Binary Adder

Lecture-1

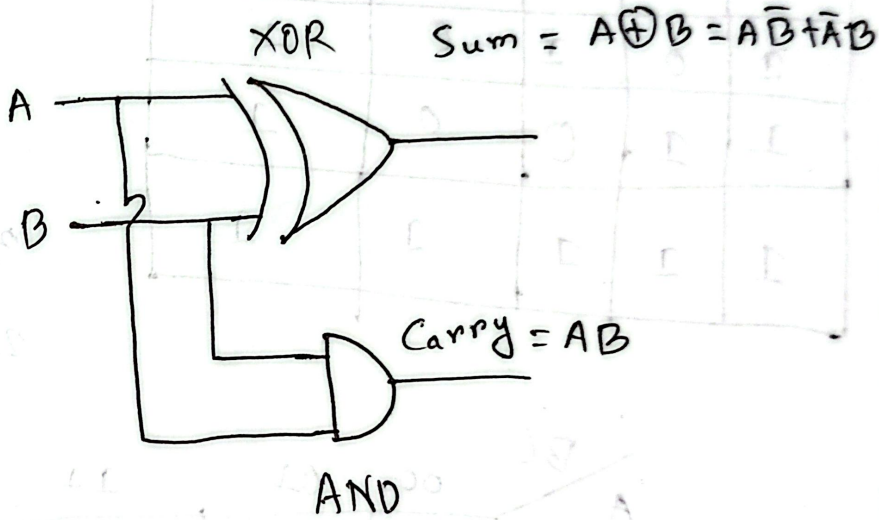
08.04.2025

#There are two types of Binary Adder. They are:

i) Half Adder ii) Full Adder

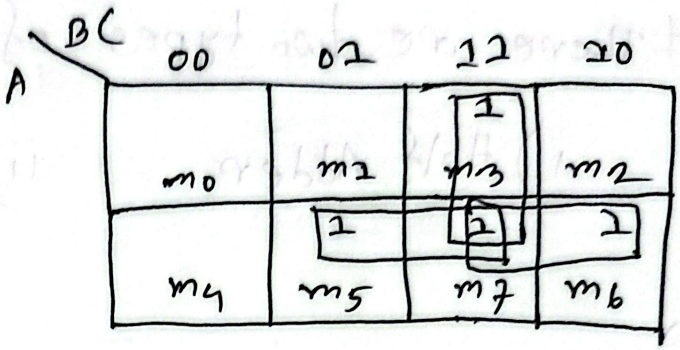
i) Half Adder: A combinational circuit that can perform binary addition of two bits.

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



ii) Full-Adder: A combinational circuit that can perform binary Addition of 3 bits.

A	B	C	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



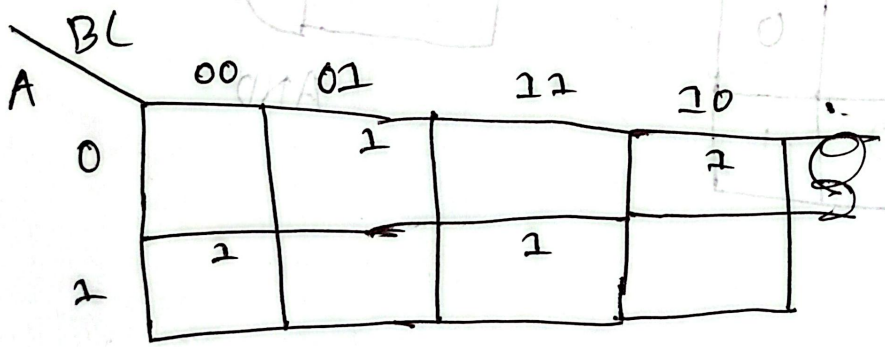
Carry = $G_I + G_{II} + G_{III}$

= $BC + AC + AB$

= $AB + BC + AC$

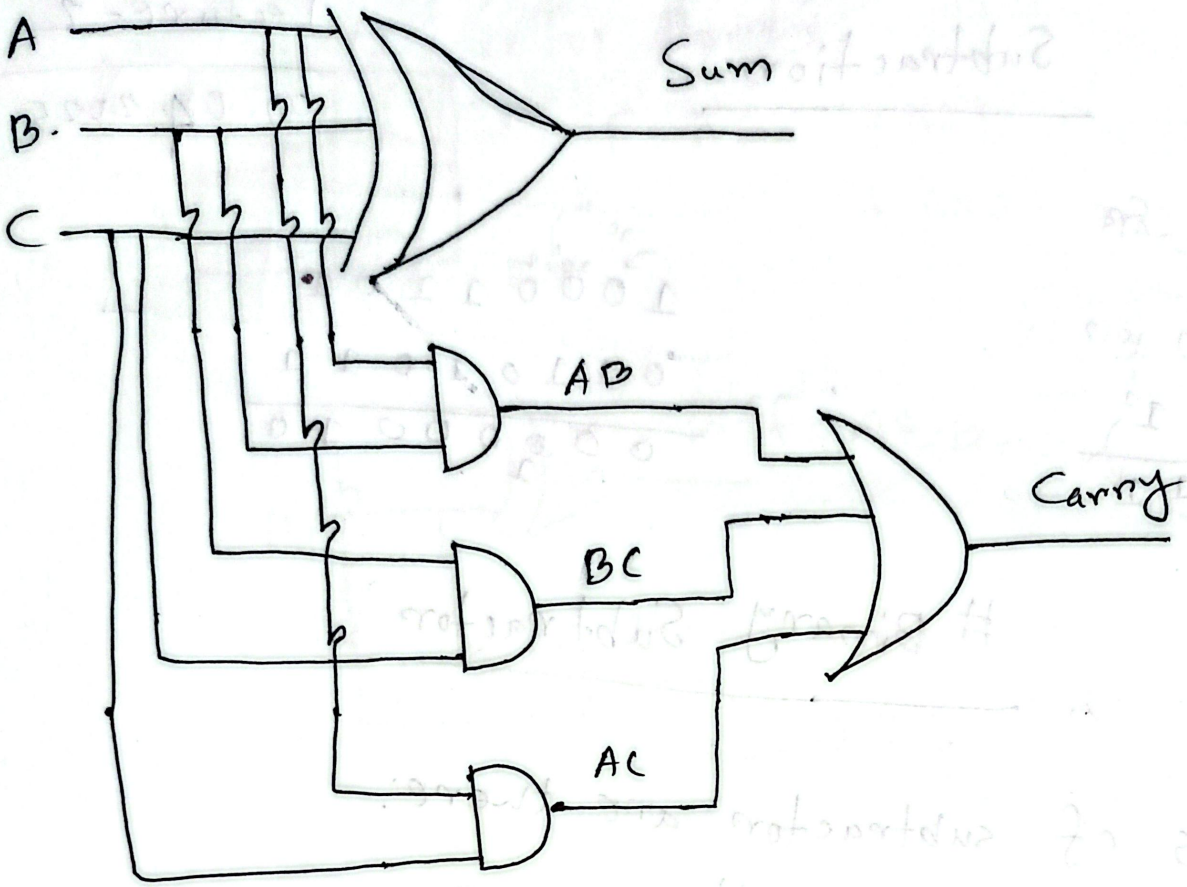
3 NAND gate

2 OR gate

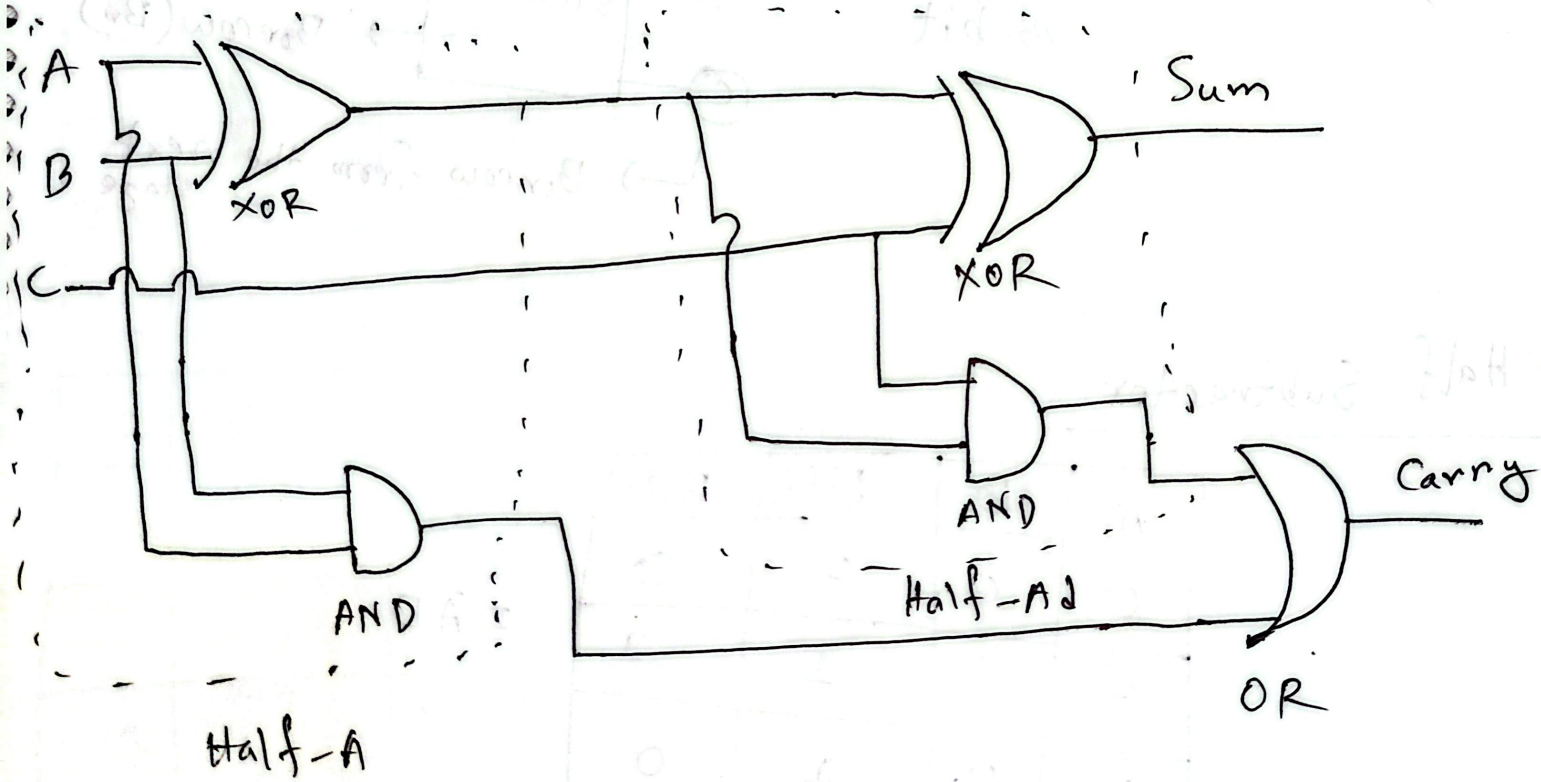


Sum = $\bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$

= $A \oplus B \oplus C$



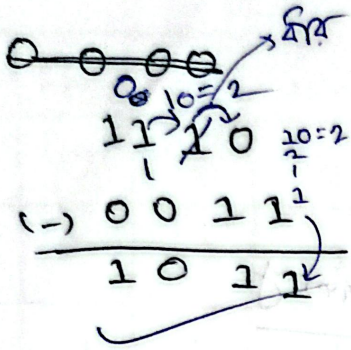
Full Adder Using Half Adders

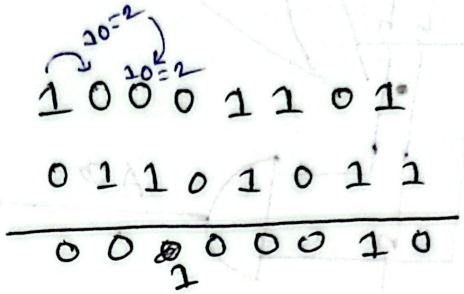


Subtraction

Lecture-2

09.08.2025



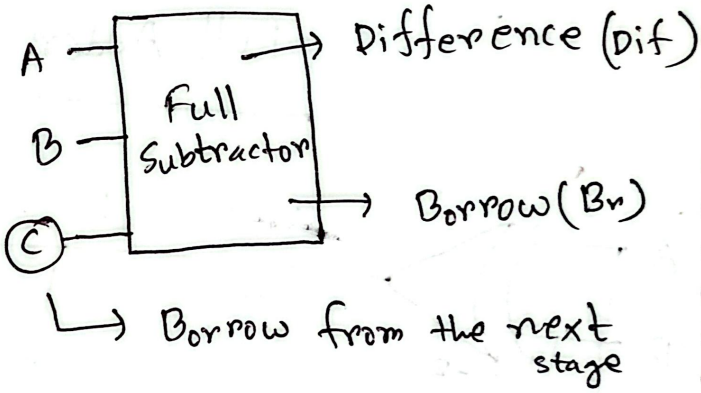


Binary Subtractor

⇒ Two types of subtractor are there:

i) Half Subtractor → 2-bit

ii) Full Subtractor
m-bit



Half Subtractor:

A	B	Diff	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

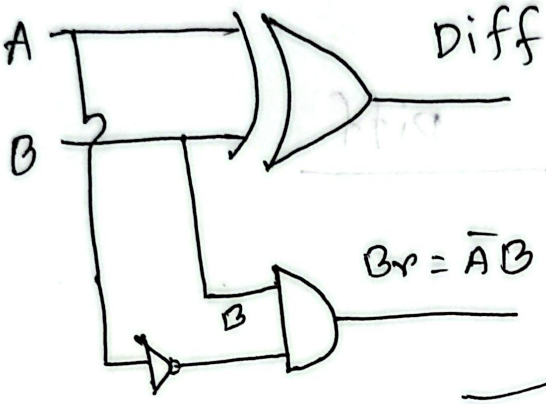
$= \bar{A}B$

A \ B	0	1
0	m_0	m_1
1	m_2	m_3

For Borrow,

$F = \bar{A}B$ 1 NOT

1 AND



→ Fig: Half Subtractor

Full Subtractor:

