NUMBER SYSTEM: Number systems are classified into four types

1.Binary numbers

2.Octal numbers

3.Decimal number system

4.Hexa decimal number system

1.BINARY NUMBER SYSTEM: It consists of two binary numbers 0,1. Its base or radix is '2'.

Eg: $(101)_2$

2.OCTAL NUMBER SYSTEM: It consists of 8 binary numbers .Its base or radix is '8'.

Eg: $(23)_8$

3.DECIMAL NUMBER SYSTEM: It consists of 10 binary numbers .Its base or radix is '10'.

Eg: $(35)_{10}$

4.HEXA-DECIMAL: It consists of 16 binary numbers. Its base (or) radix is '16'.Hexa means 6 and decimal means 10.

Eg: $(28)_{16}$

BINARY ADDDITION: 0+0=0

1+0=1

0+1=1

1+1=10(where 1=carry , 0=sum)

BINARY SUBSTRACTION: 0-0=0

0-1 =1(In this borrow is '1' and difference is '1')

1-0=1

1-1=0

CONVERSION OF HEXA DECINAL NUMBER INTO 4-bit BINARY FORM:

	HEXA NUMBER	BINARY NUMBER
	0	0000
	1	0001
	2	0010
	3	0011
	4	0100
	5	0101
	6	0110
	7	0111
	8	1000
	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Binary subtraction: In direct subtractions method subtracting a lower number from higher number doesn't give correct answer that means direct subtraction fails. In this case, to eliminate the above disadvantage complimentary methods are used.

Complimentary methods converts binary subtraction to binary addition is known as r's complimentary method and they are of two types

1.Ones's complimentary method

2.Two's complimentary method

ONE'S COMPLIMENTARY METHOD: The one's complimentary of binary method is obtained by replacing a '0' but with '1' bit and each '1' bit with '0' bit.

Eg: 1110 - > 0001

RULES:

1)Find the one's complimentary of subtrahend to minu end.
2)If carry one the subtraction is positive, round the carry bit which gives the result.
3)If there is no carry bit, the subtraction is negative then re-compliment the number and attach a negative sign before it.

a) $(101)_2 \rightarrow (111)_2$

101 - minu end

111 – substra end

↓↓↓

000 → one's compliment of subtra end

```
  101
  000
-------
 0|101
   ↓↓↓
-------
  010
```

Ans: $-(101)_2$

TWO'S COMPLIMENTARY METHOD:

RULES:

1) Find the two's compliment of subtrahend.
2) Add this two's compliment to minu end.
3) If there is carry one, cross out the carry remaining number gives no result.

4) If there is no carry , re-compliment the result and attach negative sign.

```
 1101
 0101
------
```

a) $(1101)_2 \rightarrow (1011)_2$ 0|0010

1101 – minu end

1011 – substra end

↓ ↓ ↓ ↓

0 1 0 1 → two's compliment of substra end

1

0 1 0 1 ans: $(010)_2$

Binary to Decimal Conversion:

1.Integer:

To convert a binary integer into decimal write the bits in a binary number in row wise and give their positional weights starting from right to left in increasing order cross out the bits and which lie under '0' bit.

Eg: a)1 0 1 b)1 0 1 1 1

2^2 2^1 2^0 2^4 2^3 2^2 2^1 2^0

---

4 + 1=5 16 + 4 + 2 + 1=23

2.Binary Fraction:

To convert a binary fraction into decimal stream line method is used by writing the potential weights in decreasing order from left to right.

Eg: a) . 1 0 1 b) . 1 1 1 1 0 1

2^{-1} 2^{-2} 2^{-3} 2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6}

1/2+1/8 1/2+1/4+1/8+1/16+1/64

0.5 + 0.12=0.62 0.5 + 0.25 + 0.12 + 0.06 + 0.01=0.94

3.Mixed Fraction:

Eg: a) $(100.001)_2$ b) $(1111.1011)_2$

1 0 0 . 0 0 1 1 1 1 1 . 1 0 1 1

2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3} 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3} 2^{-4}

---

4 . 1/8 8 + 4 + 2 + 1 . 1/2 + 1/8 + 1/16

4 . 0.12 = 4.12 15 . 0.5+0.12+0.06 = 15.68

4.Decimal to Binary conversion:

To convert a decimal to binary double dablee method is used. It is known as divided by two methods. In this method the progress divided by 2 and remainders are noted after each division and group of remainders are taken in reverse order(bottom to top) which gives the binary method.

Eg :

5.Binary Fraction:

In this method it is simply multiply by 2 and carries are noted after each multiplication and group of orders taken in forward direction(top to bottom)

Eg :0 .8125

0.8125*2 = 1.6250 with a carry 1

0.625*2 = 1.250 with a carry 1

1.250*2 = 0.50 with a carry 0

0.50*2 = 1.0 with a carry 1

=> 0 . 8125=(1101 .....)$_2$

6.Mixed:

Eg: 15.275

IC-DIGITAL LOGIC GATES

Gates: It is a logic circuit having no. Of inputs but only one output.

Types of Gates: 1 . Basic Gates

2 . Universal Gates

1.Basic Gates:

These Gates are basic building blocks of digital system is known as BASIC GATES.

Eg: AND,OR,NOT.

2.Universal Gates:

By using universal gates we design digital functions.

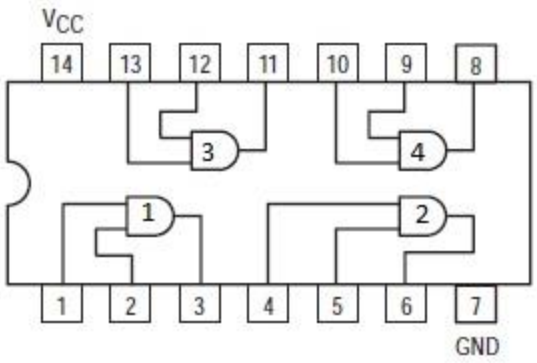
Eg: NAND,NOR.

Truth Table:

Tabular form shows input and output possibilities of logic circuits.

Timing Diagram:

The pictorial or diagrammatic representation of input and output possessbilities of a logic circuit.



AND GATE: It is a logic circuit in which output is high when both input's are high.

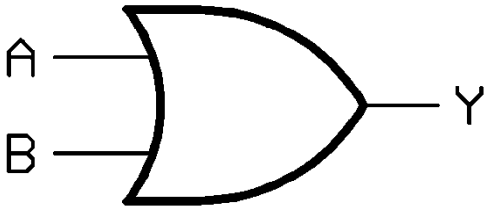
Symbol:

Truth Table:

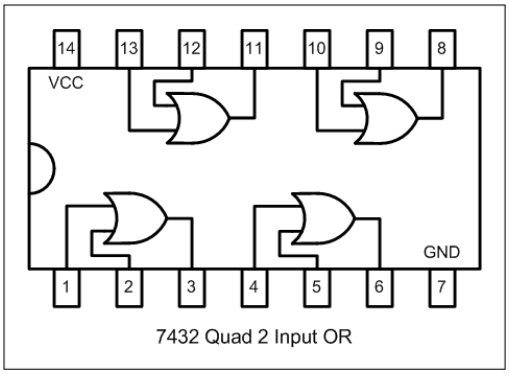
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

Pin Diagram:

Timing Diagram:



OR GATE: It is a logic circuit in which output is high when both inputs are high. (or)It is a logic circuit in which output is high when any one of the input is high.



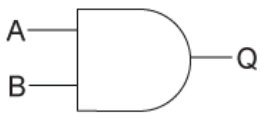
Symbol:

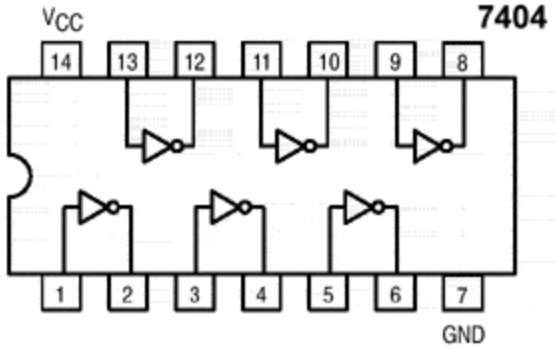
Truth Table:

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

Pin Diagram:

Timing Diagram:



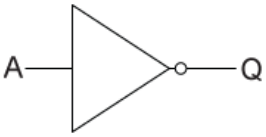


NOT GATE: It is a logic circuit in which output is compliment is that of input.

Symbol:

A	A’
0	1
1	0

Truth Table:



Pin

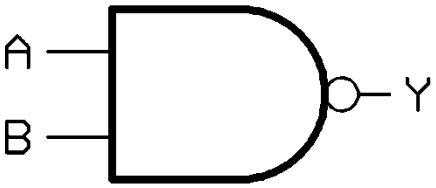
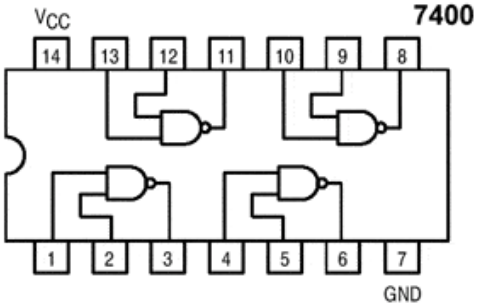


Diagram:

Timing Diagram:



NAND GATE: It is the combination of AND and NOT basic gates.

NAND = AND + NOT.

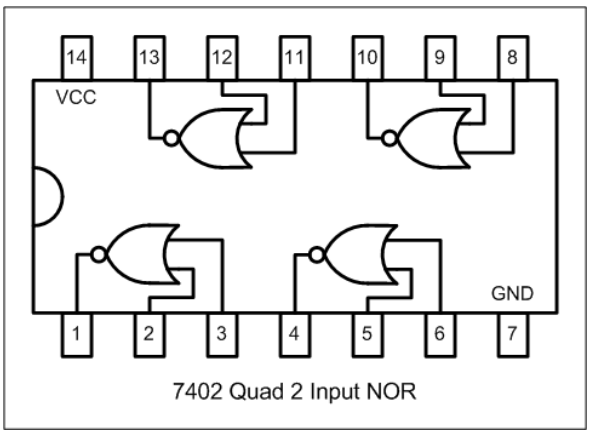
Symbol:

Truth Table:

A	B	$\bar{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

Pin diagram :

Timing diagram:



It's IC no. is 7400.It is the logic circuit in which output is high when both the inputs are low (or) any one of the input is high.

NOR GATEe: It is the combination of OR and NOT basic gates.

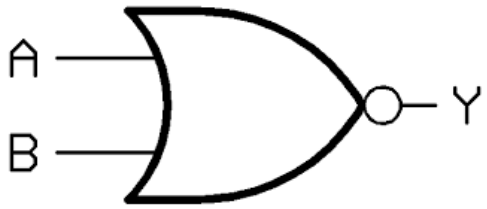
NOR = OR + NOT.

It is a logic circuit in which output is high both the inputs are low and its Ic no. is 7402.

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

Symbol:

Truth Table:

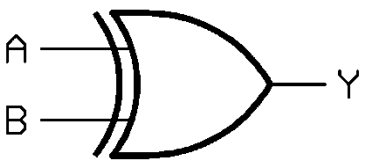


Pin diagram:

Timing Diagram:

EX – OR GATE: It is a logic iruit in which output is high when add no. of inputs arehigh.

Symbol:

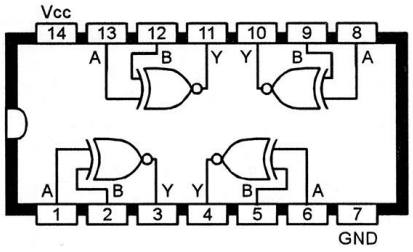


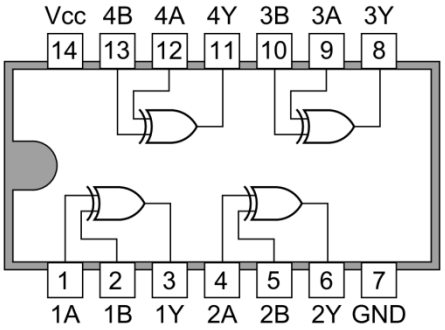
Truth Table:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Pin diagram:

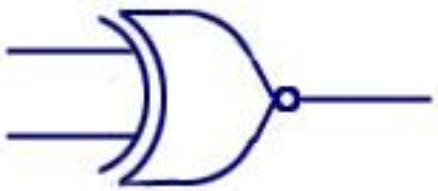
Timing diagram:





EX – NOR GATE: It is aalogic circuit in which output is low when add no. of inputs are high.

Symbol:



Truth table:

A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Pin diagram:

Timing diagram:

State and prove De-Morgans law:

First law: The logic compliment sum of the inputs is equal to the logical compliment product of individual inputs.

Truth Table:

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

$$,', \overline{A+B} = \overline{A}.\overline{B}$$

Second Law:The logical compliment product of inputs is equal to the logical compliment sum of individual inputs.

Turth Table:

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

$\therefore \overline{AB} = \overline{A}+\overline{B}$

Binary to Gray Code Conversion:

1.The most significant bit(MSB) in group code is similar to MSB in binary

2.The second bit in gray code is obtain by adding first bit and second bit in binary and so on.

3. In this sum leave the carry.

Eg:Convert $(1010)_2$ to gray code.

MSB←1 0 1 0➔LSB

1.1

2.1+0=1

3.0+1=1

4.1+0=1

$(1010)_2=(1111)_2$

Gray code to Binary code Conversion:

1.The MSB in binary code is similar to MSB in gray code.

2.The second bit in binary is obtain by adding first bit in binary with the second bit in gray code and so on

3.In this sum leave the carry.

Eg:Convert $(1111)_2$ to binary code

M.S.B ←1 1 1 1→L.S.B

1.1

2.1 + 1=0

3.0 + 1=1

4.1 + 1=0

$=>(1111)_2=(1010)_2$

BCD Addition:

1.Do the adition in column wise.

2.If the 4 bit sum is less than or equal to 9,it is valid sum.

3.If the 4 bit is greater than 9,it is invalid sum add6(0110) to LSB.

Eg: a) $(10001)_2+(10101)_2$

0 0 1 0 0 0 1 1

0 0 0 1 0 1 0 1

---

0 0 1 1 1 0 0 0→valid sum

3 8 = 38

b)$(00100011)_2+(01000111)_2$

0 0 1 0 0 0 1 1

0 1 0 0 0 1 1 1

---

0 1 1 0 1 0 1 0

6 10(A)→invalid sum

0 1 1 0 1 0 1 0

0 1 1 0

---

0 1 1 1 0 0 0 0

7 0 = 70(valid sum)

Excess-3-Code:

It is a binary code like BCD to convert a decimal no. into excess 3 code each decimal digit a 3is added and sum is converted into binary as a group of 4 bits.

