2x1 Multiplexer performs as one 16x1 Multiplexer.

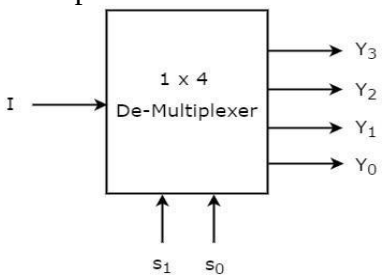
De-Multiplexer

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

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

1x4 De-Multiplexer

1x4 De-Multiplexer has one input I, two selection lines, s_1 & s_0 and four outputs Y_3,Y_2, Y_1 &Y_0. The **block diagram** of 1x4 De-Multiplexer is shown in the following figure.



The single input 'I' will be connected to one of the four outputs, Y_3 to Y_0 based on thevalues of selection lines s_1 & s0. The **Truth table** of 1x4 De-Multiplexer is shown below.

Selection Inputs		**Outputs**			
S_1	**S_0**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	0	0	0	0	**I**
0	1	0	0	**I**	0
1	0	0	**I**	0	0
1	1	**I**	0	0	0

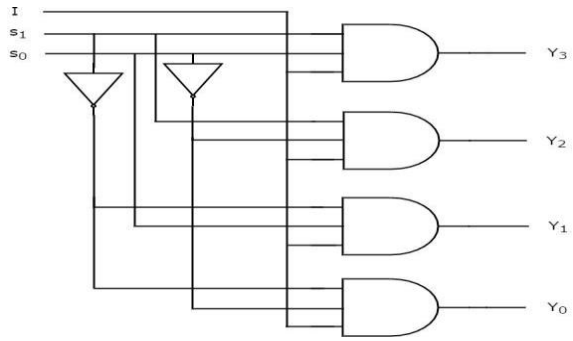
From the above Truth table, we can directly write the **Boolean functions** for each output asY{3}=s{1}s{0}I

Y{2}=s{1}{s{0}}'I

Y{1}={s{1}}'s{0}I

Y{0}={s1}'{s{0}}'I

We can implement these Boolean functions using Inverters & 3-input AND gates.The **circuit diagram** of 1x4 De-Multiplexer is shown in the following figure.



We can easily understand the operation of the above circuit. Similarly, you can implement1x8 De-Multiplexer and 1x16 De-Multiplexer by following the same procedure.

Implementation of Higher-order De-Multiplexers

Now, let us implement the following two higher-order De-Multiplexers using lower-orderDe-Multiplexers.

- 1x8 De-Multiplexer
- 1x16 De-Multiplexer

1x8 De-Multiplexer

In this section, let us implement 1x8 De-Multiplexer using 1x4 De-Multiplexers and 1x2 De-Multiplexer. We know that 1x4 De-Multiplexer has single input, two selection lines and four outputs. Whereas, 1x8 De-Multiplexer has single input, three selection lines and eight outputs.

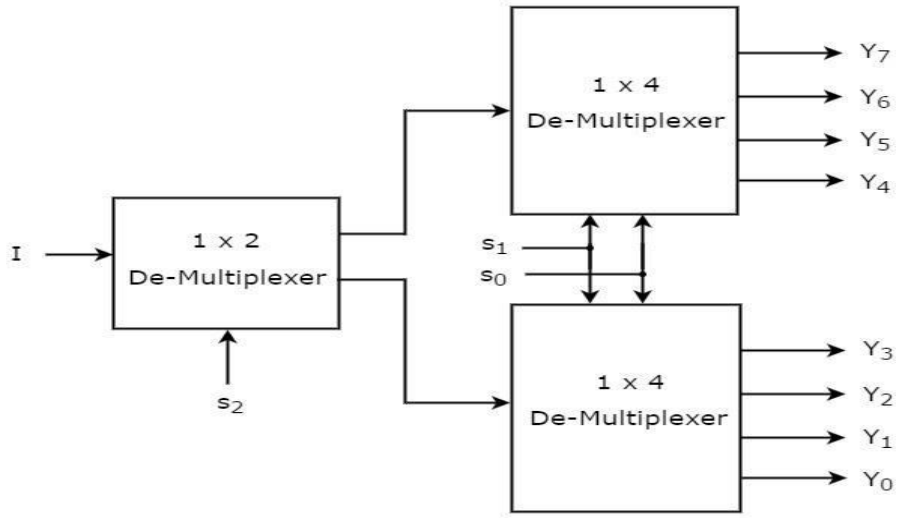
So, we require two **1x4 De-Multiplexers** in second stage in order to get the final eight outputs. Since, the number of inputs in second stage is two, we require **1x2 DeMultiplexer** in first stage so that the outputs of first stage will be the inputs of second stage. Input of this 1x2 De-Multiplexer will be the overall input of 1x8 De-Multiplexer.

Let the 1x8 De-Multiplexer has one input I, three selection lines s_2, s_1 & s_0 and outputs Y_7 to Y_0. The **Truth table** of 1x8 De-Multiplexer is shown below.

Selection Inputs			**Outputs**							
s_2	**s_1**	**s_0**	**Y_7**	**Y_6**	**Y_5**	**Y_4**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	0	0	0	0	0	0	0	0	0	**I**
0	0	1	0	0	0	0	0	0	**I**	0
0	1	0	0	0	0	0	0	**I**	0	0
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

We can implement 1x8 De-Multiplexer using lower order Multiplexers easily by consideringthe above Truth table. The **block diagram** of 1x8 De-Multiplexer is shown in the following figure.



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

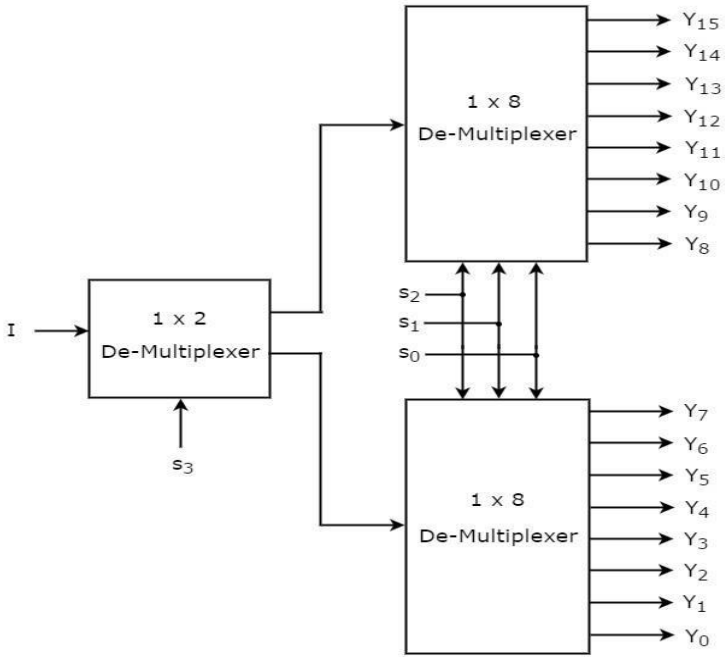
The other **selection line, s_2** is applied to 1x2 De-Multiplexer. If s_2 is zero, then one of the four outputs of lower 1x4 De-Multiplexer will be equal to input, I based on the values of selection lines s_1 & s_0. Similarly, if s_2 is one, then one of the four outputs of upper 1x4 DeMultiplexer will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

1x16 De-Multiplexer

In this section, let us implement 1x16 De-Multiplexer using 1x8 De-Multiplexers and1x2 De-Multiplexer. We know that 1x8 De-Multiplexer has single input, three selection lines and eight outputs. Whereas, 1x16 De-Multiplexer has single input, four selection lines and sixteen outputs.

