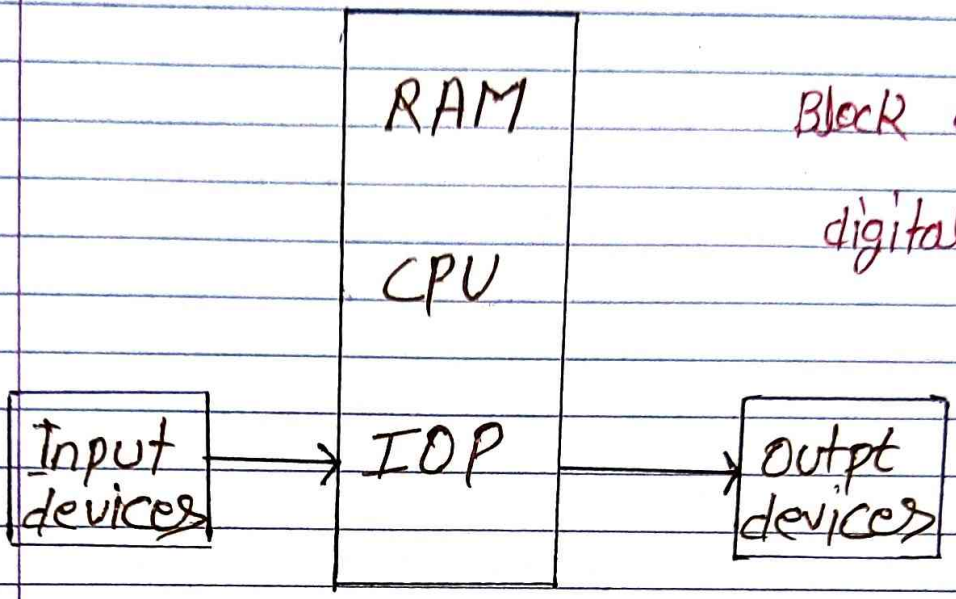
CSA - Computer System Architecture Notes

INDEX

Youtube by Coalboymannu

Name: Std: Sec: Roll No:

chapter - 0
Basic of computer



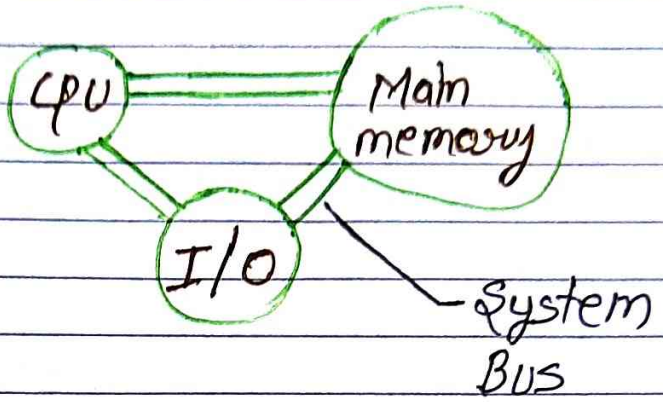
Block diagram of a digital computer.

RAM - Random access momery
CPU - Contral processing unit
IOP - Input - Output processor

Basic functions that a computer can perform

1. Data Processing - widly variety of forms
2. Data stoprage - temporary, permanent
3. Data Movement - it self, outside world [read write]
4. Contral -

The Top level structure of a computer



I/O - Input and Output, Move data between the computer (internal environment) and external environment.

System System Bus - Communication pathway used to transfer data b/w various components of a computer system.

Types of Buses

1. System Bus - high speed Bus it transfers the date b/w CPU ↔ RAM
2. Address Bus - Contains the physical address of a memory or O/I
3. Data Bus - Carring data b/w CPU, memory and other peripheral devices!

4. Control Bus - Carries control signals that coordinate various operation within the computer system. like - read and write signals, interrupt signals and clock signals.

5. Peripheral Bus - Connect peripheral devices like hard devices, graphics cards and network cards to the motherboard.

6. Communication Bus - All networking and telecommunication connection like. the Ethernet bus are used to transfer data b/w device in a network.

chapter - 4

Digital Logic Circuits

- Digital Logic Gates
- Flip flops and their characteristic table
- Logic circuit simplification using
  - Boolean algebra
  - K-map karnaugh
- Don't care conditions
- Combinational circuits
- Introduction to sequential Circuits

Logic Gates

- Signals - Binary information is represented in digital computers by physical quantities called signals.

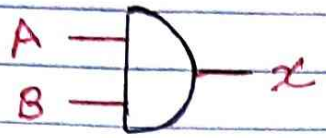
Signals contains just two voltage level or state, labelled as logic "0" and "1"

0 - low voltage (0)
1 - high voltage (+5)

Gates

- The manipulation & processing of binary information is done by logic circuits called gates.
- Gates are blocks of hardware that produces signals require of binary 1 or 0 when input logic requirements are satisfied.
- A variety of logic gates are commonly used in digital computer system :-
  - Each gate has a distinct graphic symbol
  - Operation can be expressed by means of algebric expression.
- The input-output relational of binary variable for each gate can be represented in tabular form by a Truth table

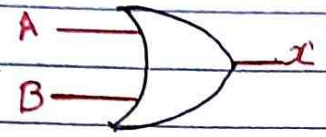
Name	Graphic Symbol	Algebraic function	Truth table



AND

$x = A.B$ or $x = AB$

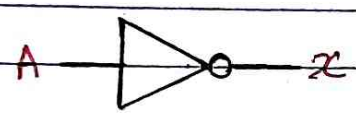
A	B	x
0	0	0
0	1	0
1	0	0
1	1	1



OR

$x = A + B$

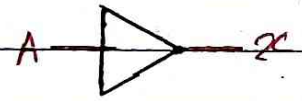
A	B	x
0	0	0
0	1	1
1	0	1
1	1	1



$x = A'$

A	x
0	1
1	0

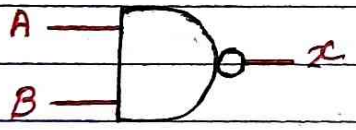
Inverter - NOT



$x = A$

A	x
0	0
1	1

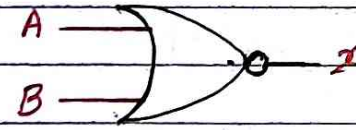
Buffer



NAND

$x = (AB)'$

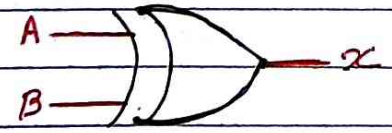
A	B	x
0	0	1
0	1	1
1	0	1
1	1	0



NOR

$x = (A + B)'$

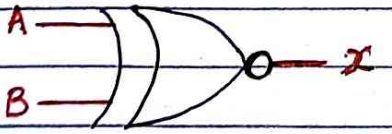
A	B	x
0	0	1
0	1	0
1	0	0
1	1	0



Exclusive-OR XOR

$x = A \oplus B$ or $x = A'B + AB'$

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0



Exclusive-NOR

$x = (A \oplus B)'$ or $x = (A'B + AB')'$

A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Boolean algebra

Boolean expression is a mathematical expression that evaluate to either true (1) or false (0)

AND - Boolean product "." &.

OR - Boolean sum "+"

NOT - Complement "'" or "¯"

Boolean expression

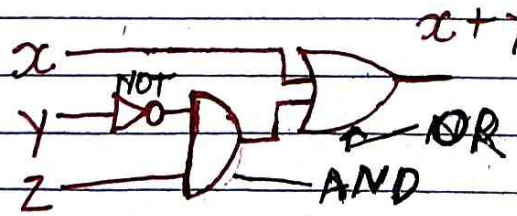
Boolean function

$f = x + y'z$

① x	② y	③ z	$y'z$	④ x	$x + y'z$
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	0	1	1

Truth Table

first we complement the all y values then and with z



Logic Digram

Basic identities of Boolean Algebra

1	$x + 0 = x$	13) $x \cdot 0 = 0$
2	$x + 1 = 1$	14) $x \cdot 1 = x$
3	$x + x = x$	15) $x \cdot x = x$
4	$x + x' = 1$	16) $x \cdot x' = 0$
5	$x + y = y + x$	17) $xy = yx$
6	$x + (y + z) = (x + y) + z$	
7	$x(y + z) = xy + xz$	
8	$(x + y)' = x'y'$	
9	$x + yx = (x + y)(x + z)$	
10	$x(yz) = (xy)z$	
11	$(xy)' = x' + y'$	
12	$(x')' = x$	

DeMorgan's law → $(x + y)' = x'y'$
$(x \cdot y)' = x' + y'$

Q Simplify the following expressions using Boolean expression

1. $A'BC + AC$
2. $A'B + ABC' + ABC$
3. $(BC' + A'D)(AB' + CD')$

1. $A'BC + AC$
$C(A'B + A)$

2. $A'B + ABC' + ABC$
$= A'B + AB(C' + C)$
$= A'B + AB(1)$
$= B(A' + A) = B$

3. $(B\bar{C} + \bar{A}D)(A\bar{B} + C\bar{D})$
$= A\bar{B}(B\bar{C} + \bar{A}D) + C\bar{D}(B\bar{C} + \bar{A}D)$
$= A\bar{B}B\bar{C} + A\bar{B}\bar{A}D + CB\bar{D}\bar{C} + C\bar{D}\bar{D}\bar{A}$
$= 0 + 0 + 0 + 0$
$= 0$

Theorem of Boolean Algebra

commutative law
$A + B = B + A$
$AB = BA$

Associative law
$A + (B + C) = (A + B) + C$
$A(BC) = (AB)C$

Distributive law
$A(B + C) = AB + AC$
$A + BC = (A + B)(A + C)$

Map Simplification

We can simplify a boolean expression using 2 methods

boolean Algebra K-map
Karnaugh map

standard forms of boolean Expression

Two standard forms (function)

SOP - Sum-of-Product
POS - Product-of-sum

These standardization make the boolean expression much more systematic and easier. evaluation, simplification, implementation.

