Digital electronics Handwritten notes

TABLE OF CONTENTS

Note :

★ This table of contents will help you quickly locate any topic.

★ Study concepts thoroughly and solve practice questions regularly.

★ These notes are specially useful for DSSSB TGT Computer Science Exam.

All the Best for your Exam ! ☺

DIGITAL ELECTRONICS

Introduction

- Digital Electronics deals with the study of discrete electronic signals which have two possible states : 0 (LOW) and 1 (HIGH).
- It uses digital circuits to process, store and transmit data in the form of binary digits.

Features of Digital Electronics

→ High speed operation
→ High accuracy and reliability
→ Low power consumption
→ Easy to store and process information
→ Less affected by noise

Number System

1. Binary Number System (Base-2)

- It uses only two digits : 0 and 1.
- Each position is a power of 2.
- Example : $(1011)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$
  $= 8 + 0 + 2 + 1 = (11)_{10}$

$(1101)_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = (13)_{10}$

2. Decimal Number System (Base-10)

- It uses ten digits : 0 to 9.
- Each position is a power of 10.
- Example : $(345)_{10} = 3\times10^2 + 4\times10^1 + 5\times10^0$

  $= 300 + 40 + 5 = (345)_{10}$

3. Hexadecimal Number System (Base-16)

- It uses 16 symbols : 0 to 9 and A, B, C, D, E, F (where A = 10, B = 11, C = 12, D = 13, E = 14, F = 15)
- Each position is a power of 16.
- Example : $(2A)_{16} = 2\times16^1 + 10\times16^0 = 32 + 10 = (42)_{10}$

4. Octal Number System (Base-8)

- It uses eight digits : 0 to 7.
- Each position is a power of 8.
- Example : $(57)_8 = 5\times8^1 + 7\times8^0 = 40 + 7 = (47)_{10}$

Conversions

Conversion	Method
Binary ↔ Decimal	Weighted method
Binary ↔ Octal	Group bits in 3's
Binary ↔ Hexadecimal	Group bits in 4's
Octal/Hex ↔ Decimal	Expand and convert
Decimal ↔ Others	Repeated division method

inary Arithmetic

Binary Addition

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Example : $(1011)_2 + (1101)_2$

$$\begin{array}{r} 1\,0\,1\,1 \\ +\,1\,1\,0\,1 \\ \hline 1\,1\,0\,0\,0 \end{array}$$

Binary Subtraction

A	B	Borrow	Difference
0	0	0	0
1	0	0	1
1	1	0	0
0	1	1	1

Example : $(1011)_2 - (0101)_2$

$$\begin{array}{r} 1\,0\,1\,1 \\ -\,0\,1\,0\,1 \\ \hline 0\,1\,1\,0 \end{array}$$

1's Complement

- Replace 1 by 0 and 0 by 1.

  Example : (10110) → (01001)

2's Complement

- Find 1's complement and add 1.

  Example : (10110)

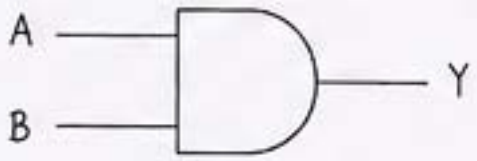
  1's comp = (01001)

  2's comp = (01001) + 1 = (01010)

Logic Gates

Logic gates are the basic building blocks of digital circuits. They perform logical operations on one or more binary inputs to produce a single binary output.

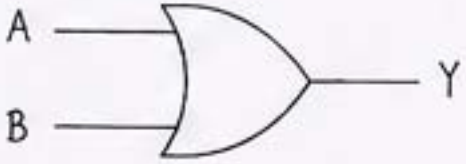
1. AND Gate



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Expression : $Y = A \cdot B$

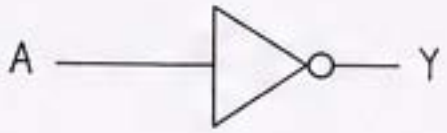
2. OR Gate



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Expression : $Y = A + B$

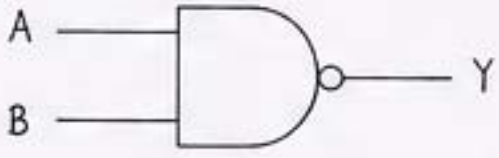
3. NOT Gate (Inverter)



A	Y
0	1
1	0

Boolean Expression : $Y = \overline{A}$

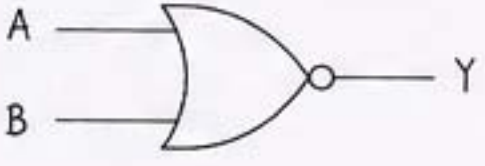
4. NAND Gate



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Boolean Expression : $Y = \overline{A \cdot B}$

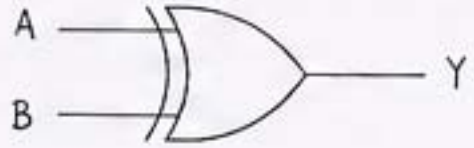
5. NOR Gate



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Expression : $Y = \overline{A + B}$

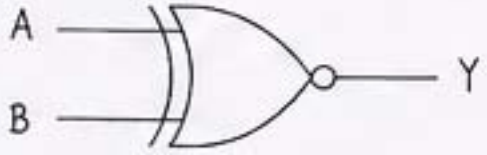
XOR Gate (Exclusive OR)



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Boolean Expression : $Y = A \oplus B = \overline{A}B + A\overline{B}$

XNOR Gate (Exclusive NOR)



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Expression : $Y = A \otimes B = AB + \overline{A}\overline{B}$

Important Points

- NAND and NOR are universal gates.
- Any digital circuit can be implemented using only NAND gates or only NOR gates.

7. Universal Gates in Detail

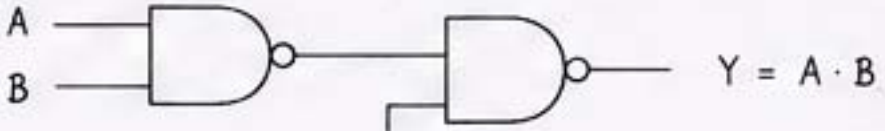
1. NAND Gate as Universal Gate

- Any logic circuit can be implemented using only NAND gates.
- NOT, AND, OR can be realized using NAND gates.

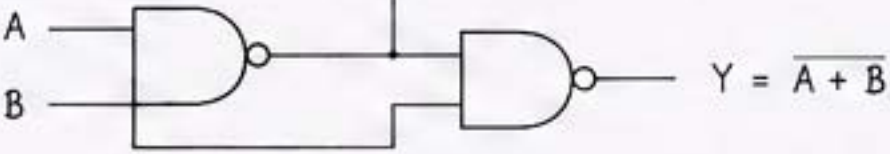
Realization using NAND gate :

NOT Gate :

AND Gate :



OR Gate :



Key Point

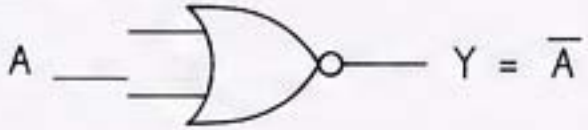
- NAND gate is called "Sheffer Stroke".

2. NOR Gate as Universal Gate

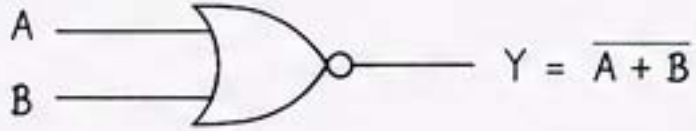
- Any logic circuit can be implemented using only NOR gates.
- NOT, AND, OR can be realized using NOR gates.

Realization using NOR gate :

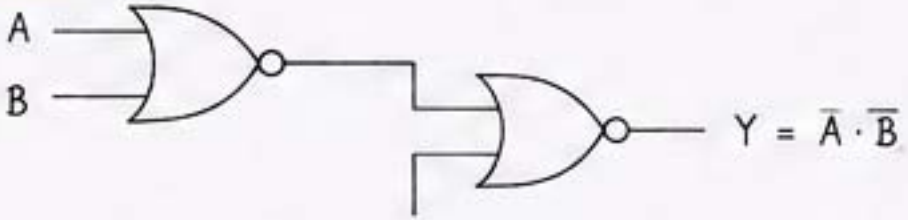
NOT Gate :



OR Gate :



AND Gate :



Key Point

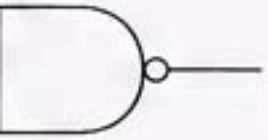
- NOR gate is called "Peirce Arrow".

Bubbled Gates

small circle (bubble) at the output indicates complement.

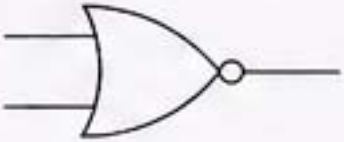
bubble at the input indicates the variable is complemented.

Bubbled AND (NAND Gate)



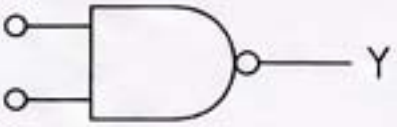
B	$Y = \overline{A \cdot B}$
0	1
1	1
0	1
1	0

2. Bubbled OR (NOR Gate)



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

3. Bubbled Inputs (Input Complement)



A	B	$Y = \overline{A} \cdot \overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0

mportant

ubble can be moved across the gate by complementing the input/output.

his is useful in simplifying circuits.

9. Boolean Algebra – Basic Laws

1. **Commutative Laws**

   $A + B = B + A$, $A \cdot B = B \cdot A$

2. **Associative Laws**

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

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

3. **Distributive Laws**

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

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

4. **Identity Laws**

   $A + 0 = A$, $A \cdot 1 = A$

5. **Null Laws**

   $A + 1 = 1$, $A \cdot 0 = 0$

6. **Complement Laws**

   $A + \overline{A} = 1$, $A \cdot \overline{A} = 0$

7. **Idempotent Laws**

   $A + A = A$, $A \cdot A = A$

Useful Theorems

(i) $\overline{\overline{A}} = A$

(ii) $\overline{A + B} = \overline{A} \cdot \overline{B}$

(iii) $\overline{A \cdot B} = \overline{A} + \overline{B}$

(iv) $A + A \cdot B = A$

(v) $A \cdot (A + B) = A$

Note

These laws help in simplifying Boolean expressions and designing digital circuits.

