

Digital Electronics

By Anita Rani

Chapter 1-Introduction

Objectives : To understand

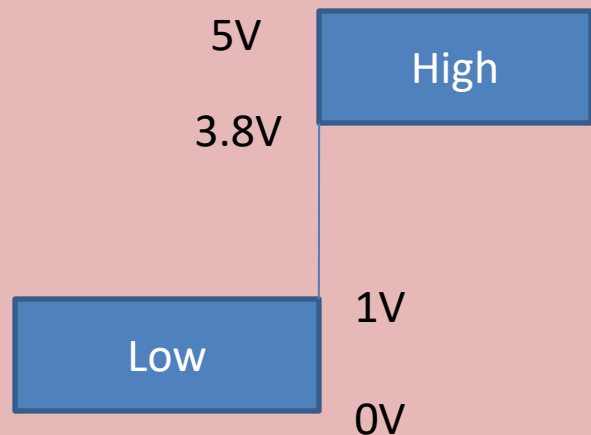
- **Digital system**
- **Analog system**
- **Need of Digital system/advantages of digital systems**
- **Applications of digital systems**

Digital signal and Digital system

- Digital signal: The signal which can have any of the two discrete voltage levels. One of these levels is Low level and other is High level.
- Digital system: The system used to process a digital signal is called a digital system.

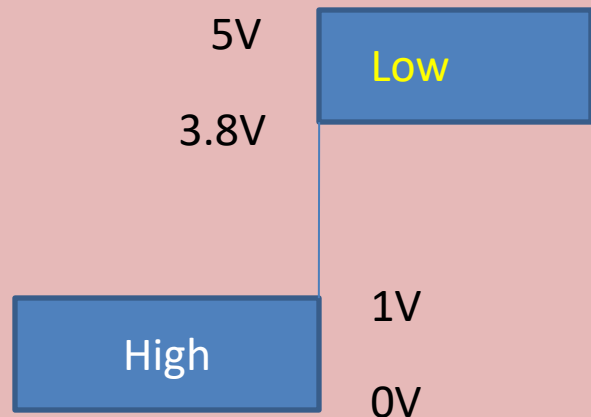
Positive Logic

- In digital system, if lower voltage level is used to represent Low and higher voltage level is used to represent High, it is called positive Logic.



Negative Logic

- In digital system, if lower voltage level is used to represent High and higher voltage level is used to represent Low ,it is called Negative Logic.



Need of digital systems/Advantages of digital systems.

- The devices used in digital systems operates in ON or OFF state.
- There are only few basic operations in digital systems.
- Digital techniques use boolean algebra that is easy to understand .
- A large no of ICs are available for performing various operations.

- Effect of ageing ,fluctuations, temperature and noise is small in digital circuits.
- Digital systems have storage capability.
- Display of data and information is convenient.
- Latest electronic systems are digital in nature.
- Designing and development of digital systems is simple due to availability of various logic families.

Applications of Digital systems.

- Computers.
- Medical Electronics.
- Instrumentation.
- Communication e.g. Mobile Cell Phones, Internet.
- Consumer products e.g. watches, calculators, smart TV
- Information systems
- Industrial Control systems
- Banking and finance
- Scientific Instruments.
- Office machines, homes, cars, education etc.

Chapter 2

NUMBER SYSTEMS

Number systems

- Decimal Number system
- Binary Number system
- Octal number system
- Hexadecimal Number system

Decimal Number system

Base/Radix – 10

Ten symbols-0,1,2,3,4,5,6,7,8,9.

Binary Number system

- Base/radix – 2(two)
- Symbols- 0,1.
- One binary digit is called a bit.e.g. 0
- Nibble: A combination of four bits.e.g.0011
- Byte: a combination of eight bits. e.g.10110011
- MSB(Most Significant bit)- the leftmost bit of the binary number
- LSB(Least Significant Bit)-The rightmost bit of the binary number

Octal Number system

- Base/radix-8
- Symbols-0,1,2,3,4,5,6,7.


Hexadecimal number system

- Base/Radix- 16
- Symbols : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Conversion from decimal to binary

- A given decimal number can be converted to binary by successive division of given number by 2 and noting down the remainders. The remainders taken in reverse order give the binary equivalent of decimal number.
- e.g. $24_{10} = 11000_2$


2	24		
2	12	-	0
2	6	-	0
2	3	-	0
2	1	-	1
2	0	-	1



Conversion from decimal to octal

- A given decimal number can be converted to binary by successive division of given number by 8 and noting down the remainders. The remainders taken in reverse order give the binary equivalent of decimal number.
- e.g. $24_{10} = 30_8$

8	24	
8	3	0
8	0	3



Conversion from decimal to hexadecimal

- A given decimal number can be converted to hexadecimal by successive division of given number by 16 and noting down the remainders in Hex. The remainders taken in reverse order give the binary equivalent of decimal number.
- e.g. $246_{10} = F0_{16}$

16		246			
16		15	-	0	0
		0	-	15	F

↑

Conversion from binary to decimal

- A binary number is converted into decimal by multiplying its bits with the powers of 2 and adding the resulting values as shown below:
- $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
 $= 8 + 0 + 2 + 1$
 $= 11$

Conversion from octal to decimal

- An octal number is converted into decimal by multiplying its digits with the powers of 8 and adding the resulting values as shown below:
- $1023_8 = 1 \times 8^3 + 0 \times 8^2 + 2 \times 8^1 + 3 \times 8^0$
 $= 512 + 0 + 16 + 3$
 $= 531_{10}$

Conversion from hexadecimal to decimal

- An hexadecimal number is converted into decimal by multiplying its digits with the powers of 16 and adding the resulting values as shown below:
- $F5_{16} = 15 \times 16^1 + 5 \times 16^0$
 $= 240 + 5$
 $= 245_{10}$

Conversion from binary to octal

- Make the groups of three bits starting from LSB and write their octal equivalent.
- e.g. $11001 = 011\ 001$
 $= 31_8$

Conversion from octal to binary

- Write the binary equivalent of each octal digit in three bits as shown below:
- $25_8 = 010\ 101$
- $37_8 = 011\ 111$

Conversion from binary to Hexadecimal

- Make the groups of four bits starting from LSB and write their hexadecimal equivalent.
- e.g. 1111001 = 0111 1001
= 79₁₆

Conversion from Hexadecimal to binary

- Write the binary equivalent of each hexadecimal digit in four bits as shown below:
- $C5_{16} = 1100\ 0101_2$
- $37_{16} = 0011\ 0111_2$

1's complement of a number

- To find the 1's complement of a number change all the 1's in the given number to '0' and all the '0's to '1'.
- e.g. 1's complement of 11000101 is 00111010

2's complement of a binary number

- To find the 2's complement of a number, find the 1's complement of given number and then add 1 to it.
- e.g. 2's complement of 11000101 is 00111011

Binary Arithmetic

- Binary addition:
- $0+0=0$
- $0+1=1$
- $1+0=0$
- $1+1=10$ (sum 0 and 1 carry)

e.g. $1100+0101= 1\ 0001$

$$\begin{array}{r} 1100 \\ 0101 \\ \hline 1\ 0001 \\ \hline \end{array}$$

Binary subtraction

- Binary subtraction:
- $0-0=0$
- $0-1=1$ and 1 borrow
- $1-0=1$
- $1-1=0$

e.g. $1100-0101=0111$

$$\begin{array}{r} 1100 \\ 0101 \\ \hline 0111 \\ \hline \end{array}$$

Binary multiplication

- e.g. Multiply 1101 by 0101

$$\begin{array}{r} 1101 \\ 101 \\ \hline 1101 \\ 0000X \\ 1101XX \\ \hline 1\ 000001 \\ \hline \end{array}$$

1's complement method of subtraction

- E.g. subtract 0111 from 1101 using 1's complement.

Step 1 Find the 1's complement of subtrahend ; i.e the number to be subtracted.

1's complement of subtrahend is 1000.

Step 2 Add it to minuend(the number to be subtracted from) i.e. $1101 + 1000$

$$\begin{array}{r} 1101 \\ 1000 \\ \hline 1,0101 \end{array}$$

Step 3 Check if there is carry. Pick and Add the carry bit

$$\begin{array}{r} 1101 \\ 1000 \\ \hline 0101 \\ 1 \\ \hline 0110 \end{array}$$

Ans

2's complement method of subtraction

- E.g. subtract 0111 from 1101 using 2's complement.

Step 1 Find the 2's complement of subtrahend ; i.e the number to be subtracted.

1's complement of subtrahend is 1001.

Step 2 Add it to minuend(the number to be subtracted from)
i.e. $1101 + 1001$

$$\begin{array}{r} 1101 \\ 1001 \\ \hline 1,0110 \end{array}$$

Step 3 Check if there is carry. Discard the carry bit

$$\begin{array}{r} 1101 \\ 1000 \\ \hline 0110 \text{ Ans} \end{array}$$

CHAPTER3

CODES

Codes.

- Straight Binary code
- BCD Code
- Grey Code
- Excess-3 code

Straight Binary code

- uses natural binary form. It is a weighted code. Each position has a weight-1,2,4,8,16 and so on.

BCD code:

- Decimal digits 0 through 9 are represented by their binary equivalent 4 bit code.
- E.g. 23 in decimal is equivalent to 0010 0011 in BCD.
- Also known as 8421 code. -8,4,2,1 are the weights of the four bits of the code.