Difference B/w SOP and POS

SOP	POS
• When function $f = 1$ then we group	• When function $f = 0$ then we group
• Here, $A \cdot A' = 0$ ($= 1$)	• Here, $A \cdot A' = 1$ ($= 0$)
• Ex- $AB + BC + AC$	• Ex $(A+B)(B+C)(A+C)$
• Boolean function $f(A,B,C) = \Sigma(0,2,4)$ or $f(A,B,C) = \Sigma m(0,2,4)$	• Boolean function $f(A,B,C) = \Pi(7,6,5,3)$ or $f(A,B,C) = \Pi M(7,6,5,3)$
• We make group of ONES (1)	• We make group of ZEROS (0)
• Term 00, 01 ↓ $\bar{A}\bar{B} + \bar{A}B$ ($A'B' + A'B$) → $A\bar{B} + A\bar{B}$ ($AB + AB'$)	
• We called it [Minterm]	• We called it [Maxterm]

K-map (karnaugh-map)

Express f in sum of products form

Ex-

AB	F
0 0	0
0 1	1
1 0	1
1 1	1

include only 1

When f = 1

$0 \rightarrow \bar{A}$

$1 \rightarrow A$

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

$$f(A,B) = m_1 + m_2 + m_3$$

Standard or Canonical form

When all the variables of the function are included in the each term of an expression.

Ex- $f(A,B) = \Sigma(1,2,3)$

$= \bar{A}B, A\bar{B}, AB$

Minterm

Express f in Product of Sum

Ex-

	A	B	f
$m_1 \rightarrow$	0	0	0
$m_2 \rightarrow$	0	1	1
$m_3 \rightarrow$	1	0	0
$m_4 \rightarrow$	1	1	1

When $f = 0$

$0 \rightarrow A$
$1 \rightarrow \bar{A}$

include only 0

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

Maxterm

Q How many terms for

3 variable $2^3 = 8$

4 variable $2^4 = 16$

⋮

Q Represent / Expand $f(A,B) = \bar{A} + \bar{B}$ in SOP and POS and find minterm and maxterms.

Create standard SOP

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

B is missing → A is missing

$A + \bar{A} = 1$
$\bar{A} \cdot 1 = \bar{A}$

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

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

$= 01 + 00 + 10$

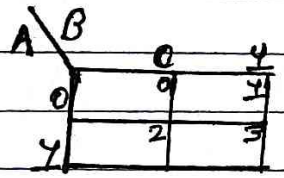
$= m_1 + m_2 + m_3$

$F(A,B) = \Sigma m(0, 1, 2)$

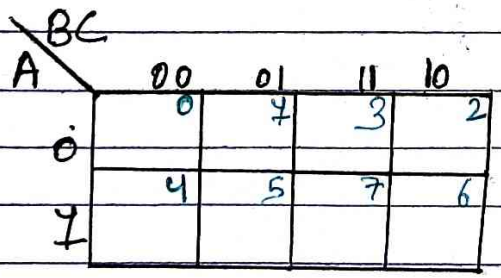
$= (\bar{A} + \bar{B})$

K - maps for two, three, four variables functions

1. Two-variable K-map $2^2 = 4$ cells

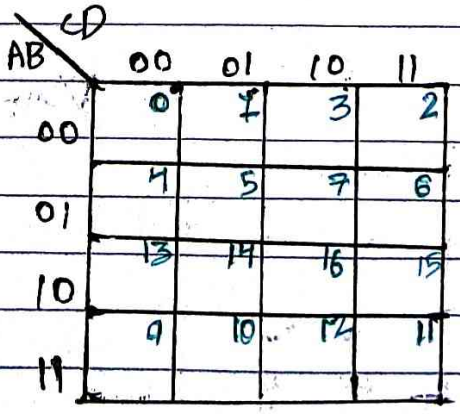


2. Three-variable K-map $2^3 = 8$ cells



3. four-variable K-map

$2^4 = 16$ cells



Questions

1. Minterm and Maxterm will be given in question

$\Sigma m\ (1,2,3,5,7,10)$

$\Pi M\ (1,2,3,5)$

2. Boolean expression will be given in question

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

Simplify boolean function using K-map in SOP and POS form and implement with gates

$f(A,B,C) = \Sigma m\ (0,2,3,4,5,6)$

Step1 ⇒ Draw the map and fill with '1'

Step2 ⇒ Grouping

Step3 ⇒ Create Expression

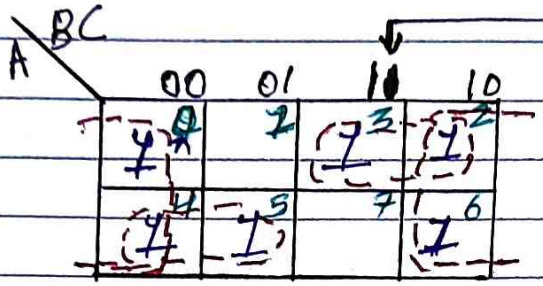
Step4 ⇒ Implement using Gates

Remember
Σm → minterm (1)
(2) ΠM → maxterm (0)

Σm → gate → AND then OR
ΠM → gate → OR then AND

$F(A,B,C) = \Sigma m(0,2,3,4,5,6)$

Step 1 =

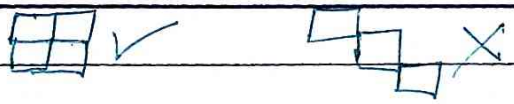


Why 11 not 10, Because We can not change Both variables at same time if we put 10 here 0→1, 1→0 ✗

• 3 groups

Step 2 = How to do grouping

1. Grouped cells must be Adjacent cells like



2. Each group should be as large as possible

3. Overlapping allowed

4. Try to create fewer no' of groups

minimum number of Groups

5. Number of cells in a group must 2^n of cells in each group

in this case $2^0 = 1$, $2^1 = 2$
$2^2 = 4$, $2^3 = 8$

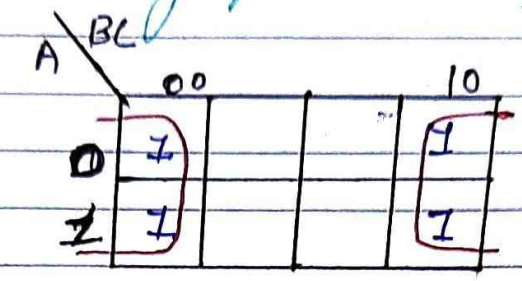
1, 2, 4, 8

group can not 6× 7× 3× contains cells.

Step 3. How to create expression

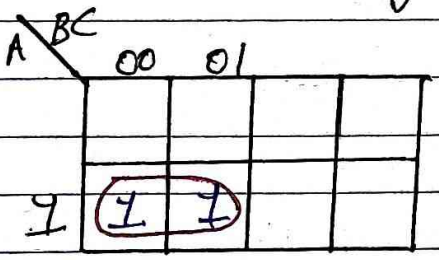
Select only those variables who are ~~NOT~~ changing in a group

like



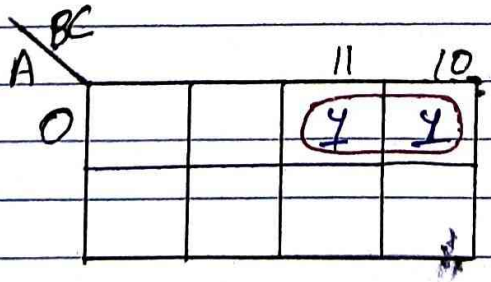
$\bar{C}$

In this Group C is not changing so we use 'C' to make expression But C is 0 that's why $0 = \bar{C}$



~~First~~ we enclude A Because A have single value and B Because it have ~~same~~ 0 same values. $0 = \bar{B}$

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

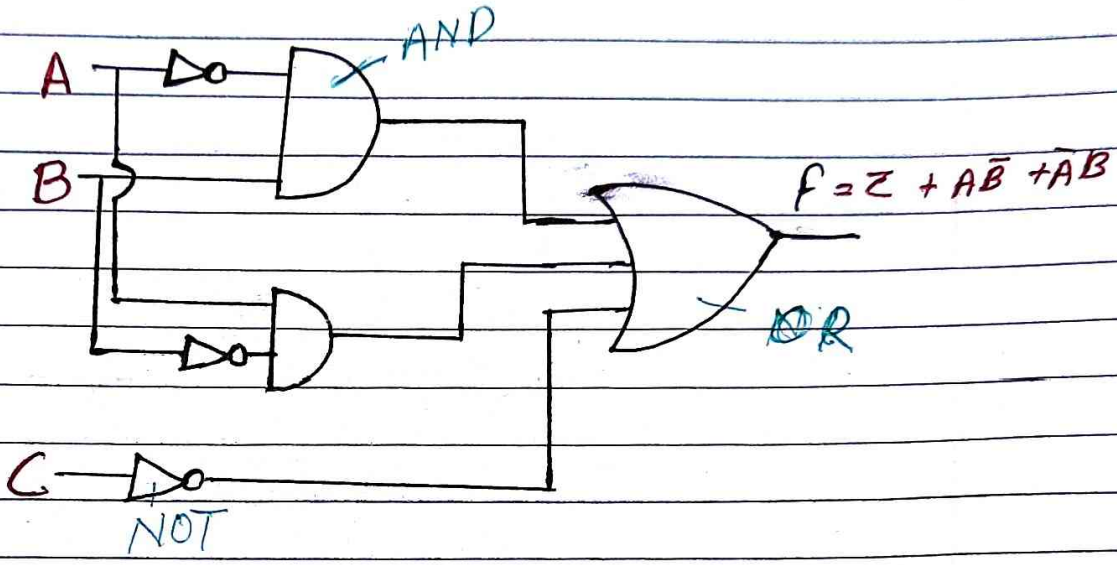


We include A and B, A have one value - B have same

Now we have completed expression

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

Step 4 Implementing using gate.



