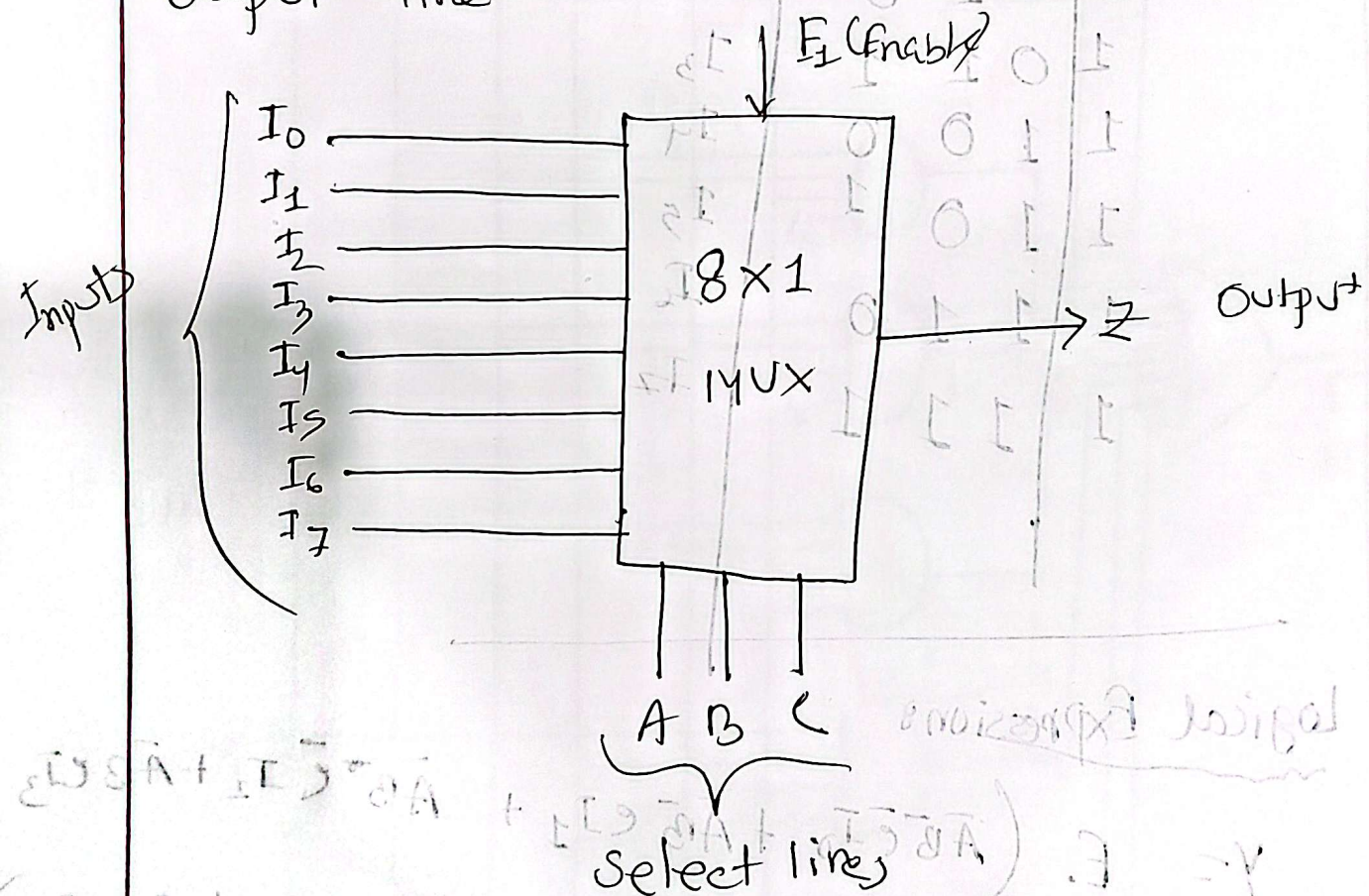


Chapter-9

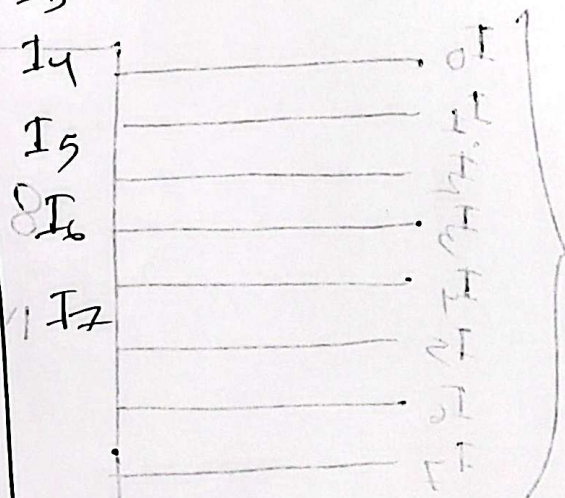
सबसे महत्वपूर्ण

☐ 8:1 Multiplexer या 8:1 Multiplexer is a combinational circuit that takes one of 8 input lines and forward it to a single output line based on the select lines.



Truth table

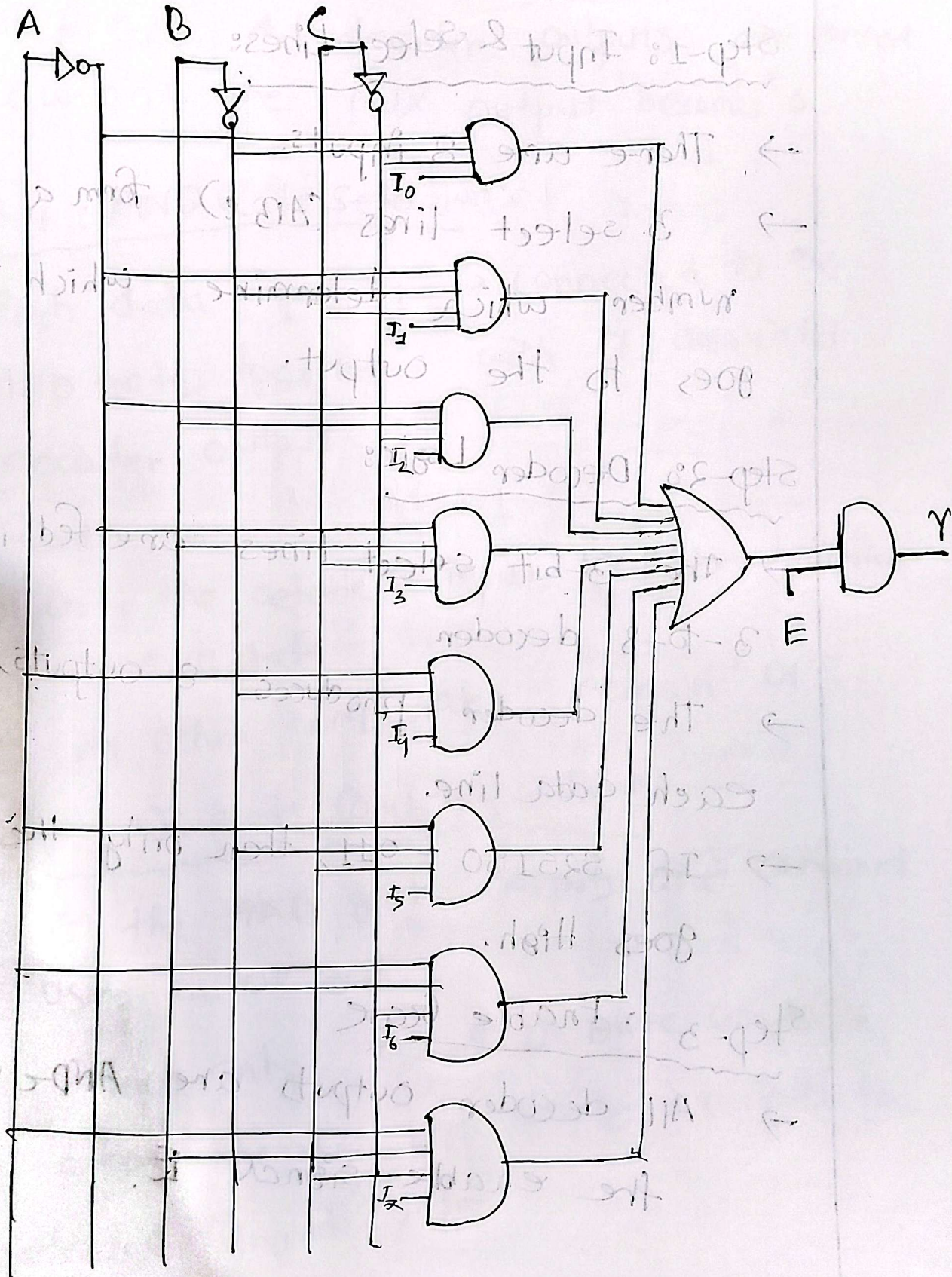
E	A	B	C	Y
0	x	x	x	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Logical Expression

$$Y = E \cdot (\bar{A}\bar{B}\bar{C}I_0 + \bar{A}\bar{B}CI_1 + \bar{A}B\bar{C}I_2 + \bar{A}BCI_3 + AB\bar{C}I_4 + AB\bar{C}I_5 + ABCI_6 + ABCI_7)$$

Circuit:



Working Principles

Step-1: Input & Select Lines

- There are 8 inputs.
- 3 select lines (A, B, C) form a 3-bit number which determine which input I_i goes to the output.

Step-2: Decoder Logic

- The 3-bit select lines are fed into a 3-to-8 decoder.
- The decoder produces 8 outputs, one for each data line.
- If $S_2 S_1 S_0 = 011$, then only the 3rd line goes High.

Step-3: Enable Gate

- All decoder outputs are ANDed with the enable signal E.

→ If $E = 1$, the decoder works normally.

→ If $E = 0$, All decoder outputs are forced Low and the MUX output becomes 0.

Step-4 : AND Gate Selection

→ Each data input I_i is connected to an AND gate, together with its associated decoder output.

→ Result:

→ Only the selected input is passed through its AND gate.

→ All other AND gates remain OFF.

Step-5 : OR gate Combines Inputs

→ All the AND gate outputs are combined using an OR gate.

→ Since only one AND gate can be on at a time, the OR gate simply outputs the selected input value.

Step 6: Final Outputs

→ Output $Y = \text{the selected input } I_i$, if $\bar{E} = 1$

→ Output $Y = 0$, if $\bar{E} = 0$

Operation for 74151 IC

1. Case 1: MUX Disabled ($\bar{E} = 1$)

→ When $\bar{E} = 1$, the MUX is disabled.

