MD IFTAKHAR KABIR SAKUR

25th BATCH

COMPUTER AND COMMUNICATION ENGINEERING

International Islamic University Chittagong

COURSE CODE: CCE-2411

COURSE TITLE: DIGITAL LOGIC DESIGN

COURSE TEACHER:

PROF. RAZU AHMED

CCE-2411

Digital Logic Design

Digital Electronics:-

Branch of electronic deals with Digital Circuit. Digital Circuit handle digital signal that accepts only two distinct value.

HIGH & LOW or Binary 1 or 0.

DLD is foundational to the Fields of Electrical Engineering & Computer Engineering.

These characteristics may involve power, current, Logical Function, protocol & user input. DLD used to develop hardware, (circuit boards & Microchip processors).

This hardware processes (User Input, System protocol, & Other data in Computers, Navigational

Cell phones or other high-tech systems.

Analog:- A quantity that has continuous values.

Examples:- Air temperature (quantities that can't change abruptly, instantaneously)

Digital:- A quantity that has discrete values.

Examples:- Air temperature sampled at a given period.

* Digital data can be processed & transmitted more efficiently & reliably than analog data.

→ Digital Data requires less space

→ Digital Data reduces noise

Digital Signals:-

⇒ Through digitizing process:

→ Analog signals as input

→ Sampling (at certain frequency rates)

→ Quantization

→ Digital signals

An Analog Electronic System.

Original Sound waves → Microphone → Audio Signal → Linear Amplifier → Amplified Audio Signal → Reproduced Sound waves

A System Using Digital & Analog Methods

CD Drive → Digital Data (Binary Beat) → Digital-to-Analog Converter → Analog Reproduction of Music Audio Signal → Linear Amplifier → Speaker

Binary Digits

→ High (1) & Low (0)

→ positive Logic: 1 is High
0 is LOW

→ Negative Logic: 0 is HIGH
1 is LOW

→ HIGH & LOW are actually representing voltage Level.

(7) Codes :- groups of bits □ to represents Numbers, symbols, etc

Logic Levels

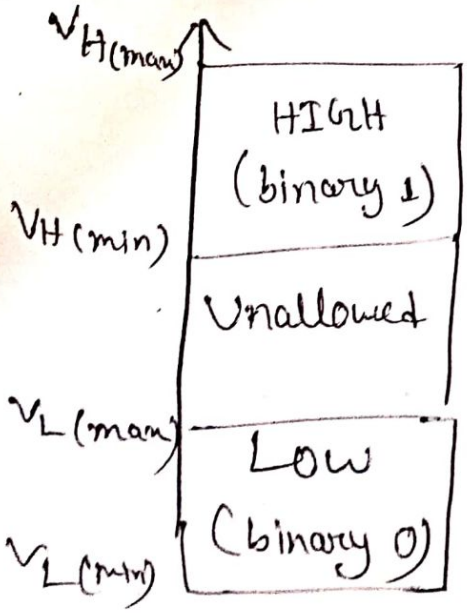
=> The voltages to represent a 1 & 0.

HIGH:-
within a specified minimum & maximum high voltage value.

LOW:-
within a specified minimum & maximum low voltage value.

Example:-

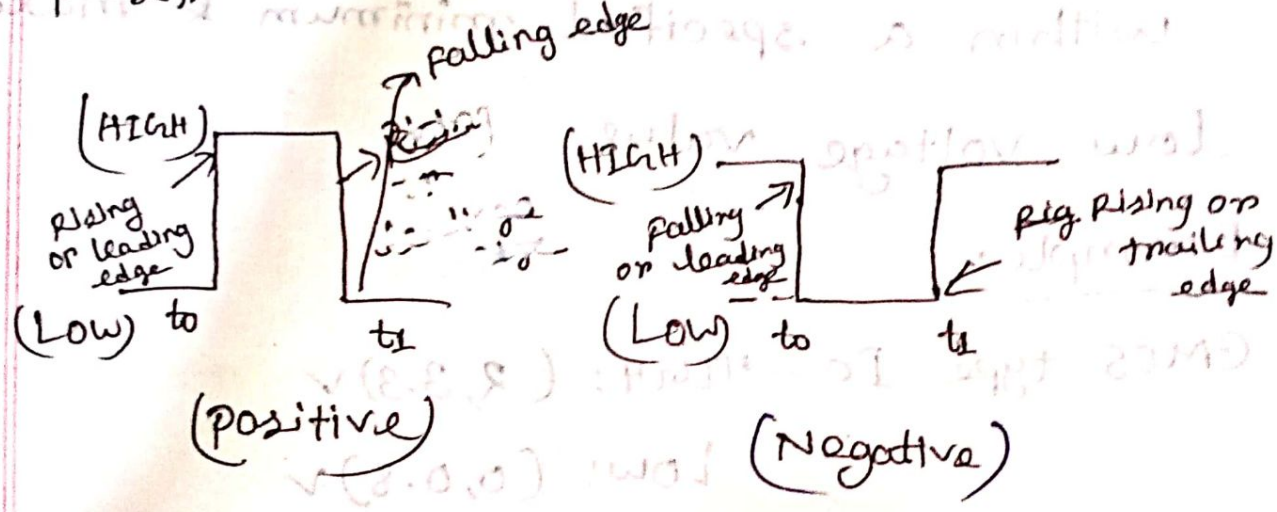
CMOS type IC - HIGH: (2, 3.3) V
LOW: (0, 0.8) V

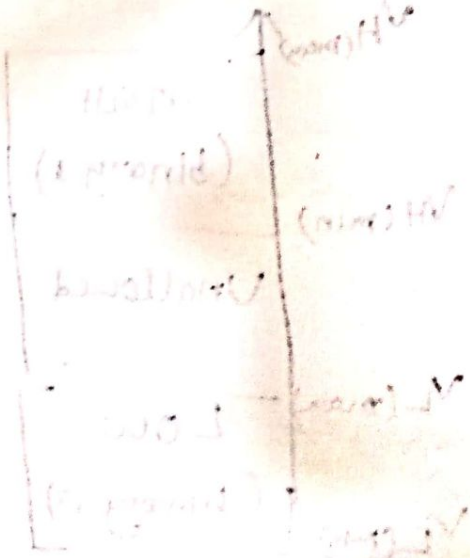


Digital Waveforms

⇒ Consists of voltage level that are changing back and forth between HIGH & LOW levels.

There are positive-going & Negative-going pulses.





More about Digital waves

pulse (ideal case):-

leading & trailing

- **Leading edges:-**

This can be rising or falling edge depending on the pulses type.

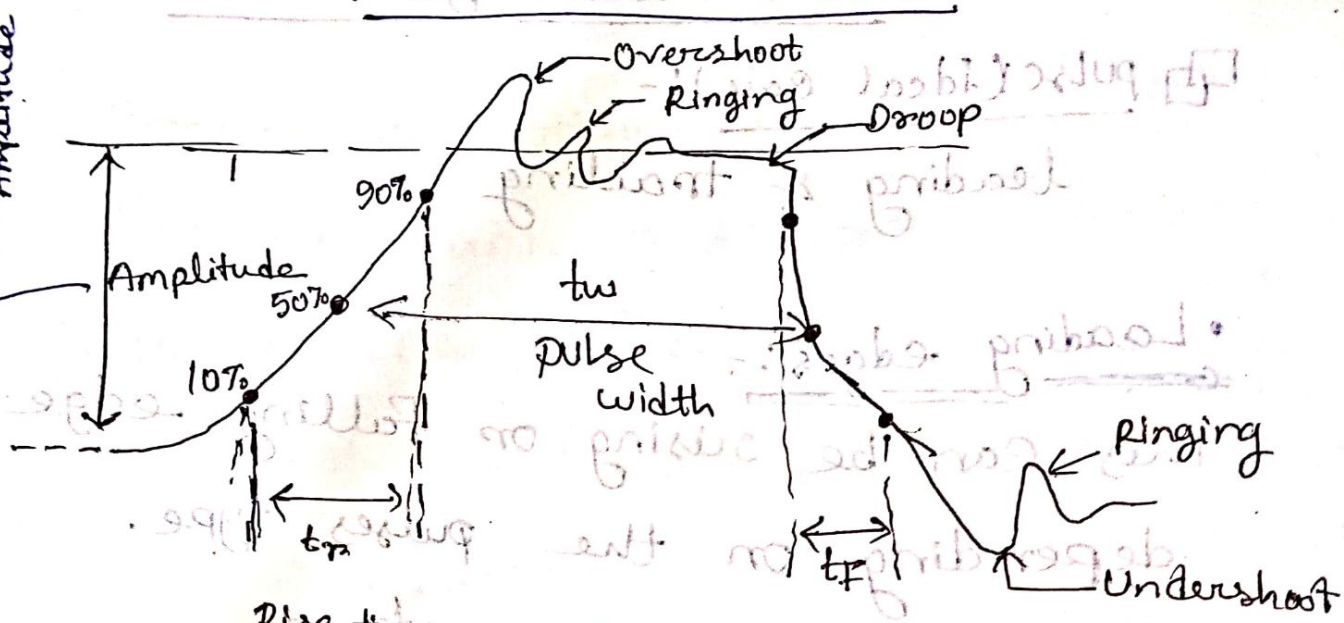
It starts at time ~~to~~ t_0.

- **Trailing Edges:-**

This can be rising or falling edge depending on the pulses type.

This edge occurs at time t_1.

pulse (NO Ideal Case)



Height measured between HIGH & LOW (or vice versa)
It is measured within 10% to 90% of pulse Amplitude

Rise time
(Time required to go from Low to HIGH)

Fall Time
(Time required to go from HIGH to LOW)

Overshoot :- Saturation

Ringing :-

Droop :-

* Duration of pulse measured at 50% points on the rising and falling edges.

Waveform Characteristics

Pulse Trains:-

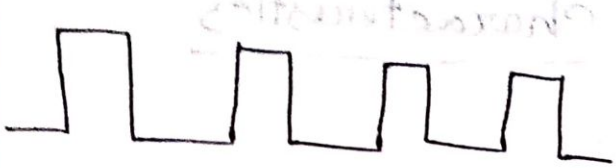
Series of pulses can be found in Digital Systems are called as pulse Trains.

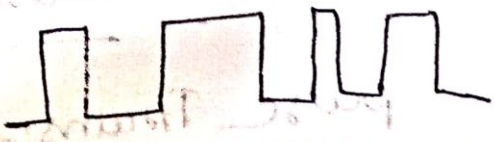
This can be classified as periodic & Non-periodic (A waveform does not repeat itself at a fixed interval).

periodic:- Repeating the same waveform at a fixed intervals.

- Total of time that a waveform repeat itself is called period.
- Rate of how many times a waveform repeats itself.

Duty Cycle:- Ratio of the pulse width to the period in percentage.





Nonperiodic

Period = $T_1 = T_2 = T_3 = T_4$

Periodic

(*)

Duty cycle:-

$T = 1/F$

Or, $F = 1/T$

So Duty cycle = $tw/T \times 100\%$

Bit time:- Each bit in a sequence occupies a defined time interval.

Clock:- A basic timing waveform that is used to synchronize all waveforms in digital systems.

Timing Diagram:-

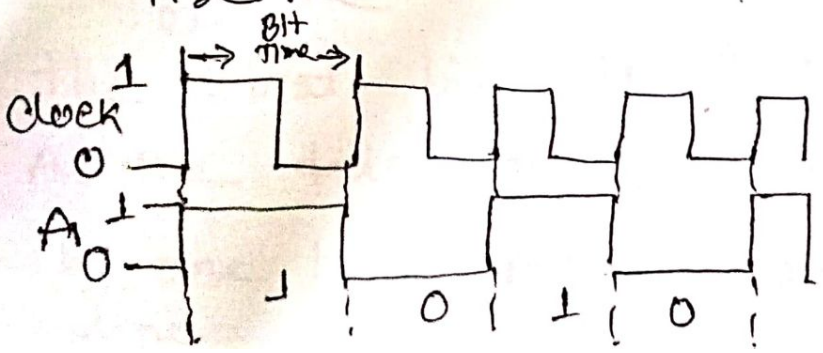
A graph OF digital waveforms showing the actual time relationship OF two or more waveforms, and how each waveform changes in relation to the others.

clock

→ Must be periodic

→ Used to synchronize all waveforms in digital Systems.

→ Each intervals between pulses in clock equals the time For one bit. & clock does not carry any information itself.



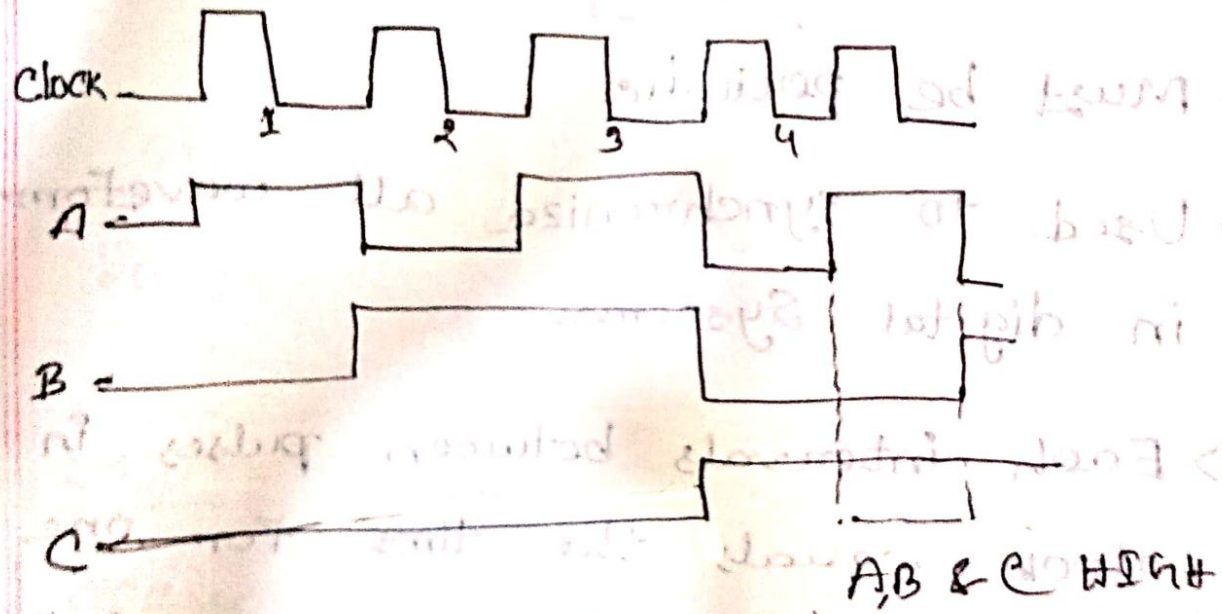
... Bit sequence represented by waveform A

Timing Diagram

The states of all waveforms at any specified time. The exact time a waveform changes relative to other waveforms

~~In the ex~~ ~~For an example:-~~

Figure of Timing Diagram



Data Transfer

→ Groups of bits that convey some type of information.

→ Binary data which are represented by digital waveforms are called Digital System.

→ To make sure digital systems work accordingly Method of transfer

→ Serial

→ parallel

Serial Transfer :-

Datas are transferred in serial form from one point to another.

During the time interval from t_0 to t_1 the First bit is transferred.

As the data have to transferred to one by one the speed is slow. Requires only one line.

Parallel Transfer :-

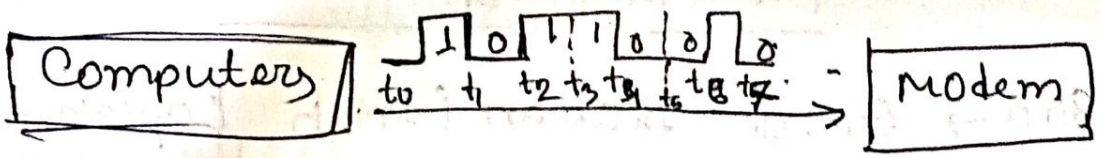
Datas are transferred in parallel form from one point to another.

All bits in a group are sent out on separate lines simultaneously.