BCD code:

Binary 0 0 0 0	BCD 0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 0
0 0 1 1	0 0 1 1
0 1 0 0	0 1 0 0
0 1 0 1	0 1 0 1
0 1 1 0	0 1 1 0
0 1 1 1	0 1 1 1
1 0 0 0	1 0 0 0
1 0 0 1	1 0 0 1

Gray code

- Gray Code is a non weighted Code.
- The two consecutive codes differ by one bit.
So also called reflected code.

Excess-3 Code

- Another form of BCD code
- The code for each decimal digit is obtained by adding decimal 3 to the natural BCD code of the digit

Excess-3 Code

Binary 0 0 0 0	BCD 0 0 0 0	EXCESS-3 0 0 1 1
0 0 0 1	0 0 0 1	0 1 0 0
0 0 1 0	0 0 1 0	0 1 0 1
0 0 1 1	0 0 1 1	0 1 1 0
0 1 0 0	0 1 0 0	0 1 1 1
0 1 0 1	0 1 0 1	1 0 0 0
0 1 1 0	0 1 1 0	1 0 0 1
0 1 1 1	0 1 1 1	1 0 1 0
1 0 0 0	1 0 0 0	1 0 1 1
1 0 0 1	1 0 0 1	1 1 0 0

Error detecting Codes

- Parity: It is an extra bit added to the binary code to make the number of ones in the code even or odd.
- Parity is used to detect single bit error.

BCD 0 0 0 0	BCD with even parity 0 0 0 0 0	BCD with odd parity 1 0 0 0 0
0 0 0 1	1 0 0 0 1	0 0 0 0 1
0 0 1 0	1 0 0 1 0	0 0 0 1 0
0 0 1 1	0 0 0 1 1	1 0 0 1 1
0 1 0 0	1 0 1 0 0	0 0 1 0 0
0 1 0 1	0 0 1 0 1	1 0 1 0 1
0 1 1 0	0 0 1 1 0	1 0 1 1 0
0 1 1 1	1 0 1 1 1	0 0 1 1 1
1 0 0 0	1 1 0 0 0	0 1 0 0 0
1 0 0 1	0 1 0 0 1	1 1 0 0 1

Chapter 4

Logic Gates

Logic gate

- A **logic gate** is an elementary building block of a digital **circuit**. A Logic gate is a digital circuit with one or more inputs and one output. Most logic gates except NOT gate have two or more inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels.

Various Logic gates

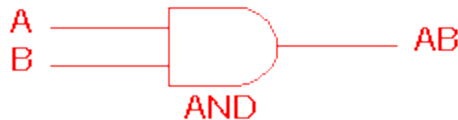
AND, OR, NOT, NAND, NOR, EXOR
and EXNOR **gates.**

Truth Tables

- Truth tables help understand the behaviour of logic gates.
- They show how the input(s) of a logic gate relate to its output(s).
- The gate input(s) are shown in the left column(s) of the table with all the different possible input combinations. This is normally done by making the inputs count up in binary.
- The gate output(s) are shown in the right hand side column.

AND Gate: The output is high if all the inputs are high. It has two or more inputs and one output.

- Symbol of AND Gate(2 input)

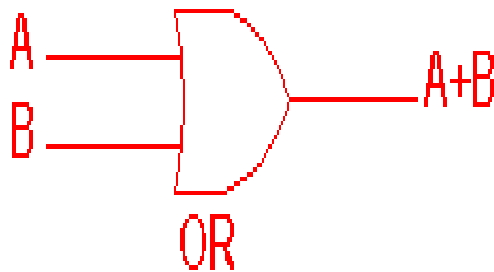


- Truth table of AND gate

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

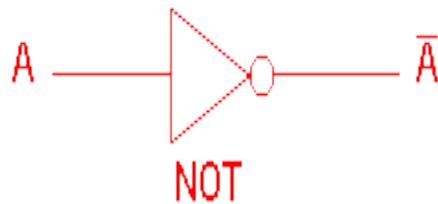
OR- GATE: The output is high if any or all the inputs are high. It has two or more inputs and one output.

Symbol of OR gate Truth table of OR gate



2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

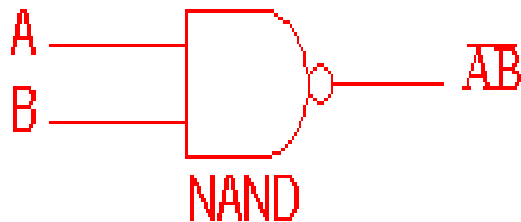
NOT GATE: It has one input and one output.



NOT gate	
A	\bar{A}
0	1
1	0

The output is complement of the input.

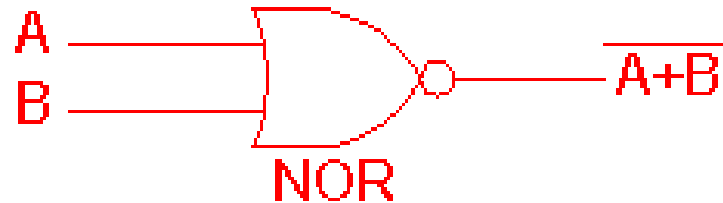
NAND GATE: The output is high if any of the inputs is low. It has two or more inputs and one output.



2 Input NAND gate		
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE- It is NOT-OR gate, If any input is high output is low

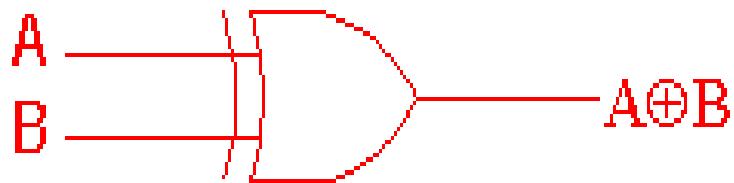
Symbol of 2 input NOR Gate



Truth Table of NOR Gate

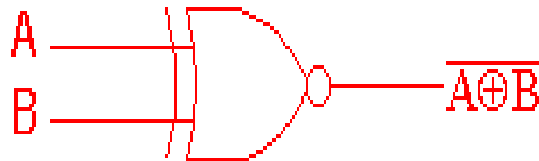
2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

EX-OR GATE



2 Input EXOR gate		
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

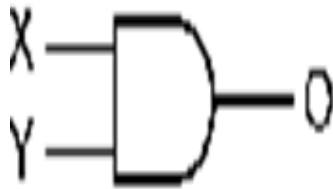
EX-NOR GATE



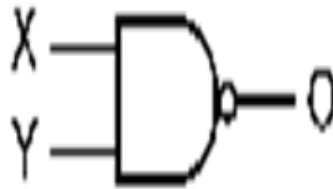
2 Input EXNOR gate		
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Symbols of all gates

AND Gate



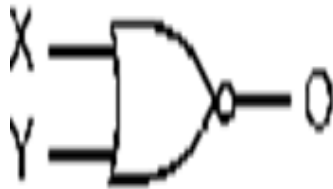
NAND Gate



OR Gate



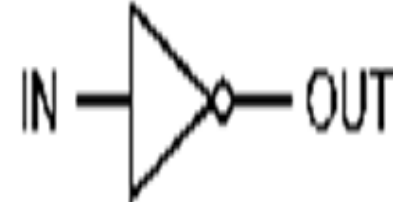
NOR Gate



Ex-OR Gate



NOT Gate



Universal Gates

NAND Gate

NOR Gate

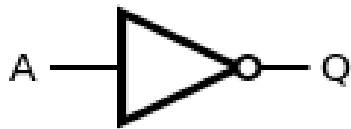
NAND GATE AS UNIVERSAL GATE

A NAND gate is a universal gate, meaning that any other gate can be represented as a combination of NAND gates.

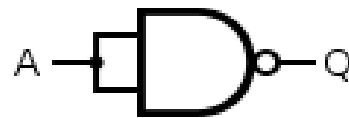
NOT

A NOT gate is made by joining the inputs of a NAND gate together. A NAND gate is equivalent to an AND gate followed by a NOT gate.

Desired NOT Gate NAND Construction



$$Q = \text{NOT}(A)$$



$$= A \text{ NAND } A$$

Truth TABLE

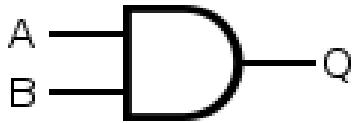
Input A	Output Q
0	1
1	0

AND USING NAND GATES

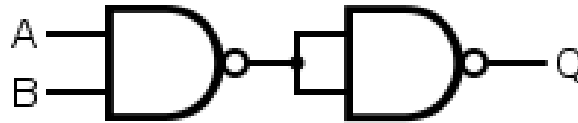
AND

An AND gate is made by following a NAND gate with a NOT gate as shown below. This gives a NOT NAND, i.e. AND.

Desired AND Gate



NAND Construction



Truth Table

Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

OR USING NAND GATES

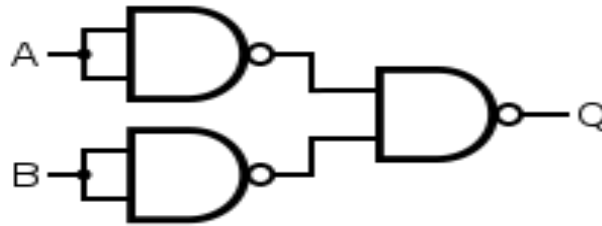
OR

If the truth table for a NAND gate is examined or by applying [De Morgan's Laws](#), it can be seen that if any of the inputs are 0, then the output will be 1. To be an OR gate, however, the output must be 1 if any input is 1. Therefore, if the inputs are inverted, any high input will trigger a high output.

