Postulates / Identities of Boolean Algebra

Existence of Identity elements

0 - Additive identity

$x + 0 = x \quad x \in B$

$(x + 1 = 1)$

1 - Multiplicative Identity

$x \in B$

$x \cdot 1 = x$

$x \cdot 0 = 0$

Commutative law

$x + y = y + x$

$x \cdot y = y \cdot x$

Distributive law

$x(y + z) = x \cdot y + x \cdot z$

$x + (y \cdot z) = (x + y) \cdot (x + z)$

Inverse law

$x \cdot x' = 0$

$x + x' = 1$

Associative law

$x + (y + z) = (x + y) + z$

$x(yz) = (xy)z$

Idempotent law

$x + x = x$

$x \cdot x = x$

DeMorgan's law

I law : $(x+y)' = x' \cdot y'$

II law : $(x \cdot y)' = x' + y'$

~~Absorption law~~

I changing all ~~or~~ OR operations to AND

II changing all AND operations to OR

complementing of a function in true table is interchanging 1's and 0's

~~Absorption law~~

In algebraic form complements can be achieved by demorgans law.

Absorption law

$x + x \cdot y = x$

proof

$x(1+y) = x \cdot 1 = x$

$x \cdot (x+y) = x$

proof

$(x \cdot x + x \cdot y) = x + x \cdot y$

$= x$ //

Involution

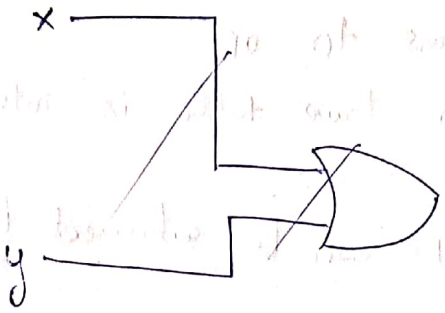
$(x')' = x$

The purpose of boolean algebra is to facilitate the analysis and design of digital circuits.

Boolean algebra is used when no. of variables are less ie if the no. of variables 1, 2 & 3. It is good to use boolean algebra.

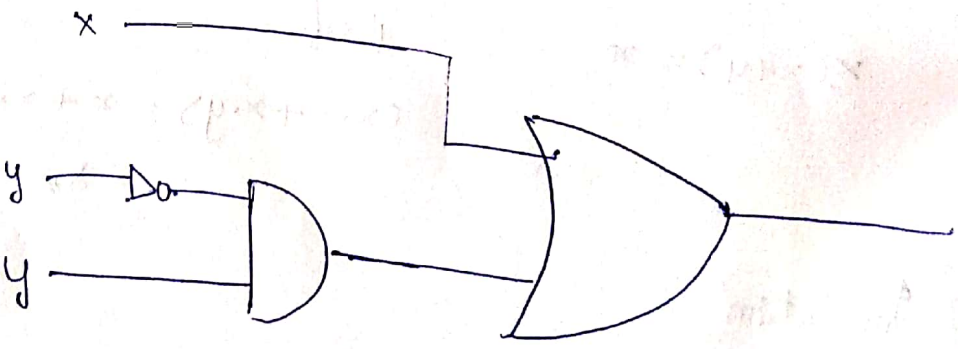
If it more no. of variables then we use K-map method. for minimization of logic expression

$$F = x + y'z$$



$F = x + y'z$

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



y'	$y'z$	$x+y'z$
1	0	0
1	1	1
0	0	0
0	0	0
0	0	1
1	1	1
1	0	1
0	0	1
0	0	

Proofs of Imp theorems

(a) $x+x=x$

(b) $x.x=x$

(a) LHS

$= x+x$

$= (x+x).1$

$= (x+x)(x+\bar{x})$

$= (x.x + x.x + x.\bar{x} + x.\bar{x})$

$= x$

Ex: $AB'+C'D+AB'+C'D$

Let $x = AB'+C'D$

exp can be written as

$x+x$

Then expression can be reduced to

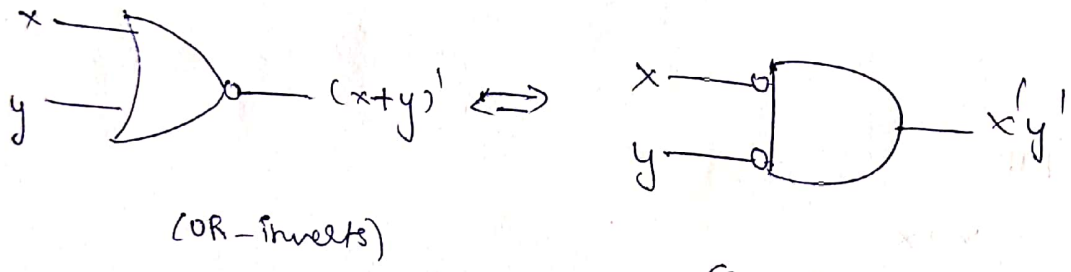
$$\underbrace{AB'+C'D}_{x} + \underbrace{AB'+C'D}_{x} = \underbrace{AB'+C'D}_{x}$$

Two graphic symbols for NOR gate & NAND gate

Demorgan's theorem is very important in dealing with NOR & NAND gate

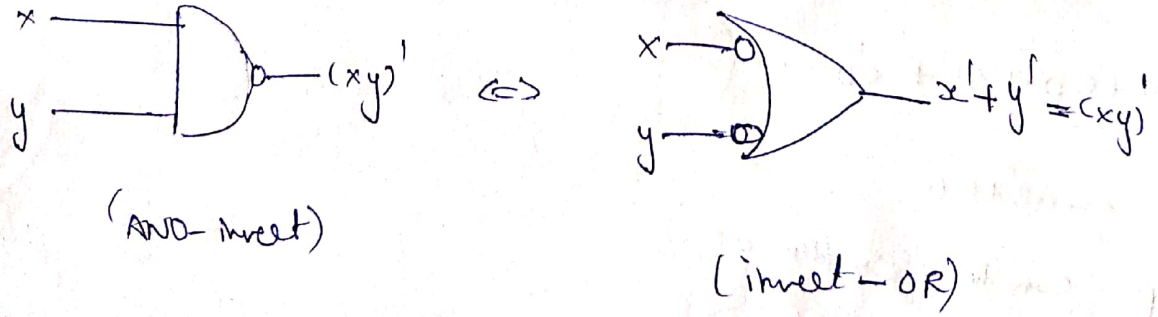
NOR gate

$(x+y)' \Longleftrightarrow x'y'$ hence two symbols

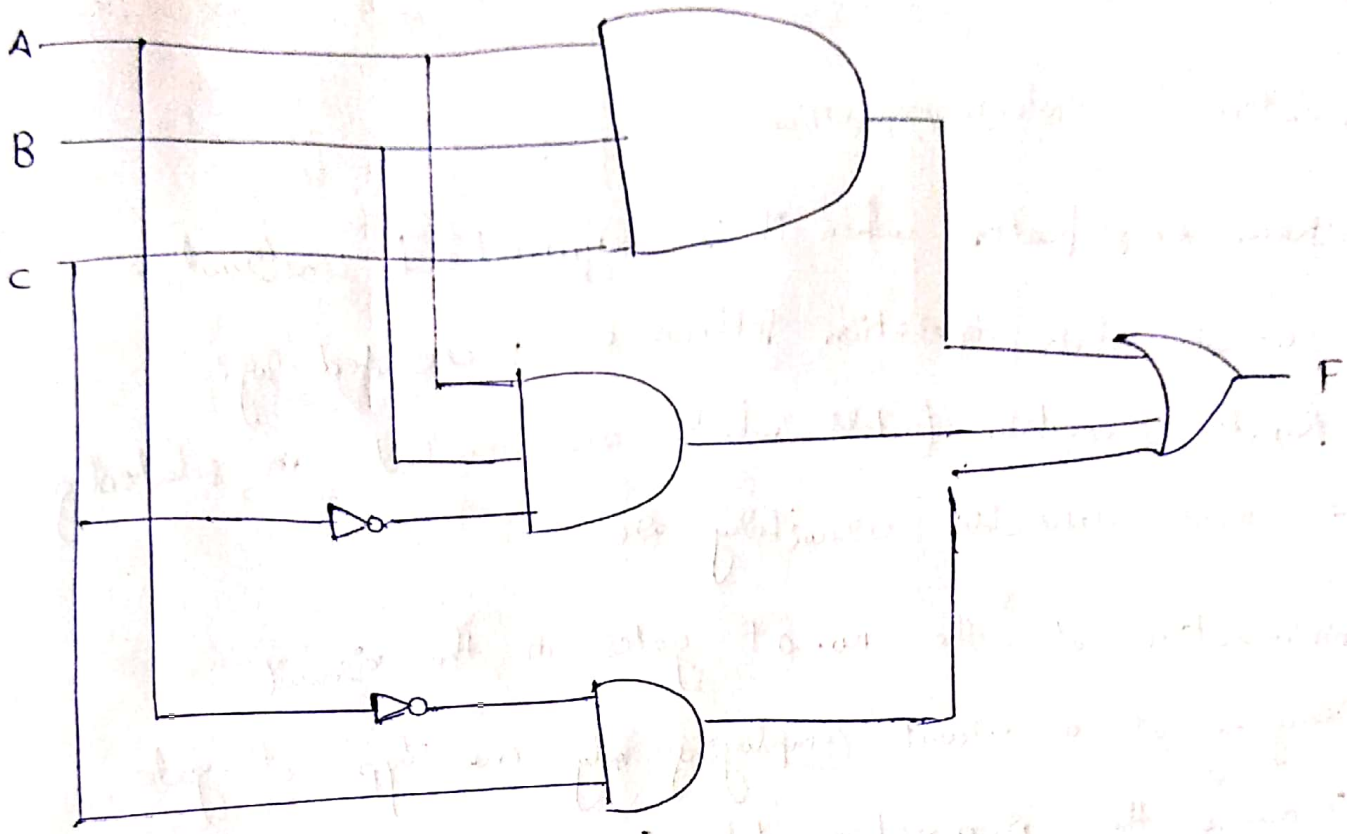


NAND gate

$(x.y)' \Longleftrightarrow x'+y'$



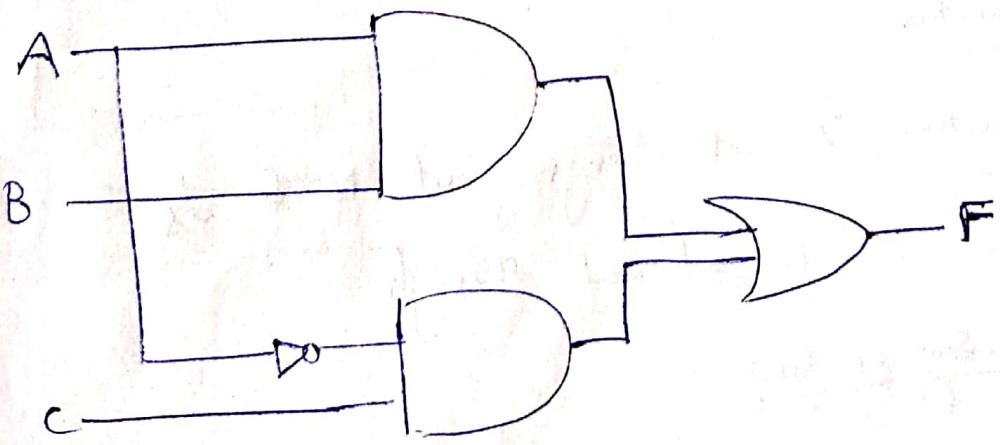
Simplication of Digital Circuit using Boolean Algebra



$F = ABC + AB C' + A' C$

$F = AB(C + C') + A' C$

$F = AB + A' C \quad (\because x + x' = 1)$



Complement of a Function

Complement of a function in truth table is

Simplification of Boolean expression

(1) Algebraic simplification when it is required to construct a logic circuit whose operation follows a prescribed logic function certain factors which are involved in selecting the most desirable circuitry are

(i) minimization of the no. of gates in the circuit
(ii) Designing of a circuit employing only one type of gate
(iii) Minimizing the propogation delay (time taken for input to output of a circuit)

Literal

A variable which is complemented or uncomplemented then it is called as literal.