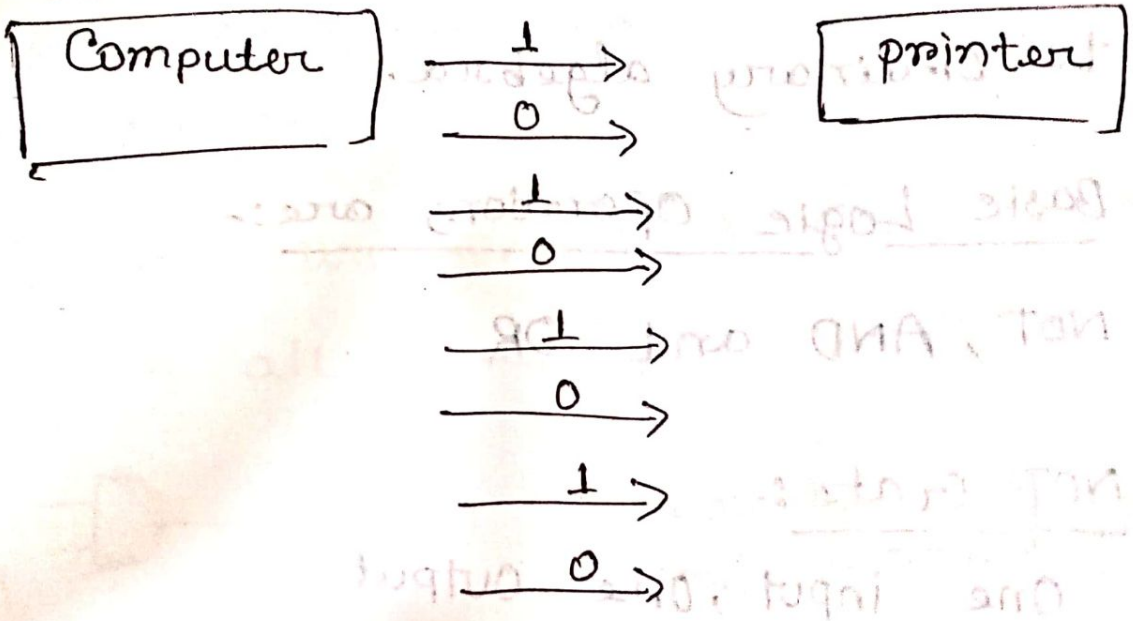
A few bits can be sent at one time.

Requires a few lines.

	Serial	parallel
Transfer Type	Serial	parallel
Speed	Slow	Fast
Cost	Low (good)	High (Not Good)



Serial transfer of 8 bits of binary data from computer to modem. Interval to to t_7 is first.



Basic Logic Operations

1850 → George Boolean (Irish Logician & Mathematician)

He developed a mathematical system for formulating logic statements with symbols so that problems can be written & solved in a manner similar to ordinary algebra.

Basic Logic operations are:-

NOT, AND and OR

NOT Gate:-

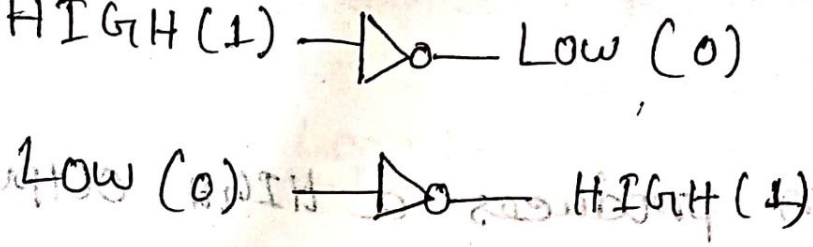
One input, one output

→ It is also known as inverter

→ The output will always be opposite to the input.

* Operation that changes one logic level to the opposite logic level.

NOT Gate

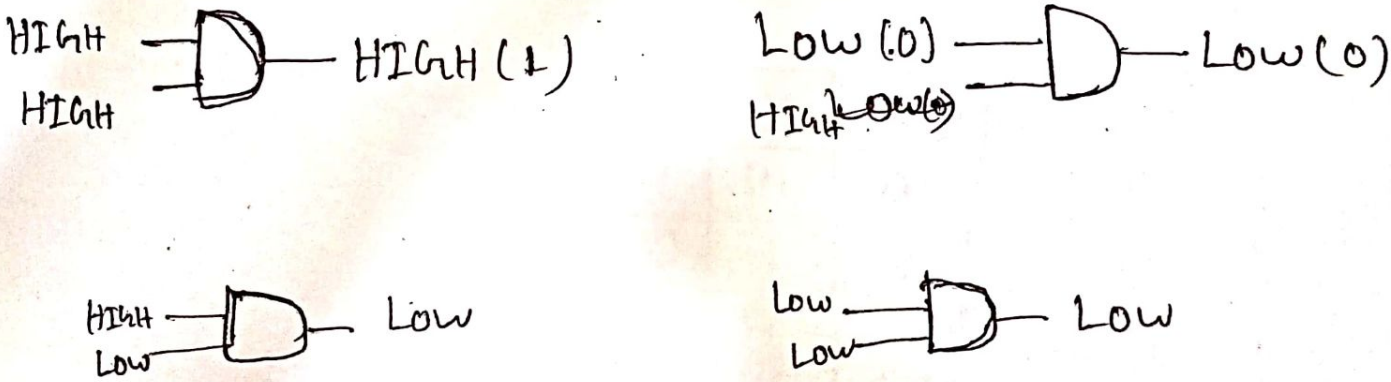


AND Gate

Input : at least two

⇒ Output : one

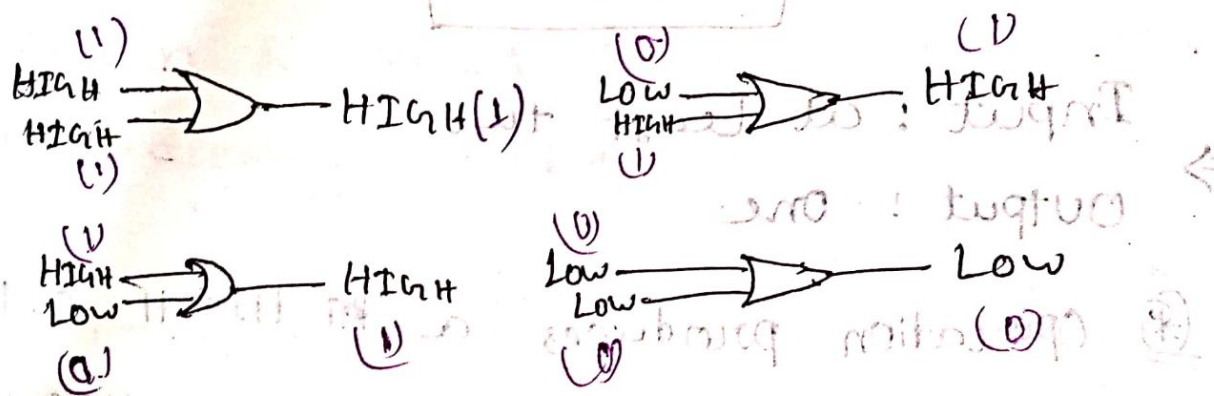
(*) Operation produces a HIGH output only when all the inputs are HIGH



OR Gate

Input can be at least two and output is one.

=> operation produces a HIGH output when one or more inputs are HIGH.



Chapter-02 PdF: Lecture_4

(Combinational Logic Circuits)

Binary Logic:-

Take two ~~variables~~ discrete values: It deals with binary variables.

~~And~~

AND → xy or $x \cdot y$

x	y	xy
0	0	0
1	0	0
0	1	0
1	1	1

OR → $x+y$

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT → x'

x	x'
1	0
0	1

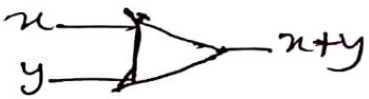
Logic Gate

Each of basic operations can be implemented in hardware using a logic gate.

- Symbols for each of the logic gates are shown below

AND

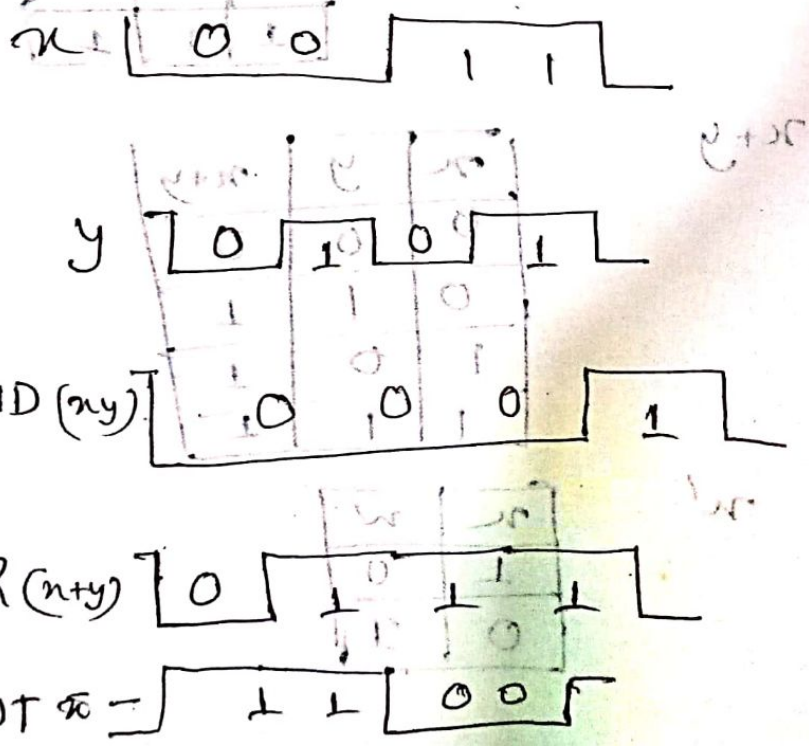
OR



NOT

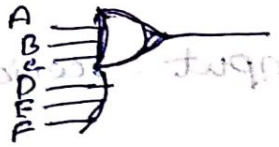
x	y	xy
0	0	0
0	0	0
0	1	0
1	1	1

Timing Diagram



Gates More than two Inputs

$F = A + B + C$



$F = A + B + C + D + E + F$

Boolean Function:-

$F(x, y, z) = (x + y')z + x'$

→ (x, y, z) are the input variables. Representing 0 & 1.

→ The function above has four literals: x, y', z & x'

→ NOT has the highest precedence.

→ Fully parenthesized, the function above would be kind of messy:-

$F(x, y, z) = (((x + (y'))z) + x')$

Truth Table

⇒ A truth table shows all possible inputs & outputs of a function. Remember that each input variable represents either 1 or 0.

There are only finite number of values (1 & 0). Truth table themselves are finite.

$F(x, y, z) = (x + y')z + x'$

$F(0,0,0) = (0+1)0 + 1 = 1$

$F(0,0,1) = (0+1)1 + 1 = 1$

$F(0,1,0) = (0+0)0 + 1 = 1$

$F(1,0,0) = (1+1)0 + 1 = 1$

$F(1,0,1) = (1+1)1 + 0 = 1$

$F(0,1,1) = (0+0)1 + 1 = 1$

$F(1,1,0) = (1+0)0 + 0 = 0$

$F(1,1,1) = (1+0)1 + 0 = 1$

Rules & Regulation OF DLD Circuit

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

7408 - IC (যেভাবে Draw করতে হবে)

(1) Not gate:- 1 no input → 2 no. output

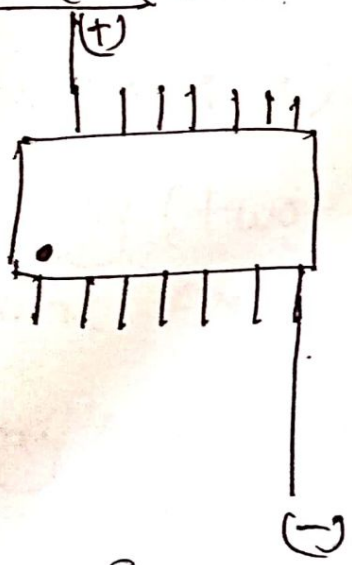
3 " " → 4 no "

(1) And gate:- 1 no & 2 no port ⇨ Input

→ 3 no port ⇨ output

Or gate এও সেইম

IC এর চিত্র:-



→ যেকোনো IC এর জন্য-

⇒ Output এর উপর LED বা output এর সাথে যুক্ত থাকবে।

Switch

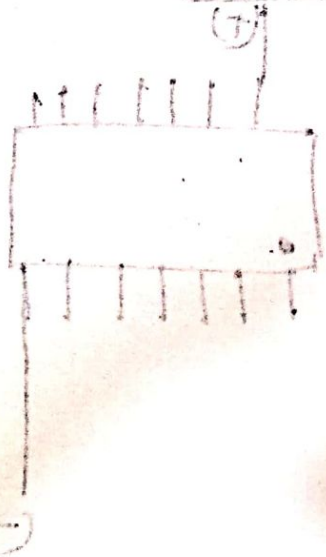
দুইটি Terminal

⇒ Left one → 1 → positive

⇒ Right one → 2 → Negative

⇒ Common টি থাকবে IC circuit এর সাথে যুক্ত।

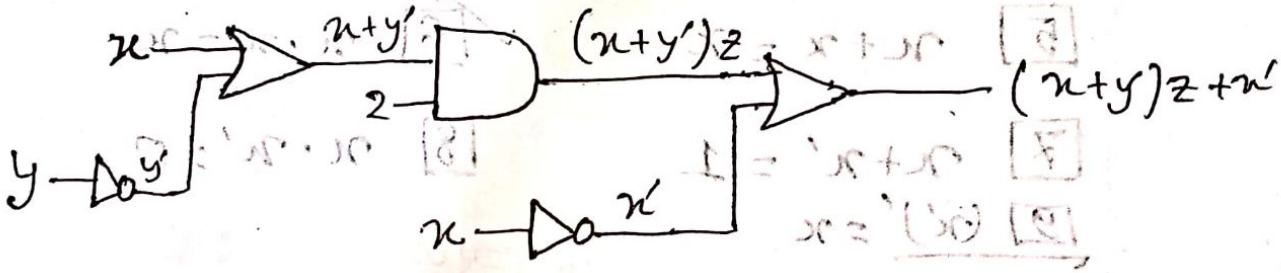
⇒ Resistor use করা হয় IC circuit বাঁচানোর জন্য।



Expressions and circuits

⇒ যেকোনো Boolean Expression can be converted into a circuit by combining basic gates.

⇒ $(x+y').z + x'$



Boolean Algebra

A set of elements B, which needs at least two elements (0 & 1).

⇒ Two binary (two-argument) operations OR (+) and AND (.).

⇒ A unary (one-argument) NOT (').

OR = + হয়

AND = গুণ অবস্থায় থাকে

- 7, 8, 9, 16, 17 axioms deal with complement operation
- 3, 6, 15 (Specially 15) are different from regular Algebra.

The axioms below must always be true

~~1 $x+0=x$ 2 $x+1=1$ 3 $x+x=$~~

1 $x+0=x$	2 $x\cdot 1=x$
✓ 3 $x+1=1$	4 $x\cdot 0=0$
5 $x+x=x$	✓ 6 $x\cdot x=x$
7 $x+x'=1$	8 $x\cdot x'=0$
9 $(x')'=x$	

10 $x+y=y+x$ 11 $xy=yx$ → Commutative

12 $x+(y+z)=(x+y)+z$ 13 $x(yz)=(xy)z$ → Associative

14 $x(y+z)=xy+xz$ ✓ 15 $x+yz=(x+y)(x+z)$ → Distributive law

16 $(x+y)'=x'y'$ 17 $(xy)'=x'+y'$ → De Morgan

(*) Exchange all ANDs with ORs and 0s with 1s.

(*) The duality principle of Boolean Algebra:-

A boolean equation remains valid if we take the dual of the expression on both side of the equal signs.