Q Simplify the following boolean expression using K-map

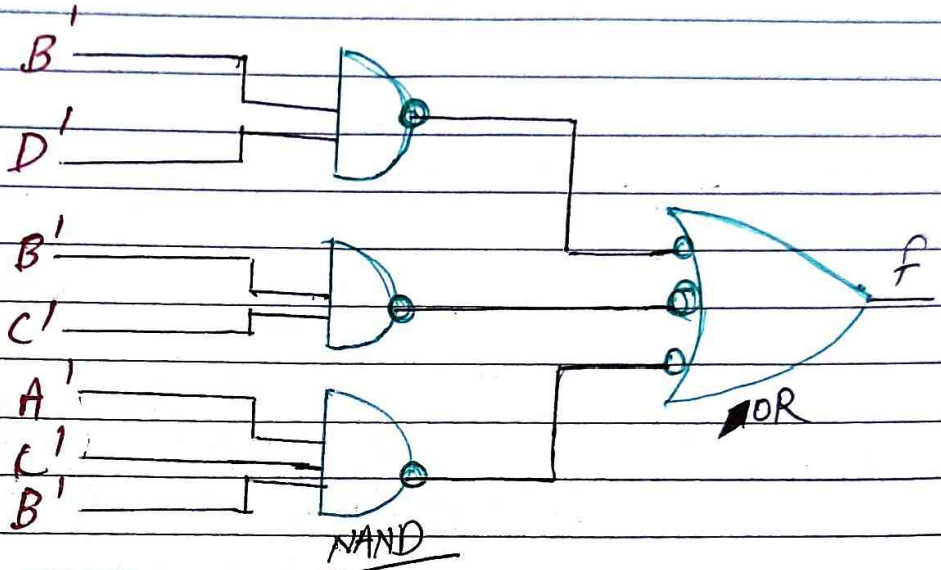
$f(x,y,z) = \Sigma(1,2,3,6,7)$
$f(x,y,z) = \Sigma(0,1,5,7)$
$f(x,y,z) = \Sigma(3,5,6,7)$
$f(x,y,z) = \Sigma(0,2,3,4,6)$

$\Rightarrow f(A,B,C,D) = \Sigma(0,1,2,6,8,9,10)$

NAND Implementation

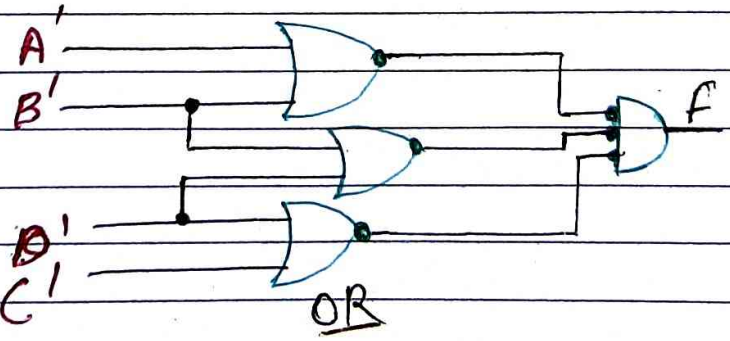
A **SOP** expression can de implemented with NAND gates

$$F = B'D' + B'C' + A'CD$$



A **POS** expression can be implemented with NOR gates

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

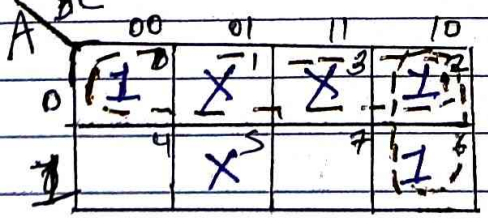


Don't-Care Condition

- The 1's and 0's in the map represent the minterm that make the function equal to 1 or 0.
- There are occasion when it does not matter if the function produce 0 or 1 for the given minterm We don't care what the function's output in ~~the~~ to be for this minterm.
- Marked with 'X' in the map.
- These don't care condition can be used for providing function implification of the algebraic expression.
- if the don't care condition helps to minimize the number of groups. use it
- Usi don't care condition help to minimize No. of gates.

don't care condition

$F(A,B,C) = \Sigma(0,2,6)$ $d(A,B,C) = \Sigma(1,3,5)$



$F = \bar{A} + AB\bar{C}$

Without Don't care condition

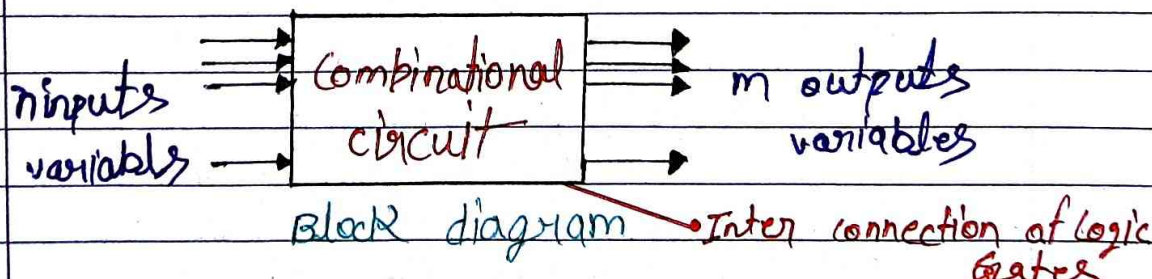
$F = \bar{A}\bar{C} + B\bar{C}$

Combinational Circuits

Digital logic circuits are basically categorized into 2 types
1. Combinational circuits
2. Sequential circuits

- Combinational
  - No feedback Path
    - Not a mediator
    - No take ~~input~~ from output to input
  - NO Memory

- Sequential
  - There exists feedback paths from outputs to inputs
    - Gives a output that used for other's input.
  - They have memory.



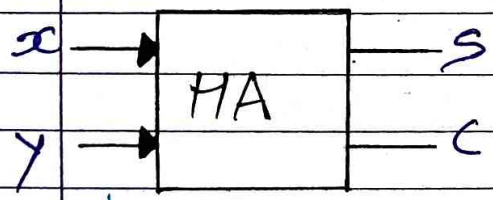
A combinational circuit transform binary information from the given input data to the required output data.

- Combinational Circuits are specifically designed for efficiently adding multiple numbers.

Half Adder

- A half adder is an arithmetic circuit that generates the sum of two binary digits.
- Circuit has two inputs and outputs.
- The Output variables produce the sum and carry.

S C



Block diagram

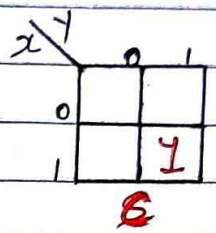
Truth Table

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

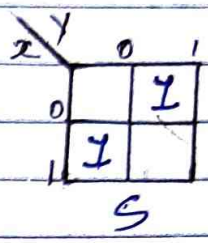
The Boolean expre function for the two outputs (s c) can be obtained directly from the truth table.

$$S = \bar{x}y + x\bar{y} = x \oplus y$$

$$C = xy$$



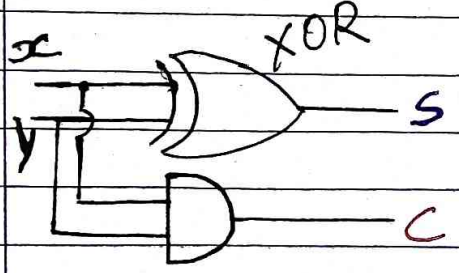
C



S

K-map

Logic diagram

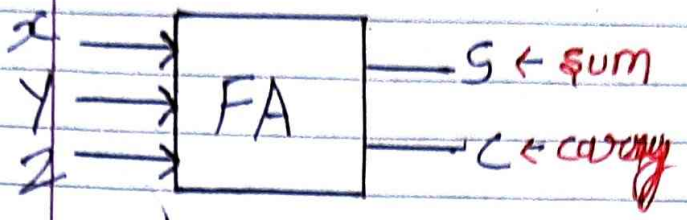


Full-Adder

Combinational circuit that perform addition of 3 bits

$$\begin{array}{r} 1 \\ 1 \\ +\ 1 \\ \hline 1\ 1 \end{array}$$

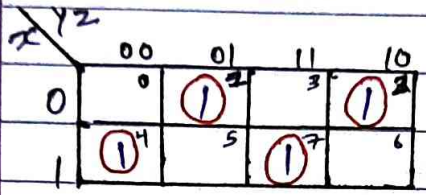
It consist of three input and two output