Desired OR Gate



NAND Construction



$$Q = A \text{ OR } B = (A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B)$$

Truth Table

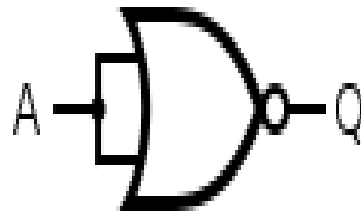
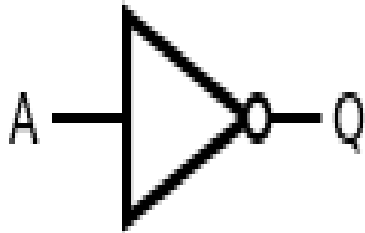
Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

NAND GATE AS UNIVERSAL GATE

NOT

This is made by joining the inputs of a NOR gate.

Desired NOT Gate NOR Construction



OR USING NOR GATES

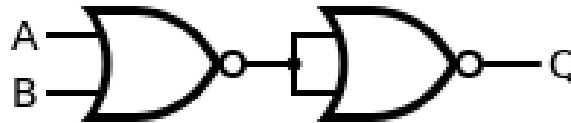
OR

The OR gate is equivalent to a NOR gate followed by a NOT gate.

Desired OR Gate



NOR Construction



Truth Table

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

AND USING NOR GATES

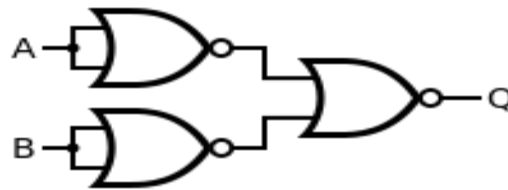
AND

An AND gate gives a 1 output when both inputs are 1; a NOR gate gives a 1 output only when both inputs are 0. Therefore, an AND gate is made by inverting the inputs of a NOR gate.

Desired AND Gate



NOR Construction



Truth Table

Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

Chapter 5

LOGIC SIMPLIFICATION

Logic Simplification

- To simplify the logic expressions so as to minimise the number of gates required.
- There are two methods of logic simplification.
 - Using Laws of boolean Algebra
 - Using K-Map

Laws of Boolean Algebra

- Commutative Law
 - $A+B=B+A$ (OR Law)
 - $A.B=B.A$ (AND Law)

A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Associative Law

- $A+(B+C) = (A+B)+C$
- $A.(B.C) = (A.B).C$

A	B	C	$A+(B+C)$	$(A+B)+C$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Distributive Law

- $A(B+C)=AB+AC$

A	B	C	B+C	A(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

AND LAWS

- $A \cdot 0 = 0$
- $A \cdot 1 = A$
- $A \cdot A = A$
- $A \cdot \underline{A} = 0$

OR LAWS

- $A+0=0$
- $A+1=1$
- $A+A=A$
- $A+A=1$

NOT LAWS

- IF $A=0$, $\bar{A}=1$
- IF $A=1$, $\bar{A}=0$

Some other Laws

$$A + \bar{A}B = A + B$$

$$\bar{\bar{A}} = A$$

Demorgan's Theorm

$$\overline{A.B} = \overline{A} + \overline{B} \text{ (first Theorm)}$$

$$\overline{A+B} = \overline{A}. \overline{B} \text{ (Second Theorm)}$$

Demorgan's First Theorem: It states that the complement of product of variables is equal to the sum of individual complements of variables

- $\overline{A.B} = \overline{A} + \overline{B}$

A	B	A.B	$\overline{A.B}$	\overline{A}	\overline{B}	$\overline{A+B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Demorgan's Second Theorem: It states that the complement of sum of variables is equal to the product of individual complements of variables

- $\overline{A+B} = \overline{A} \cdot \overline{B}$:

A	B	A+B	$\overline{A+B}$	\overline{A}	\overline{B}	$\overline{A} \cdot \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Karnaugh Map

It is a graphical method of logic simplification of logic expressions so as to minimize the number of gates required and hence the cost of the logic circuit.

2,3,4 -Variable K-Map

A \ B	0	1
0		
1		

A \ BC	00	01	11	10
0				
1				

AB \ CD	00	01	11	10
00				
01				
11				
10				

Consider the Sum of Product form,
example: $f = x'y'z' + x'yz' + xy'z' + xyz' + xyz$

x \ yz				
	00	01	11	10
0	1	0	0	1
1	1	0	1	1

$$f = x'y'z' + x'yz' + xy'z' + xyz' + xyz$$

Designing of k-map

- Make the largest possible groups of adjacent 1's (remembering that the map wraps on its edges).
- Every 1 must be in a group, overlapping groups are ok as are single 1's.
- The number of adjacent 1's in any group must be an integral power of 2, i.e. 1, 2, 4, 8, 16.
- Translate each group into a product term by including each variable or its complement if the variable does not change value over the group

Solution

x \ yz	00	01	11	10	
	0	1	1	0	
0	1	0	0	1	z'
1	1	0	1	1	

xy

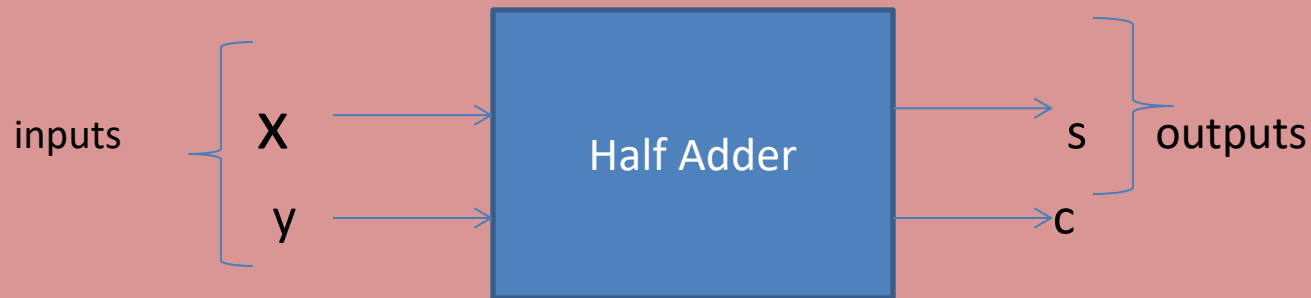
$$\begin{aligned}f &= x'y'z' + x'yz' + xy'z' + xyz' + xyz \\ &= xy + z'\end{aligned}$$

Chapter 6

Arithmetic circuits

Half adder

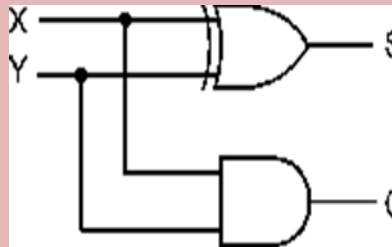
A digital circuit that adds two bits. It has two inputs (say X ,Y)and two outputs(sum and carry).



Half adder

Input1	Input2	Carry	Sum
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

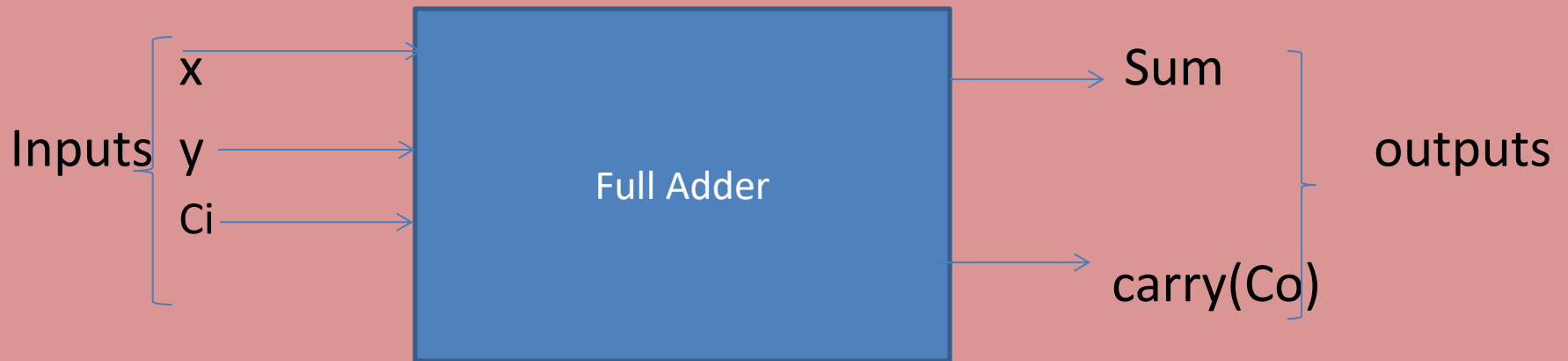
Truth table of Half adder



Logic Diagram of Half Adder

Full Adder

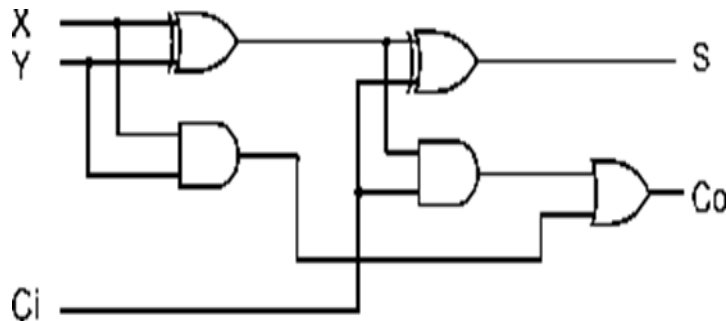
A digital circuit that can add three bits. So it has three inputs (say X,Y C_{in}) and two outputs(sum,carry C_o)