A	B	C	Diff	Br
0	0	0	0	0
0	$10=2$ 0	1	1	1
$10=2$ 0	1 (1)	0	1	1
$10=2$ 0	1 (1)	1	0	1
1	0 (1)	0	1	0
1	0 (1)	1	0	0
1	1 (0)	0	0	0
1	$10=2$ 1	1	1	1

For Diff,

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

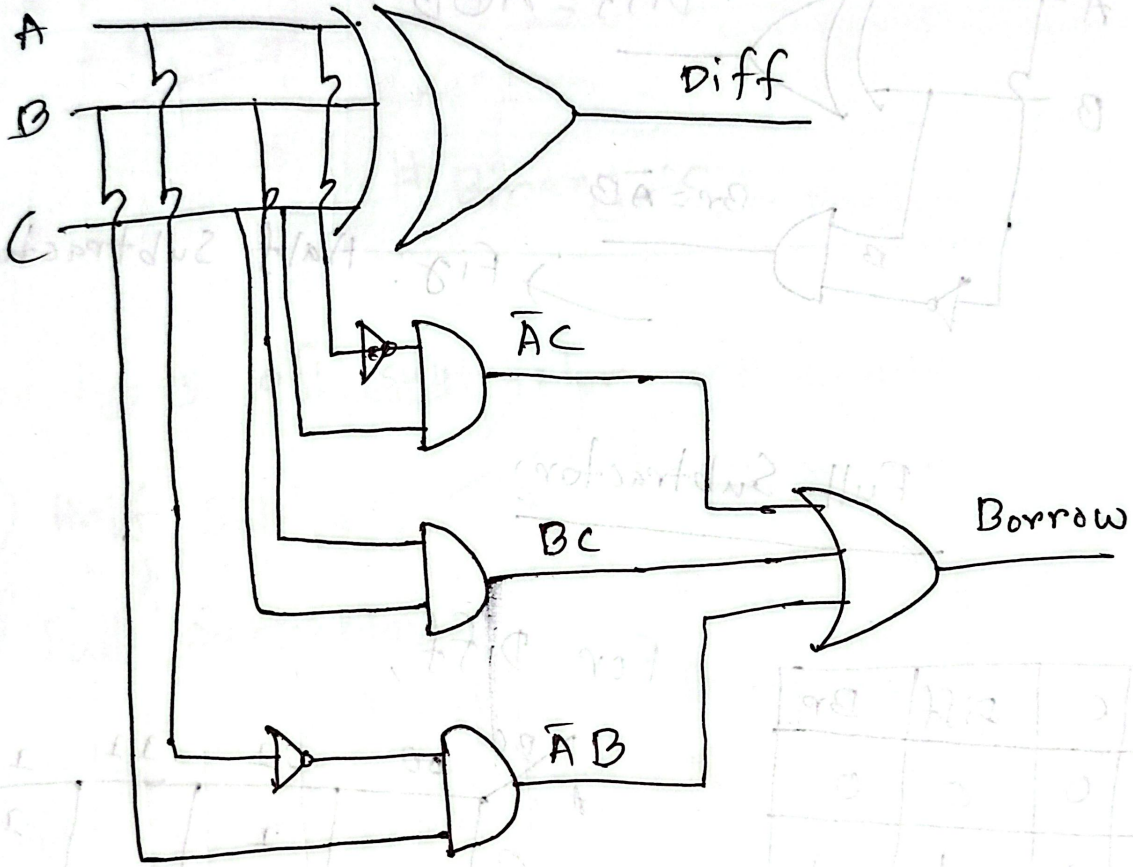
Difference $= \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

$= A \odot B \odot C$

For Borrow,

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

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

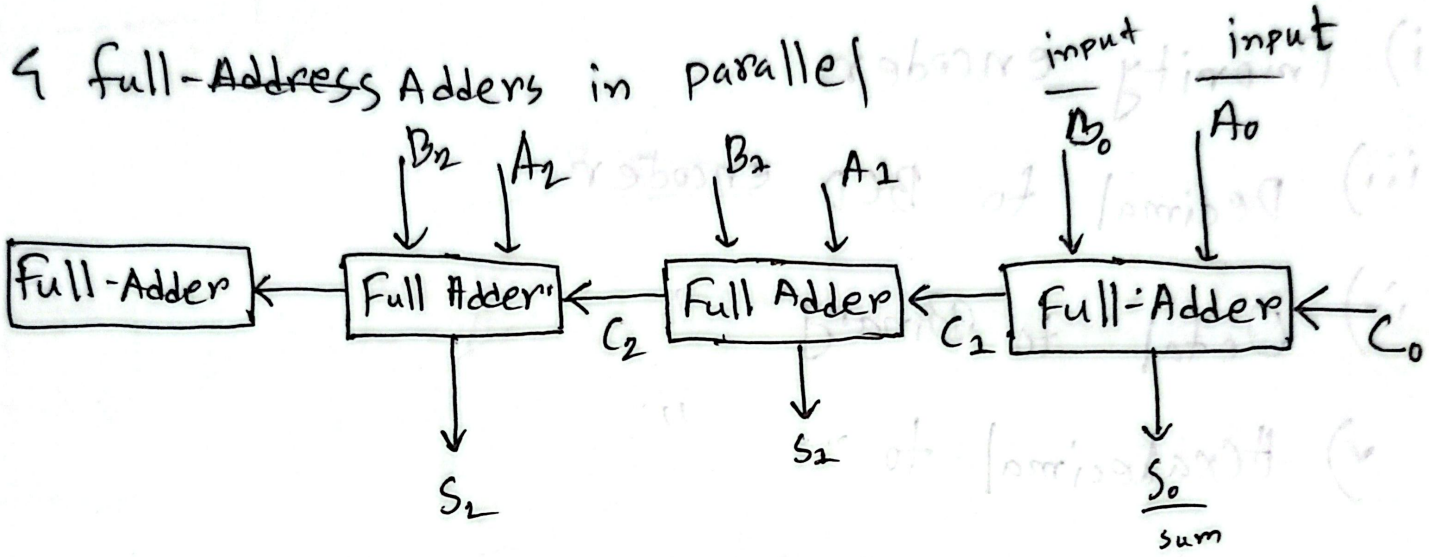


Lecture-3

15.04.2025

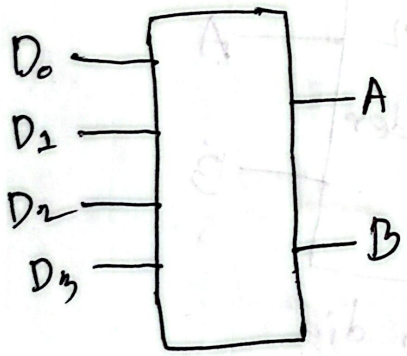
Binary Parallel Adder (4-bit)

∋ 4 full-~~Address~~ Adders in parallel



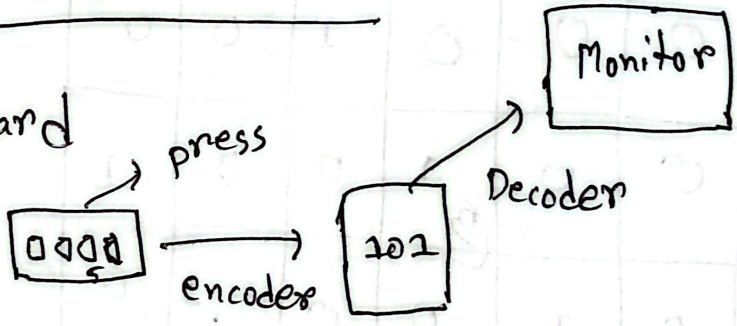
Encoder

∋ A binary encoder is a combinational circuit that converts 2^n input lines into n inputs.



Application

i) Keyboard



ii) Communication system for data compression

iii) Microprocessor

Types: i) Binary encoder: $2^n \rightarrow n$

condition : only one input is high at a time.

ii) Priority encoder

iii) Decimal to BCD encoder

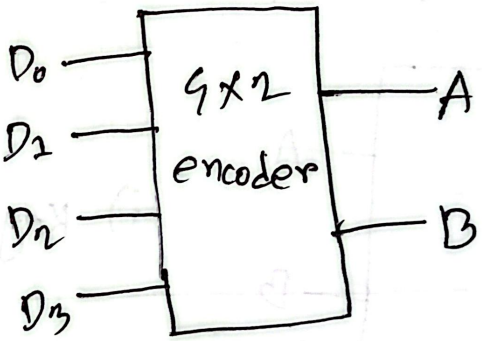
iv) Octal to Binary "

v) Hexadecimal to " "

Binary Encoder

4x2 → 4 input 2 output (pattern)

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

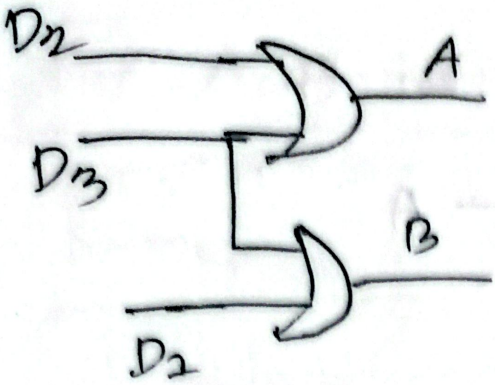


pin dig

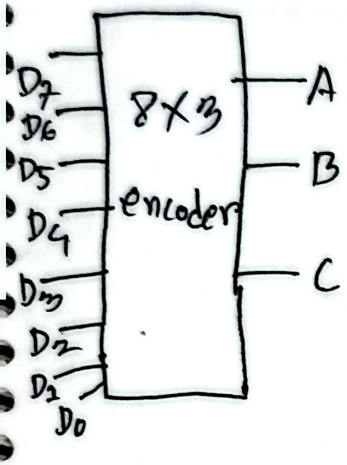
$A = D_2 + D_3$

$B = D_1 + D_3$

2 OR



8×3 Binary Encoder

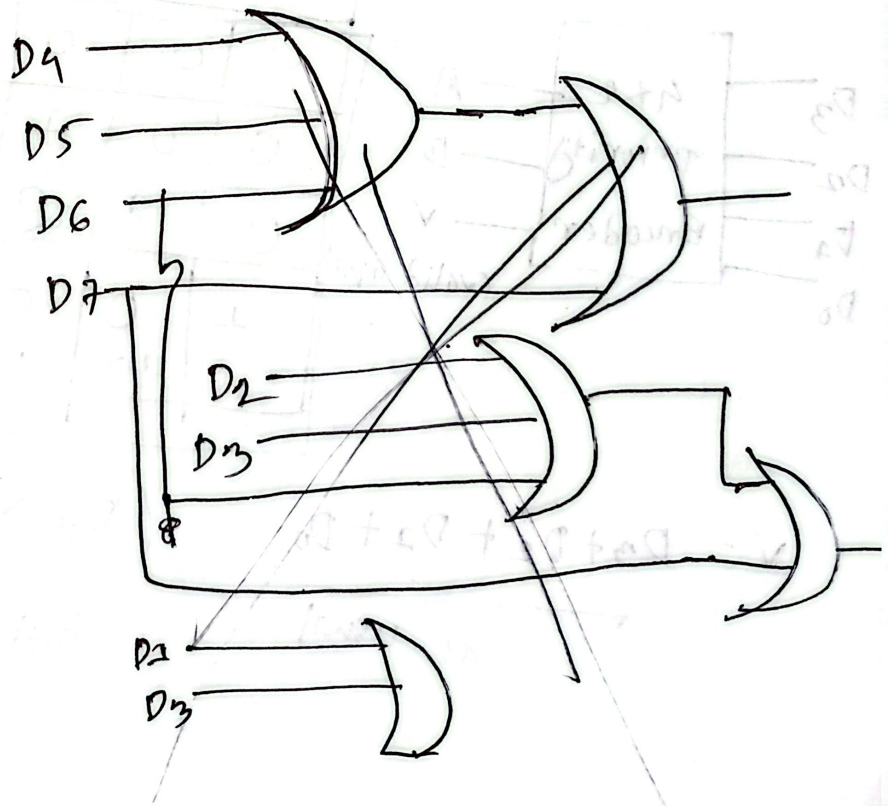


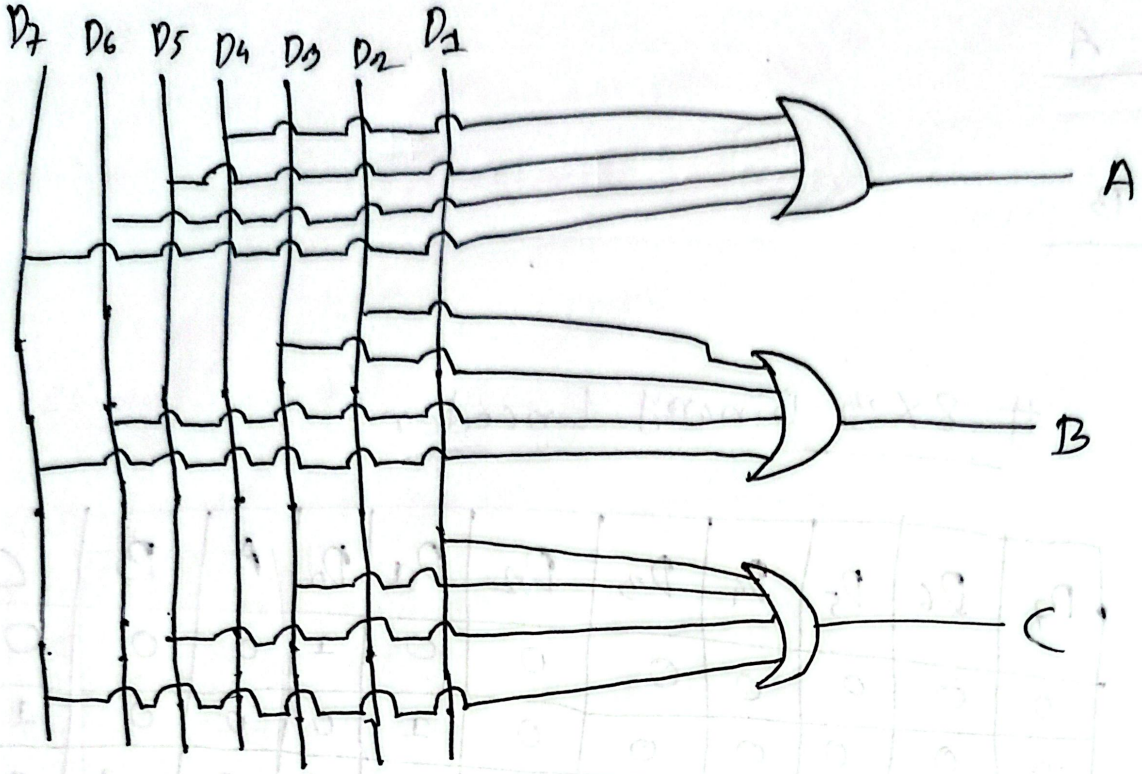
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A	B	C
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