Axioms Real?

yes they are

DeMorgan's Theorem:-

x	y	x+y	(x+y)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

x	y	x'	y'	x'y'
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

As both are same, this shows that (x+y)' & x'y' are equivalent.

$$\overline{x_1 + x_2 + x_3 + \cdots + x_n} = \overline{x_1}\,\overline{x_2} \cdots \overline{x_n}$$

Simplification with axioms

$x'y' + xyz + x'y$

$= x'(y + y') + xyz \quad [x'y' + x'y = x'(y' + y)]$

$= x' \cdot 1 + xyz \quad [\because y + y' = 1]$

$= x' + xyz$

$= (x' + x)(x' + yz) \quad [\text{Distributive}]$

$= 1 \cdot (x' + yz) \quad [\because (x' + x) = 1]$

$= x' + yz$

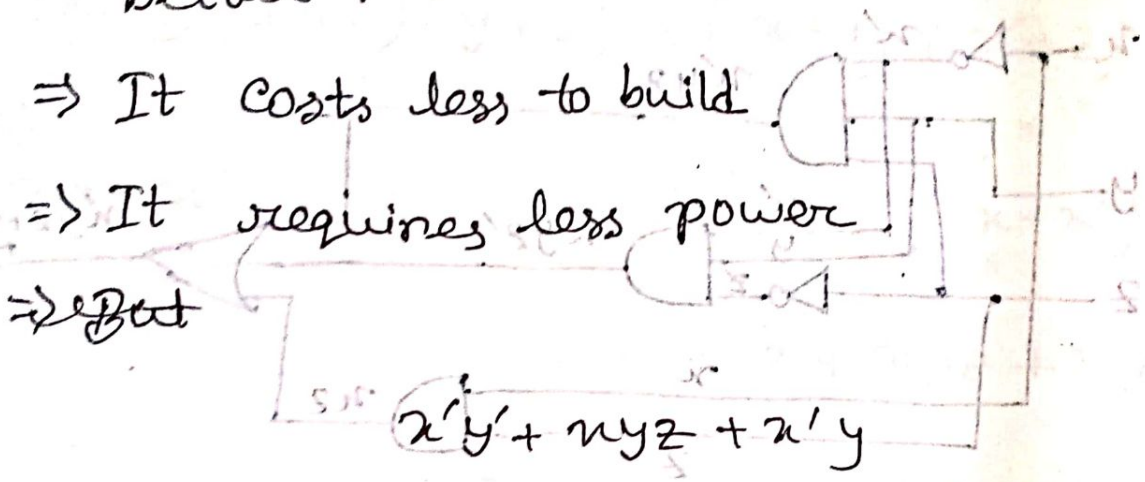
Comparing Two Circuits

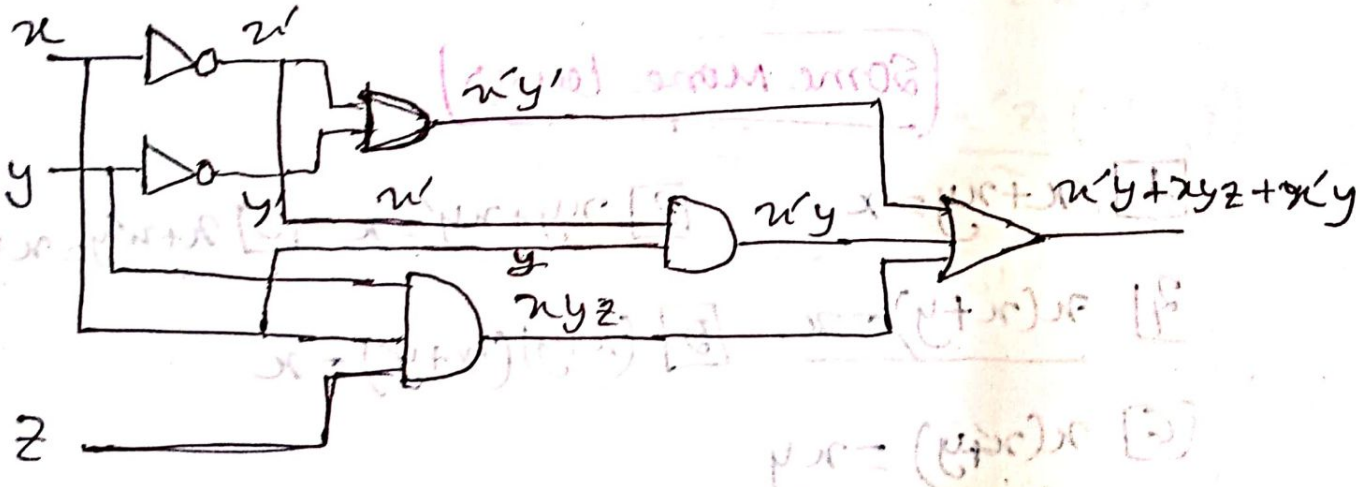
=) In general the one with fewer gates is "better" :-

⇒ It costs less to build.

⇒ It requires less power.

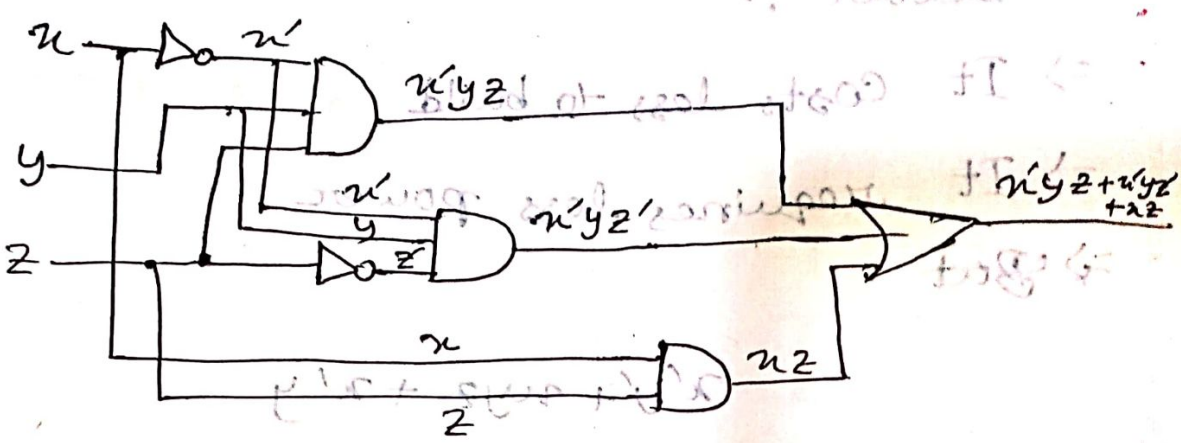
⇒ But





Prime & Bar are same

$\Rightarrow x'yz + x'yz' + xz$



Some More Laws

1. $x + xy = x$
2. $xy + xy' = x$
3. $x + x'y = x + y$
4. $x(x+y) = x$
5. $(x+y)(x+y') = x$
6. $x(x+y) = xy$

Consensus Theorem

$xy + x'z + yz = xy + x'z$ on its dual

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

$\Rightarrow$

$xy + x'z + yz = xy + x'z + yz(x+x')$

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

$\Rightarrow xy + xyz + x'z + x'yz$

$= xy(1+z) + x'z(1+y)$

$= xy + x'z$

The Complement of a function

যেখানে 0 থাকবে সেখানে 1 হবে

যেখানে 1 " 0 হবে

$F(x, y, z) = x' + xyz'$

x	y	z	F(x,y,z)

→ $F'(x, yz)$

[যেখানে 0 হবে সেখানে 1 এবং যেখানে 1 হবে সেখানে 0 বসাতে হবে]

আর এটাই হলো Complement of a function.

A Dual:- A function is said to be self dual if and only if its dual is equivalent to the given function.
If a given function is $F(x,y,z) = (xy+yz+zx)$ then its dual is $F_d(x,y,z) = (x+y)(y+z)(z+x)$

Complementing a function Algebrically

De Morgan's law can be used to keep "pushing" the complementing inwards.

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

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

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

$$= x' + (y'z')' + (yz)' \quad \text{[De Morgan's law]}$$

$$= x' + (y+z) + (y'+z') \quad [$$

(*) If $F(x,y,z) = x(y'z' + yz)$

The dual of F is $F_d = (x + (y'+z')(y+z))$

Dual means
Interchanging multiplication - Addition
0's & 1's
→ It is denoted F_d

So, Complementing OF $F_d = x'+(y+z)(y'+z')$

So, $F'(x,y,z) = x'+(y+z)(y'+z')$

Standards forms of expression

A Sum OF products (SOP) expression Contains:-

→ Only OR(sum) operations at the "outermost" level.

→ Each term that is summed must be a products OF literals.

$F(x,y,z) = x'y'+xy'z'+xz$

Advantage:-

It Can be implemented using "Two-level Circuit"

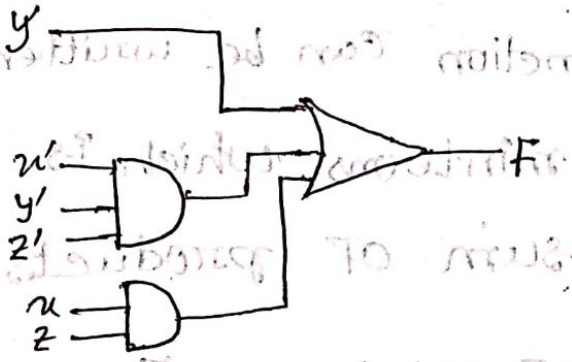
→ Literals and their Complement in the "0'th" level.

→ AND gates at the first level.

Literal:- A complemented or uncomplemented boolean variable.

a and $\bar{a}$ are distinct literals. $\bar{a}+cd$ is not

→ A single OR gate at the second level.



→ NOT gates are implicit

→ Literals are reused.

Minterm

⇒ It is a special product of literals, in which each input variable appears exactly once. (Special product মানে প্রতিটি Input exactly একবার আসে)

⇒ A Function with n variables has 2^n minterms.

⊛ A 3 variable function has $2^3 = 8$ minterms

0 0 0	0 0 1	0 1 0	0 1 1
$x'y'z'$	$x'y'z$	$x'yz'$	$x'yz$
m_0	m_1	m_2	m_3
1 0 0	1 0 1	1 1 0	1 1 1
$xy'z'$	$xy'z$	xyz'	xyz
m_4	m_5	m_6	m_7

000
001
010
011
100
101
110
111

Sum of minterms Form

* Every function can be written as a sum of minterms, which is a special kind of sum of products form.

* Sum of minterms forms for any function is unique.

*

$$F = x'y'z' + xy'z + x'yz' + xyz + xyz'$$
$$= m_0 + m_1 + m_2 + m_3 + m_6$$
$$= \Sigma m(0,1,2,3,6)$$

0 → 000
3 → 001
2 → 010
→ 011
110

$$F' = xyz' + xy'z + xyz$$
$$= m_4 + m_5 + m_7$$
$$= \Sigma m(4,5,7)$$

The Dual Idea: Products of Sums

- A product of sums (POS) expression Contains:-
  - → Only (AND) (product) operations at the "Outermost" level
  - → Each term must be a sum of literals

$$F(x, y, z) = y' \cdot (x'+y+z')(x+z)$$

  - → ~~OR~~ Literals & other Complements at the "0th level.
  - → OR gate at the First level.
  - → AND gate second level

Maxterms

A 'maxterm' is a sum of literals, in which each input variable appears exactly ~~once~~ once

→ 2^n → maxterm in a function

• $2^3 = 8$

$x'+y'+z'$	$x'+y'+z$	$x'+y+z'$	$x'+y+z$
(m_0)	(m_1)	(m_2)	(m_3)
$x+y'+z'$	$x+y'+z$	$x+y+z'$	$x+y+z$
(m_4)	(m_5)	(m_6)	(m_7)

Product of maxterms form

⇒ Every function can be written as a unique product of maxterms.

x	y	z	F(x, y, z)	F'(x, y, z)
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

অথবা, $F(x, y, z) = (x+y+z)(x'+y+z')(x'+y'+z')$

$= M_4 M_5 M_7$

$= \Pi(4, 5, 7)$

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

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

$= M_0 M_1 M_2 M_3 M_6$

$= \Pi M(0, 1, 2, 3, 6)$

Minterms & Maxterms are related

Any minterm m_i is the complement of the corresponding maxterm M_i.

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

$\Rightarrow$ $\boxed{m_4' = M_4}$ because $(xy'z')' = x'+y+z$

Converting between Standard form

We can convert a sum of minterm to a product of maxterm.

Before, $F = \Sigma m(0, 1, 2, 3, 6)$

$F' = \Sigma m(4, 5, 7)$

$= m_4 + m_5 + m_7$

Complementing $(F')' = (m_4 + m_5 + m_7)'$

$F = m_4' m_5' m_7'$ [De morgan]

$= M_4 M_5 M_7$

$= \Pi M(4, 5, 7)$

In general just replace the minterms with maxterms, using maxterm numbers that don't appear in the sum of minterms.

$F = \Sigma m(0, 1, 2, 3, 6) = \Pi M(4, 5, 7)$

K-Map:

2 variable:-

$2^2 = 4$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

$$F = x'y + xy' + xy$$

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

$$= y \cdot 1 + xy'$$

$$= (y + x)(y + y')$$

$$= x + y$$

x \ y	0	1
0	0	1
1	1	1

(*) Design a circuit which have 3 inputs Output will be high when maximum inputs will be high.

=> Here 3 variables are : x, y, z

x	y	z	F(x,y,z)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

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

$F(x, y, z) = (x+y+z)(x+y+z')(x+y'+z')$

⊛ 4 variables:-

$2^4 = 16$

w	x	y	z	F
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

wx \ yz	00	01	11	10
00	1 (0)	0 (1)	1 (3)	1 (2)
01	1 (4)	0 (5)	0 (7)	1 (6)
11	1 (12)	0 (13)	0 (15)	1 (14)
10	1 (8)	1 (9)	1 (11)	1 (10)

চারটা 0 কে প্রাধান্য দিলে লুপ করতে হবে

	1	1	1	0
	0	1	0	0

01/03/2022

Basic Operations:- AND, OR, NOT

Boolean Algebra helps us simplify expressions, which can also be directly translated into a hardware circuit.

Karnaugh Map:-

A graphical technique for simplifying an expression into a minimal sum of products (MSP)

Minterm:-

A product term in which all the variables appear exactly once.

Maxterms:-

A sum term in which all the variables appear exactly once. Either complemented or uncomplemented..

Review: Sum of Minterms

A Boolean Function can be represented algebrically

(4	2	1)		xyz			
x	y	z		F	F	F	F
0	0	0	→	0		1	0
0	0	1	→	0		0	1
0	1	0	→	0		1	0
0	1	1	→	0		0	1
1	0	0	→	0		0	1
1	0	1	→	0		1	0
1	1	0	→	0		0	1
1	1	1	→	1		1	0

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

$$= m_0 + m_2 + m_5 + m_7$$

$$F(x,y,z) = \Sigma m\,(0,2,5,7)$$

(Review: product of Maxterm)

4	2	1			
x	y	z	→ F	→ $\bar{F}$	
0	0	0	→ 1	→ 0	
0	0	1	→ 0	→ 1	
0	1	0	→ 1	→ 0	
0	1	1	→ 0	→ 1	
1	0	0	→ 0	→ 1	
1	0	1	→ 1	→ 0	
1	1	0	→ 0	→ 1	
1	1	1	→ 1	→ 0	

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

$= M_1 \cdot M_3 \cdot M_4 \cdot M_6$

$F(x,y,z) = \pi M(1,3,4,6)$

Important properties of Minterm

→ 2^n minterms for n Boolean variables. These minterms can be evaluated from the binary numbers from 0 to $2^n - 1$

→ $F(x, y, z) = \Sigma m(0, 2, 5, 7)$

$F'(x, y, z) = \Sigma m(1, 3, 4, 6)$

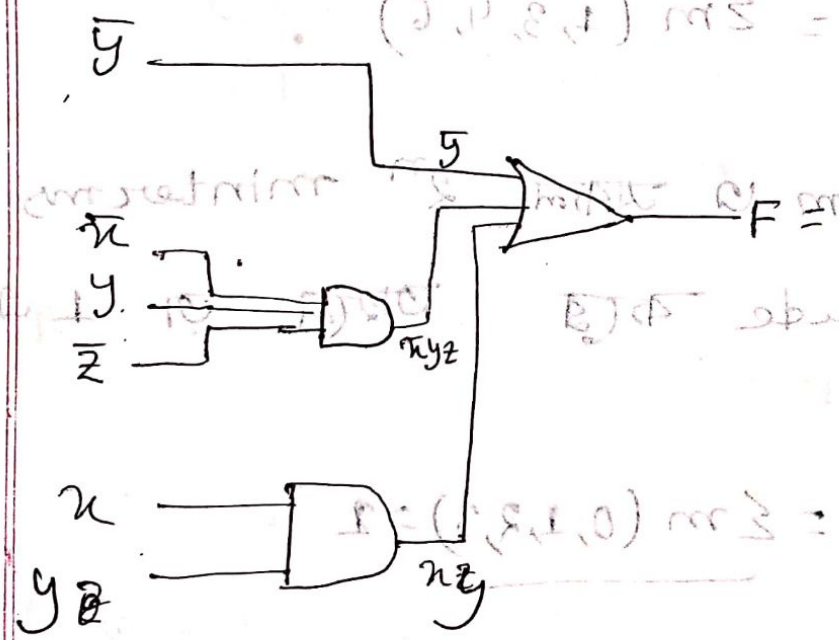
⇒ যে minterm এ সকল 2^n minterms কে যদি include করে তাহলে তা 1 এর সমান।

Or $(x, y) = \Sigma m(0, 1, 2, 3) = 1$

~~(*) When we~~

Sum of products :-

When we simplify a function in SOM form by reducing the number of literals in the terms, the simplified expression is said to be in Sum-of-products form



Sum of Minterm ; $F = \Sigma m(0,1,2,3,4,5,7)$

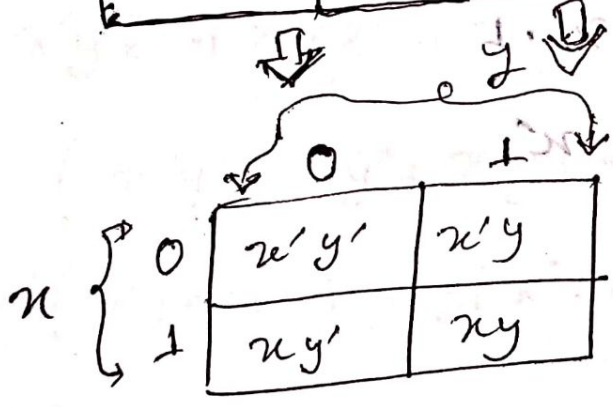
Sum of products, $F = y' + x'yz + xz'$

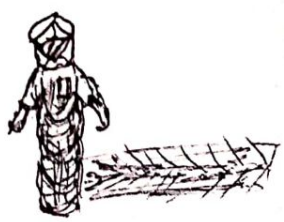
Re-arranging The Truth Table

* A variable function has four possible minterms. We can re-arrange these minterms into a Karnaugh map.