10. Boolean Expression & Simplification

- Boolean Expression is an algebraic expression that uses variables, logic operations $(+, \cdot, \overline{\cdot})$.

  Example : $Y = A \cdot \overline{B} + \overline{A} \cdot B + A \cdot B$

Steps for Simplification :

1. Apply Boolean Algebra laws.
2. Combine similar terms.
3. Remove redundant terms.
4. Obtain simplest form.

Example : Simplify $Y = A \cdot B + A \cdot \overline{B}$

Solution :

$$Y = A \cdot B + A \cdot \overline{B}$$
$$= A \cdot (B + \overline{B}) \quad \text{[Distributive Law]}$$
$$= A \cdot 1 \quad \text{[Complement Law]}$$
$$= A$$

Important

- Simplification reduces the number of gates.
- It reduces cost, power consumption and delay.

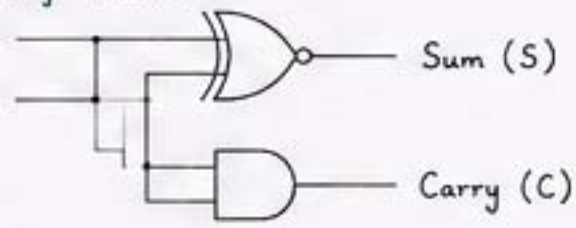
Combinational Logic Circuits

The output of a combinational circuit depends only on present inputs.

There is no memory element.

mples :

Half Adder



A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Full Adder

puts : A, B, Cin (Carry in)

tputs : Sum (S), Carry out (Cout)

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
1	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

y Point

Combinational circuits are building blocks of digital systems.

13. Sequential Circuits

- Sequential circuits are those circuits whose output depends on present input as well as past state.
- They have memory element.
- They are built using flip flops or latches.

Comparison :

Sequential Circuits	Combinational Circuits
Output depends on present input + past state.	Output depends only on present input.
Have memory element.	No memory element.
Use feedback.	No feedback.
More complex.	Less complex.

Examples of Sequential Circuits :

→ Counters → Registers → Shift Registers
→ Flip Flops → Latches → Timers
→ Digital Clock → Sequence Generators

Important : Flip Flops and Latches are the basic building blocks of all sequential circuits.

14. Latches and Flip Flops

- Latches and Flip Flops are bistable multivibrators.
- They can be in one of two stable states.
- They are used to store 1 bit of information.

Difference between Latch and Flip Flop :

Latch	Flip Flop
Level sensitive device.	Edge triggered device.
Output changes as long as input is active.	Output changes only at the triggering edge.
Simple and faster.	More complex.
Used in asynchronous circuits.	Used in synchronous circuits.

Characteristics :

1. Two stable states → 0 and 1
2. Can store 1 bit of data.
3. Complementary outputs (Q and $\overline{Q}$).
4. Built using feedback.

Remember :
Latch → Transparent (level sensitive)
Flip Flop → Edge Triggered (rising or falling edge)

. Types of Flip Flops

p Flops are classified on the basis of inputs.

SR Flip Flop 2. JK Flip Flop 3. D Flip Flop 4. T Flip Flop

nparison Table :

Type	Inputs	Characteristic
R Flip Flop	S (Set), R (Reset)	Simplest flip flop, but has invalid state.
K Flip Flop	J, K	Improved SR flip flop, no invalid state.
Flip Flop	D (Data)	No invalid state, used in registers.
Flip Flop	T (Toggle)	T = 0 → No change T = 1 → Toggle

ath Table Summary (Characteristic Equation)

lip Flop	Characteristic Equation
SR	$Q_{n+1} = S + \overline{R} \cdot Q_n \quad (S \cdot R = 0)$
JK	$Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$
D	$Q_{n+1} = D$
T	$Q_{n+1} = T \oplus Q_n$

Note :

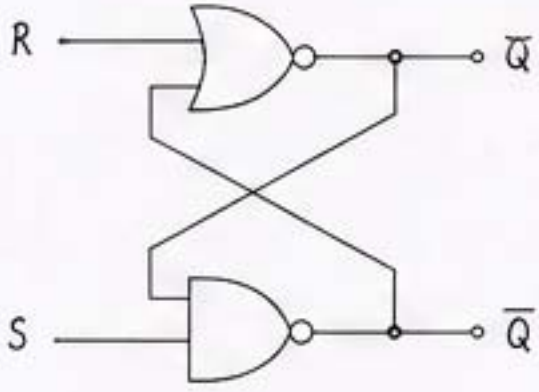
SR flip flop has one invalid state (S = 1, R = 1).

16. SR Flip Flop

It is the simplest flip flop.

It has two inputs : S (Set) and R (Reset).

It can be implemented using cross coupled NOR gates.



SR Flip Flop (NOR based)

S	R	Q_{n+1}	$\overline{Q}_{n+1}$	Operation
0	0	Q_n	$\overline{Q}_n$	No Change
0	1	0	1	Reset
1	0	1	0	Set
1	1	–	–	Invalid

Explanation :

- S = 1, R = 0 → Set → Q = 1
- S = 0, R = 1 → Reset → Q = 0
- S = 0, R = 0 → No change
- S = 1, R = 1 → Invalid (both outputs 0)

Important Points :

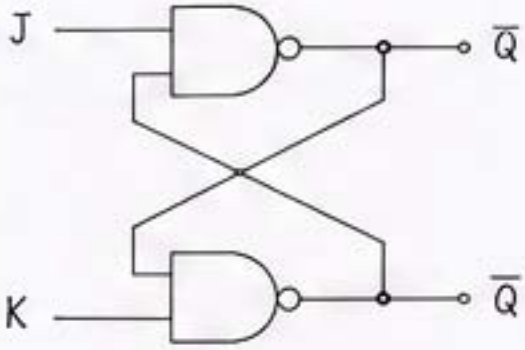
- Avoid invalid state (S=1, R=1).
- NOR implementation is active high.
- Used in simple memory applications.

17. JK Flip Flop

It is an improvement of SR flip flop.

It has no invalid state.

It is implemented using cross coupled NAND gates.



JK Flip Flop

J	K	Q_{n+1}	Operation
0	0	Q_n	No Change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q_n}$	Toggle

Characteristic Equation : $Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$

Key Points :

- J = 1, K = 1 → Toggle (Changes state).
- No invalid state.
- Very useful in counters and control circuits.
- NAND implementation is active low.

Summary of Flip Flops

pe	Inputs	Circuit Used	Special Feature	Characteristic Equation
R	S, R	NOR (active high) or NAND (active low)	Simplest, has invalid state	$Q_{n+1} = S + \overline{R} \cdot Q_n$ $(S \cdot R = 0)$
K	J, K	NAND (active low)	No invalid state, toggle operation	$Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$
	D	NAND	No invalid state, used in registers	$Q_{n+1} = D$
	T	NAND	Toggle when T = 1	$Q_{n+1} = T \oplus Q_n$

xam Tip (DSSSB TGT CS) :

Know truth table and characteristic equation of all flip flops.

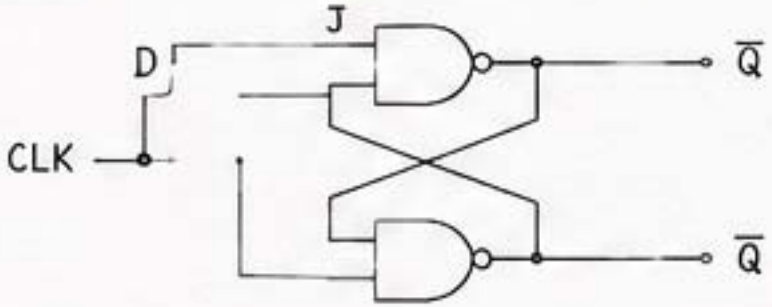
Remember invalid state of SR flip flop.

Understand difference between level triggered latch and edge triggered flip flop.

Practice conversion of one flip flop to another.

19. D Flip Flop

- It is a data (delay) flip flop.
- It has single input D (Data).
- Output follows the input at the active edge of clock.
- It has no invalid state.
- Implemented using two cross coupled NAND gates.



Truth Table :

D	Q_n	Q_{n+1}	Operation
0	0	0	Reset
0	1	0	Reset
1	0	1	Set
1	1	1	Set

Characteristic Equation :

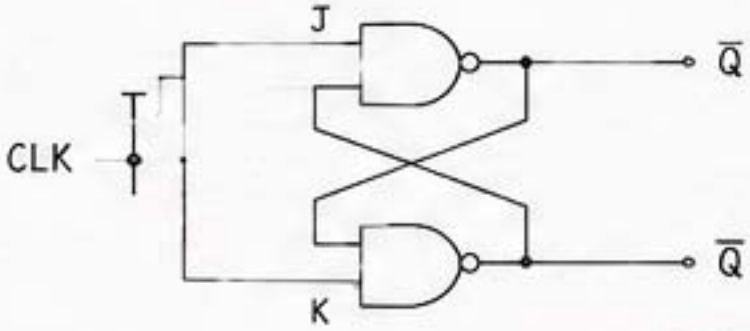
$$Q_{n+1} = D$$

Key Points :

- D = 0 → Reset
- D = 1 → Set
- All invalid states are eliminated.
- Used in registers and shift registers.

20. T Flip Flop

- It is a toggle flip flop.
- It has single input T.
- T = 0 → No change in state.
- T = 1 → Toggle the state (0 ↔ 1).
- Implemented using JK flip flop by connecting J = K = T.



Truth Table :

T	Q_n	Q_{n+1}	Operation
0	0	0	No Change
0	1	1	No Change
1	0	1	Toggle
1	1	0	Toggle

Characteristic Equation :

$$Q_{n+1} = T \oplus Q_n$$

Key Points :

- T = 0 → Hold
- T = 1 → Toggle
- Used in counters and frequency dividers.