So, we require two **1x8 De-Multiplexers** in second stage in order to get the final sixteen outputs. Since, the number of inputs in second stage is two, we require **1x2 DeMultiplexer** in first stage so that the outputs of first stage will be the inputs of second stage. Input of this 1x2 De-Multiplexer will be the overall input of 1x16 De-Multiplexer.

Let the 1x16 De-Multiplexer has one input I, four selection lines s_3, s_2, s_1 & s_0 and outputs Y_{15} to Y_0. The **block diagram** of 1x16 De-Multiplexer using lower order Multiplexers is shown in the following figure.



The common **selection lines s_2, s_1 & s_0** are applied to both 1x8 De-Multiplexers. The outputs of upper 1x8 De-Multiplexer are Y_{15} to Y_8 and the outputs of lower 1x8 DeMultiplexer are Y_7 to Y_0.

The other **selection line, s_3** is applied to 1x2 De-Multiplexer. If s_3 is zero, then one of the eight outputs of lower 1x8 De-Multiplexer will be equal to input, I based on the values of selection lines s_2, s_1 & s_0. Similarly, if s3 is one, then one of the 8 outputs of upper 1x8 De- Multiplexer will be equal to input, I based on the values of selection lines s_2, s_1 & s_0.

Decoder

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

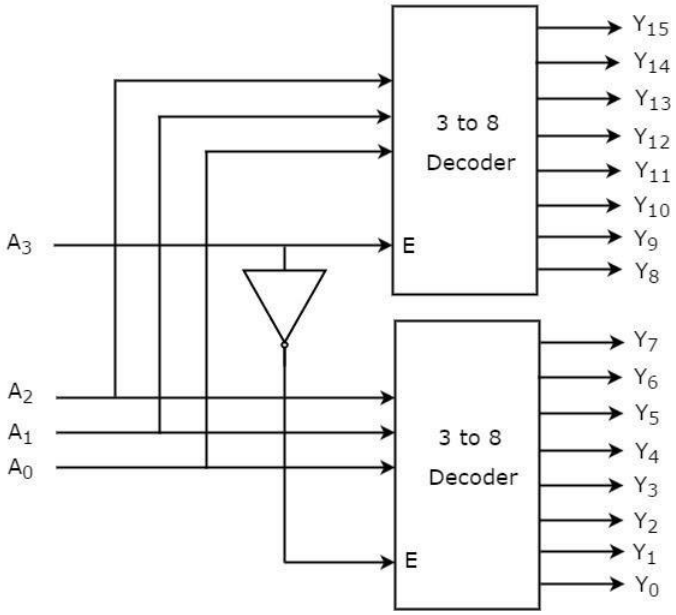
1 of 16 Decoder

In this section, let us implement **4 to 16 decoder using 3 to 8 decoders**. We know that 3 to 8 Decoder has three inputs A_2, A_1 & A_0 and eight outputs, Y_7 to Y_0. Whereas, 4 to 16 Decoder has four inputs A_3, A_2, A_1 & A_0 and sixteen outputs, Y_{15} to Y_0

We know the following formula for finding the number of lower order decoders required.

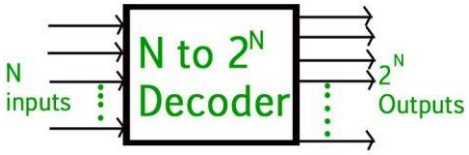
Therefore, we require two 3 to 8 decoders for implementing one 4 to 16 decoder.

The parallel inputs A_2, A_1 & A_0 are applied to each 3 to 8 decoder. The complement of input, A3 is connected to Enable, E of lower 3 to 8 decoder in order to get the outputs, Y_7 to Y_0. These are the **lower eight min terms**. The input, A_3 is directly connected to Enable, E ofupper 3 to 8 decoder in order to get the outputs, Y_{15} to Y_8. These are the **higher eight min terms**. The **blockdiagram** of 4 to 16 decoder using 3 to 8 decoders is shown in the following figure.



BCD to Decimal Decoder

In Digital Electronics, discrete quantities of information are represented by binary codes. A binary code of **n bits** is capable of representing up to **2^n distinct elements** of codedinformation. The name **"Decoder"** means to translate or decode coded information from one format into another, so a digital decoder transforms a set of digital input signals into an equivalent decimal code at its output. A **decoder** is a **combinational circuit** that converts binary information from **n input lines** to a maximum of **2^n unique output lines**.



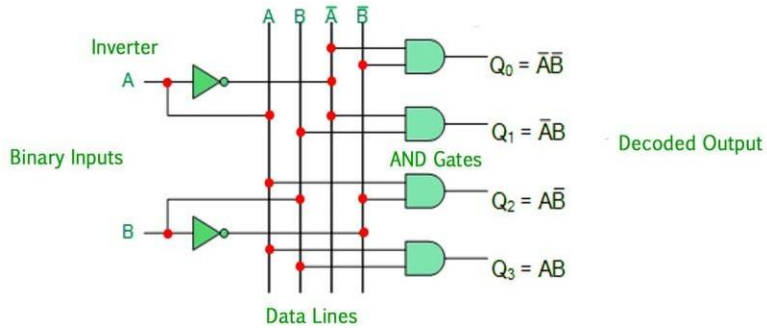
Binary Decoder

- Binary Decoders are another type of digital logic device that has inputs of 2-bit, 3-bitor 4-bit codes depending upon the number of data input lines, so a decoder that has a setof two or more bits will be defined as having an n-bit code, and therefore it will be possible to represent 2^n possible values.
- If a binary decoder receives n inputs it activates one and only one of its 2^n outputs based on that input with all other outputs deactivated. If the n -bit coded information hasunused combinations, the decoder may have fewer than 2^n outputs.
- Example, an inverter (NOT-gate) can be classified as a 1-to-2 binary decoder as 1- input and 2-outputs is possible. i.e an input A can give either A or A complement as theoutput.
- Then we can say that a standard combinational logic decoder is an n-to-m decoder,where m <= 2^n, and whose output, Q is dependent only on its present input states.
- Their purpose is to generate the 2^n (or fewer) minterms of n input variables. Eachcombination of inputs will assert a unique output.