0 = x′ / y′	
1 = x / y	

x	y	minterm
0	0	x′ y′
0	1	x′ y
1	0	x y′
1	1	x y





K-Map Simplifications

⇒ Two-variable sum of minterms:

$x'y' + x'y$

⇒

		y
	$x'y'$	$x'y$
x	xy'	xy

⇒ What happens if you simplify this expression using Boolean algebra?

$$x'y + x'y = x'(y' + y) \quad \text{[Distributive]}$$

$$= x' \cdot 1$$

$$= x'$$

3 variables Karnaugh Map

Inputs = x, y, z

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

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

$\Rightarrow x'y'z + x'yz$

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

$= x'z \cdot 1$

$= x'z$

$\Rightarrow x'y'z' + xy'z' + x'yz' + xyz'$

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

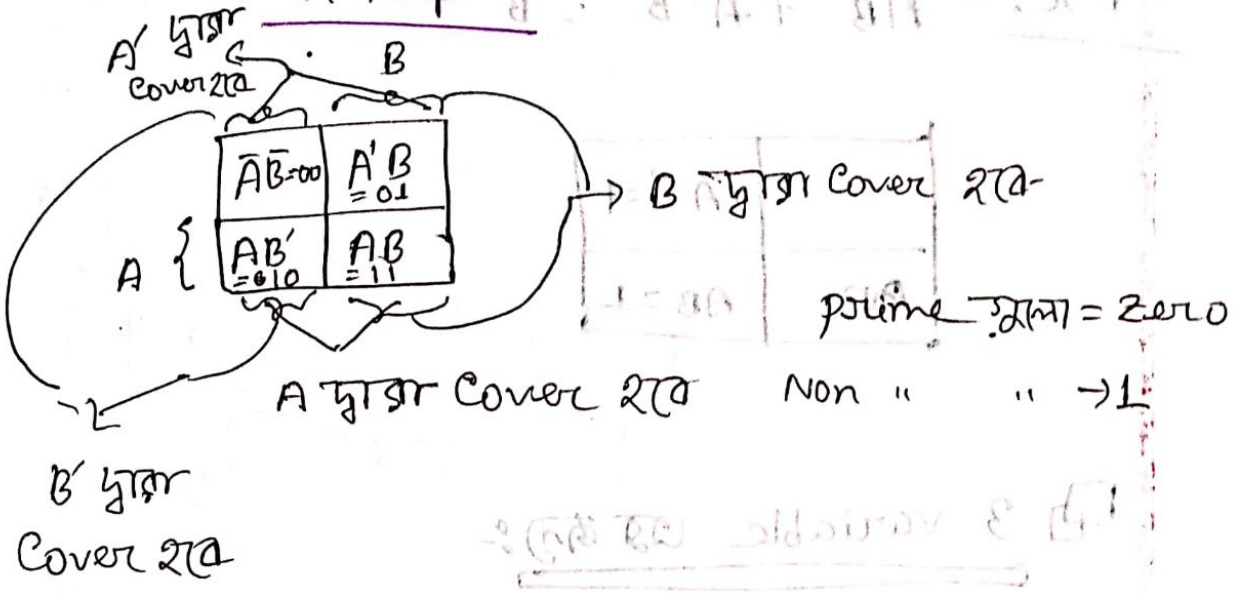
$= z'(y'(x' + x) + y(x' + x))$

$= z'(y' \cdot 1 + y \cdot 1)$

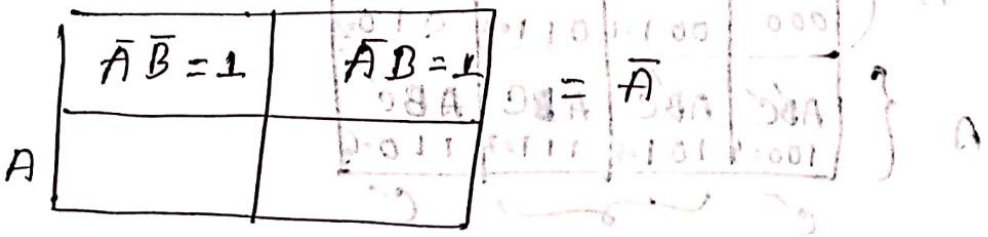
$= z'(y' + y) = z' \cdot 1 = z'$

Karanaugh Map

K-Map



=> Equation:- $\bar{A}\bar{B} + \bar{A}B$



এখানে, just $\bar{A}$ টা লেখলেই উভয় Box Cover হয়ে যাবে। তাহলে $\bar{A}$ হলো Answer এবং K-map হতে সহজেই পাওয়া যায়।

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

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

Ex:- $AB + \bar{A}B = B$

	$\bar{A}B = 1$
~~AB~~	$AB = 1$

3 variable এর জন্য:-

	B′C′	B′C	BC	BC′
A′	$\bar{A}\bar{B}\bar{C}$ 000	$\bar{A}B\bar{C}$ 001=1	$\bar{A}BC$ 011=3	$\bar{A}BC'$ 010=2
A	$A\bar{B}\bar{C}$ 100=4	$A\bar{B}C$ 101=5	ABC 111=7	$AB\bar{C}$ 110=6
	C′	C		C′

Equation:- $A\bar{B}\bar{C} + A\bar{B}C + ABC + \bar{A}C$ → $\bar{A}$ ও C যোগফল এই দুইটা বক্স

=>

	B′C′	B′C	BC	BC′
A′	$\bar{A}\bar{B}\bar{C}$ 000	① $\bar{A}\bar{B}C$ 001	① $\bar{A}BC$ 011	$\bar{A}BC'$ 010
A	$A\bar{B}\bar{C}$ 100	① $A\bar{B}C$ 101	① ABC 111	$AB\bar{C}$ 110
	C′	C		C′

= C

Final

Half Adder:-

Half Adder circuit needs two binary input & two binary outputs.

If we arbitrarily assign symbols x & y to the two inputs and S (for sum) and C (for Carry) to the outputs.

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Adder:-

The most basic arithmetic operation, no doubt, is the addition of ~~binary~~ two binary digits. This simple addition consists of four possible elementary operations,

namely,

0+0 = 0,

0+1 = 1

1+0 = 1

1+1 = 10

$S = x'y + xy'$

$C = xy$

Here all three summations are having one digit. But 1+1 summation consists of two digits.

The higher significant bit of this result is called a carry.

The carry obtained from the addition of two bits is added to the next higher-order pair of significant bits.

A combinational circuit of two bits is called half adder.

And three bits (two significant bits and a previous carry) is a full adder.

Two half-adders can be employed to implement a full adder.

Circuit Design

1. 6-steps 2-

(1) problem state

(2) Inputs Number of available input variables and required output variables is determined

(3) Assign variable name.

(4) Truth Table

(5) Simplified Boolean function

(6) Drawing of Logic circuit.

Full Adder

A Full adder circuit is a combinational circuit that forms the arithmetic sum of three input bits.
It consists of three inputs and two outputs.

X-OR Gate

Exceptional OR gate

~~Odd~~ Input — Output

variable ৭, odd number 1 থাকলে → 1

even number 1 থাকলে → 0

$C = xy$

$S = x'y + xy'$

$= x \oplus y$ (x-OR gate)

IC ⊕ circuit = 7486 IC

~~ABC + AC + AB~~

BC / A	00	01	11	10
A'0	0	0	1	0
A1	0	1	1	1

~~ABC + ABC~~ +

Full Adder → 3 variable

Truth Table এবং Function নির্ণয়:

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

$S = A'B'C + A'BC' + AB'C' + ABC$

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

$= A'(B \oplus C) + A(B \odot C)$

$= A'(B \oplus C) + A(B \oplus C)'$

$= A'x + A \cdot x'$ $\leq$ $A \oplus x$

Let $[B \oplus C = x]$ $= A \oplus B \oplus C$ (Ans)

Subtractor

The subtraction of two binary numbers may be accomplished by taking the complement of the subtrahend & adding it to the minuend.

Half- Subtractor:-

A half- subtractor is a combinational circuit that subtracts two bits and produces their difference.

$x, y \rightarrow$ Two bit

If $x > y \rightarrow x=0, y=0 \rightarrow 0-0 \rightarrow 0$

$\rightarrow x=1, y=0 \rightarrow 1-0 \rightarrow 1$

$\rightarrow x=1, y=1 \rightarrow 1-1 \Rightarrow 0$

If $x < y$,

$x=0, y=0 \rightarrow 0-0 \rightarrow 0$

$x=0, y=1 \rightarrow 0-1 \rightarrow$

(It is necessary to borrow a 1 from the next

higher stage adds 2 to the minuend bit.

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

0-1

Adding 1 from next higher stage । फलतः

1-0 or decimal value 2 या 2।

So, A=2, A-B=2-1=1

B=1, D=1

Function:-

$D = x'y + xy'$

$B = x'y$

Full Subtractor:-

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

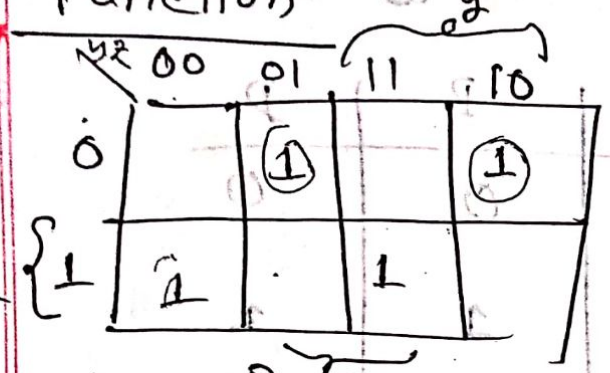
$x - y - z$

$0 - 0 - 1 = -1$ সেটা হবে না 1 so, next stage হতে 2 borrow করে নিয়ে আসতে হবে। $2 - 0 - 1 = 1$ 1 carry = 1 → B →

$0 - 1 - 0 = -1$ (Same as before)

So, carry = 1 এবং B = 1

Function



C, → D

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

$x'yz$ $x'yz'$ $xy'z'$ xyz

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

$= x'y(1) + x(y \oplus z)'$

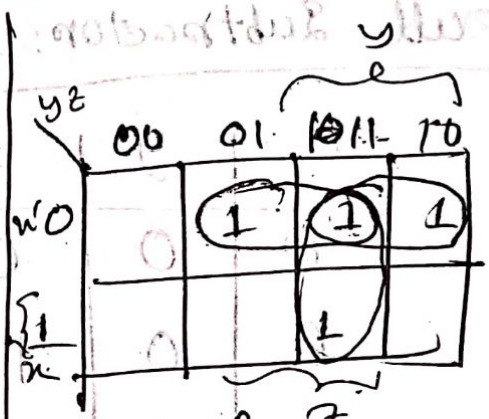
$= x'y + x(y \oplus z)'$

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

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

$= x'y + xy'$

$= x \oplus y$



→ B

$B = x'y + xy'$

$D = x'yz + x'y'z + x'yz' + xyz$

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

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

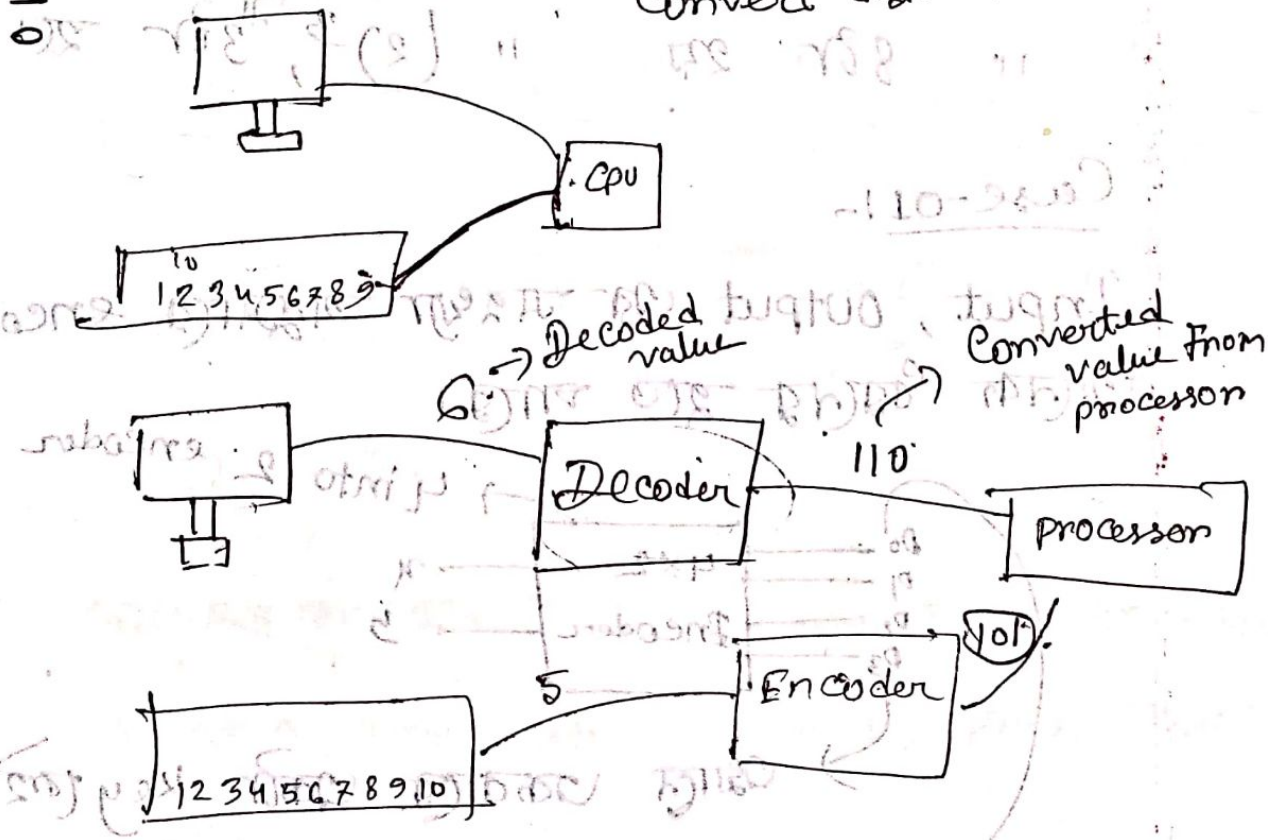
$B = x'z + x'y + xyz$

$= x'y + xy'$

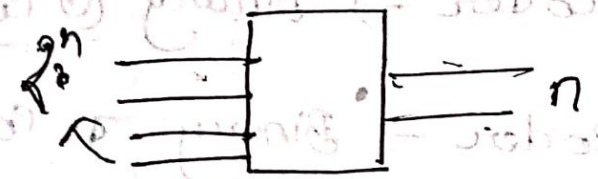
Encoder :-

Encoder → Binary তে Convert করবে

Decoder → Binary কে Computer এর Language তে মানুষের বোঝার Convert করবে।



Encoder এর input 2^n হলে output হবে n সংখ্যক

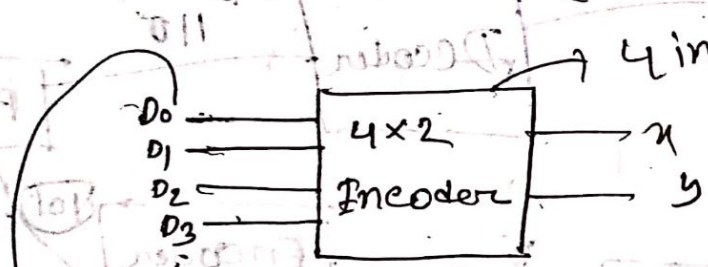


যদি 4টা হয় তবে (2) 2টা হবে output

" 8টা হয় " (2) 3, 3টা হবে output

Case-01:-

Input, output এর সংখ্যা অনুসারে encoder অনেক প্রকারের হতে পারে।



এখানে একবারে একটা key কেই press করা যাবে। এবং এই switch এর decimal value কে binary Convert করে।

2×1
4×2
8×3
16×4

Input

Output key এর decimal binary value আসবে

D_0	D_1	D_2	D_3	x	y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

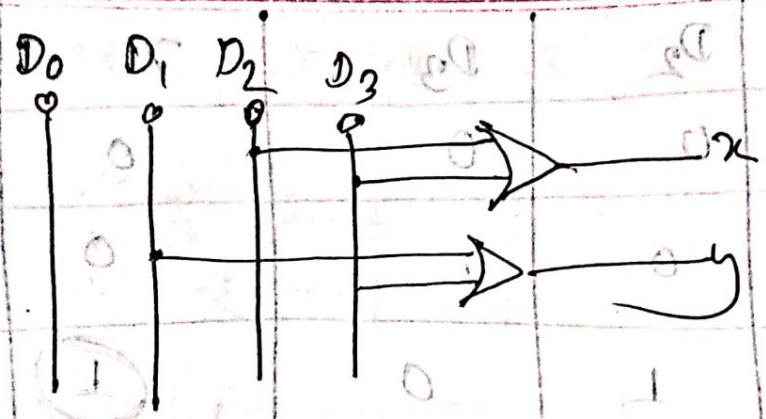
Output

So, $x = \bar{D_0}\,\bar{D_1}\,D_2\,\bar{D_3} + D_0\,\bar{D_1}\,\bar{D_2}\,D_3$

$= D_2 + D_3$

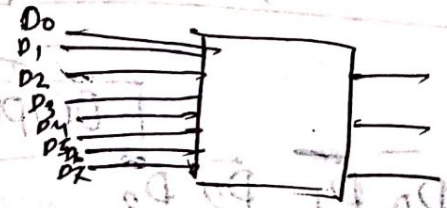
(বাল্বগুলোকে বাম দেওয়া হয়েছে কারণ encoder এ at a time একটা switch press করা যায়)