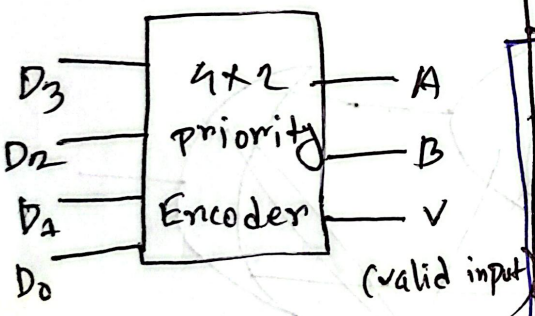
$C = D_1 + D_3 + D_5 + D_7$





Priority Encoder

$D_3 > D_2 > D_1 > D_0$



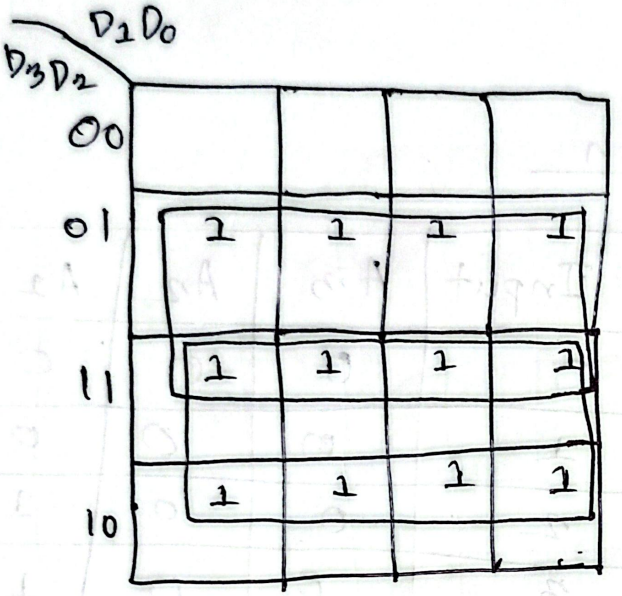
D_3	D_2	D_1	D_0	A	B	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	1X	0	1	1
0	1	1X	1X	1	0	1
1	1X	1X	1X	1	1	1

$V = D_3 + D_2 + D_1 + D_0$ as usual

($D_3 > D_2 > D_1 > D_0$) Applied instead 1 "X" will be used

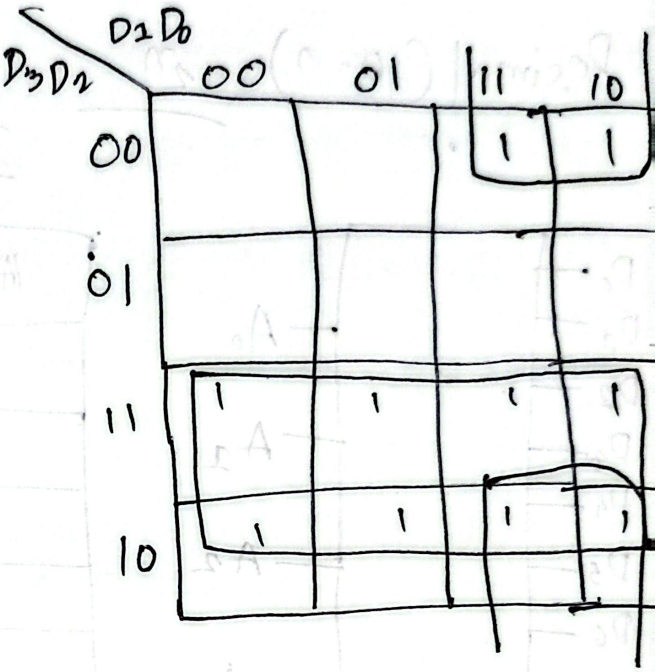
for A, B input

K-map A.



$A = D_3 + D_2$

B



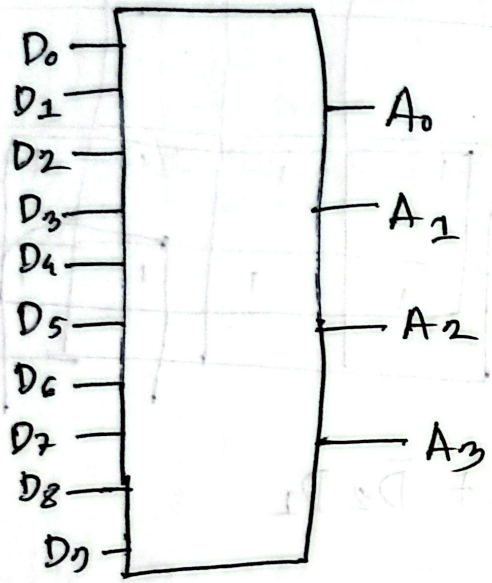
$B = D_3 + \bar{D}_2 D_1$

Lecture-4

16.04.2025

Decimal to BCD encoder

Decimal (0-9) 2^n



Truth

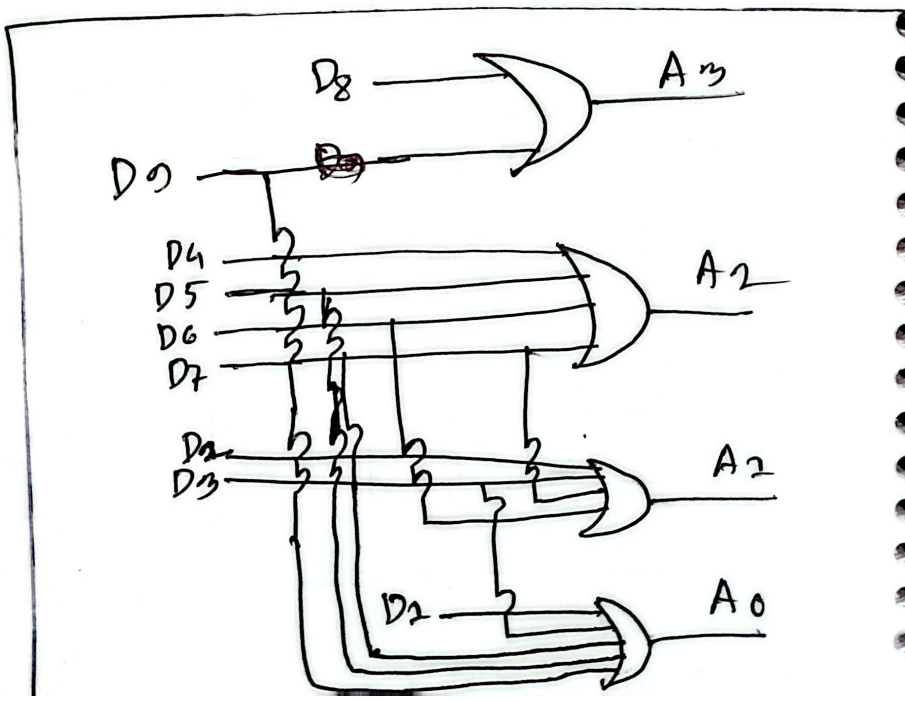
High Input	A3	A2	A1	A0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

$A_3 = D_8 + D_9$

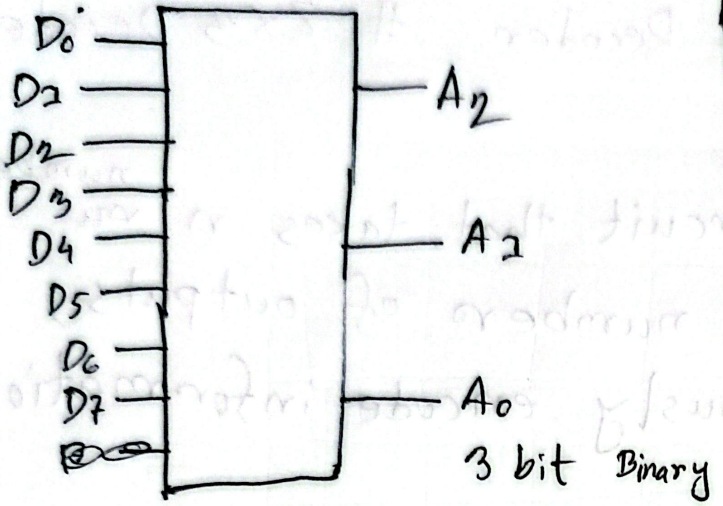
$A_2 = D_4 + D_5 + D_6 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_0 = D_1 + D_3 + D_5 + D_7 + D_9$



Octal to Binary

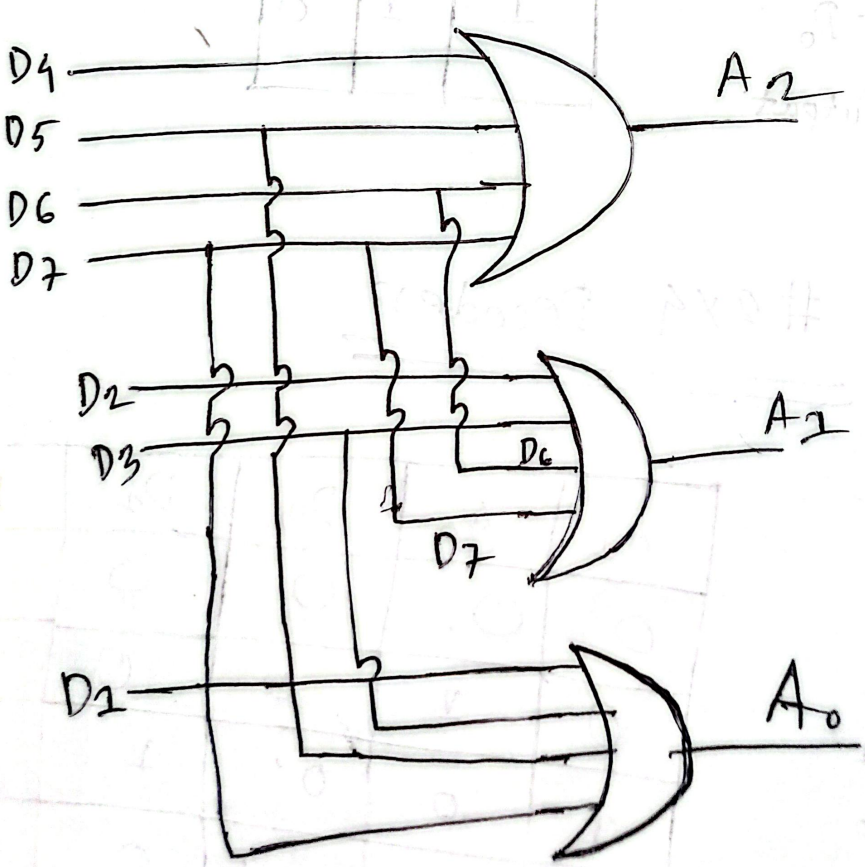


High Input	A_2	A_1	A_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

$A_2 = D_4 + D_5 + D_6 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_0 = D_1 + D_3 + D_5 + D_7$

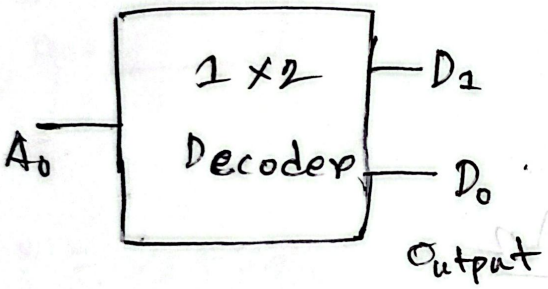


Decoder

#2×4 Decoder #1×2 Decoder #8×3 Decoder

Definition: A combinational circuit that takes n number of inputs and provides 2^n number of outputs, used to decode the previously encode information.