Eg: $(12)_{10}$

1 2

3 3

---

4 5 = 0100 0101 => $(12)_{10}$ = (0100 0101)$_2$

Decimal Complimentary Methods:

They are 2 types : 1.9's complimentary method

2.10's complimentary method

9's complimentary method:

9's complimentary method of decimal no. is obtain by subtracting each decimal digit by 9.

Eg: $(23)_{10}$ = 9 9

2 3

---

7 6

Rules:

1.Find 9's complimentary of subtrahend

2.Add this 9's to minuend.If there is no carry the subtraction is possible.Round the carry bit and add to L.S.B.

3.If there is no carry the subtraction is negative recompliment the result and attach a negative sign before it

Eg:Subtrat 89 to 23

89→minu end 9 9

23→subtrahend 2 3

---

8 9 7 6

7 6

---

carry←1 6 5 => 6 5 + 1 =66

10's complimentary method:

9's + 1

Eg:find 10's complimentary of $(23)_{10}$

9 9

2 3

---

7 6 +1

---

7 7→10's complimentary of 23

Rules:

1.Find the 10's compliment of subtrahend.

2.Add this 10's compliment to minu end

3.With there is carry the subtraction is positive ceoss out the carry and remaining bit give the result.

3.Iftheir is no carry the subtraction is negative recompliment the result and attach a negative sign before it.

Eg:$(23)_{10}$-$(89)_{10}$

23→minuend, 89→subtrahend

2 3	9 9	9 4
1 1	8 9	3 4
-----	------	------
carry<--0 3 4	1 0+1=11	6 5 + 1 = 66

$(2\ 3)_{10}$-$(89)_{10}$= -66 (recomplimentary function)

BOOLEAN ALGEBRA:

A set with no. of elements specified by braces A= {1,2,3,4} i.e., the elements of set A or numbers 1,2,3,4.Consider a relation A*B=C. In this '*' is a binary operator to find the values of C such that A,B belongs to S while C does not belongs to S.

Commutative Law:-

Binary operator '*' on set S is said to be commutative,

x * y = y * x (x + y = y + x) for all x,y belongs to S.

Identity law:-

A set S is have an identity element w.r.t binary operator 'x' on S,

x * e = e * x = x.

Associative Law:-

(x + y) + z = x +(y + z) for all x,y,z belongs to S.

(x * y) * z = x *(y * z) for all x,y,z belongs to S.

Distributive Law:-

x + (y + z) = (x + y) * (x + z) for all x,y,z belongs to S. x * (y z) = (x * y) * (x * z) for all x,y,z belongs to S.

Axiomatic Definition of Boolean algebra:-

In 1854 George Boolean introduce a systematic treatment of logic for algebra.

In 1907 E.V.Huntington defined on set of elements together with two operators '+' and '*'.

In 1938 C.E.Shannon introduce two valued Boolean algebra.

Boolean Vs Arithmetic:-

Comparing ordinary algebra with Boolean algebraHuntington doesn't include associate law. The distributive law over x + (y*z)=(x+y)*(x+z) is valid for Boolean not for ordinary algebra.Boolean algebra don't have additive or multiplicative inverse.

The two value Boolean algebra is operate on set of elements two(1,0) and it can satisfies six Huntington postulates.

AND Gate OR Gate NOT Gate

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

A	A'
0	1
1	0

The distributive law x(y+z)=xy+yz are derived from truth table.

x	y	z	y+z	x(y+z)	xy	xz	xy+yz
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	1	1
1	1	1	1	1	1	1	1

From the compliment table, => x+x'=1 since 0+0'=1(0+1=1)

x*x'=0 since 0*0'=0(0*1=0)

Canonical and standard form:-

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

- A Binary variable may appear either in it's normal form(x) or in its compliment form(x').
- Now consider variables x and y combine with an AND operator.
- There are four possible combinations. They are x'y',xy',x'y,xy.
- Each min term is obtain from an 'AND' term of n variables with each variable mean prime with corresponding binary no. '0' and unprime binary no. '1'.
- Again x and y are combine with an OR operator.
- There are four possible combinations. They are x+y,x'+y,x+y',x'+y'.

	Min Terms		Max Terms	
x y z	Terms	Designation	Terms	Designation

- In this each max term is obtain from an OR operator of n variables mean unprime with an corresponding binary no. is '0' 0r prime with no. is '1'.
- Each max term is the compliment of corresponding min term.
- In a Boolean function the min terms are which produce 1 in a function by taking OR of all the terms.
- A symbol each min term as a form m_j where 'j' denotes decimal equivalent of binary number.

0	0	0	x'y'z'	m_0	x+y+z	M_0
0	0	1	x'y'z	m_1	x+y+z'	M_1
0	1	0	x'yz'	m_2	x+y'+z	M_2
0	1	1	x'yz	m_3	x+y'+z'	M_3
1	0	0	xy'z'	m_4	x'+y+z	M_4
1	0	1	xy'z	m_5	x'+y+z'	M_5
1	1	0	xyz'	m_6	x'+y'+z	M_6
1	1	1	xyz	m_7	x'+y'+z'	M_7

Eg: The function taking a combination (1,4 ,7) has the min term F_1=(m1 m4 m7)=(x'y'z', xy'z', xyz)

The compliment of F_1 can be read as F'=(0, 2, 3, 5, 6) = $M_0 M_2 M_3 M_5 M_6$ is the product of max terms = (x + y+ z).

Boolean function can be expressed as sum of Min terms (or) product of Max terms is said to be canonical form.

Sum of Min Terms:

The sum of min terms from can be extended by using one or more variables it is ended with an expression x+x'.

Eg: F=A+B'C

=A (C+C') +B'C(A+A')

=AC+AC'+AB'C+A'B'C

=AC (B+B')+AC'(B+B')+AB'C+A'B'C

=ABC+AB'C+ABC'+AB'C'+AB'C+A'B'C

=ABC+AB'C+ABC'+AB'C'+A'B'C

$=m_7+m_5+m_6+m_4+m_1$

The above expression can be represented in the short notation as sum of Min terms

$F(A,B,C)=\sum(m_7+m_5+m_6+m_4+m_1)$

F(A,B,C)=∑(1,4,5,6,7)

Product of Max Terms:

To express the Boolean function product of Max terms it must be brought into form of OR terms and done by using Distributive law. In this we ended with an expression xx'.

Eg: F=(x+y)(x+z)

=(x+y+zz')(x+yy'+z)

=(x+y+z) (x+y+z')(x+y+z)(x+y'+z)

=(x+y+z)(x+y+z')(x+y'+z)

=M_0, M_3, M_2

,', F(A,B,C)=∏(0, 2 ,3)

Conversion between canonical forms:

In Boolean function, the sum of min terms can be represented as F(A,B,C)=∑(1,4,5,6,7)

By complimenting the function F_1 and by using Demorgan's law we can convert the sum of min terms to the product of max terms (or) vise-versa

,',F^l(A,B, C)=∏(0, 2, 3)

,', it holds a relation

$$m_j^1 = M_j$$

BOOLEAN BASIC THEOREMS AND POSTULATES OF BOOLEAN ALGEBRA:

In Huntington postulates has been listed in pairs designated as a path. One path may be obtained from the other. If binary operator (or) identity operator interchange. This principal is known as DUALITY principle. The dual of algebraic expression is obtained by interchanging OR and AND operators and replace 0's by 1's and 1's by 0's. In this we are having six theorems and 4 postulates. The theorems and postulates are gives the relationship in Boolean algebra "The postulates are basic axioms of algebraic structure and need no proof but the theorems must be proved by postulates".

Postulates:

Postulate 2: a)x+0=x b)x.1=x

Postulate 3(commutative law): a) x+y=y+x b)xy=yx

Postulate 4(distributive law): a) x(y+z)=xy+xz , b)x+yz=(x+y)(x+z)

Postulate 5: a)$x+x^l=1$ b)$x. x^l=0$

Theorems:

Theorem 1: a)x+x=x b) x.x=x

Theorem 2: a)x+1=1 b)x.0=0

Theorem 3: $(x^l)^l=x$

Theorem 4(associative law): a)x+(y+z)=(x+y)+z , b)x(yz)=(xy)z

Theorem 5(demorgan's law): a)$(x+y)^l=x^ly^l$, b)$(xy)^l=x^l+y^l$

Theorem 6(absorptive law): a)x+xy=x , b)x(x+y)=x

Theorem 1(a): x+x=x by postulate

x+x=(x+x).1 2(b)

=(x+x)(x+x) 5(a)

=x+x.x 4(b)

=x+0 5(b)

=x 2(a)

Theorem 1(b): x.x=x by postulate

x.x= x.x+0 2(a)

= x.x+ $x.x^l$ 5(b)

=$x(x+x^l)$ 4(a)

=x.x 5(a)

=x 2(b)

Theorem 2(a): x+1=1 by postulate

x+1=1.(x+1) 2(b)

=$(x+x^l)$ (x+1) 5(b)

=$(x+x^l)$.1 4(b)

=1 5(a)

Theorem 6(a): x+xy=x by postulate

x+xy=x.1+xy 4(a)

=x(1+y) 3(a)

=x.1 2(a)

=x 2(b)

Theorem 2(b): x.0=0 by duality

Theorem 6(b): x(x+y)=x by duality

Operator procedure:

For evaluating Boolean expression a) parenthesis b) AND, OR,NOT. These expressions inside the parenthesis must be evaluated before operations.

Ex: Demorgan's law: (i) $(A+B)^1=A^1.B^1$

In the above expression, the first side solved by using OR operation within parenthesis and then complimented. The x and y are individually complimented then ANDED.

Basic function:

A Boolean expression formed with binary variables. The binary operators OR, AND and NOT. If a function $F_1=xyz^1$, the $F_1=1$ if x=1,y=1,z^1=1 otherwise $F_1=0$. These values are represented in below truth

table. Consider the second function $F_2 = x + y^1 z$. Consider again the function $F_3 = x^1 y^1 z + x^1 yz + xy^1$, $F_4 = xy^1 + x^1 z$ in both function F_3 and F_4 we are having four 0's and four 1's are shown in below truth table.

x	y	z	x^1	y^1	z^1	F_1	F_2	F_3	F_4
0	0	0	1	1	1	0	0	0	0
0	0	1	1	1	0	0	1	1	1
0	1	0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0	1	1
1	0	0	0	1	1	0	1	1	1
1	0	1	0	1	0	0	1	1	1
1	1	0	0	0	1	1	1	0	0
1	1	1	0	0	0	0	1	0	0

Compliment of the function:

The compliment of a function F is F^1 and it is obtained by interchanging 0 for 1 and 1 for 0. The compliment of function will be algebraically proved then Demorgan's laws can be extended to 3 or more variables.

Eg: $(A+B+C)^1 = (A+X)^1$

$= A^1.X^1$

$= A^1.(B+C)^1$

$= A^1.B^1.C^1$

,', $(A+B+C.....F)^1 = A^1.B^1.C^1.....F^1$

Combinational logic circuit: The logic circuit formed by combination of various gates. It is known as combinational logic circuit.

Eg: adders, subs tractor, encoders, decoders, multiplexers, de-multiplexers, parallel binary adder.

In this property logic circuit, present output depends on present input it has no memory.

Adders: It performs binary addition and it is divided into 3 types

1. Half adder

2. Full adder

3. parallel binary adder

Half adder: It is a logical circuit which adds two binary digits at a time and produce two outputs known as sum and carry. Half adder contains 2 inputs (A&B) and 2 outputs(C&S)

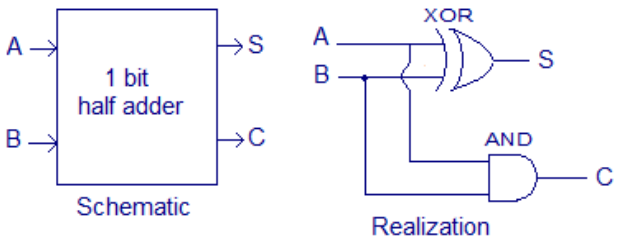
By taking fundamental products from T.T, the logical expression for sum of products and carry is

$S=\bar{A}B+A\bar{B}$ =A+B; C=AB

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half adder circuit can be designed by using X-OR and AND GATE.

Full adder: It is a logical circuit which adds 3 binary digits at a time and produces 2 outputs known as carry and sum



Full adder contains 3 inputs (A B C) and 2 outputs (C&S)

Truth table:

A	B	C	S	C
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

By taking fundamental products from the T.T, the logical expression for sum of products and carry is

$S=\bar{A}\bar{B}C+\bar{A}B\bar{C}+A\bar{B}\bar{C}+ABC$

$=A\oplus B\oplus C$

$C=\bar{A}BC+A\bar{B}C+AB\bar{C}+ABC$

$=\bar{A}BC+A\bar{B}C+AB(\bar{C}+C)$

$=\bar{A}BC+A\bar{B}C+AB$

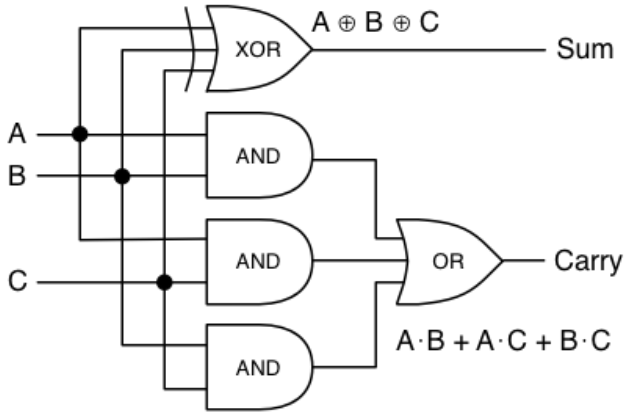
$=\bar{A}BC+A(\bar{B}C+B)$

$=\bar{A}BC+A(C+B)$

$=\bar{A}BC+AC+AB$

$=C(\bar{A}B+A)+AB$

$=C(B+A)+AB$



=CB+CA+AB

Full adder can be constructed by using X-OR and OR GATE

Parallel binary adder: It is a logical circuit adds two binary digits, is known as parallel binary adder

It is a combination of full adder with one half adder

Consider a four bit binary adder which contains 4 bits

The L.S.B column contains two bits Ao and Bo to add this two bits half adder may be used (or) full adder is also used by connecting 3rdinput to ground. Half adder is used to produce sum& carry (so&co). co acts as input to the next F.A. It contains 3 inputs A1,B1&CO and produce output S1&C1. In similar way the next F.A contains A2,B2&C1 and produce output S2&C2.

	A3	A2	A1	Ao
C3	B3	B2	B1	BO
C3	S3	S2	S1	SO

The MSB full adder has inputs A3,B3&C2 and produce outputs S3&C3. Therefore final sum of 4 bit & parallel binary adder is C3 S3 S2 S1 SO.