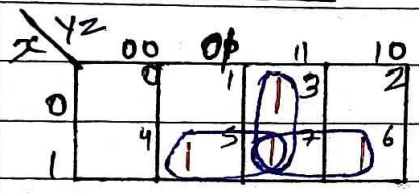
x	y	z	C	S	
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	2
0	1	1	1	0	3
1	0	0	0	1	4
1	0	1	1	0	5
1	1	0	1	0	6
1	1	1	1	1	7

Truth table

Boolean expression



$$x \oplus y \oplus z = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$

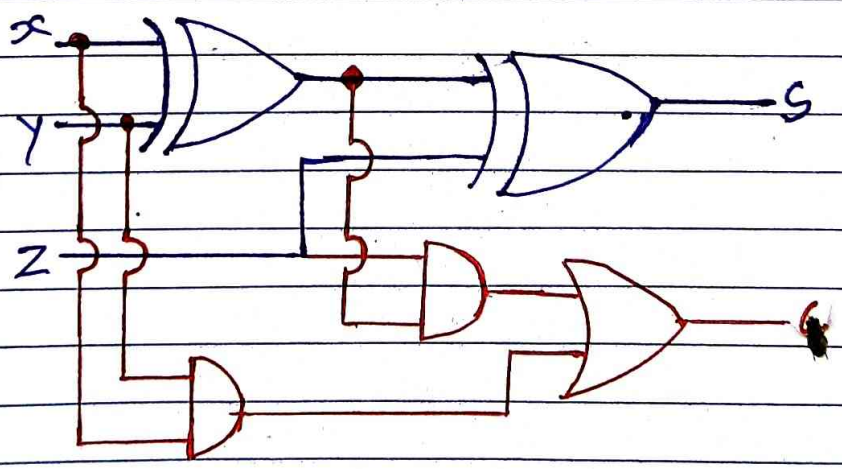


$$= xy + (x \oplus y)z$$

x

$xz + yz + xy$

Logic Diagram

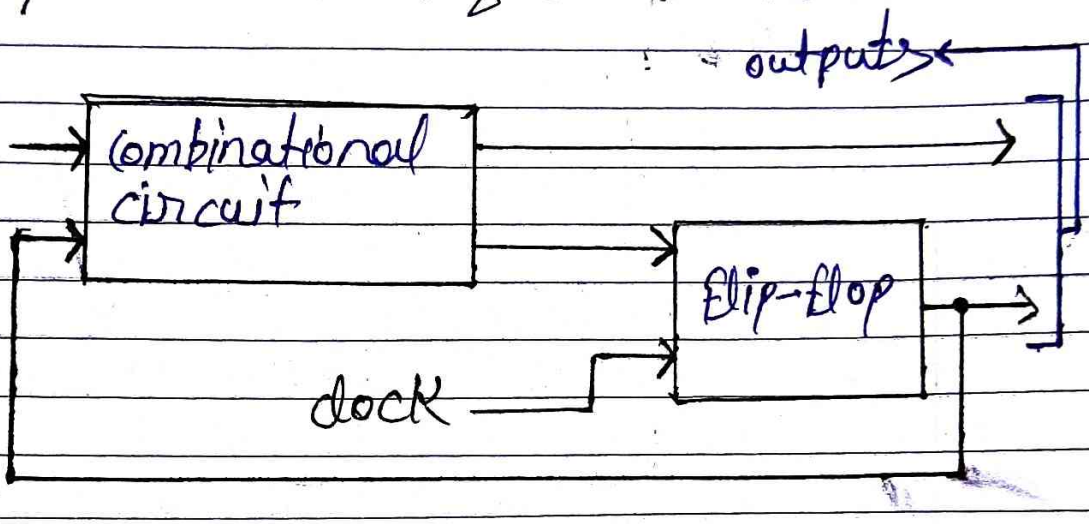


Sequential Circuits

A sequential circuit is an interconnection of flip-flop and gats.

Output depends not only on the present combinations of input signals but also on ~~past~~ sequence of past inputs.

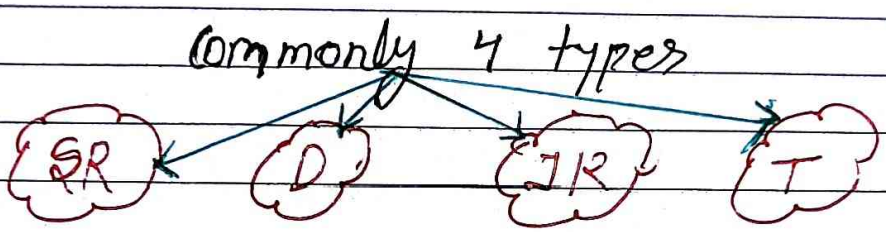
The block diagram of a clocked synchronous sequential circuit



Flip-Flops

Flip-flops are a fundamental building block of sequential logic circuits.
- store the data
- Fundamental bulding block for creating memory elements.

- The behavior of flip-flops is closely tied to clock signals.



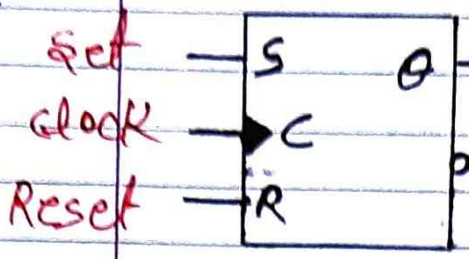
1. SR Flip-flop

- S - set R - Reset
- if there is no signal at clock input C, the output of the circit cannot change
- Only when clock signal change from 0 to 1 can the output be affected according to the value in input S and R.
- The Interminate condition make SR flip-flop difficult to manage. (Read naly)

3 inputs

Graphic Symbol

2 output



$Q(t)$ → Present state

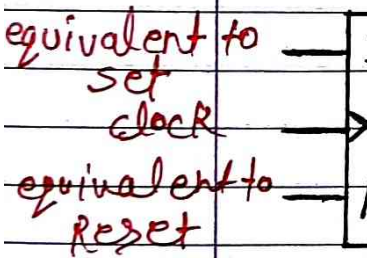
$Q(t+1)$ → Next state

Characteristic Table

S	R	$Q(t+1)$	
0	0	$Q(t)$	→ No change
0	1	0	→ Clear to 0
1	0	1	→ Set to 1
1	1	?	→ Indeterminate [Unpredictable] [0/1]

2. JR flip-flop

- It is refinement of the SR FF, the indeterminate condition of the SR type is defined in the JR type

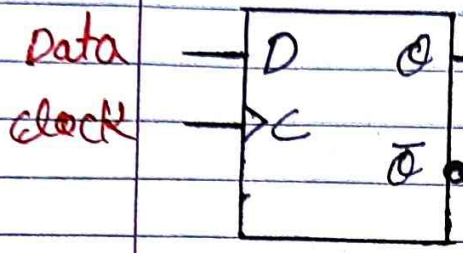


J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$ ↓ complement of $Q(t)$

- $Q(t+1) = Q'(t)$ when both J and K are equal to 1.

3. D Flip flop

- Data flip-flop
- With a signal data input it transfers the input directly to the output.
- It widely used in registers, counters and other sequential logic circuit.



D	Q(t+1)	
0	0	clear to 0
1	1	set to 1

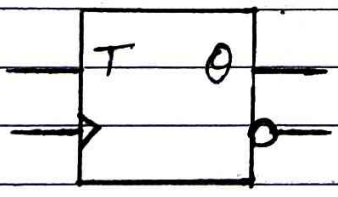
$$Q(t+1) = D$$

4. T Flip - flop {T - Toggle}

- The flip - flop is obtained from JK type when inputs J and K are connected to provide a single input designated by T.
- The T -ff therefore hav only two condition when

$T = 0$ [J=K=0] (no change)

$T = 1$ [J=K=1] (complement)



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

$Q(t+1) = Q(t) \oplus T$

chapter - 2

Digital Components

- Decoders
- Encoders
- Multiplexers.
- Binary Adder
- Binary Adder subtractor
- Binary Incrementor
- Registers
- Registers
- Memory units

Digital components refer to the basic building blocks used in the construction of digital circuits and systems.