Full Adder

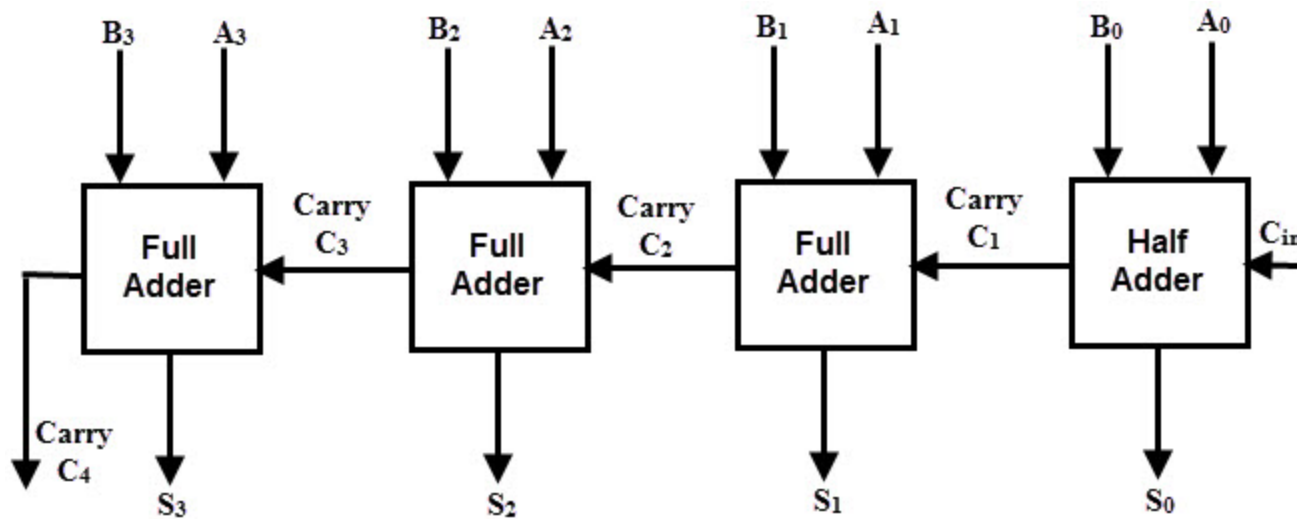
Truth Table :

Input1	Input 2	Input Carry	Output Carry	Sum
X	Y	Ci	C _o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



4 bit adder circuit

It is a digital circuit that is capable of adding two 4-bit binary numbers.



CHAPTER 7

Decoders, Multiplexers, De-Multiplexers and Encoder

Objectives

To study function and block diagram of encoder

To study function and block diagram of decoder

To study basic functions and block diagram of MUX and DEMUX with different ICs

Decoder

A decoder is a combinational digital circuit that detect the presence of binary input and makes the corresponding decimal output high.

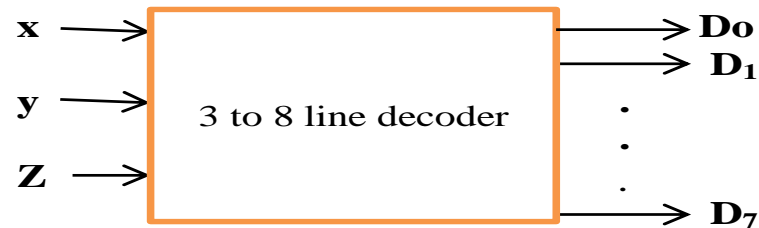


Fig. 5.3 Block diagram of 3 to 8 line decoder

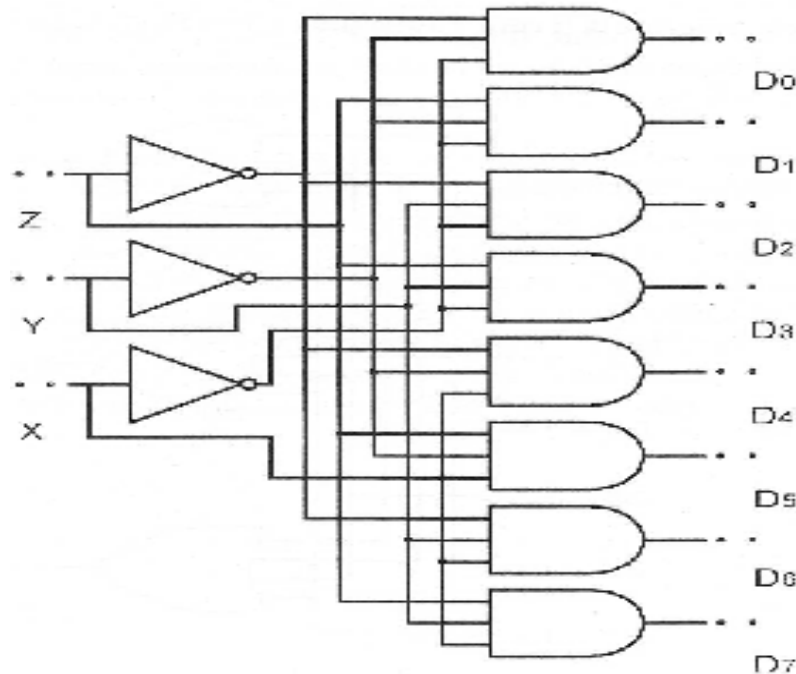


Fig. 5.3 Logic diagram of 3 to 8 line decoder

Truth Table of 3 line to 8 line decoder

X	Y	Z	Do	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

Encoder

An Encoder is a combinational digital circuit that detect the presence of decimal input and indicates on the binary outputs.

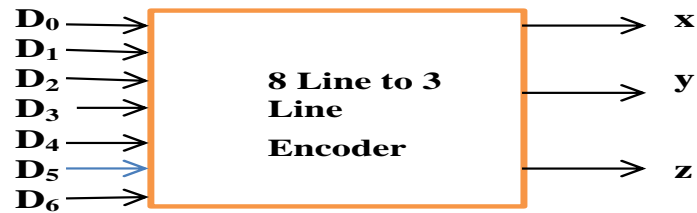


Fig. 5.1 Block Diagram of 8 line to 3 line encoder

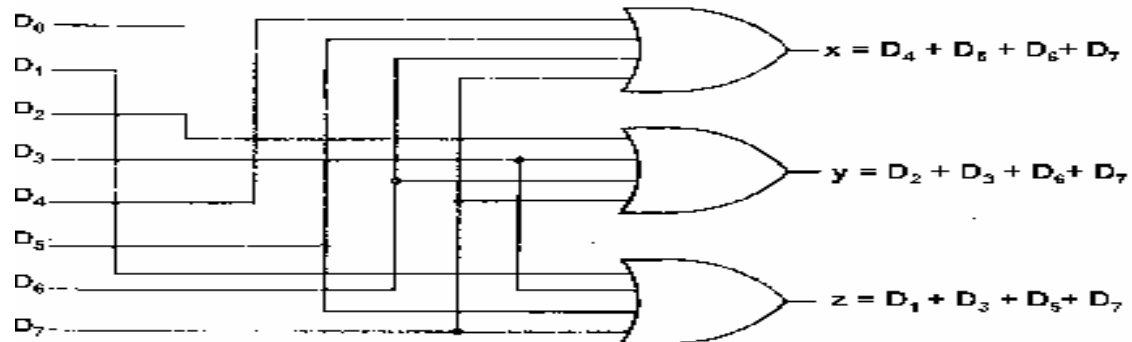


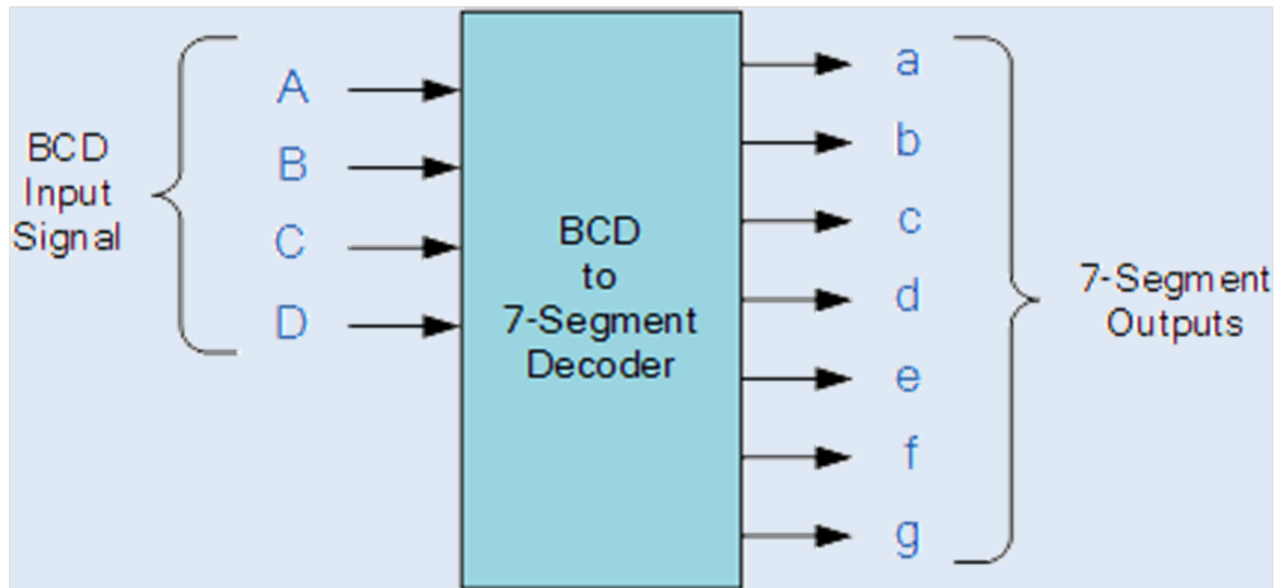
Fig. 5.2 Logic Diagram of 8 line to 3 line encoder

Truth table of 8 line to 3 line encoder(active low inputs)

D0	D1	D2	D3	D4	D5	D6	D7	x	y	z
L	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	L
H	H	L	H	H	H	H	H	H	L	H
H	H	H	L	H	H	H	H	H	L	L
H	H	H	H	L	H	H	H	L	H	H
H	H	H	H	H	L	H	H	L	H	L
H	H	H	H	H	H	L	H	L	L	H
H	H	H	H	H	H	H	L	L	L	L

BCD to seven segment decoder.