Excitation Tables of Flip Flops

xcitation table shows the required inputs for
desired state transition.

R Flip Flop

	Q_{n+1}	S	R
	0	0	X
	1	1	0
	0	0	1
	1	X	0

→ Don't Care

2. JK Flip Flop

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

X → Don't Care

Flip Flop

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

4. T Flip Flop

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

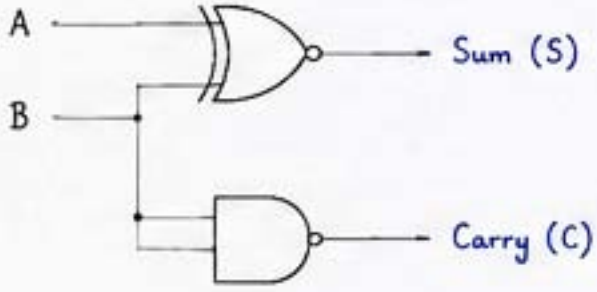
Remember :

Use excitation table to design the required Flip Flop circuit.

22. Half Adder

- A half adder adds two 1-bit numbers.
- It has two inputs (A, B) and two outputs (Sum, Carry).
- It does not consider carry from previous stage.

Logic Diagram :



Truth Table :

A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Boolean Expressions :

$S = A \oplus B$
$C = A \cdot B$

Implementation :

1 XOR Gate for Sum

1 AND Gate for Carry.

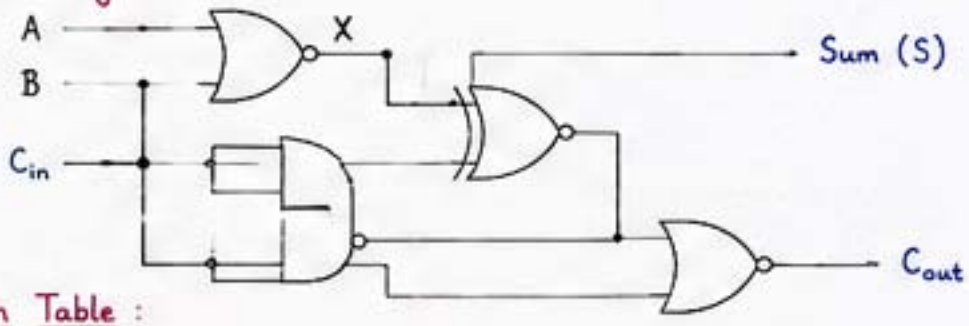
Important :

Half adder is used in LSB stage of an adder.

23. Full Adder

- A full adder adds three 1-bit numbers.
- Inputs : A, B and Carry in (C_{in})
- Outputs : Sum (S) and Carry out (C_{out})

Logic Diagram :



Truth Table :

A	B	C_{in}	Sum (S)	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Boolean Expressions :

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + BC_{in} + AC_{in}$$

Note :

Full adder is used in all stages except LSB.

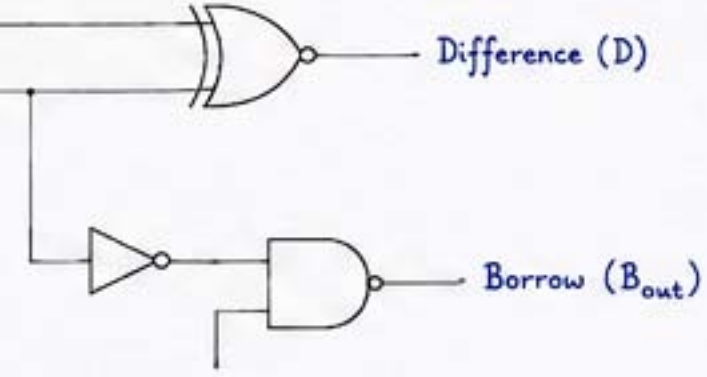
Half Subtractor

half subtractor performs subtraction of two 1-bit numbers.

has two inputs (A : Minuend, B : Subtrahend).

tputs : Difference (D) and Borrow (B_{out})

Diagram :



Truth Table :

A	B	D	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

oolean Expressions :

$$D = A \oplus B$$

$$B_{out} = \overline{A} \cdot B$$

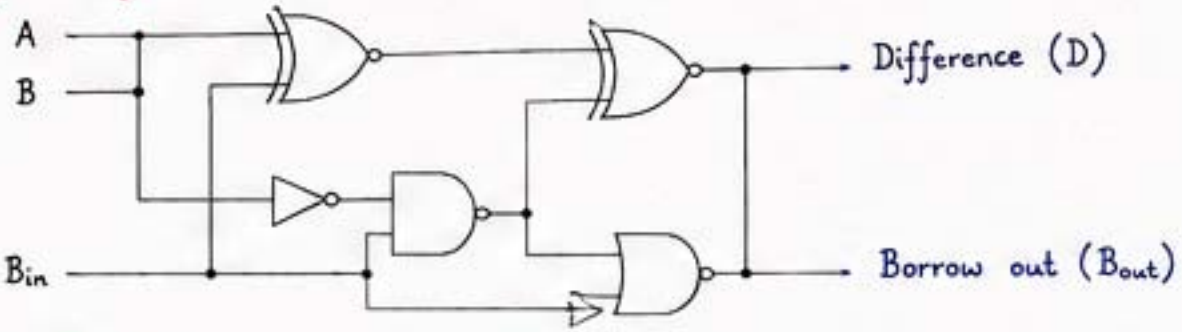
portant :

row occurs only when A = 0 and B = 1.

25. Full Subtractor

- A full subtractor subtracts three 1-bit numbers.
- Inputs : A (Minuend), B (Subtrahend), B_{in} (Borrow in)
- Outputs : Difference (D), Borrow out (B_{out})

Logic Diagram :



Truth Table :

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Boolean Expressions :

$$D = A \oplus B \oplus \overline{B_{in}}$$

$$B_{out} = \overline{A}\,\overline{B} + \overline{A}\,\overline{B_{in}} + B\,\overline{B_{in}}$$

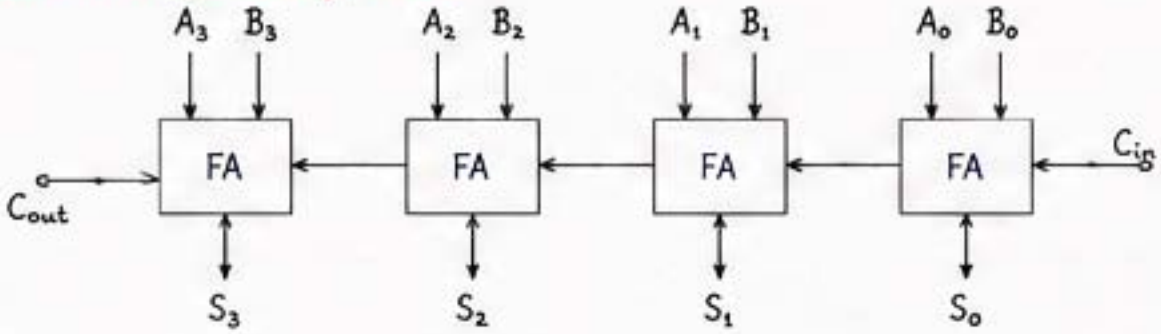
Important :

Full subtractor is used in subtraction of multi-bit numbers.

26. Ripple Carry Adder (RCA)

- It is a combinational circuit that adds multi-bit numbers.
- It is formed by cascading (n) full adders.
- Carry out of one stage becomes carry in of next stage.

4-Bit Ripple Carry Adder :



Truth Table (1-bit Full Adder) :

A	B	C_{in}	Sum (S)	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Features :

- Simple to design.
- Uses n full adders for n-bit addition.
- Carry ripples from LSB to MSB.

Note :
RCA is widely used in ALU of computers.

7. Delay in Ripple Carry Adder

- In RCA, carry has to propagate through all stages.
- Therefore, delay increases with number of bits.

Delay Analysis :

Let,

t_{FA} = Delay of one full adder

t_c = Carry propagation delay

t_s = Sum (XOR) delay

For n-bit RCA,

$$T_{pd} = (n-1)\,t_c + t_s$$

Hence, delay $\propto n$ (Linearly proportional to number of bits)

Important for DSSSB TGT CS

★ RCA is easy but slow for large number of bits.

★ To reduce delay, advanced adders like Carry Look Ahead Adder are used.

Example :

For 4-bit adder,

$$T_{pd} = 3t_c + t_s$$

For 8-bit adder,

$$T_{pd} = 7t_c + t_s$$

Key Point :

Delay of RCA increases linearly with number of bits.

28. Carry Look Ahead Adder (CLA)

- CLA reduces the delay by computing carry in advance.
- It uses Generate (G) and Propagate (P) signals.

Definitions :

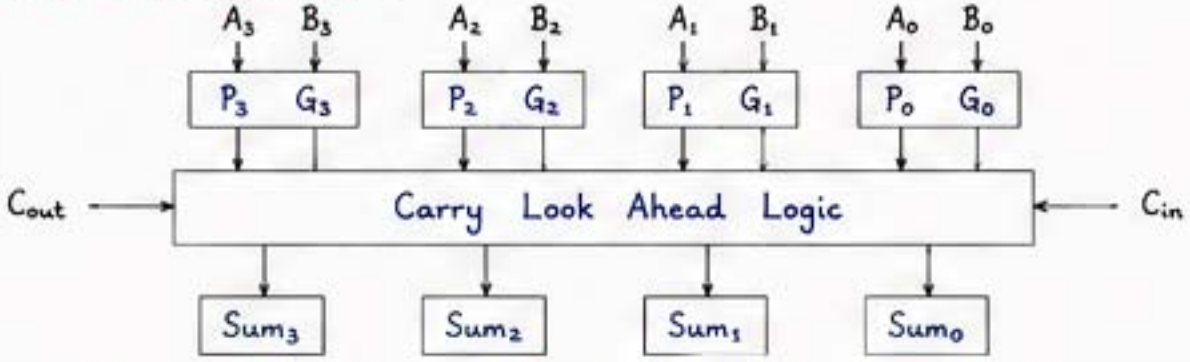
Propagate : $P_i = A_i \oplus B_i$ (If P = 1, carry will be propagated)

Generate : $G_i = A_i \cdot B_i$. (If G = 1, carry will be generated)

Carry Equations :

$$C_{i+1} = G_i + P_i \cdot C_i$$

4-Bit CLA Structures :



Advantages :

- Faster than RCA.
- Carry is calculated in parallel.
- Used in high speed ALU and processors.