→ Output $Z = 0$

→ Inverted Output $\bar{Z} = 1$

Case 2: MUX Enabled ($\bar{E} = 0$)

→ When $\bar{E} = 0$, the MUX is active.

→ The 3-bit select ~~line~~ code picks one of 8 inputs.

* $Z = \text{Selected Input}$

* $\bar{Z} = \text{Inverted Selected Input}$

Output Stages

- The OR gate combines all ~~AND~~ AND gate outputs to produce Z .
- An inverter generates $\bar{Z}$.

Example:

MUX Enabled ($\bar{E}=0$), $S_2 S_1 S_0 = 011$

→ Selected Input $= I_3$

→ Outputs:

$$\begin{aligned} Z &= I_3 \\ \bar{Z} &= \bar{I_3} \end{aligned}$$

MUX disabled ($\bar{E}=1$), Any select Input

→ Output:

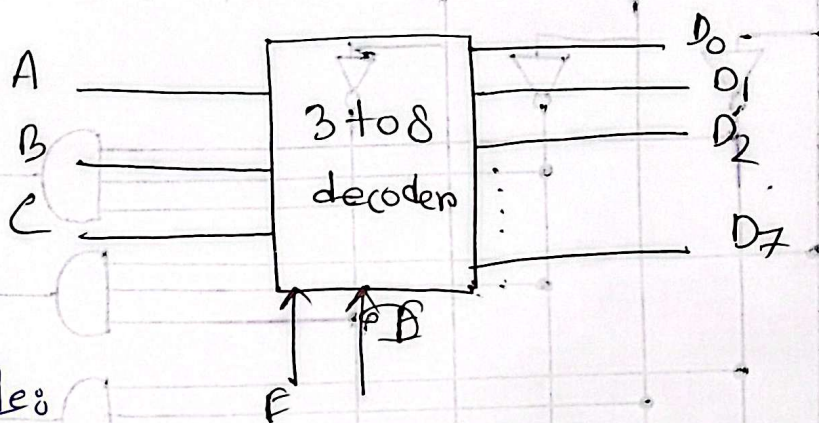
$$Z = 0$$

$$\bar{Z} = 1$$

→ Binary to Octal decoder

65

3 to 8 decoder / 1 of 8 decoder:



Truth table:

	C	B	A	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
0	x	x	x	0	0	0	0	0	0	0	0

Circuit
A

B

Enable for 8 input lines

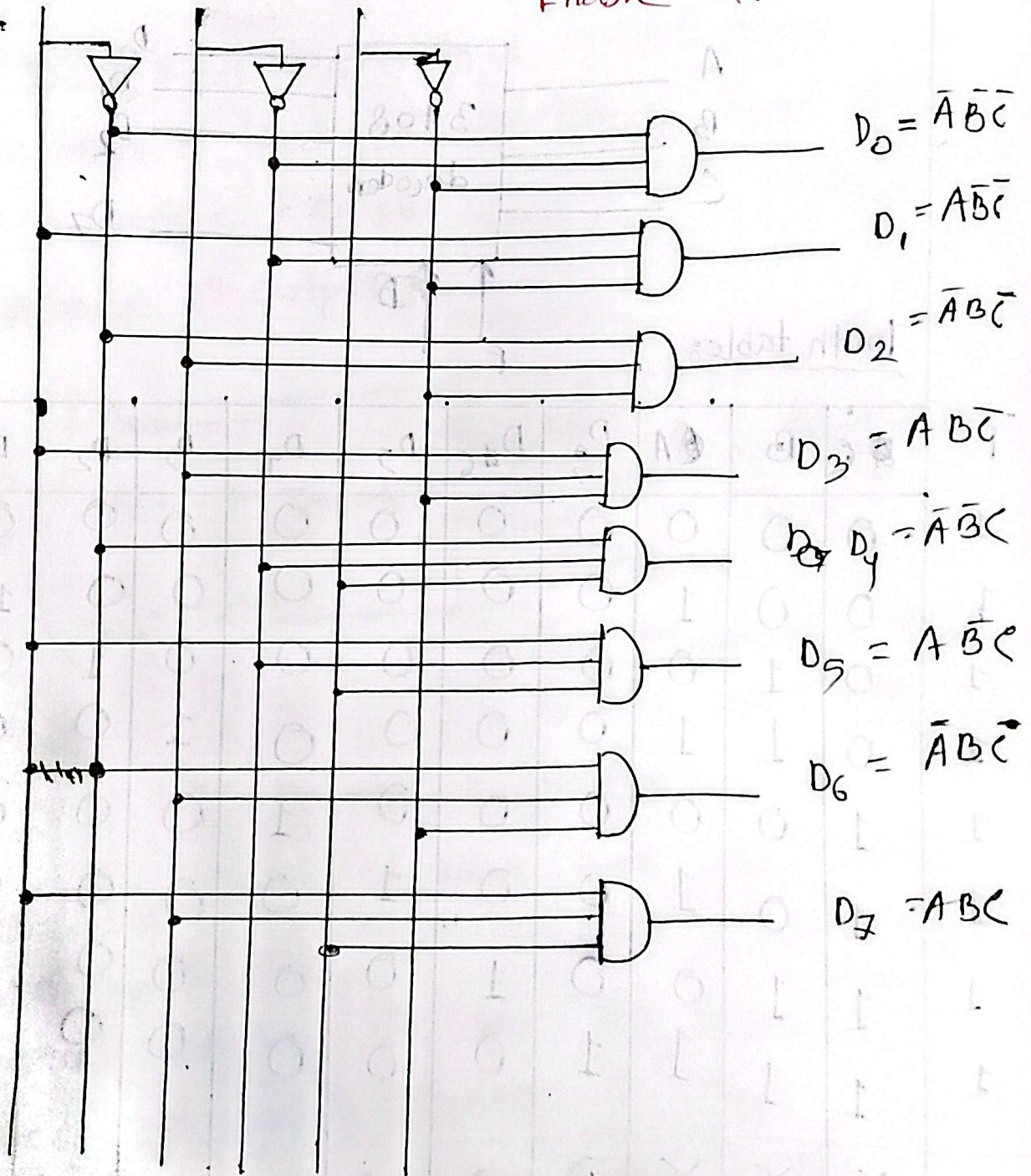


Fig. 1 of 8 / 3x8 decoder.

3x8 74138 Decoder ICs

→ 74ALS138 is an integrated chip, which is 3

to 8 decoder

→ It has total 16 pins

→ It has three inputs pins, three Enable input pins and eight Active Low output pins

Inputs $\bar{E}_1, \bar{E}_2, E_3$ are ~~sp~~ separate enable inputs that are combined in the AND gate.

In order to enable the output NAND-gate to respond to the input code at A_2, A_1, A_0 this AND gate output must be High. This

Occur only when $\bar{E}_1 = \bar{E}_2 = 0$ and $E_3 = 1$.
(Truth Table)

$\bar{E}_1$	$\bar{E}_2$	E_3	Output
0	0	1	Respond to input code $A_2 A_1 A_0$
1	X	X	Disabled - all High
X	1	X	Disabled - all
X	X	0	Disabled - all

Circuito

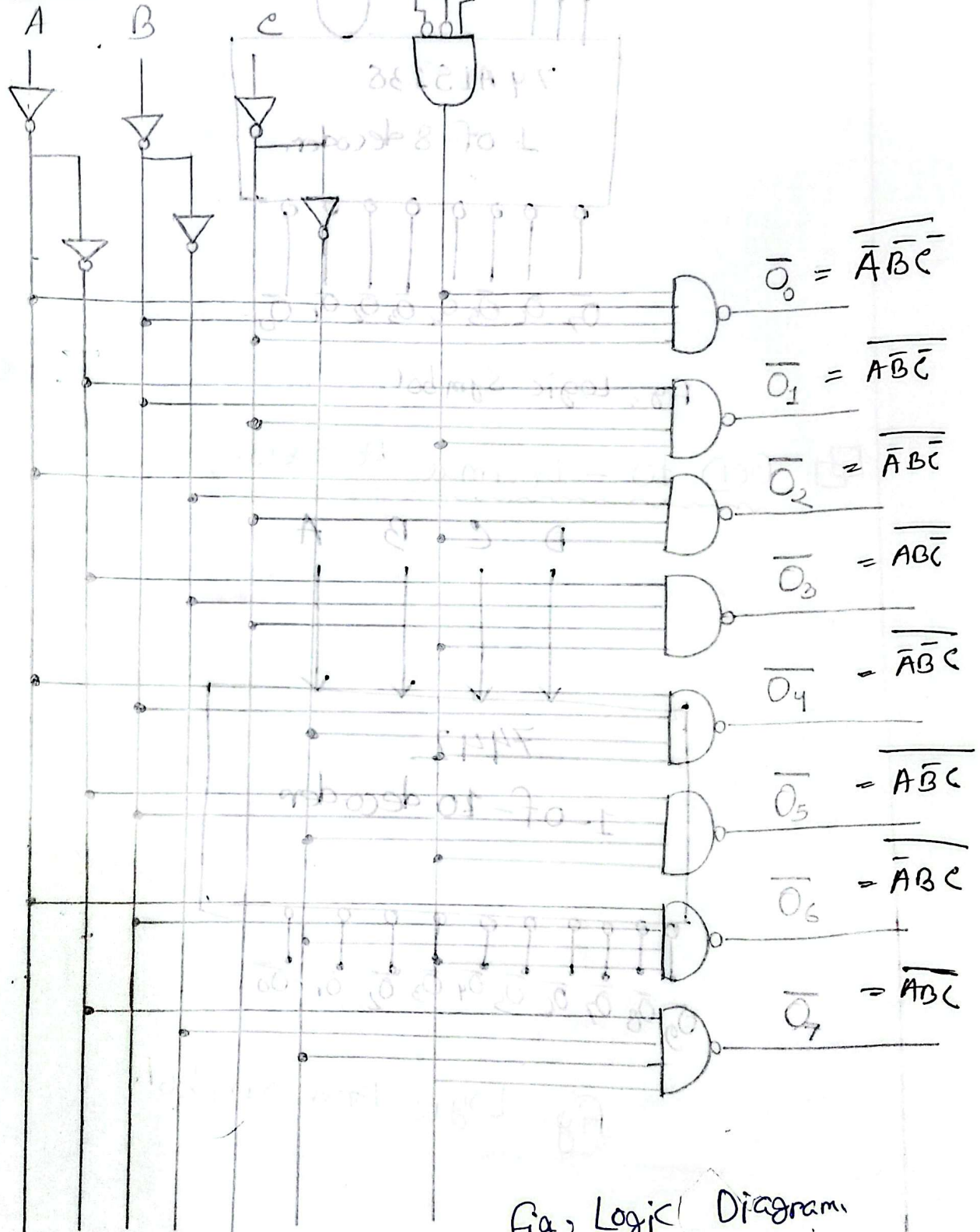


Fig. Logic Diagram

10

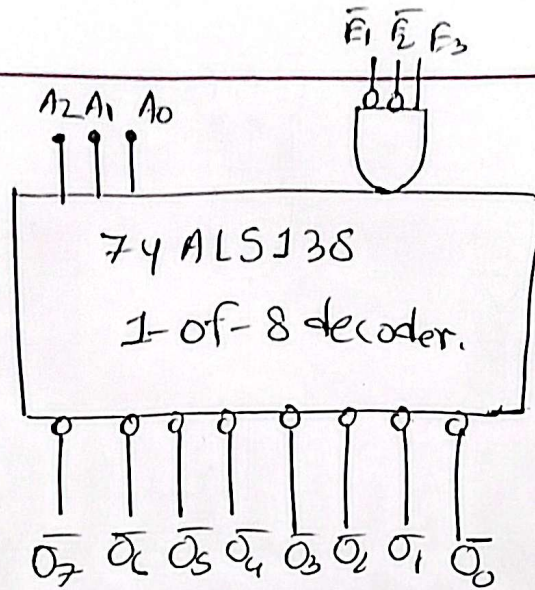


Fig. Logic Symbol.