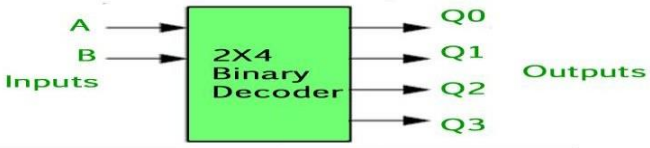
A Binary Decoder converts coded inputs into coded outputs, where the input and output codesare different and decoders are available to "decode" either a Binary or BCD (8421 code) inputpattern to typically a Decimal output code.

Practical "binary decoder" circuits include 2-to-4, 3-to-8 and 4-to-16 line configurations.

2-to-4 Binary Decoder



The 2-to-4 line binary decoder depicted above consists of an array of four AND gates. The 2 binary inputs labeled A and B are decoded into one of 4 outputs, hence the description ofa 2-to-4 binary decoder. Each output represents one of the minterms of the 2 input variables, (each output = a minterm).



A	B	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

The output values will be:

Q_0 = A'B'
Q_1 = A'B
Q_2 = AB'
Q_3 = AB

The binary inputs A and B determine which output line from Q0 to Q3 is "HIGH" at logic level "1" while the remaining outputs are held "LOW" at logic "0" so only one output canbe active (HIGH) at any one time. Therefore, whichever output line is "HIGH" identifies the binary code present at the input, in other words, it "decodes" the binary input.

Some binary decoders have an additional input pin labeled "Enable" that controls the outputs from the device. This extra input allows the outputs of the decoder to be turned "ON" or "OFF" as required. The output is only generated when the Enable input has value 1; otherwise, all outputs are 0. Only a small change in the implementation is required: the Enableinput is fed into the AND gates which produce the outputs.

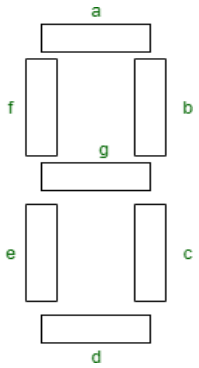
If Enable is 0, all AND gates are supplied with one of the inputs as 0 and hence no output is produced. When Enable is 1, the AND gates get one of the inputs as 1, and now theoutput depends upon the remaining inputs. Hence the output of the decoder is dependent on whether the Enable is high or low.

Seven Segment Decoder

Light Emitting Diode (LED) is the most widely used semiconductor which emits eithervisible light or invisible infrared light when forward biased. Remote controls generate invisiblelight. A Light emitting diode (LED) is an optical electrical energy into light energy when voltage is applied.

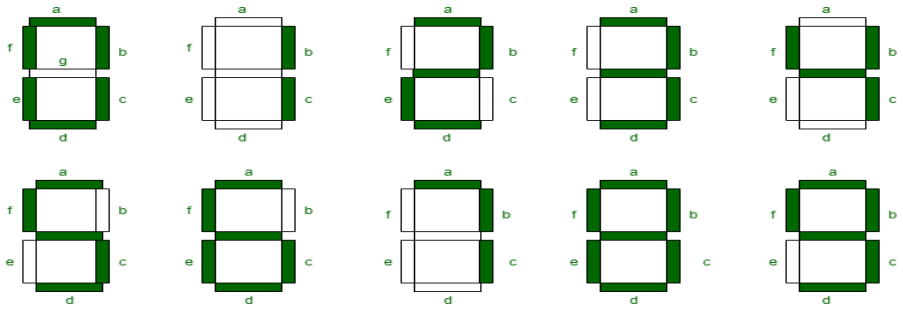
Seven Segment Displays

Seven segment displays are the output display device that provide a way to display information in the form of image or text or decimal numbers which is an alternative to the morecomplex dot matrix displays. It is widely used in digital clocks, basic calculators, electronic meters, and other electronic devices that display numerical information. It consists of seven segments of light emitting diodes (LEDs) which is assembled like numerical 8.



Working of Seven Segment Displays

The number 8 is displayed when the power is given to all the segments and if you disconnect the power for ‘g’, then it displays number 0. In a seven segment display, power (orvoltage) at different pins can be applied at the same time, so we can form combinations of display numerical from 0 to 9. Since seven segment displays can not form alphabet like X andZ, so it can not be used for alphabet and it can be used only for displaying decimal numerical magnitudes. However, seven segment displays can form alphabets A, B, C, D, E, and F, so theycan also used for representing hexadecimal digits.



We can produce a truth table for each decimal digit

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

Therefore, Boolean expressions for each decimal digit which requires respective lightemitting diodes (LEDs) are ON or OFF. The number of segments used by digit: 0, 1, 2, 3, 4, 5,6, 7, 8, and 9 are 6, 2, 5, 5, 4, 5, 6, 3, 7, and 6 respectively. Seven segment displays must be controlled by other external devices where different types of microcontrollers are useful to communicate with these external devices, like switches, keypads, and memory.

Types of Seven Segment Displays

According to the type of application, there are two types of configurations of sevensegment displays: common anode display and common cathode display.

1. In common cathode seven segment displays, all the cathode connections of LED segments are connected together to logic 0 or ground. We use logic 1 through a currentlimiting resistor to forward bias the individual anode terminals a to g.
2. Whereas all the anode connections of the LED segments are connected together tologic 1 in common anode seven segment display. We use logic 0 through a current limiting resistor to the cathode of a particular segment a to g.

Common anode seven segment displays are more popular than cathode seven segment displays,because logic circuits can sink more current than they can source and it is the same as connecting LEDs in reverse.

Applications of Seven Segment Displays

Common applications of seven segment displays are in:

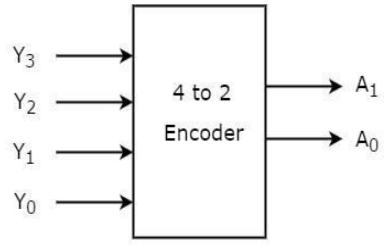
1. Digital clocks
2. Clock radios
3. Calculators
4. Wristwatchers
5. Speedometers
6. Motor-vehicle odometers
7. Radio frequency indicators

Encoder

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

4 to 2 Encoder

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



At any time, only one of these 4 inputs can be '1' in order to get the respective binary codeat the output.
The **Truth table** of 4 to 2 encoder is shown below.