Key Point :

CLA reduces delay to a logarithmic or constant level (independent of n).

29. Comparison of Adders

Parameter	Ripple Carry Adder (RCA)	Carry Look Ahead Adder (CLA)
Technique	Carry ripples from LSB to MSB	Carry computed in advance using P and G
No. of Full Adders	n	n (plus look ahead logic)
Speed	Slow (Delay $\propto n$)	Fast (Delay $\approx$ constant / log n)
Hardware	Less hardware	More hardware
Complexity	Simple	More complex
Power	Less	More
Used In	General purpose, small systems	High speed ALU, Microprocessors

Summary :

→ RCA is simple and economical but slow.

→ CLA is fast but hardware is more.

→ For high speed applications CLA is preferred.

30. Summary of Combinational Circuits (So Far)

Circuit	Inputs	Outputs	Boolean Expressions	Used For
Half Adder	A, B	Sum, Carry	$S = A \oplus B$ $C = \bar{A} \cdot B$	Addition of two 1-bit numbers
Full Adder	A, B, C_{in}	Sum, C_{out}	$S = A \oplus B \oplus C_{in}$ $C_{out} = AB + BC_{in} + AC_{in}$	Addition of three 1-bit numbers
alf Subtractor	A, B	D, B_{out}	$D = A \oplus B$ $B_{out} = \bar{A} \cdot \bar{B}$	Subtraction of tw 1-bit numbers
ull Subtractor	A, B, B_{in}	D, B_{out}	$D = A \oplus B \oplus B_{in}$ $B_{out} = \bar{A}B + \bar{A}B_{in} + BB_{in}$	Subtraction of thr 1-bit numbers
CA (n-bit)	$A_{n-1..0}$, $B_{n-1..0}$, C_{in}	$S_{n-1..0}$, C_{out}	Slow (Delay $\propto$ n)	General addition
LA (n-bit)	$A_{n-1..0}$, $B_{n-1..0}$, C_{in}	$S_{n-1..0}$, C_{out}	Fast (Delay $\approx$ constant)	High speed additi

Exam Tip (DSSSB TGT CS) :

- ★ Learn truth tables and characteristic equations of all circuits.
- ★ Understand the difference between ripple carry and look ahead adders.
- ★ Practice problems on adders and subtractors.

31. Minimization of Boolean Expressions using K-Maps (Karnaugh Maps)

- K-map is a graphical method used to minimize Boolean expressions.
- It provides simplified SOP (Sum of Products) or POS (Product of Sums) form.
- It helps to reduce the number of gates, cost and power consumption.
- K-map is applicable for up to 4 variables (16 cells).

Advantages :

→ Simple and systematic
→ Fewer steps than algebraic method
→ Easy to handle don't care conditions
→ Gives minimal expression directly

Important Points :

1. Adjacent cells differ in only one variable.
2. Corners are adjacent to each other.
3. Group sizes must be 1, 2, 4, 8, 16 ... (power of 2).
4. Use as large groups as possible.
5. Groups may overlap but no cell can be left unused.

32. K-Map for Two Variables

1. K-Map Structure :

A \ B	0	1
0	m_0	m_1
1	m_2	m_3

Cell Numbering :

$m_0 = A'B'$

$m_1 = A'B$

$m_2 = AB'$

$m_3 = AB$

2. Example : Simplify $F(A, B) = \Sigma m(0, 2, 3)$

Step 1 : Place 1's in K-map

A \ B	0	1
0	1	0
1	1	1

Step 2 : Make groups

A \ B	0	1
0	1	0
1	1	1

Groups :

- $(m_0, m_1) \rightarrow B'$
- $(m_2, m_3) \rightarrow A$

Simplified Expression :

$$F = \overline{B'} + A$$

Tip : Adjacent cells are vertically or horizontally. Corners are also adjacent.

K-Map for Three Variables

K-Map Structure :

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Cell Numbering (Decimal to Binary) :

$m_0 = 000 = A'B'C'$

$m_1 = 001 = A'B'C$

$m_2 = 010 = A'BC'$

$m_3 = 011 = A'BC$

$m_4 = 100 = AB'C'$

$m_5 = 101 = AB'C$

$m_6 = 110 = ABC'$

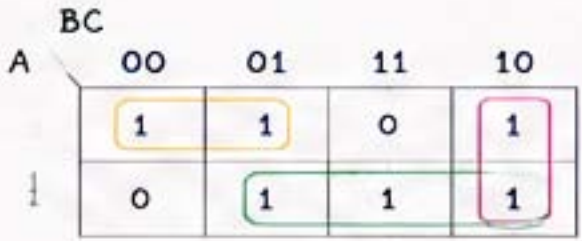
$m_7 = 111 = ABC$

Example : Simplify $F(A, B, C) = \Sigma m(0, 1, 2, 5, 6, 7)$

Step 1 : Fill K-map

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

Step 2 : Make groups



Groups :

$(m_0, m_1) \rightarrow A'B'$

$(m_2, m_6) \rightarrow BC'$

$(m_5, m_7) \rightarrow AC$

Simplified Expressions :

$$F = A'\overline{B'} + BC' + AC$$

Remember : Use largest possible groups to get minimum terms.

34. K-Map for Four Variables

1. K-Map Structure :

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

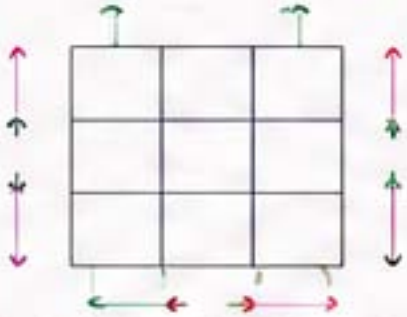
Row & Column Order (Gray Code)

00 , 01 , 11 , 10

2. Important Note :

- Opposite edges are adjacent.
- Corners are adjacent.
- Total cells = 16
- Group size = 1, 2, 4, 8, 16

Quick View of Adjacency



Memory Trick : Think of K-map as a torus (donut).
Left is adjacent to right, top is adjacent to bottom.

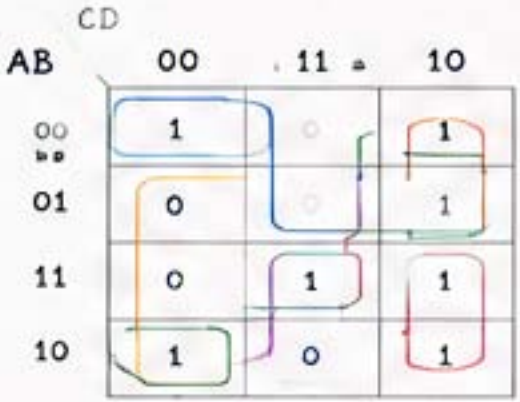
35. Example of 4-Variable K-Map

Example : Simplify $F(A, B, C, D) = \Sigma m(0, 1, 2, 5, 6, 8, 9, 10, 14, 15)$

Step 1 : Fill K-map

AB \ CD	00	11	10
00	1	0	1
01	0	0	1
11	0	0	1
10	1	0	1

Step 2 : Make groups



Groups Formed :

1. $(m_0, m_1, m_8, m_9) \rightarrow B'D'$
2. $(m_2, m_6, m_{10}, m_{14}) \rightarrow BD'$
3. $(m_5, m_9) \rightarrow A'CD$
4. $(m_{14}, m_{15}) \rightarrow ABC$

Simplified Expressions :

$$F = B'\overline{D'} + BD' + A'CD + ABC$$

Tip : Try different groupings to get minimum number of terms. Above expression has 4 terms which is minimal.

Don't Care Conditions in K-Map

Don't care terms (X) can take value 0 or 1 whichever helps o minimize the expression.

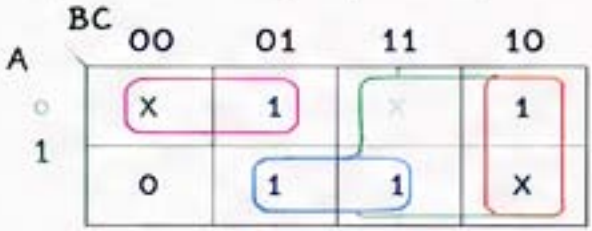
They are used to make larger groups.

ple : Simplify $F(A, B, C) = \Sigma m(1, 2, 5, 7)$, $d = \Sigma d(0, 3, 6)$

1 : Fill K-map (X for don't care)

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

Step 2 : Make groups using X



ps :

- (m_0, m_1) → A'B' (using X at m_0)
- (m_1, m_5) → BC'
- (m_5, m_7) → A'D (using X at m_5)

Simplified Expressions :

$$F = A'\overline{B'} + BC' + A'C$$

ey Point :

se don't care terms to form the largest possible groups.

. leads to minimum gates and simpler circuits.

37. Grouping in K-Maps

- Groups (loops) are formed to eliminate variables.
- A valid group must contain only 1's (or X's).
- Group size must be 1, 2, 4, 8, 16 ... (power of 2).
- Make largest possible groups first.

1. Types of Groups :

1 Cell	2 Cells	4 Cells	8 Cells	16 Cells
1	1 1	1 1 1 1	1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1

2. Rules for Grouping :

- Groups can overlap.
- A cell may be included in more than one group.
- Use don't care (X) to form larger groups.
- After making groups, write the simplified expression.

Important :

Minimum number of literals ⇒ Minimum cost, power and delay.

8. Essential Prime Implicants (EPIs)

A prime implicant that covers at least one 1 which is not covered by any other prime implicant is called Essential Prime Implicant.

EPIs must be included in the final expression.