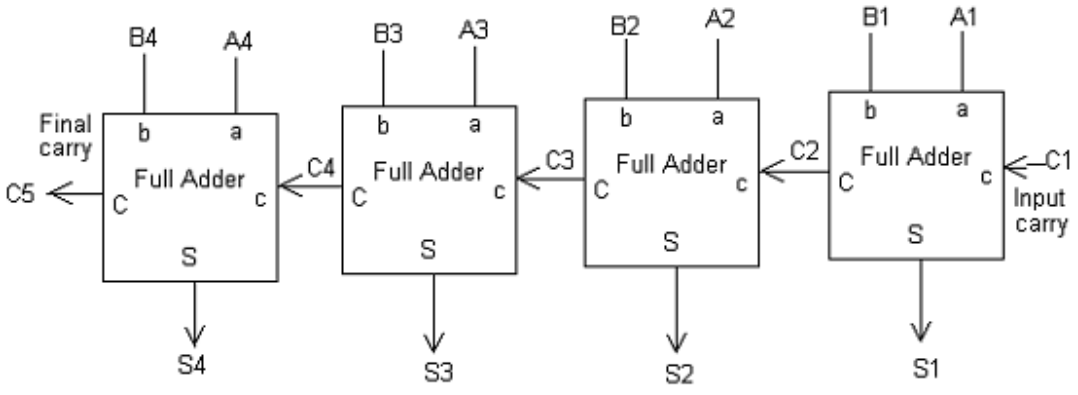
Subtractor:

It is used to subtract the binary numbers and are of two types.

1) Half Subtractor 2) Full Subtractor

Half subtract or:

It is a logic circuit which subtracts two binary no's at a time and produce o/p is known as difference and barrow.

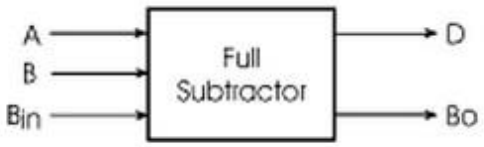


Truth table:

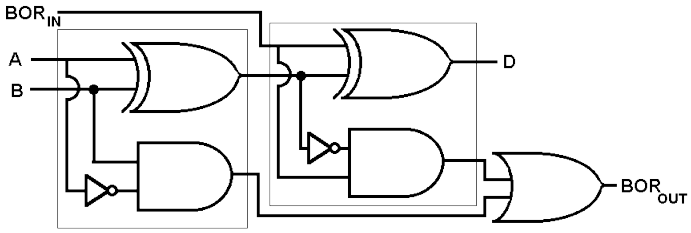
symbol:

A	B	D	B

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



Logic circuit:



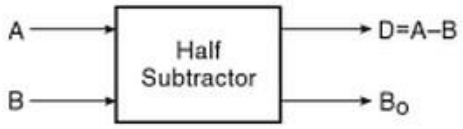
The logic expression for difference and barrow is

$$D = A \oplus B \ , \ B = \bar{A}B$$

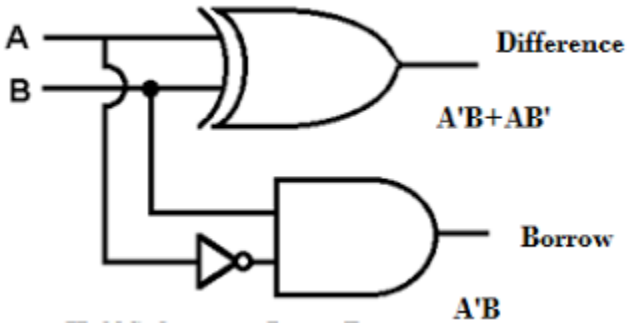
Full Subtractor:

It is a logic circuit which subtracts binary no's at a time and produce output difference and barrow.

Symbol:



Logic circuit:



Truth table:

The logic expression for difference and barrow are

$D = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$ and

$$B = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$$

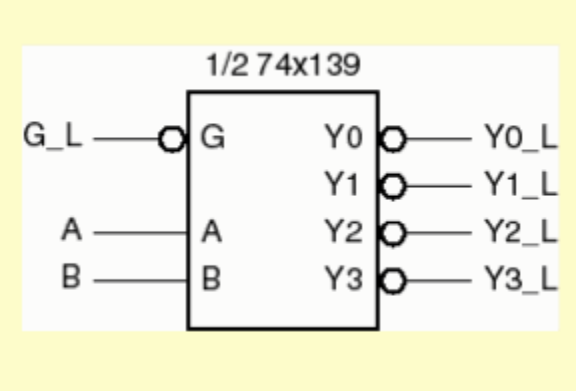
A	B	C	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Decoder:

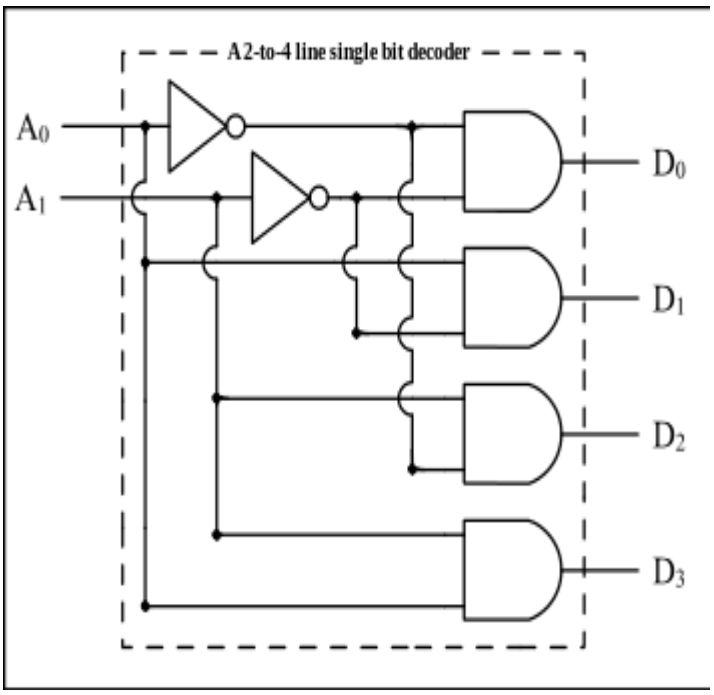
It is a combination logic circuit that converts binary information of "n" i/p lines in to maximum of 2^n unique o/p lines.

Ex: A 2*4 decoder has shown below having two input lines decoder in to 4 output lines each output represents one of main term of two input variables.

Symbol:



Logic circuit:



Truth table:

A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Encoder:

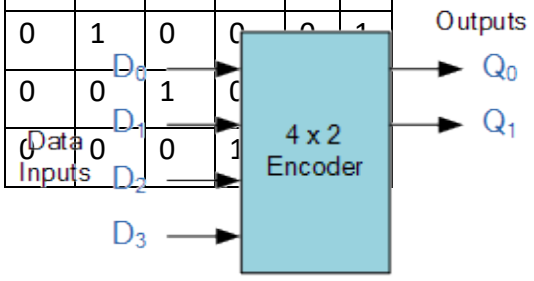
It is used to perform reverse process of a decoder.

- ➔ It is having 2^n input lines and n o/p lines.
- ➔ This form of encoder known as priority encoder.
- ➔ It is constructed with OR Gates.
- ➔ In the below truth table D_2 and D_3 are logic one that D_3 has highest priority than D_2 .

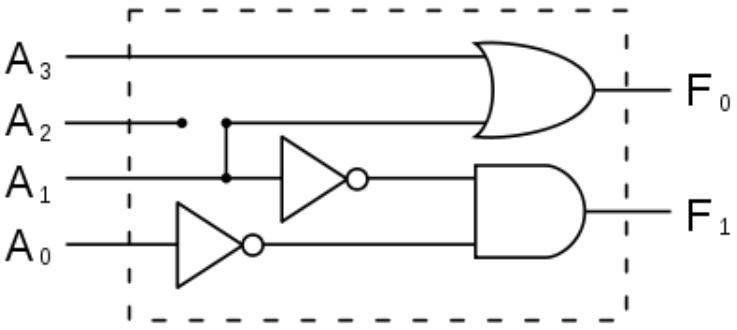
Symbol:

Truth Table:

D_0	D_1	D_2	D_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



Logic circuit:



Logic expression:

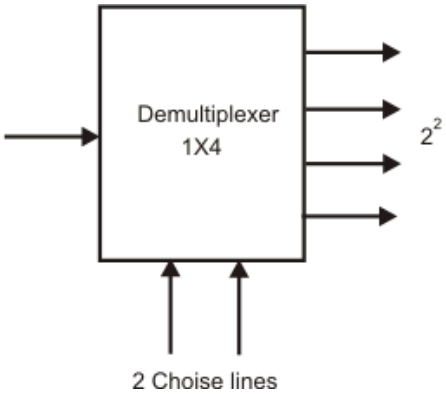
$$A = D_2 + D_3$$

$$B = D_1 + D_3$$

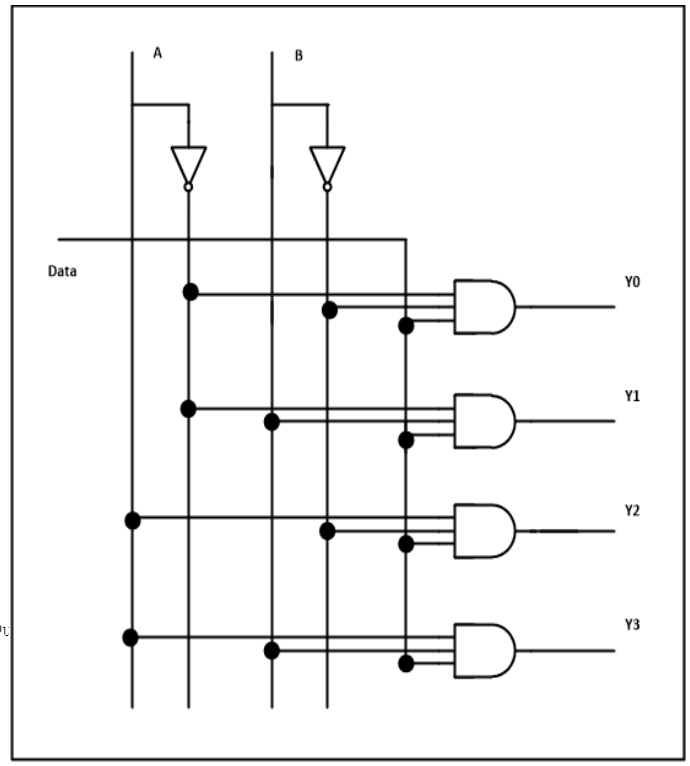
Multiplexer:

It is used to select binary information from one of many input lines and directly used single input line the selection of particular input line is controlled by set of selection line the selection line S_1 , S_0 are to select the particular 'AND' Gate the output of AND Gate is given to 'OR' Gate to produce one output line. It is also known as data select line.

Symbol:

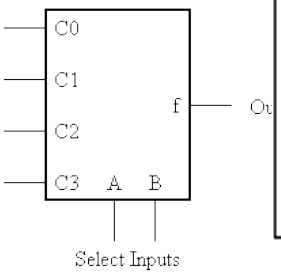


Logic circuit:



Truth table:

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

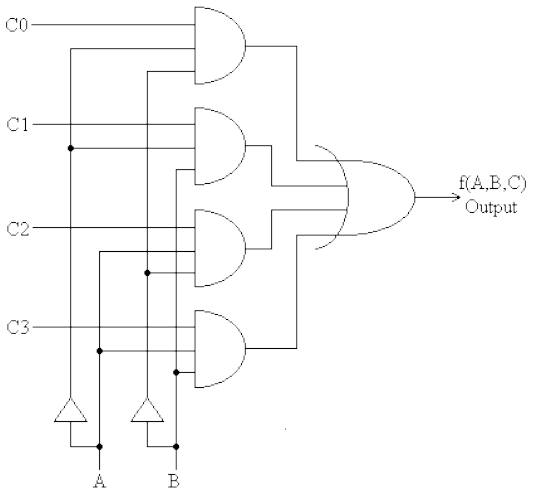


De-Multiplexer:

A decoder with enable input function act as de-multiplexer. It is a circuit that receives information from a single line and directs into number of 2^n possible output line.

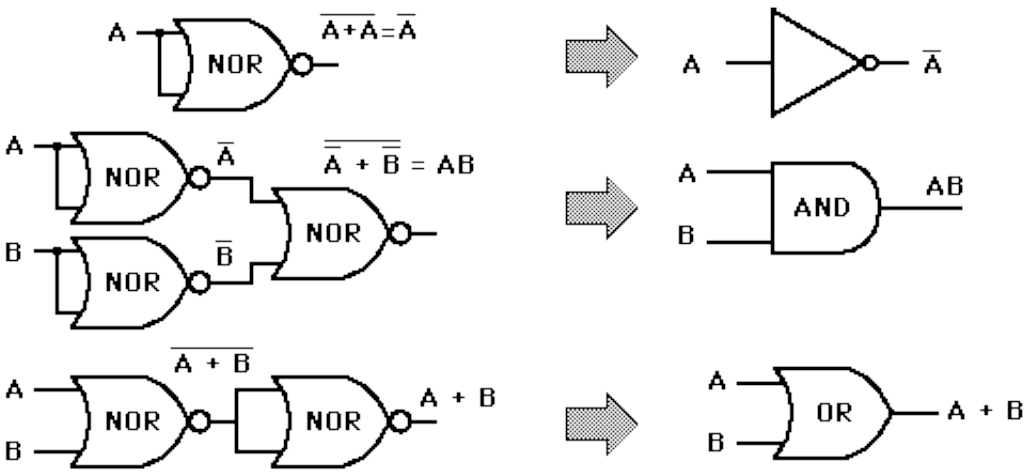
EX: If selection line AB = 10 the output 'D' will be same as input value f by while other outputs are maintained at one.

Symbol: Logic circuit:



Truth Table:

E	A	B	D_0	D_1	D_2	D_3
0	X	X	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1



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

Multi level NAND Gate:

Combinational logic circuit is frequently constructed with NAND and NOR rather that and AND, OR , and NOT Gates.

NAND and NOR are available in integrated circuit from any digital system can be implemented with them.

Boolean Function Implementation:

The implementation of Boolean function is with NAND may be obtained by means of simple block diagram techniques.

In the given algebra function expression draw the logic diagram by using AND, OR , and NOT.

Draw the second logic diagram with equivalent of NAND logic for each AND, OR , and NOT.

Remove any two cascade inverters from the diagram since double inversion donot perform any logic function.

Multilevel NOR Gate:

NOR gate is also a universal gate in this any digital system can be implemented with them and a flip flop can be designed by using NAND and NOR Gates.

Boolean Function Implementation:

The implementation of Boolean functions with NOR may be obtained by means of simple block diagram techniques.

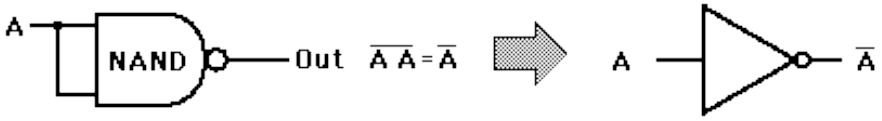
In the given algebra function expression draw the logic diagram by using AND, OR , and NOT.