A **BCD to Seven Segment decoder** is a combinational logic circuit that accepts a decimal digit in **BCD** (input) and generates appropriate outputs for the **segments** to **display** the input decimal digit.

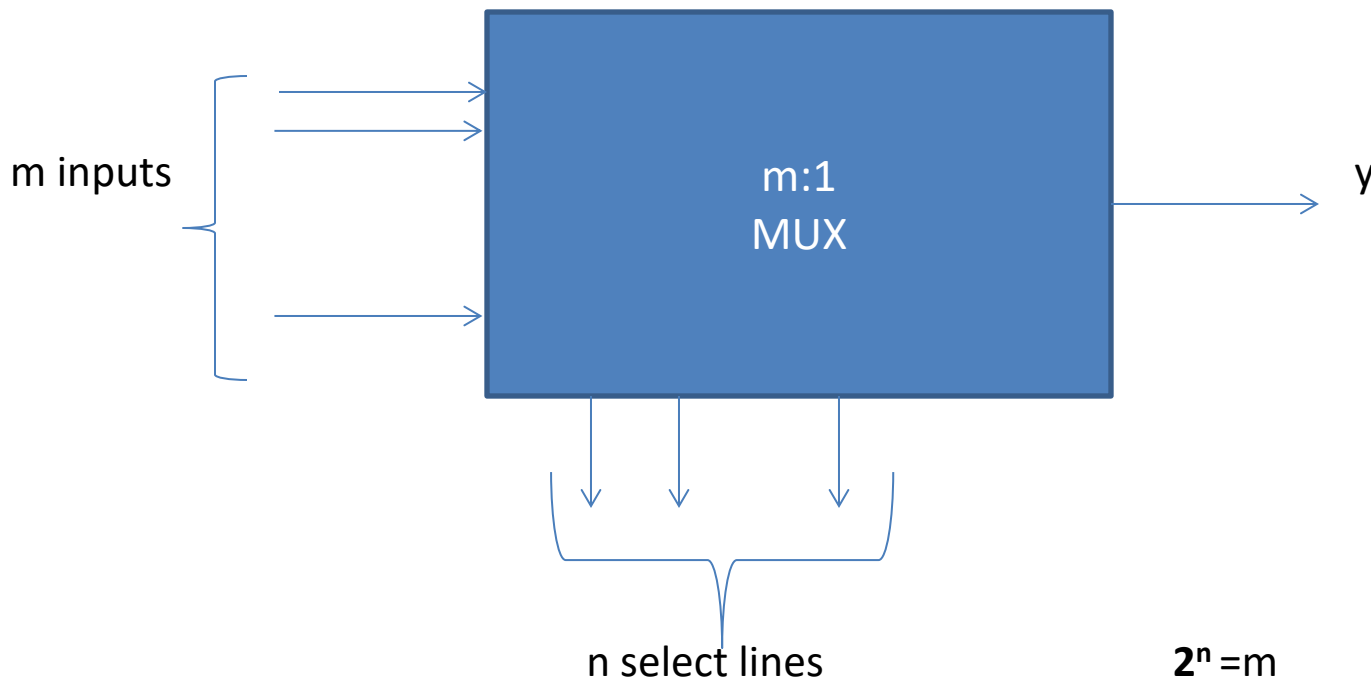


Truth Table of BCD to 7-Segment Decoder

A	B	C	D	a	b	c	d	e	f	g
L	L	L	L	H	H	H	H	H	H	L
L	L	L	H	L	H	H	L	L	L	L
L	L	H	L	H	H	L	H	H	L	H
L	L	H	H	H	H	H	H	L	L	H
L	H	L	L	L	H	H	L	L	H	H
L	H	L	H	H	L	H	H	L	H	H
L	H	H	L	H	L	H	H	H	H	H
L	H	H	H	H	H	L	L	L	L	L
H	L	L	L	H	H	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H

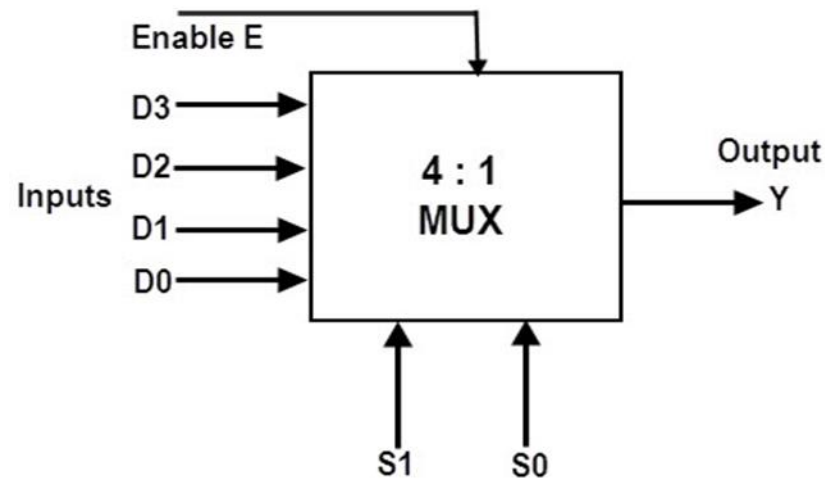
Multiplexer

- A multiplexer is a digital logic circuit with many inputs say m , and one output and n select lines such that $2^n = m$. Any of the m inputs is transferred to the output depending on the data on the select lines.



4:1 Multiplexer :

It has 4 inputs and 1 output and two select lines. Any of the inputs is transferred to the output depending on select lines

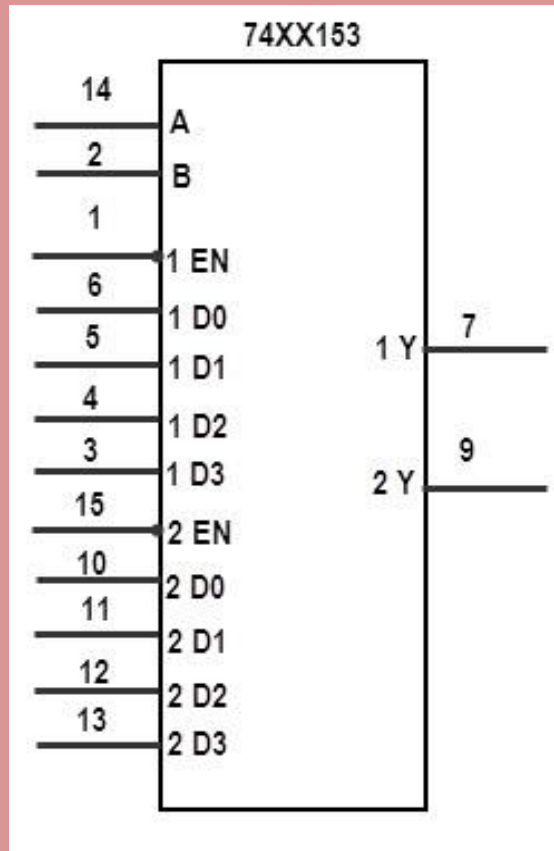


Truth table of 4:1 multiplexer

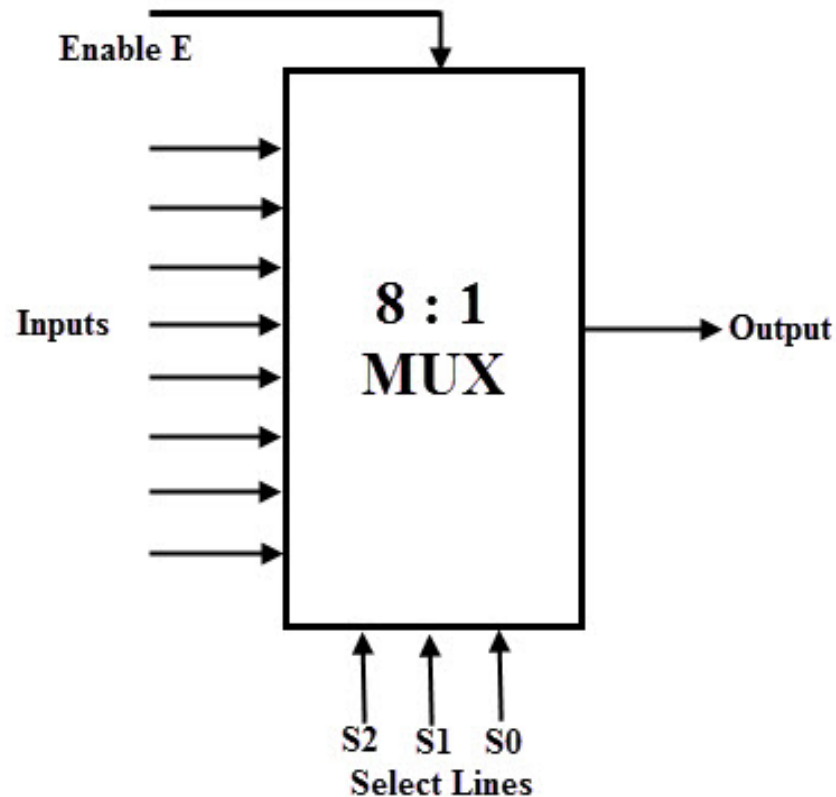
S1	S0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

4:1 MUX IC

Pin Diagram of 74153 (Dual 4:1 Mux)



8:1 MUX : It has 8 inputs, one output and 3 select lines.