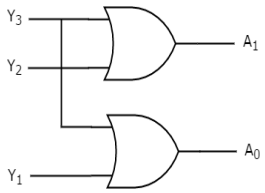
Inputs				**Outputs**	
Y_3	**Y_2**	**Y_1**	**Y_0**	**A_1**	**A_0**
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

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

$$A\{1\}=Y\{3\}+Y\{2\}$$

$$A\{0\}=Y\{3\}+Y\{1\}$$

We can implement the above two Boolean functions by using two input OR gates. The **circuit diagram** of 4 to 2 encoder is shown in the following figure.



The above circuit diagram contains two OR gates. These OR gates encode the four inputs with two bits

Exclusive-OR gate

The full form of Ex-OR gate is **Exclusive-OR** gate. Its function is same as that of ORgate except for some cases, when the inputs having even number of ones.

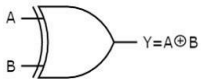
The following table shows the **truth table** of 2-input Ex-OR gate.

A	B	Y = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Here A, B are the inputs and Y is the output of two input Ex-OR gate. The truth table of Ex- OR gate is same as that of OR gate for first three rows. The only modification is in the fourth row. That means, the output (Y) is zero instead of one, when both the inputs are one, since the inputs having even number of ones.

Therefore, the output of Ex-OR gate is '1', when only one of the two inputs is '1'. And it is zero, when both inputs are same.

Below figure shows the **symbol** of Ex-OR gate, which is having two inputs A, B and one output, Y.



Ex-OR gate operation is similar to that of OR gate, except for few combination(s) of inputs. That's why the Ex-OR gate symbol is represented like that. The output of Ex-OR gate is '1',when odd number of ones present at the inputs. Hence, the output of Ex-OR gate is also called as an **odd function**.

Parity Bit Generator

There are two types of parity bit generators based on the type of parity bit being generated. **Even parity generator** generates an even parity bit. Similarly, **odd parity generator** generates an odd parity bit.

Even Parity Generator

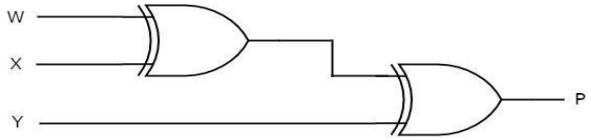
Now, let us implement an even parity generator for a 3-bit binary input, WXY. It generates an even parity bit, P. If odd number of ones present in the input, then even parity bit, P should be '1' so that the resultant word contains even number of ones. For other combinations of input, even parity bit, P should be '0'. The following table shows the **Truthtable** of even parity generator.

Binary Input WXY	Even Parity bit P
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

From the above Truth table, we can write the **Boolean function** for even parity bit as

$$P=\{W\}'\{X\}'Y+\{W\}'X\{Y\}'+W\{X\}'\{Y\}'+WXY$$

The following figure shows the **circuit diagram** of even parity generator.

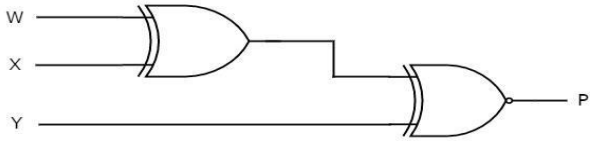


This circuit consists of two **Exclusive-OR gates** having two inputs each. First ExclusiveOR gate having two inputs W & X and produces an output W ⊕ X. This output is given as one input of second Exclusive-OR gate. The other input of this second Exclusive-OR gate is Y and produces an output of W ⊕ X ⊕ Y.

Odd Parity Generator

If even number of ones present in the input, then odd parity bit, P should be '1' so that the resultant word contains odd number of ones. For other combinations of input, odd parity bit, P should be '0'.

Follow the same procedure of even parity generator for implementing odd parity generator. The **circuit diagram** of odd parity generator is shown in the following figure.



The above circuit diagram consists of Ex-OR gate in first level and Ex-NOR gate in second level. Since the odd parity is just opposite to even parity, we can place an inverter at the output of even parity generator. In that case, the first and second levels contain an ExORgate in each level and third level consist of an inverter.

Parity Checker

There are two types of parity checkers based on the type of parity has to be checked. **Even parity checker** checks error in the transmitted data, which contains message bits along with even parity. Similarly, **odd parity checker** checks error in the transmitted data, which contains message bits along with odd parity.

Even parity checker

Now, let us implement an even parity checker circuit. Assume a 3-bit binary input, WXY is transmitted along with an even parity bit, P. So, the resultant word (data) contains 4bits, which will be received as the input of even parity checker.

It generates an **even parity check bit, E**. This bit will be zero, if the received data contains an even number of ones. That means, there is no error in the received data. This even parity check bit will be one, if the received data contains an odd number of ones. That means, thereis an error in the received data.

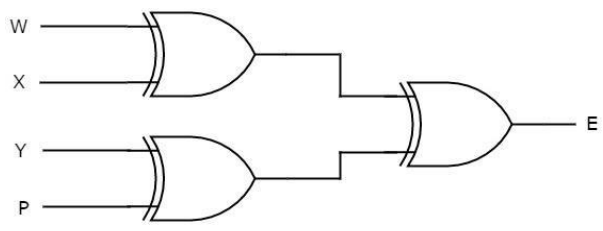
The following table shows the **Truth table** of an even parity checker.

4-bit Received Data WXYP	Even Parity Check bit E
0000	0
0001	1
0010	1
0011	0
0100	1
0101	0
0110	0
0111	1
1000	1
1001	0
1010	0
1011	1
1100	0
1101	1
1110	1
1111	0

From the above Truth table, we can observe that the even parity check bit value is '1', when odd number of ones present in the received data. That means the Boolean function of even parity check bit is an **odd function**. Exclusive-OR function satisfies this condition. Hence, we can directly write the **Boolean function** of even parity check bit as

$$E=W+X+Y+P$$

The following figure shows the **circuit diagram** of even parity checker.



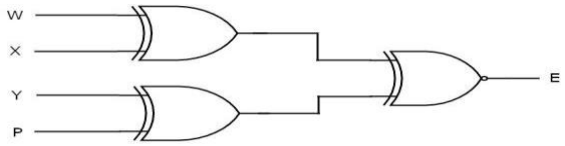
This circuit consists of three **Exclusive-OR gates** having two inputs each. The first level gates produce outputs of W + X & Y + P. The Exclusive-OR gate, which is in second level produces an output of W + X + Y + P.

Odd Parity Checker

Assume a 3-bit binary input, WXY is transmitted along with odd parity bit, P. So, theresultant word (data) contains 4 bits, which will be received as the input of odd parity checker.

It generates an **odd parity check bit, E**. This bit will be zero, if the received data containsan odd number of ones. That means, there is no error in the received data. This odd parity check bit will be one, if the received data contains even number of ones. That means, thereis an error in the received data.

Follow the same procedure of an even parity checker for implementing an odd paritychecker. The **circuit diagram** of odd parity checker is shown in the following figure.