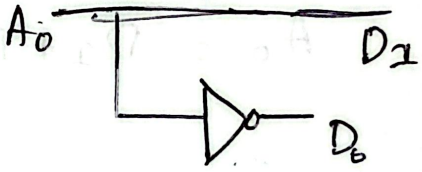
#1×2 Decoder



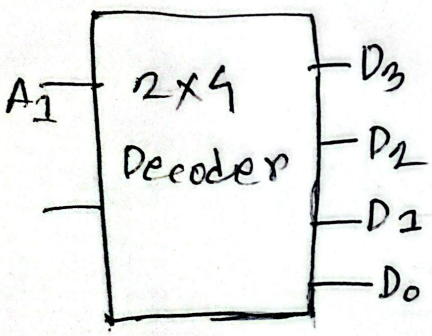
A_0	D_1	D_0
0	0	1
1	1	0

$D_1 = A_0$

$D_0 = \overline{A_0}$



#2×4 Decoder



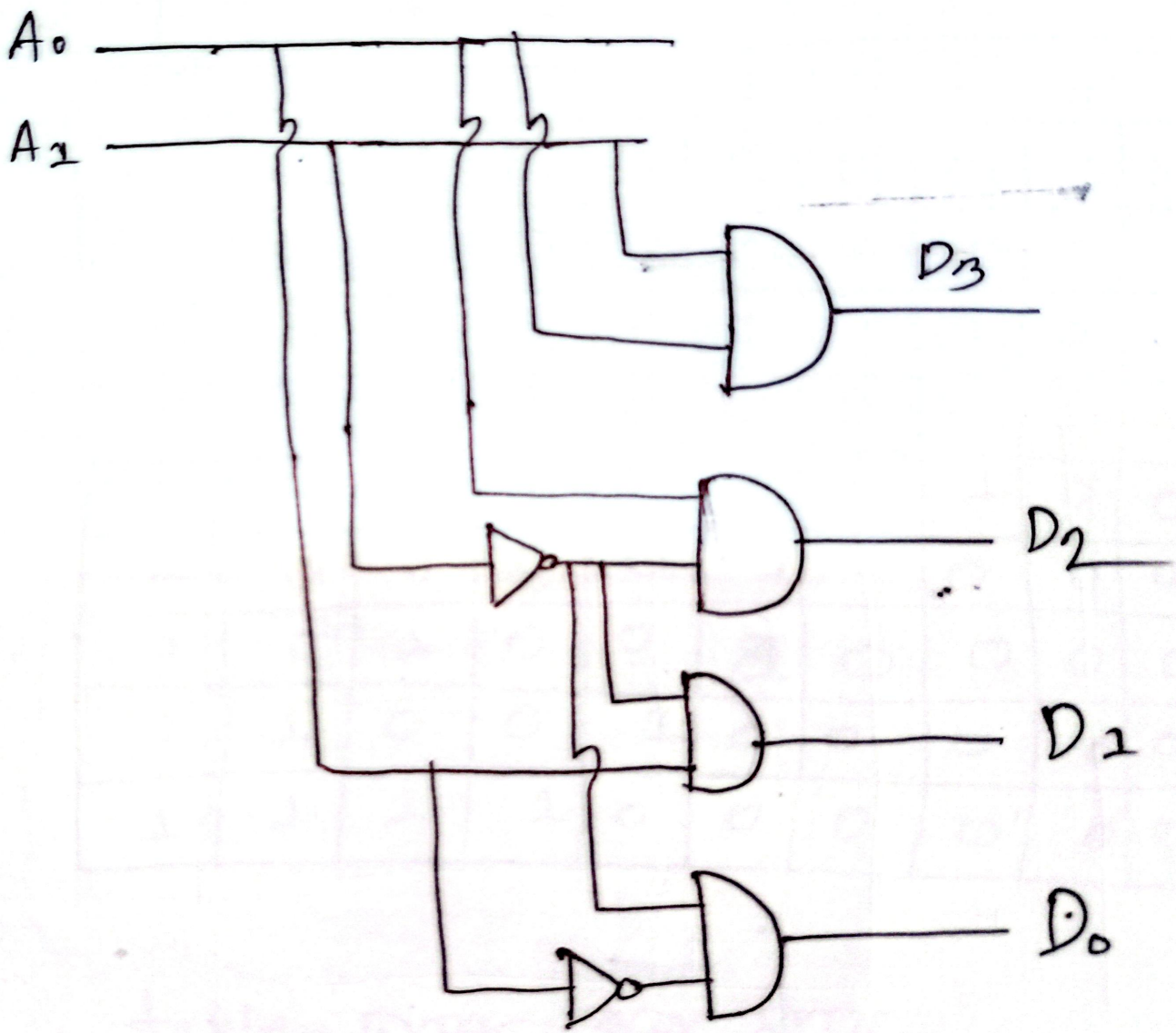
A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$D_3 = A_1 A_0$

$D_2 = A_1 \overline{A_0}$

$D_1 = \overline{A_1} A_0$

$D_0 = \overline{A_1}\,\overline{A_0}$

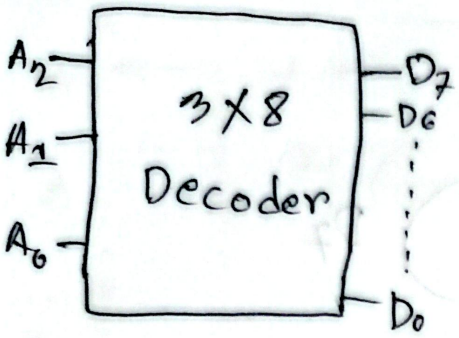


Lecture-06

22.04.2025

#3X8 Decoder

Lecture-5



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

$D_7 = A_2 A_1 A_0$

$D_6 = A_2 A_1 \bar{A}_0$

$D_5 = A_2 \bar{A}_1 A_0$

$D_4 = A_2 \bar{A}_1 \bar{A}_0$

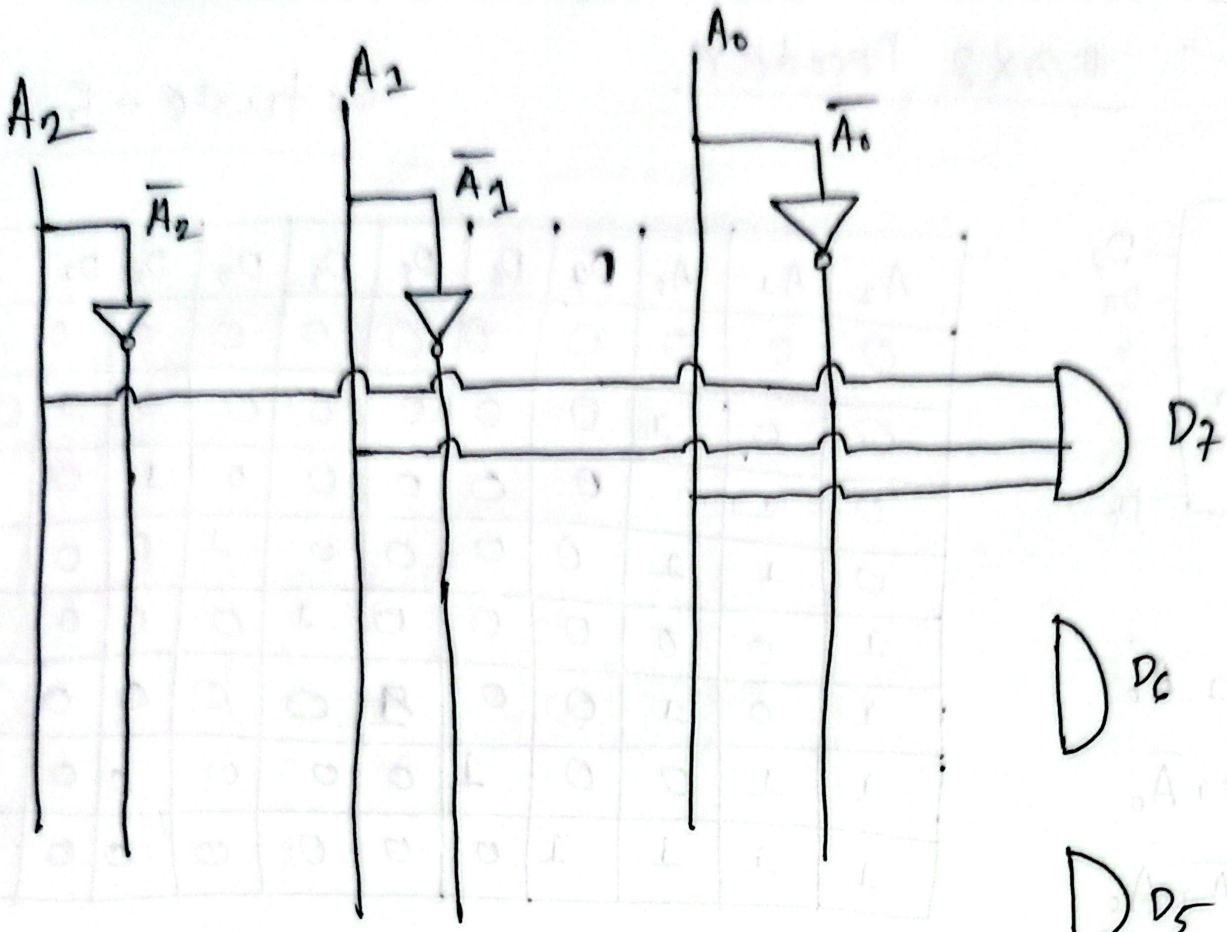
$D_3 = \bar{A}_2 A_1 A_0$

$D_2 = \bar{A}_2 A_1 \bar{A}_0$

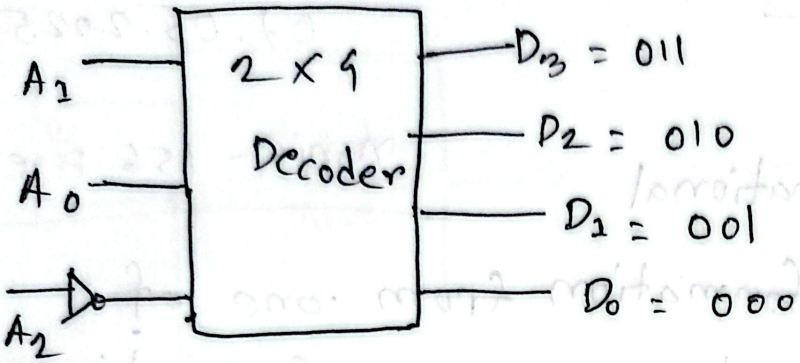
$D_1 = \bar{A}_2 \bar{A}_1 A_0$

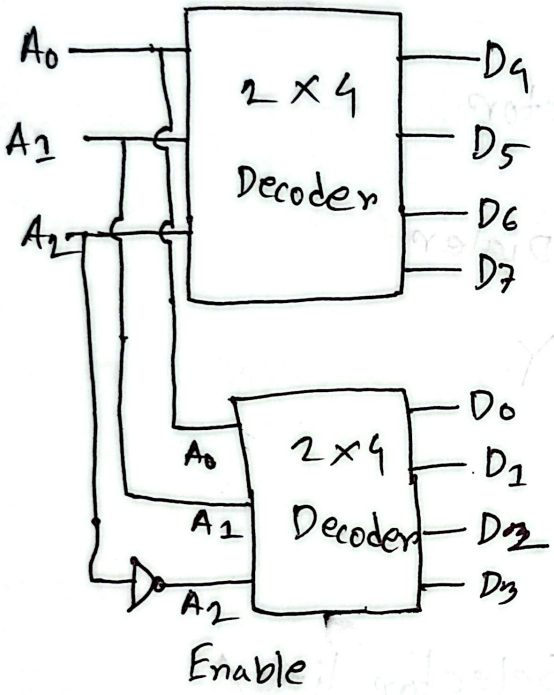
$D_0 = \bar{A}_2 \bar{A}_1 \bar{A}_0$ {3 bit Binary minterm}

table-ছাড়া কাজ করা যাবে,
corresponding binary বিশ্লেষণ


A2
A1
A0
$\overline{A_2}$
$\overline{A_1}$
$\overline{A_0}$
D7
D6
D5
D4
D3
D2
D1
D0

3×8 Decoder using two 2×4 Decoder





A0	A1	A2
0	0	0
0	0	1
0	1	0
0	1	1
0	1	0
1	0	1
1	0	0
1	1	1
1	1	1

এখানে ইনপুট দিতে হবে।

Mano

chapter-3

p-(144-146) Decoder

p- 151 (Adder using)

p-153 (Encoder)

-153 (priority Encoder)

Multiplexer

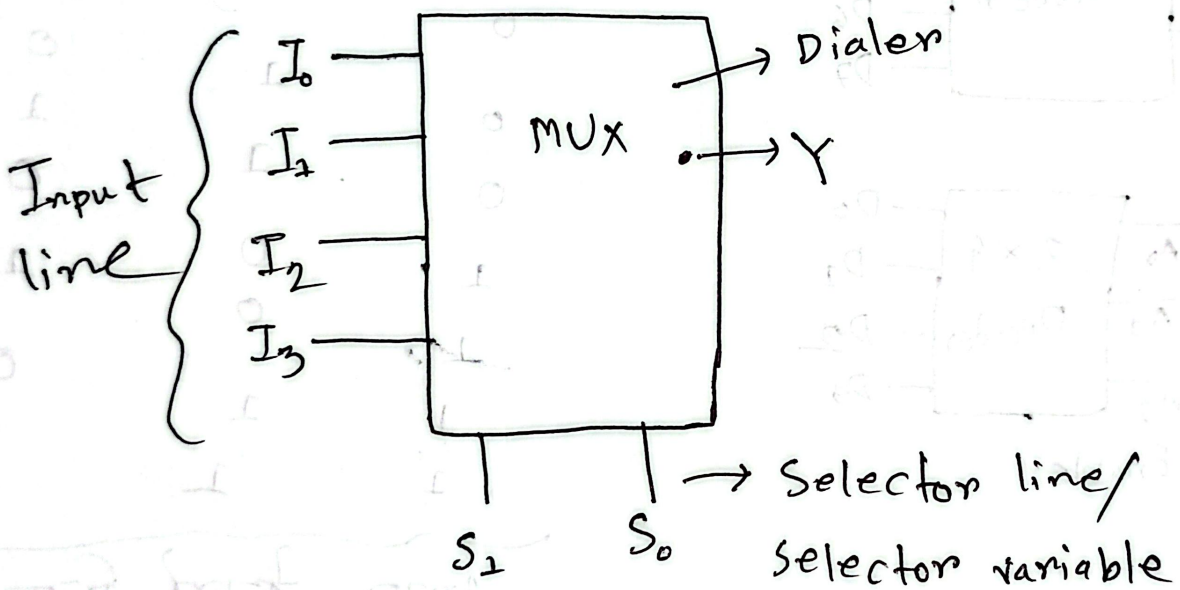
Lecture-6

07.05.2025

Mano's - 156 page

(*) Multiplexer is a combinational circuit that selects information from one of many input lines and directs the information to only one output line.