Integrated Circuits

Digital circuits are constructed with Integrated circuits (ICs)

ICs - An Ics is small silcon semiconductor crystal, called chip containing the electronic components for logic (digital Gates)

Decoders

Decoders are essential digital components that play a crucial role in converting code ~~outputs~~ inputs into a output (set of output)

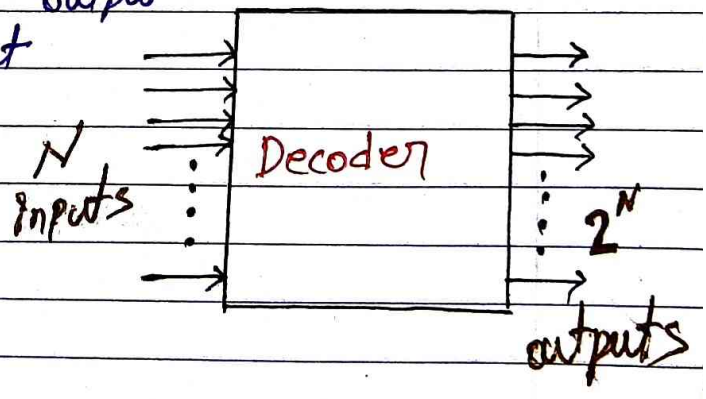
They are often used in digital systems to select one specific output from multiple possibilities. based on the input code applied.

The combinational circuits that change the binary information into 2^N Output lines

N - Number of Inputs
2^N - Number of Outputs

At a time, Only one input line is activated for simplicity.

The Produced 2^N bit output code in equivalent to the binary information.

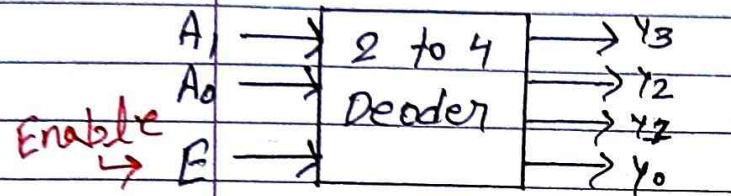


There are various types of Decoders

1. 2 to 4 Line decoder
2. 3 to 8 Line decoder
3. 4 to 16 Line decoder

1. 2 to 4 Line decoder

Block diagram



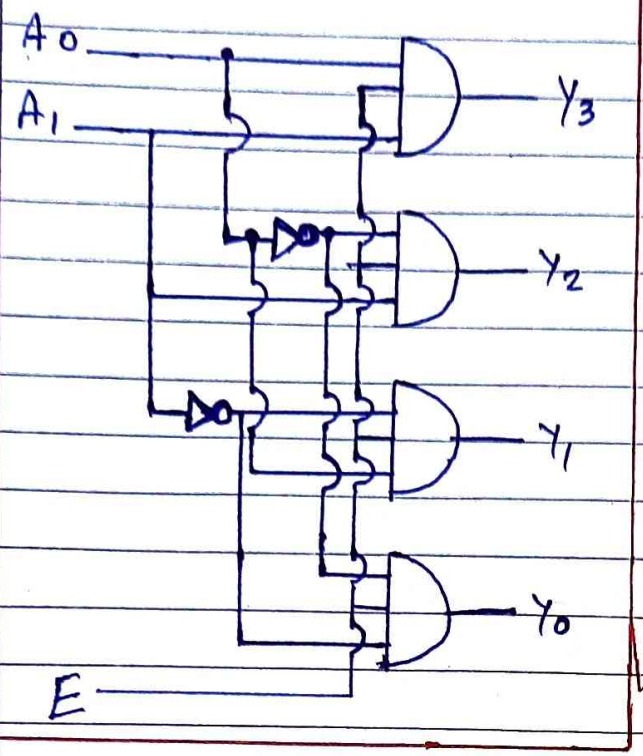
for each combination of inputs when 'E' is set to 1, one of these from output will be 1.

Enable	Inputs		Outputs				
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0	
0	X	X	0	0	0	0	X
1	0	0	0	0	0	1	0
1	0	1	0	0	1	0	1
1	1	0	0	1	0	0	2
1	1	1	1	0	0	0	3

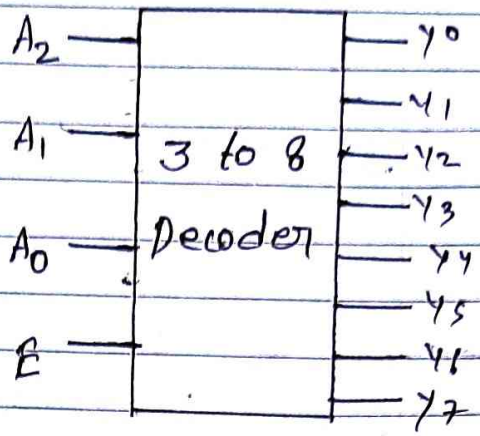
The logical expression of the term Y_0, Y_1, Y_2, Y_3

$Y_0 = E.\overline{A_0}\overline{A_1}$
$Y_1 = E.\overline{A_0}A_1$
$Y_2 = E.A_1\overline{A_0}$
$Y_3 = E.A_1A_0$

Logic



Block Diagram



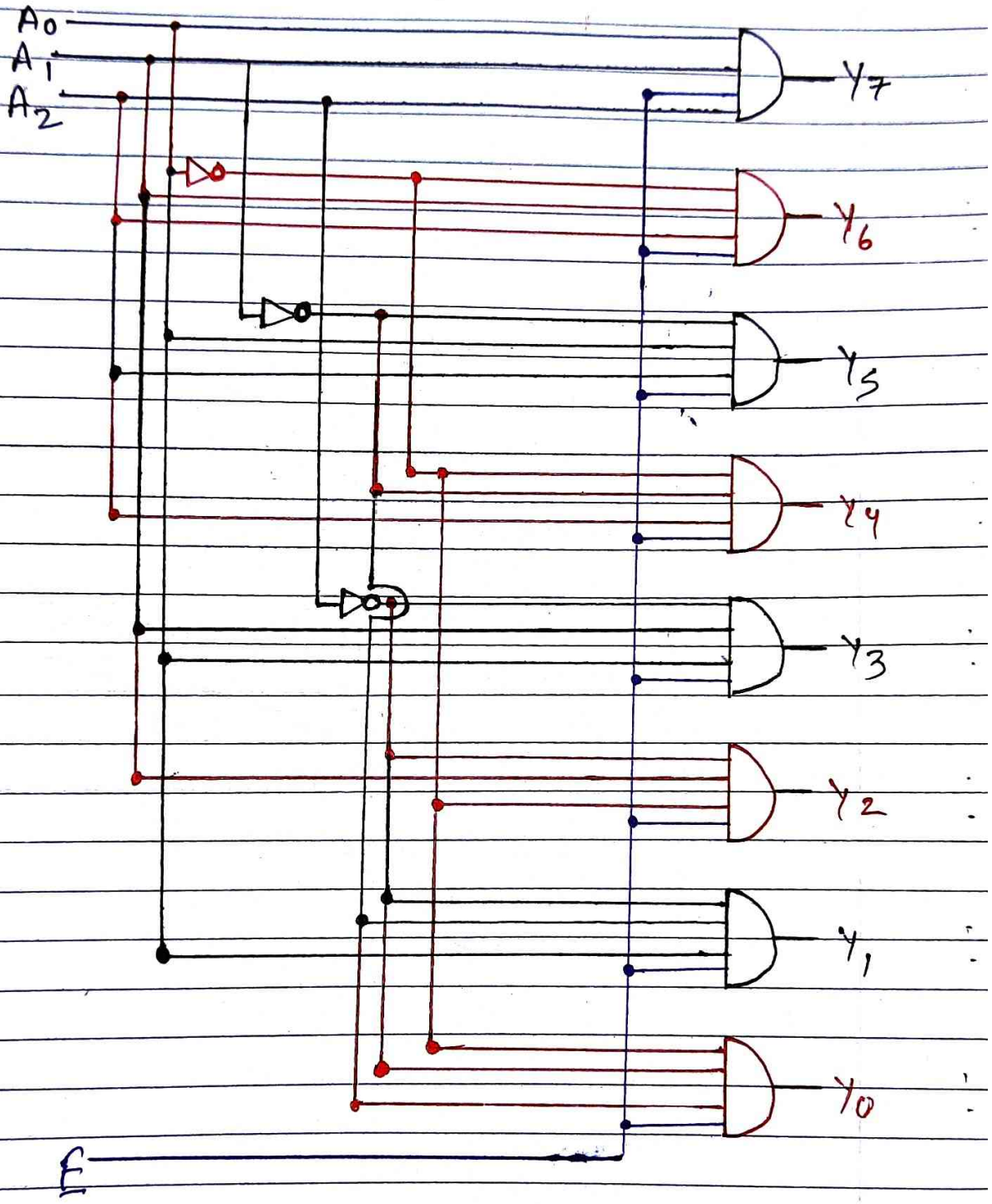
2. 3 to 8 Line Decoder

Truth table

Enable	Inputs			Outputs								
E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	
0	X	X	X	0	0	0	0	0	0	0	0	X
1	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	1	0	1
1	0	1	0	0	0	0	0	0	1	0	0	2
1	0	1	1	0	0	0	0	1	0	0	0	3
1	1	0	0	0	0	0	1	0	0	0	0	4
1	1	0	1	0	0	1	0	0	0	0	0	5
1	1	1	0	0	1	0	0	0	0	0	0	6
1	1	1	1	1	0	0	0	0	0	0	0	7

Logical expression

$Y_7 = A_2 . A_1 . A_0 . E$
$Y_6 = A_2 . A_1 . \bar{A}_0 . E$
$Y_5 = A_2 . \bar{A}_1 . A_0 . E$
$Y_4 = A_2 . \bar{A}_1 . \bar{A}_0 . E$
$Y_3 = \bar{A}_2 . A_1 . A_0 . E$
$Y_2 = \bar{A}_2 . A_1 . \bar{A}_0 . E$
$Y_1 = \bar{A}_2 . \bar{A}_1 . A_0 . E$
$Y_0 = \bar{A}_2 . \bar{A}_1 . \bar{A}_0 . E$



Multiplexers (MUX)

A multiplexer is a combinational circuit that receive binary information from 1 to 2^n input ~~lines~~ data lines and directs it to a single output line.