Example : Minimize $F(A, B, C) = \Sigma m(1, 3, 7)$

Step 1 : K-Map

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

Step 2 : Groups (Prime Implicants)

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

Prime Implicants :

$PI_1 = B'C$

$PI_2 = AC$

Step 3 : Essential Prime Implicants

Prime (01) are essential.

Step 2 : (001) is covered only by B'C

Cell m7 (111) is covered by both.

⇒ Both PI_1 and PI_2 are essential.

Simplified Expression : $F = B'C + AC$

More Examples

mple 1 : $F(A, B, C) = \Sigma m(0, 1, 2, 5, 6, 7)$

1 : K-Map

BC	00	01	11	10
0	1	1	1	1
1	0	1	1	1

Groups and Terms :

G_1 (top row) → A'

G_2 (columns 11 & 10) → B

G_3 (bottom row, 11 & 10) → A

G_4 (column 01) (pairs) → $B'C$

2 : Groups

BC	00	01	11	10
0	0	1	1	1
1	0	1	1	1

Simplified Expressions :

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

$$= A' + B$$

$(A + A' = 1$ absorbed by $A' + B)$

mple 2 : $F(A, B, C, D) = \Sigma m(0, 2, 5, 7, 8, 10, 13, 15)$

wer : $F = \overline{A'D'} + \underline{BD'} + \underline{A'C} + \underline{CD}$

(Try on your own)

40. POS (Product of Sums) using K-Map

- For POS form, group the 0's instead of 1's.
- The groups are used to obtain sum terms.
- The final expression is product of sum terms.

Example : Simplify $F(A, B) = \Pi M(0, 2)$

(Maxterms 0 and 2)

Step 1 : Mark 0's in K-Map

A \ B	0	1
0	0	1
1	0	1

Step 2 : Group 0's

A \ B	0	1
0	0	1
1	0	1

The group covers A' (since A = 0 in group)

POS Expression : $F = (A)$

Note : For POS, use (variable) when cell has 0 and (variable') when cell has 1 in the group.

1. Consensus Theorem & K-Map

.. Consensus Theorem :

$$AB + A'C + BC = AB + A'C$$

(The term BC is redundant and can be removed.)

.. Using K-Map :

Redundant terms appear due to overlapping groups.

Remove the group which is not essential.

Example : $F(A, B, C) = AB + A'C + BC$

Step 1 : K-Map

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

Step 2 : Groups

Group 1 → AB

Group 2 → A'C

Group 3 → BC

(Group 3 is redundant)

Result : By Consensus Theorem, remove BC.

Simplified Expressions : $F = AB + A'C$

Important : K-Map automatically removes redundant terms if groups are selected properly.

Neutral & Self-Dual Functions

Neutral Functions

A function is neutral if it is independent of one or more variables.
It behaves like a function with fewer variables.

mple : $F(A, B, C) = A + B$ (Independent of C)

For $C = 0$ or 1, $\boxed{F = A + B}$

Self-Dual Functions

A function is self-dual if its dual is equal to its complement.
If F is a function, its dual $\overline{F}^d$ is obtained by interchanging + and . , and 0 and 1.

Self-dual if $F^d = F'$

mple : $F(A, B) = A \oplus B$

of F : $F^d = AB + A'B'$

plement of F : $F' = (A \oplus B)' = AB + A'B'$

$\Rightarrow$ F is self-dual.

Properties :

- Parity functions are self-dual.
- NAND and NOR are not self-dual.

Exam Tip (DSSSB TGT CS) :

Learn definitions with examples.
Identify variables on which function is independent.
Practice finding dual and verifying self-duality.

43. Realization of Boolean Expressions

- A Boolean expression can be realized (implemented) using logic gates.
- Steps to realize any Boolean expression :

1. Simplify the expression (using K-map / Boolean algebra).
2. Use basic gates (AND, OR, NOT) to implement.
3. Draw the logic diagram.

Example : Realize $F(A,B,C) = A'B + BC + A'C'$

Step 1 : Simplify using K-map

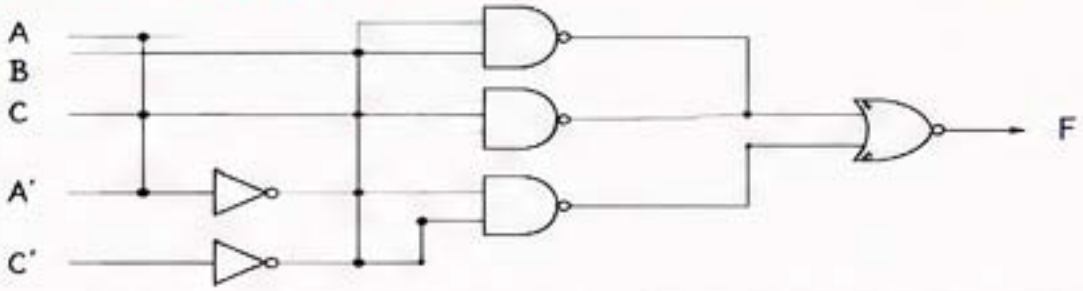
A \ BC	00	01	11	10
0	1	1	1	0
1	1	0	1	0

Groups :

- $(A'B)$ → pair $(m_0, \overline{m_1})$
- (BC) → pair $(m_3, m7)$
- $(A'C')$ → pair $(m_0, \overline{m_4})$

Simplified Expression : $F = A'B + BC + A'C'$

Step 2 : Logic Diagram



4. Don't Care Conditions – More Examples

Example 1 : $F(A,B,C) = \Sigma m(1,2,5)$, $d = \Sigma d(0,3,7)$

Step 1 : K-map (X for don't care)

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

Step 2 : Make groups (use X)

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

Groups formed :

Group-1 : (1,3) → $A'C$

Group-2 : (5,7) → AC

Simplified Expressions :

$$F = \overline{A'C} + AC$$

Example 2 : $F(A,B,C) = \Sigma m(0,2,4,6)$, $d = \Sigma d(1,3,5,7)$

Step 1 : K-map with X

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

Step 2 : Groups (8-cell group)

00	01	11	10
1	X	X	1
1	X	X	1

Simplified Expression : $F = 1$ (All cells covered)

Tip : Don't care terms are used to form larger groups and

5. Hazard in Combinational Circuits

Hazard is a temporary unwanted output change when inputs change.

It occurs due to different propagation delays in different paths.

Types of Hazards :

Static Hazard – Output should remain 1 (or 0) but changes momentarily.

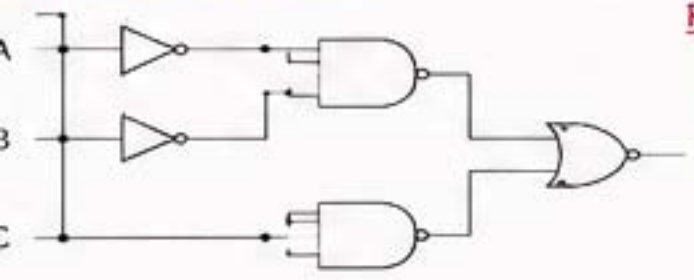
Dynamic Hazard – Output should change once but changes multiple times.

ample – Static-1 Hazard

nction : $F(A,B,C) = A'B + AB'$

hen input changes from (A,B) = (0,1) to (1,0) with C = 1.

rcuit (Two-Level SOP)



Remedy : Add consensus term AB.

New Function :

F

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

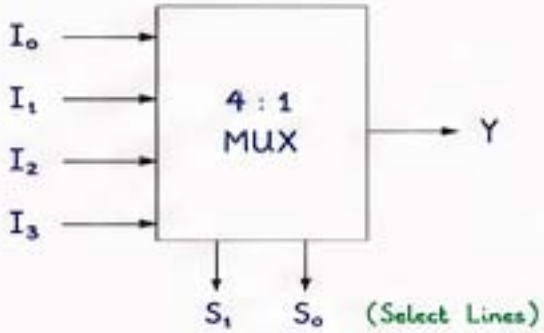
Important for DSSSB TGT CS) :

Hazards occur in multi-level circuits and affect reliability.

46. Multiplexers (Data Selectors)

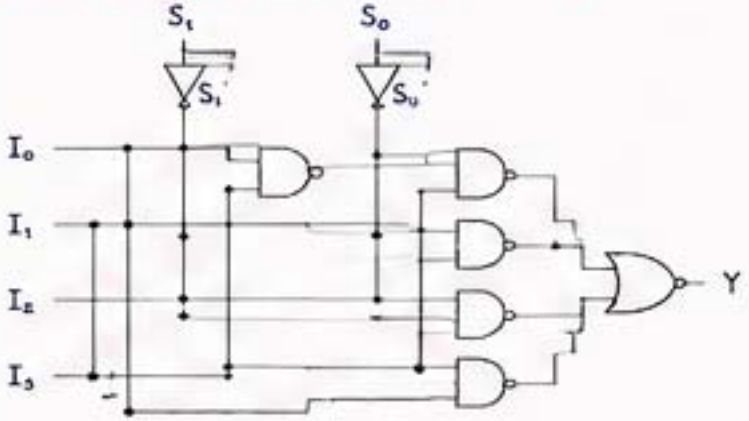
- Multiplexer selects one of many inputs and directs it to the output.
- A $2^n : 1$ MUX has 2^n inputs, n select lines and one output.

4 : 1 Multiplexer



S_1	S_0	Y (Output)
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Logic Diagram of 4 : 1 MUX



Applications :

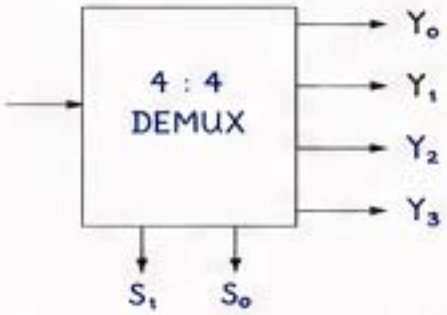
- Data routing
- Communication systems
- Implementing logic functions

7. Demultiplexers (Data Distributors)

Demultiplexer routes one input to one of many outputs.