(*) Multiplexer works as a selector



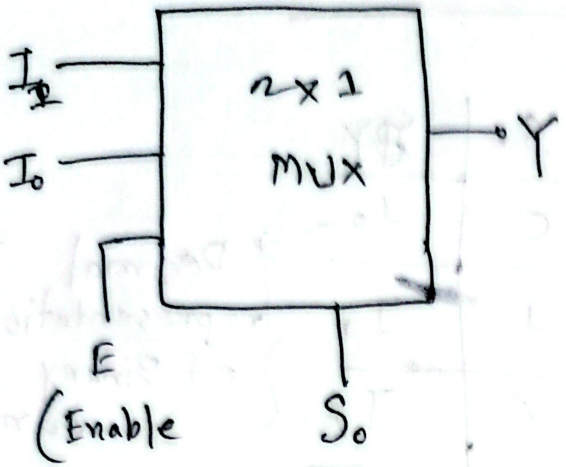
rule:

$2^n \rightarrow$ Input line

$n \rightarrow$ selection line

$1 \rightarrow$ Output line

#2×1 Multiplexer



Truth Table

E	S	Y
0	X	0
1	0	I_0
1	1	I_1

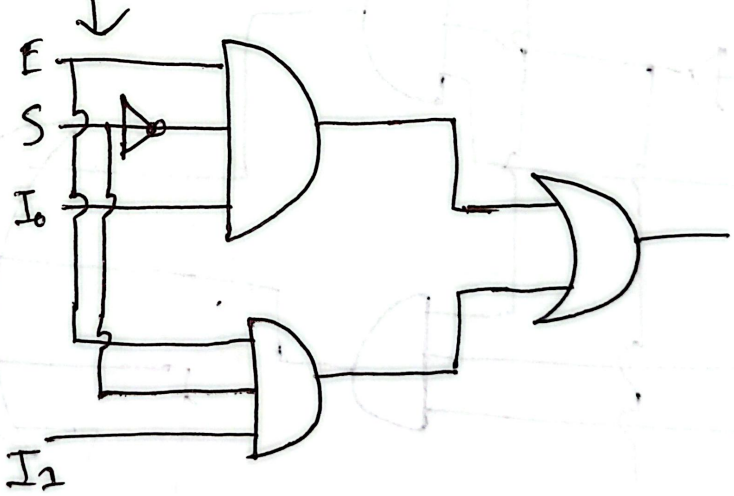
{ Enable 0 হলে circuit কাজ করবে না।
Enable 1 হলে circuit active

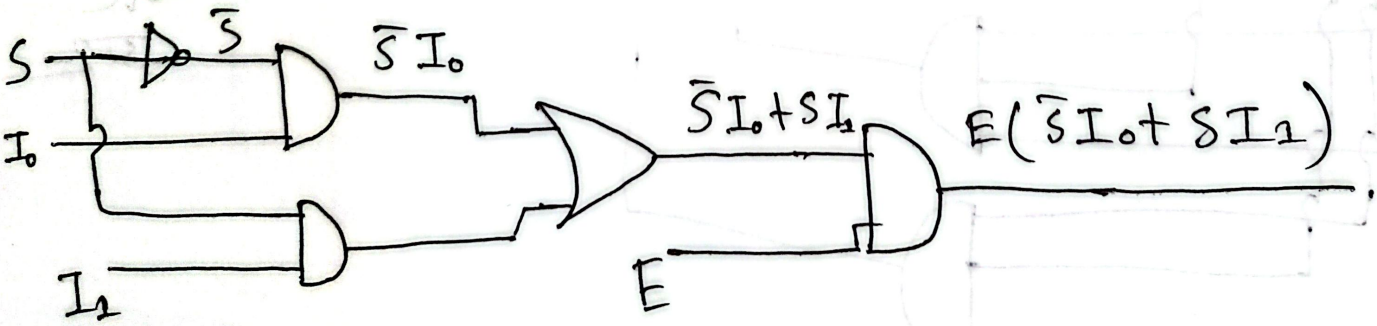
{ S এর value 0 হলে

Circuit:

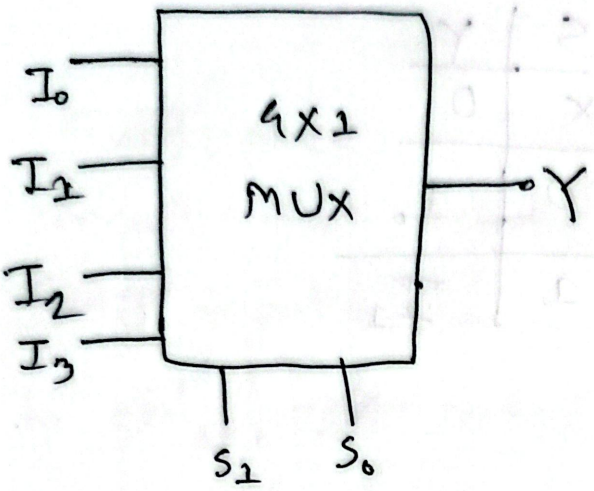
$E\bar{S}I_0 + ESI_1$

$= E(\bar{S}I_0 + SI_1)$





#4x1 Multiplexer



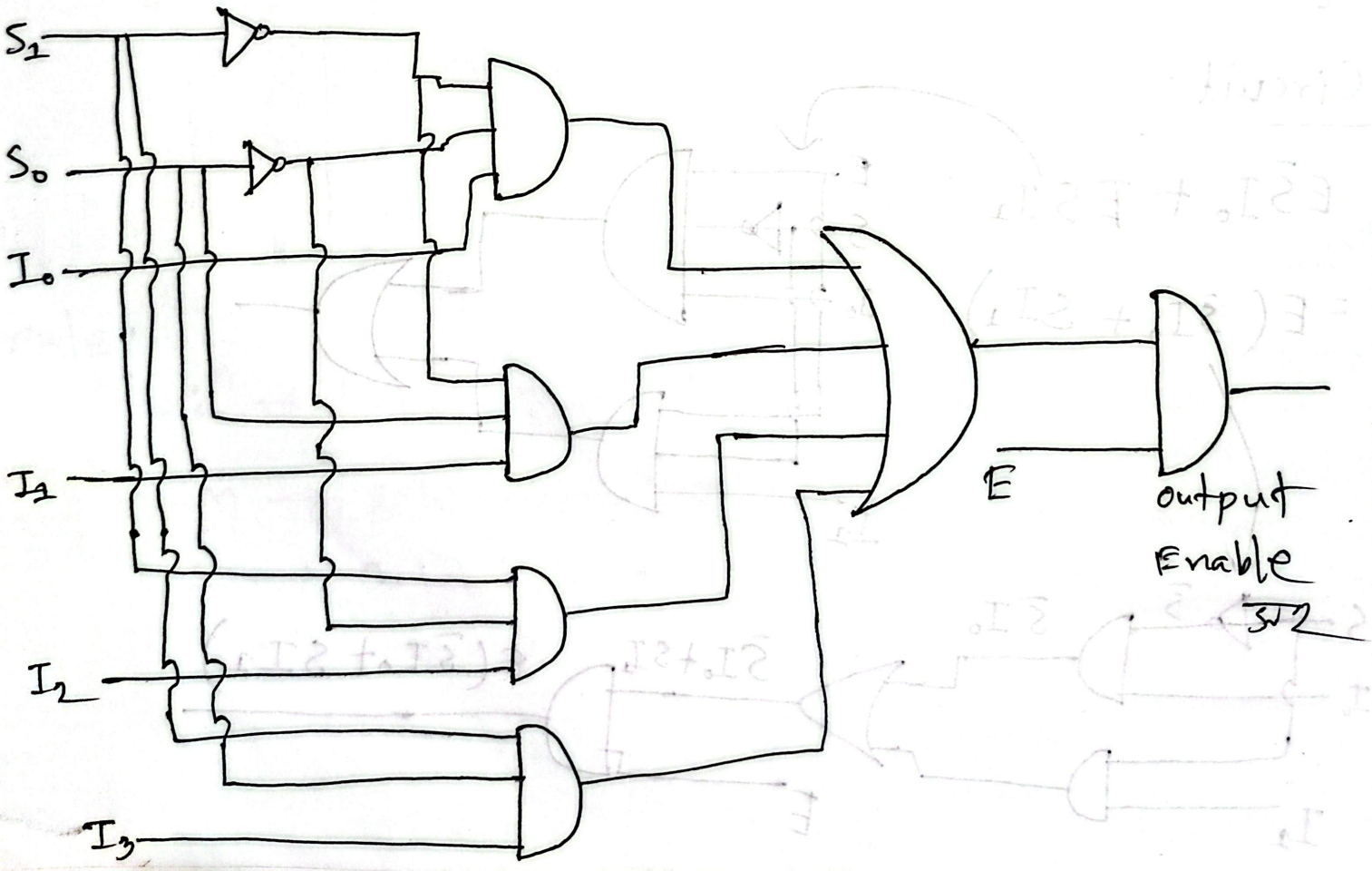
Truth Table:

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

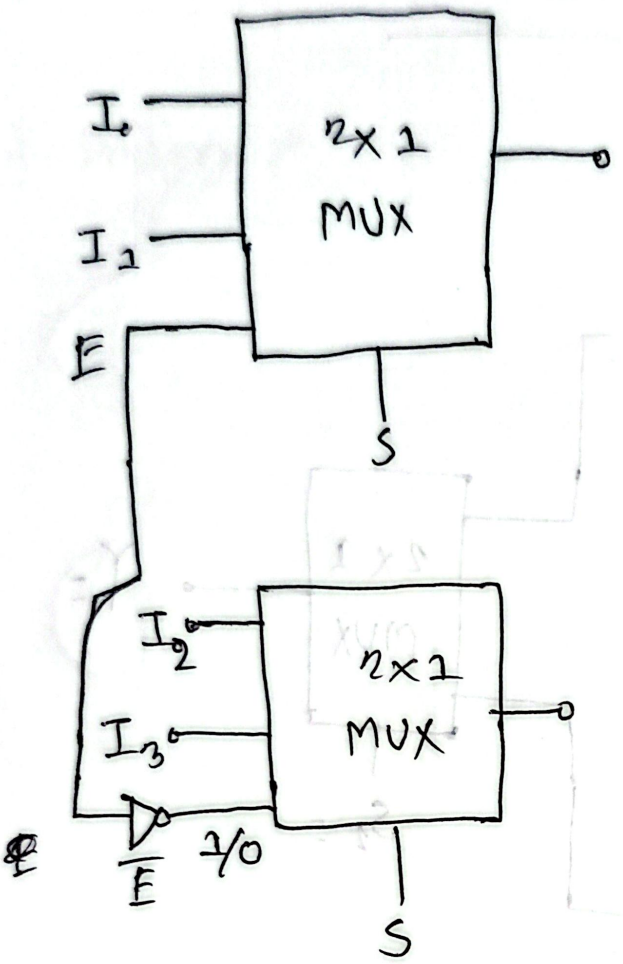
Decimal representation of Binary num

Circuit:

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



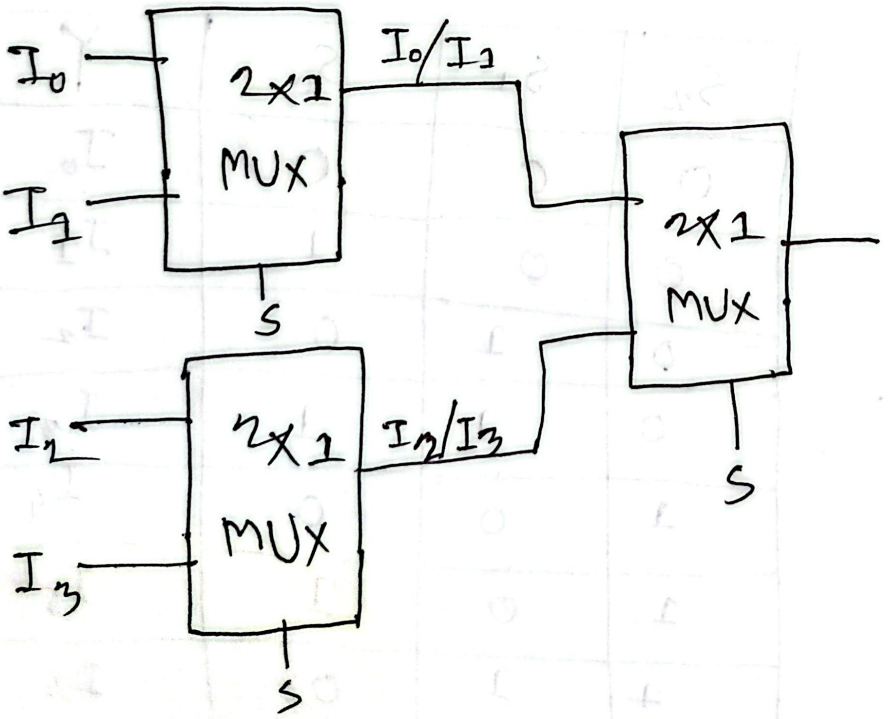
2টি 2×1 MUX দিয়ে 1টি 4×1 MUX :