BCD to - Decimal Decoders

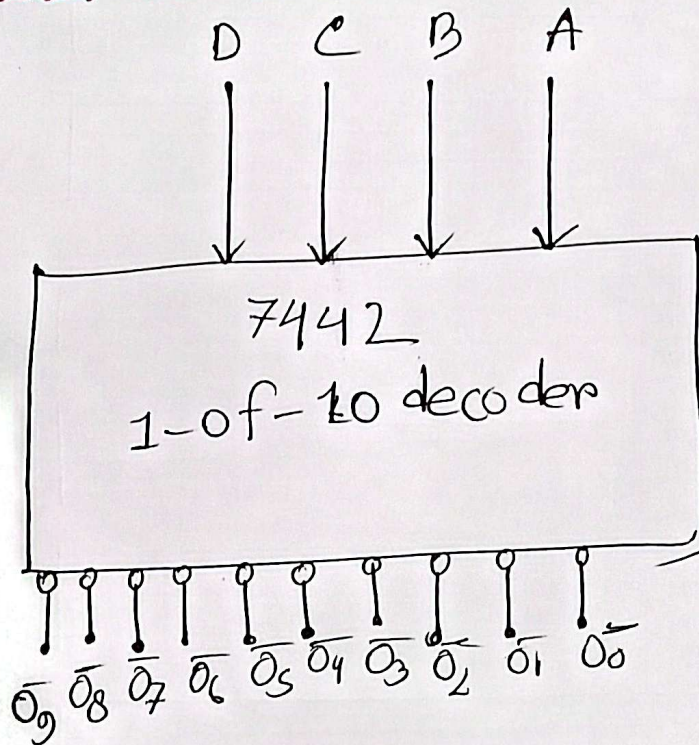


Fig. Logic Symbol.

Truth table:

D	C	B	A	Active Output
0	0	0	0	$\overline{O_0}$
0	0	0	1	$\overline{O_1}$
0	0	1	0	$\overline{O_2}$
0	0	1	1	$\overline{O_3}$
0	1	0	0	$\overline{O_4}$
0	1	0	1	$\overline{O_5}$
0	1	1	0	$\overline{O_6}$
0	1	1	1	$\overline{O_7}$
1	0	0	0	$\overline{O_8}$
1	0	0	1	$\overline{O_9}$
1	0	1	0	None
1	0	1	1	None
1	1	0	0	None
1	1	0	1	None
1	1	1	0	None
1	1	1	1	None

0 1 2 3 4

▽ ▽ ▽ ▽ ▽

— — — — —

▽ ▽ ▽ ▽

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

Handwritten notes in red ink at the bottom of the page.

It is also available as a 74LS42 and a 74HC42.

Each output goes low only when its corresponding BCD input is applied. For example, $\overline{O}_5$ will go low only when input DCBA = 0101. This decoder will also be referred to as a 4 to 10 decoder or 1 of 10 decoder.

BCD to Decimal Decoder/Driver: The TTL 7445 is a BCD-to-Decimal decoder/driver. The term driver is added to its description because this IC has open-collector outputs that can operate at higher current and voltage limits than a normal TTL output. The 7445's outputs can sink up to 80 mA in the Low state, and they can be pulled up to 30V in the High state. This makes them suitable for directly driving loads such as indicator LEDs or lamps, relays, or DC motors.

B CD-to-7 segment decoder/driver

A BCD-to-7-segment decoder/driver is used to take a four bit BCD input and provide the outputs that will pass current through the appropriate segments to display the decimal to digit.

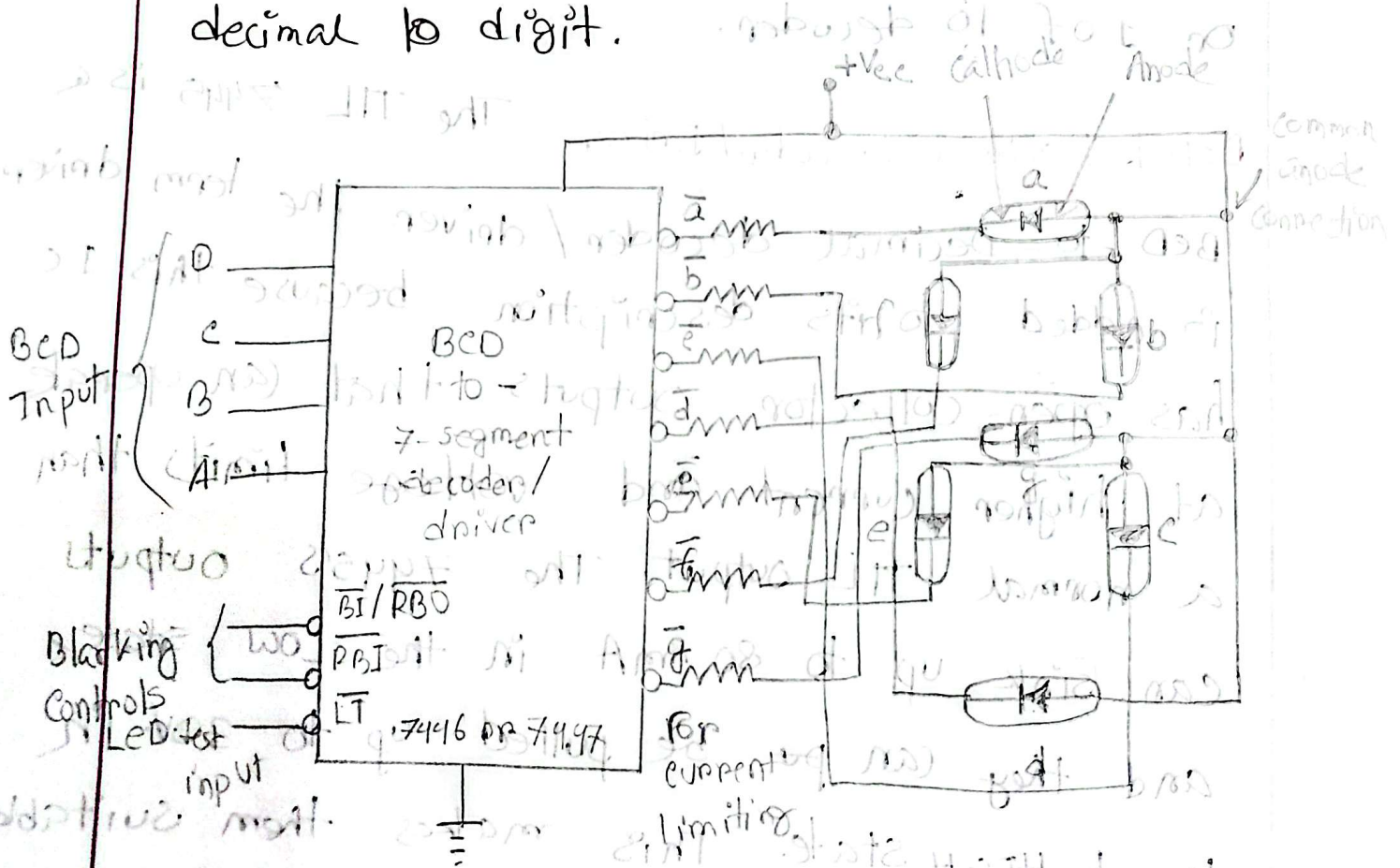


fig: BCD-to-7-segment decoder/driver driving a common-anode 7-segment LED Display.

The figure shows a BCD-to-7-segment decoder/driver (TTL 7446 or 7447) being used to drive a 7-segment LED readout. Each segment consists of an LED. Diodes are solid-state devices that allow current to flow through them in one direction, but block ^{flow into} the other direction whenever the anode of an LED is more positive than the cathode by approximately 2V, the LED will light up.

Liquid Crystal Display (LCD)

How it works:

A liquid crystal display controls the reflection of available light. The available light may simply be ambient light such as sunlight or normal room lighting. Reflective LCDs use ambient light.