So, $y = D_1 + D_3$



8 x 3 encoder :-

$2^{(3)} = 8$ input, output = n

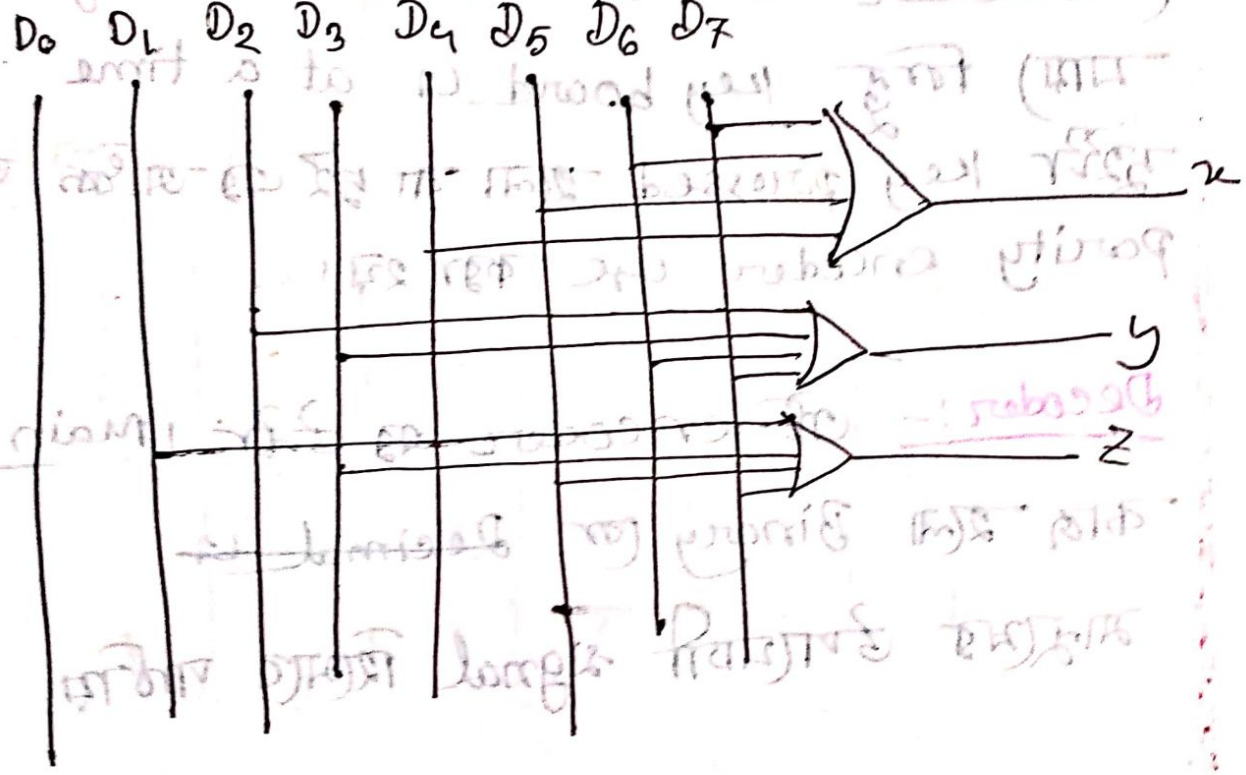


Input								Output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$x = D_4 + D_5 + D_6 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_7$



এখানে Input ৪টি, Input Octal এ 0 to 7 total 8 টি number। আর একটু এটাকে Octal number.

Decoder

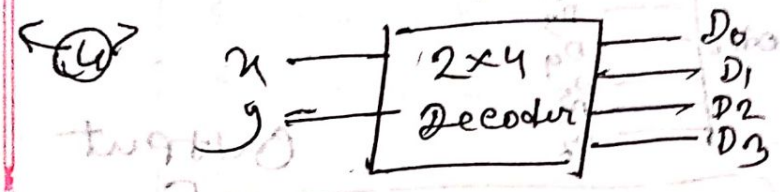
Parity encoder :-

মনে করা যাক কোনো key-board এ, (encoder - এ At a time একটা signal যায়) কিন্তু key board এ at a time দুইটা key pressed হলো বা দুই এর অধিক তখন Parity encoder use করা হয়।

Decoder :- এটি encoder এর উল্টা। Main কাজ হলো Binary কে ~~Decimal এ~~ মানুষের উপযোগী signal হিসেবে পাওয়া

Decoder → Input n বহুপদক হলে

Output → 2^n টি।



Truth Table:-

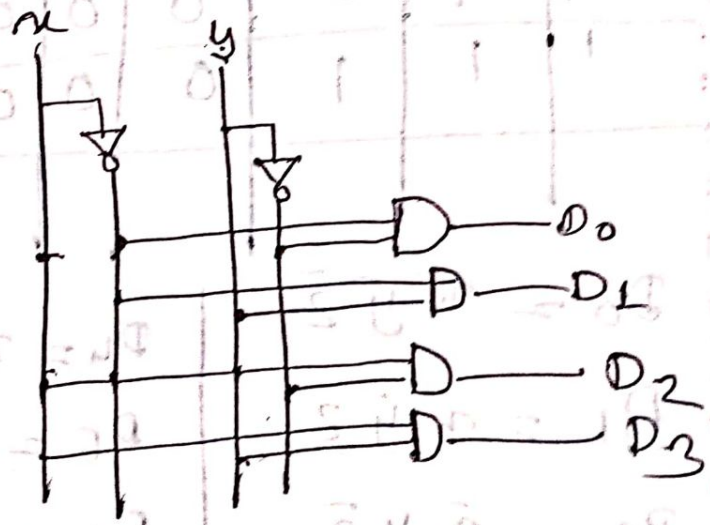
x	y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$D_0 = \bar{x} \cdot \bar{y}$

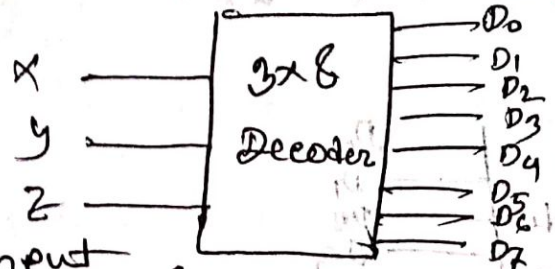
$D_1 = \bar{x} \cdot y$

$D_2 = x \cdot \bar{y}$

$D_3 = xy$



3×8 Decoder:-



Input | Output

x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$D_0 = \bar{x}\bar{y}\bar{z}$

$D_1 = \bar{x}\bar{y}z$

$D_2 = \bar{x}y\bar{z}$

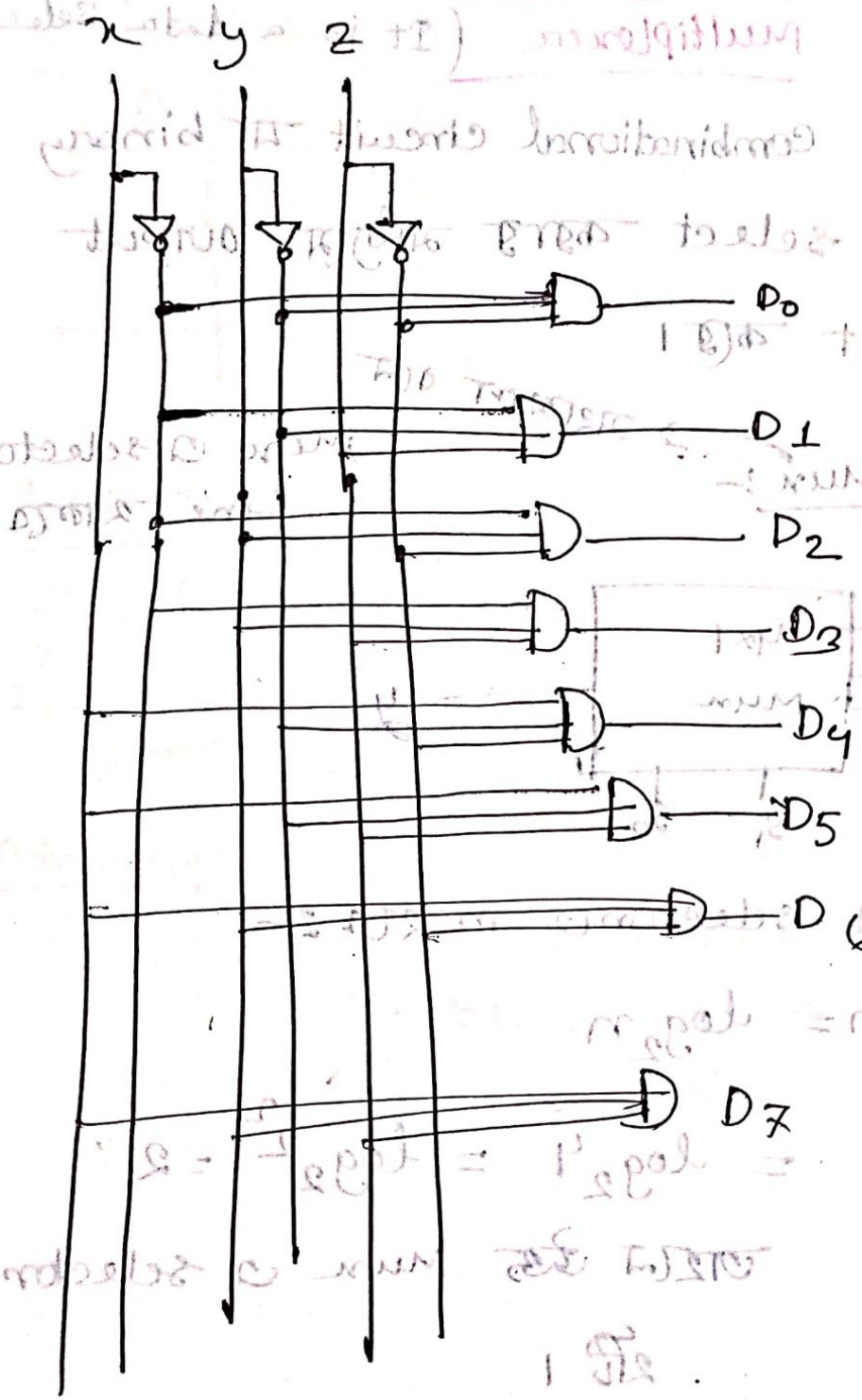
$D_3 = \bar{x}yz$

$D_4 = x\bar{y}\bar{z}$

$D_5 = x\bar{y}z$

$D_6 = xy\bar{z}$

$D_7 = xyz$

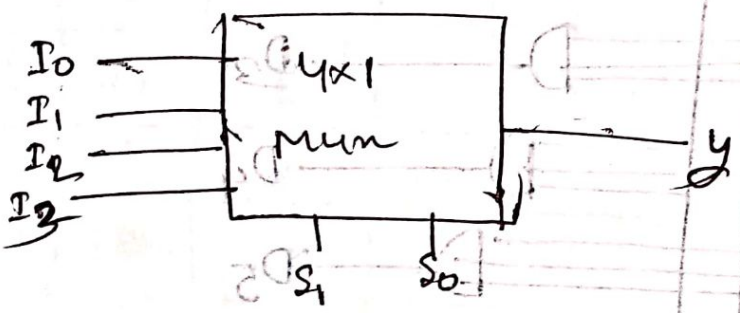

x
y
z
D0
D1
D2
D3
D4
D5
D6
D7

Multiplexer (It is a data selector)

It is a combinational circuit যা binary info কে select করার মাধ্যমে output represent করে।

4×1 Mux :- → সহজভাবে বলে

Mux এ selector line থাকে



Input যত selector তত n হবে 2-

$$n = \log_2 n$$

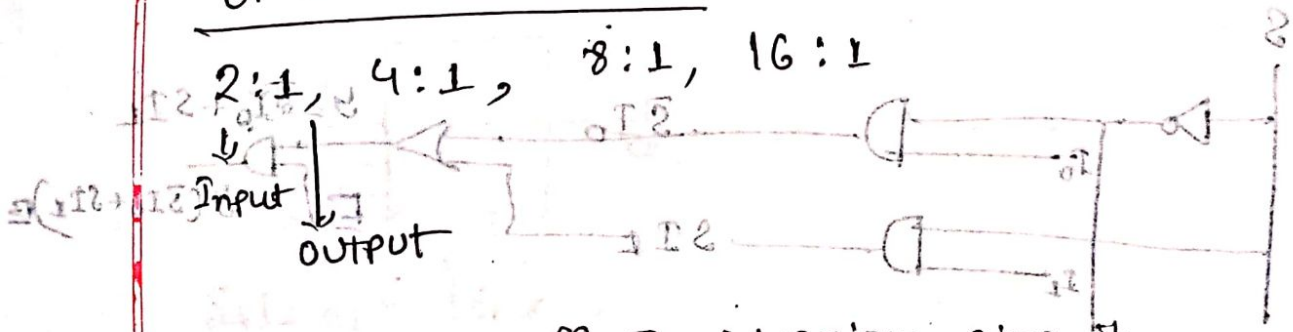
$$= \log_2 4 = \log_2 2^2 = 2$$

তাহলে উক্ত Mux এ selector line ২টি।

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

Types of Mux:-

2:1, 4:1, 8:1, 16:1



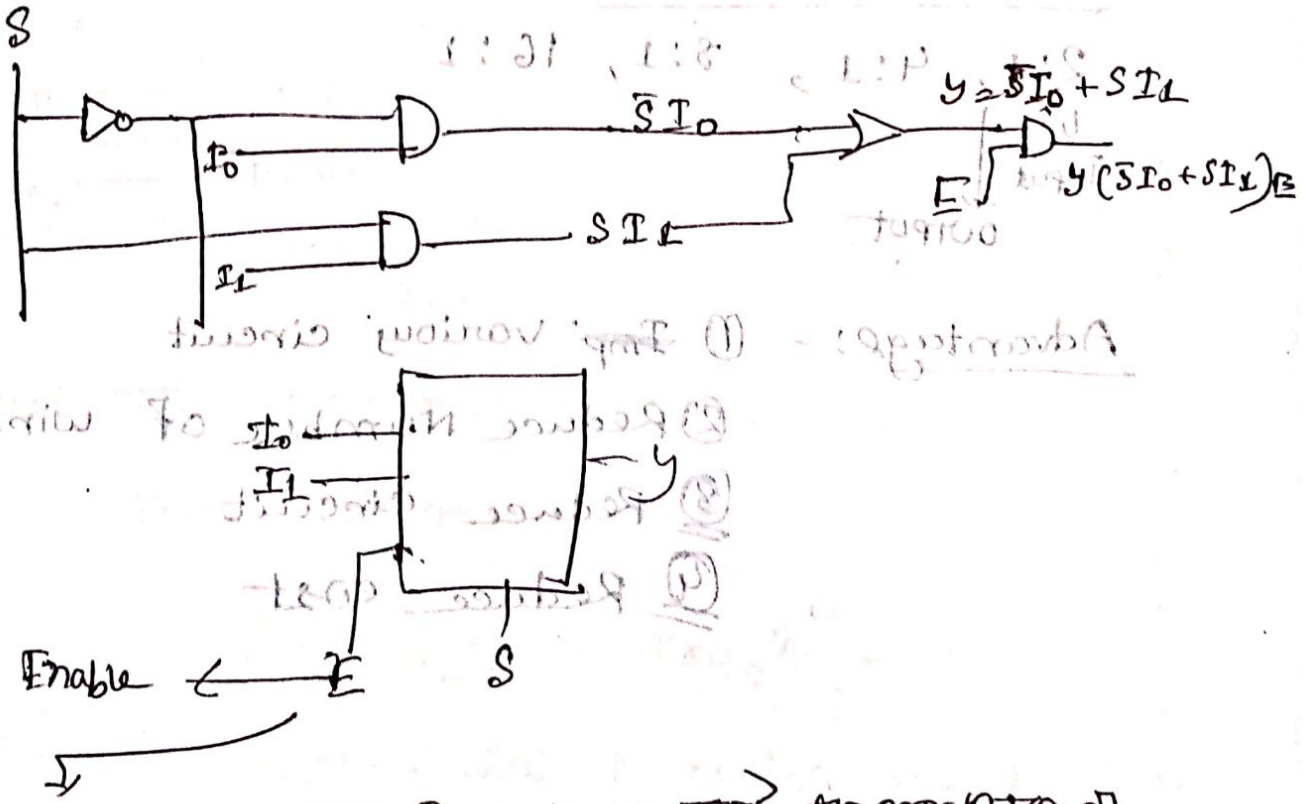
Advantage:-

1. ~~Imp~~ various circuit
2. Reduce Number of wires
3. Reduce circuit
4. Reduce cost