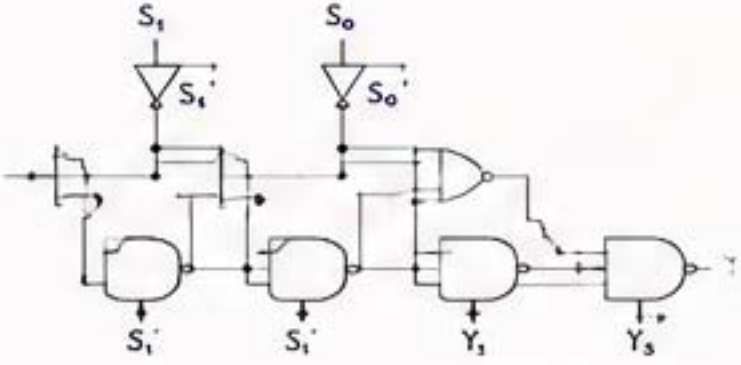
A $1 : 2^n$ DEMUX has one input, n select lines and 2^n outputs.

: 4 Demultiplexer



S_1	S_0	Outputs (only one is 1)
0	0	$Y_0 = D,\ Y_1 = Y_2 = Y_3 = 0$
0	1	$Y_1 = D,\ Y_0 = Y_2 = Y_3 = 0$
1	0	$Y_2 = D,\ Y_0 = Y_1 = Y_3 = 0$
1	1	$Y_3 = D,\ Y_0 = Y_1 = Y_2 = 0$

gic Diagram of 1 : 4 (DEMUX)



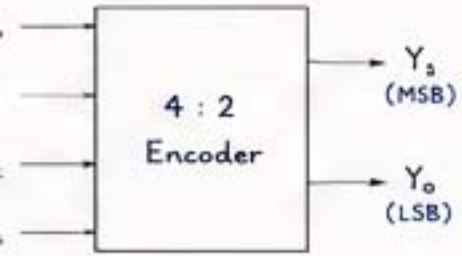
Applications :

- Data distribution
- Memory decoding
- Communication

Note : MUX → Many to One DEMUX → One to Many

3. Encoders and Decoders

: 2 Encoder

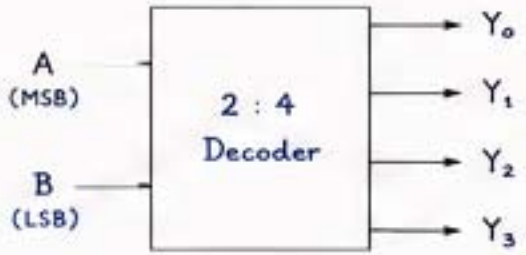


th Table (Assume one input is 1 at a time)

D_3	D_2	D_1	D_0	Y_1	Y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

cations : Keyboard encoder, Priority encoder, Data compression, Interrupt control

2 : 4 Decoder



Truth Table

A	B	Y_0	Y_1	Y_2	Y_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

Applications : Memory decoding, Display decoding, Address decoding

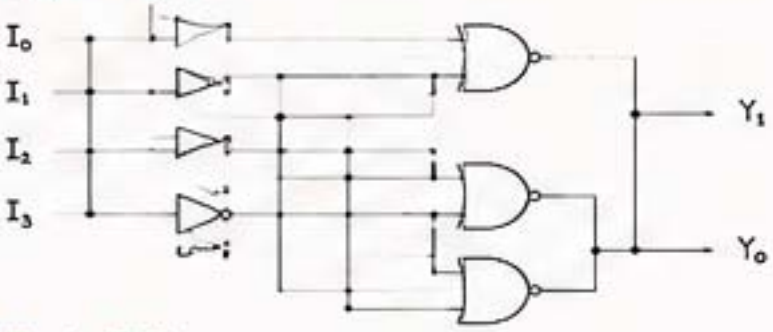
Encoders → Many to Few Decoders → Few to Many

49. Priority Encoders

- In normal encoder, if more than one input is 1 at a time, the output combination is invalid.
- Priority encoder assigns a fixed priority to inputs.
- The input with highest priority (highest order) is encoded.

Example : 4-to- 2 Priority Encoder (I_3 highest priority)

Logic Diagram :



Priority Order :

$I_3 > I_2 > I_1 > I_0$

(Higher index = Higher priority)

Truth Table :

I_3	I_2	I_1	I_0	Y_1	Y_0	Valid Output
0	0	0	0	X	X	No input is 1
0	0	0	1	0	0	I_0
0	0	1	0	0	1	I_1
0	1	0	0	1	0	I_2
1	0	0	0	1	1	I_3
1	x	x	x	1	1	I_3 (Highest priority)

Note :

x → Don't care

X → Invalid

0. Code Converters

Binary to Gray Code Converter

Consecutive Gray codes differ in only one bit.

Used to reduce errors in switching circuits.

ruth Table :

Binary (A B C)			Gray (G_2 G_1 G_0)		
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

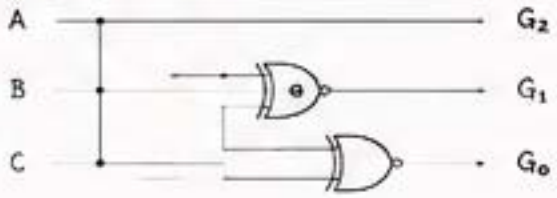
Logic Equations :

$$G_2 = A$$

$$G_1 = A \oplus B$$

$$G_0 = B \oplus C$$

Logic Diagram :



Gray to Binary Code Converter

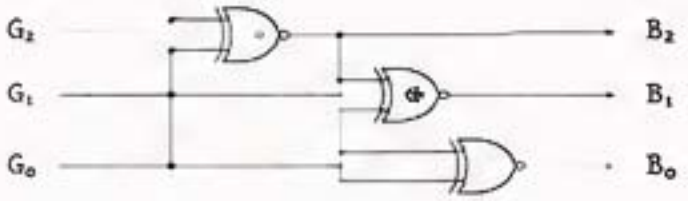
gic Equations :

$$B_2 = G_2$$

$$B_1 = G_2 \oplus G_1$$

$$B_0 = G_2 \oplus G_1 \oplus G_0$$

Logic Diagram :



Remember : Gray code is not weighted. It is a cyclic code.

51.

1. B
- P
- I
- I

Logic

D_0 —
D_1
D_2 —
D_3 —

2. B
- C
  is
- O
- O

Logic

D_0
D_1
D_2 —
D_3 —

N
-
-

Parity Generators and Checkers

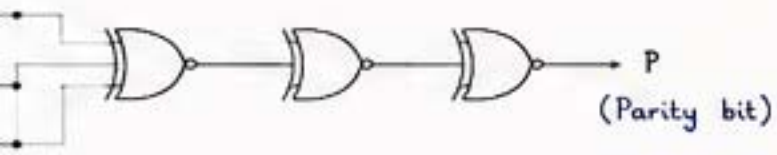
n Parity Generator (4-bit)

ty bit (P) is added so that total number of 1's is even.

number of 1's in data is even → P = 0

number of 1's in data is odd → P = 1

iagram :



Equation :

$$P = D_0 \oplus D_1 \oplus D_2 \oplus D_3$$

n Parity Checker (5-bit)

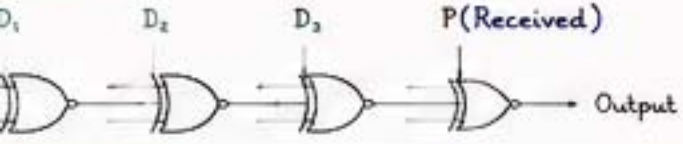
ks whether total number of 1's (including parity bit)

ven or not.

ut = 1 → Even parity (No error)

ut = 0 → Odd parity (Error)

iagram :



Equation :

$$\text{Output} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus P$$

e :

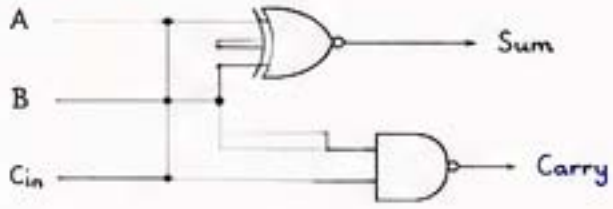
dd parity is similar (just invert the parity bit).

rity is used for error detection (not correction).

52. Adders - Carry Save Adder (CSA)

- Used in fast arithmetic circuits like multipliers.
- Adds three numbers of n-bits and gives Sum and Carry.
- Sum and Carry are calculated without waiting for carry propagation.

Logic Diagram (1-bit) :



Equations :

$$Sum = A \oplus B \oplus C_{in}$$

$$Carry = \overline{AB} + AC_{in} + BC_{in}$$

Truth Table (1-bit) :

A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Applications :

- Used in multipliers
- High speed adders
- Digital signal processing

Note : CSA does not propagate carry to next stage. It saves time in partial product addition.

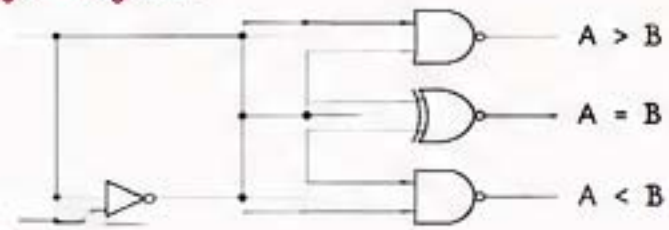
3. Comparators

Compares two binary numbers.

Outputs indicate whether A > B, A = B or A < B.

1-bit Comparator

Logic Diagram :



Equations :

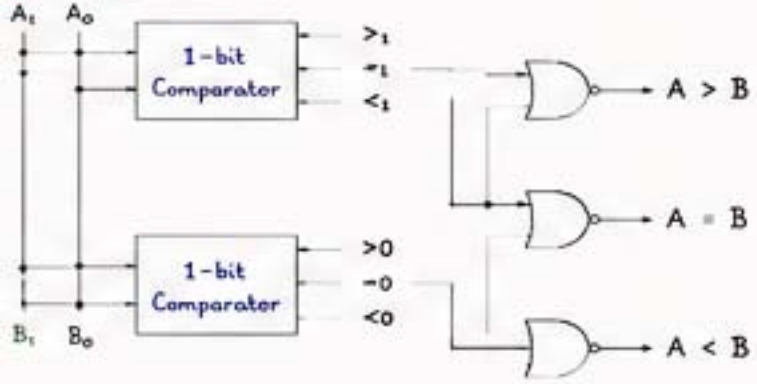
$$A > B = A\,\overline{B}$$

$$A = B = A \odot B$$

$$A < B = \overline{A}\,B$$

2-bit Comparator

Logic Diagram :



Note :

- Compare MSB first.
- If MSBs are equal, compare next bits.
- Can be extended to n-bits.