Draw the second logic diagram with equivalent of NOR logic for each AND, OR , and NOT.

Remove any two cascade inverters from the diagram since double inversion donot perform any logic function.

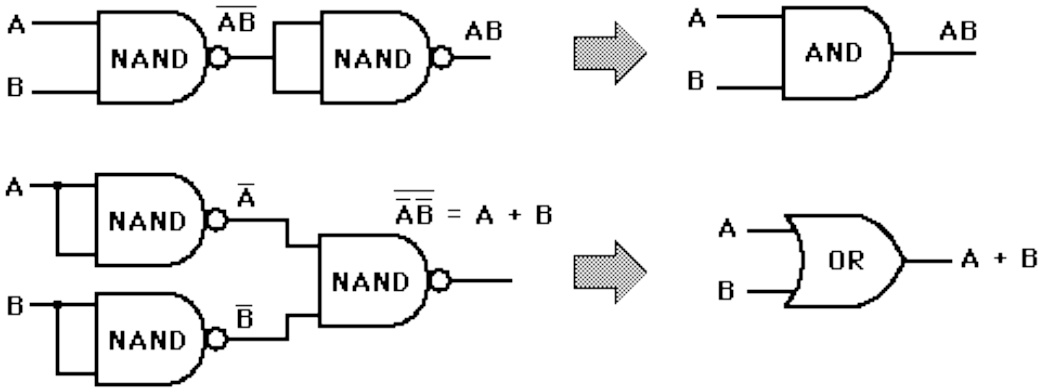
K-MAP Method:

The map method was first discovered by veitch and later modified by karnaugh. The map is made up of square and each square is made up of min term.



Two variable map method:

In two variable map method there are four min term hence the map consists of 4 squares for min terms the zeros and ones are marked to each row and column with the values of X and Y respectively.



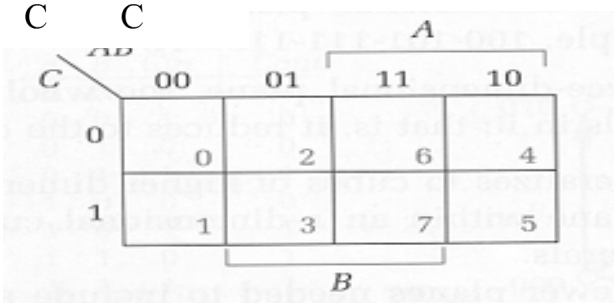
1) X appears primed in row Zero and unprimed in row 1.
2) Similarly Y appears column primed in zero and unprimed in 1.

	A	A	$\overline{A}$	$\overline{A}$	
B	ABCD	AB$\overline{C}$D	$\overline{A}$B$\overline{C}$D	$\overline{A}$BCD	D
B	ABC$\overline{D}$	AB$\overline{C}\overline{D}$	$\overline{A}$B$\overline{C}\overline{D}$	$\overline{A}$BC$\overline{D}$	$\overline{D}$
$\overline{B}$	A$\overline{B}$C$\overline{D}$	A$\overline{B}\overline{C}\overline{D}$	$\overline{A}\overline{B}\overline{C}\overline{D}$	$\overline{A}\overline{B}$C$\overline{D}$	$\overline{D}$
$\overline{B}$	A$\overline{B}$CD	A$\overline{B}\overline{C}$D	$\overline{A}\overline{B}\overline{C}$D	$\overline{A}\overline{B}$CD	D
	C	$\overline{C}$	$\overline{C}$	C	

Three variable map method:

In three variables they are having 8 min terms and map consists of 8 squares.

Eg: The square assigned to m_5 is xy^1z corresponding to row 1 and column 01 the two numbers are concentrated to give the binary no:101 and decimal equivalent no is 5.



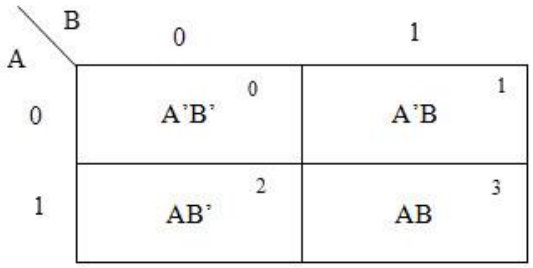
Any two adjacent squares in map refer only one variable which is primed in one and unprimed zero(0).

Eg: m_5 and m_7 are lying in adjacent square in this variable y is primed in m_5 and unprimed in m_7. The sum of two min terms can be simplified in to a single term.

yz / x	00	01	11	10
0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'

Four variable map method:

The map consists of four binary variables and listed as 16 min terms and each square are assigned each min term. In this the rows and columns are numbered in reflected code sequence with only one changing the value between two adjacent squares of rows and columns.



Eg: the number third row 1, 1 and column 2 is 0,1 concentrated to give binary number 1101 and binary equivalent is m_{13}.

- First represent one min term crimping a term of four literal.
- Two adjacent squares represented term of three literals.
- Four adjacent squares represented term of two literals.
- Eight adjacent squares represent term of one literal.

	C'D'	C'D	CD	CD'
A'B'	0	1	3	2
A'B	4	5	7	6
AB	12	13	15	14
AB'	8	9	11	10

Don't care condition:

The logical sum of min terms associated with a Boolean function specifies the condition under which the function is equal to zero for rest of min terms which assumes all the combination of values for the variables of the function are valued. It should be released that don't care condition can be used on a map to provide further simplification of Boolean expression to distinguish the don't care condition from zeros and ones an 'X' is used. The 'X' inside a square in the map indicate that we don't care weather the value zero (or) one assigned to 'F' for a particular min term.

Eg:

Simplifying the Boolean function.

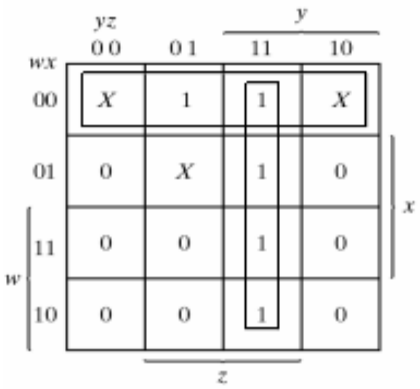
$$f(w,x,y,z) = \Sigma(1,3,7,11,15)$$

A don't care condition d(w,x,y,z) = sigma(0,2,5) the min terms in 'F' are the variable combination that make the function is equal to 1 the mean terms D(or) don't care min terms that may be assigned either zero (or) one. In this 'D' marked as 'X' and 'F' marked as '1' and remaining squares are filled with zeros.

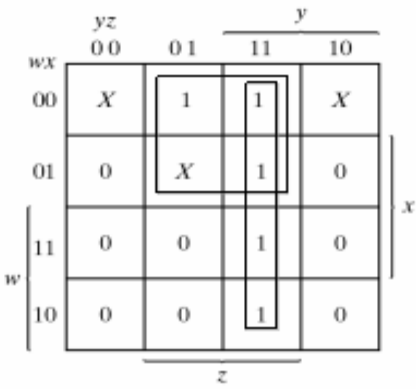
i) $m_0 + m_1 + m_3 + m_2 = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D}$

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

$= \bar{A}\bar{B} = \bar{w}\bar{x}$



(a) $F = yz + w'x'$



(a) $F = yz + w'z$

$m_3 + m_7 + m_{15} + m_{11} = \bar{A}\bar{B}C\bar{D} + \bar{A}BCD + ABCD + A\bar{B}CD$

$= \bar{A}CD + ACD$

$= CD = yz$

$$F = \bar{w}\bar{x} + yz$$

ii) $m_1 + m_3 + m_5 + m_7 = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BCD$

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

$$= \bar{A}D = \bar{w}z$$

$$m_3 + m_7 + m_{15} + m_{11} = \bar{A}\bar{B}CD + \bar{A}BCD + ABCD + A\bar{B}CD$$

$$= \bar{A}CD + ACD = CD = yz$$

$$F = \bar{w}z + yz$$

In the above example tha map are initially marked with 'x' are considered as zero (or) one. The two simplified expressions can be obtained as

$$F(w, x, y, z) = \Sigma(0,1,2,3,711,15) = \bar{w}\bar{x} + yz$$

$$F(w, x, y, z) = \Sigma(1,3,5,7,11,15) = \bar{w}z + yz$$

In both expressions (1,3,7,11,15) make the function F=1 the don't care min terms 0,2,5 are treated differently in each expression. In the first expression '0' and '2' with '1' and leaves the min term '5' with '1' and leaves the min term zero the two expression represents two functions algebraically they are unequal.

,', it is possible to obtain product of sum expression for the function '0' and '2' which gives the simplified complement function. F'=z'+wy' taking the compliment of F' we get product of sums.

,', f(wxyz) = z(w'+y)= $\Sigma(1,3,5,7,11,15)$

In this case min term in to with zeros and the min term '5' with one's.

IC-DIGITAL LOGIC FAMILIES:

It is a group of compactable device having some logic levels and supply voltages. Compactability means we can connect the output of one circuit to the input of another circuit without any interfacing circuit. Generally logic families are designed by combining a group of Boolean functions and it is of two types Bi-polar logic families and Mos logic families.

I)Bi-polar logic families:

In these BJT are used and having SSI (Small Scale Integration <10, MSI(Medium Scale Integration)<100, LSI (Large Scale Integration) < 1000 and VLSI (Very large Scale Integration)< 10000.

Eg: RTL, DTL, TTL, ECL.

RTL (Resistor Transistor Logic):

It is used to provide high performance and low noise margines.

DTL (Diode Transistor Logic):

It provides better noise margine, good fanout, and these are slow in speed of operation.

TTL (Transistor Transistor Logic):

It provides greater operating speed, good fan in and fan out, easy interface with any circuit.

ECL (Emitter coupled Logic):

It provides highest switching speed for any logic state.

II)Mos families:

In this p-channel Mosfet are used. It is the oldest and slowest logic circuit.

In this n-channel Mosfet are used. Digital IC'S are designed by using N-mos and used in Micro-Boosters and Memories.

C-Mos families:

It is a combination of both p-channel and n-channel and it is used to provide low power displations. Generally, IC designed by using C-Mosare digital wrist watches, pocket calculators.

SOLVE THE BOOLEAN EXPRESSIONS:

i) $y = (A+B)(A+\bar{B})(\bar{A}+C)$

$= (AA + A\bar{B} + AB + B\bar{B})(\bar{A}+C)$

$= (A + A\bar{B} + AB + 0)(\bar{A}+C)$

$= (A + A(\bar{B}+B))(\bar{A}+C) = (A+A)(\bar{A}+C) = A(\bar{A}+C) = (A\bar{A} + AC) = (0 + AC) = AC$

ii) $y = AB + ABC + AB + A\bar{B}C$

$= AB + ABC + A\bar{B}C$

$= AB + AC(B+\bar{B})$

$= AB + AC$

$= A(B+C)$

iii) $y = (AB+C)(AB+D)$

$= (AB + ABD + ABC + CD)$

$= AB(1+D) + ABC + CD$

$= AB(1) + ABC + CD$

$= AB + ABC + CD$

$= AB(1+C) + CD = AB + CD$

iv) $y = A + \bar{A}B$

$= A(1) + \bar{A}B$

$= A(1+B) + \bar{A}B$

$= (A + AB) + \bar{A}B$

$= A(1) + AB + \bar{A}B$

$= A(\bar{A}+A) + AB + \bar{A}B$

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

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

v) $y = A + \bar{A}B + AB$

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

$= A + B$

vi) $y = (A+B)(A+C)$

$= (A + AC + AB + BC)$

$= (A + AB + AC + BC)$

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

$= A + BC + AC$

$= A(1+C) + BC$

$= A + BC$

vii) $y = \bar{A} + A\bar{B}$

$= \bar{A}(\bar{B}+1) + A\bar{B}$

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

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

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

HEXA DECIMOL TO DECIMOL NUMBER CONVERSION:

To convert a hexa decimal number into decimal number stream line method is used.

Eg:(3 B A)$_{16}$

3 B A

16^2 16^1 16^0

= 3*16^2+11*16+10*1

=3*256+176+10

= (954)$_{10}$

DECIMOL TO HEXA DECIMOL NUMBER CONVERSION:

To convert a decimal to hexa decimal number divided by 16 method is used.

Eg: (2479)

……..

Ans: (9 A F)$_{16}$

HEXA DECIMOL TO OCTAL CONVERSION:

To convert a hexa decimal into octal

i) Convert the given hexa decimal to decimal.

ii) Convert the above decimal number into octal

Eg: $(9\ A\ 6)_{16}$

9	A	6
9	10	6
16^2	16^1	16^0

$= 9*16^2+10*16+6*1$

$=9*256+160+6$

$=2304+160+6$

$=2460$

----------division

CONVET OCTAL NUMBER INTO HEXA DECIMOL:

To convert octal number into decimal number stream line method is used. Convert the decimal number to hexa decimal divided by 16 method is used.

Eg: $(123)_8$

1	2	3
8^2	8^1	8^0

$= 1*64+8*2+1*3$

$=64+16+3$

$=83$

------- division

HEXA DECIMAL ADDITION:

Do the addition column wise. If sum is equal to 15 (or) less than 15 write the corresponding Hexa decimal digit. If sum is greater than 15 subtract 16 from the sum. Write the corresponding hexa decimal digit and put '1' in the next column.

Eg: $(23)_{16}+(3B)_{16}$

2	3
3	B

---

5 E

HEXA DECIMAL SUBTRACTION:

In this the subtraction is done by using 2's compliment method.

Eg: $(24)_{16} - (18)_{16}$

0010 0100 – minu end

0 0 0 1 1 0 0 0 – subtrahend

↓↓↓↓ ↓↓↓↓

1110 0 1 1 1

1

1110 10 0 0

now,

0010 0100

1110 1000

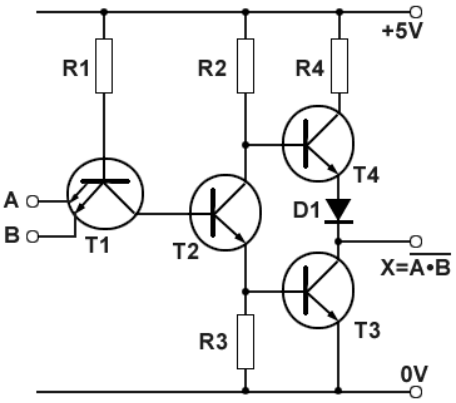
1|0000 1100

0 12

\

TTL NAND GATE:

This circuit consists of multiple emitter transistors 'Q_1'.Here each terminal acts as a diode i.e., transistor Q_1 acts as AND Gate and remaining part of the circuit acts as NOT. The total circuit acts as NAND Gate.



The transistor Q3 and Q4 provides "totem pole connection". The advantage of this connection is to provide low output impedance due to this time constant of the circuit increases. The diode 'D' prevents the transistor Q_3 and Q_4 conduction at the same time and it is used to increase the speed of operation i.e., when Q_3 is on,Q_4 is off and output y is high. When Q_3 is off and Q_4 is on and output y becomes low.