The selection of a particular input data line for the output is determined by a set of selection inputs.

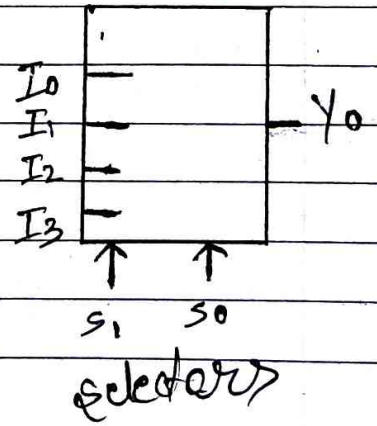
2^n → input data lines
n → input selection lines - Whose ~~input~~ bit combination determine which input data are selected for the output.

It is simply a data selector.

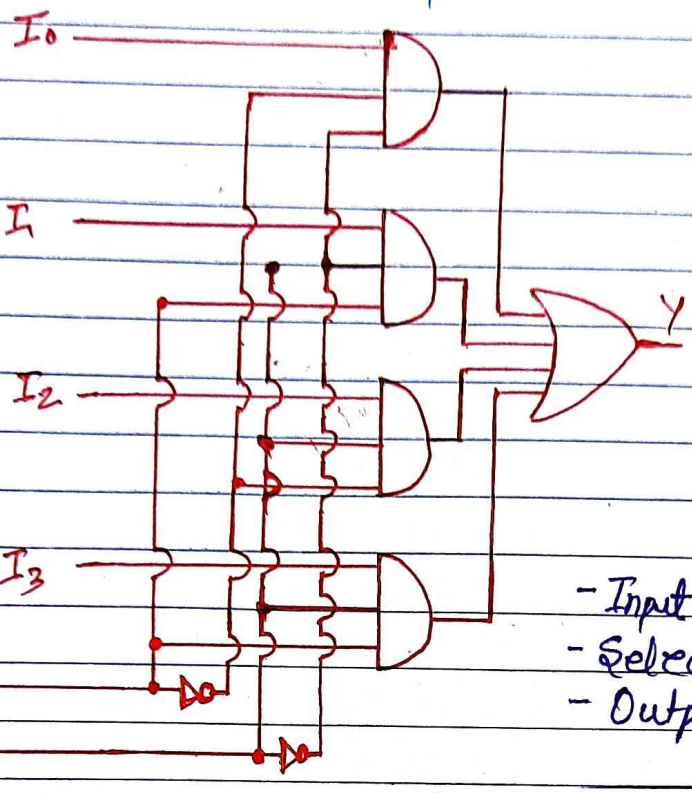
Advantage of MUX

- Reduces no. of wires
- Reduces circuit complexity and cost
- Implementation of various circuit using MUX

Block Diagram

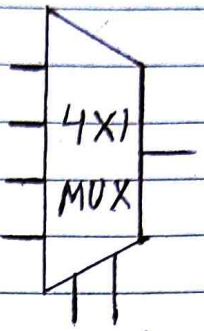


4 to 1 line multiplexer



(1 0 1 0) S_0
(1 1 0 0) S_1

can be represent as



- Input line ($I_0, I_1, I_2 I_3$)
- Select line (S_0, S_1)
- Output line (Y)

Typical application of MUXs are

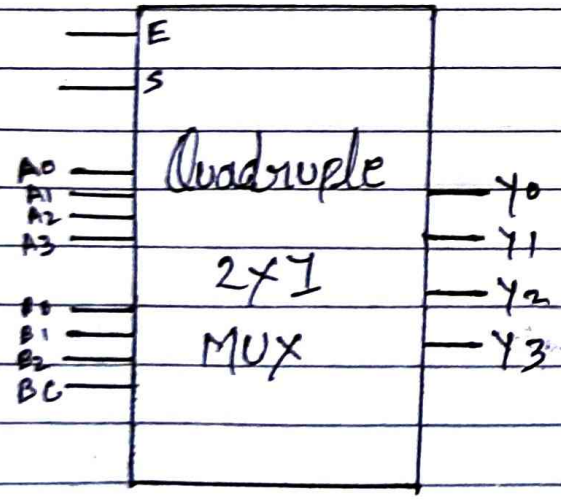
1. Data sorting
2. Parallel - serial conversion
3. Logic function generation

function Table for 4-to-1 line MUX

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

- Now 4X1 MUX has 6 inputs (4 input lines + 2 selectors) and one output.
- A truth table describing the circuit needs 64 rows (2^6 binary combinations) will require an excessively long table.
- So a more convenient way to describe the operation of MUX in by means of a function table.
- In some cases two or more multiplexers are enclosed within a single IC package.
- One example in a Quadruple 2-to-1 line multiplexers.

Block diagram



E	S	Y
0	X	All 0's
1	0	A
1	1	B

- It contain 4 MUXs
- It is also a circit that selects one of two 4-bit data line
- if E=1 and S=0 then A input have path to from output, if S=1 The then B inputs are applied the output

- This circuit has four MUXs each capable of selecting one of two input lines, for example output Y_0 can be selected to come from either A_0 or B_0 Similarly output Y_1 may have the value of A_1 or B_1 and so on.

→ One input selection line S selects one of the line in each of the four MUXs

Memory Unit

- A collection of storage cells
- The memory stores binary information in group of bits called words

- A memory word is a group of 1's and 0's, and may representation
  - a number
  - an information code
  - one or more character, alphanumeric "
  - or any binary coded information

- Most computer memories use words whose number of bits is a multiple of bytes.
  - 16-bit word contains 2 bytes
  - 32-bit word contains 4 bytes

Chapter - 3
Data Representation

- Number System
- Binary representation of both numeric and alphanumeric data.
- Representation of numeric data in different number system
- Binary, Octal, Decimal and Hexadecimal
- Conversion from 1 NS to another Number System.

Complements

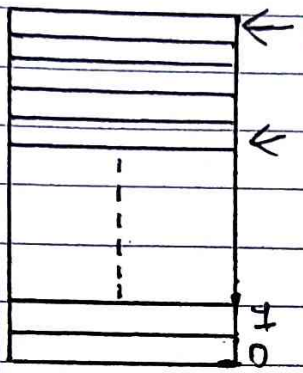
- Representation of signed and unsigned
- Addition and subtraction of signed and Unsigned Numbers

fixed - Point Representation

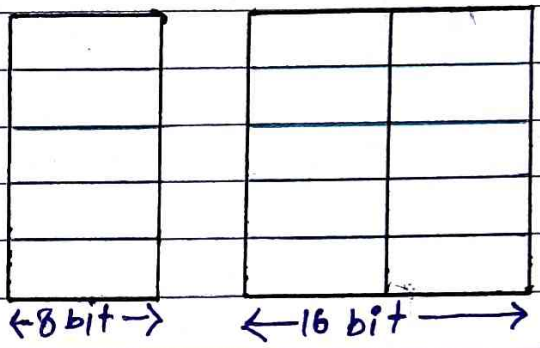
- 3.1 - Data Type
- 3.2 - Complements
- 3.3 - fixed - float Point Represent.

- The capacity of memories in commercial computers in normaly state or the Total number of bytes that can be store
Ex. 8 Gb RAM , 16 GB RAM , 256 Gb SSD.,

Internal Structure of Memory unit



No. of Words it contains, and no. of bits in each word.



3.1. Data Type

- Data is a collection of facts (Unorganized or raw ex: number, symbol, character, word code graph etc)
- Information is a meaning full data (Processed, organized ex: Product price comparison, website traffic change).
- Binary information in computer is stored in memory or processor register.
- Registers are made of of flip-flop

Number System

- A number system of base/redix r in a system that used distinct symbol for r digits.
- To determine the quantity that the number represent, It is necessary to multiply each digit by a integer power of r and then from the sum of all weighted digits.