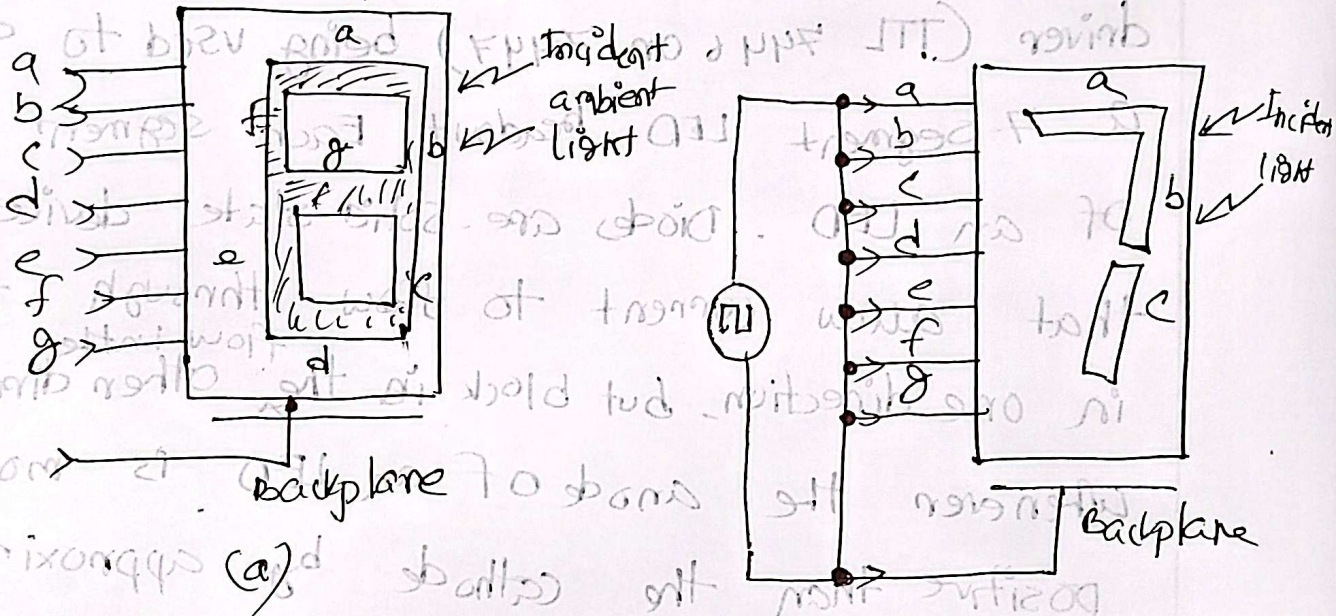


Fig: Liquid crystal display : (a) basic arrangement

(b) applying a voltage between the segment and the backplane turns ON the segment. Zero voltage turns the segment OFF.

Working:

1. Light comes from Outside: LEDs don't glow by themselves. They use ambient light.

2. Two Main Parts:

+ Segment

+ Backplane

3. OFF segment (NO Display):

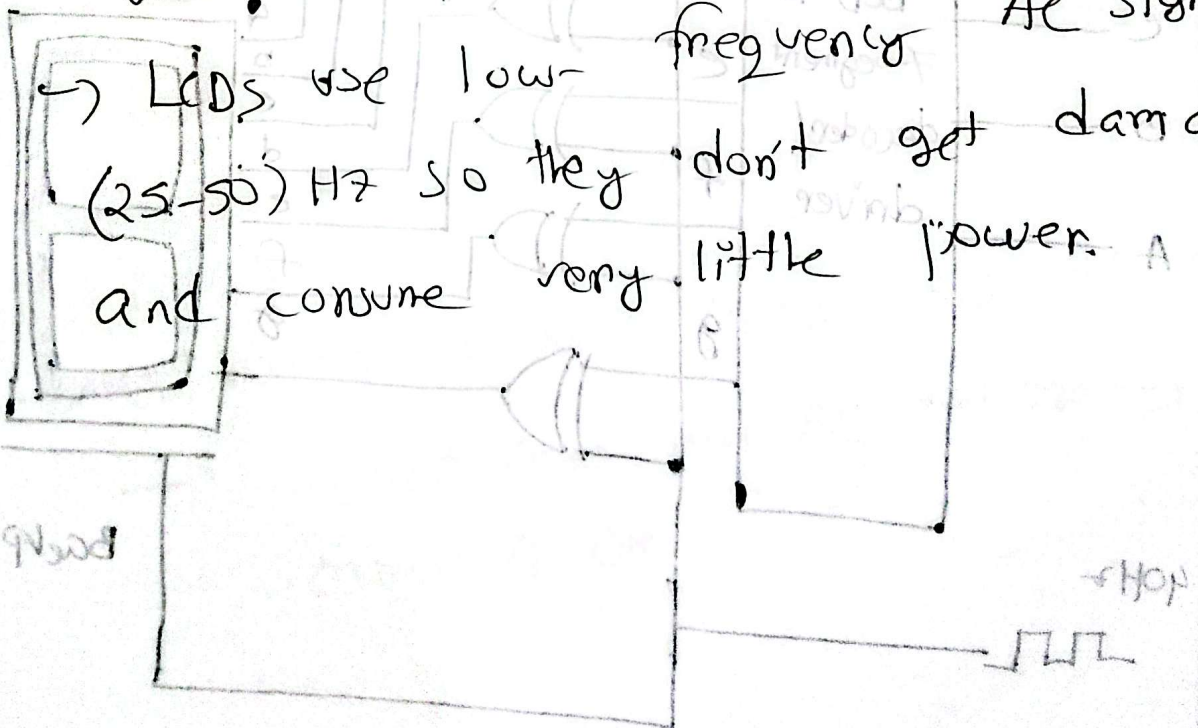
→ when both segment and backplane have the same voltage, no current flows → light passes through → looks white or blank.

4. ON segment (Visible Display):

→ when different voltage is applied between segment and backplane → the crystal twists → blocks light → appears black.

5. AC signal used (NOT DC):

→ LEDs use low frequency AC signals (25-50) Hz so they don't get damaged and consume very little power.

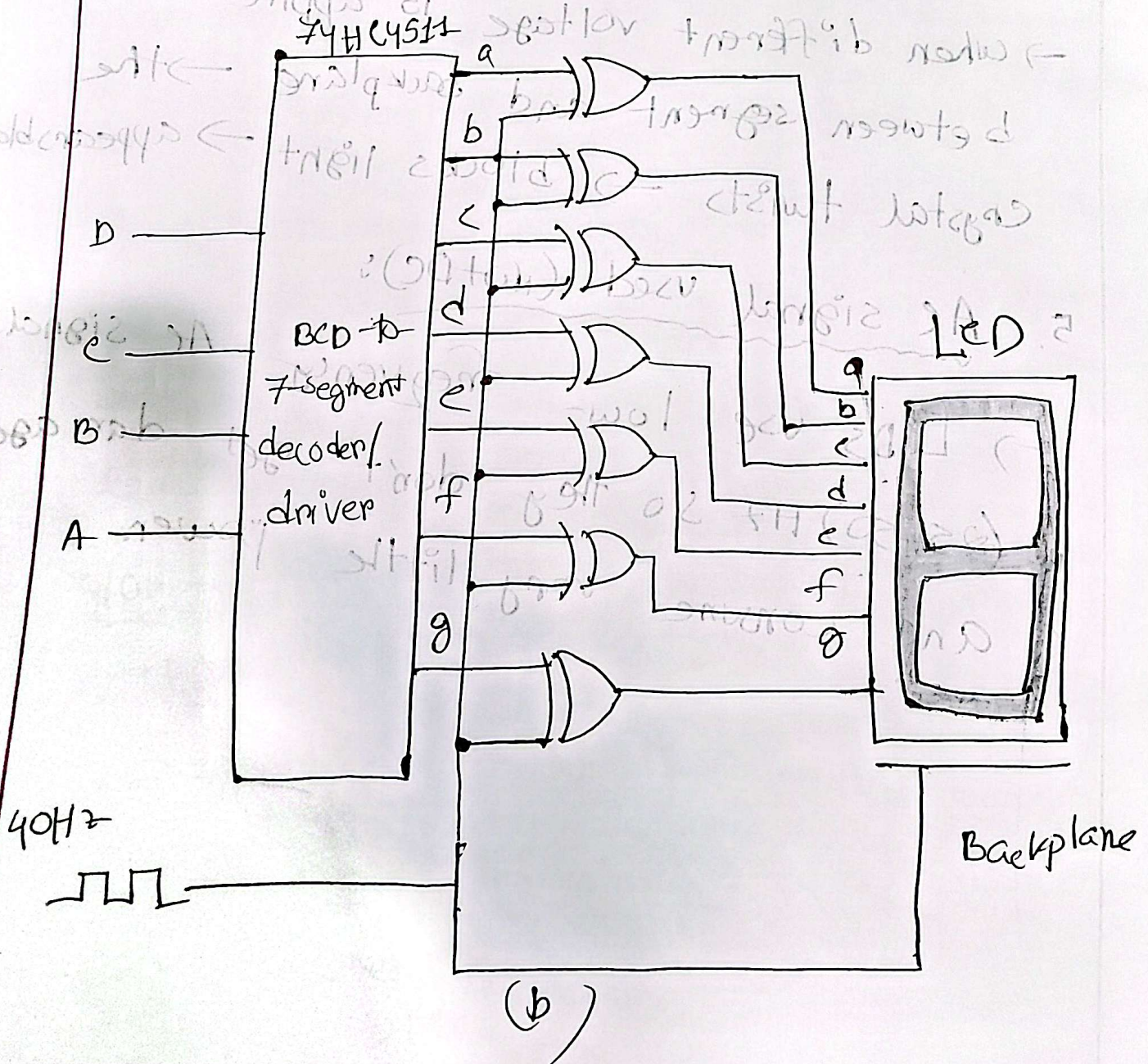
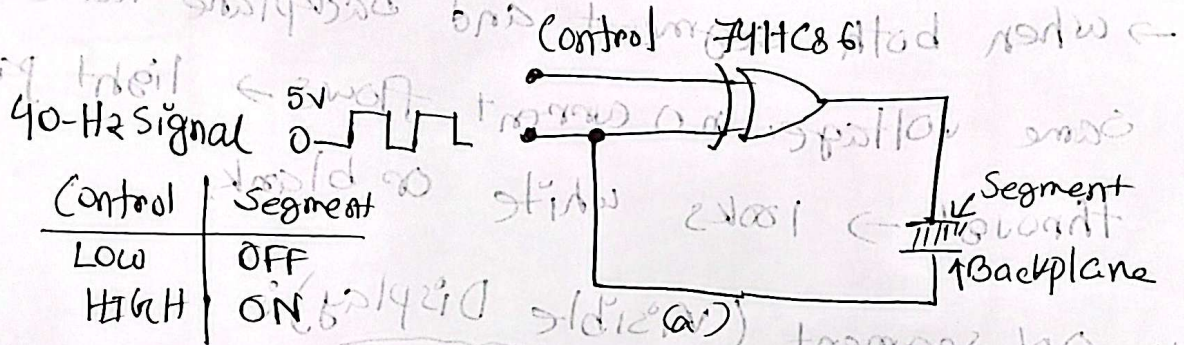


(a)

78

method

driving a

Driving an LCD (b) ~~BCD to~~ 7-segment driver

Step 1: Apply AC signal (40Hz):

→ A low signal frequency AC voltage is generated.

→ This signal is applied to the LED backplane.

Step 2: Use XOR gates to control segments

→ Each ~~LED~~ LED segment gets its own XOR gate

→ One input of the XOR = AC signal

→ Other input = Control signal

→ XOR gate output decide Segment status

Step 3: XOR gate output

→ Control = 0 → XOR outputs the same as backplane → NO voltage difference → Segment is OFF.

→ Control = 1 → XOR inverts the AC signal
→ segment sees full AC voltage → Segment is ON

Step-4: Connect BCD to - 7 segment Decoder

→ A BCD (Binary Coded Decimal) to 7-segment decoder (like 74LS47) gives signals to the XOR gates.