WORKING:

Case 1: When both A and B are low.

Case 2: When A is low.

Case 3: When any one of the input is low, the base to emitter junction Q_1 is forward bias i.e., Q_1 is ON and hence collector current is maximum and collector voltage is '0'.

The voltage $V_{C1}=V_{B2}=0$

Hence the transistor Q_2 becomes OFF , and forces Q_4 to OFF AND Q_3 becomes ON. Therefore, output is high (or) logic '1'.

Case 4: When A and B are high , the Q_1 is reverse bias due to this Q_1 becomes OFF i.e., collector current $IC_1=V_{CC}$

Since $V_{C1}=V_{B2}=V_{CC}$, the transistor Q_2 ON and forces Q_3 to OFF and Q_4 to ON. Hence output y is low (or) logic '0'.

TRUTH TABLE:

ANALOG VOLTAGE		
A	B	Y
0(low)	0(low)	4.33(v)
0(low)	1(high)	4.37(v)
1(high)	0(low)	4.37(v)
1(high)	1(high)	0.47

DIGITAL VOLTAGE		
0	0	1
0	1	1
1	0	1
1	1	0

C_MOS USING NOR GATE:

- Complimentary MOSFET is combination of both p-channel and n-channel MOSFET. The above circuit indicates two inputs NOR Gate using C-MOS logic.
- The p-channel E-MOSFET is ON for logic '0' or negative gate voltage.
- N-Channel E-MOSFET is ON for logic '1' or positive gate voltage.
- Output Y is high when both Q_3 and Q_4 is OFF.
- Output y is low when any one of Q_3 and Q_4 is ON.

Case 1:

When A and B are low.

- Due to low input voltage at A Q_1 is ON and Q_4 is OFF.
- Due to low input voltage at B Q_2 is ON and Q_3 is OFF.
- The electrical switch equivalent circuit is as shown below.
- Therefore, Q_3 and Q_4 is OFF the output voltage is maximum i.e., approximately equal to V_{DD} and hence output is high or at logic '1'.

Case 2:

When A is low and B is high.

- Due to low input voltage at A, Q_1 is ON and Q_4 is OFF.
- Due to high input voltage at B, Q_2 is OFF and Q_3 is ON.
- The electrical switch equivalent circuit as shown in below.
- Q_3 is 'ON' it is ground potential and Q_4 are connected in parallel, hence low potential of Q_3 pulls the high potential to ground potential and hence output is low or logic zero.

Case 3:

When A is high and B is low.

- Due to high input voltage at A, Q_1 is OFF and Q_4 is 'ON'.
- Due to low input voltage at B Q_2 is on and Q_3 is 'OFF'.
- The electrical switch equivalent circuit as shown in below.
- Therefore,Q_4 is ON, it is at ground and Q_3 is OFF, it is at high potential. Both Q_3 and Q_4 are connected in parallel. Hence low potential of Q_4 pulls the high potential Q_3 to ground potential zero. Hence output Y is low or logic '0'.

Case 4:

When both A and B are high.

- Due to high input voltage at A, Q_1 is OFF and Q_4 is ON.
- Due to high input voltage at B, Q_2 is OFF and Q_3 is ON.
- The electrical switch equivalent circuit is shown below.
- Q_3 and Q_4 are ON, both are at ground potential

Therefore, Output Y is low or logic '0'.

SEQUENCITIAL LOGIC CIRCUIT:

It is a combinational logic circuit with feedback i.e., previous output are present in the feedback circuit . Hence feedback circuit is known as Memory element.

In sequential logic circuit present output depends on present input and also previous output

Eg: Flip-flops, Registers, Counters.

Sequential logic circuits are of two types

a) Synchronous

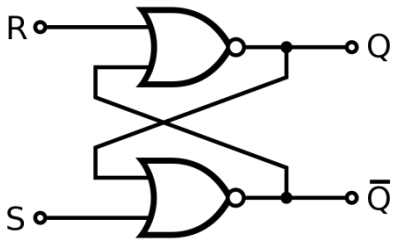
b) Asynchronous

a) Synchronous:

The output logic circuit state was controlled by an external input signal known as clock. Hence these are known as Synchronous (or) clocked (or) timed sequential logic circuits.

b) Asynchronous:

The output logic state depends on previous output but not on clock pulse.



Flip Flop:

Generally a clocked latch is known as Flip-flop. Latch means a bistable multivibrator which stores either '0' bit (or) '1' bit.

Flip flop is a single bit memory storage element and it has two outputs. The basic property of flip-flop is one output must be compliment to other ($Q = \bar{Q}$)

Eg: R-S flip-flop, j-k flip-flop, D flip-flop, T flip-flop, Master_Slaveflipflop.

R-S FLIP FLOP:

It is the basic sequential circuit in flip flop. Here R means RESET and S means SET.

If output Q=0 then flip flop is said to be in reset condition.

If output Q=1 then the flip flop is said to be in set condition.

WORKING:If input of first NOR Gate R and $\bar{Q}$ then input of second NOR Gate is S and Q.

Case 1: When R=0 and S=0 and assume previous outputs are Q=0 and $\bar{Q}$ =1

- Input of first NOR Gate is R=0 and $\bar{Q}$ =1.
- Output of first NOR Gate is $Q_{n+1} = 0 = Q$.
- Input of second NOR Gate is S=0 and Q=0.
- Output of second NOR Gate is $Q_{n+1} = 1 = \bar{Q}$.

,', Output Q_{n+1} remains in its previous output state (Q_n) i.e., no change

Case 2: When R=0 and S=1 assume previous outputs are Q=0 and $\bar{Q}$ =1(Q_n)

- Inputs of first NOR Gate is R=0 and $\bar{Q}$ =1.
- Output of first NOR Gate is Q_{n+1}=0=Q.
- Input of second NOR Gate is S=1 and Q=0.
- Output of second NOR Gate is $Q_{n+1} = 0 = \bar{Q}$
- Again input of first NOR Gate is R=0 and $\bar{Q}$ =0
- Output of first NOR Gate is $Q_{n+1} = 1 = \bar{Q}$
- Input of second NOR Gate is S=1 and Q=1.
- Output of second NOR Gate is $Q_{n+1} = 0 = \bar{Q}$.

,', $Q_{n+1} = 1$ flip flop is said to be in set condition.

Case 3: When R=1 and S=0

- Input of first NOR Gate is R=1 and $\bar{Q}$ =1.
- Output of first NOR Gate is $Q_{n+1} = 0 = \bar{Q}$
- Input of second NOR Gate is S=0 and Q=0.
- Output of second NOR Gate is $Q_{n+1} = 1 = \bar{Q}$.

,', output Q_{n+1} =0 then the flip flop to be in reset condition.

Case 4: When R=1 and S=1

- Input of first NOR Gate is R=1 and $\bar{Q}$ =1.
- Output of first NOR Gate is $Q_{n+1} = 1 = Q$.
- Input of second NOR Gate is S= and Q=0.

- Output of second NOR Gate is $Q_{n+1} = 0 = \bar{Q}$.

At this condition, the flip flop evaluate the basic property i.e., $Q \neq \bar{Q}$. Hence it is known as Forbidden (or) invalid condition.

R-S flip flop using NAND:-

It is a basic sequential logic circuit in flip flop 'R' means reset and 's' means set. If output Q=1,the flip flop is said to be in set condition if output Q=0 then i/p is said to be in reset condition.

Working:-

- If i/p of first NAND gate is R & ~Q and i/p of a second NAND gate is S and Q.
- When R=0 & S=0 Assume previous output are Q =0,~Q=1 (Qn)
- If i/p of 2nd NAND is S=0,Q=0 o/p of NOR gates is Qn+1=Qn=1
- If i/p 2nd NAND is s=0,Q=0 output of NOR is Qn+1=1=~Q

The output Qn+1 remains in the previous output state Qn that is invalid.

Case(i) :When R=0,S=1 the i/p of NAND is R=0 & Q=1 then o/p of NAND gate is Qn+1=!=~Q

- Input of 2nd NAND gate is s=1& Q+0 then O/p of NAND gate is Qn+1 =1=~Q

Case(ii):

When i/p of 1st NAND gate is R=0,a+0 then o/p of NAND gate is o/p of NAND gate is Qn+1=1=Q.

- When S=1,Q=0 then o/p of NAND gate is Qn+1=1=~Q.
- The output Qn+1=0 & i/p is said to be "SET".

Case(iii): When R=1,S=0

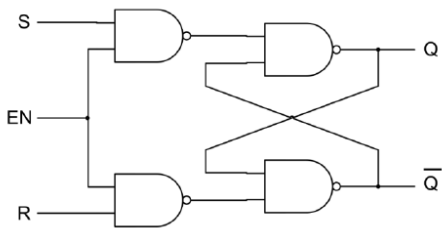
- The i/p of 1st NAND gate is R=1,~Q=1 then output of NAND gate is Qn+1=0=~Qn
- The output of 2nd NAND gate is S=0,Q=0 then o/p of NAND gate is Qn+1=1=~Qn
- When o/p Qn+1=1 and i/p is said to be in reset condition.

Case(iv): When R=1,S=1

- The input of 1st NAND gate is R=1,~Q=1 then the o/p NAND gate is Qn+1=0=Q
- If i/p of 2nd NAND gate is S=1,Q=0 then o/p of NAND gate is Qn+1=~Q=1
- When o/p Qn+1=1 and flip-flop is said to be reset condition.

Truth table:

Clk	R	S	OUTPUT	ACTION
↓	X	X	Q_n	NC
↑	0	0	Q_n	Invalid
↑	0	1	1	Set
↑	1	0	0	Reset
↑	1	1	1	NC



Clocked R_S Flip flop using NOR Gate:

Generally flip flops are said to be transparent. Any change in input is immediately transmitted by output logic state. In order to hold the output logic state for some duration of time we are using a external trigger pulse known as clock pulse. Hence this flip flop is known as clocked RS FLIP FLOP.

By the addition of two AND gates the flip flop may be enable or disable by using external control signal known as Clock. When clock input is low output of both AND gates is low then the flip flop remains in its previous state irrespective of inputs R and S. At this condition, the flip flop is disable. When clock is high, the change in input is transmitted into the output. At his condition flip flop is enable.

Case 1: When clock is high, R=0,S=0 the output $Q_{n+1} = Q_n$ (or) remains in its previous output state.

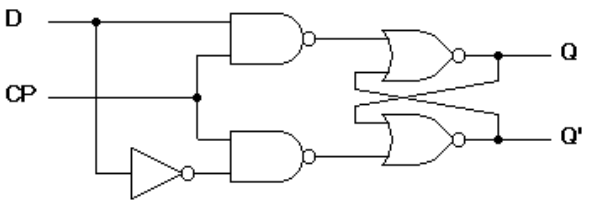
Case 2: When clock is high, R=0,S=1, then the output $Q_{n+1} = 1$ (or) remains in its Set condition.

Case 3: When clock is high, R=1, S=0 then the output is $Q_{n+1} = 0$ (or) remains in its Reset condition.

Case 4: When clock is high, R=1 S=1 the output is $Q_{n+1} = ?$ (or) it is in invalid condition.

Truth table:

Clk	R	S	OUTPUT	ACTION
↓	X	X	Q_n	NC
↑	0	0	Q_n	NC
↑	0	1	1	Set
↑	1	0	0	Reset
↑	1	1	1	Invalid



R-S-D FLIP FLOP:

R-S flip flop contains two data inputs known as Reset and Set. If S=1, The flip flop is in SET condition. If R=0, Then the flip flop is said to be in RESET condition. In some applications, it is difficult to generate two data input signals to drive the outputs. Hence , the two inputs are converted as single flip flop known as D flip flop.

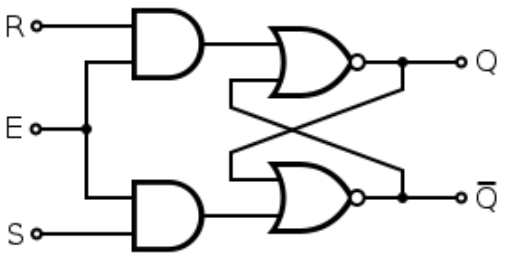
In this flip flop, the output follows the input after some time delay when clock input is high.

WORKING:

- When clock is low, the flip flop remains in its previous output state i.e., $Q_{n+1} = Q_n$.
- When clock is high and D=0 , the flip flop is in RESET condition i.e., $Q_{n+1} = 0$.
- When clock is high and D=1, the flip flop is in SET condition i.e., $Q_{n+1} = 1$.

TRUTH TABLE:

Clk	**D**	Q_{n+1}	ACTION
↓	X	Q_n	NC



↑	0	0	Reset
↑	1	1	Set

J-K Flip flop:

It is a modified form of R-S Flip flop. It has two control inputs known as J and K. J means Set and K means Rest. The j-k flip flop eliminates the forbidden condition in R-S flip flop i.e., when clock is high J=K=1 then output $Q_{n+1} = \bar{Q}_n$ i.e., present output state is equal to compliment of previous output state. This condition is known as Toggle condition state.

Positive edge triggered J-K flip flop:

Case 1: When clock is low, J=K=1 then flip flop is said to be in previous output state (or) no change.

Case 2: When clock is high, J=K=0 , then flip flop remains in its previous output state i.e., $Q_{n+1} = Q_n$

Case 3: When clock is high , J=0 K=1 , then the flip flop is in Reset condition i.e., $Q_{n+1} = Q_n = 0$

Case 4: When clock is high , J=1 K=0 then the flip flop is in Set condition. i.e., $Q_{n+1} = Q_n = 1.$

Case 5: When clock is high, J=K=1 then the flip flop is in toggle condition. i.e., $Q_{n+1} = \bar{Q}_n$.

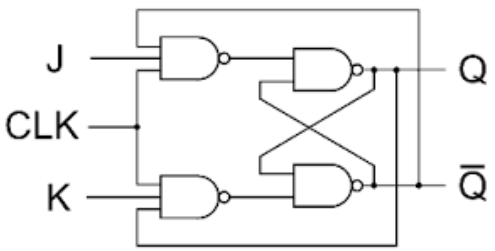
a)In previous output $Q_n = 0$, present output $Q_{n+1} = \bar{Q}_n = 1$ then flip flop is in Set condition.

b) In previous output $Q_n = 1$ present output $Q_{n+1} = \bar{Q}_n = 0$ then the flip flop is in Rest condition.

Truth table:

Clk	J	K	OUTPUT	ACTION
↓	X	X	Q_n	NC
↑	0	0	Q_n	NC
↑	0	1	1	Reset
↑	1	0	0	Set
↑	1	1	$\bar{Q}_n$	Toggle

Disadvantages:



- When duration of clock pulse (t_p) is greater than propagation delay time (Δ_t) then output of J-K flip flop changes from 0 to 1 and 1 to 0.
- When J=K=1 ,(t_p<Δ_t) then the output races between 0 to 1. This condition is known as Race around condition.
- The race around condition is eliminated when t_p<Δ_t<t where

t_p is duration of clock pulse, Δ_t is propagation delay time of the gates, t is time period of the clock pulse. This disadvantage is eliminated by using a modified J-K flip flop known as Master Slave J-K flip flop.