The above circuit diagram consists of Ex-OR gates in first level and Ex-NOR gate in second level. Since the odd parity is just opposite to even parity, we can place an inverter at the output of even parity checker. In that case, the first, second and third levels contain two Ex-OR gates, one Ex-OR gate and one inverter respectively.

ARITHMETIC CIRCUITS

Binary Addition

It is a key for binary subtraction, multiplication, division. There are four rules ofbinary addition.

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

In fourth case, a binary addition is creating a sum of (1 + 1 = 10) i.e. 0 is written in the given column and a carry of 1 over to the next column.

Example − Addition

0011010 + 001100 = 00100110

```
  1 1          carry
  0 0 1 1 0 1 0  = 26(10)
+ 0 0 0 1 1 0 0  = 12(10)
-----------------
  0 1 0 0 1 1 0  = 38(10)
```

Binary Subtraction

Subtraction and Borrow, these two words will be used very frequently for thebinary subtraction. There are four rules of binary subtraction.

Case	A - B	Subtract	Borrow
1	0 - 0	0	0
2	1 - 0	1	0
3	1 - 1	0	0
4	0 - 1	0	1

Example − Subtraction

0011010 - 001100 = 00001110

```
          1 1      borrow
  0 0 1 1 0 1 0  = 26(10)
- 0 0 0 1 1 0 0  = 12(10)
-----------------
  0 0 0 1 1 1 0  = 14(10)
```

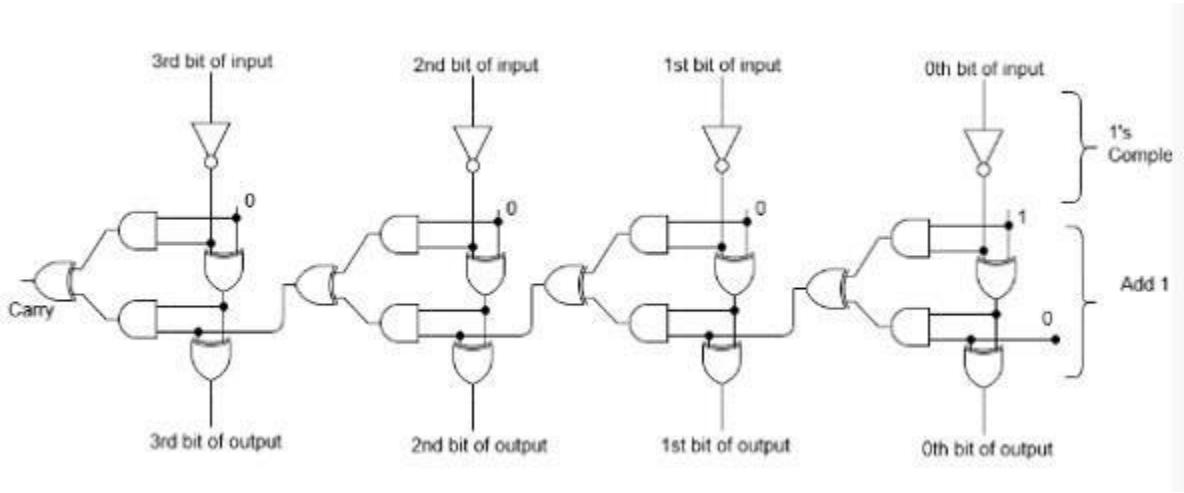
2's Complement Representation

Binary Number System is one the type of most popular Number Representationtechniques that used in digital systems. In the Binary System, there are only two symbols or possible digit values, i.e., 0 (off) and 1 (on). Represented by any device that only 2 operating states or possible conditions.

Generally, there are two types of complement of Binary number: 1's complement and 2's complement. To get 1's complement of a binary number, simply invert the given number. For example, 1's complement of binary number 110010 is 001101. To get 2's complement of binary number is 1's complement of given number plus 1 to the least significant bit (LSB). For example 2's complement of binary number 10010 is (01101) + 1 = 01110.

2's Complement of a Binary Number

There is a simple algorithm to convert a binary number into 2's complement. To get 2's complement of a binary number, simply invert the given number and add 1 to the least significant bit (LSB) of given result. Implementation of 4-bit 2's complementation number isgiven as following below.



Eg

Find 2's complement of binary number 10101110.

Simply invert each bit of given binary number, which will be 01010001. Then add 1 to theLSB of this result, i.e., 01010001+1=01010010 which is answer.

Eg

Find 2's complement of binary number 10001.001.

Simply invert each bit of given binary number, which will be 01110.110 Then add 1 to theLSB of this result, i.e., 01110.110+1=01110.111 which is answer.

Eg

Find 2's complement of each 3 bit binary number.

Simply invert each bit of given binary number, then add 1 to LSB of these inverted numbers,

Binary number	1's complement	2's complement
000	111	000
001	110	111
010	101	110
011	100	101
100	011	100
101	010	011
110	001	010
111	000	001

Uses of 2's Complement Binary Numbers

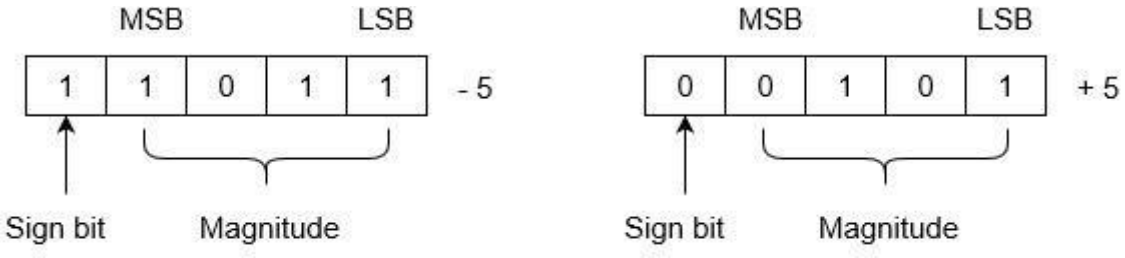
There are various uses of 2's complement of Binary numbers, mainly in signed Binarynumber representation and various arithmetic operations for Binary numbers, e.g., additions, subtractions, etc. Since 2's complement representation is unambiguous, so it very useful in Computer number representation.

2's Complementation in Signed Binary number Representation

Positive numbers are simply represented as simple Binary representation. But if the number is negative then it is represented using 2's complement. First represent the number with positive sign and then take 2's complement of that number.

Eg

Let we are using 5 bits registers. The representation of -5 and +5 will be as follows:



+5 is represented as it is represented in sign magnitude method. -5 is represented using the following steps:

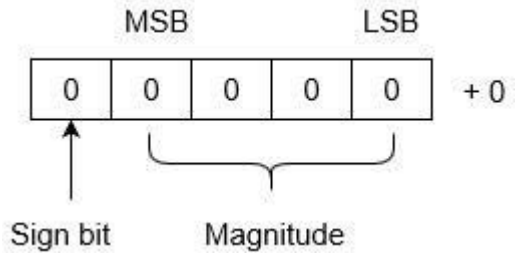
(i) +5 = 0 0101

(ii) Take 2's complement of 0 0101 and that is 1 1011. MSB is 1 which indicates that numberis negative.