→ Each number (0-9) turns ON the required segments using control signals.

Step-5: Display works without DC:

→ This method ensures no constant DC

between the segment and the backplane.

→ Protects the LED from damage and

increase lifespan.

→ 20-21 → Final (20-21) → 40

81

Ex Use 74LS138 ICs to design a 1-of-32 decoder.

⇒ The 74138 is a 3-to-8 line decoder (1-of-8 decoder), but we can combine multiple 74138 ICs to create a larger 1-of-32 decoder.

Step 1: Understand the 74138 decoder:

The 74138 has:

→ 3 input select lines (A_0, A_1, A_2) → 8 possible outputs ($\bar{O}_0 - \bar{O}_7$)

→ 3 enable input pins ($\bar{E}_1, \bar{E}_2, E_3$) → Must be correctly set for the chip to work.

→ Active-Low outputs ($\bar{O}_0 - \bar{O}_7$ are low when selected).

Step-2: Expanding to a 1-of-32 decoder:

→ Since each 74ALS138 can only decode 3 bits, we need to extend it to 5-bit decoding (for 32 outputs).

→ To achieve this, four 74ALS138 ICs (21, 22, 23, 24) are used together.

→ The two extra address bits (A_3 and A_4) decide which IC should be active.

Step-3: Address Connections

→ The lower 3 bits (A_0, A_1, A_2) are connected in parallel across all four ICs.

→ The upper 2 bits (A_3, A_4) are used to control which IC is selected via the enable inputs (E).

Step-4: Enable Logic (selecting an IC)

→ A3 and A4 act as the decoder for the

enable lines:

- * $A_3 = 0, A_4 = 0 \rightarrow Z_1$ active (outputs $\bar{O}_0 - \bar{O}_7$)
- * $A_3 = 0, A_4 = 1 \rightarrow Z_2$ active (outputs $\bar{O}_8 - \bar{O}_{15}$)
- * $A_3 = 1, A_4 = 0 \rightarrow Z_3$ active (outputs $\bar{O}_{16} - \bar{O}_{23}$)
- * $A_3 = 1, A_4 = 1 \rightarrow Z_4$ active (outputs $\bar{O}_{24} - \bar{O}_{31}$)

Step-5: Final Output Mapping:

→ Each IC has eight outputs.

- * $Z_1: \bar{O}_0$ to $\bar{O}_7$
- * $Z_2: \bar{O}_8$ to $\bar{O}_{15}$
- * $Z_3: \bar{O}_{16}$ to $\bar{O}_{23}$
- * $Z_4: \bar{O}_{24}$ to $\bar{O}_{31}$

So, for any given 5-bit input, only one of the 32 outputs ($\bar{O}_0 - \bar{O}_{31}$) will be active (Low).

Logic Circuit (Decoder) 1-of-32

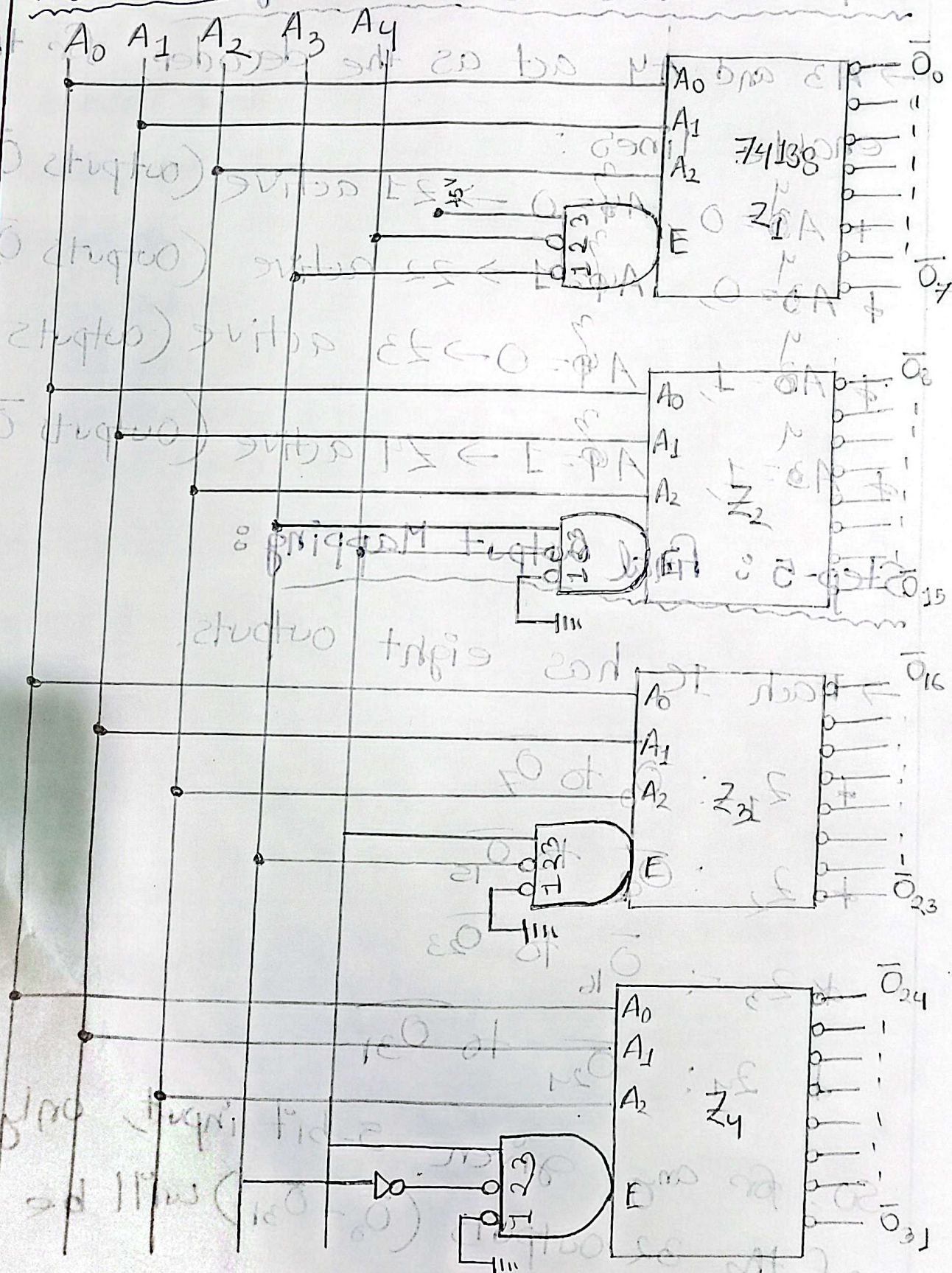


Fig: 1-of-32 decoder.

Example:

If the input is:

$$A_4 A_3 A_2 A_1 A_0 = 10010 (\text{binary}) = 18 (\text{decimal})$$

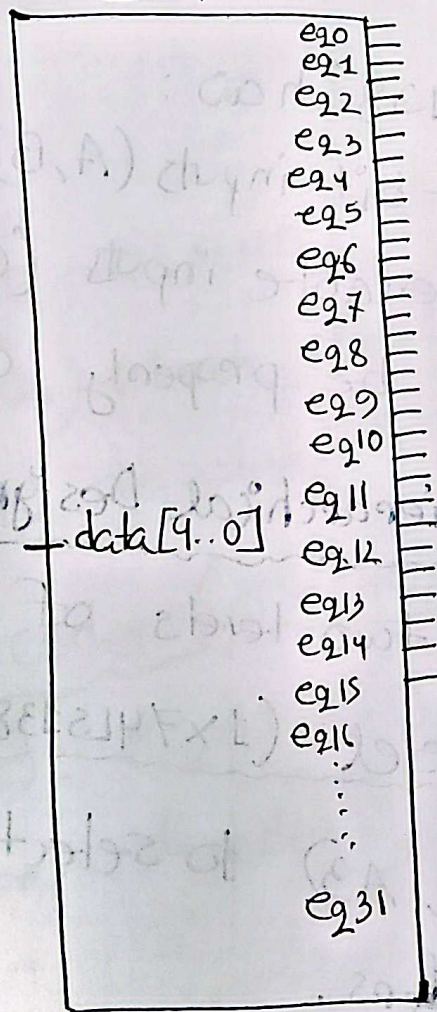
Then,

* $A_4 A_3 = 10 \rightarrow z_3$ is enabled.

* $A_2 A_1 A_0 = 010 \rightarrow 2$ in binary $\rightarrow$ output

$\rightarrow$ 0_{18} is active (Low).

1 of 32



inst2

Fig: 1-of-32 decoder negafuncion.

2 tubes for

86

1-of-64 decoder using 74LS138 IC

⇒ The 74LS138 is a 3-to-8 line decoder (active low outputs). To build a 1-of-64 decoder, we need 8x 74LS138 ICs in a hierarchical structure.

Step-1: Understand the requirements:

→ 6 bit input ($A_5, A_4, A_3, A_2, A_1, A_0$) → 64 possible outputs

→ 74LS138 has:
* 3-bit inputs (A, B, C) → 8 outputs ($Y_0 - Y_7$)
* 3-enable inputs (G_1, G_2A, G_2B) → Must be properly controlled.

Step-2: Hierarchical Design (Two levels):

we use two levels of decoders:

1. First level ($1 \times 74LS138$) → Decodes 3 MSBs (A_5, A_4, A_3) to select one of 8 secondary decoders.

2. Second Level (8x74LS138) → Each decodes 3 LSBs (A_2, A_1, A_0) to produce 8 outputs (total $8 \times 8 = 64$).

Step 3: Circuit Connections:

Primary Decoder (1x74LS138)

→ Inputs (A, B, C): A_5, A_4, A_3 (selects which secondary decoder is active).

→ Enable Pins:

* $G_1 = 1 (VCC)$

* $G_2A = G_2B = 0 (GND)$ enables

→ Outputs ($0-7$): Each

One secondary decoder.

Secondary Decoder (8x74LS138):

→ Inputs (A, B, C): A_2, A_1, A_0 (common to all secondary decoders).

→ Enable Pins: $(Y_0 - Y_2)$
 + G_1 → Controlled by Primary Decoder
 + $G_2A = G_2B = 0$ (GND)

→ Outputs $(Y_0 - Y_7)$: Final 64 outputs $(O_0 - O_{63})$.