Truth table

E	S	Y
1	0	I_0
1	1	I_1
0	0	I_2
0	1	I_3

without using Enable



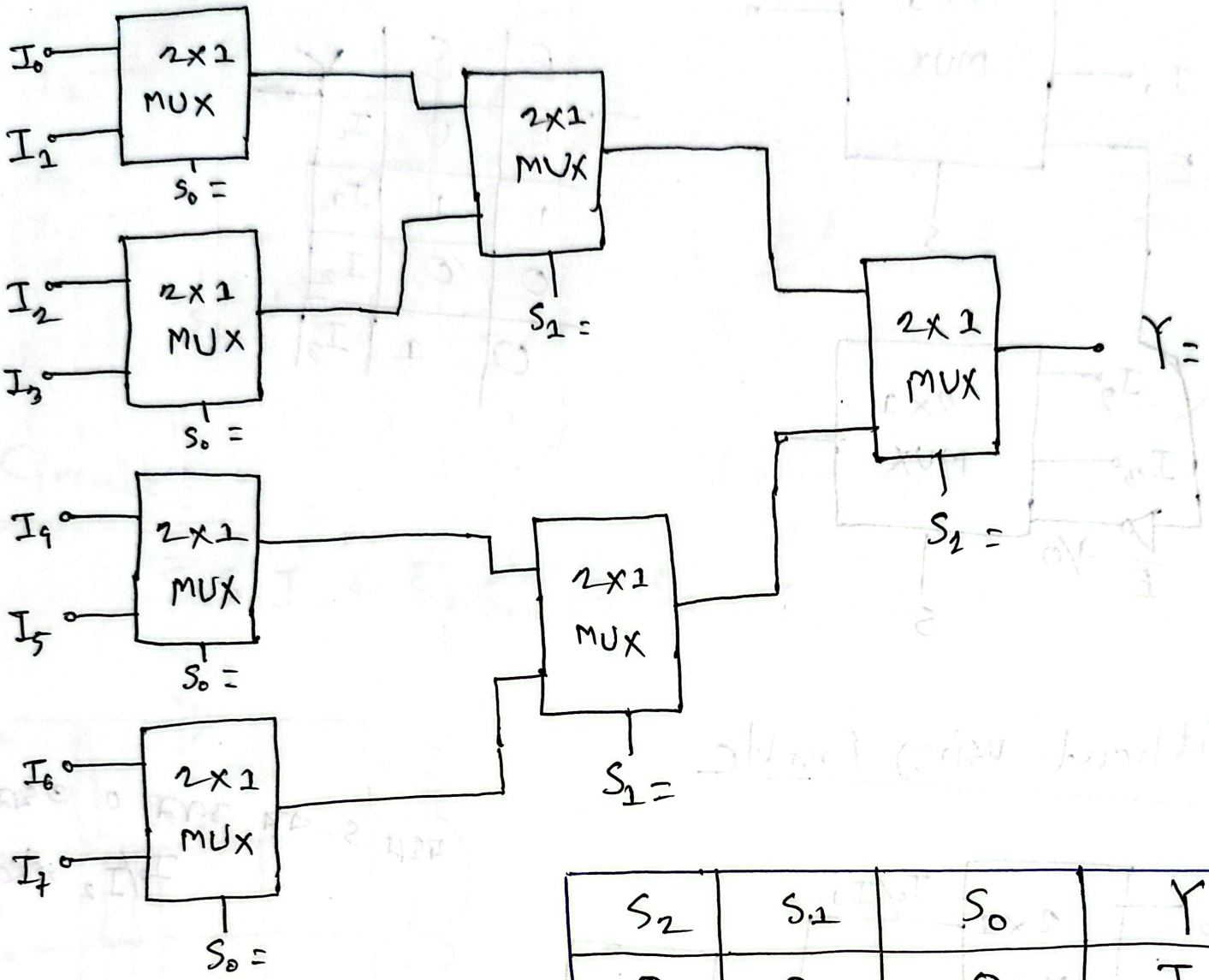
যখন S এর মান 0 তখন, I_0/I_2 select হবে

যখন S এর মান 1 তখন, I_1/I_3 select হবে

Lecture-7

13.05.2025

8×1 MUX using 2×1 MUX



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

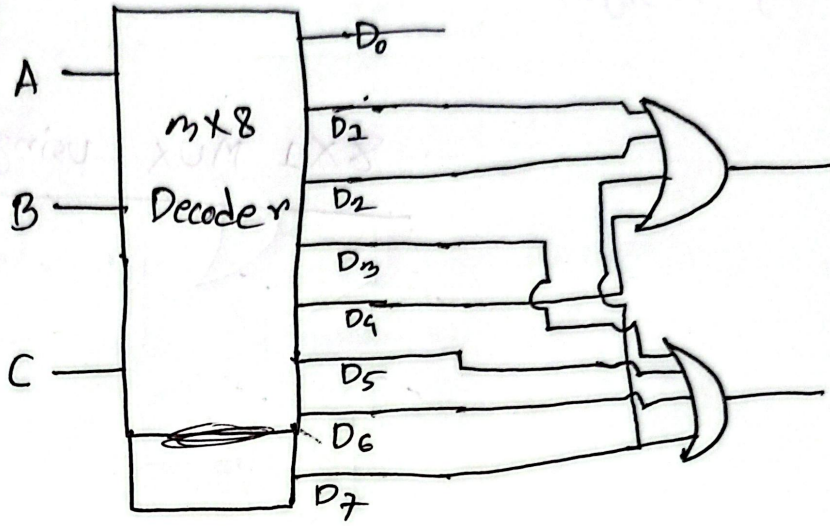
Q. Design a 8x1 MUX using MUX (2x1), given that $Y = I_4$

Q. Design " 4x1 " " 2x1 MUX, for $Y = I_2$.

8x1 MUX using 4x1 MUX

Full Adder Using Decoder

A	B	C	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



$Sum = D_1 + D_2 + D_4 + D_7$

$= D_3 + D_5 + D_6 + D_7$

Half Adder Using Decoder

Sequential Circuit

❖ Sequential circuit is a logic circuit whose output depends on the present input as well as past output on past sequence of input

Application:

i) Flip-Flop (memory)

ii) Timers

iii) Counter

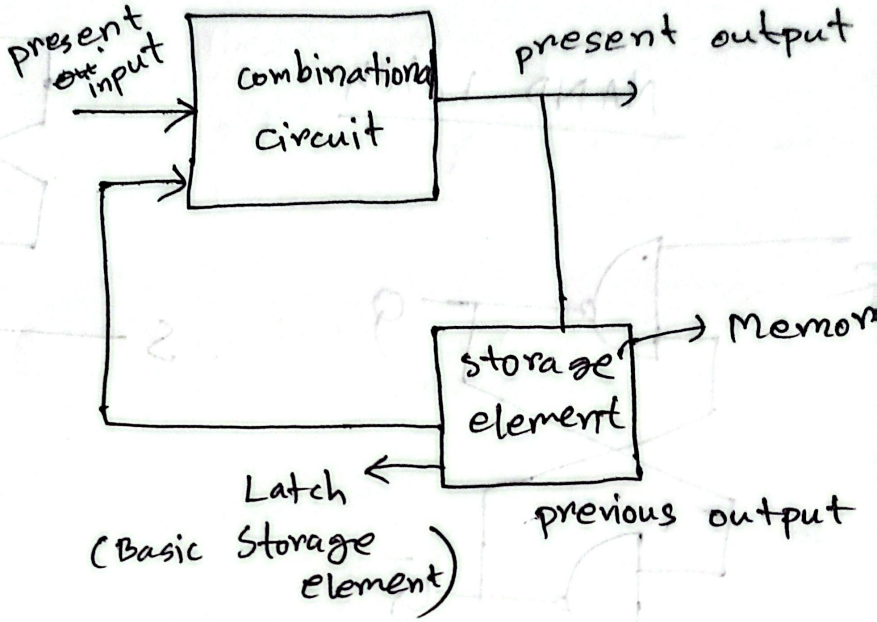


Fig: Block diagram of a sequential circuit.

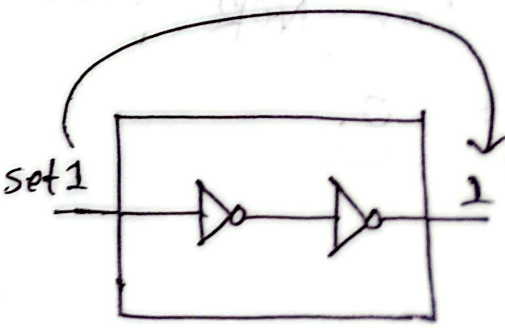
Latch

Types:

i) SR Latch (Set-Reset Latch)

ii) T Latch (Toggling Latch)

iii) D Latch

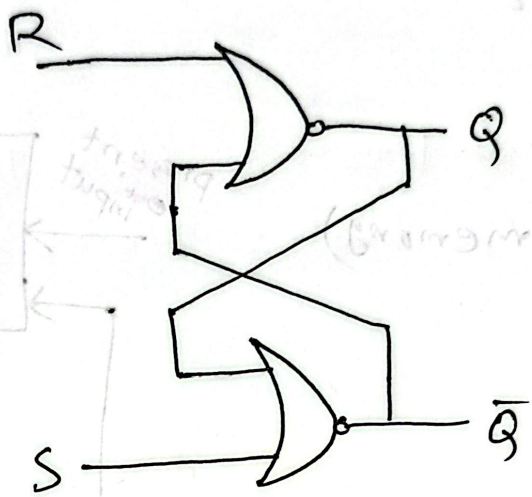


i) SR Latch (Set-Reset):

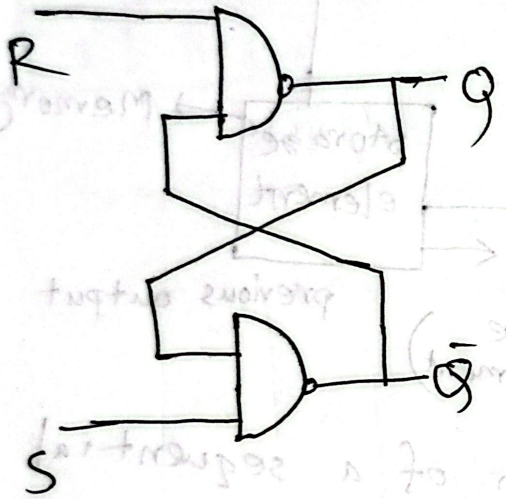
Can be implemented in two ways:

i) NAND Latch
ii) NOR Latch

NOR Latch



NAND Latch



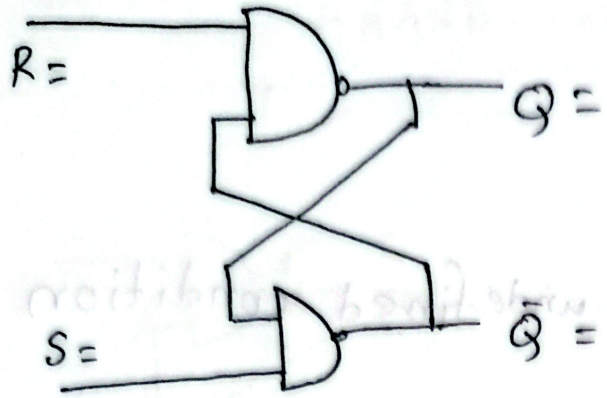
NOR SR Latch

Lecture-8
14.05.2025

For NOR gate if any of the input 1 output will be 0.

NOR (TT) →

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



$Q=1$, set state; set the output to 1

Reset মানে 0 তে নামানো

Case-1:

For, $R=0$, $S=1$

Output $Q=1$, $\bar{Q}=0$

previous stage

For, $R=0$, $S=0$

Output → $Q=1$, $\bar{Q}=0$

∴ therefore the previous output has been restored.

Case-2:

For, $R=1$, $S=0$

Output, $Q=0$, $\bar{Q}=1$

↙

Reset state

reset to 0

If both of the inputs are removed,

$R=0$, $S=0$

Output $Q=0$, $\bar{Q}=1$.

∴ therefore previous output has been restored

Case-3:

For, $R = 1, S = 1$

Output, $Q = 0, \bar{Q} = 0$ } → undefined condition

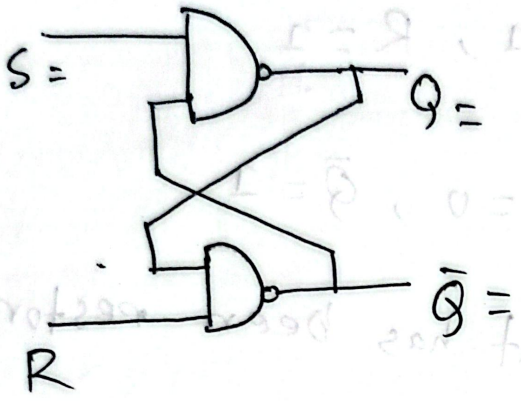
[since $Q \neq \bar{Q}$]*

$S = 1; R = 1$, this will not be used in case of NOR SR Latch, as the result output is not acceptable.

Truth Table for NOR SR Latch

S	R	Q	$\bar{Q}$	states
1	0	1	0	set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined state

#NAND SR Latch



Truth Table (NAND)

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

In case of NAND gate, if any of the input is 0 output will be 1

Case-1:

When, $S=0$, $R=1$

Output, $Q=1$, $\bar{Q}=0$

when, $S=1$, $R=1$

Output, $Q=1$, $\bar{Q}=0$

set state

therefore the previous output has been restored

Case-2:

When,
$S = 1, R = 0$

Output, $Q = 0, \bar{Q} = 1$

~~Both of inputs~~

when,
both inputs will be turned 1.

$S = 1, R = 1$

Output,

$Q = 0, \bar{Q} = 1$

therefore the previous output has been restored.

Case-3

When,

$S = 0, R = 0$; Output, $Q = 1, \bar{Q} = 1$ } → undefined state

$\therefore Q \neq \bar{Q}$ {condition}

$S = 0; R = 1$, this will not be used in case of NAND SR Latch; as the result output is not acceptable.

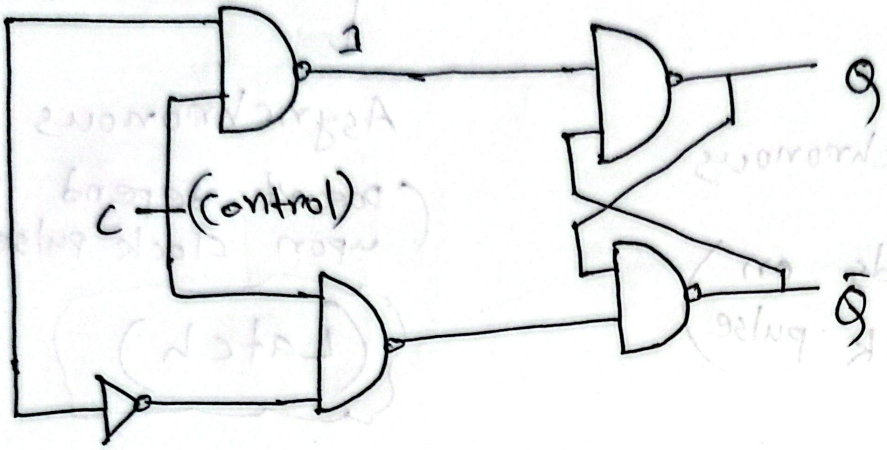
TT for NAND SR Latch;

S	R	Q	$\bar{Q}$	State
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	undefined state

Morris mano
Chap-4 (217-p)

D Latch

D = X



Initially, C = 0

S = 1, R = 1

Output of the previous state will be restored.

When,

C = 1, D = 0

Q = 0, $\bar{Q}$ = 1

Reset state

When,

C = 1, D = 1

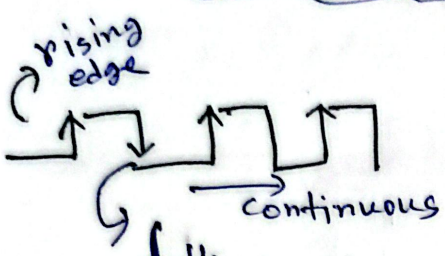
Q = 1, $\bar{Q}$ = 0

set state

Lecture-9

20.05.2025

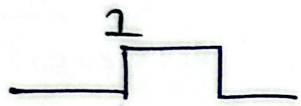
edge triggering



Sequential Circuit

- **Synchronous** (Depends on clock pulse)
- **Asynchronous** (Doesn't depend upon clock pulse) (Latch)

Flip-Flop, Counter



level triggering

if low, state will not change

if high, " " change

Flip - Flop

i) SR Flip - Flop (Set - Reset Flip...)