Complement of a function

Complement of a function is interchanging of 1's & 0's

In ~~Ag~~ complement can be achieved through

DeMorgan's law

$(x_1 + x_2 + x_3 + \cdots + x_n)' = x_1' x_2' x_3' \cdots x_n'$

$(x_1 x_2 x_3 \cdots x_n)' = x_1' + x_2' + x_3' + \cdots + x_n'$

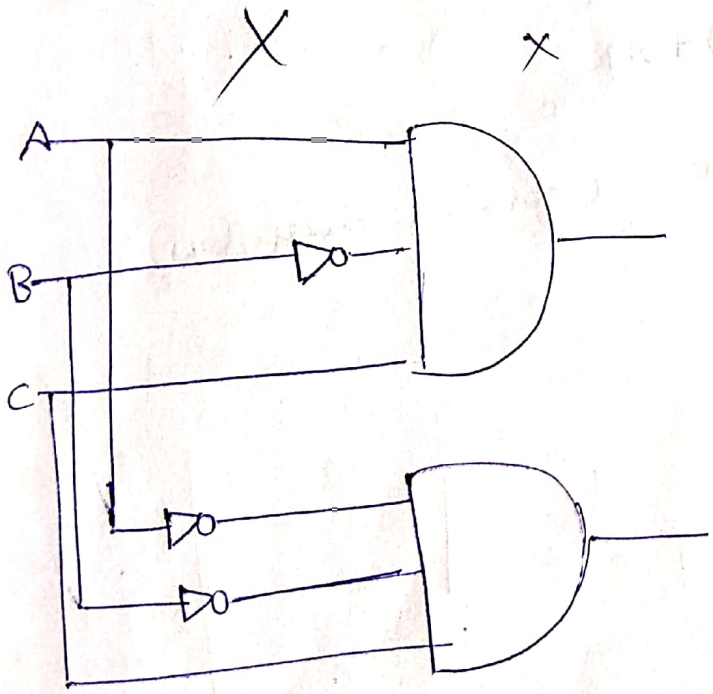
i.e. changing all 'OR' operations to 'AND' 'AND' operations to 'OR'

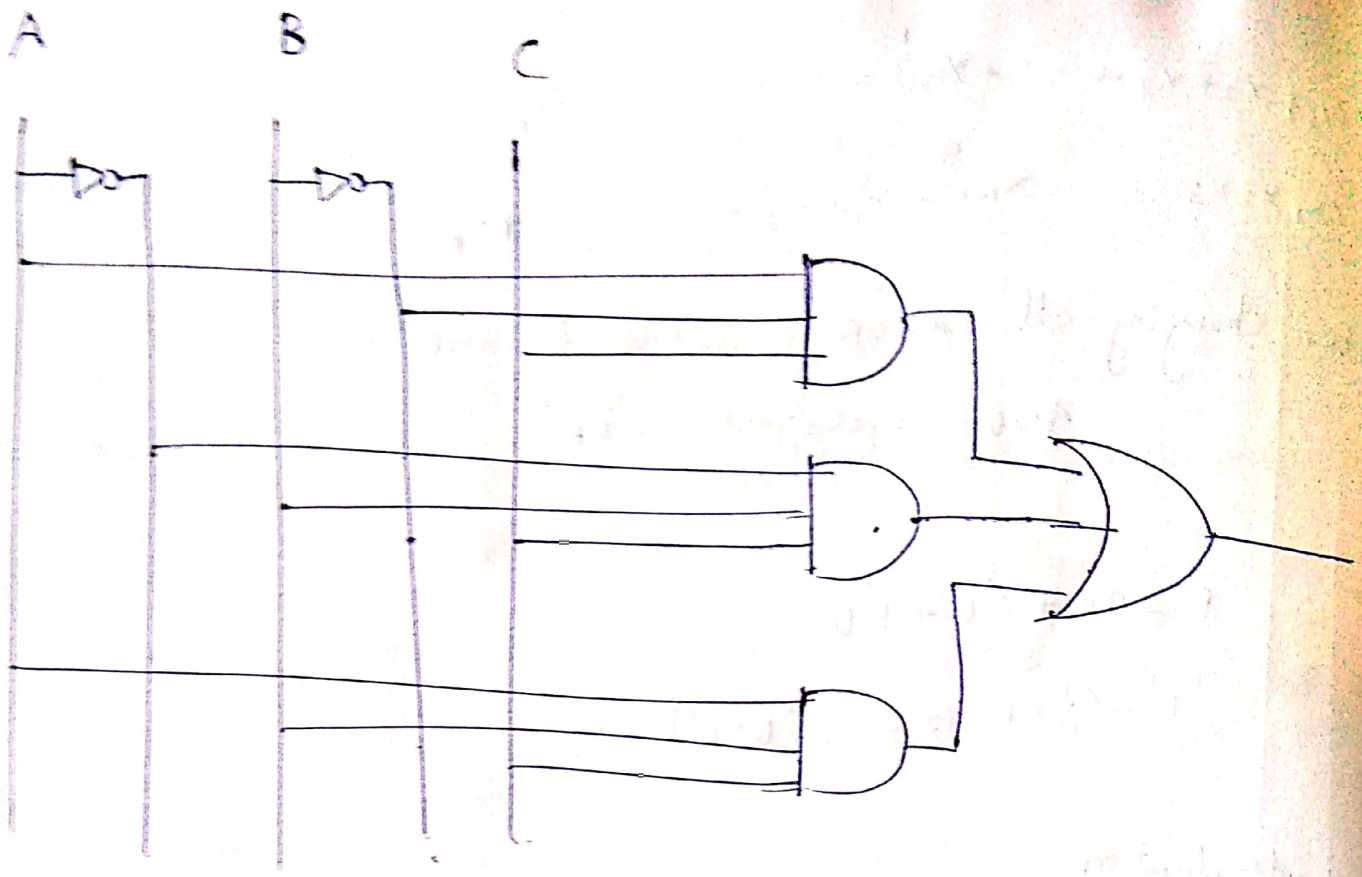
Ex: $F = AB + C'D' + B'D$

$F' = (A' + B')(C + D)(B + D')$

Simplification

$F = AB'C + A'B'C + ABC \rightarrow (1)$





6 gates

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

$= AB' + A'B' + AB = B'C(A + A') + ABC$

$= B'C(1) + ABC$ ($x + x' = 1$. inverse law)

$F_2 = B'C + ABC$

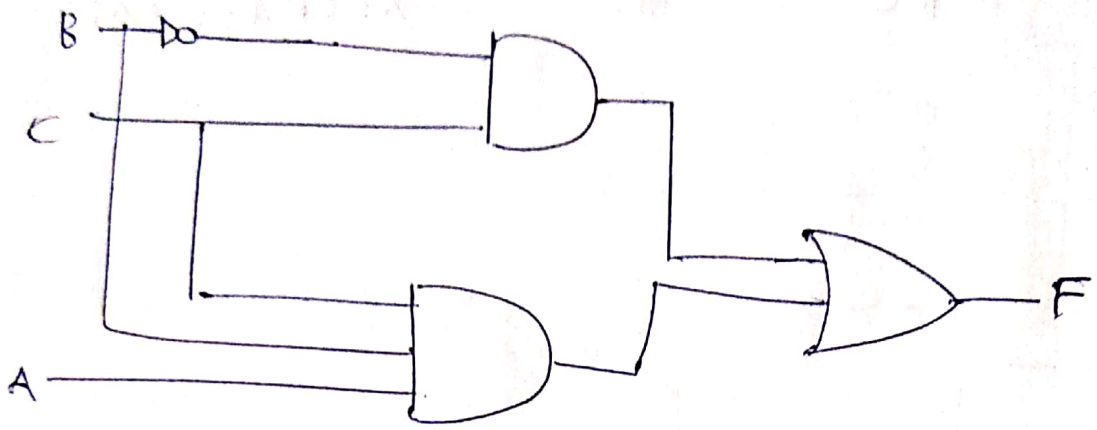
Truth table comparison

A	B	C	$AB'C$	$A'B'C$	ABC	$F_1 = AB'C + A'B'C + ABC$
0	0	0	0			
0	0	1	0			
0	1	0	0			
0	1	1	0			
1	0	0	0			
1	0	1	1			
1	1	0	0			
1	1	1	0			

Truth table.

A	B	C	$B'C$	ABC	$F_2 = B'C + ABC$
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	0	1	1

$F_2 = B'C + ABC$



(2) Simplify the Boolean expression

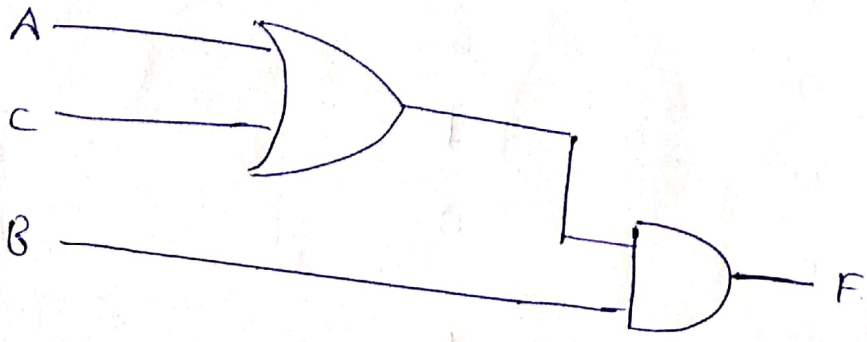
$= AB + BBC + BCC$ (Distributive law)

$= AB + BC + BC$ (Identity law $A \cdot A = A$)

$= AB + BC$

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

$F = B(A+C)$.



Truth table

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

Map simplification

K map - is used to simplify the Boolean expression in pictorial form without using Boolean laws & theorems.