2:1 Mux:- $n=2$, $m=1$

S	y
0	I_0
1	I_1

$$y = \bar{S} I_0 + S I_1$$



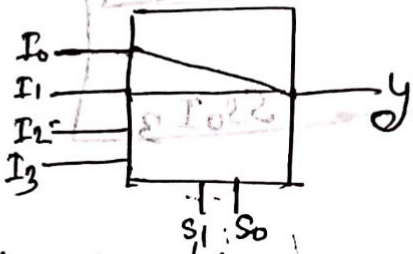
এর value 0 হলে Input এ যাই দেওয়া হোক না কেন value 0 হবে।

$E = 1$ হলে Mux এর Output show করবে

E	y
0	0
1	y

4×1 Mux:-

Input 4 bit and output 1 bit



Select line:-

$n = 4 \quad s = ?$

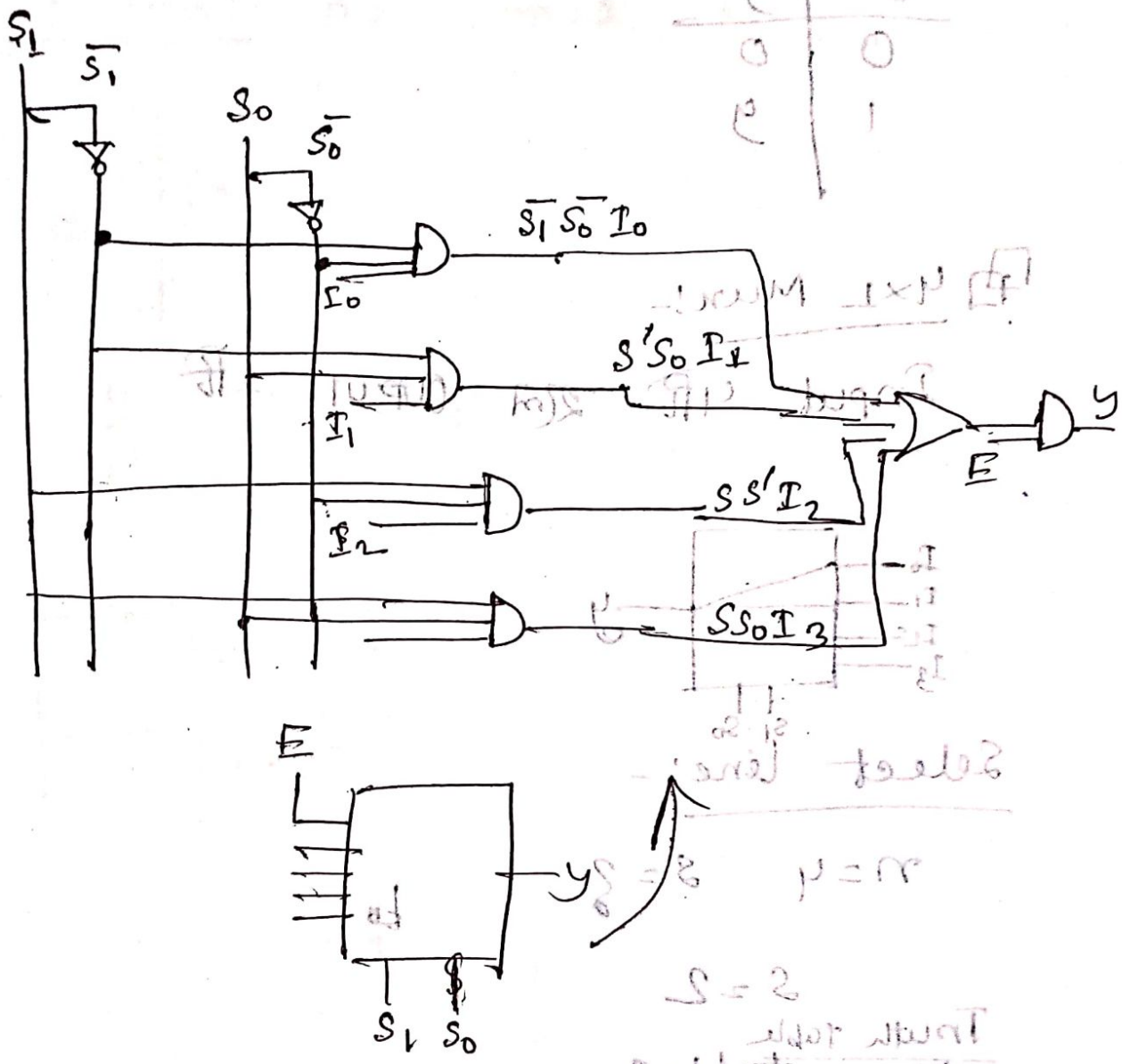
$2^s = 4$

$s = 2$

Truth table

S	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

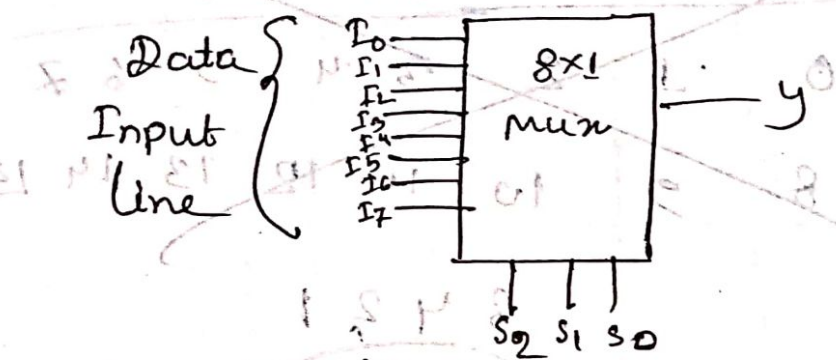
$y = \bar{S}\,\bar{S_0} \cdot I_0 + S' \cdot S_0 \cdot I_1 + S \cdot S_0' \cdot I_2 + S S_0 I_3$



E = 0 হলে সকল value 0

E = 1 হলে value আসবে

8×1 Multiplexer :-



Ex $\log_2 2^n = \log_2 2^3 = 3$

Truth table

S_2	S_1	S_0	y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Minterm Function

$$= S_2' S_1' S_0 I_0 + S_2' S_1 S_0' I_1 + S_2' S_1 S_0' I_2 + S_2' S_1 S_0 I_3 + S_2 S_1' S_0' I_4 + I_5 S_2 S_1' S_0 + S_2 S_1 S_0' I_6 + S_2 S_1 S_0 I_7$$

	I_0	I_1	I_2	I_3
S_2'	0	1	2	3
S_2	4	5	6	7

Mux table

Input থেকে select করতে হলে Multiplexer আর output এ কোনটির connection এ যাবে তাকে demultiplexer বলা হয়।

মূল কথা selected sender থেকে selected demultiplexer receiver এ info পাঠানোতে Multiplexer use করা হয়।

একবারে একটি input-কে select করা যাবে।

Example:-

(a) $\bar{P}Q + Q\bar{R} + P\bar{Q}R$

(4×1) mux

$\log_2 2^2 = 2$ bit S

$y = \bar{S}_1\bar{S}_0 I_0 + S_1'S_0 I_1 + S_1 S_0' I_2 + S_1 S_0 I_3$

$= \bar{P}\bar{Q} I_0 + P'Q I_1 + P\bar{Q} I_2 + PQ I_3$

$S_1 = P$, $S_0 = Q$

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

De Multiplexer

Input 1 টি, output অনেক।

Multi Multiplexer হলে বিপরীত output demultiplexer

Input

1:4 Demux:-

$n = 4$

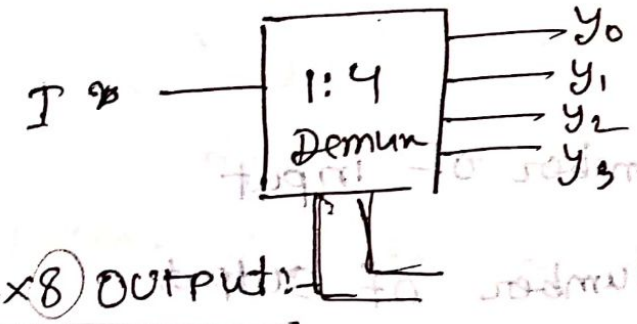
$m = \log_2 4$

$= 4$

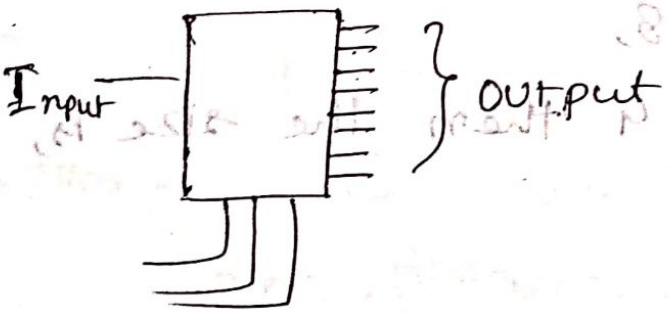
$n = 4 = 2^2$

$2^2 = 2^3$

$\log_2$



1×8 output:-



1 টি Input 8টি output.

ROM: (Read ~~only~~ Only Memory)

ROM is non volatile memory, so when power is OFF, so digital data of memory will stay stored in it. (Means the content of memory will stay stored)

ROM Structure:-

→ Decoder + programmable OR gates

→ AND gates (Fixed) + programmable OR gates

Size:-

$= 2^x * x\,y$

where x = Number of input

y = Number of output

⇒ Ex: If,

Total Input = 8,

Total output = 4 then the size is,

$x = 8$

$y = 4$

Size of ROM $= 2^8 \times 4 = 256 \times 4 = 1024$ bits

$= (1024/8)$ bytes $= 128$ bytes

Classification of ROM:-

PROM → programmable ROM

EPROM → Erasable programmable ROM (UV ray)

EEPROM → Electrically Erasable programmable ROM

Flash Memory →

Mask ROM →

PROM:- Empty when it is made. we can store data only once. And can't change or delete datas.

EPROM - we can erase datas by uv rays

→ After that we can reprogram this memory.

EEProm → we can erase data electrically many times. Efficiency of memory will delay with respect erase.

Erase happens bit by bit

Flash Memory:-

→ Speed of erase is faster that than EEPROM.

→ Data erase is done block by block.

Mask ROM:-

→ It is PROM only.

→ Is programmed by chip maintanence

→ This ROM is masked off ~~chasing~~ during photograph.

PLD (Programmable Logic Devices)

(1) ROM (Read Only Memory) :-

PLD Structure

First Stage → 2nd Stage

AND → OR

(1) ROM :-

AND → OR

Fixed → programmable

(2) PLA :-

AND → OR

Program → Fixed Program

(3) PAL :-

AND → OR

Prog → Fixed

ROM :-

AND :: Fixed → Means we are using a decoder

OR :: Programmable

(*) Design a combinational circuit using ROM. Accept 3-bit number & outputs a binary number equal to the square of the input number.

	A	B	C	B_6 (32)	B_5 (16)	B_4 (8)	B_3 (4)	B_2 (2)	B_1 (1)
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1
2	0	1	0	0	0	0	1	0	0
3	0	1	1	0	0	1	0	0	1
4	1	0	0	0	1	0	0	0	0
5	1	0	1	0	1	1	0	0	1
6	1	1	0	1	0	0	1	0	0
7	1	1	1	1	1	0	0	0	1

B_5 B_4 B_3 B_2 B_1 B_0

32 16 8 4 2 1

procedure

Square of maximum value of 3 bits are

(7)^2 = 49

Here max value is = 49

2^1 / 2^2 / 2^3 / 2^4 / 2^5 non of them are greater or equal to 49

But 2^6 is. so we have taken 6 bits here

=> Then we're doing square to decimal number upto seven as it is written in question.

Then we are just inserting binary value on the table.

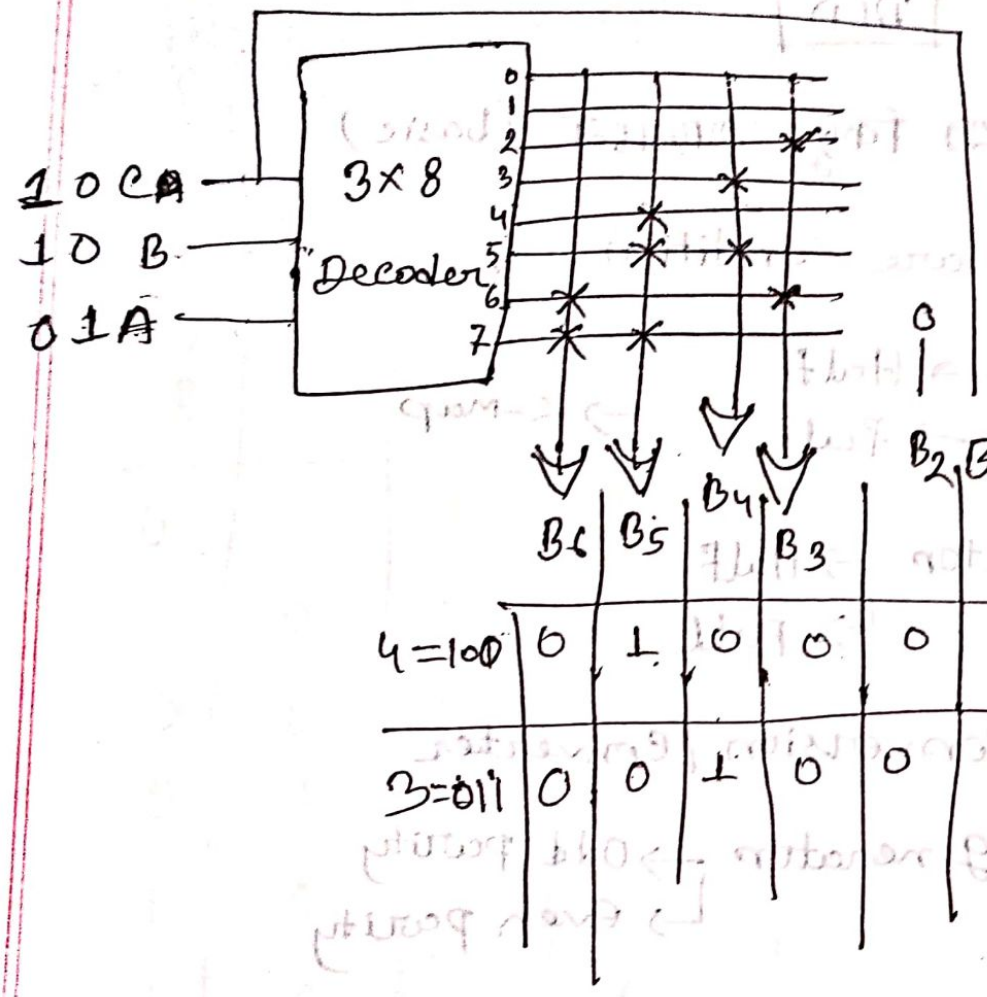
=> After doing we need to decrease the number of OR gates, As B_2 all values are equal to zero.

Also B_1 is equal to C so we can directly take input from C.

=) After ~~dwa~~ drawing the circuit we will have find out that in where OR gates ~~will~~ have '1' from truth table.

=) After this we have to check the truth table. Suppose I am ~~take~~ taking decimal number 4 as input or 3 as input we will have to check it. Like if the decimal value is 3 The input binary is 1 there in output, in 3 where there are connections will have 1 as output.