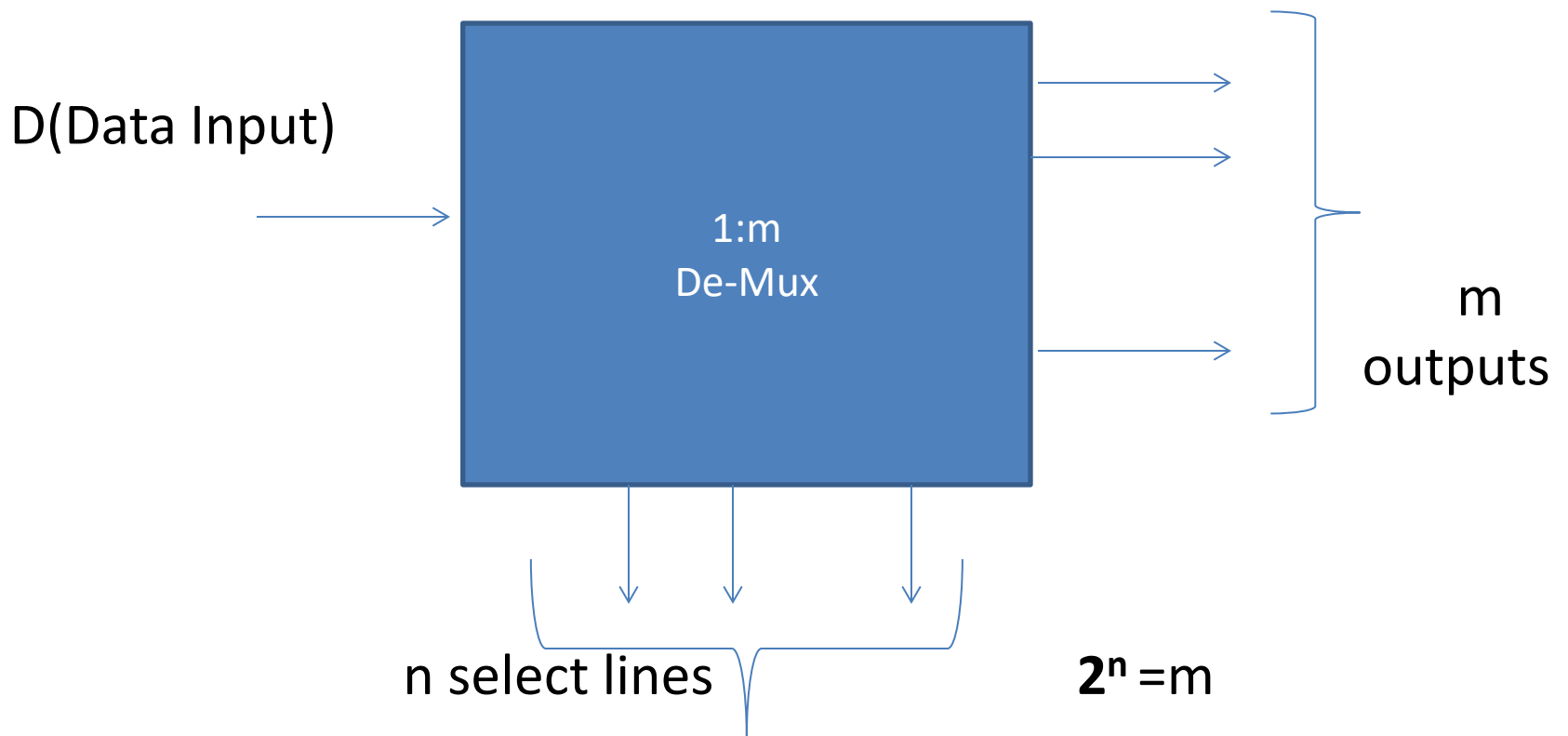
Truth table of 8:1 multiplexer

S2	S1	S0	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

De-Multiplexer

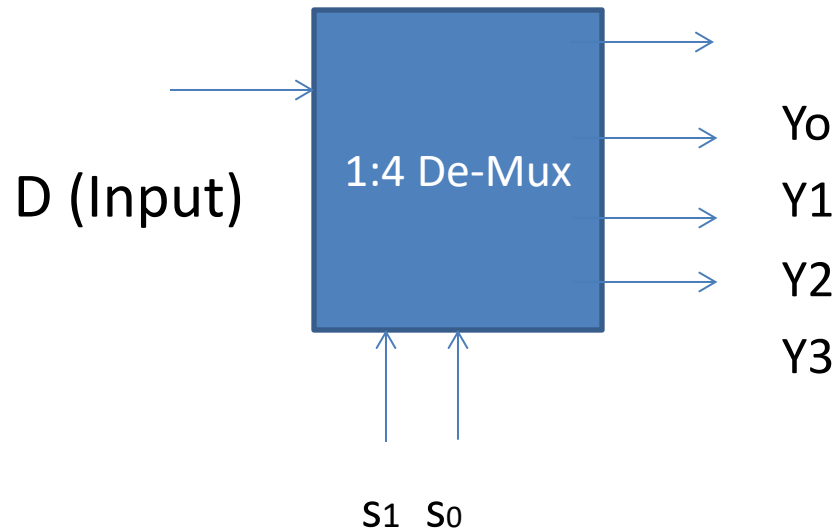
- A de-multiplexer is a digital logic circuit with one input and many outputs say m , and n select lines such that $2^n = m$. The input is transferred to any of the output depending on the data on the select lines.

De-Multiplexer

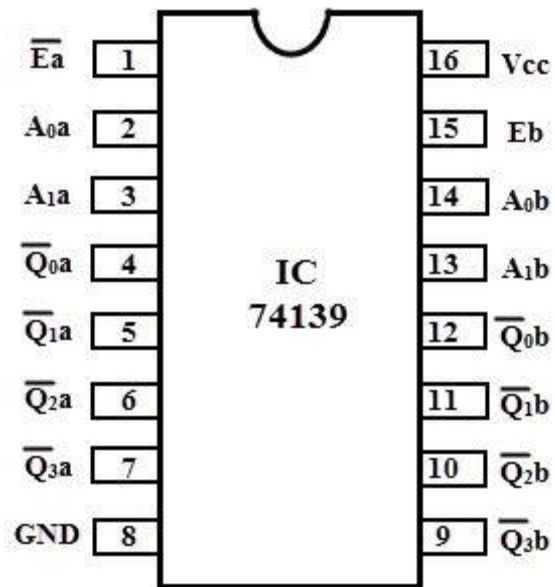


1:4 De-mux :

It has 4 outputs and 1 input and two select lines. The input is transferred to any of the outputs depending on select lines



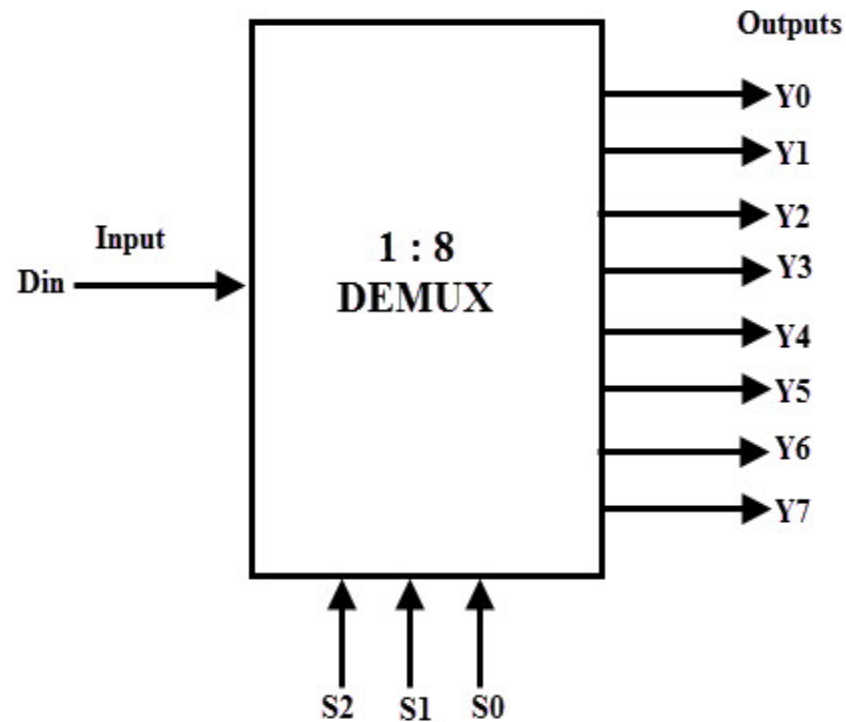
1:4 De-mux IC



Truth table for 1:4 De-Mux (with active low data o/p)

S ₁	S ₀	Y ₃	Y ₂	Y ₁	Y ₀
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

1:8 De-mux



Truth Table of 1:8 De-mux

S2	S1	S0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	Din	0	0	0	0	0	0	0
0	0	1	0	Din	0	0	0	0	0	0
0	1	0	0	0	Din	0	0	0	0	0
0	1	1	0	0	0	Din	0	0	0	0
1	0	0	0	0	0	0	Din	0	0	0
1	0	1	0	0	0	0	0	Din	0	0
1	1	0	0	0	0	0	0	0	Din	0
1	1	1	0	0	0	0	0	0	0	Din

CHAPTER8

Latches and flip flops

Objectives

To study the concept and types of latch with their working and applications

Operation and truth tables of RS, T, D, Master/Slave JK flip flops.