It was developed by karnaugh in the year 1953

K-map is pictorial representation of truth table

It contains squares

Each square represents a min term

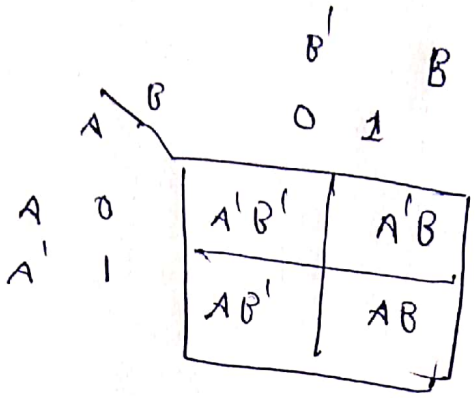
n- inputs $\Rightarrow 2^n$ squares

Two variable map

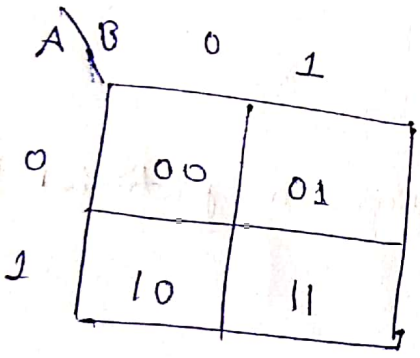
$2^n = 2^2 = 4$ min terms

m_0 m_1

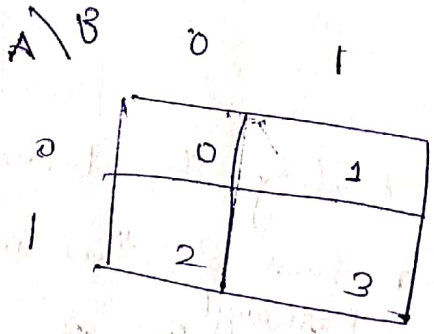
m_2 m_3



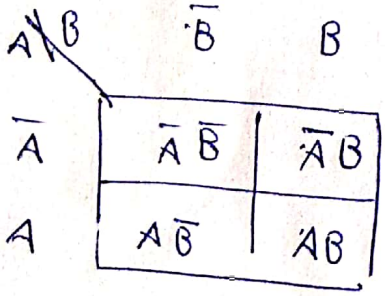
(i) Binary Representation



(ii) Decimal Representation



(iii) Variable Representation



Rules in creating groups

(i) No 0's are allowed

(ii) groups can vertical & horizontal but cannot be diagonal

(iii) overlapping is allowed.

(iv) Groups should be as large as possible

(v) Groups must contain 2^n cells.

(vi) Rule for adjacency

(i) Adjacency cells have only one variable changing

(ii) The two variables in the adjacent cell cannot change

(iii) Grouping K map table.

pairing reduces the no. of variables

Rules:

(a) Pair must contain one 1, 2, 4, 8 & 2^n cells

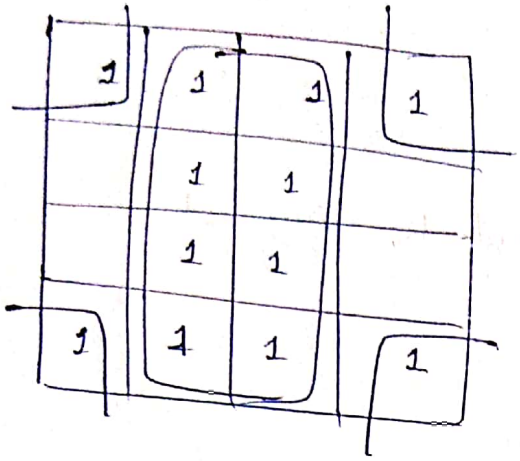
(b) Each pair should be as large as possible

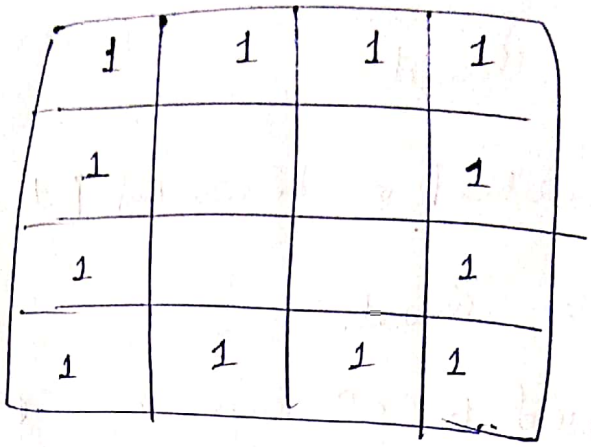
(c) Four cells are said to be adjacent of one another

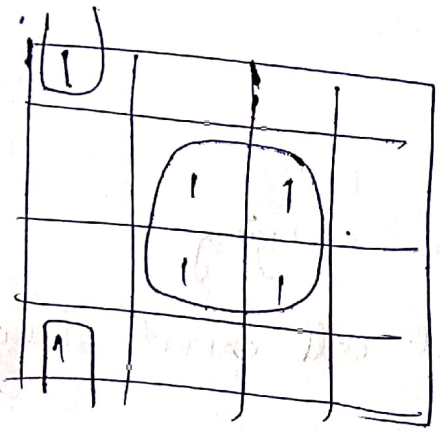
(d) Four consecutive 1's either in any row & in any column are said to be adjacent of one another.

(e) Four 1's forming of square anywhere in the table

(f). Rear may be horizontal & vertical but not diagonal.







Three variable K map

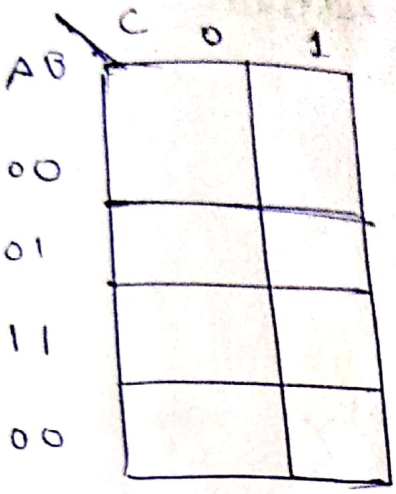
K map is a graphical method in which grey code representation is used to minimize the logical exp

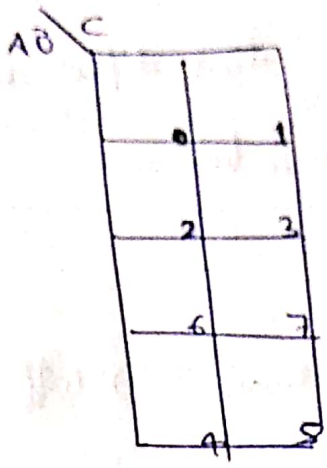
In grey code only 1 bit changes at a time

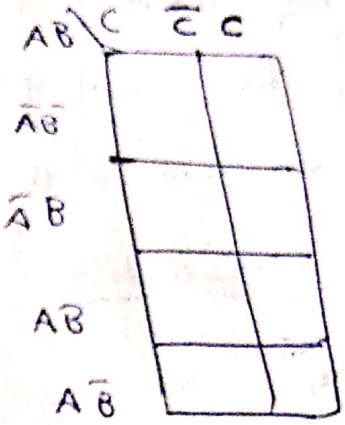
Two cells pairs and four cells pairing is called quad

eight cells pairing is called

vertical Representation







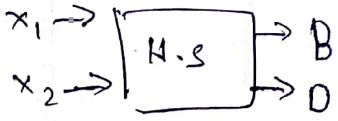
Half subtractor:

A half subtractor is a combinational circuit that subtracts two bits and generates their difference. It has two other outputs i.e borrow

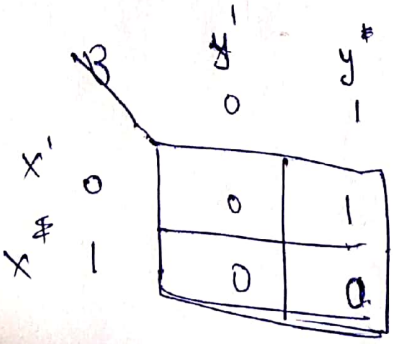
Truth table

x_1	x_2	B(Borrow)	D(Difference)
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Block Diagram

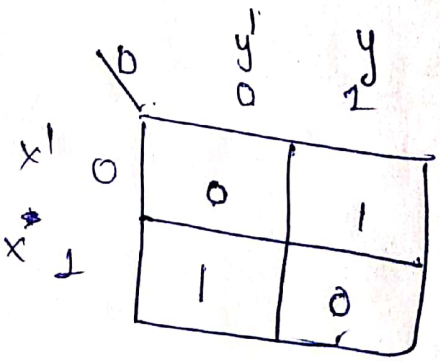


K-map for Borrow:



$B = x'y$

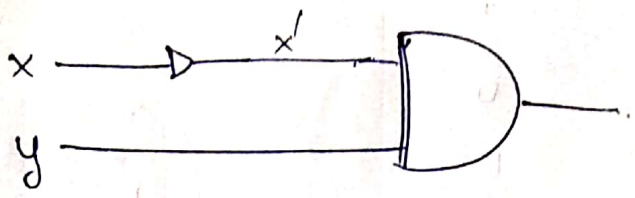
K map for Difference



$D(x,y) = x'y + xy'$

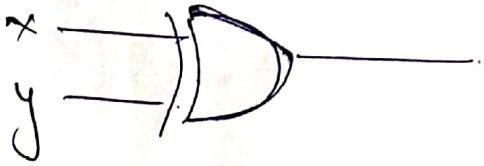
Logic gate for Bollow:

$$B(x,y) = x'y$$



Logic gate for Diffelence.

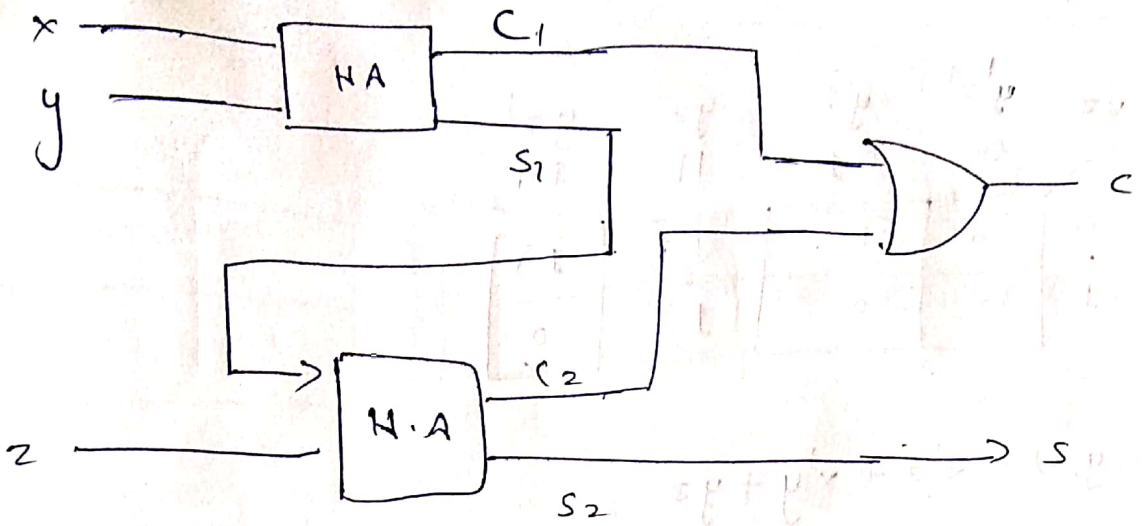
$$D(x,y) = x'y + xy'$$



Full Addel

A half addel only two inputs and no plovision to add a celly from lowel addel bits when multi bit addition is peifolmed. A full addel is a combinational cilcuit that adds thlee bits simultaneously (two significant bit takes a plevious celly)

Implementation of full adders using two half Adder



Full subtractor.