NS- Number System

We have 4 type of number system

Decimal
base - 10
Symbols
0, 1, 2, 3, 4, 5, 6
7, 8, 9

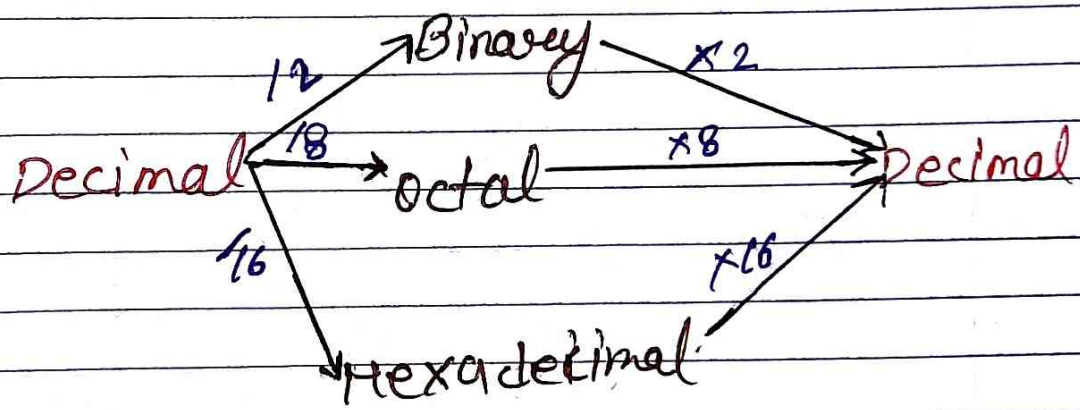
Binary
base - 2
Symbols
1, 0

Octal
base - 8
0, 1, 2, 3, 4
5, 6, 7

Hexadecimal
base - 16
0, 1, 2, 3, 4, 5,
6, 7, 8, 9, A
B, C, D, E, F

A = 10 B = 11
C = 12, D = 13
E = 14, F = 15

Conversion [Decimal to other NS
Other NS to Decimal]



LSB - Least Significant bit
MSB - Most " "

Ex - $(128)_{10}$ ← ( )2, ( )8, ( )16

LSB

2	128	0
2	64	0
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
2	1	1
2	0	(MSB)

$(10000000)_2$

8	128	0
8	16	0
8	2	2
	0	

$(200)_8$

16	128	0
16	8	8
	0	

$(80)_{16}$

Binary to Decimal

$(10101)_2 = (\ )_{10}$
43210

$= 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 16 + 0 + 4 + 0 + 1$

$= (21)_{10}$

Positive Power of 2 (Whole No.)

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

Negative power of 2 (factional no)

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
0.5	0.25	↓	0.0625	↓			
		0.125		0.03125	.....		

Octal to Decimal

$(345)_8 = (\ )_{10}$

$= 3 \times 8^2 + 4 \times 8^1 + 5 \times 8^0$

$= 3 \times 64 + 4 \times 8 + 5$

$= (229)_{10}$

Hexadecimal to Decimal

$(1A7)_{16} = (\ )_{10}$

A = 10 B = 11
C = 12 D = 13
E = 14 F = 15

$= (1 \times 16^2) + (A \times 16^1) + (7 \times 16^0)$

$= 1 \times 256 + 10 \times 16 + 7 \times 1$

$= 256 + 160 + 7$

$= (423)_{10}$

$(0.625)_{10} = (\ )_2 \quad (\ )_8 \quad (\ )_{16}$

0.625 × 2 → 1.250 → 0.500 → 1.000

$(0.101)_2$

0.625 × 8 → 5.000

$(0.5)_8$

0.625 × 16 → 10.000

10 → A

$(0.A)_{16}$

When to stop:
1. Remaining fraction part become ZERO
2. Exhibits a Repeating pattern (...), (333), (48348 3...)

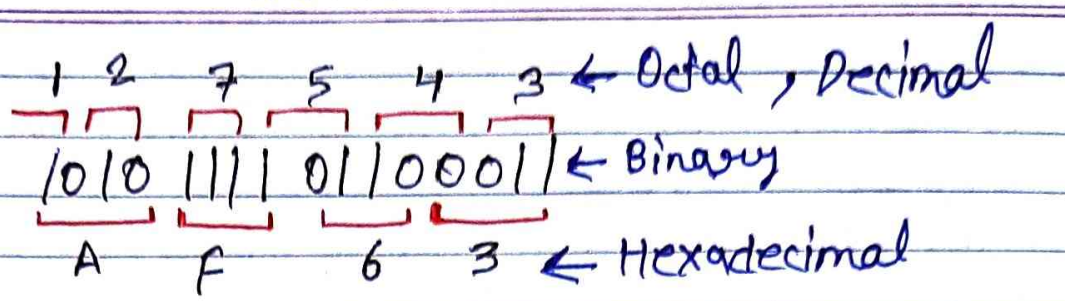
Conversion of Hex, binary, Octal to Decimal

$(736.4)_8 = 7\times8^2 + 3\times8^1 + 6\times8^0 + 4\times8^{-1}$

$= (7\times64) + (3\times8) + (6\times1) + (4\times\frac{1}{8})$

$= (478.5)_{10}$

10	00010000	16
50	01010000	80
FF	11111111	255



Binary - Coded Octal Number (BCO)

Octal No.	BCD	Decimal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
10	001,000	8
~~11~~	001001	9
12	001010	10
⋮	⋮	⋮

Binary coded Hexadecimal Number (BCH)

Hexade. No.	BCH	Deimal No.
0	0000	0
1	0001	1
⋮		
9	1001	9
A	1010	10
B	1011	11
⋮		
F	1111	15

Binary - coded Decimal No (BCD)

Decimal No.	BCD No.
0	0000
1	0001
2	0010
3	0011
⋮	⋮
9	1001
10	0001 0000
11	0001 0001
50	0101 0000

Complements

→ Complements are used in digital computer's for simplifying the subtraction operation and for logic manipulation.

2 Types of complements for each base system

	Ex-	Decimal	binary
1.	r's complement	10's	2's
2.	$(r-1)$'s complement	9's	1's

$(r-1)$'s Complement

Suppose a number N in base r having n digits the $(r-1)$'s of N is defined as $(r^n - 1) - N$

Ex- 9's complement of
546700

$n = 6$ No. of digits = 6
$r = 10$ base = 10
$N = 546700$

$1000000 - 1 = 999999$

$999999 - 546700 = 453299$

9's complement of 546700 is 453299

Ex- 1's complement of
1011001 → binary
0100110 base 2

change the bit from 0 → 1 or 1 → 0 (Reverse The Bit)

(1011001)' complement is
0100110

(r's) complement | 10's complement

Add 1 to (r-1)'s complement

Add 1 to 9's complement

Ex - 10's complement of
2389

9999
-2389
7610 9's comp.

7610
+1
7611 10's comp.

2's complement

Add 1 to 1's complement

Ex - 2's complement of
101011

101011
010100 1's comp.

010100
+1
010101 2's comp

M ← Minuend
−N ← subtrahend

Subtraction of Unsigned Number.

↓
Both Num. are positive

STEPS:-

Add The minuend (M) to the 9's comp. of the subtrahend (N)

if $M \geq N$, the sum will produce an end carry, which is discarded, and what is left in the result M−N.

if $M < N$, the sum does not produce any end carry and is equal to 9's complement of (N−M).
To obtain the answer, take the 9's complement of the sum and place a negative sign in front of.

Subtraction in Decimal Num. Sys.

$M \geq N$,
72532 − 13250
(M) (N)↓
10's comp

```
  99999
- 13250
  86749   ← 9's comp.
+     1
  86750   ← 10's comp
```

Add
Now ~~Subtract~~ 10's comp. of 13250 with 72532

```
  72532
+ 86750
 159282
```

Ans is 59280

M<N

13250 - 72532

```
  99999
- 72532
-------
  27467   ← 9's comp.
+     1
-------
  27468   ← 10's comp.
```

```
  13250
+ 27468
-------
  40718   ← comp. of result
```

```
  99999
  40718
-------
  59281   ← 9's comp.
+     1
-------
  59282   ← 10's comp.
```

← (-) is important

Ans is -59282

Subtraction in binary Num. Sys.

M ≥ N

M = 1010100 - N = 1000011

```
  1000011
  0111100   1's comp
+       1
-------
  0111101   2's comp
```

Now add both

```
  1010100
+ 0111101
---------
 10010001
```

Ans is 0010001

M ≤ N

M = 1000011 - N = 1010100

```
  1000011
+ 0101100   ← 2's compl. of N
-------
  1101111
```

2's comp of result

```
  1101111
  0010001
```

Ans is -0010001

Fixed-Point Representation

- Positive integer including zero, can be represented an unsigned number.
- To represent negative integer, we need a notion for negative value.

(+) } signs are used in ordinary arithmetic
(−) }

- Because of hardware Limitation computer must represent every thing with 1's and 0's, including the sign of a number
- It is customary to represent the sign with a bit placed in the left-most position of the number. Ex- 0 1 0 3 0 0

0 for positive

1 for negative

- In addition to sign, a number may have a binary (Decimal) point.

[1001.0001]
[250.1756]

is needed to represent

fraction integer mixed-integer fraction Number.

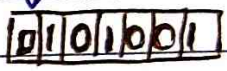
There are two ways of specifying the binary point in a register.

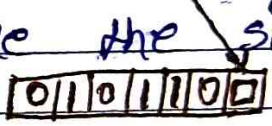
By giving it fixed position

By employing a floating-point representation.

Assumes that the binary point is always fixed in one position

To position most widdely used

a binary point in the extreme left of the register to make the stored number a fraction. Ex- 

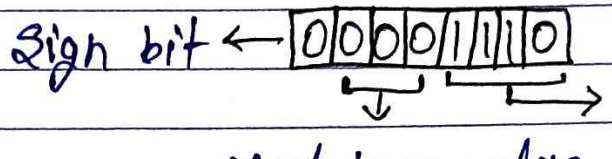
a binary point in the extreme right of the register to make the stored number an integer. Ex - 

In both the case the binary point is not actually present but its presence is assumed from the fact that the number stored in the register is treated as a fraction or an integer

Integer Representation

When an integer binary number is positive, the sign is represented by 0, and the magnitude by a positive number.