Difference between a latch and a flip flop

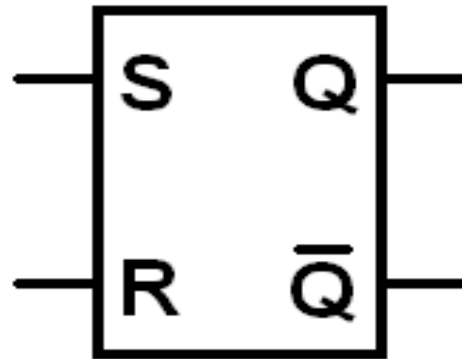
Latches and Flip Flops

Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information. They are bi-stable elements i.e. they have two stable states-LOW or HIGH. If they are in LOW state they will remain to continue in LOW state unless triggered. Similarly if they are in HIGH state they will remain to continue in HIGH state unless triggered

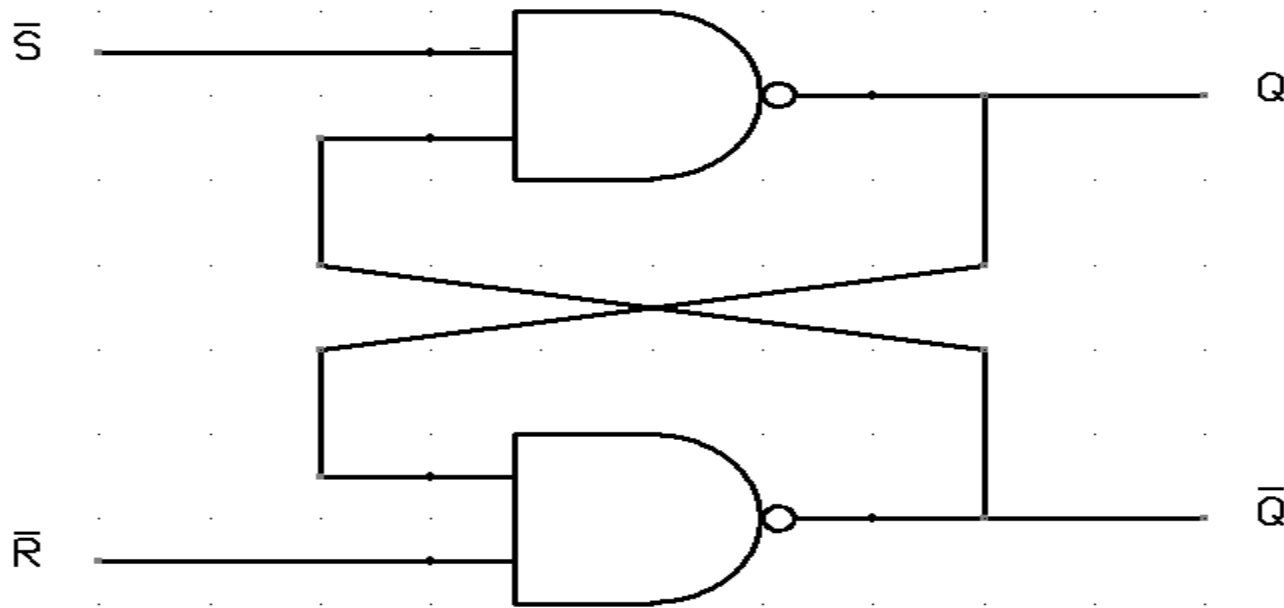
Difference between a latch and a flip flop

The main difference between a latch and flip-flop is that for latches, their outputs are constantly affected by their inputs as long as they are enabled. when they are enabled, their content changes immediately when their inputs change. Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the clk signal. The outputs change with the clock signal. After the rising or falling edge of the clock, the flip-flop output remains same even if the input changes. There are basically four main types flip-flops: SR, D, JK, and T.

SR LATCH : It has two input S and R and two outputs Q and \bar{Q}



Logic diagram of SR Latch: It consists of two NAND gates. It has two inputs S and R and two outputs Q and \bar{Q} .

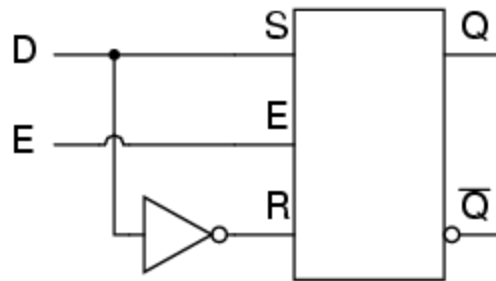


Truth table of SR Latch

S'	R'	Q	Q'
0	0	Invalid	Invalid
0	1	1	0
1	0	0	1
1	1	No change	No change

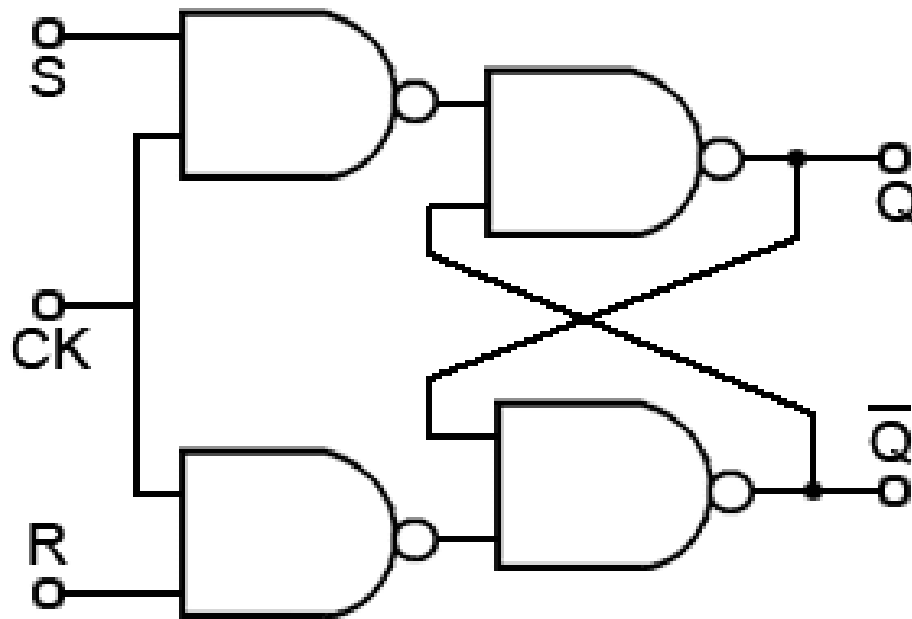
D LATCH : A D latch can be obtained from the SR latch as shown in the following diagram.

- Block diagram and truth table of D latch



D	Q	Q'
0	0	1
1	1	0

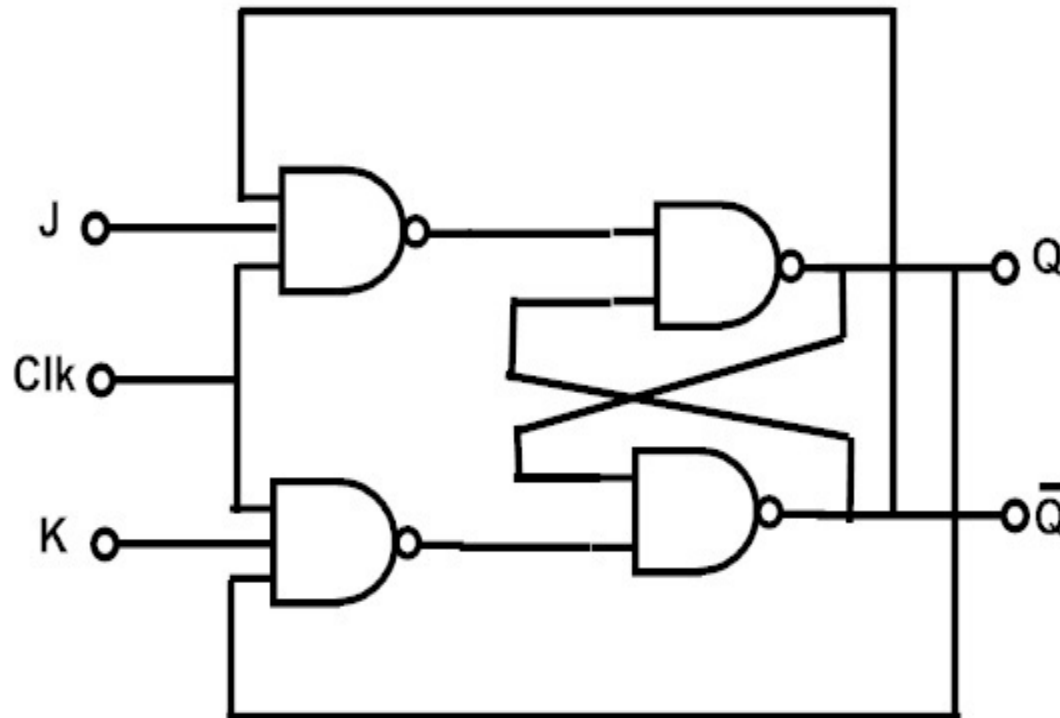
SR Flip Flop: It has two input S and R, one clock input CK and two output Q and \overline{Q}