MSB is always 1 in case of negative numbers.

Range of Numbers −For k bits register, positive largest number that can be stored is $(2^{(k-1)}-1)$and negative lowest number that can be stored is $-(2^{(k-1)})$.

The advantage of this system is that 0 has only one representation for -0 and +0. Zero (0) is considered as always positive (sign bit is 0) in 2's complement representation. Therefore, it isunique or unambiguous representation.



Let's see arithmetic operations: Subtractions and Additions in 2's complement binarynumbers.

2's Complement Arithmetic Subtractions by 2's Complement

The algorithm to subtract two binary number using 2's complement is explained as following below −

- Take 2's complement of the subtrahend
- Add with minuend
- If the result of above addition has carry bit 1, then it is dropped and this result will bepositive number.
- If there is no carry bit 1, then take 2's complement of the result which will be negativeNote that

subtrahend is number that to be subtracted from the another number, i.e., minuend. Also, note that adding *end-around carry-bit* occurs only in 1's complement arithmetic operations but not 2's complement arithmetic operations.

- **Eg (Case-1: When Carry bit 1)** −Evaluate 10101 - 00101
- According to above algorithm, take 2's complement of subtrahend 00101, which will be 11011, then add both of these. So, 10101 + 11011 =1 10000. Since, there is carry bit 1, so dropped this carry bit 1, and take this result will be 10000 will be positive number.
- **Eg (Case-2: When no Carry bit)** −Evaluate 11001 - 11100
- According to above algorithm, take 2's complement of subtrahend 11110, which will be 00100. Then add both of these, So, 11001 + 00100 =11101. Since there is no carry bit 1, so take 2's complement of above result, which will be 00011, and this is negative number, i.e, 00011, which is the answer.
- Similarly, you can subtract two mixed (with fractional part) binary numbers.

Additions by 2's Complement

There are difference scenario for addition of two binary numbers using 2'scomplement. These are explained as following below.

Case-1 − Addition of positive and negative number when positive number has greatermagnitude: When positive number has greater magnitude, then take simply 2's complement of negativenumber and carry bit 1 is dropped and this result will be positive number.

Example −Add 1110 and -1101.

So, take 2's complement of 1101, which will be 0011, then add with given number. So,1110+0011=1 0001, and carry bit 1 is dropped and this result will be positive number, i.e., +0001.

Note that if the register size is big then use sign extension method of MSB bit to preservesign of number.

Case-2 − Addition of positive and negative number when negative number has greatermagnitude −
When the negative number has greater magnitude, then take 2's complement of negative number and add with given positive number. Since there will not be any end-around carry bit,so take 2's complement of the result and this result will be negative.

Example −Add 1010 and -1100 in five-bit registers.

Note that there are five-bit registers, so these new numbers will have 01010 and -01100. Nowtake 2's complement of 01100 which will be 10100 and add 01010+10100=11110. Then take2's complement of this result, which will be 00010 and this will be negative number, i.e., - 00010, which is the answer.

Case-3 − Addition of two negative numbers −
You need to take 2's complement for both numbers, then add these 2's complement of numbers. Since there will always be end-around carry bit, so it is dropped. Now, take 2's complement also of previous result, so this will be negative number.
Alternatively, you can add both of these Binary numbers and take result which will be negative only.

Example − add -1010 and -0101 in five bit-register.

These five bit numbers are -01010 and -00101. Add 2's complements of these numbers, 10110+11011 =1 10001. Since, there is a carry bit 1, so it is dropped. Now take the 2's complement of this result, which will be 01111 and this number is negative, i.e, -01111, which is answer.

Note that 2's complement arithmetic operations are much easier than 1's complement because of there is no addition of *end-around-carry-bit.*

Arithmetic Building Blocks

Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer. Some of the characteristics of combinational circuits are following −

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

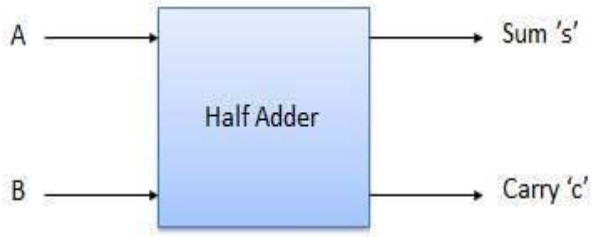
Block diagram

We're going to elaborate few important combinational circuits as follows.

Half Adder

Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**.

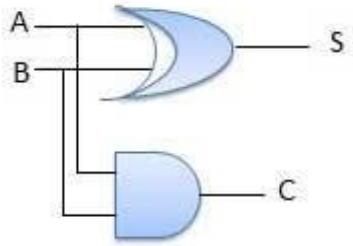
Block diagram



Truth Table

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

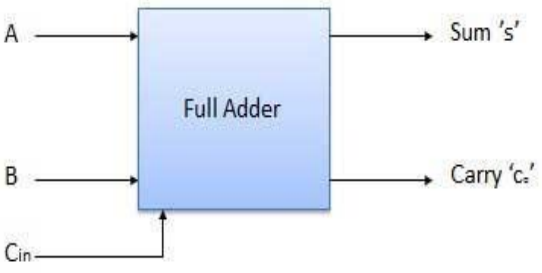
Circuit Diagram



Full Adder

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

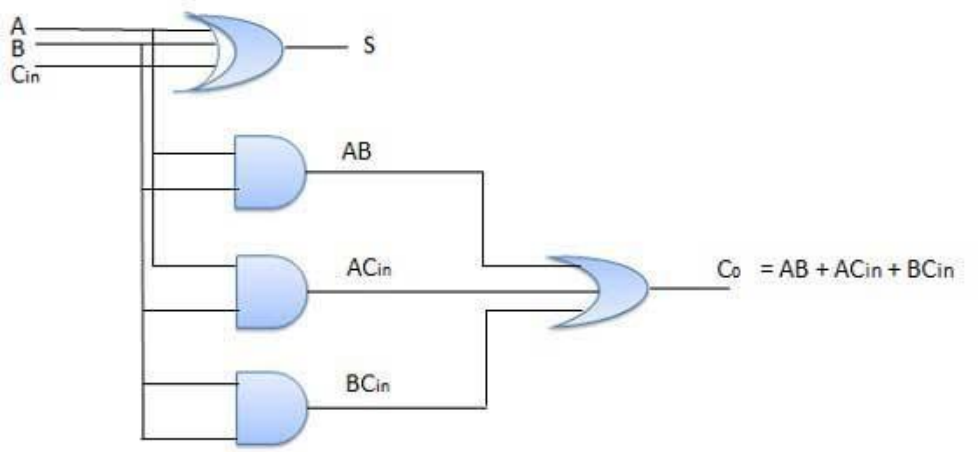
Block diagram



Truth Table

Inputs			Output	
A	B	Cin	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Circuit Diagram