T-Flip flop:

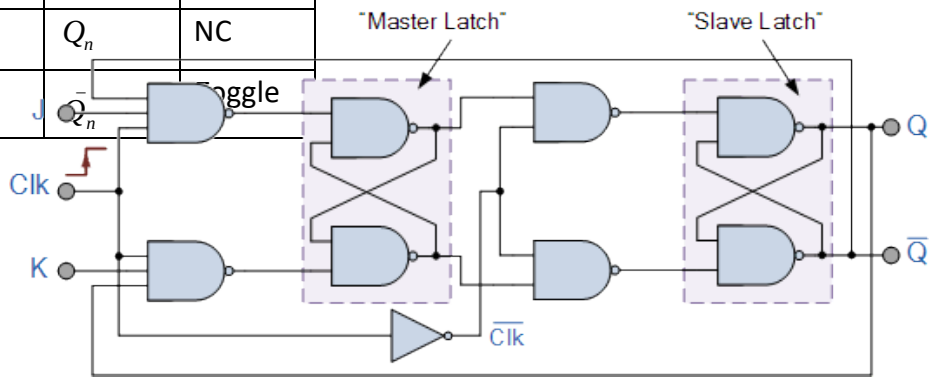
T means toggle flip flop. In j-k flip flop both J and K are combined through a common controlled input known as T.

Working:

- When clock is low, T=X , the output Q_{n+1} remains in its previous output state i.e., $Q_{n+1} = Q_n$.
- When clock is high, T=0 then the output remains in its previous state i.e., $Q_{n+1} = Q_n$.
- When clock is high, T=1 then the output Toggle i.e., $Q_{n+1} = \bar{Q}_n$.

Truth table:

clk	**T**	Q_{n+1}	ACTION
↓	X	Q_n	NC
↑	0	Q_n	NC
↑	1	$\bar{Q}_n$	Toggle

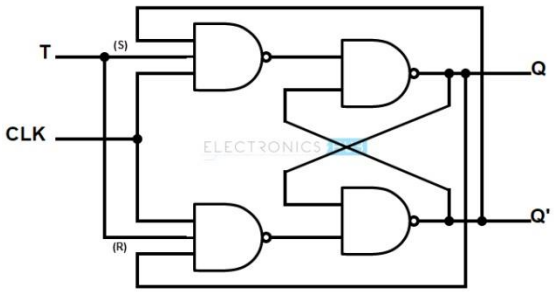


Master_Slave flip flop:

It is a modified form of J-K flip flop. It is widely used as practical arrangement to eliminate race around condition.

The above logic circuit consists of a J-K flip flop commenced in cascade configuration. The output of second flip flop is feedback to the input of first flip flop , In this first flip flop is known as Master J-K flip flop and second flip flop is known as Slave J-K flip flop. The master flip flop is positive edge trigger and output changes during leading edge of the clock pulse, The same clock input is applied to slave flip flop to a NOT gate i.e., negative edge trigger. The slave flip flop changes during output logic state during lagging edge of clock pulse.

Slave flip flop copies the action of master i.e., in the clock pulse duration t_p. The output remains constant hence race around condition is eliminated. The pre-set and clearance are known as Direct (or) Asynchronous inputs when p_r=1, c_r=1 , clock pulse is high the flip flop is enable.



Working conditions:

Case 1:

- When J=K=0 , clk1, then the master flip flop remains in its previous output state i.e., $(Q_m)_{n+1} = Q_m$.

- The output of master flip flop drive the input of slave flip flop i.e., $(Q_m)_{n+1} = s = Q_n$ and $(\bar{Q_m})_{n+1} = R = \bar{Q_n}$.
- During negative edge of the clock pulse the slave flip flop copes the action of master i.e., $(Q_s)_{n+1} = (Q_m)_n$ i.e., no change.

Case 2:

- When J=0, K=1,clk=1 then the output of master flip flop is $(Q_m)_{n+1} = 0$, $(\bar{Q_m})_{n+1} = 0$.
- Now the inputs of slave flip flop is $(Q_m)_{n+1} = S = 0$ and $(\bar{Q_m})_{n+1} = R = 1$.
- During negative edge of the clock pulse, the output $Q_s = (Q_m)_{n+1} = 0$ i.e., Reset.

Case 3:

- When J=1, K=0 , clk is high. During leading edge of the clock pulse, the master flip flop output is $(Q_m)_{n+1} = 1$, $(\bar{Q_m})_{n+1} = 0$.
- Now, the inputs of slave flip flop is $(\bar{Q_m})_{n+1} = R = 0$,and $(Q_m)_{n+1} = S = 1$.
- During tale edge of the clock pulse, the output $(Q_s)_{m+1} = (Q_m)_{n+1} = 1$ i.e., Set condition.

Case 4:

- When J=K=1 , during leading edge of the clock pulse, the master flip flop output is Toggle's i.e., $(Q_m)_{n+1} = (\bar{Q_m})_n$.
- During tale edge of the clock pulse, the slave output is $(Q_s)_{n+1} = (\bar{Q_m})_n$.

Truth table:

Clk	J	K	OUTPUT		ACTION
			$(Q_m)_n$	$(Q_s)_n$	
↓	X	X	Q_m	Q_s	NC
↑	0	0	Q_m	Q_s	NC
↑	0	1	0	0	Reset
↑	1	0	1	1	Set
↑	1	1	1	1	Toggle

REGISTERS:

A group of flip flop's is known as Register. To store group of bits (or) binary number group of flip flop's are required.

Eg: Consider a 4 bit number it is known as Nibble. to store 4 bits, 4 flip flop's are required and this register is known as a 4 bit register.

-8 bit register consists of 8 flip flop's to store 8 bits.

Shift Register:

A group of flip flop's connected in such a way that the binary information (or) data may be shift into out of flip flop.

It is shifted in two ways.

i) In this, the binary number is shifted one bit at a time i.e., in serial fashion starting from either LSB,MSB.
ii) In this, the binary data shifted simultaneously (or) all bits at a time i.e., parallel fashion.

It is classified into 4 types

i) Serial in SI – Serial out SO:
In this type of register binary data is given in input in serial fashion i.e., one bit after another bit starting from either LSB, MSB. The binary output is also obtained in serial fashion.
ii) Parallel in PI- Parallel out PO:

In this type of register binary data is given in input in parallel fashion i.e., all bits at a time. The binary output is also obtained in parallel fashion.

iii) Serial in SI– Parallel out PO:
In this type of register binary data is given in input in serial fashion i.e., one bit after another bit starting from either LSB, RSB. The binary output is obtained in parallel fashion i.e., all bits at a time.
iv) Parallel in PO – Serial out SO:
In this type of register binary data is given in input in parallel fashion i.e., all bits at a time. The binary output is obtained in serial fashion i.e., one bit after another bit starting from either LSB, RSB.

Shift Left Register (SLR):

In this the binary data is shifted from left to right. Hence the name known as Shift left Register. These registers are constructed with D-FLIP FLOP'S (or) J-K FLIP FLOP'S. The sequential logic circuit serial In- serial Out shift left 4 bit registers use in D-FLIP FLOPS.

---------CIRCUIT---------

The above logic circuit consists of 4 positive edge triggered D flip flop connected in cascade configuration. The output of first flip flop Q_0 drives the input of second flip flop D_1. Similarly Q_1 drives D_2 and Q_2 drives D_3.

In this the binary data is applied to the input in serial fashion and output is obtained in serial fashion.

When D_{in}=0, all flip flop's are in reset condition.

When D_{in}=1, the first leading edge of the clock pulse arrives the clock input of first flip flop goes into Set condition. (or) Q_0=1.

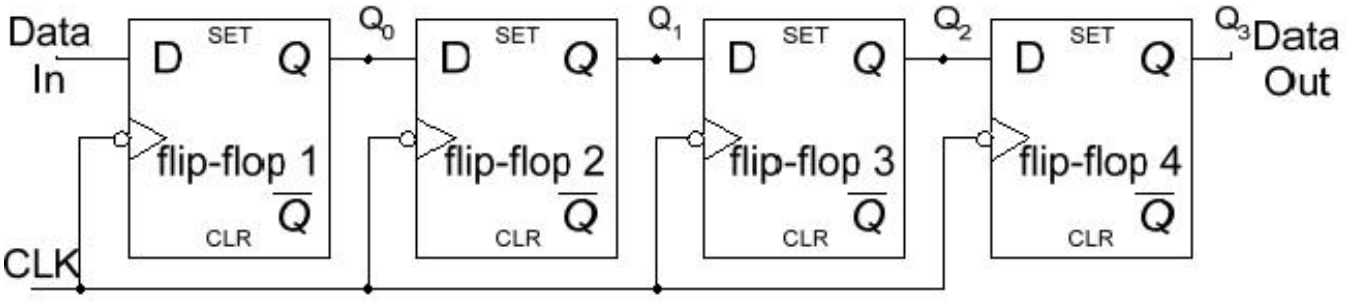
Now , the stored binary word is $Q_3Q_2Q_1Q_0$ =0001.

For second Leading edge of the clock pulse, the second flip flop goes into Set condition and the stored binary word is $Q_3Q_2Q_1Q_0$ = 0011

The above process repeats up to successive leading edge of the clock pulse as long as D_{in}=1.

The register contains for the successive clock pulse as shown below.

Q_3 Q_2 Q_1 Q_0



First +ve	0	0	0	1
Second +ve	0	0	1	1
Third +ve	0	1	1	1
Fourth +ve	1	1	1	1

If D_{in}=0, the register contains for the successive leading edge of the clock pulse are shown below.

Fifth +ve	1	1	1	0
Sixth +ve	1	1	0	0
Seventh +ve	1	0	0	0
Eighth +ve	0	0	0	0

Shift Right Register (SRR):

In this the binary data is shifted from right to left. Hence the name known as Shift right Register. These registers are constructed with D-FLIP FLOP'S (or) J-K FLIP FLOP'S. The sequential logic circuit serial In- serial Out shift left 4 bit registers use in D-FLIP FLOPS.

The above logic circuit consists of 4 positive edge triggered D flip flop connected in cascade configuration. The output of first flip flop Q_0 drives the input of second flip flop D_1. Similarly Q_1 drives D_2 and Q_2 drives D_3.

In this the binary data is applied to the input in serial fashion and output is obtained in serial fashion.

When D_{in}=0, all flip flop's are in reset condition.

When D_{in}=1, the leading edge of the clock pulse arrives the clock input of first flip flop goes into Set condition. (or) Q_0=1.Now , the stored binary word is $Q_3Q_2Q_1Q_0$ =0001.For second edge of the clock pulse, the second flip flop goes into Set condition and the stored binary word is $Q_3Q_2Q_1Q_0$ = 0011

The above process repeats up to successive leading edge of the clock pulse as long as D_{in}=1. The register contains for the successive clock pulse as shown below.

	Q_3	Q_2	Q_1	Q_0
First +ve	1	0	0	0
Second +ve	1	1	0	0
Third +ve	1	1	1	0
Fourth +ve	1	1	1	1

If D_{in}=0, the register contains for the successive leading edge of the clock pulse and five positive clock edge.

Fifth +ve	0	1	1	1
Sixth +ve	0	0	1	1
Seventh +ve	0	0	0	1
Eighth +ve	0	0	0	0

COUNTERS:

It is a register which counts the number of clock pulses arrived at its clock input. Counters are generally two types.

→Asynchronous

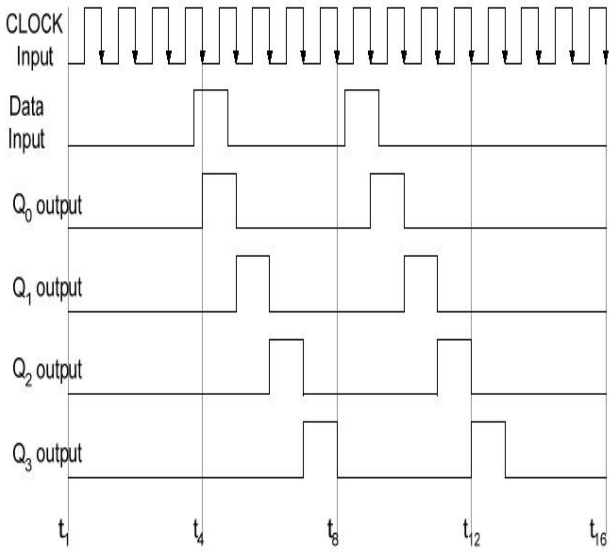
→Synchronous

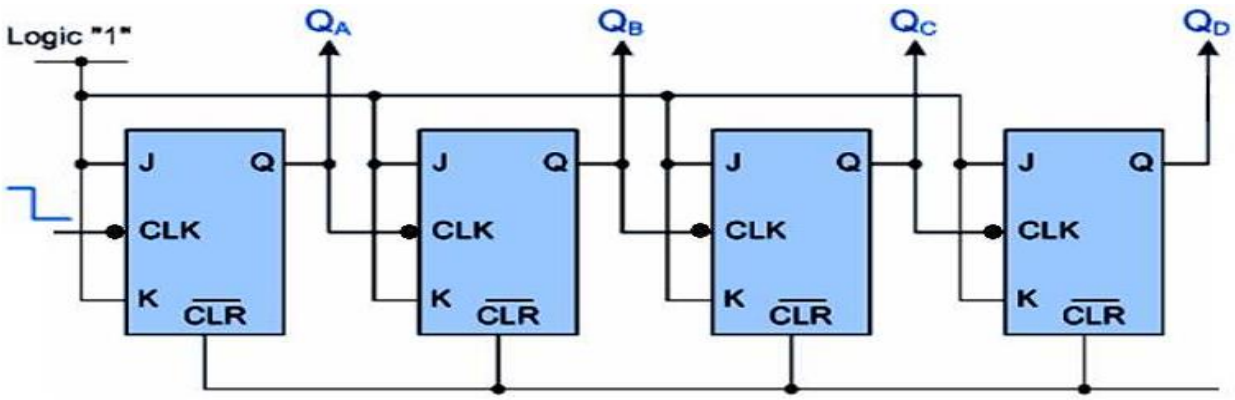
i) Asynchronous: In this counter, the change in output logic state is not controlled by the clock. But in this counters output of first flip flop will drive the input of next flip flop.
Eg: MOD 16(Ripple) , MOD 8 , MOD 10, UP-DOWN Counter.

ii) Synchronous: In this counters, the change in output logic state is controlled by external signal. It is called as clock. Hence these are known as high speed, clocked and timed.
Eg: 4-Bit synchronous, ring counter.