Circuit Drawing



4 = 100

	B_6	B_5	B_4	B_3	B_2	B_1
4=100	0	1	0	0	0	0
3=011	0	0	1	0	0	1

Ex-01

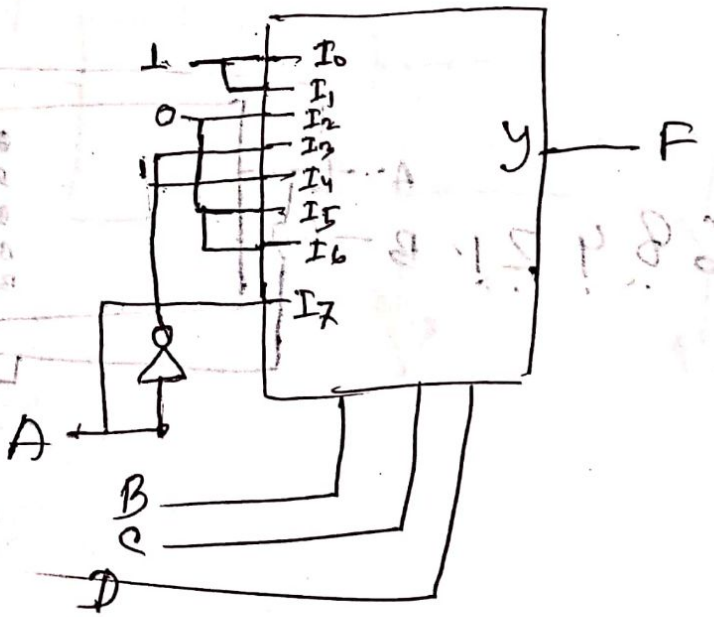
$F(A,B,C,D) = \sum(0, 1, 3, 4, 8, 9, 15)$

Minterm	A	B	C	D	F
0	0	0	0	0	I_0
1	0	0	0	1	I_1
2	0	0	1	0	I_2
3	0	0	1	1	I_3
4	0	1	0	0	I_4
5	0	1	0	1	I_5
6	0	1	1	0	I_6
7	0	1	1	1	I_7
8	1	0	0	0	I_8
9	1	0	0	1	I_9
10	1	0	1	0	I_{10}
11	1	0	1	1	I_{11}
12	1	1	0	0	I_{12}
13	1	1	0	1	I_{13}
14	1	1	1	0	I_{14}
15	1	1	1	1	I_{15}

Mux Table :-

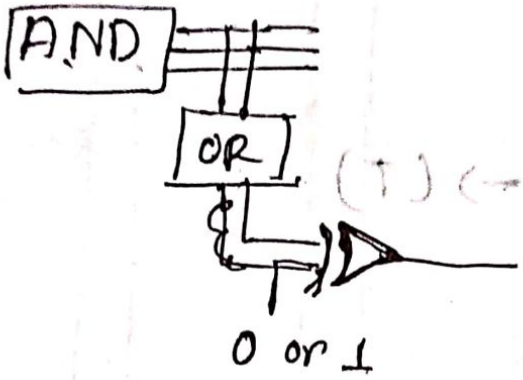
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	A'	A'	0	0	A

✱ Mux table এ যাকে Input দিবো তার value Main table হতে বসাবো। সেখানে 0 হলে prime সহ 1 হলে prime ছাড়া



Programmable Logic Array (PLA)

AND OR



Ex-01

Implement the following booleans function with PLA $F_1(A,B,C) = \Sigma(0,1,2,4)$

$F_2(A,B,C) = \Sigma(0,5,6,7)$

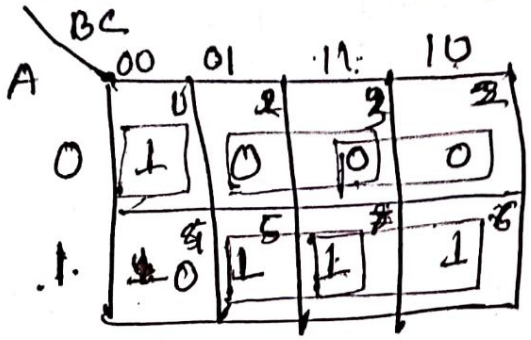
=>

A \ BC	00	01	11	10
A' 0	1	1	0	1
A 1	1	0	0	0

$F_1 = (A'B' + A'C' + B'C')$ → True value

Compliment, (C) $F_1 = (AC + AB + BC)$

For f_2, 0, 5, 6, 7.

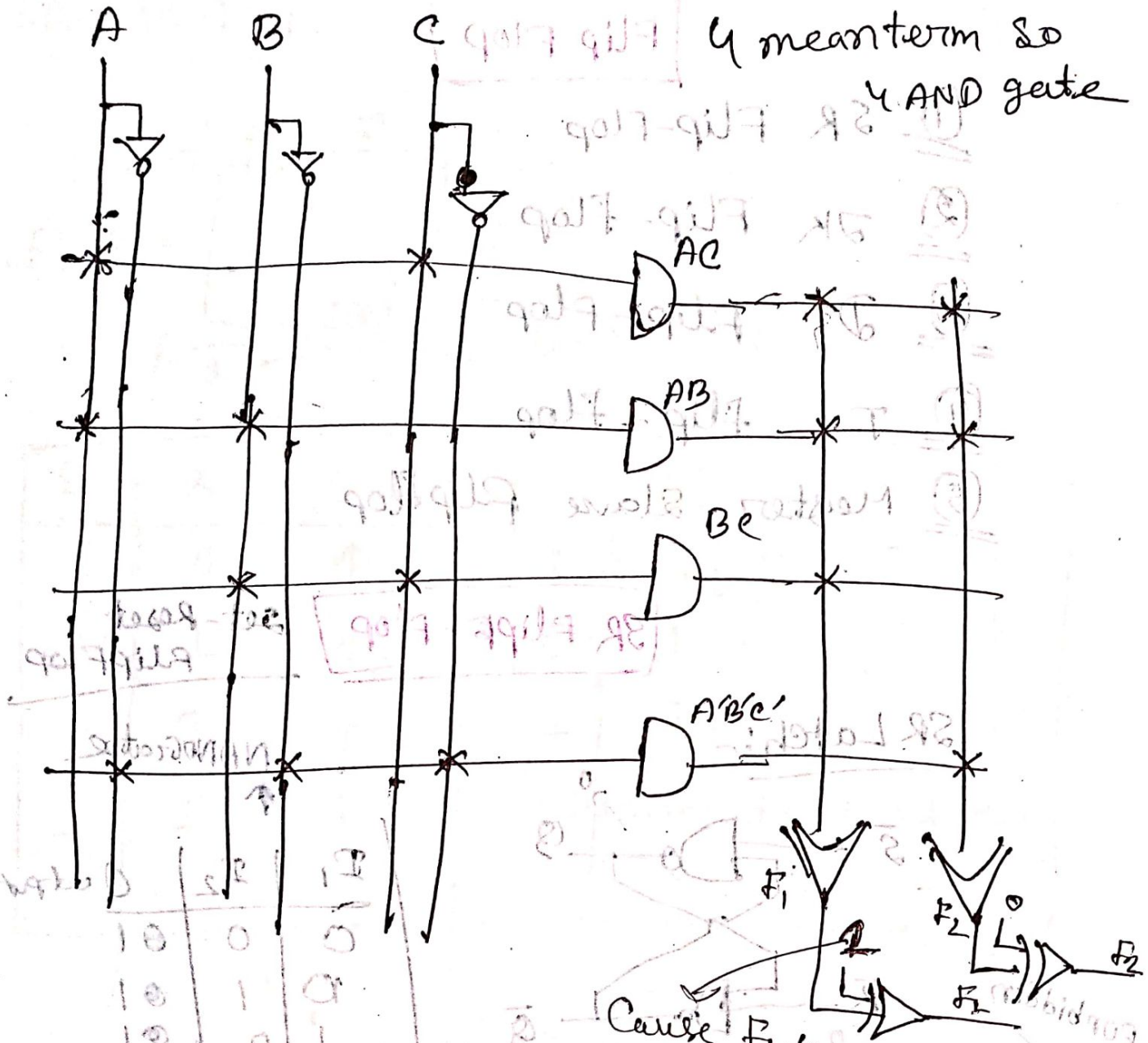


$\therefore f_2[T] = (A'B'C' + AC + AB) \rightarrow (T)$

$F_2(C) = (A'C + A'B + A\bar{B}\bar{C})$

we r using $F_1(C)$ & $F_2(T)$

Minterms	variables A	B	C	outputs F_1 (C)	F_2 (T)
AC	1	–	1	1	1
AB	1	1	–	1	1
BC	–	1	1	1	1
A'B'C'	0	0	0	1	1



X-OR Gate works as an Controlled inverter

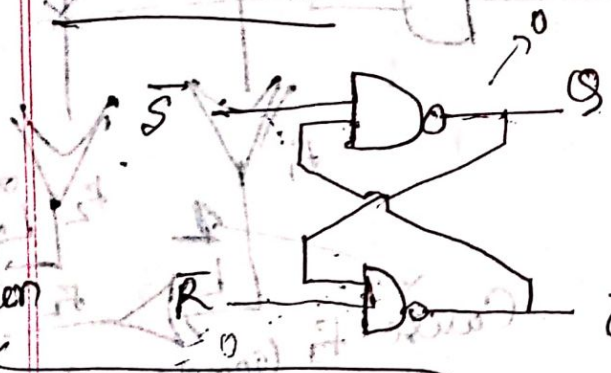
Flip Flop

1. SR Flip-Flop
2. JK Flip-Flop
3. D Flip-Flop
4. T Flip-Flop
5. Master-Slave Flip Flop

SR Flipe-Flop

set-Reset Flip Flop

SR Latch:-



NAND Gate

I_1	I_2	Output
0	0	1
0	1	1
1	0	1
1	1	0

Forbidden
इसमें Q और Q' का value same

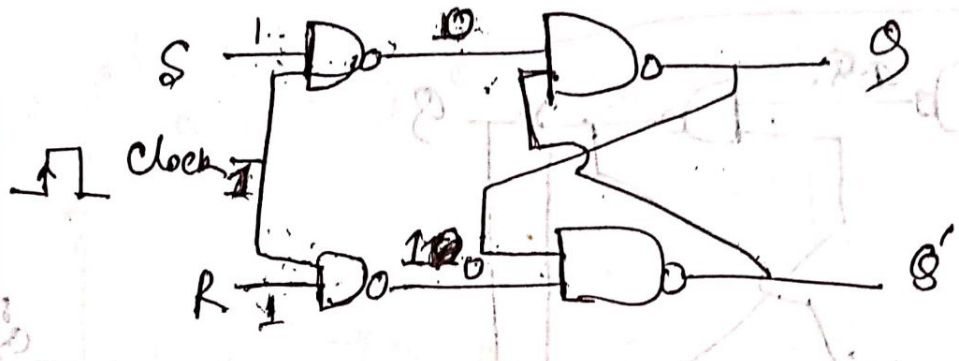
S	R	Q	Q'
0	0	(Forbidden)	
0	1	1	0
1	0	1	0
1	0	0	1
1	1	0	1 (unchanged)

दोनों 1 होने पर output zero

NOT + AND

AND $1 \cdot 1 = 1$ NOT $\bar{0}$

SR Flip-Flop:-



S	R	Clock	Q	Q'
0	0	↑	(unchanged)	
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	Indeterminate / Forbidden	

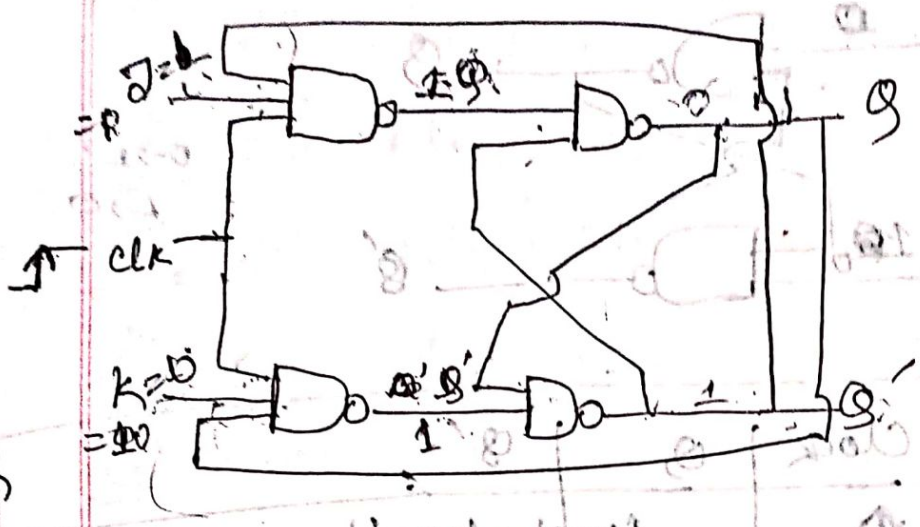
Set মানে output এর value = 1 ;

Reset " " " = 0 অর্থাৎ

Reset = 1 output = 0

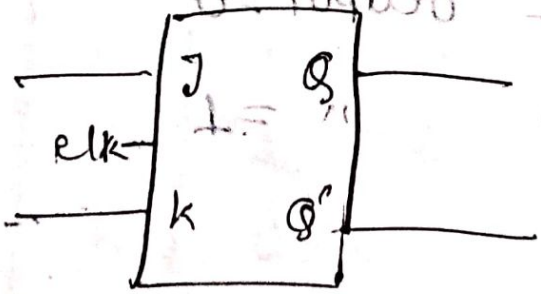
Set = 0

J-K Flip Flop

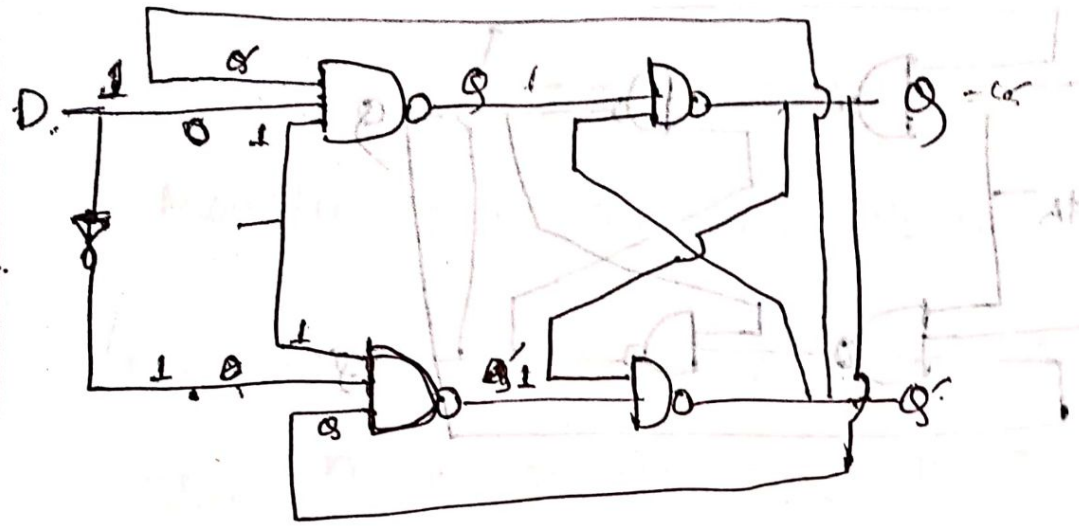


J	K	clk	Q	Q'
0	0	↑	(unchanged)	
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	Q (toggle)	Q' (toggle)

এই অবস্থাকে toggle বলে



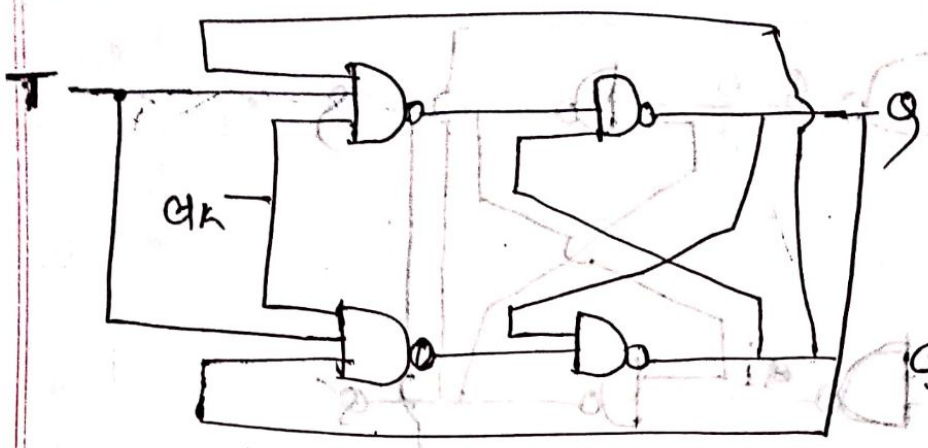
D - Flip-flop :- (डेटा फ्लिप फ्लॉप) (Data FlipFlop)



J	K	clk	Q	Q'
1	0	1	1	0
0	1	1	0	1

D	clk	Q	Q'
0	↑	0	1
1	↑	1	0

T-FlipFlop:-



T	clk	Q	Q'
0	↑	অপরিবর্তিত	
1	↑	Toggle	

J	K	clk	Q	Q'
0	0	↑	অপরিবর্তিত	
1	1	↑	Q'	Q

Counter

→ Counter যতগুলো সংখ্যা গুনতে পারে তাকে তার Modulus / Mod number বলে।

n → সংখ্যক Flip-Flop use করা হয়

তবে 2^n " সংখ্যা গুনতে পারে।

Asynchronous Counter / Ripple counter

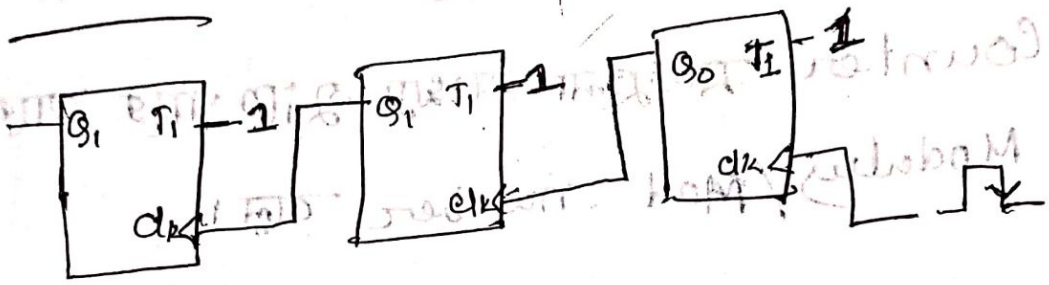
→ Ripple up Counter:- 0, 1, 2, 3, ...