Introduction to Sequential Circuits

SEQUENTIAL LOGIC CIRCUITS

Combinational logic output depends on the present input level where are sequential logic output not only depends on the input levels but also stored levels

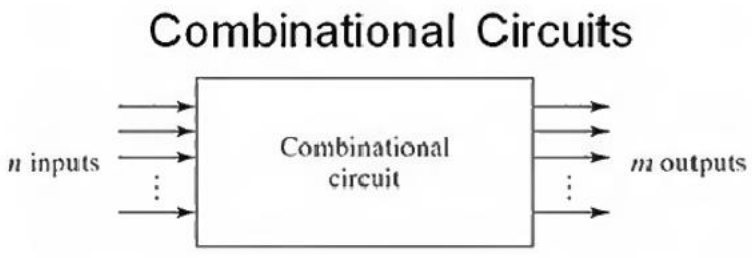
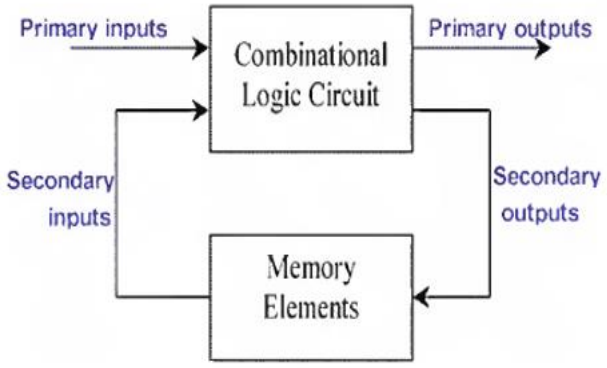


Fig. Block Diagram of Combinational Circuit

Sequential Circuit



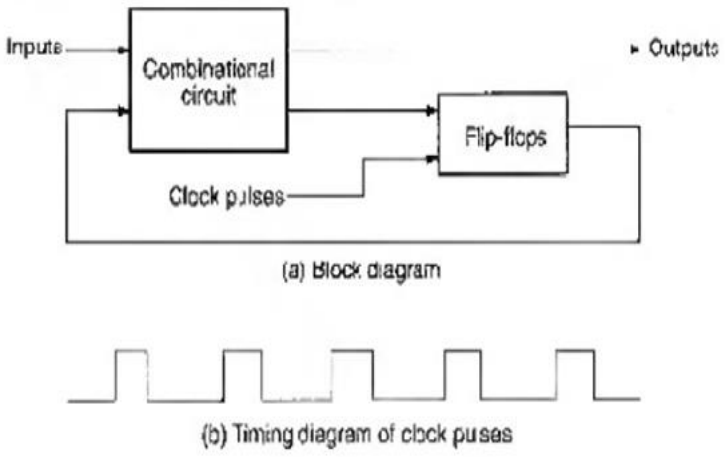
The memory elements are device capable of storing binary information the binary information stored in the memory elements at any given time defines the state of the sequential circuit the input and the present state of the memory element determine the output memory elements next state is also a function of external Input and capable of storing binary information the binary information stored in the memory elements at any given time defines the state of the sequential circuit the input and the present state of the memory element determine the output memory elements next state is also a function of external import and present state. A sequential circuit is specified by a time sequence of input, outputs and internal States.

There are two types of sequential circuit their classified there are two types of sequential circuit their classification depends on the timing of their signals:

1. **synchronous sequential circuits**
2. **asynchronous sequential circuits**

Asynchronous sequential circuit

This is a system whose outputs depend upon the order in which its input variables change and can be affected at any instant of time



Synchronous sequential circuit

This type of system uses storage elements called flip flops that are employed to change their binary value only at discrete instants of time.

Synchronous sequential circuits uses logic gates and flip flop storage devices. sequential circuits have a clock signal as one of their inputs. All state transitions in such circuits occur only when the clock value is either 0 or 1 or happen at the Rising or falling edge of the clock depending on the type of memory element used in the circuit. Synchronization is achieved by a timing device called a clock pulse generator. Clock pulses are distributed throughout the system in such a way that the flip flops are affected only with the arrival of the synchronization pulse synchronous sequential circuit that use clock pulses in the inputs are called clocked sequential circuits.

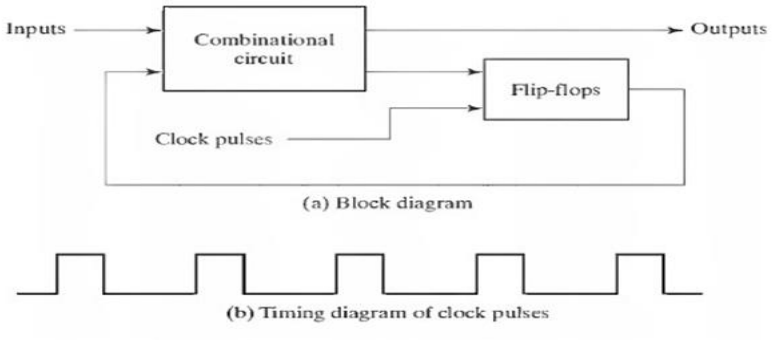


Fig. 5-2 Synchronous Clocked Sequential Circuit

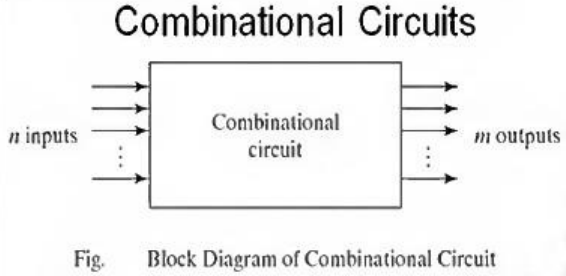
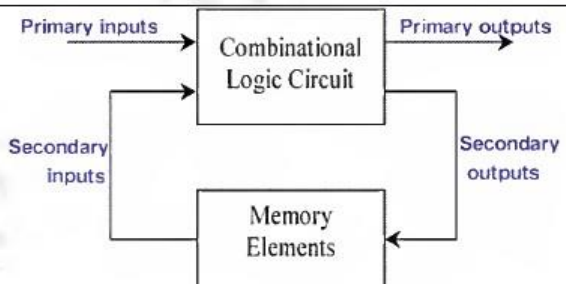
Combinational Circuits	Sequential Circuits
1. The circuit whose output at any instant depends only on the input present at that instant only is known as combinationational circuit.	1. The circuit whose output at any instant depends not only on the input present but also on the past output a is known as sequential circuit
2. This type of circuit has no memory unit.	2. This type of circuit has memory unit for store past output.
3. Examples of combinational circuits are half adder, full adder, magnitude comparator, multiplexer, demultiplexer e.t.c.	3. Examples of sequential circuits are Flip flop, register, counter e.t.c.
4. Faster in Speed	Slower compared to Combinational Circuit
	

Fig. Block Diagram of Combinational Circuit

Latches and Flip-Flops

- Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information.
- The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs as long as the enable signal is asserted.
- Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal.
- This enable signal is usually the controlling clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes.
- There are basically four main types of latches and flip-flops:
  1. SR
  2. D
  3. JK
  4. T
- The major differences in these flip-flop types are the number of inputs they have and how they change state.