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	1
1	1	1	1	

K map for B(Borrow)

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

$$B(x,y,z) = x'z + x'y + yz$$

$$= \cancel{x'y + z(x'+y)} \quad = x'y + z(x'+y)$$

K map for D(Difference)

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

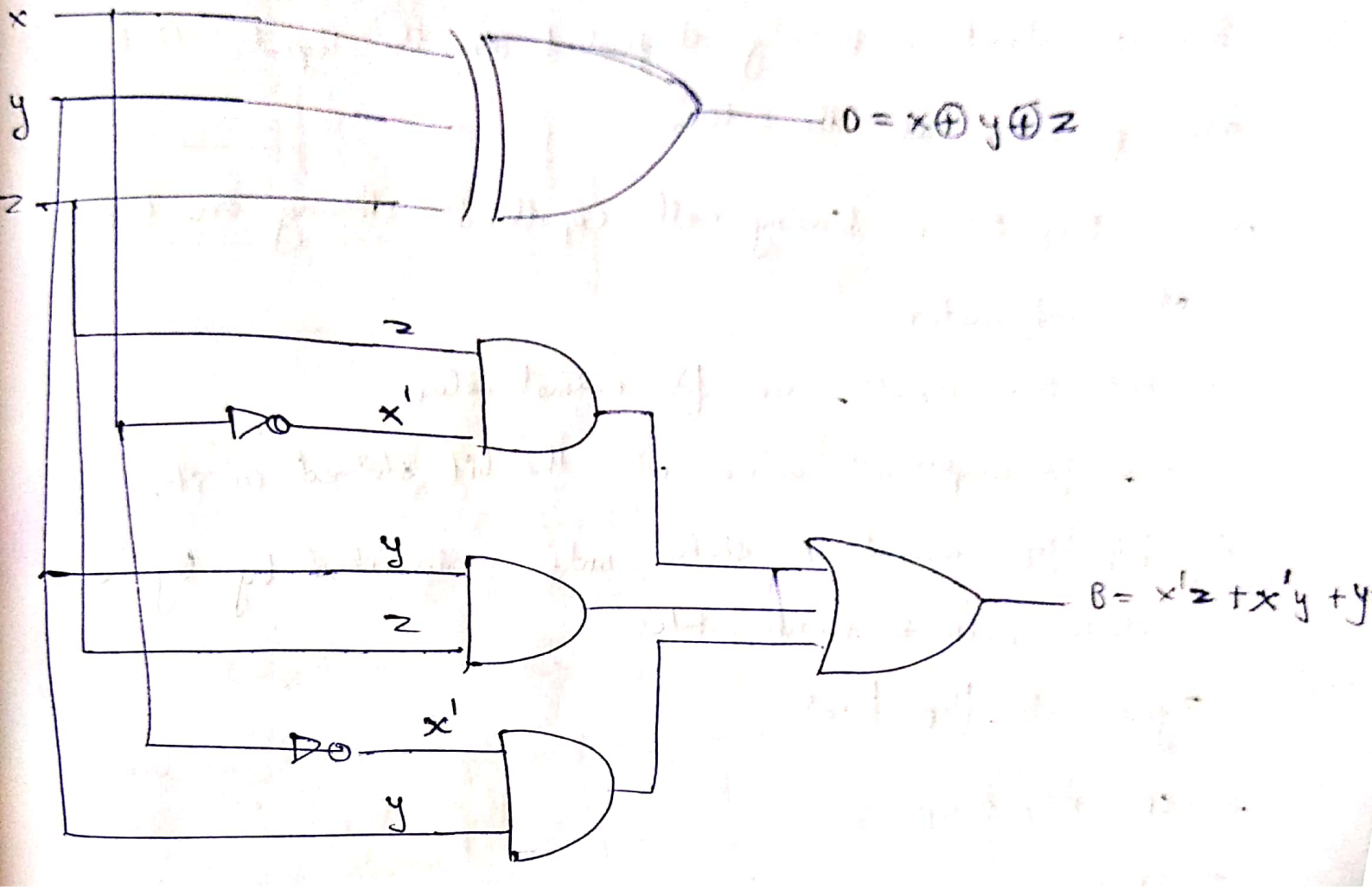
$$D(x,y,z) = xy'z' + x'y'z + xyz + x'yz'$$

$$= z'(xy' + x'y) + x'y'z + xyz$$

$$= z'(x \oplus y) + z(x \oplus y)'$$

$$= x \oplus y \oplus z$$

Logic circuit



Flip-flops

- Combinational circuit

In cc output is purely dependent on the inputs that are present at the time.

A Flip flop is a binary cell capable of storing one bit of information

It has two inputs - one for normal value one for complement value of the bit stored in it.

A flip flop maintains states until directed by a clock pulse to switch states

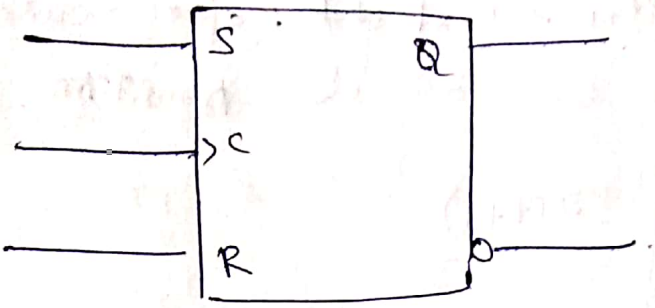
Types of flip-flops

- SR Flip-flop
- JK FF (Jack & Kill)
- MS - FF (Master slave)
- D - FF (Data or Delay)
- T - FF (Toggle

SR FlipFlop

S R stands for set, Reset

Graphic symbol & Block diagrams



It has 3 inputs S (for set)
R (Reset)
C (clock)

output is Q and

sometimes the complemented output indicated with a small circle

Arrow head shaped symbol infront of C is to designate dynamic input

Dynamic input

FF responds to a positive transition (0 to 1) of input clock signal C=0 - operation doesn't takes place
C=1

operations of SR FF

(1) NO signal at C ie if C=0 the output of the circuit cannot change irrespective of values of S and R

(2) When C (clock signal) changes from 0 to 1 then only output is effected according to S and R

Q_n / Q_t → previous state of present

Q_{t+1} → next state, after occurence of clock transition

S	R	Q(t+1)
0	0	Q(t) No change.
0	1	0 clear to 0
1	0	1 Reset to 1
1	1	Invalid

If both S=0, R=0 during clock change, the output doesnot change. Q(t)

If S=0, R=1 the clock pulse is changing from 0 to 1 then output Q is clear to 0

If S=1, R=0 and clock is changing from 0 to 1 then output Q is set to 1

If S=1, R=1, C = 0 to 1 then output is unpredictable and may go to either 0 or 1 depending on timing delays that occur within the circuit

JK Flip-flop.

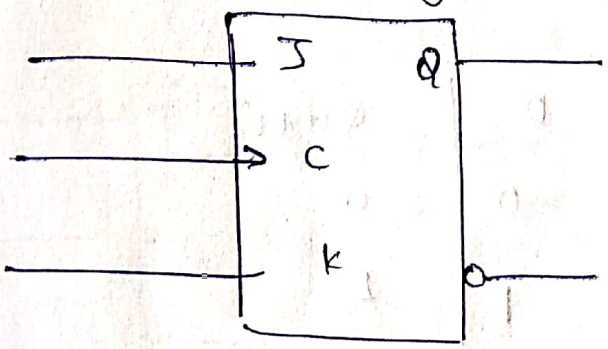
→ It is a refinement of SR-FF, in that the indeterminate condition of SR type is defined by JK type.

→ Inputs J & K behave like inputs S and R to set and clear the FF, respectively.

→ when inputs J & K = 1, a clock transition switches the outputs of the FF to their complement state.

$Q'(t)$

Block diagram



J	K	
0	0	Q(t) no change
0	1	0
1	0	1
1	1	Q'(t) complement

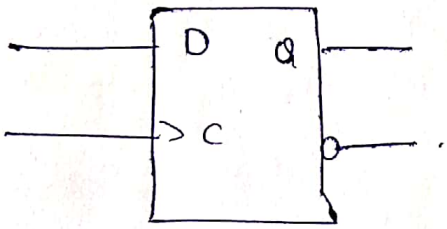
D - Flip-Flop

An SR FF can be converted to a D FF by inserting an inverter between S & R and assigning the symbol 'D' to single input 'D'.

→ If D=1 during C → 0 to 1
o/p of FF goes to state '1'.

→ If D=0 during C → 0 to 1
o/p of FF goes to state '0'

Block diagram.



D	Q(t+1)
0	0
1	1

Advantage of D-FF is having a single input D.

Disadvantage is its characteristic table doesn't have a no change condition

ie $Q(t+1) = Q(t)$

→ "No change" condition can be accomplished either by

* disabling clock signal
* Feeding o/p back to input

T- Flip Flop

T - Toggle

When a JK type FF inputs are connected to provide a signal input designated by 'T'.

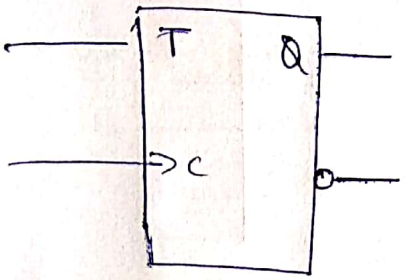
→ The T-FF therefore has two conditions

- when T=0 (J=K=0), C → 0 to 1

  O/P state doesn't change ⇒ Q(t)

- when T=1 (J=K=1), C → 0 to 1

  O/P state complements ⇒ Q'(t)



T	Q(t+1)
0	Q(t)
1	Q'(t).

Edge Triggered FF

The most common type of FF used to synchronize the state change during a clock pulse transition is the Edge-Triggered FF.

→ In this type of FF, output transitions occurs at a specific level of clock pulse.

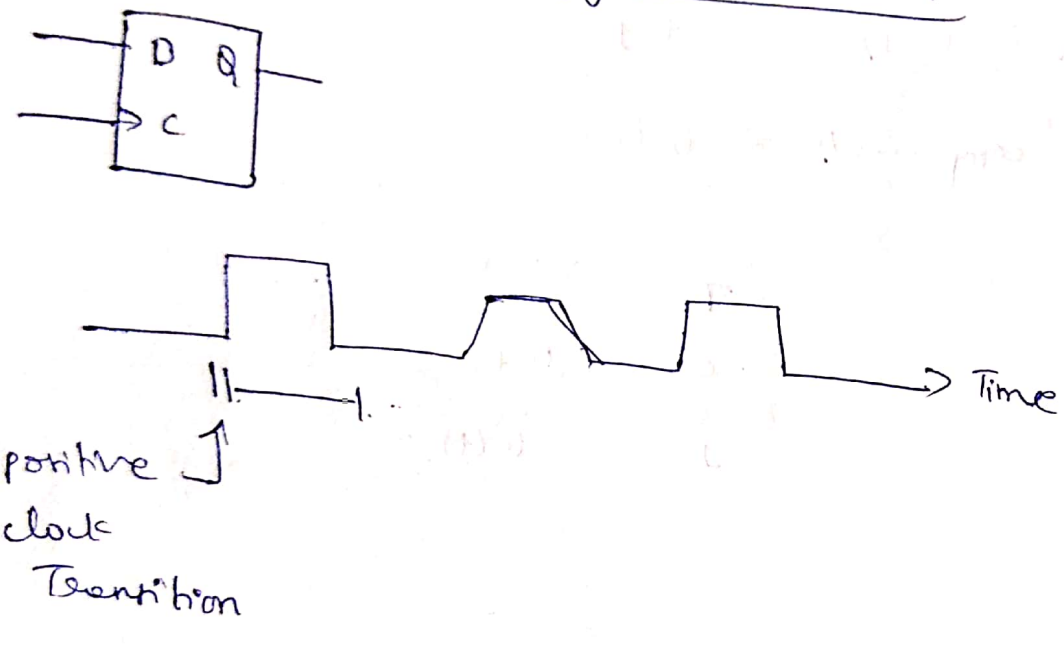
Positive-Edge Transition

Some edge triggered FF cause a transition on the rising edge of the clock signal.