Step-4: Truth Table:

A5 A4 A3	Primary Output ($Y_0 - Y_7$)	Enabled Secondary Decoder	Output range
0 0 0	$Y_0 = 0$ (active)	IC1	$\bar{O}_0 - \bar{O}_7$
0 0 1	$Y_1 = 0$ (active)	IC2	$\bar{O}_8 - \bar{O}_{15}$
0 1 0	$Y_2 = 0$ (active)	IC3	$\bar{O}_{16} - \bar{O}_{23}$
0 1 1	$Y_3 = 0$ (")	IC4	$\bar{O}_{24} - \bar{O}_{31}$
1 0 0	$Y_4 = 0$ (")	IC5	$\bar{O}_{32} - \bar{O}_{39}$
1 0 1	$Y_5 = 0$ (")	IC6	$\bar{O}_{40} - \bar{O}_{47}$
1 1 0	$Y_6 = 0$ (")	IC7	$\bar{O}_{48} - \bar{O}_{55}$
1 1 1	$Y_7 = 0$ (")	IC8	$\bar{O}_{56} - \bar{O}_{63}$

Step-5: Circuit:

89

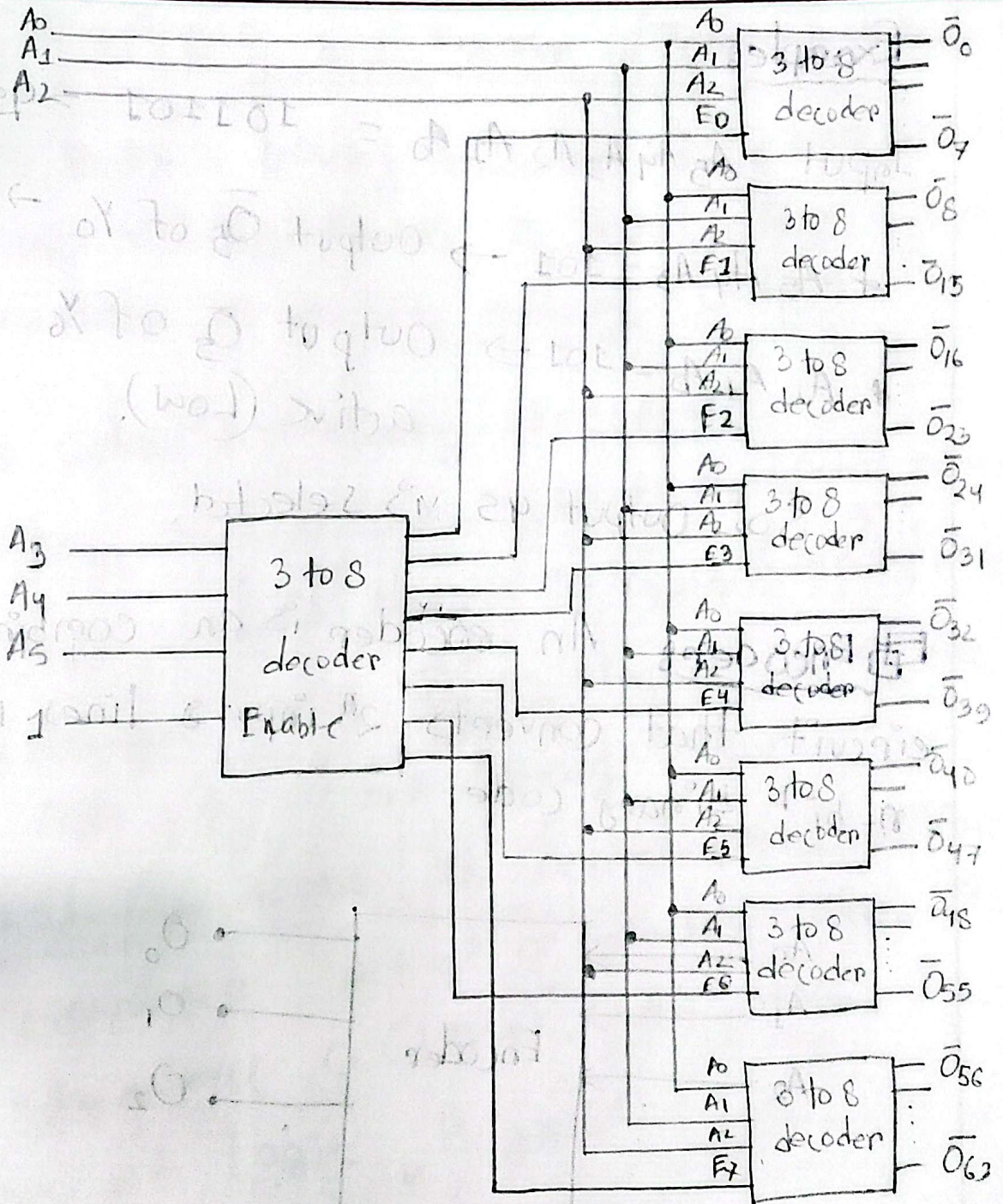


Fig: 10 to 64 decoder circuit

Examples

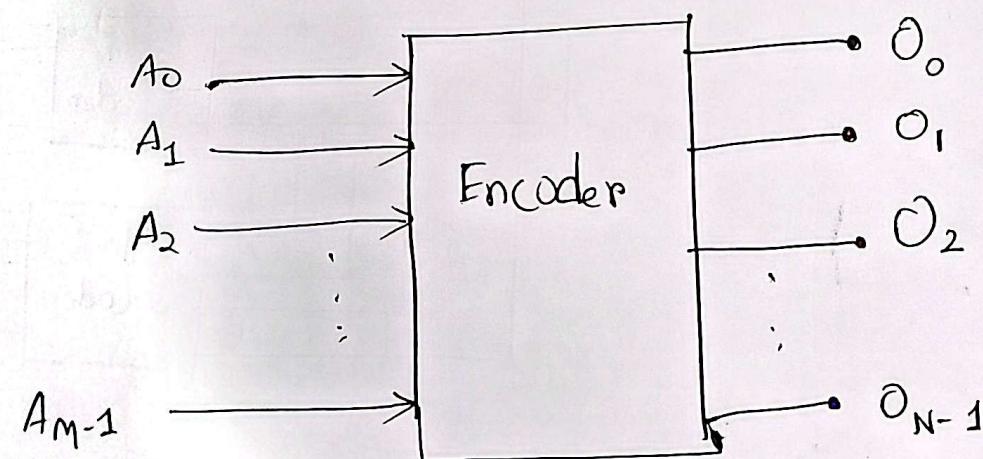
Input $A_5 A_4 A_3 A_2 A_1 A_0 = 101101 = 45$

* $A_5 A_4 A_3 = 101 \rightarrow$ Output $\bar{O}_5$ of $Y_0 \rightarrow$ Enable Y_6

* $A_2 A_1 A_0 = 101 \rightarrow$ Output $\bar{O}_5$ of Y_6 becomes active (Low).

So, Output 45 is selected.

Encoder An encoder is a combinational circuit that converts 2^n inputs lines into an n -bit binary code.



M inputs
only one high
at a time.

N -bit Output Code

Fig : Encoder block diagram,

91

→ Final (2021)
7EC) → Final (1929)
864

Octal to Binary
~~Binary to Octal~~

1 of 8 8:1 Encoder / Octal to Binary Encoders

An 8-to-3 binary encoder is a combinational circuit that takes 8 input lines and produces a 3-bit binary output.

An Octal-to-binary encoder:

* has 8 input lines: A_0 to A_7 (Octal = base 8)

* has 3 ^{output} lines: O_2, O_1, O_0 (Binary = base 2)

* Convert which single input is active into a 3-bit binary output.

Active Low inputs In active-low, a line is considered "active" or selected when it's at logic level 0 (Low). All other inputs are at Logic High (1).

How it works:

When one input is Low, the encoder outputs the binary number of that input's index.

For example:

+ $A_3 = 0$ (others = 1)

→ output = 011 (binary for 3)

+ $A_5 = 0$ (others = 1)

→ output = 101 (binary for 5)

~~A0~~

→ If all inputs are high, the output is 000.

→ $A_0 = 0$ (others = 1) output is also 000

→ So, A_0 doesn't need to go through logic gates, because output is 000 by default when no other input is low.

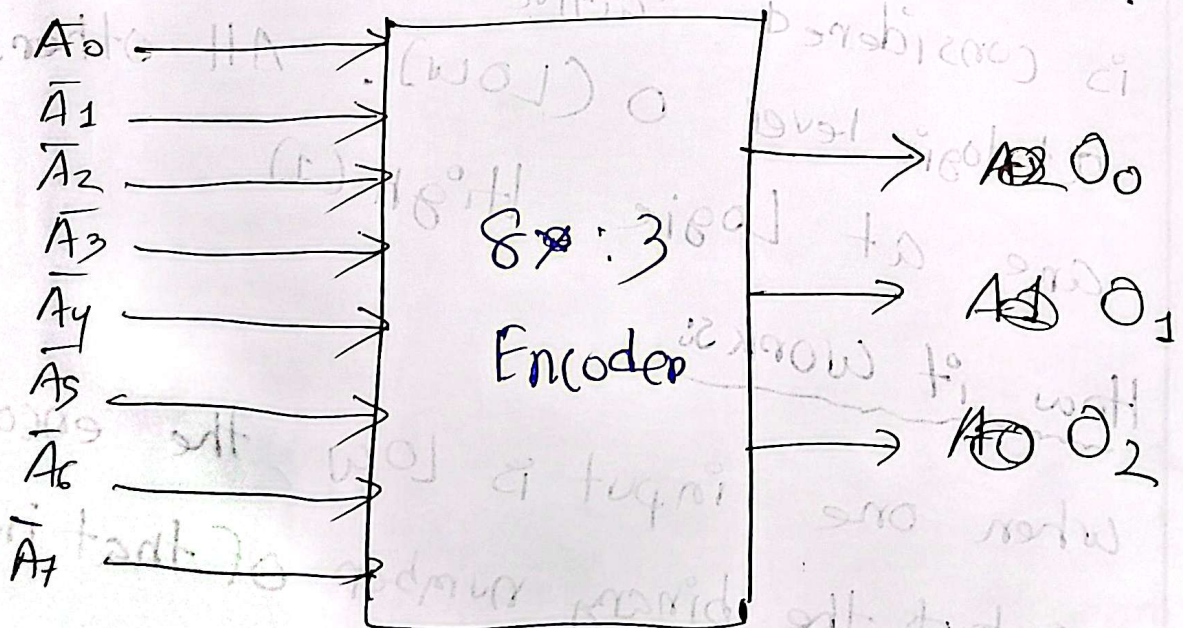


Fig: Block Diagram

Truth Table

$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	Q_2	Q_1	Q_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