Flip-flops (FF)

A FF is an electronic device that has two stable states. One state is assigned the logic 1 value and the other is the logic 0. In other words, the memory elements used in sequential circuits are the flip flop. These circuits are binary cells capable of storing one bit of information.

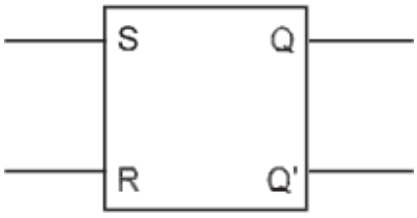
- The basic 1-bit digital memory circuit is known as a flip-flop.

TYPES OF FLIP-FLOPS

There are different types of flip-flops depending on how their inputs and clock pulses cause transition between two states. We will discuss four different types of flip-flops in this chapter, *viz*., S-R, D, J-K, and T. Basically D, J-K, and T are three different modifications of the S-R flip-flop.

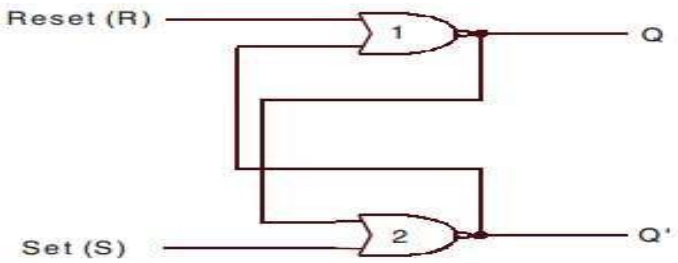
S-R (Set-Reset) Flip-flop

An S-R flip-flop has two inputs named Set (S) and Reset (R), and two outputs Q and Q'. The outputs are complement of each other, *i.e.*, if one of the outputs is 0 then the other should be 1. This can be implemented using NAND or NOR gates. The block diagram of an S-R flip-flop is shown in Figure below:-



S-R Flip-flop Based on NOR Gates

An S-R flip-flop can be constructed with NOR gates at ease by connecting the NOR gates back to back as shown in Figure below. The cross-coupled connections from the output of gate 1 to the input of gate 2 constitutea feedback path. This circuit is not clocked and is classified as an asynchronous sequential circuit. The truth table for the S-R flip-flop based on a NOR gate is shown in the table below



Inputs		Outputs		Action
S	R	Q_{n+1}	Q'_{n+1}	
0	0	Q_n	Q'_n	No change
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	0	Forbidden (Undefined)
0	0	–	–	Indeterminate

The outputs for all the possible conditions as shown in the above table are described as follows.

Case 1. For S = 0 and R = 0, the flip-flop remains in its present state (Q_n). It means that the next state of the flip-flop does not change, *i.e.*, $Q_{n+1} = 0$ if $Q_n = 0$ and vice versa.

- First let us assume that $Qn = 1$ and $Q'_n = 0$.Thus the inputs of NOR gate 2 are 1 and 0, and therefore its output $Q'n+1 = 0$. This output $Q'_{n+1} = 0$ is fed back as the input of NOR gate1, thereby producing a 1 at the output, as both of the inputs of NOR gate 1 are 0 and 0; so $Q_{n+1} = 1$ as originally assumed.
- Now let us assume the opposite case, *i.e.,* $Q_n = 0$ and $Q'_n = 1$. Thus the inputs of NOR gate 1 are 1 and 0, and therefore its output $Q'_{n+1} = 0$. This output $Q_{n+1} = 0 = 0$ is fed back as the input of NOR gate 2, thereby producing a 1 at the output, as both of the inputs of NOR gate 2 are 0 and 0; so $Q'_{n+1} = 1$ asoriginally assumed.
- Thus we find that the condition S = 0 and R = 0 do not affect the outputs of the flip-flop, which means this is the memory condition of the S-R flip-flop.

Case 2. The second input condition is S = 0 and R = 1. The 1 at R input forces the output of NOR gate 1 to be 0(*i.e.*, $Q_{n+1} = 0$).

- Hence both the inputs of NOR gate 2 are 0 and 0 and so its output $Q'_{n+1} = 1$. Thus the condition S = 0 and R = 1 will always reset the flip-flop to 0.
- Now if the R returns to 0 with S = 0, the flip-flop will remain in the same state.

Case 3. The third input condition is S = 1 and R = 0. The 1 at S input forces the output of NOR gate 2 to be 0 (*i.e.*, $Q'_{n+1} = 0$).

- Hence both the inputs of NOR gate 1 are 0 and 0 and so its output $Q_{n+1} = 1$. Thus the conditionS = 1 and R = 0 will always set the flip-flop to 1.
- Now if the S returns to 0 with R = 0, the flip-flop will remainin the same state.

Case 4. The fourth input condition is S = 1 and R = 1. The 1 at R input and 1 at S input forces the output of both NOR gate 1 and NOR gate 2 to be 0. Hence both the outputs of NOR gate 1 and NOR gate 2 are 0 and 0; *i.e.*, $Q_{n+1} = 0$ and $Q'_{n+1} = 0$.

- Hence this condition S = 1 and R = 1 violates the fact that the outputs of a flip-flop will always be the complement of each other.
- Since the condition violates the basic definition of flip-flop, it is called the ***undefined*** condition.
- Generally, this condition must be avoided by making sure that 1s are not applied simultaneously to both of the inputs.

Case 5. If case 4 arises at all, then S and R both return to 0 and 0 simultaneously, and then any one of the NOR gates acts faster than the other and assumes the state. For example, if NOR gate 1 is faster than NOR gate 2, then Q_{n+1} will become 1 and this will make $Q'_{n+1} = 0$.

- Similarly, if NOR gate 2 is faster than NOR gate 1, then Q'_{n+1} will become 1 and this will make $Q_{n+1} = 0$.
- Hence, this condition is determined by the flip-flop itself. Sincethis condition cannot be controlled and predicted it is called the ***indeterminate*** condition.