Negative edge Transition

Some FF cause a transition on the falling edge of clock signal

(a) positive-edge trigged D-FlipFlop



→ The value in the 'D' input is transferred to the 'Q' output when the clock makes a positive transition

→ The output cannot change when the clock is in the '1' level ie the '0' level or in transition from 1 to 0

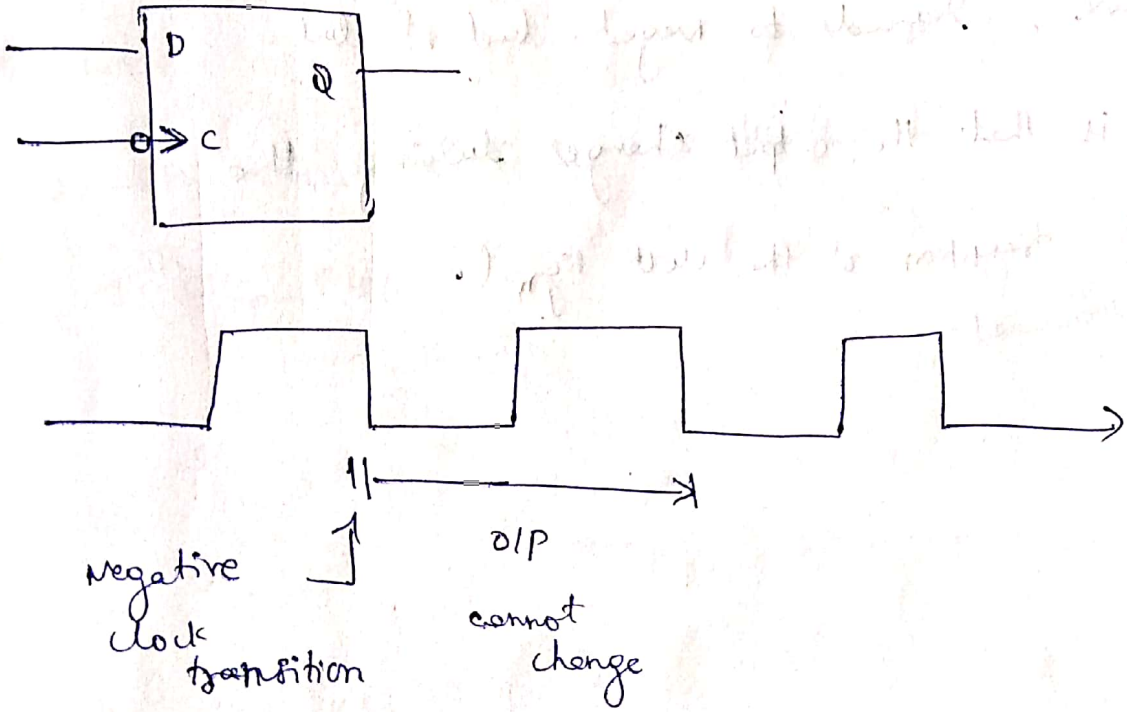
Set up time

The minimum time for effective positive clock transition, in which the D Input must remain a constant value before transition.

Hold Time

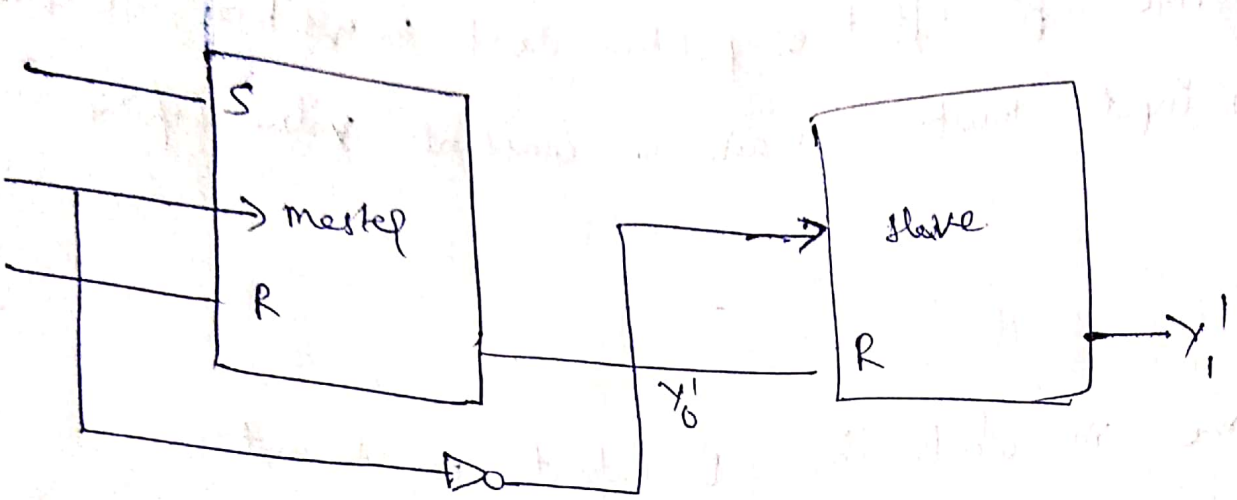
The definite time in which the 'D' output must not change after the positive transition.

1) Negative - edge trigged D-F-F



→ The graphic symbol indicates a negation circle infront of the dynamic

Master slave Flip Flop



The circuit consists of two FF

* First is master, responds to positive level of clock.
* Second is slave, responds to negative level of clock.

The result is that the output changes during the 1-to-0 transition of the clock signal.

Extraction Tables

The characteristic tables of flip-flop specifies the next state when the ~~ip~~ inputs and present states are known.

During the design of sequential circuit we usually know the required transition from present state to next state and to find flip flop input conditions that will cost the required transition.

The table that list the required input combinations for given change of state is called flip flop ~~exitat~~ excitation table

1- ~~SR~~ Excitation Tables for flip flop

(1) SR Flip-Flop

Truth table

~~characteristic Table~~

S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	x

characteristic table:

To findout the value of Q_{n+1} and next state depends on your SR input and Q_n present state and output is Q_{n+1}. So characteristic table is determined by truth table

Characteristic Table

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

Excitation Table

(1) An excitation tables lists the required input combinations for a given change of state

(2) Each table consist of two columns for present state and next state and a column for each input to show how the input is achieved

(3) There are four possible transitions from present state to next state.

Excitation Table

Qn	Qn+1	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

JK Flip-flop

Truth table

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t) complement

Characteristic Table

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Excitation table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

D - Flip Flop

D	Q_{n+1}
0	0
1	1

Characteristic Table.

D	Q_{n+1}	Q_n	Q_{n+1}
0		0	0
0		1	0
1		0	1
1		1	1

Excitation Table.

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Toggle Flip-Flop:

T	~~Q(t+1)~~ Q_{n+1}
0	$Q(n)$
1	Q'_n

Characteristic Table

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table

Q_n	a_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Sequential circuit

The information stored in the memory element at any given time defines the present state of sequential circuits. The present state and the external inputs determine the outputs and the next state of sequential circuits.

Sequential circuits can be specified by a time :

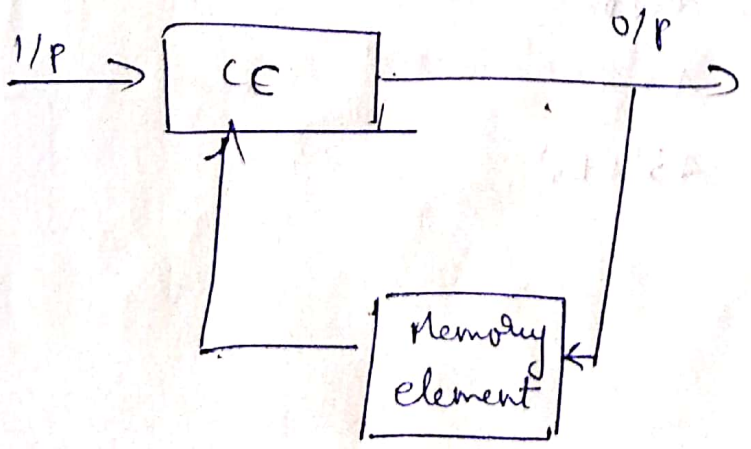
(1) Time sequence

(2) Internal states (present and next state)

(3) Outputs

Examples: Counters and registers

Memory element used in sequential circuit is a flip-flop which is capable of storing two bit binary information



State Table

A sequential circuit is specified by a state table that relates outputs and next states as a function of input and present state. A state table consists of present state, input state, next state, output state.

The declination of a state table consists of first listing of all possible binary combinations of present state and inputs. In this case we have eight binary combinations from 000 → 111

The next state values are then determined from flip-f input equations. The next state A is

$D_A = Ax + Bx$ (since the next state value of each flip is equal to it d input value in present state)

Next state B is

$D_B = A'x$

output equation $y = Ax' + Bx'$

State Table

A is $D_A = Ax + Bx$

$D_B = Ax'$

Output

$y = Ax' + Bx'$

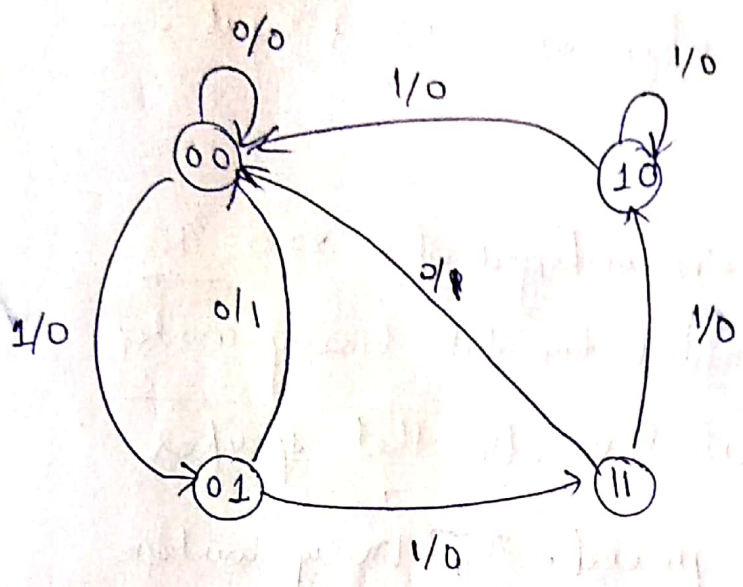
Present state			Next state		Output
A	B	x	A	B	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

state diagram

The graphical representation of state table is called state diagram. State is represented by a circle. Transition is indicated by directed lines. Binary number inside circle is state of flip-flop. The directed lines are labeled with two binary numbers separated by a slash. The input value during present state is first and the number

after flash is output of present state

State Diagram of Sequential circuit

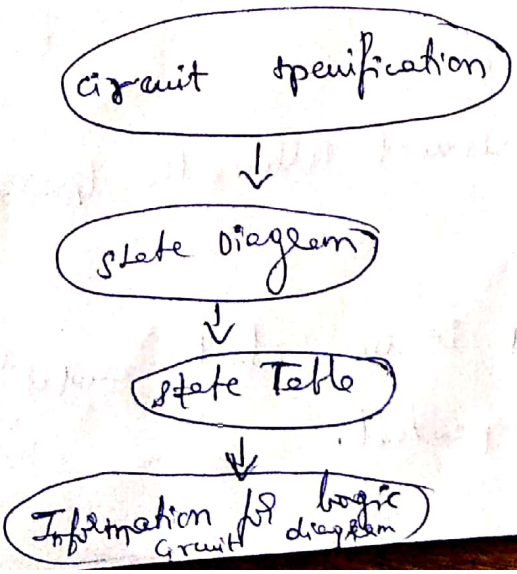


Design example of sequential circuit

The designed procedure consists first translating the circuit specification to state diagram.