MOD – 16 Counter (or) Up counter:

Modules means no. of outputs counts. Here the counter counts from 0 to 15. Number of flip flops required is $\log_2^N = \log n$ where N= Number of output counts. Therefore, $n = \log_2^{2^4}$ is equal to 4 flip flops. Here, the counting takes place in ascending order. Hence it named as Up counter.





For all the flip flop inputs J and K are tied together and given a +5v supply (or) logic 1. When clearness=0, all flip flop's are in reset condition i.e $Q_3Q_2Q_1Q_0$ =0000. When clearness=1, the flip flop is enable and counter is ready to perform counting action.

Case i): When –ve edge of the clock pulse is arrived at its clock input i.e., $J_0=K_0=1$ AND Clk=1, the LSB flip flop Toggles i.e., $Q_0=1$. The stored binary number is $Q_3Q_2Q_1Q_0$ = 0001

Case ii): When second –ve edge of the clock pulse is arrived at its clock input. The LSB flip flop Toggle i.e., $Q_0=0$. At the same time, the previous output of LSB flip flop $Q_0=1$ drives the clock input of second flip flop $J_1=K_1=1$, clk=$Q_0=1$. Hence second flip flop Toggle i.e., $Q_1=1$. The stored binary number is $Q_3Q_2Q_1Q_0$ =0010.

Case iii): When third –ve edge of the clock pulse arrived at its clock input. The LSB flip flop Toggles i.e., $Q_0=1$ drives the clock input of second flip flop i.e., $J_1=K_1=1$, clk=$Q_0=1$ and it remains in its previous state. The stored binary number is $Q_3Q_2Q_1Q_0$ =0011.

Case vi): When fourth –ve edge of the clock pulse arrived at its clock input. The LSB flip flop Toggles i.e., $Q_0=0$. When in the previous input $Q_0=1$ drives the clock input of second flip flop $J_1=K_1=1$,clk=$Q_0=1$ i.e., second flip flop Toggles $Q_1=0$. At the same time the previous output $Q_1=1$ drives clock input of third flip flop i.e., $J_2=K_2=1$, clk=$Q_1=1$. Hence third flip flop toggles i.e., $Q_2=1$. The stored binary number is $Q_3Q_2Q_1Q_0$ =0101

,', In this way, the process repeats upto 15th –ve edge clock pulse and the stored binary data is $Q_3Q_2Q_1Q_0$ =1111.For the next –ve edge of the clock pulse, all flip flop's goes to Reset condition i.e., $Q_3Q_2Q_1Q_0$ =0000.

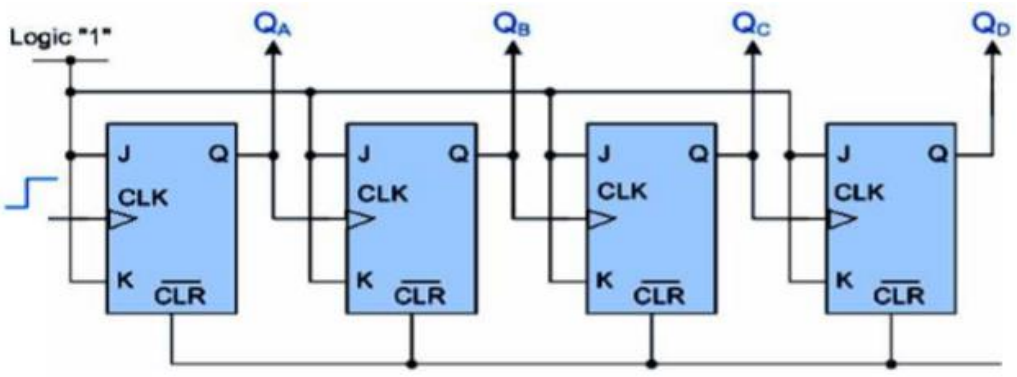
	Q_3	Q_2	Q_1	Q_0
0 →	0	0	0	0
1 →	0	0	0	1
2 →	0	0	1	0

Timing diagram:

3 → 0 0 1 1

4 →
1
0

0
0



0
0

5 →
1
1

0
0

6 →
1
0

0
1

7 → 0 1 1 1

8 → 1 0 0 0

9 → 1 0 0 1

10→ 1 0 1 0

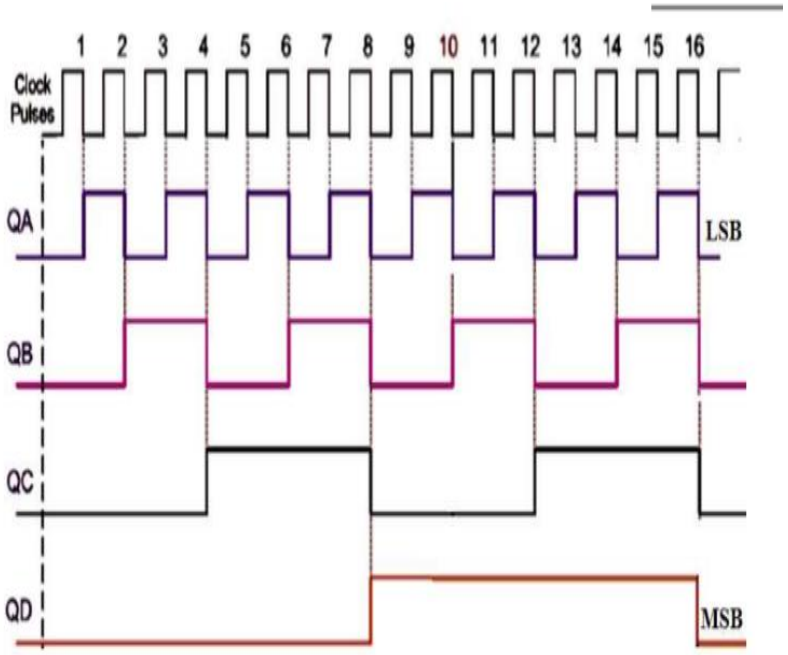
11→ 1 0 1 1

12→ 1 1 0 0

13→ 1 1 0 1

14→ 1 1 1 0

15→ 1 1 1 1



In the above counter, the carry propagates through the flip flops like ripples on water. Hence the name known as Ripple counter.

Down counter:

It is a sequential logic circuit which counts the number of clock pulses arrived at its clock input in descending order. It is known as down counter.

The above logic circuit consists of four edge triggered J-K Flip flops. The J-K inputs of all flip flop's are connected to +v (or) logic 1 state. This counter is known as Asynchronous flip flop i.e., output of first flip flop drives the clock input of next flip flop and so on. But here to obtain down counting compliment output $\bar{Q}_0$ drives the clock input of next flip flop and so on.

When pre-set=0, all flip flops are in Set condition i.e., $Q_3Q_2Q_1Q_0$=1111

When pre-set=1, the counter is enable ready to start counting action in descending order.

Case i): When –ve edge of first clock pulse arrived at its clock input, the LSB flip flop Toggle's i.e., $Q_0 = 0$, $\bar{Q}_0 = 1$. The stored binary number is 1110

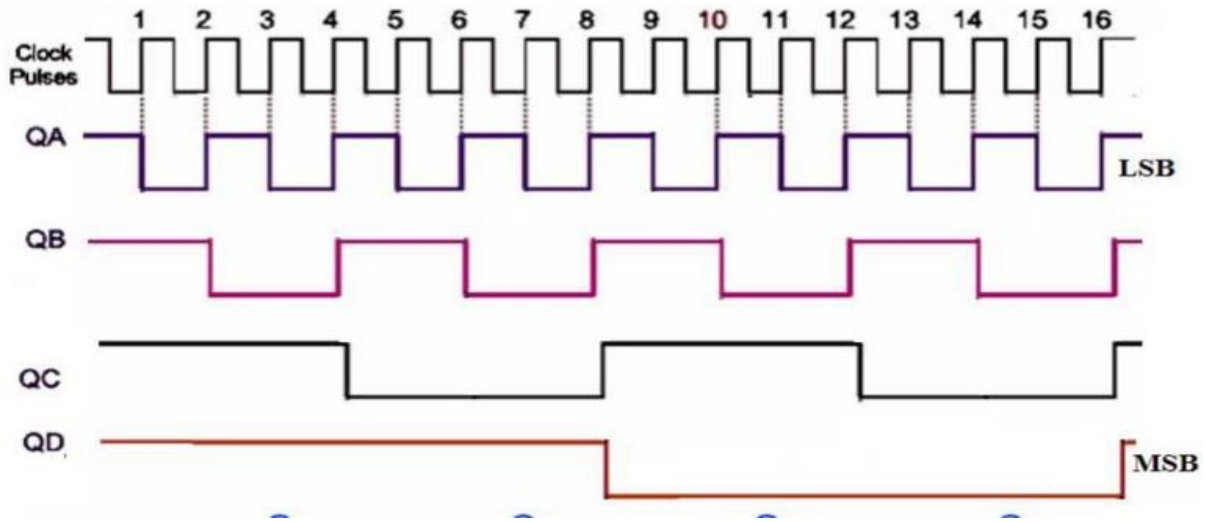
Case ii): When –ve edge of second clock pulse arrived at its clock input of LSB flip flop positive toggle takes place i.e., $Q_0 = 1, \bar{Q}_0 = 0$. At the same time previous output $\bar{Q}_0 = 1$ drives the clock input of second flip flop. Hence –ve toggle takes place $Q_1 = 0, \bar{Q}_1 = 1$. Now the stored binary number is 1101

Case iii): When –ve edge of third clock pulse arrived at its clock input the LSB flip flop toggles $Q_0 = 0, \bar{Q}_0 = 1$. Now at the same time the previous output compliment $\bar{Q}_0 = 0$ drives the second flip flop since no toggle takes place. Hence, the stored binary number is 1100.

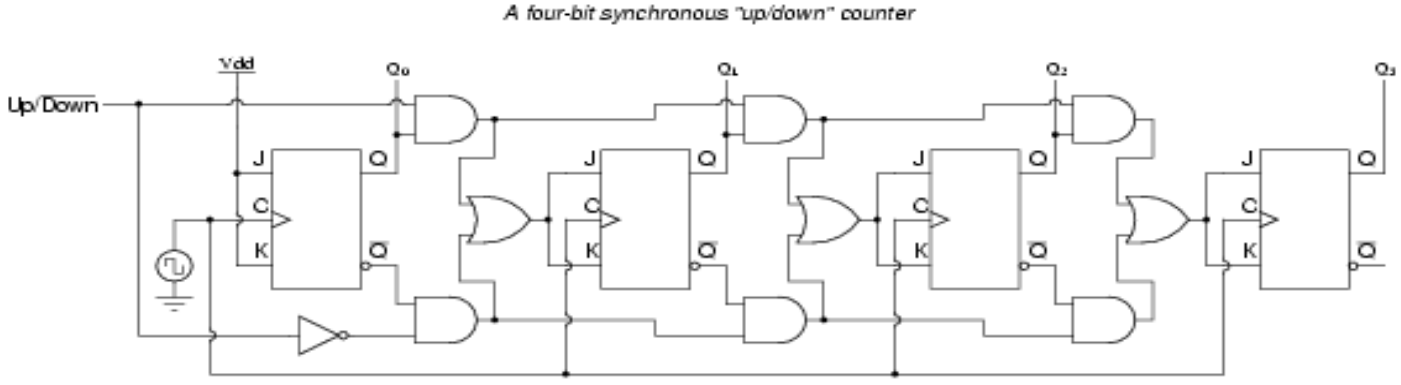
The above process repeats up to 15th –ve edge clock pulse and the stored binary number is 0000. For the next successive –ve edge of the clock pulse process repeats.

Shift pulse	counts	Q_3	Q_2	Q_1	Q_0
0	15	1	1	1	1
1	14	1	1	1	0
2	13	1	1	0	1
3	12	1	1	0	0
4	11	1	0	1	1
5	10	1	0	1	0
6	9	1	0	0	1
7	8	1	0	0	0
8	7	0	1	1	1
9	6	0	1	1	0
10	5	0	1	0	1

11	4	0	1	0	0
12	3	0	0	1	1
13	2	0	0	1	0
14	1	0	0	0	1
15	0	0	0	0	0



UpDown Counter:



- The counter which counts the number of clock pulses arrived at its up(X=1) and Down(X=0) clock input in ascending (or) descending order is known as Up-Down counter.
- The input J-K of all flip flops are connected to +5v (or) logic to obtain Toggle condition.
- If control signal X=1 then un complimented output Q_0 of LSB drives the clock input of next flip flop's and so on. Then the counter acts as Up counter.
- When control signal X=0, the complimented output $\bar{Q}_0$ of LSB drives the clock input of next flip flop's and so on. Then counter acts as down counter.
- UP-DOWN counting is performed by control signal X and by using AND, OR gates.

CASE I): Up counter CASE II): Down counter

DISADVANTAGES:

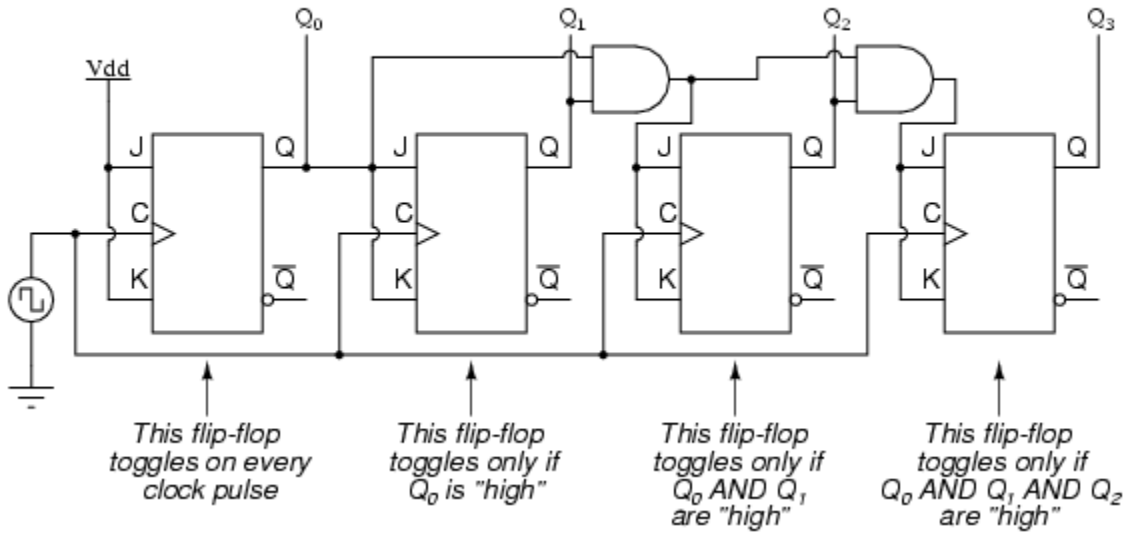
It's having high propagation delay and they are known as Low speed counters. To eliminate this disadvantage, the clock is applied to all flip flops. Therefore, propagation delay decreases and speed of counter increases known as synchronous counters.