Ascending Order Count

→ Ripple Down Counter:-

7, 6, 5, 4, 3, 2, ... Descending order

Ripple up :- (Counter)



Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Negative transition হলে value change হবে।

Negative transition → Toggle করবে

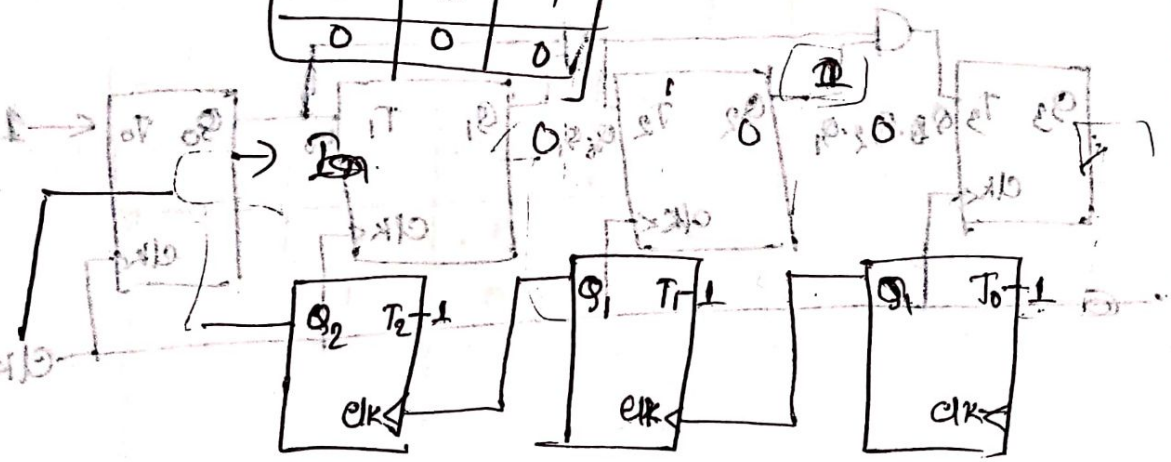
FlipFlop Active ← Toggle করবে FlipFlop Active

Active হলে Q এর মান পরিবর্তন

Ripple Down Counter :-

(সবগুলো Flip Flop positive transition এ সময় Active হবে)

Q_2	Q_1	Q_0
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

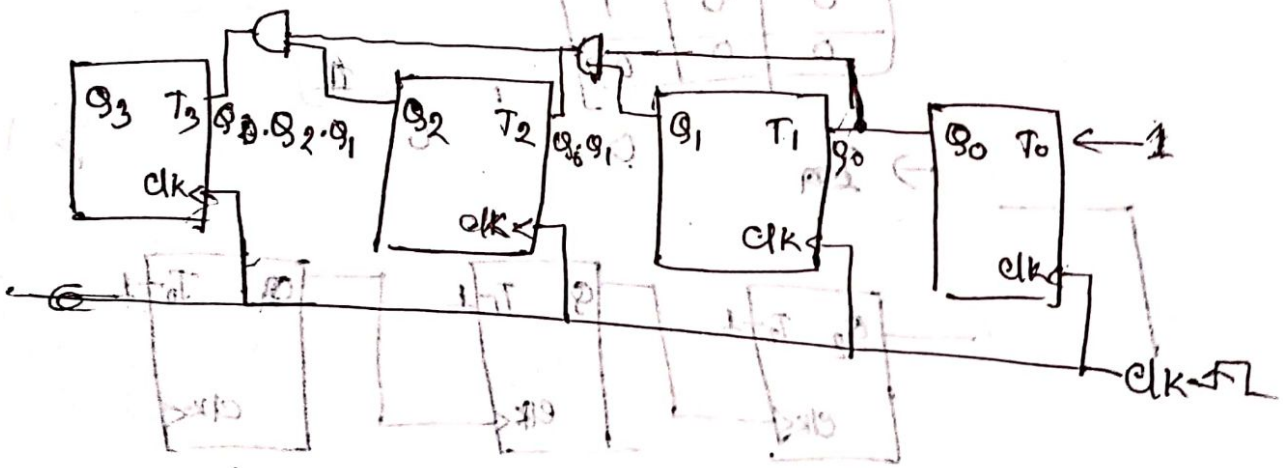


(*) Negative transition এ Active হবে না

Asynchronous Counter (propagation)

এখানে সবগুলো Toggle করে প্রথম Flip-flop এর Output হতে কোনোটা কোন অবস্থা পাবে Toggle করবে, আর এটাই হলো propagation Delay

Synchronous Counter :-



$T = 1$ হলে Toggle করবে

$T = 0$ " " " না।

1 / 0 হলে Toggle

এখানে সবগুলো একই সাথে Toggle করছে। ফলে value ও একই " Change হয়। propagation Delay হয় না।

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Sequential Circuit :-

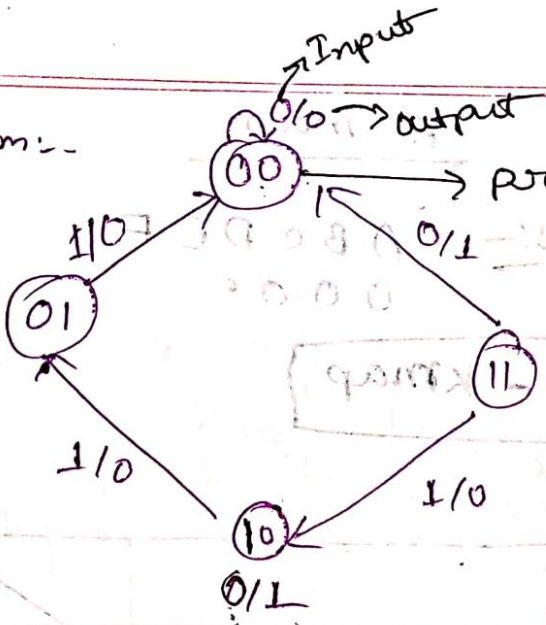
* Design sequential circuit having
  1 input → 1 output

⇒

State Diagram → State Table → State Reduction → State Assignment → Determine No. of Flip-flop (অর্থাৎ, যতগুলো state থাকবে ততগুলো Flip-flop
2^m = No. of states,
m = Flip-flop)

→ Derive Circuit Excitation table from state table → Implement circuit

State Diagram:-



State Table:-

Present State		Next State				Output	
		x=0		x=1			
A	B	A	B	A	B	x=0 y	x=1 y
0	0	0	0			0	1
0	1	1	0	0	0	0	0
1	0						
1	1						

Spring-2019

Magnitude Comparator:-

The comparison of two numbers is an operation that determines if one number is greater than, or equal to the other number.

Suppose A & B two numbers that determines their relative magnitudes.

There are three variables that indicate

$A > B,\ A = B,\ A < B$

$x_i = A_i B_i + A_i' B_i'$

AB+A'B'

$A = B \Rightarrow x_3 \cdot x_2 \cdot x_1 \cdot x_0$

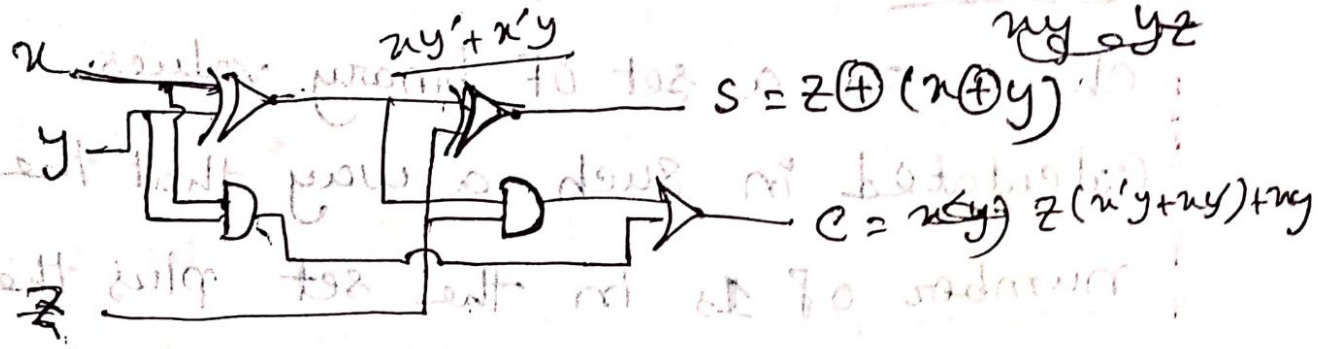
$A > B \Rightarrow A_3 B_3' + x_3 A_2 B_2' + x_2 \cdot x_3 A_1 B_1' + x_3 \cdot x_2 \cdot x_1 \cdot A_0 B_0'$

$(A < B) = A_3' B_3 + x_3 A_2' B_2 + x_2 \cdot x_3 A_1' B_1 + x_3 \cdot x_2 \cdot x_1 A_0' B_1$

Spring-19

2(b) Two half adder & $x+y$

Full Adder with extra OR gate:-



Spring-18

$C = (xy'+x'y)\cdot z + xy$

(*) Understand Combinational circuit:-

→ perform various types of digital operation

→ It is the combination of gates

→ It has no memory

→ Means the present output of the circuit is not depending on the previous output.

→ spring-19

(3)(a)

Parity bit :- bit which act ~~take~~ as a checker on a set of binary values, calculated in such a way that the number of 1s in the set plus the parity bit should always be even.

Autum-18

3(a) The parity generator & parity checker's main function is to detect errors in data transmission.

The parity is nothing but number of 1's and there are two types of parity bits they are even & odd bit. In Odd parity the code must be in odd number.

For example:-

5-bit code 10011 → there are three ones so it is said to be odd parity.

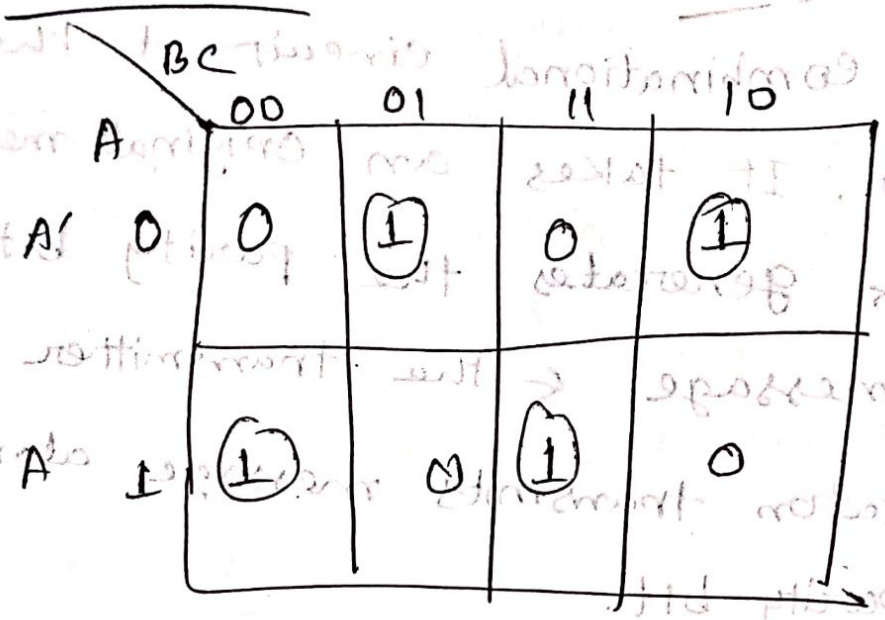
parity generator:-

It is a combinational circuit at the transmitter. It takes an original message as input & generates the parity bit for that message & the transmitter in this generator transmits message along with its parity bit.

3 (b) **3 bits even parity :-**

A	B	C	even p.
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

K-Map



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

$$= A'A(B'C + BC') + B(A'C' + AC)$$

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

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

$(x + x')$
$B \cdot 1$
$x \cdot \bar{x}$

$$= B'(A \oplus C) + B$$

$$= B'\overline{(A \oplus C)} + B\left(\overline{\overline{AC}} + \overline{\overline{A'C'}}\right)$$

$$= B'(A \oplus C) + B\left(\overline{A \oplus C}\right)$$

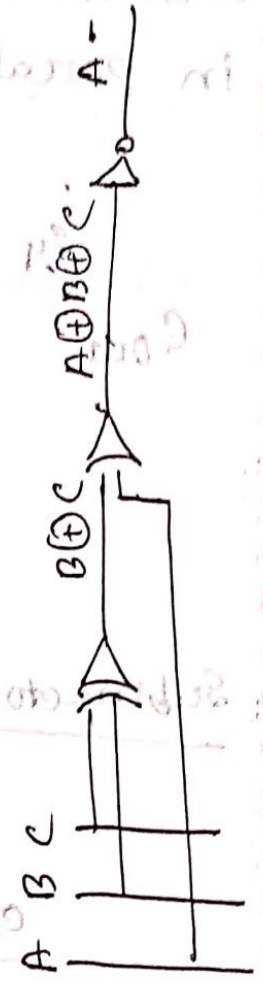
$$= (A \oplus C)(B + B') \qquad B'x + Bx' = B \oplus x$$

$$= B \oplus (A \oplus C)$$

Odd parity :-

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

A \ BC	00	01	11	10
A' 0	(1)	0	(1)	0
A 1	0	(1)	0	(1)



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

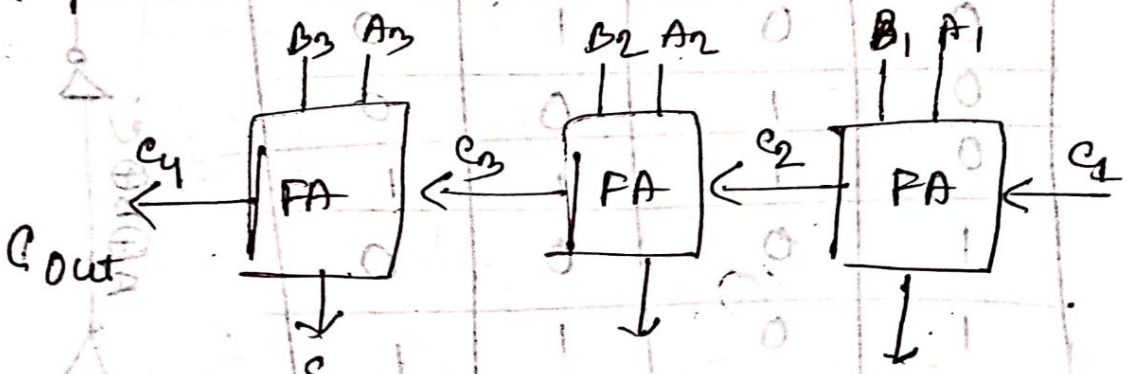
$= A(B'C + BC') + A'(B'C' + BC)$

$= A(B \oplus C) + A'(\overline{B \oplus C})$

$= Ax + A'\bar{x} = \overline{A \oplus x} = \overline{A \oplus (B \oplus C)} = A \oplus B \oplus C$

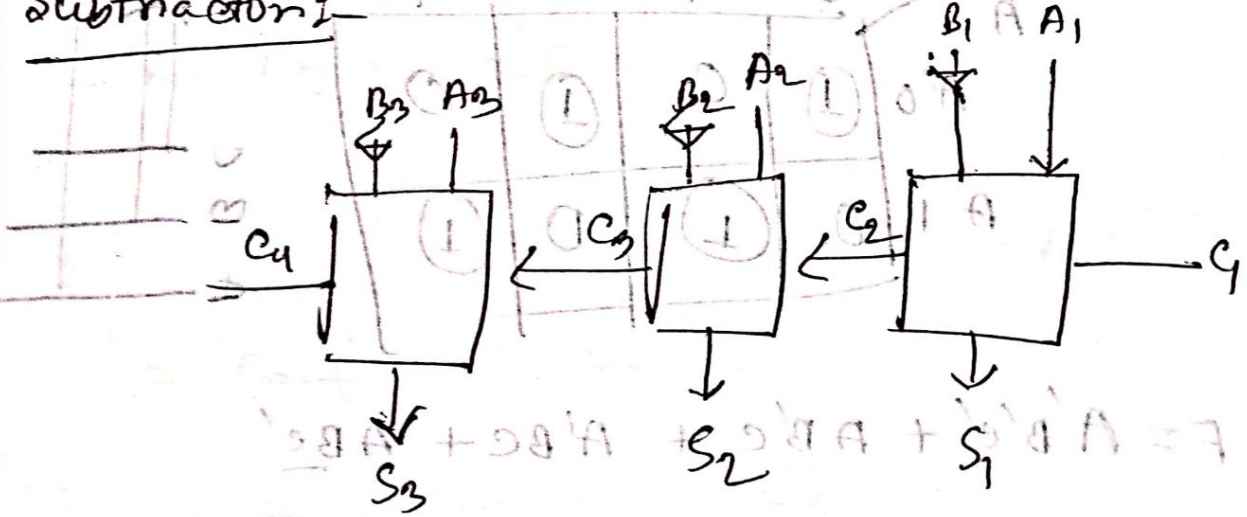
Binary parallel Adders

Digital function that produces the arithmetic sum of two binary numbers in parallel. Consists of Full adder



3 bit parallel Adder

Subtractor:



[THE END]