The state diagram is converted to state table

From state table we obtain the information for obtaining the logic circuit diagram.



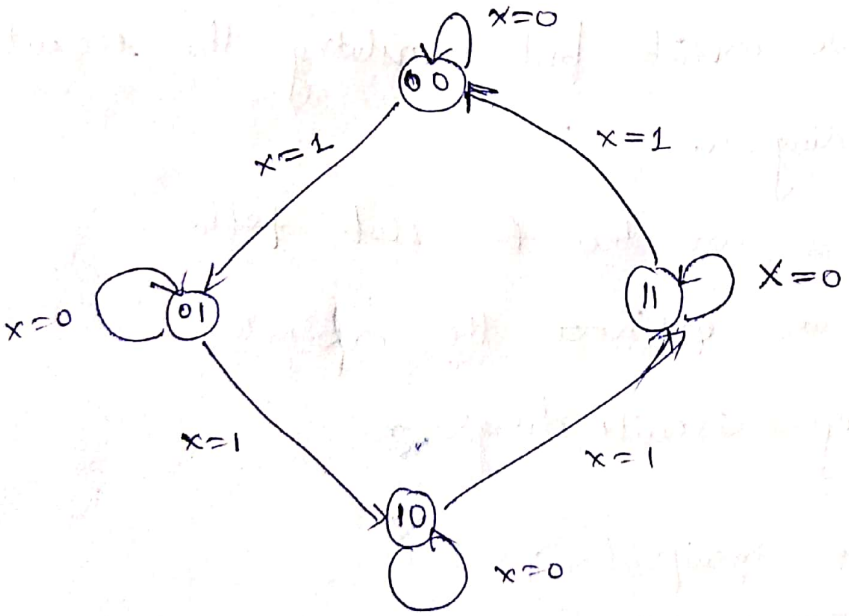
To design a clocked sequential circuit that goes through a sequence of repeated binary states of

00, 01, 10, 11 when an external input.

$x = 1$

The state of circuit remains unchanged when $x=0$. This type of circuit is called two bit binary counter

Input x is the control variable that specifies when the count should proceed. The binary counter needs two flip-flops to represent the two bits



The state of the circuit follow the binary count as long as $x=1$

The state following 11 is 00 which causes the count to be repeated.

If $x=0$ the circuit remains unchanged.

There is no external output for the circuit

The state of the flip flop is considered as output as counter.

Excitation table for binary counter.

The excitation table of sequential circuit is an extension of the state table.

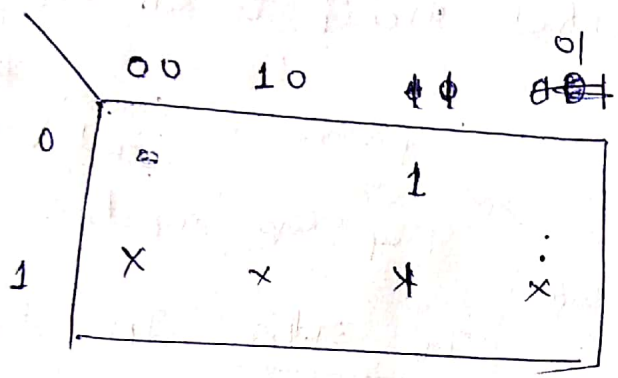
Present state		input	Next state		Flip Flop inputs			
					From JK excitation table present A, next A			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	0	1	x	x	1
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

The simplified boolean function for the combinational circuits can be derived - The input variables are A, B, x

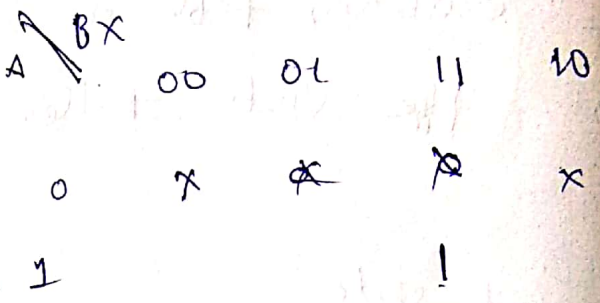
The outputs are J_A, J_B, K_A, K_B.

The information from excitation table is transfered to map

J_A

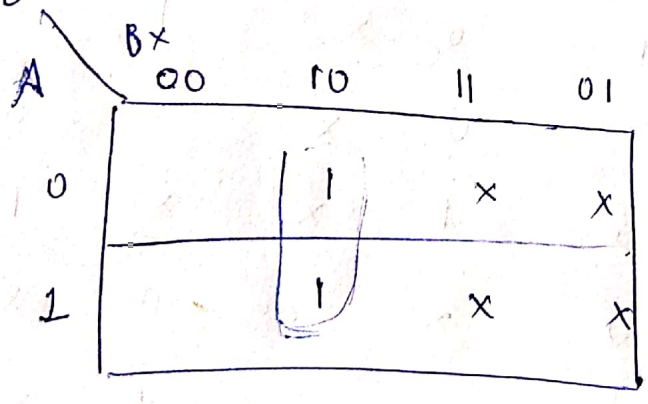


$J_A = Bx$

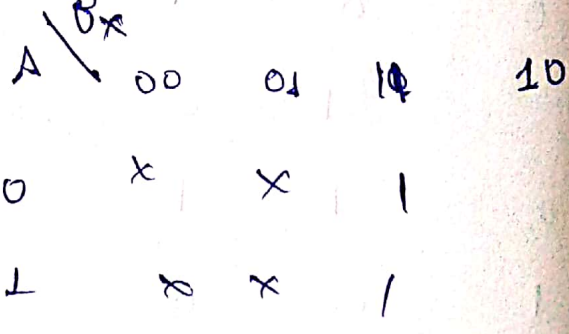


$K_A = Bx$

J_B

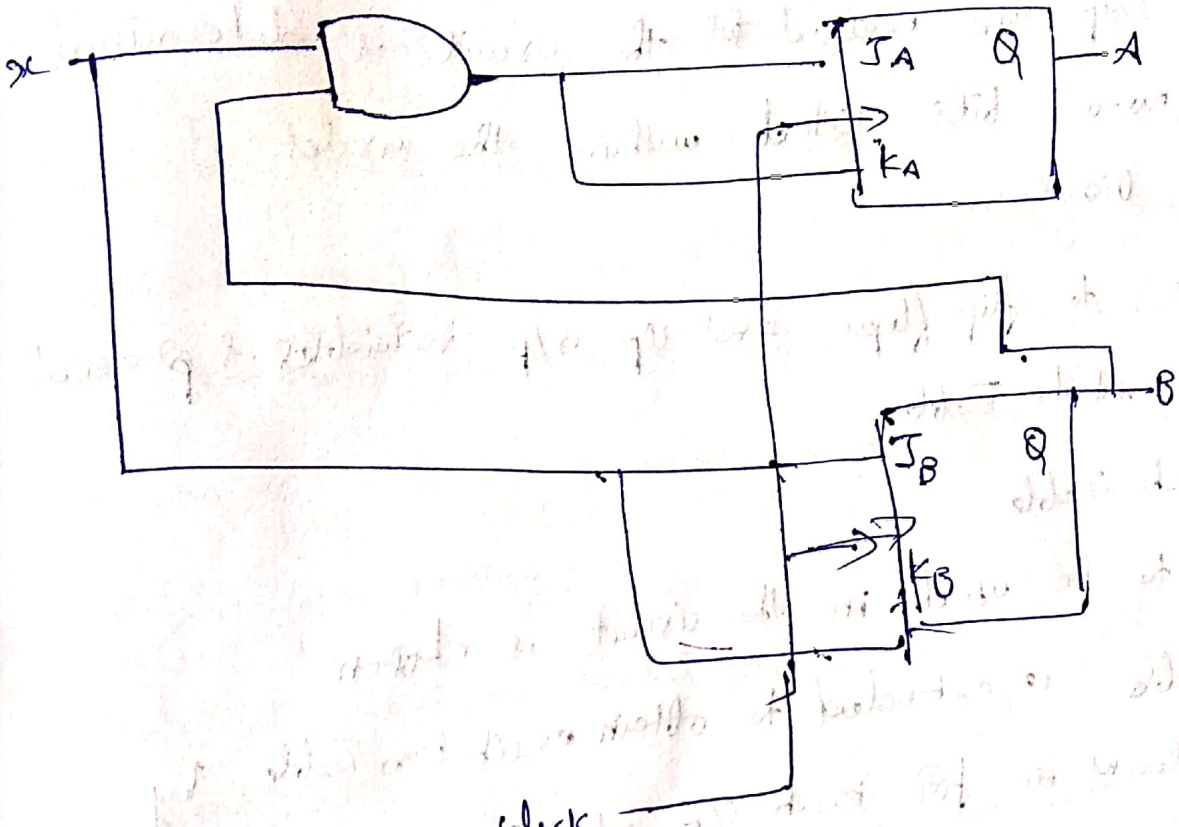


$J_B = x$



$K_B = x$

maps for combinational circuit of counter.



Logic diagram of a 2-bit binary counter.

Design Procedure

- The behaviour of circuit is first formulated in a state Diagram.

→ The No. of flip-flops needed for the circuit is determined from the no. of bits listed within the circles of the state Diagram

→ Assign letters to flip-flops and i/p, o/p variables & proceed to obtain state Table.

→ Obtain State Table

→ FF type to be used in the circuit is chosen

→ The state Table is extended to obtain excitation Table by including columns in for each i/p of FF

→ Truth Tables → K maps (For simplification)

→ Final circuit Diagram

Integrated circuits:

An integrated circuit is a small silicon semi-conductor crystal called a chip containing electronic components for digital gates. The various gates are interconnected inside the chip to form the required circuit. The chip is mounted in a ceramic/plastic container and connections are welded by thin gold wires to external pins to form the Integrated chips (IC).

The no. of chips may vary from 14-100 in small ICs to larger IC package.

→ Each IC has a Number printed on its surface for identification.

→ Each vendor publishes a book/catalog that contains exact description of all necessary information of IC

→ IC technology has increased so that more no. of gates can be packed on IC

i.e. more gates ∝ more functionality

Based on no. of gates the IC package is classified as

SSI - Small scale integration) - (gates < 10)

MSI (Medium Scale Integration)

Consists of 10-200 gates in a single package and digital functions performed on decoders, adders registers.

→ LSI (Large Scale Integration)

Consists of 1000 gates

It includes processors, memory chips and programming modules

VSLSI (Very Large scale integration)

Thousands of gates

large memory arrays, complex micro computer chips.

Digital circuits are classified by specific circuit Technology

They are:

TTL – Transistor Transistor Logic

ECL – Emitter coupled Logic

MOS – Metal Oxide Semi conductor

CMOS – Complementary metal-oxide Semi conductor

1) TTL

This was an evolution of a previous technology that used diodes and transistor for the basic NAND gate called as DTL. Later diodes are replaced by Transistor called as TTL.

2) ECL

This family provides the highest speed digital circuits in integrated form.