4-bit synchronous counter:

In this clock pulses are given simultaneously through all the flip-flops

In this output of first flip-flop will drives the second similarly second drives third flip-flop.

A four-bit synchronous "up" counter



Ring counter:---

It is also known as Johnson counter.

It contains only one single high bit.

In this 1-bit flows in a circular path starting from either LSB (or) MSB.

It is also a shift left register in this one bit moves in rotate left operation

In this the 4 flip-flop output is connected to input of 1 flip-flop as feedback path.

If clr=0 all –ve edge triggerd d-flipflops are in reset condition.

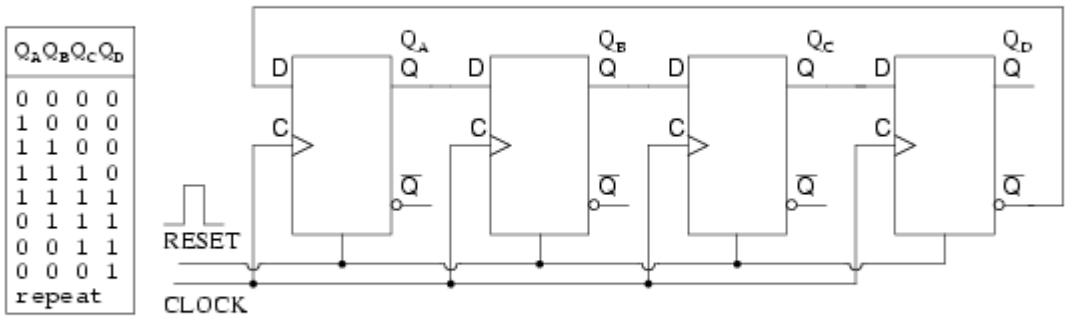
If clr=pre=1 a the output of first flip-flop goes in set condition.(0001)

If first –ve edge of clock pulse the second flip-flop goes in to set condition.(0010)

If second –ve edge of clock pulse the third flip-flop goes in to set condition.(0100)

If third –ve edge of clock pulse the fourth flip-flop goes in to set condition.(1000)

Therefore one bit moves in a circular path starting from LSB to MSB.



Johnson counter (note the $\overline{Q_D}$ to D_A feedback connection)

Applications:

1. Measurement of time

2.Measurement of distance

3.Measurement of frequency

4.Frequency converters

5.Digital computers.

RESISTOR TRANSISTOR LOGIC:--

Resistor–transistor logic (**RTL**) is a class of digital circuits built using resistors as the input network and bipolar junction transistors (BJTs) as switching devices. RTL is the earliest class of transistorized digital logic circuit used; other classes include diode–transistor logic (DTL) and transistor–transistor logic (TTL). RTL circuits were first constructed with discrete components, but in 1961 it became the first digital logic family to be produced as a monolithic integrated circuit. RTL integrated circuits were used in the Apollo Guidance Computer, whose design started in 1961.

RTL INVERTER:---

A bipolar transistor switch is the simplest RTL gate (inverter or NOT gate) implementing logical negation. It consists of a common-emitter stage with a base resistor connected between the base and the input voltage source. The role of the base resistor is to expand the negligible transistor input voltage range (about 0.7 V) to the logical "1" level (about 3.5 V) by converting the input voltage into current. It's resistance is chosen low enough to saturate the transistor and high enough to obtain high input resistance. The role of the collector resistor is to convert the collector current into voltage; its resistance is chosen high enough to saturate the transistor and low enough to obtain low output resistance (high fan-out).

One-transistor RTL NOR gate:--

The circuit consists of two or more base resistors (R_3 and R_4) instead of one, the inverter becomes a two-input RTL NOR gate .

The logical operation OR is performed by applying consecutively the two arithmetic operations addition and comparison (the input resistor network acts as a parallel *voltage summer* with equally weighted inputs and the following common-emitter transistor stage as a *voltage comparator* with a threshold about 0.7 V).

The equivalent resistance of all the resistors connected to logical "1" and the equivalent resistance of all the resistors connected to logical "0" form the two legs of a composed voltage divider driving the transistor.

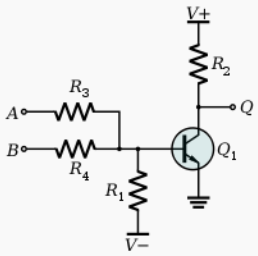
The base resistances and the number of the inputs are chosen (limited) so that only one logical "1" is sufficient to create base-emitter voltage exceeding the threshold and, as a result, saturating the transistor. If all the input voltages are low (logical "0"), the transistor is cut-off.

The pull-down resistor R_1 biases the transistor to the appropriate on-off threshold. The output is inverted since the collector-emitter voltage of transistor Q_1 is taken as output, and is high when the inputs are low.

Thus, the analog resistive network and the analog transistor stage perform the logic function NOR.

ADVANTAGES:

In this minimum number of transistors are required.



DTL LOGIC:--

Diode–transistor logic (**DTL**) is a class of digital circuits that is the direct ancestor of transistor–transistor logic. It is called so because the logic gating function (e.g., AND is performed by a diode network and the amplifying function is performed by a transistor (in contrast with RTL and TTL).

TWO INPUT DTL NAND GATE:

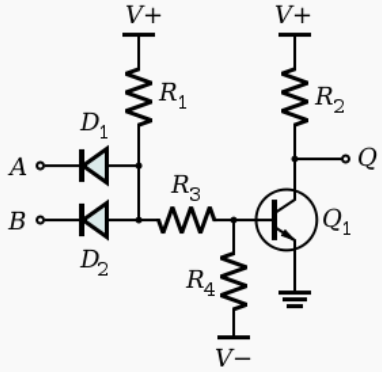
The DTL circuit consists of three stages: an input diode logic stage (D1, D2 and R1), an intermediate level shifting stage (R3, R4 and V−) and an output common-emitter amplifier stage (Q1 and R2).

case(i): If both inputs A and B are high (logic 1; near V+), then the diodes D1 and D2 are reverse biased. Resistors R1 and R3 will then supply enough current to turn on Q1 and also supply the current needed by R4. There will be a small positive voltage on the base of Q1 (V_{BE} about 0.3 V for germanium and 0.6 V for silicon transistors). The turned on transistor's collector current will then pull the output Q low.

Case(ii): If either or both inputs are low, then one of the input diodes conducts and pulls the voltage at the anodes to a value less than about 2 volts. R3 and R4 then act as a voltage divider that makes Q1's base voltage negative and consequently turns off Q1. Q1's collector current will be essentially zero, so, R2 will pull the output voltage Q high (logic 1; near V+).

Advantage:

It is to provide high propagation delay.



Programmable logic array:

A **programmable logic array** (**PLA**) is a kind of programmable logic device used to implement combinational logic circuits. The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes, which can then be conditionally complemented to produce an output. It has 2^N AND Gates for N input variables and for M outputs from PLA, there should be M OR Gates, each with programmable inputs from all of the AND gates. This layout allows for a large number of logic functions to be synthesized in the sum of products (and sometimes product of sums) canonical forms.

PLA's differ from Programmable Array Logic devices (PALs and GALs) in that both the AND and OR gate planes are programmable.

In 1970, Texas Instruments developed a mask-programmable IC based on the IBM read-only associative memory or ROAM. This device, the TMS2000, was programmed by altering the metal layer during the production of the IC. The TMS2000 had up to 17 inputs and 18 outputs with 8 JK flip flop for memory. TI coined the term *Programmable Logic Array* for this device.

Implementation:--

1. Preparation in SOP (sum of products) form.
2. Obtain the minimum SOP form to reduce the number of product terms to a minimum.
3. Decide the input connection of the AND matrix for generating the required product term.
4. Then decide the input connections of OR matrix to generate the sum terms.
5. Decide the connections of invert matrix.

6. Program the PLA.

PLA block diagram:

1ST BLOCK	2ND BLOCK	3RD BLOCK	4TH BLOCK	5TH BLOCK
INPUT BUFFER	AND MATRIX	OR MATRIX	INVERT/ NON INVERT MATRIX	FLIP FLOP OUTPUT BUFFER

Advantages over read-only memory

The desired outputs for each combination of inputs *could* be programmed into a read-only memory, with the inputs being loaded onto the address bus and the outputs being read out as data. However, that would require a separate memory location for *every* possible combination of inputs, including combinations that are never supposed to occur, and also duplicating data for "don't care" conditions (for example, logic like "if input A is 1, then, as far as output X is concerned, we don't care what input B is": in a ROM this would have to be written out twice, once for each possible value of B, and as more "don't care" inputs are added, the duplication grows exponentially); therefore, a programmable logic array can often implement a piece of logic using fewer transistors than the equivalent in read-only memory. This is particularly valuable when it is part of a processing chip where transistors are scarce (for example, the original 6502 chip contained a PLA to direct various operations of the processor[1]

Applications

One application of a PLA is to implement the control over a data path. It defines various states in an instruction set, and produces the next state (by conditional branching). [e.g. if the machine is in state 2, and will go to state 4 if the instruction contains an immediate field; then the PLA should define the actions of the control in state 2, will set the next state to be 4 if the instruction contains an immediate field, and will define the actions of the control in state 4]. Programmable logic arrays should correspond to a state diagram for the system.

Other commonly used programmable logic devices are PAL, CPLD and FPGA.

Note that the use of the word "programmable" does not indicate that all PLAs are field-programmable; in fact many are mask-programmed during manufacture in the same manner as a mask ROM. This is particularly true of PLAs that are embedded in more complex and numerous integrated circuits such as microprocessors. PLAs that can be programmed after manufacture are called FPGA (Field-programmable gate array), or less frequently FPLA (Field-programmable logic array).

Programmable array logic:--

is a family of programmable logic device semiconductors used to implement logic functions in digital circuits introduced by Monolithic Memories, Inc. (MMI) in March 1978.MMI obtained a registered trademark on the term PAL for use in "Programmable Semiconductor Logic Circuits". The trademark is currently held by Lattice Semiconductor.

PAL devices consisted of a small PROM (programmable read-only memory) core and additional output logic used to implement particular desired logic functions with few components.

Using specialized machines, PAL devices were "field-programmable". PALs were available in several variants:

- "One-time programmable" (OTP) devices could not be updated and reused after initial programming (MMI also offered a similar family called HAL, or "hard array logic", which were like PAL devices except that they were mask-programmed at the factory.).
- UV erasable versions (e.g.: PALCxxxxx e.g.: PALC22V10) had a quartz window over the chip die and could be erased for re-use with an ultraviolet light source just like an EPROM.

- Later versions (PALCExxx e.g.: PALCE22V10) were flash erasable devices.

In most applications, electrically-erasable GALs are now deployed as pin-compatible direct replacements for one-time programmable PALs.

History

Before PALs were introduced, designers of digital logic circuits would use small-scale integration (SSI) components, such as those in the 7400 series TTL (transistor-transistor logic) family; the 7400 family included a variety of logic building blocks, such as gates (NOT, NAND, NOR, AND, OR), multiplexers (MUXes) and demultiplexers (DEMUXes), flip flops (D-type, JK, etc.) and others. One PAL device would typically replace dozens of such "discrete" logic packages, so the SSI business declined as the PAL business took off. PALs were used advantageously in many products, such as minicomputers**.**

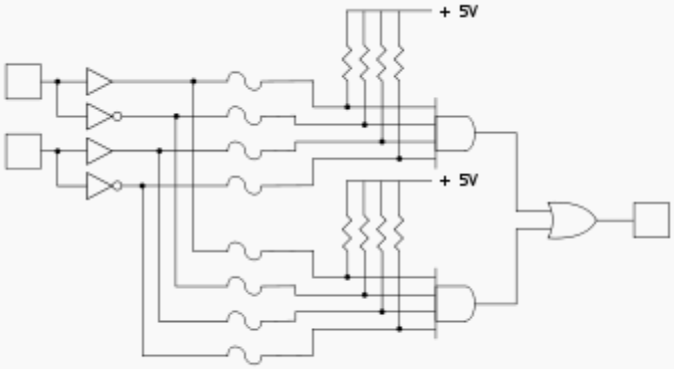
PALs were not the first commercial programmable logic devices; Signe tics had been selling its field programmable logic array (FPLA) since 1975. These devices were completely unfamiliar to most circuit designers and were perceived to be too difficult to use. The FPLA had a relatively slow maximum operating speed (due to having both programmable-AND and programmable-OR arrays), was expensive, and had a poor reputation for testability. Another factor limiting the acceptance of the FPLA was the large package, a 600-mil (0.6", or 15.24 mm) wide 28-pin dual in-line package (DIP).

Process technologies

Early PALs were 20-pin DIP components fabricated in silicon using bipolar transistor technology with one-time programmable (OTP) titanium-tungsten programming fuses. Later devices were manufactured by Cypress, Lattice Semiconductor and Advanced Micro Devices using CMOS technology.

The original 20 and 24-pin PALs were denoted by MMI as medium-scale integration (MSI) devices.

PAL architecture



Simplified programmable logic device

The programmable elements (shown as a fuse) connect both the true and complemented inputs to the AND gates. These AND gates, also known as *product terms*, are ORed together to form a *sum-of-products* logic array.

The PAL architecture consists of two main components: a logic plane and output logic macro cells.

Programmable logic plane

The programmable logic plane is a programmable read-only memory (PROM) array that allows the signals present on the devices pins (or the logical complements of those signals) to be routed to an output logic macro cell.PAL devices have arrays of transistor cells arranged in a "fixed-OR, programmable-AND" plane used to implement "sum-of-products" binary logic equations for each of the outputs in terms of the inputs and either synchronous or asynchronous feedback from the outputs.

Output logic:-

The 20-pin PALs had 10 inputs and 8 outputs. The outputs were active low and could be registered or combinational. Members of the PAL family were available with various output structures called "output logic macro cells" or OLMCs. Prior to the introduction of the "V" (for "variable") series, the types of OLMCs available in each L were fixed at the time of manufacture. (The PAL16L8 had 8 combinational outputs and the PAL16R8 had 8 registered outputs. The PAL16R6 had 6 registered and 2 combinational while the PAL16R4 had 4 of each.) Each output could have up to 8 product terms (effectively

AND gates), however the combinational outputs used one of the terms to control a bidirectional output buffer. There were other combinations that had fewer outputs with more product terms per output and were available with active high outputs. The 16X8 family or registered devices had an XOR gate before the register. There were also similar 24-pin versions of these PALs.

This fixed output structure often frustrated designers attempting to optimize the utility of PAL devices because output structures of different types were often required by their applications. (For example, one could not get 5 registered outputs with 3 active high combinational outputs.) So, in June 1983 AMD introduced the 22V10, a 24 pin device with 10 output logic macro cells. Each macro cell could be configured by the user to be combinational or registered, active high or active low. The number of product terms allocated to an output varied from 8 to 16. This one device could replace all of the 24 pin fixed function PAL devices. Members of the PAL "V" ("variable") series included the PAL16V8, PAL20V8 and PAL22V10.