Truth Table of SR Flip-Flop

clk	S	R	Q_{n+1}	Q'_{n+1}
↑	0	0	Q_n	Q_n'
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	Invalid	

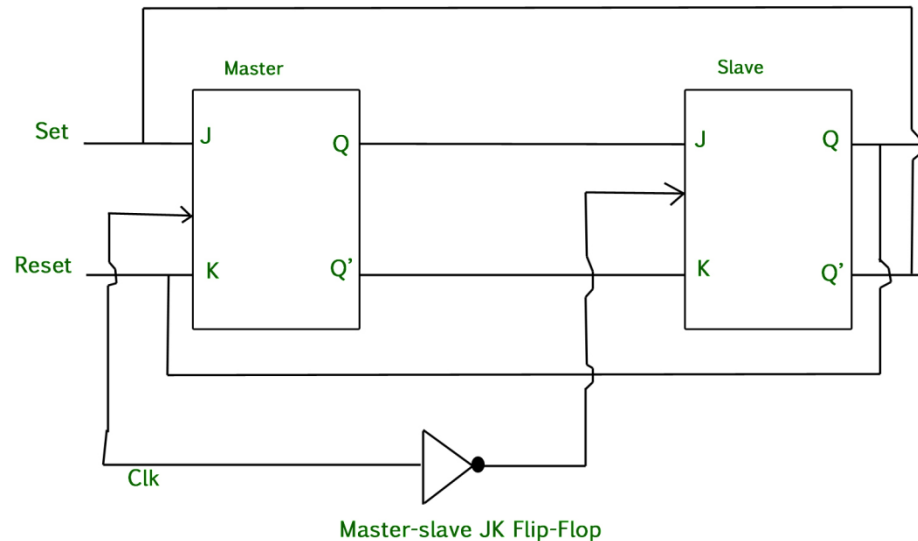
JK FLIPFLOP: In SR flip flop one of the states is not utilised(invalid). This is overcome in JK flip flop.



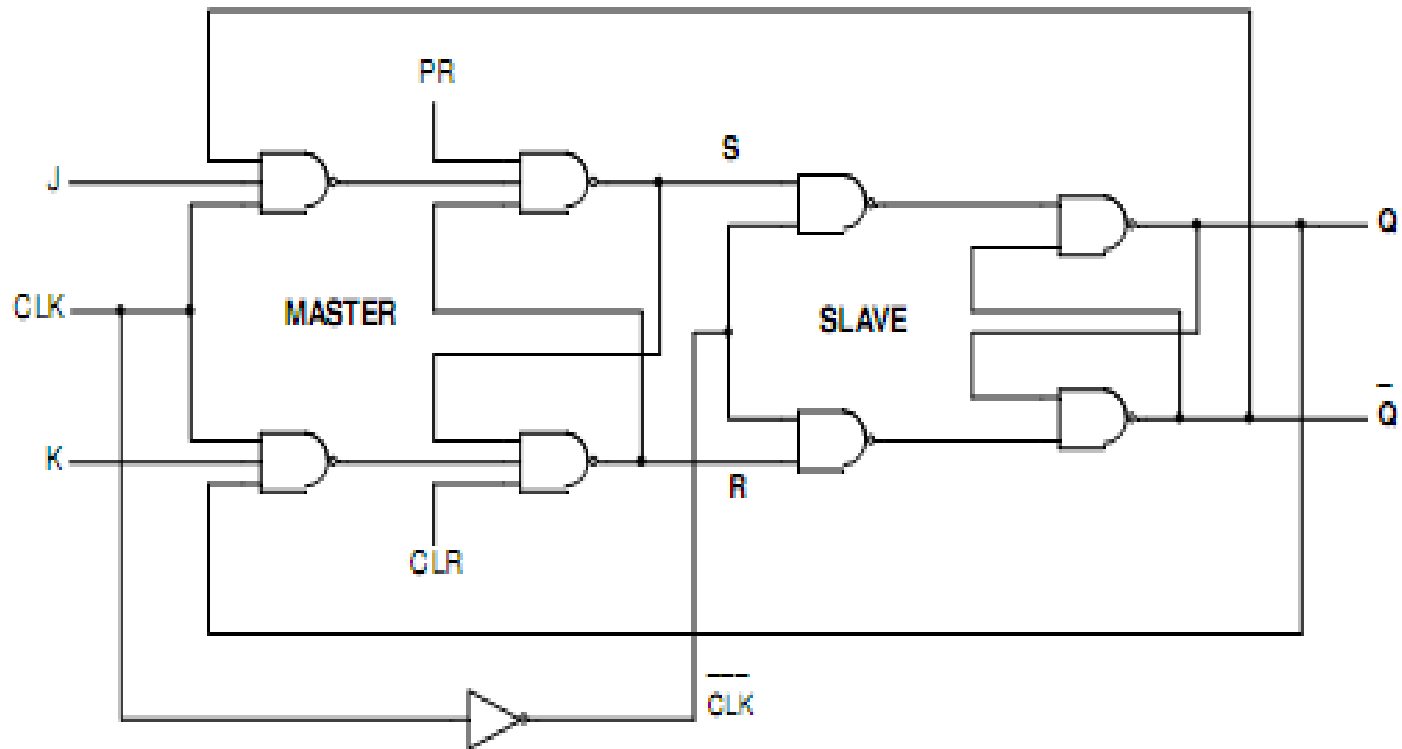
TRUTH TABLE OF JK FLIPFLOP

clk	J	K	Q_{n+1}	Q'_{n+1}	
	0	0	Q_n	Q_n'	No change
	0	1	0	1	set
	1	0	1	0	reset
	1	1	Q_n'	Q_n	toggle



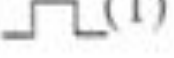

JK MASTER SLAVE FLIPFLOP : It is used to remove the race around condition in the toggle mode of JK flip flop. In race around condition flip-flop toggles more than once in a clock and the output is unpredictable. Master Slave JK Flip Flop contains a master JK Flip flop and a slave JK flip flop. Master toggles at the positive edge of the pulse . Slave follows the master but at the negative edge of the clock pulse. As a result the flip-flop toggles after the complete pulse.



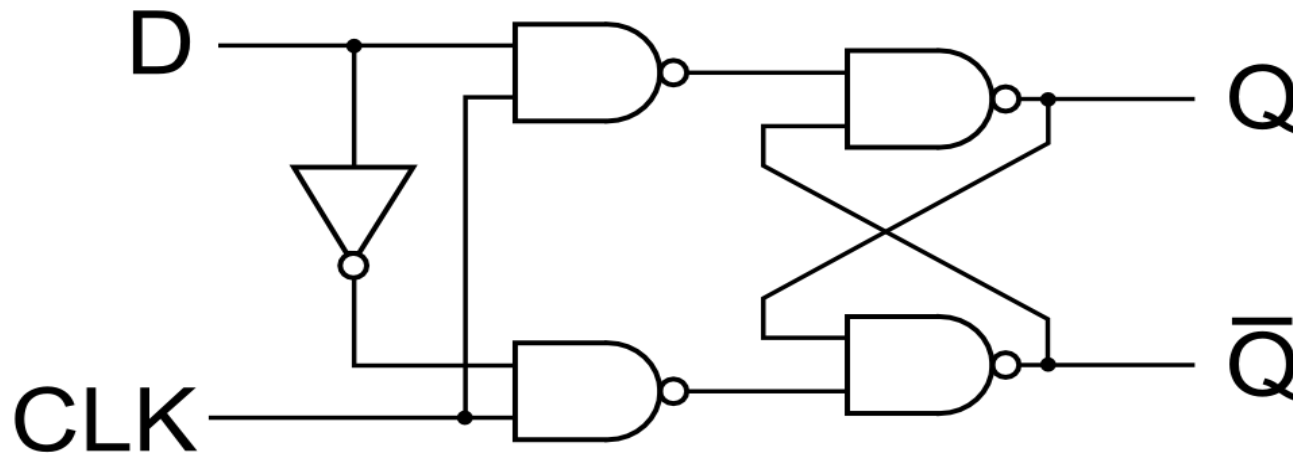
JK MASTER SLAVE FLIPFLOP




TRUTH TABLE OF JK MASTER SLAVE FLIPFLOP

Case	Inputs			Outputs		Remark
	CLK	J	K	Q_{n+1}	\bar{Q}_{n+1}	
I	\times	0	0	Q_n	\bar{Q}_n	No change
II	 (1)	0	0	Q_n	\bar{Q}_n	No change
III	 (1)	0	1	0	1	Reset
IV	 (1)	1	0	1	0	Set
V	 (1)	1	1	\bar{Q}_n	Q_n	Toggle

D Flip Flop : It has one input D and two outputs Q and \bar{Q} . The output changes with the input at the application of clk .i.e. it delays the input by one clk pulse. That is why called as D Flip Flop.



Truth Table of D Flip Flop

CLK	D	Q	Q'
	0	0	1
	1	1	0

REFERENCES

Digital Electronics and Applications by Malvino Leach, Tata McGraw Hill Education Pvt Ltd, New Delhi

Digital Logic Designs by Morris Mano, Prentice Hall of India, New Delhi

Digital Electronics by RP Jain, Tata McGraw Hill Education Pvt Ltd, New Delhi

Thank You