- ECL is used in supercomputers & signal processors where high speed is essential

3) MOS

The MOS is a unipolar transistor that depends on the flow of only type of carriers. which may be electrons (n-channel) or holes (p-channel)

- A p channel MOS is referred as pMOS n-channel is referred as NMOS

(4) CMOS

This ~~tech~~ technology uses PMOS + NMOS transistors connected in a complementary fashion in all circuits

Adv - low power consumption
- High packing density

Decoders.

A decoder is a combinational circuit that converts information from encoded inputs to maximum 2^n unique outputs.

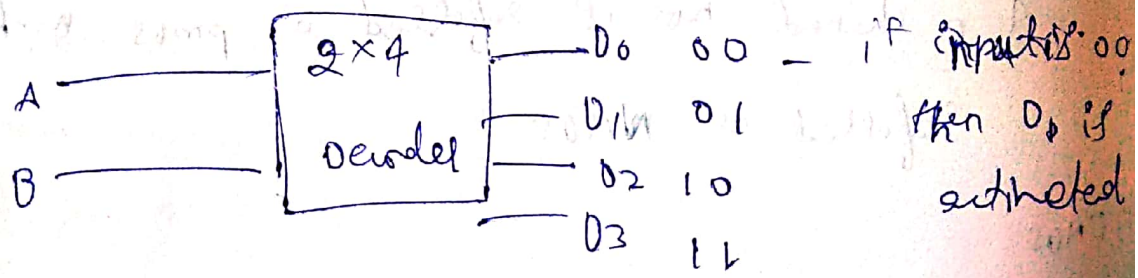
The decoder is a combinal circuit that has n input line and maximum of 2^n output lines. One of the output will be active based on the inputs i.e. decoder detects a particular code. The outputs of the decoders are min terms of n input variables

The decoders may be of following sizes

1×2, 2×4, 3×8, 4×16, ... $n\times2^n$

input lines / output lines

In this only one i/p will be active



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

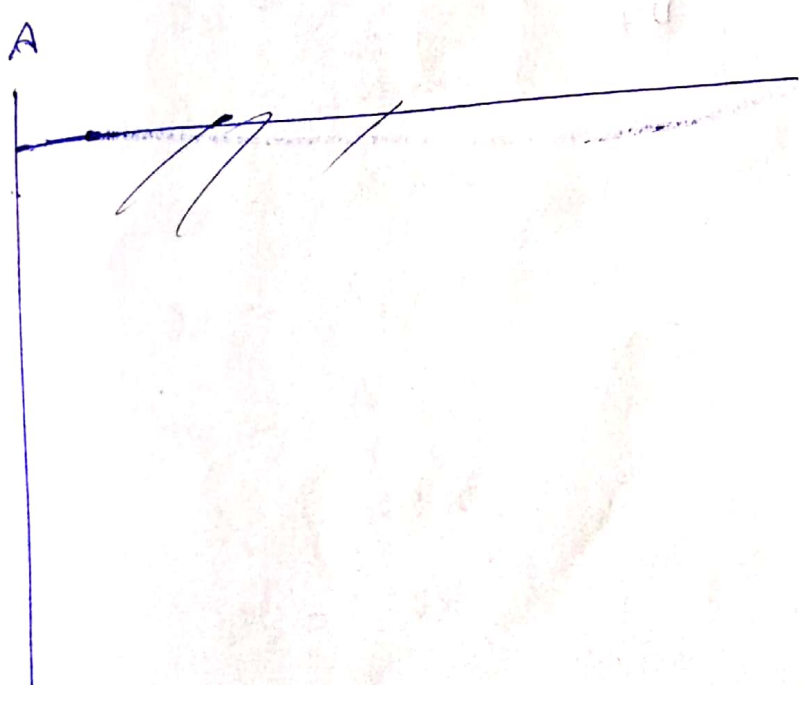
Output Expression

$D_0 = \overline{A}\,\overline{B}$ — D_0 is activated when both A and B are 0,0

$D_1 = \overline{A}\,B$ — D_1 — 0,1

$D_2 = A\,\overline{B}$ — D_2 — 1,0

$D_3 = A\,B$ — D_3 — 1,1



NAND Gate decoder

$E=1$ disabled

$E=0$ is enabled in nand gate

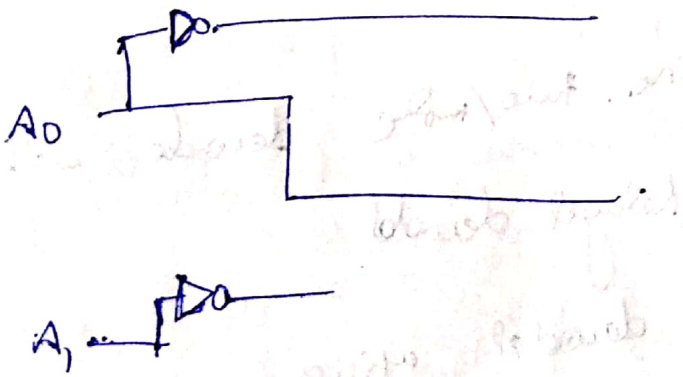
Some decoders are constructed with NAND instead of AND gates.

- NAND gates produces the AND operation

A 2-to-4 line decoder with NAND gates

E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

Logic Diagram:



The circuit operates with complemented outputs and a complemented enable input E

The decoder is enable when E is equal to 0 only one output is equal to 0 at any given time the other 3 o/ps are equal to 1

The o/p whose values is 0 represents equivalent binary no. in inputs A_1 & A_0.

The * circuit is disabled when E=1, whatever may be inputs

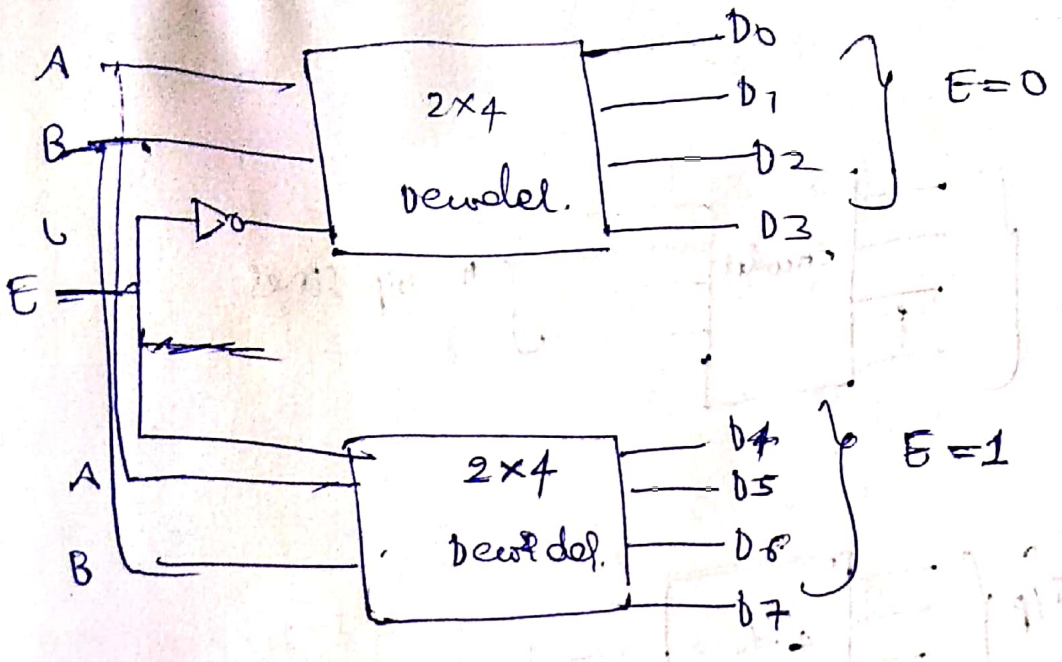
→ when circuit is disabled, none of the outputs are selected & all outputs are equal to 1

Decoder Expansion

When a certain size decoder is needed but only smaller sizes are available.

→ Then it is possible to combine two/more decoders with enable inputs to form a larger decoder.

For example: Design of a 3×8 decoder using 2×4 decoder

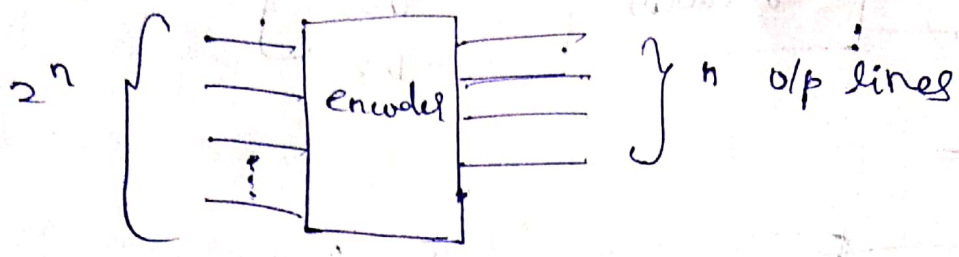


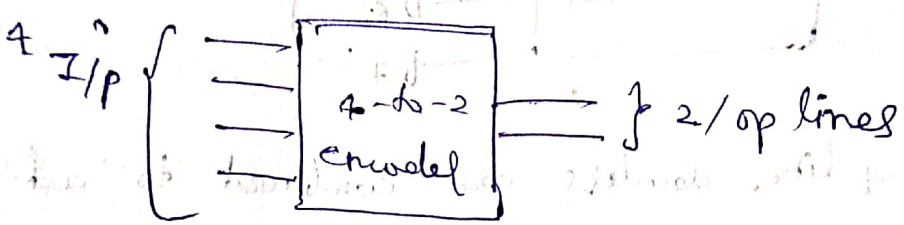
- Two 2-to-4 line decodels are combined to achieve 3-to-8 line decodel
- The two least significant bits of input are connected to both decodels
- The most significant bit is connected to the enable input of one decodel & through an inverter to the enable input of the other decodel

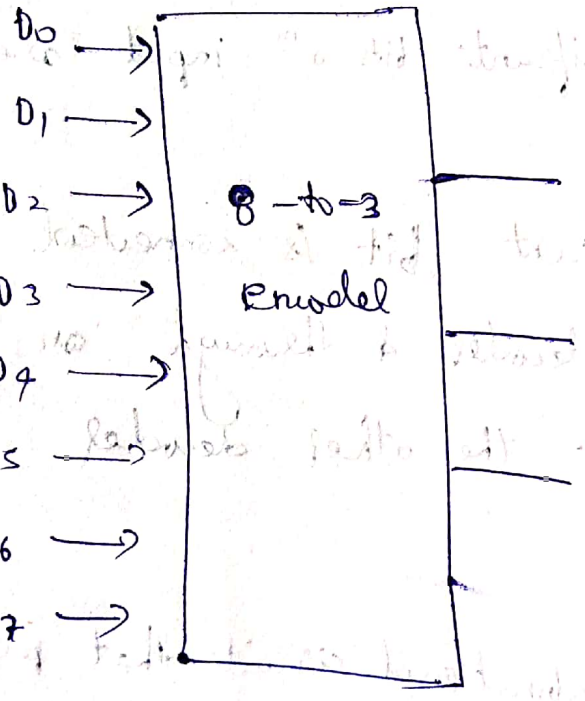
Encodels

An encodel is a combinational circuit that performs the reverse operation of decodel.

It has maximum of 2^n inputs and n outputs, size of encodels may 2×1, 4×2, 8×3, ---, $2^n\times n$.







Example of encoder is octal-to-binary encoder.