Logical Expression for ~~A_2, A_1, A_0~~ Q_2, Q_1, Q_0

$$A_2 = Q_2 = \bar{A}_4 + \bar{A}_5 + \bar{A}_6 + \bar{A}_7$$

$$A_1 = Q_1 = \bar{A}_2 + \bar{A}_3 + \bar{A}_6 + \bar{A}_7$$

$$A_0 = Q_0 = \bar{A}_1 + \bar{A}_3$$

Logic Circuits

Eight inputs

+ Only one low input at a time

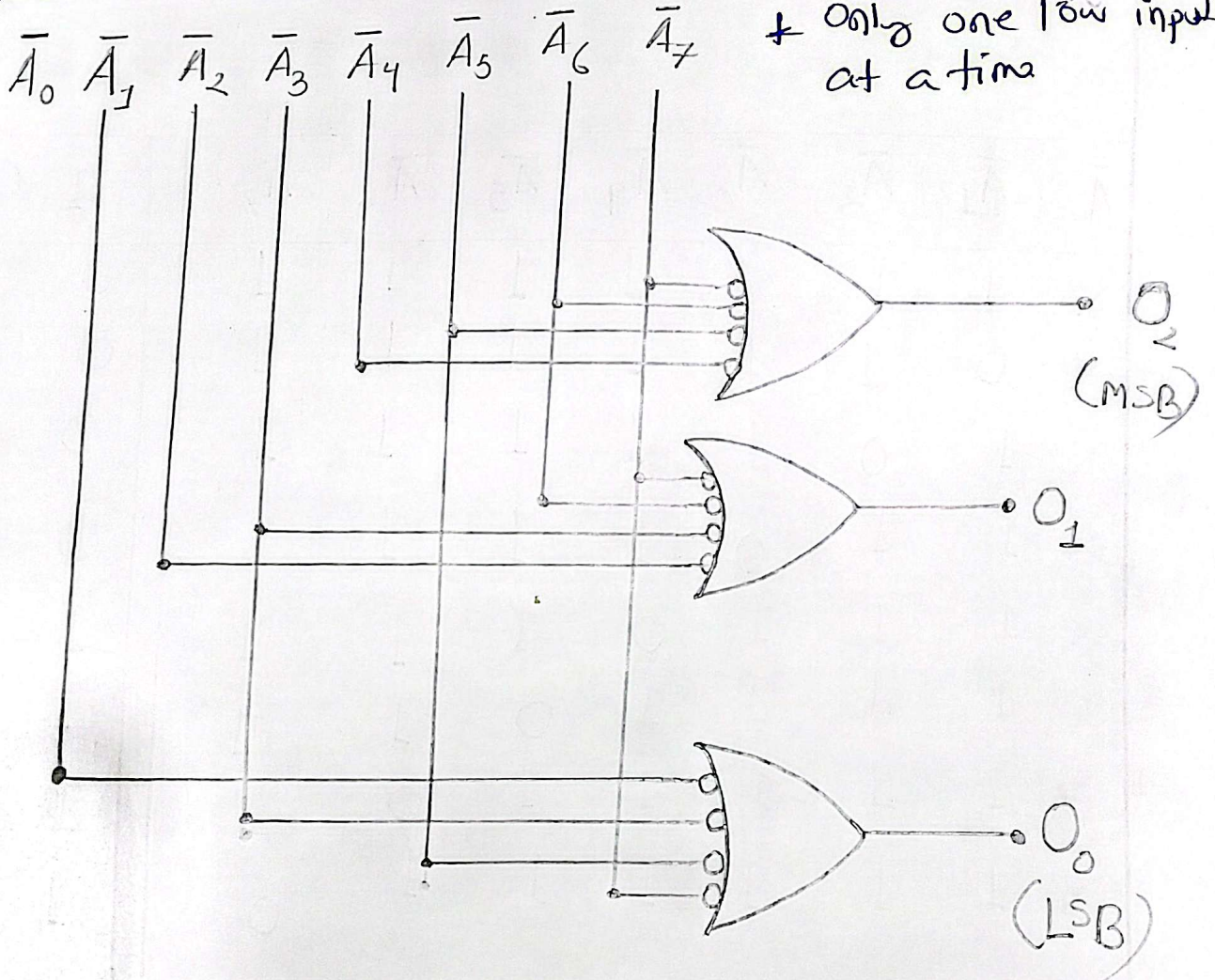


Fig. Logic Circuit for an 8-to-3 line encoder (8-line-to-3-line) encoder

Multiplexers (Data Selectors) / MUX's

A multiplexer or MUX is a combinational logic circuit that selects one input from multiple inputs and forwards it to a single output line — based on the select lines.

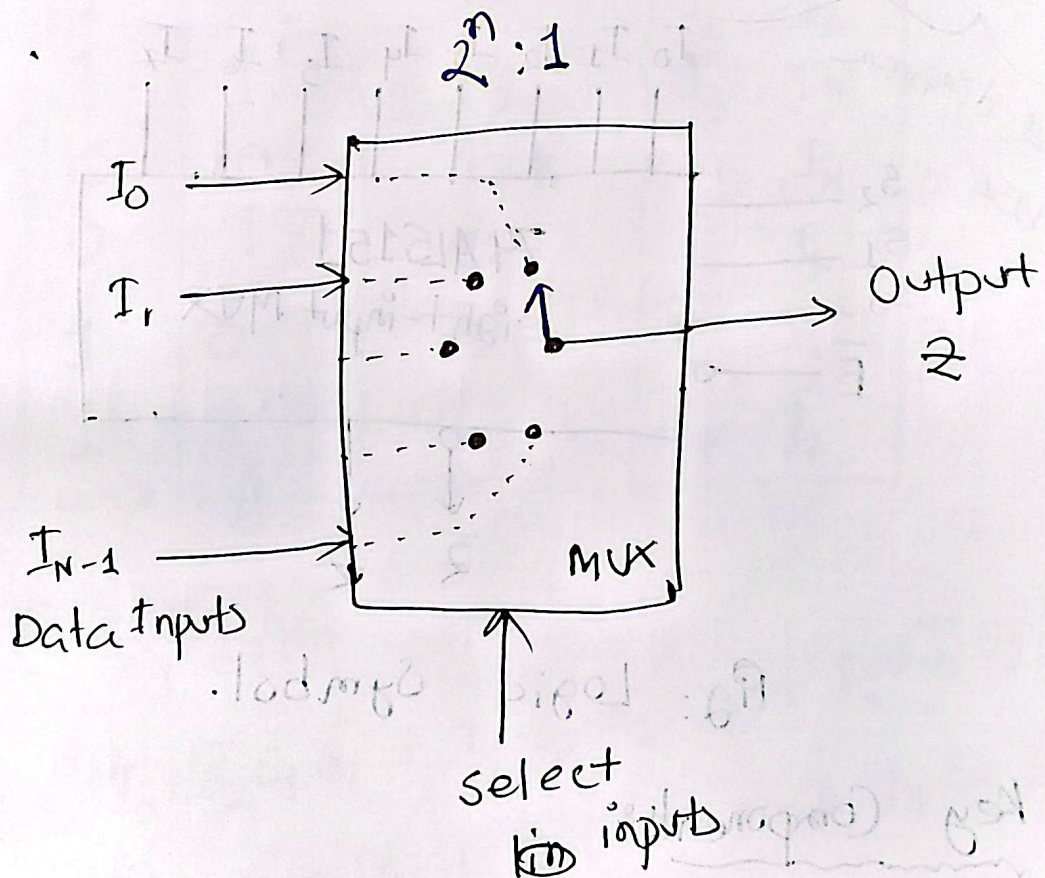


Fig: Functional Diagram of MUX.

8:1 Multiplexers: An 8:1 multiplexer is a combinational logic circuit that selects one of 8 input lines (I_0 to I_7) and forwards it to a single output line, based on the values of 3 select lines (S_2, S_1, S_0).

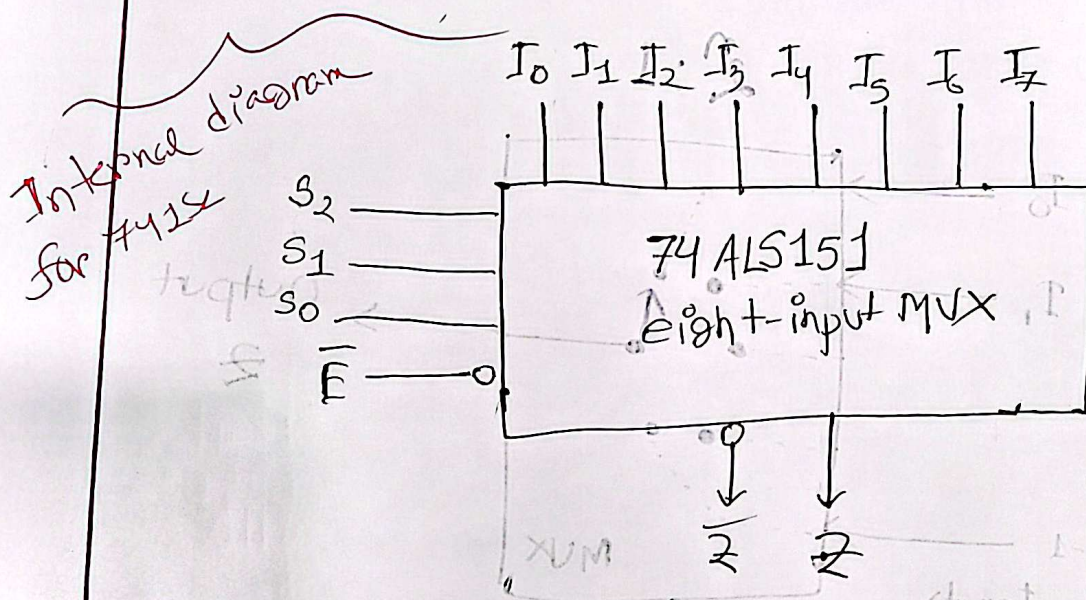


Fig: Logic Symbol.