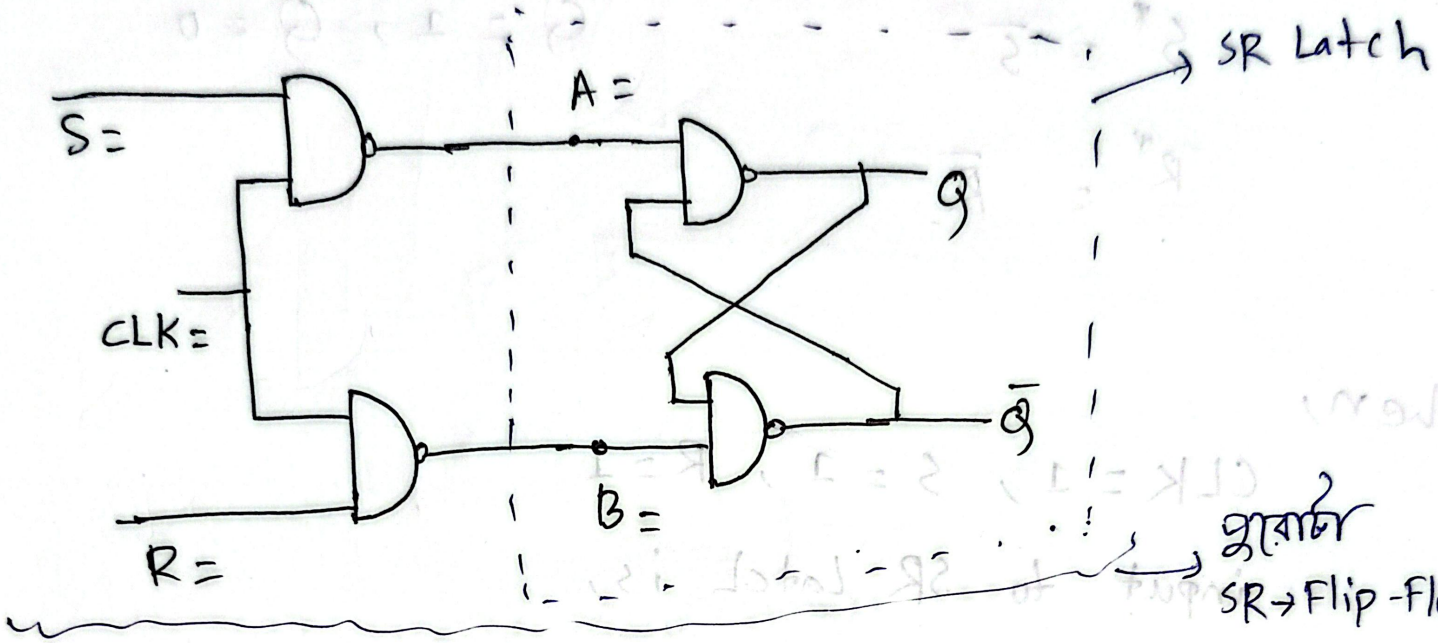
ii) D - Flip - Flop

iii) Jk - Flip - Flop (Toggling)

iv) T - Flip - Flop ~~(Toggling)~~

Flip-Flop

SR Flip-Flop



If CLK = 0, A = 1, B = 1, $Q_{n+1} = Q_n$

present previous

memory state

$A = \overline{S.CLK}$

$B = \overline{R.CLK}$

$\overline{Q_{n+1}} = \overline{Q_n}$

present previous

when,

CLK = 1, S = 0, R = 1

$A = \bar{S}$

$B = \bar{R}$

input to SR Latch is,

S = 1, R = 0

Q = 0, $\bar{Q}$ = 1

when,

CLK = 1, S = 1, R = 0

input to SR Latch is, $S^* = 0$, $R^* = 1$

$S^* = \overline{S}$ $\quad Q = 1, \overline{Q} = 0$

$R^* = \overline{R}$

when,

CLK = 1, S = 1, R = 1

input to SR Latch is,

$S^* = 0$; $R^* = 0$

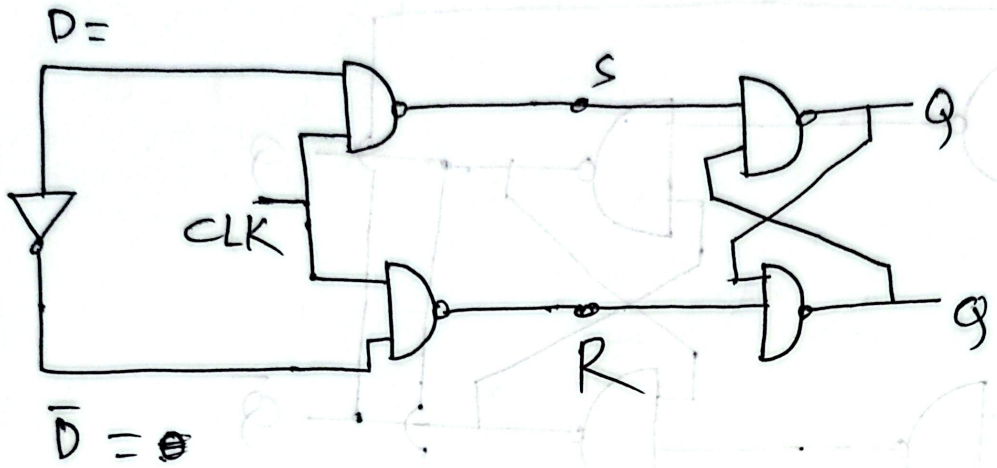
$Q = 1, \overline{Q} = 1$

$Q \neq \overline{Q}$ {undefined condition}

Truth table

CLK	S	R	Q_{n+1}	$\overline{Q_{n+1}}$
0	x	x	Q_n	$\overline{Q_n}$ (memory)
1	0	1	0	1
1	1	0	1	0
1	0	0	Q_n	$\overline{Q_n}$ (memory)
1	1	1	1	1 (invalid)

D= Flip-Flop

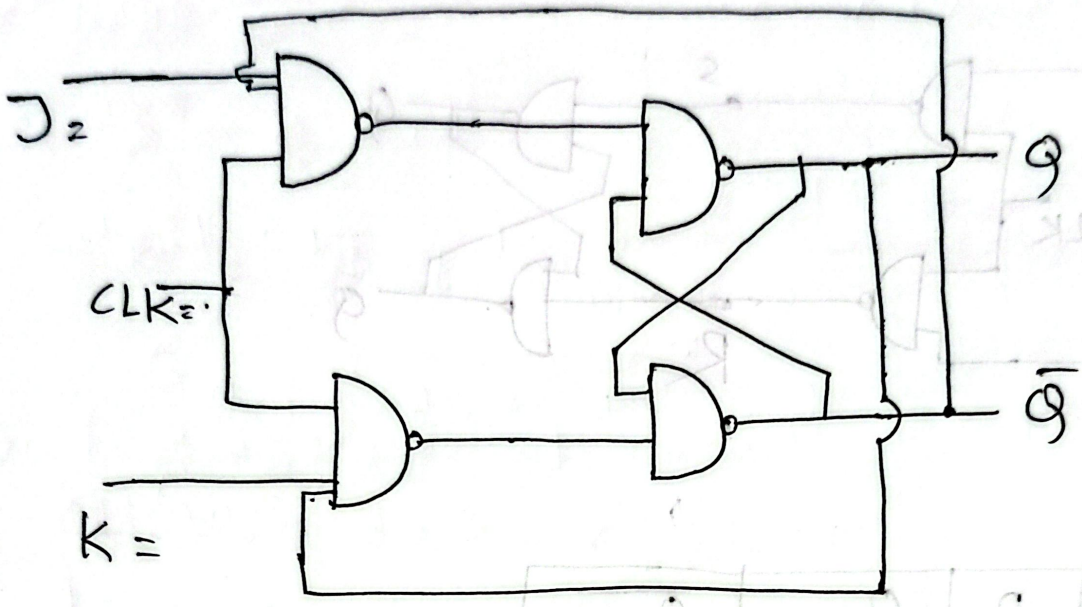


CLK	D	Q_{n+1}	$\overline{Q_{n+1}}$
0	x	Q_n	$\overline{Q_n}$ (memory)
1	0	0	1
1	1	1	0

JK-Flip-Flop

Lecture - 10

21.05.2025



Truth Tables

(SR Latch)

S	R	Q	$\bar{Q}$
0	1	1	0
1	0	0	1

(SR Flip-Flop)

S	R	Q	$\bar{Q}$
0	1	0	1
1	0	1	0

When,

CLK = 0, J = x, K = x

$Q_{n+1} = Q_n$ (present = previous)

$\overline{Q_{n+1}} = \overline{Q_n}$

When,

CLK = 1, J = 0, K = 1

Output, $Q = 0$, $\bar{Q} = 1$

When,

CLK = 1, J = 1, K = 0

Output $Q = 1$, $\bar{Q} = 0$

When,

CLK = 1, J = 0, K = 0

Output,

$Q_{n+1} = Q_n$, $\overline{Q_{n+1}} = \bar{Q}_n$

When,

CLK = 1, J = 1, K = 1

Let's assume,

$Q_n = 0$

$\bar{Q}_n = 1$

Output,

$Q_{n+1} = 1$

$\overline{Q_{n+1}} = 0$

For NAND gate-1, Inputs are, 1, 1, 1

$\therefore$ Output = 0, S = 0

For NAND gate-2, Inputs are, 0, 1, 1

Output = 1, R = 1

Now,

$Q_{n+1} = 1$, $\overline{Q_{n+1}} = 0$

For NAND gate-1

Inputs = 0, 1, 1

∴ Output = 1, S = 1

For NAND gate-2

inputs = 1, 1, 1

Output = 0, R = 0

Q_n (0, 1, 0, 1, ...)

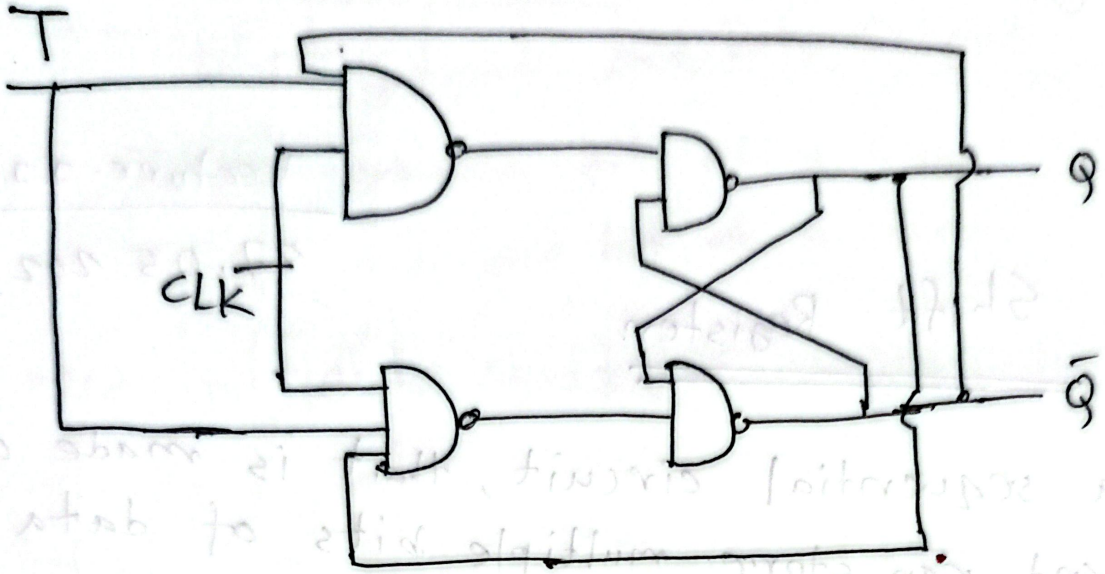
$\overline{Q_n}$ (1, 0, 1, 0, ...)

Output ~~keeps~~ Q and $\bar{Q}$ keeps changing from 0 to 1, 1 to 0 (toggling)

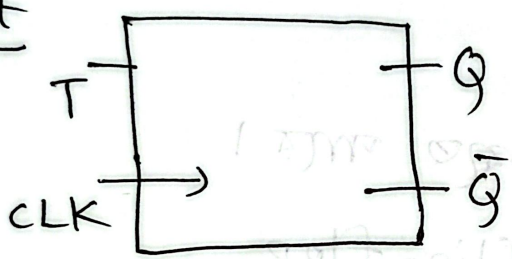
Truth Table

CLK	J	K	Q_{n+1}	$\overline{Q_{n+1}}$	state
0	X	X	Q_n	$\overline{Q_n}$	memory
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	0	0	Q_n	$\overline{Q_n}$	memory
1	1	1	0, 1, 0, 1	1, 0, 1, 0	toggle

T flip-flop



ব্লক circuit



Truth Table

CLK	T	Q_{n+1}	$\overline{Q_{n+1}}$
0	X	Q_n	$\overline{Q_n}$
1	0	Q_n	$\overline{Q_n}$
1	1	0, 1, 0, 1	1, 0, 1, 0

(toggle)

Q. Synchronus vs Asynchronous

Q. Toggling

Lecture-11

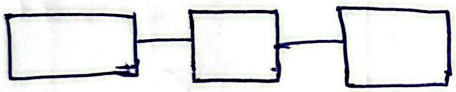
27.05.2025

Shift Register

⇒ Shift is a sequential circuit, that is made of flip-flops and can store multiple bits of data basically D flip-flop দিয়ে.