E	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A_2	A_1	A_0
1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	0	0	0	1	1
1	0	0	0	0	1	0	0	0	1	0	0
1	0	0	0	0	0	1	0	0	1	0	1
1	0	0	0	0	0	0	1	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1	1
0	x	x	x	----	----	----	----	x	0	0	0

~~output expressions~~

- The encoder can be implemented with OR gates whose inputs are determined directly using Truth table.

$$A_0 = D_1 + D_3 + D_5 + D_7$$

$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

The encoder can be implemented with 3-OR gates

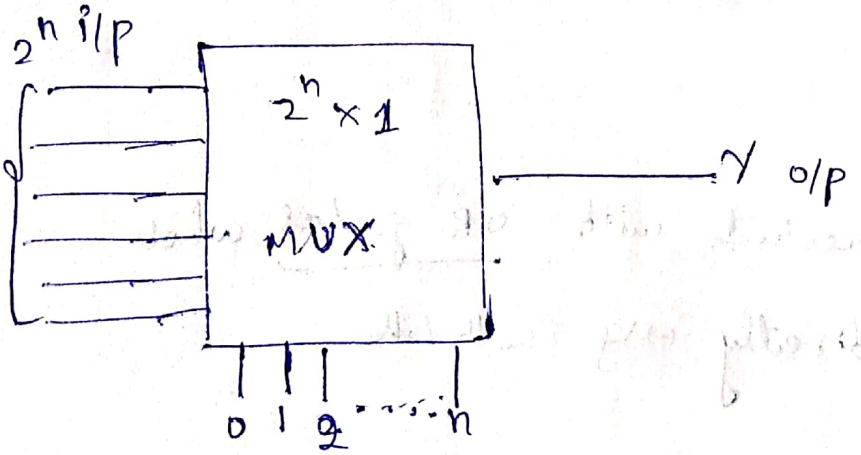
Multiplexer

Multiplexer is a device that receives multiple inputs and this multiple inputs are in the power of 2 i.e. 2^n inputs and one output and there are n selection lines.

A multiplexer is also known as data selector

Digital multiplexer (MUX)

- MUX is a combinational circuit

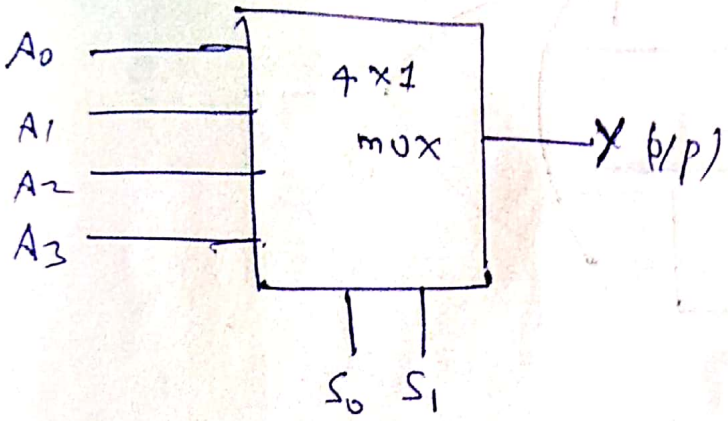


n selection lines

$q = 2^{(2)}$ → selected lines.

i/p's	selection lines	o/p lines	
$2 = 2^1$	1	1	2×1 MUX
$4 = 2^2$	2	1	4×1 MUX
$8 = 2^3$	3	1	8×1 MUX
$16 = 2^4$	4	1	16×1 MUX
2^n	n	1	2^n×1 MUX

Design a 4×1 multiplexer.



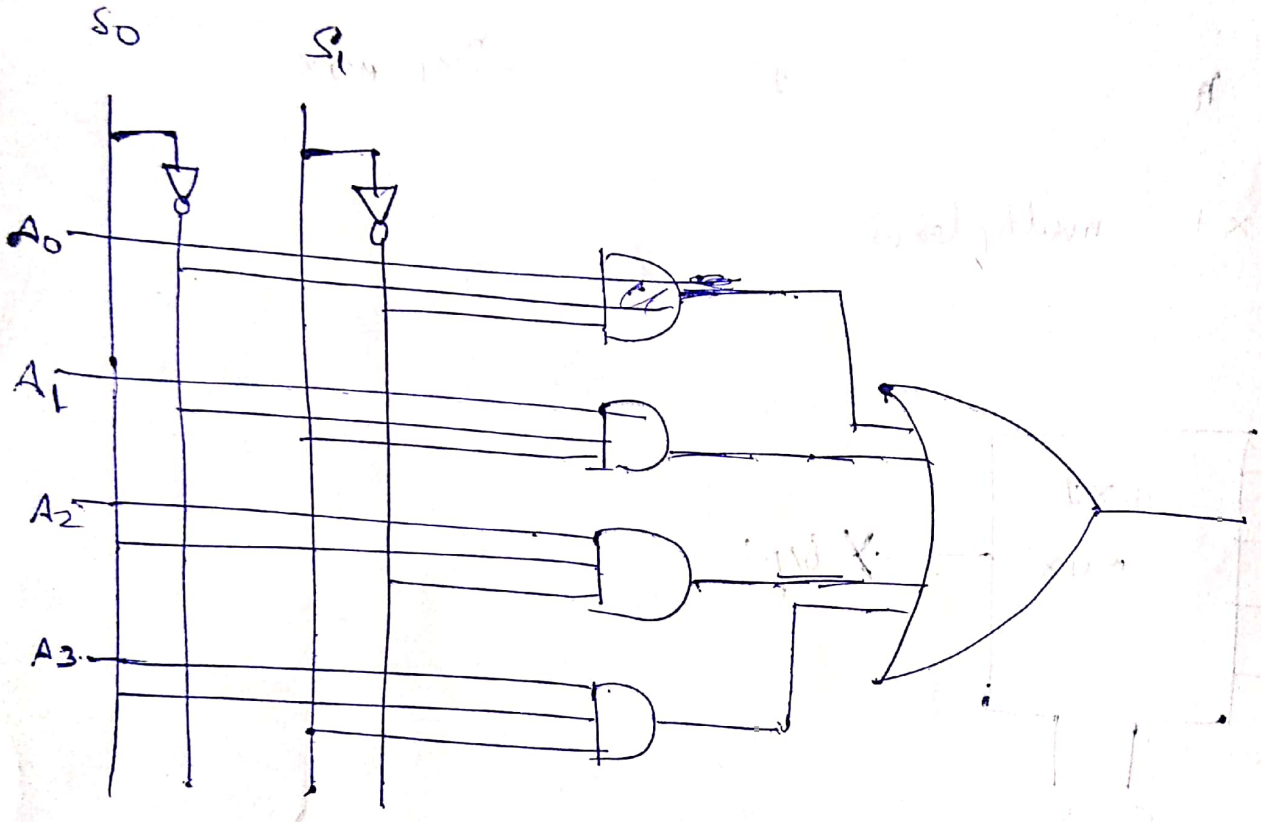
Step2: Truth Table for 4×1 MUX

S_0	S_1	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

Step3: o/p Expression

$$Y = A_0 \bar{S}_0 \bar{S}_1 + A_1 \bar{S}_0 S_1 + A_2 S_0 \bar{S}_1 + A_3 S_0 S_1$$

step 4: Draw the circuit



Design 8 × 1 multiplexer.

Step1: Block Diagram

Number System

→ Decimal, Hexadecimal, octal, binary

→ Binary to decimal
Decimal to binary
Decimal to hexadecimal
Hexadecimal to Decimal
Dec to oct
oct to Dec

Binary to octal
oct to binary
Binary to Hex
Hex to Bin
Hex to oct
oct to hex

Complements

Complements are used in digital computer for simplifying subtraction and logical manipulation

Two types of complement, for each base r system:

(r) 's complement, (r-1)'s complement.

Base 2 → 1's, 2's

$\bar{x} = (2^k - 1) - x$

$\bar{x} = (2^k - x)$

Base 8 → 7's, 8's

$\bar{x} = (8^k - 1) - x$

$\bar{x} = (8^k - x)$

Base 10 — 9's: $\bar{x} = (10^k - 1) - x$

Base 10 — 10's: $\bar{x} = (10^k - x)$

Base 16 — 15's: $\bar{x} = (16^k - 1) - x$

Base 16 — 16: $\bar{x} = (16^k - x)$

Given a number n is befe 4 having n digits

The $(r-1)$'s complement of n is defined as

$(r^n - 1) - N$

Ex: The 9's complement of n is $(10^n - 1) - N$.

→ Ex. The 9's complement of 546700 is

$999999 - 546700 = 453299.$

→ The 9's of 453299

$999999 - 453299 = 546700$

→ The 1's complement of 1011001 is

1111111 − .

The r's complement of n-digit number N in base r is defined as $r^n - N$

This is same as adding 1 to the (r-1)'s complement.

Find the 10's complement of 2389 is.

First Find 9's complement and add 1.

```
  9999
  2389
------
  7610
    +1
------
  7611
```

```
    11
010011
   +1
------
010100
```

Find the 2's complement of 101100 is 010011 + 1 = 010100.

Subtraction of unsigned n-digit numbers M - N

* Add M to the r's

Example of M < N : 13250 − 72532 = −59282

M = 13250

10's complement of N = 99999
− 72532
27467
+ 1
27468

if the no. is − find the complement.

Sum of M = 13250
10's complement of N = 27468
40718

Example for X = 1010100 and Y = 1000011.

X = 1010100

2's complement of Y = ~~1~~ ~~1111111~~ 0111100
+1
0111101

~~Discard~~ Sum = 10 101 00
01 111 01
100 0001

Discard and carry. 010001

(ii) Y - X =

Y = 1000011

2's of X = 0101100

1101111

as negative find 2's complement and minus.

- 0010001

1101110

Fixed - Point Representations:

Positive integers and

+6 00000110

+13 00001101

+ 00010011

-6 00000110 — Find 2's complement

+13 00001101

11111001 +1

-6 11111010

+13 00001101

+7 00000111

00000111 +1

11111000

+1

11111001

+6 0000 0011

−13 1111 0011

1111 1100

−6 1111 1010

−13 1111 0011

1111 1101

Binary code:

BCD, 2421 code, 84-2-1

	8421 BCD	84-2-1	2421	Excess 3 code
			0+3	0011
0	0000	0000	0000	0100
1	0001	0111	0001	0101
2	0010	0110	0010	0110
3	0011	0101	0011	0111
4	0100	0100	0100	1000
5	0101	1011	1011	1001
6	0110	1010	1100	1010
7	0111	1001	1101	1011
8	1000	1000	1110	1100
9	1001	1111	1111	

5+3 → 1000

6+3 → 1001

Complement

Biquinary code — Length of one digit is 7 bits only

Two bits are 1 and other all 5 bits are 0

It is a weighted code

It is used for error detection

	5	0	4	3	2	1	0	
0	0	1	0	0	0	0	1	0+0
1	0	1	0	0	0	1	0	0+1
2	0	1	0	0	1	0	0	0+2
3	0	1	0	1	0	0	0	0+3
4	0	1	1	0	0	0	0	0+4
5	1	1	0	0	0	0	0	5+0
6	1	0	0	0	0	1	0	5+1
7	1	0	0	0	1	0	0	5+2
8	1	0	0	1	0	0	0	5+3
9	1	0	1	0	0	0	0	5+4

Gray code or unit distance code

Two successive values differ in only 1 bit

It is an unweighted code

No positional weights

It is a unit distance code and minimum error code

~~How to correct~~

Error detection Codes

An error detection code is a binary code that detects the error during transmission.