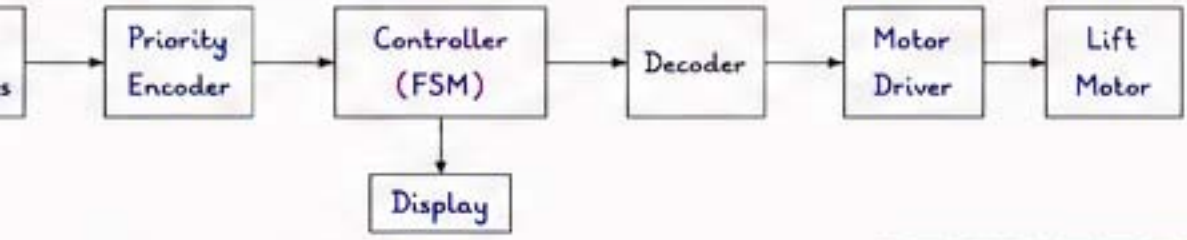
Used in digital systems, decision making, ALU, sorting etc.

Lifts, Barriers and Basic Digital Systems

: (Elevator) Control System

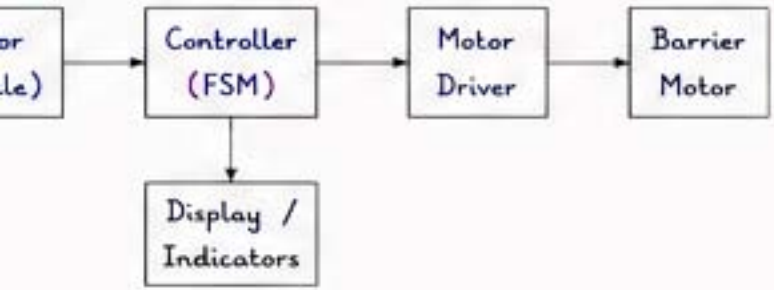
s sequence counters and decoders.

Diagram :



omatic Barrier System

iagram :



Digital systems use :

- Logic gates
- Flip-flops
- Counters
- Decoders
- Timers

tions :

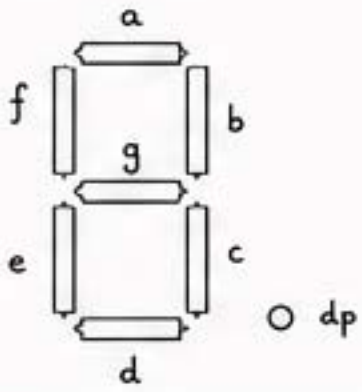
- ic lights
- trial automation
- Ticket vending machines
- Security systems

Digital Electronics is the foundation of all modern electronic systems.

55. Seven Segment Display

- A seven segment display has 7 LEDs (a to g) used to display decimal digits 0 to 9.
- It can be used in Common Anode (CA) or Common Cathode (CC) configurations.

Segment Diagram :



7-Segment Truth Table (Common Cathode)

Digit	Segments (1 = ON, 0 = OFF)					
	a	b c	c	e	f	g
0	1	1	1	1	1	0
1	0	1	1	0	0	0
2	1	1	0	1	0	1
3	1	1	1	1	0	1
4	0	1	1	0	1	1
5	1	0	1	0	1	1
6	0	0	1	1	1	1
7	1	1	1	0	0	0
8	1	1	1	1	1	1
9	1	1	1	1	0	1

Note :

- For Common Anode, 1 = OFF, 0 = ON.
- Use current limiting resistor with each segment.

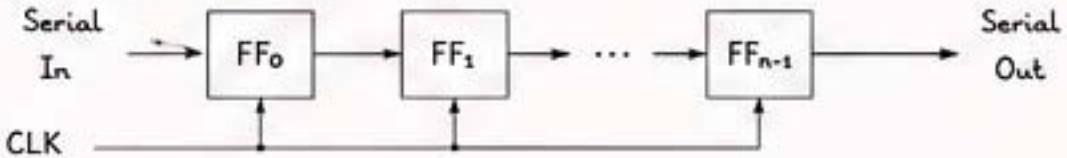
Applications :

- Digital clocks, Counters, Calculators, Measuring instruments, Elevator floor indicators, etc.

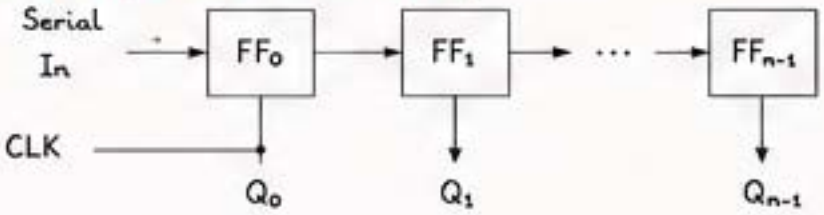
56. Registers

- A register is a group of flip-flops used to store binary information.
- All flip-flops share a common clock.
- Types - SISO, SIPO, PISO, PIPO (Parallel In Parallel Out)

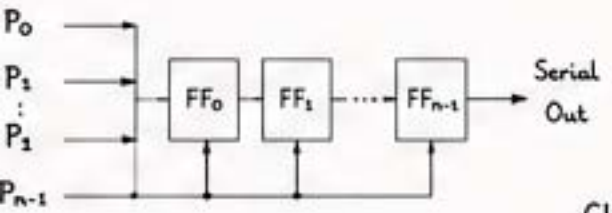
1. SISO (Serial In Serial Out)



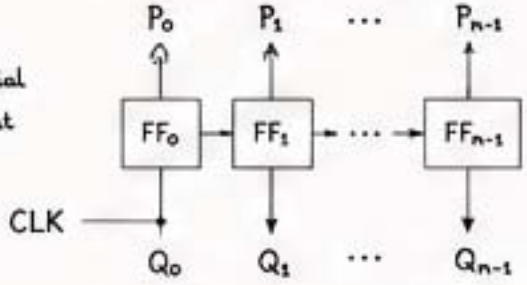
2. SIPO (Serial In Parallel Out)



3. PISO (Parallel In Serial Out)



4. PIPO (Parallel In Parallel Out)



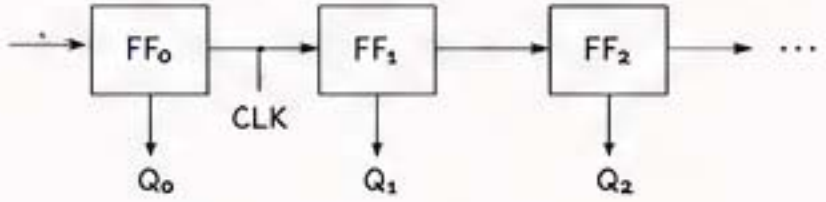
Note : Registers are basic building blocks of memory and digital systems.

Counters

counter is a sequential circuit that goes through a fixed sequence
states.
is built using flip-flops.
:

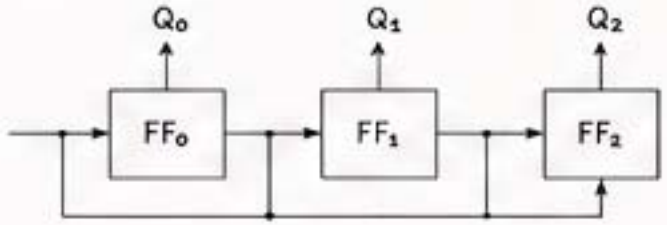
synchronous (Ripple) Counter
ynchronous Counter

synchronous (Ripple) Counter



> **Note** :
> - Simple but has accumulated propagation delay.

ynchronous Counter



> **Note** :
> - All flip-flops receive clock simultaneously.
> - Faster and more reliable.

cations :

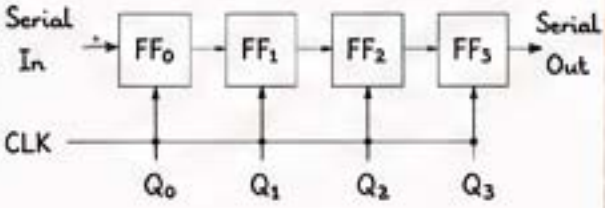
equency division, Timing, Digital clocks, Event counting,
quence generations, etc.

58. Shift Registers & Universal Shift Register

1. Shift Register

- Shifts data left or right by one position on each clock pulse.
- Used for data transfer, delay, conversion, etc.

4-bit Shift Register (Right Shift)



Note :

- On each clock pulse, data moves one flip-flop to the right.
- Left shift is similar in opposite direction.

2. Universal Shift Register (4-bit)

- Can perform – Hold, Shift Right, Shift Left, and Parallel Load.
- Uses control inputs S_1 and S_0.
- Built using D Flip-Flops.

Mode Selection Table :

S_1	S_0	Mode	Operation
0	0	Hold	No change
0	1	Shift Right	Move right
1	0	Shift Left	Move left
1	1	Parallel Load	Load data

Applications :

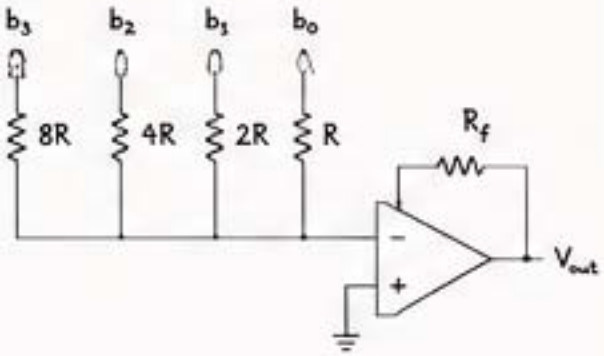
- Data transfer, Serial to parallel conversion, Buffer storage, CPU data handling, Communication systems.