⇒ multiple data store করতে পারে।

⇒ 1 bit data use করে flip-flop



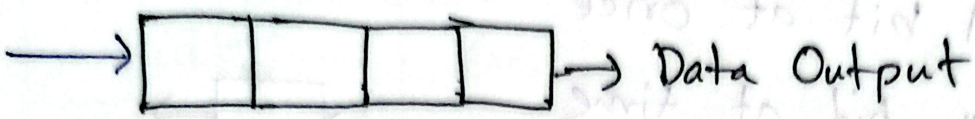
= 3 bit data store করতে পারে।

A n bit registor is made of n number of flip-flop and can store n bit of data.

মানে 4 টা বিট সম্বলিত flip-flop 4 bit টা store করবে।

Types:

i) Serial In Serial Out (SISO)



Input → one bit at a time

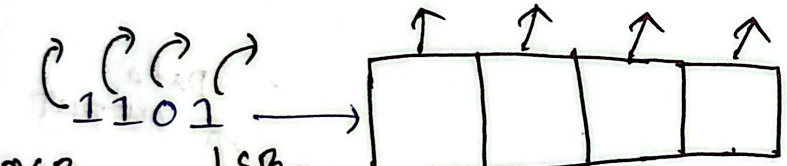
Output → one bit at a time.

→ used in right shifting or left shifting

ii) Serial In Parallel Out (SIPO)

Input → one bit at a time

Output → all bits simultaneously



{ input একসাথে হবে কিন্তু output আলাদা আলাদা ভাবে পাওয়া যাবে

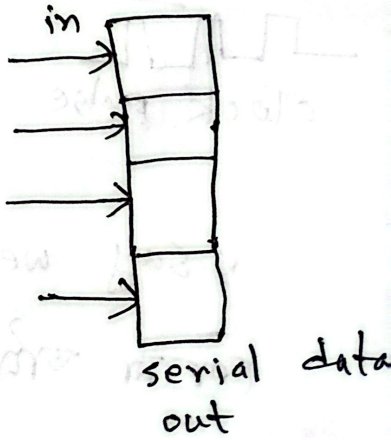
used in serial to parallel conversion

iii) Parallel In Serial Out (PISO)

Input → all bit at a time

Output → one bit at a time

{

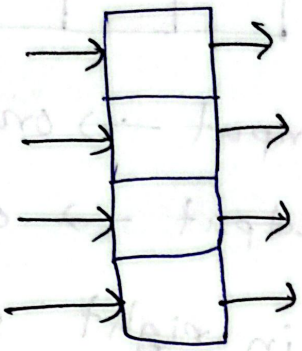


used in parallel to serial conversion

iv) Parallel In Parallel Out (PIPO)

Input → All bit at once

Output → All bit at time

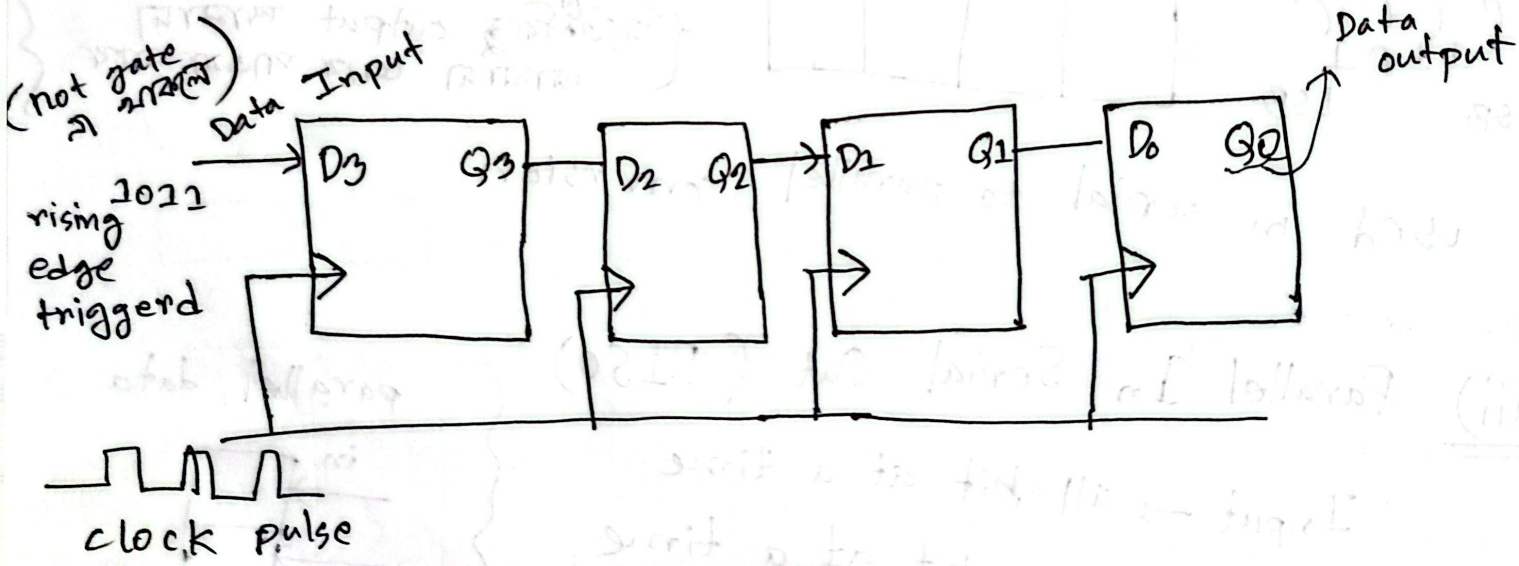


No

#^Shifting operation in PIPO, only used as storage.

SISO register

4 bit SISO



* Say we want to right shift 1011

আগের ডাটা আমরা ধরে নিচ্ছি all 0 তে আছে।

প্রথমবার 1 bit Q_3 তে 1 bit store হবে।

CLK	Q_3	Q_2	Q_1	Q_0
Initial Stage	0	0	0	0
1st rising edge / clock pulse	1	0	0	0
2nd clock rising edge (↑)	1	1	0	0
3rd clock (↑)	0	1	1	0
4th clock (↑)	1	0	1	1
5th Clock (↑)	0	1	0	1
6th clock (↑)	0	0	0	1

data read করতে বা output নিতে চাইলে!

for a siso registor to load n bit data, n num of clock pulse

একটা SISO data load করতে হলে circuit এর truth table store করতে ~~হবে~~ read/output পাইতে লাগবে 2n-1

1021 → input 4 bit

output → 8-1

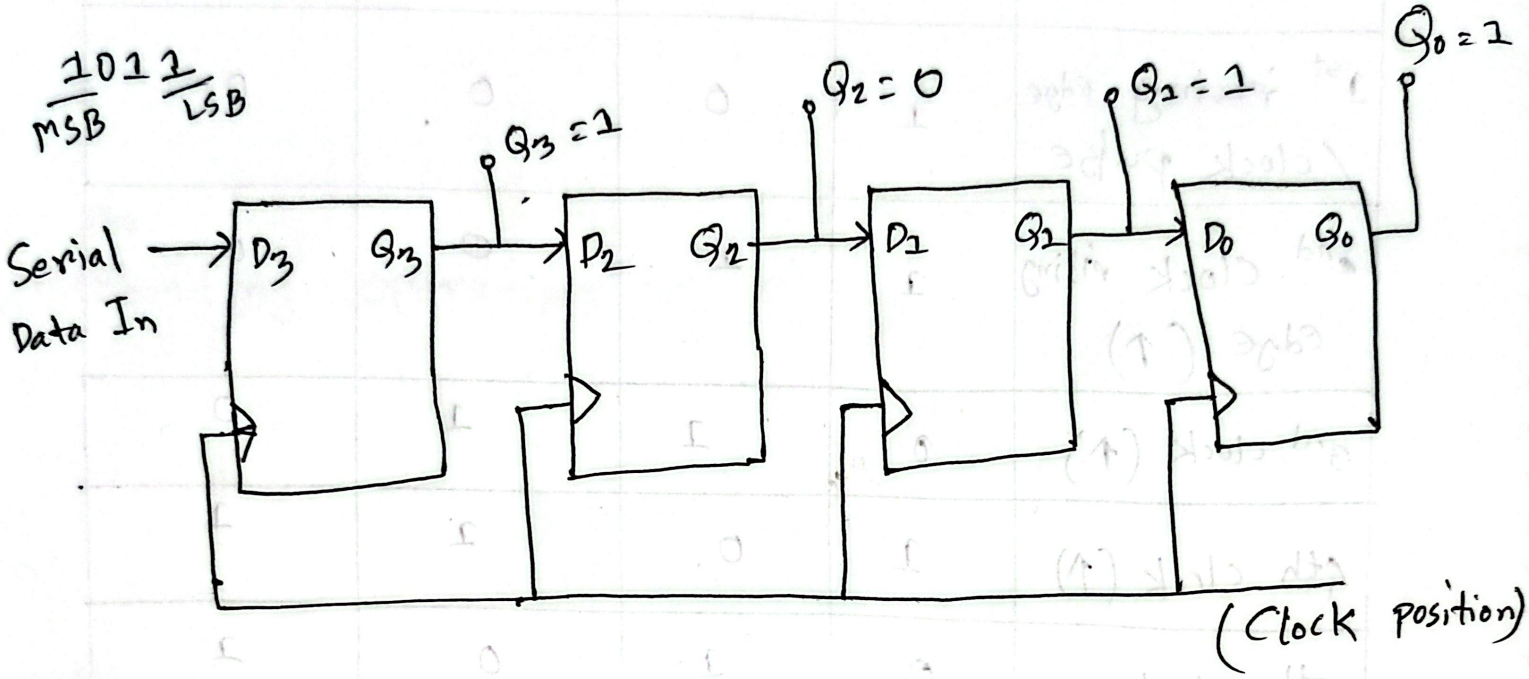
→ 7

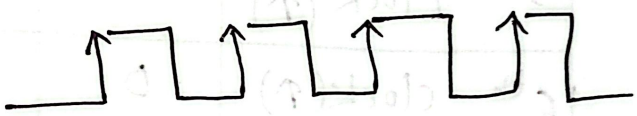
Shift registor

Lecture-12

28.05.2025

⊛ Serial In Parallel Out (SIPO) registor :





rising edge এলে change হবে

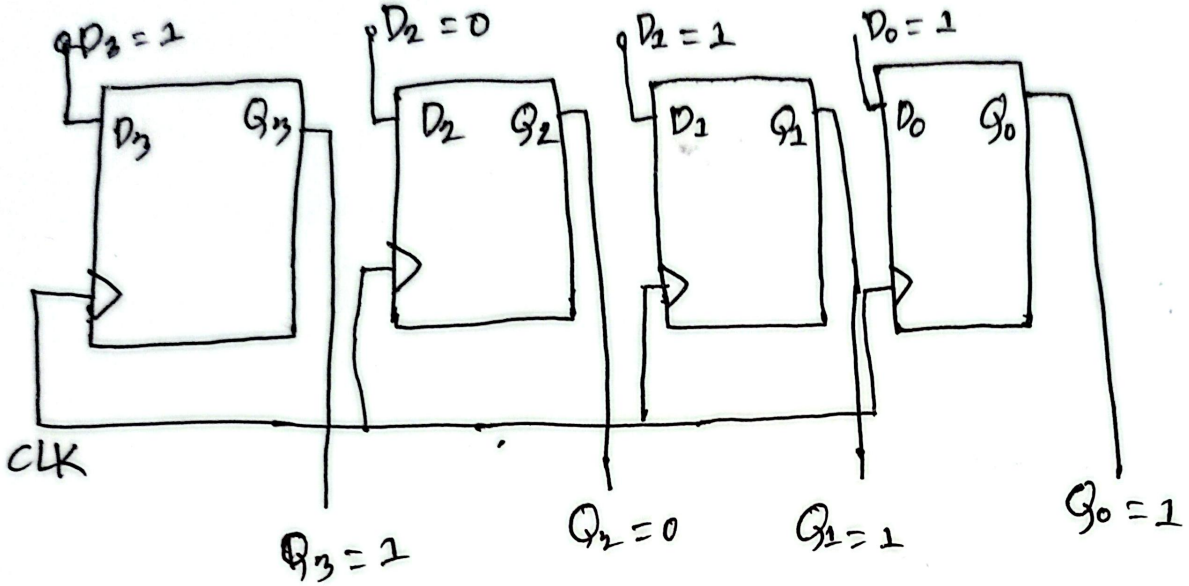
Truth Table:

CLK	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1st CLK (↑)	1	0	0	0
2nd CLK (↑)	1	1	0	0
3rd CLK (↑)	0	1	1	0
4th CLK (↑)	1	0	1	1

Note: In case of a n bit SIPO register n numbers of clock pulse are needed to load and store/read the data.

(*) Difference between SISO and SIPO

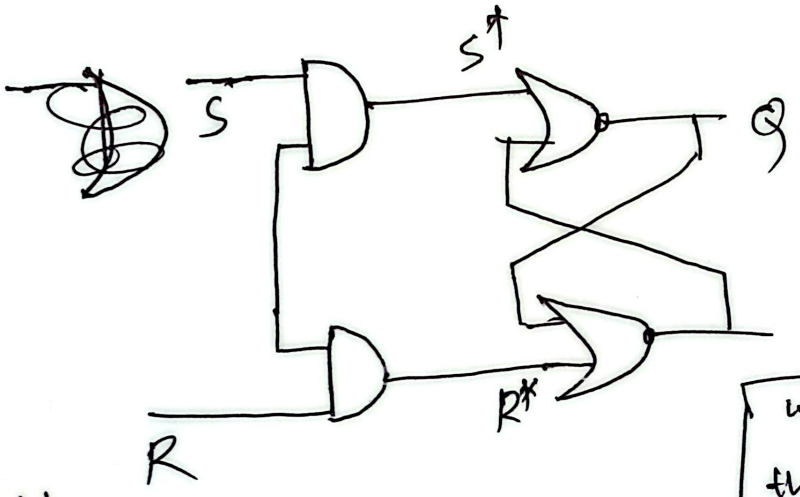
Parallel In Parallel Out (PIPO) register.



No shifting operation in PIPO can only store Data.

Note: In case of a PIPO register only one clock pulse is needed read/store data.

Extara : SR Flip-Flop using NOR Latch



truth table.

SR flip-flop same

when, $S = 0$, $R = 1$, CLK = 1

then $S^* = 0$, $R^* = 1$

$Q = 0$, $\bar{Q} = 1$