Ex. +14 is represented in 8-bit register as



When the number is negative, the sign is represented by 1 but the rest of the number may be represented in one of the three possible ways :-

1. Signed - magnitude representation
2. Signed - 1's complement "
3. Signed - 2's complement "

1. Signed - magnitude representation

for ex - -14 is obtained for +14 by complementing only the sign bit

+14 ⇒ 0 0001110
-14 ⇒ 1 0001110

2. Signed - 1's complement representation

for ex - -14 is obtained from +14 by complement all the bits (including sign bit)

+14 ⇒ 00001110
-14 ⇒ 11110001 — 1's comp. of 14

3. signed - 2's complement representation

for ex - -14 is obtained from +14 by taking 2's complement of all the bits (including sign bit)

+14 ⇒ 00001110
-14 ⇒ 11110010

Signed magnitude system is used in ordinary arithmetic, but in awkward when used in computer arithemetic

1's complement impose difficulties because it have 2 representation of 0 (+0 and -0) so are seldom used for arithmetic operations, but very useful for logical operation.

2's complement is used for signed binary arithmetic (or signed -2's complement is used for representing negative number in morden computing).

Arithmetic Addition

(2's comp. Addition.)

Add the two numbers including their sign bit, and discard any carry out of the sign (left most) bit position.

Negative number must be in 2's complement and if the sum obtained after addition is negative, it is in 2's complement form.

Ex +6 00000110
+13 +00001101
+19 00010011

-6 00000110 11111010 ← 2's comp
+13 + 00001101
+7 00000111

+6 00000110
-13 +11110011
-7 11111001 ← 2's comp of +7

-6 11111010] 2's comp of +6 and +13
-13 +11110011
-19 11101101 ← 2's comp of +19

Arithmetic Subtraction

Subtraction of two signed binary numbers when negative numbers are in 2's complement form is very simple and can be started as follows:

Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including the sign bit) A carry out of sign bit is discarded.

$$(\pm A) - (+B) = (\pm A) + (-B)$$
$$(\pm A) - (-B) = (\pm A) + (+B)$$

A subtraction operation can be changed to an addition operation if the sign of the subtrahand is changed.

$+6 - (+13) \rightarrow +6 + (-13) = -7$

$00000110 + 11110011 = 11111001$ (subtraction through addition)

↳ 2's comp. of +7

$+6 - (-13) \rightarrow +6 + (+13)$

$00000110 + 00001101 = 00010011$

$-6 - (+13) \rightarrow -6 + (-13) = -19$

$11111010 + 11110011 = 111101100$

11101100

$-6 - (-13) \rightarrow -6 + (+13) = +7$

$11111010 + 00001101 = 100000111$

Chapter - 4
Arithmetic Microoperations and Register Transfer ~~Register Transfer~~

Arithmetic Microoperations
- Binary Adder
- Binary Adder-Subtractor
- Binary Incrementor

Microoperation – An elementary operation performed with the data stored in register.

four types of microoperations encountered in digital components.

1. Register transfer
2. Arithmetic microoperation
3. Logic Microoperation
4. Shift Microoperation

transfer data from one register to another

perform arithmetic operation on numeric data

bit manipulation on nonnumeric data

perform shift operation on data store in register.

Register transfer microoperation does not change the information content when the binary information move from the source register to the destination register. for Ex.

$R_2 \leftarrow R_1$

destination source

Other three types of microoperation change the information content during the transfer.

The basic arithmetic microoperations are-

1. Addition
2. Subtraction
3. Increment
4. decrement
5. shift

Addition

- The arithmetic microoperation defined by the statement

$R_3 \leftarrow R_1 + R_2$

add microoperation

It states that the content of R_1 are added to the content of R_2 and the sum transferd to R_3

- To implement this statement with hardware we need.

  → 3 register (R_1, R_2, R_3)
  → And the digital component that perform Addition operation.

Subtraction

- The subtraction microoperation is implement through complementation and Addition.

$$R_3 \leftarrow R_1 + \overline{R_2} + 1$$

is equivalent to
$R_3 \leftarrow R_1 - R_2$

- $\overline{R2}$ is 1's comp of R_2
- $\overline{R_2} + 1$ is 2's comp of R_2

Arithmetic microoperation

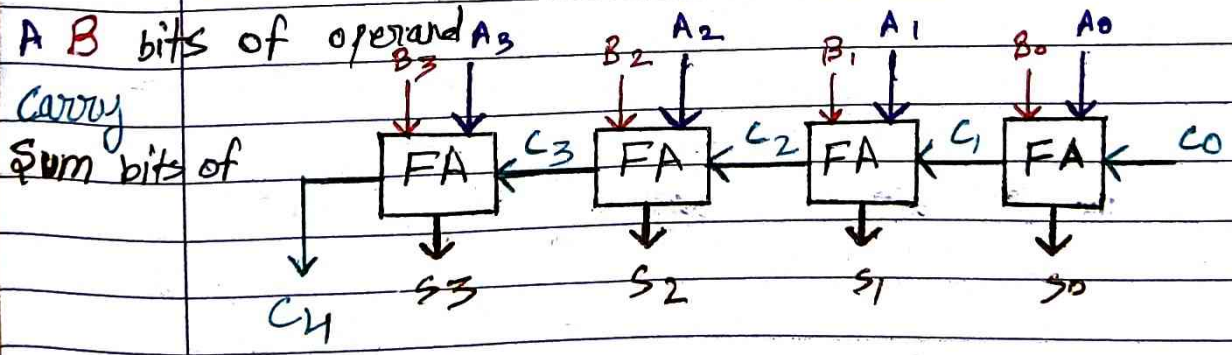
Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Content of R1 + R2 is transfered to R3
$R3 \leftarrow R1 - R2$	" " R1 - R2 " "
$\overline{R2} \leftarrow \overline{R2}$	R2's comp save to R2
$R2 \leftarrow \overline{R2} + 1$	2's " " "
$R3 \leftarrow R1 + \overline{R2} + 1$	R1 + 2's comp [Subtraction]
$R1 \leftarrow R1 + 1$	Increment
$R1 \leftarrow R1 - 1$	Decrement

- The arithmetic operations, multiply and divide are not included in the basic set of microoperation
- The multiplication operations is implemented with a sequence of add and shift microoperation
- The Division operation is implemented with a sequence of subtraction and shift microoperation

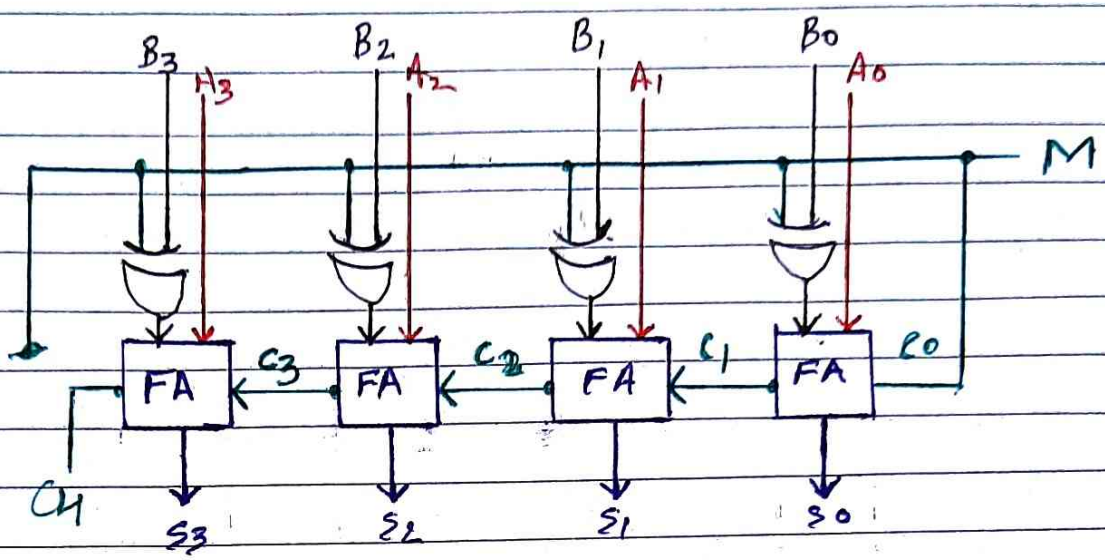
4-bit Binary Adder

The (BA) is constructed with full-adder circuit connected in cascade, with output carry from one full-adder connected to the input carry of the next full-adder

Register will hold the data
Full-Adder is digital component that perform arithmetic sum



4-bit Binary Adder Subtractor



The mode input M controls the operation, when M = 0 the circuit in adder. and when M = 1 The circuit become subtractors.

Each X-OR gate recives input M and one of the B when m = 0, when have $B \oplus 0 = B$. FA received The value of B and input carry in 0, the circuit perform A plus B,

When m = 1 we have $B \oplus 1 = B'$ and $c_0 = 1$ The B input are all complemented and a 1 is added through the input carry.

The circuit perform the operation of A plus

The 2's complement of B

for unsigned numbers, this gives A-B if $A \geq B$ or the 2's complement of B-A if $A < B$. for signed number, the result is A-B provided that there is no overflow.