59. DAC and ADC

1. DAC (Digital to Analog Converter)

- Converts digital (binary) data into analog output voltage/current.
- Types - Weighted Resistor DAC, R-2R Ladder DAC, etc.

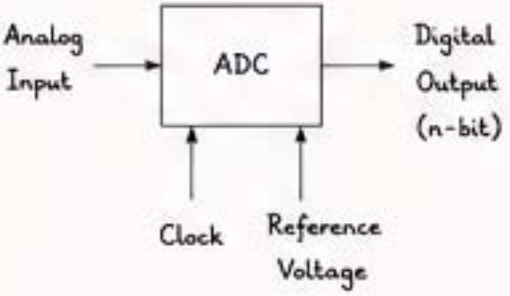
Basic 4-bit DAC (Weighted Resistor)



2. ADC (Analog to Digital Converter)

- Converts analog input voltage into digital (binary) output.
- Types - Flash ADC, SAR ADC, Dual Slope ADC, etc.

Basic ADC Block Diagram



Equation : $V_{out} = -V_{ref}\left(\frac{b_3}{2^3} + \frac{b_2}{2^2} + \frac{b_1}{2^1} + \frac{b_0}{2^0}\right)$

Where $b_i = 0$ or 1

Applications :

- DAC - Audio systems, Signal generators, Motor control
- ADC - Data acquisition, Sensors, Communication systems.

Summary of Important Digital Electronics Topics

Topic	Key Points
gic Gates	Basic, Universal, Bubbled gates – building blocks of digital circuits.
ential Circuits	Latches, Flip-flops (SR, JK, D, T), Registers, Counters.
inational Circuits	Adders, Subtractors, Multiplexers, Demultiplexers, Comparators, Encoders, Decoders.
mization	K-Maps, Don't care conditions, Grouping, EPIs, POS, SOP.
metic Circuits	RCA, CSA, CLA – used for fast arithmetic operations.
Converters	Binary-Gray, Gray-Binary, Parity generators & checkers.
y & Indicators	7-Segment Display, LED, Indicators.
cations	Control systems, Digital systems, Communication, Automation, Measuring instruments, Computers.

xam Tips (DSSSB TGT CS)

- Learn truth tables and characteristic/excitation tables of flip-flops.
- Practice K-map problems (3- & 4-variable) regularly.
- Understand differences (RCA vs CLA, MUX vs DEMUX, Encoder vs Decoder).
- Solve previous year questions and time yourself.

61. Practice Set – 1 (Basic Logic & Boolean Algebra)

Q1. Simplify the following Boolean expressions using Boolean algebra :

(a) $A + A'B$

(b) $(A + B)(A' + C)$

(c) $A'B + AB' + BC + B'C$

(d) $(A + B + C)(A' + B)$

(e) $(A' + B')(A + B + C)$

Recall :

$A + A = A, \quad A + A' = 1$

$AA = A, \quad A\,A' = 0$

$A + 0 = A, \quad A \cdot 1 = A$

$(A + B)' = A' \cdot B'$

$(A \cdot B)' = A' + B'$

Q2. Using K-map, minimize the following functions :

(a) $F(A, B, C) = \Sigma m(1, 3, 5, 7)$

(b) $F(A, B, C) = \Sigma m(0, 2, 4, 6) + d(1, 3, 5, 7)$

(c) $F(A, B, C, D) = \Sigma m(0, 1, 2, 5, 6, 8, 9, 13) + d(10, 14)$

Q3. Convert the following to POS form using K-map :

$F(A, B, C) = \Sigma m(0, 2, 3, 5, 6, 7)$

Q4. State Dual of the expression : $F = A'B + AC' + BC$

Q5. Simplify using Boolean algebra and verify using truth table :

$F = (A + B)'(A + B) + (A \cdot B)'$

Marks Distribution : Q1 (5 marks)) Q2 (6 marks) Q3 (3 marks)
Q4 (1 mark) Q5 (5 marks) Total = 20 Marks

62. Practice Set – 2 (Combinational Circuits)

Q1. Design Half Adder using logic gates. Write its truth table.

Q2. Design Full Adder using two Half Adders. Draw the circuit and write truth table.

Q3. Design Full Subtractor using logic gates and verify with truth table.

Q4. A 4-bit binary number A = 1011 and B = 0110.

(a) Add A and B using Ripple Carry Adder. Show all steps.

(b) What is the result using Carry Look Ahead Adder?

Q5. Draw the logic diagram of 4-bit Carry Look Ahead Adder.

Q6. Compare Ripple Carry Adder and Carry Look Ahead Adder on the basis of delay, hardware and speed.

Q7. Implement the following expression using basic gates :

$F(A, B, C, D) = \Sigma m(0, 1, 2, 5, 6, 90, 10, 14)$

Answer Hints (for Self Practice)

HA : Sum = $A \oplus B$, Carry = $A.B$

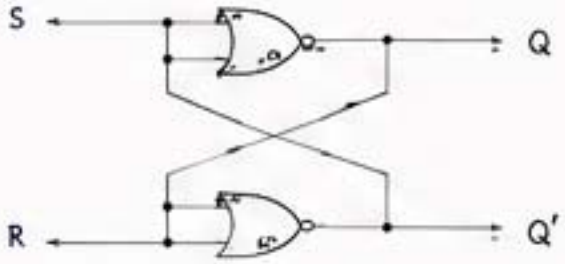
FA : Sum = $A \oplus B \oplus Cin$, Cout = $AB + BCin + ACin$

FS : $D = A \oplus B \oplus Bin$, Bout = $(A'B) + (Bin(A \oplus B)')$

Marks Distribution : Each Q = 2 to 3 marks Total = 20 Marks

63. Practice Set – 3 (Sequential Circuits – Flip Flops)

Q1. Draw the circuit diagram of SR Flip-Flop using NOR gates. Derive its characteristic table.



Q2. Draw the circuit diagram of JK Flip-Flop using NAND gates. Derive its characteristic table.

Q3. Show that JK Flip-Flop works as Toggle Flip-Flop when J = K = 1.

Q4. Draw the circuit diagram of D Flip-Flop using NAND gates. Derive its characteristic table.

Q5. Draw the circuit diagram of T Flip-Flop using JK Flip-Flop. Derive its characteristic table.

Q6. Differentiate between SR, JK, D and T Flip-Flops.

Marks Distribution : Each Q = 3 to 4 marks

Total = 20 Marks

64. Practice Set - 4 (K-Maps & Logic Minimization)

Q1. Minimize the following functions using K-map :

(a) $F(A, B, C, D) = \Sigma m(0, 1, 2, 5, 6, 8, 9, 13, 14, 15) + d(3, 7, 11)$

(b) $F(A, B, C) = \Pi M(0, 1, 3, 4, 6) + d(2, 5)$

Q2. Find the essential prime implicants of :

$F(A, B, C, D) = \Sigma m(1, 3, 7, 11, 15)$

Q3. Using K-map, minimize to minimum SOP and minimum POS :

$F(A, B, C, D) = \Sigma m(0, 2, 5, 6, 8, 9, 10, 14)$

Q4. What are the advantages of using K-map over algebraic methods?

Q5. Simplify using K-map (with don't care conditions) :

$F(A, B, C, D) = \Sigma m(1, 2, 4, 7, 8, 10, 11, 15) + d(0, 3, 5, 12, 13, 14)$

Important :

- ★ Group largest 1's (or 0's for POS).
- ★ Use don't care (X) to form bigger groups.
- ★ Number of groups should be minimum.

AB \ CD	00	01	11	10
00	1	1	X	0
01	1	X	1	0
11	0	1	1	X
10	0	X	1	1

Marks Distribution : Each Q = 4 Marks Total = 20 Marks

65. Practice Set – 5 (MUX, DEMUX, Encoders & Decoders)

Q1. Implement the function $F(A, B, C) = \Sigma m(1, 2, 5, 6)$ using 4 : 1 Multiplexer. Show the data inputs and select lines.

Q2. Implement the function $F(W, X, Y, Z) = \Sigma m(0, 1, 2, 5, 9, 13)$ using 8 : 1 Multiplexer.

Q3. Explain with diagram how 1 : 4 Demultiplexer works.

Q4. Realize the function $F(A, B, C, D) = \Sigma m(2, 4, 6, 8, 10, 12, 14)$ using 1 : 4 Demultiplexer.

Q5. Draw the logic diagram of 8 : 3 Encoder. Write its truth table.

Q6. Draw the logic diagram of 3 : 8 Decoder. Write its truth table.

Q7. What is the use of priority encoder? Design 4 : 2 priority encoder and write truth table.

Key Points to Remember :

- ★ MUX → Many to One (Data Selector)
- ★ DEMUX → One to Many (Data Distributor)
- ★ Encoder → Converts 2^n inputs to n outputs
- ★ Decoder → Converts n inputs to 2^n outputs

Marks Distribution : Each Q = 2 to 3 marks Total = 20 Marks

66. Practice Set – 6 (Miscellaneous & Short Answer)

Q1. Design a 3-bit Binary to Gray code converter.

Q2. Design a Gray to Binary code converter.

Q3. Design a 4-bit Parity Generator. Write its truth table.

Q4. Design a 5-bit Parity Checker. How does it detect errors?

Q5. Compare Synchronous and Asynchronous Counters.

Q6. Draw the logic diagram of 3-bit Synchronous Up Counter.

Q7. What are Hazards in combinational circuits? Explain static and dynamic hazards.

Q8. Explain the working of 7-Segment Display. Write the segment pattern to display decimal digit '5' (assume common cathode).

Q9. Differentiate between DAC and ADC.

Q10. Write any four applications of Digital Electronics.

DSSSB TGT CS – Exam Tips

★ Practice truth tables and characteristic tables of all flip-flops.

★ Focus on K-map problems (3 & 4 variable).

★ Understand timing diagrams of sequential circuits.

★ Solve previous year questions for better preparation.

Marks Distribution : Each Q = 2 Marks Total = 20 Marks

T CS)