Key Components

1. 8 input lines $\rightarrow I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7$
2. 3 Select lines $\rightarrow S_2, S_1, S_0$
3. 2 output line $\rightarrow \overline{2}, 2$ (inverted)
4. Enable input line $\rightarrow \overline{E}$

Truth table :

$\bar{E}$	S_2	S_1	S_0	selected input	Outputs $\frac{Z}{2}$
1	X	X	X	-	1 0
0	0	0	0	I_0	$\bar{I}_0$ I_0
0	0	0	1	I_1	$\bar{I}_1$ I_1
0	0	1	0	I_2	$\bar{I}_2$ I_2
0	0	1	1	I_3	$\bar{I}_3$ I_3
0	1	0	0	I_4	$\bar{I}_4$ I_4
0	1	0	1	I_5	$\bar{I}_5$ I_5
0	1	1	0	I_6	$\bar{I}_6$ I_6
0	1	1	1	I_7	$\bar{I}_7$ I_7

Logical Expression :

$$\begin{aligned}
 Z = & I_0 \cdot \bar{S}_2 \bar{S}_1 \bar{S}_0 + I_1 \cdot \bar{S}_2 \bar{S}_1 \cdot S_0 + I_2 \cdot \bar{S}_2 S_1 \bar{S}_0 \\
 & + I_3 \cdot \bar{S}_2 S_1 S_0 + I_4 S_2 \bar{S}_1 \cdot \bar{S}_0 + I_5 S_2 \bar{S}_1 S_0 \\
 & + I_6 S_2 S_1 \bar{S}_0 + I_7 \cdot S_2 \cdot S_1 \cdot S_0
 \end{aligned}$$

How it Works: (74151) $\rightarrow$ select input

1. Select lines Decodes the Inputs

$\rightarrow$ The 3-bit input (s_2, s_1, s_0) acts like a binary number (0 to 7).

$\rightarrow$ Example: If $s_2=1, s_1=0, s_0=1$ (101 in binary = 5), then I_5 is selected.

2. Only One Input is Passed to Outputs

$\rightarrow$ The MUX activates only one input at a time based on the select lines.

$\rightarrow$ All other inputs are ignored.

3. Output Follows the Selected Inputs

$\rightarrow$ If $E_3=1$ and it is selected, then $Z=1$

$\rightarrow$ If $E_3=0$ and it is selected then $Z=0$.

99 Logic Circuit: / Internal diagram of 74ALS151 multiplexer:

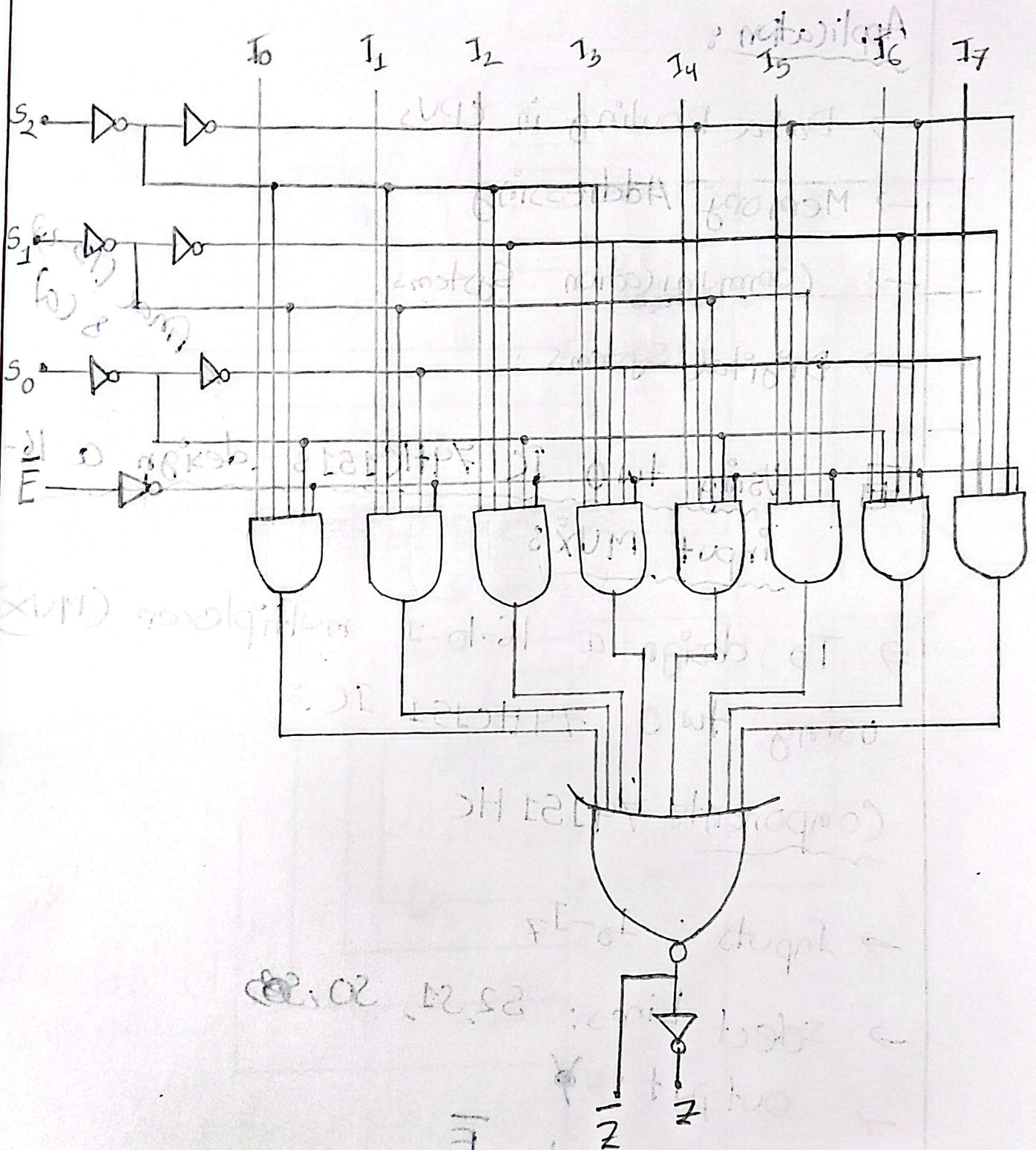


Fig. 8.1 ~~8:1 multiplexer~~ / Logic Diagram
of ~~74ALS138 IC~~,
74ALS151 IC.

Application :

- Data Routing in CPUs
- Memory Addressing
- Communication Systems.
- Digital Systems.

→ Final (19-20)
8 (a)

☐ Using two IC 74HC151s design a 16-
input MUX

⇒ To design a 16-to-1 multiplexer (MUX)
using two 74HC151 IC.

Components 74151 IC

→ Inputs : $I_0 - I_7$

→ select Lines : S_2, S_1, S_0, S_0

→ output : Y

→ Enable input : $\overline{E}$

~~Unit 1: Digital Logic Design (DLD) - Multiplexers and Demultiplexers~~

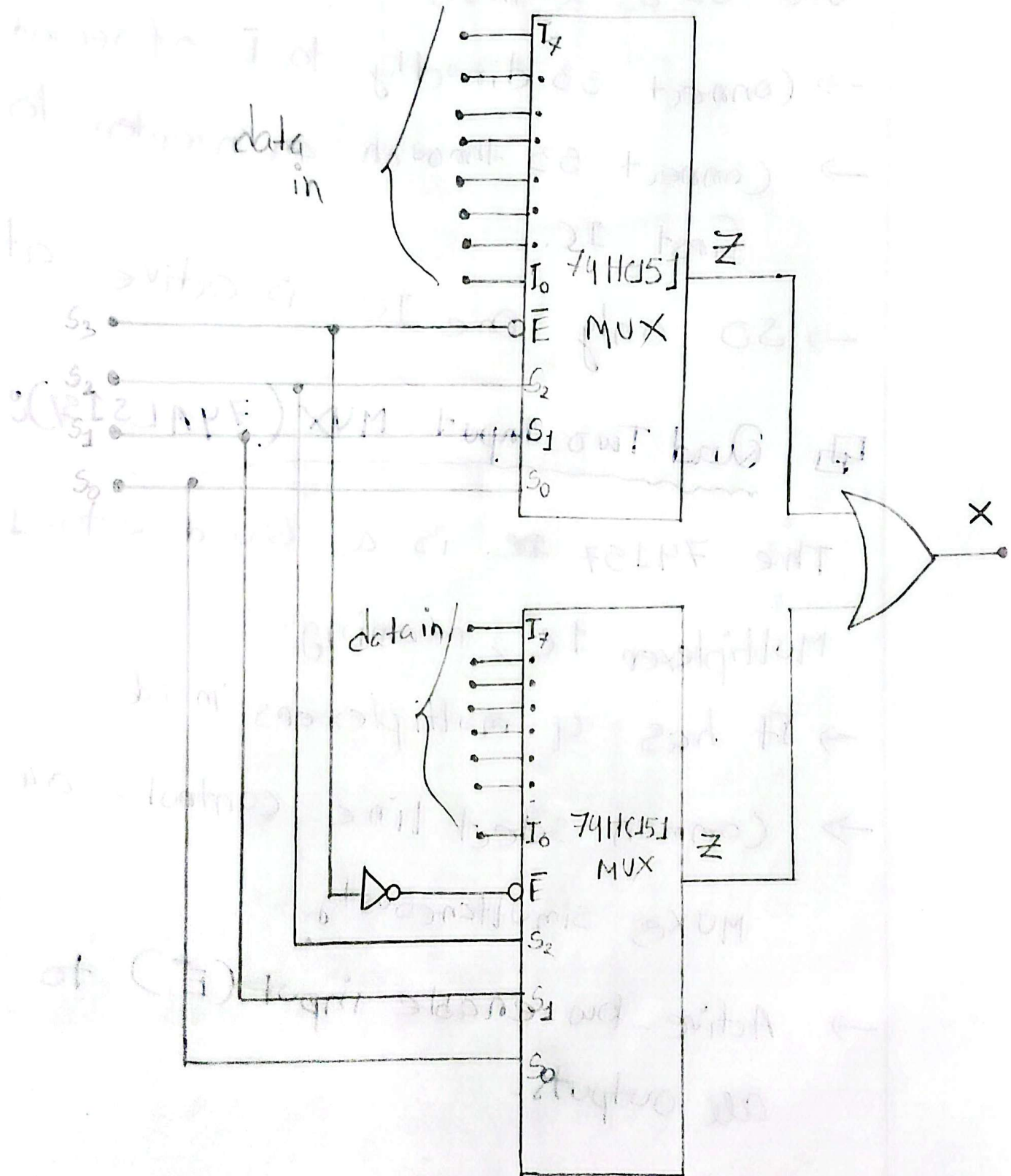


Fig. 16-to-1 MUX using two 74HC151 ICs

Use S3 as control;

102

- Connect S3 directly to $\bar{E}$ of second IC.
- Connect S3 through an inverter to $\bar{E}$ of first IC.
- So only one IC is active at a time.

Quad Two Input MUX (74ALS157):

The 74157 ~~IC~~ is a Quad 2-to-1 Multiplexer IC; meaning:

- It has 4 multiplexers inside
- Common select line controls all four MUXes simultaneously.
- Active-low enable input ($\bar{E}$) to disable all outputs.

2420) HNF out gnuv xum 1521 611

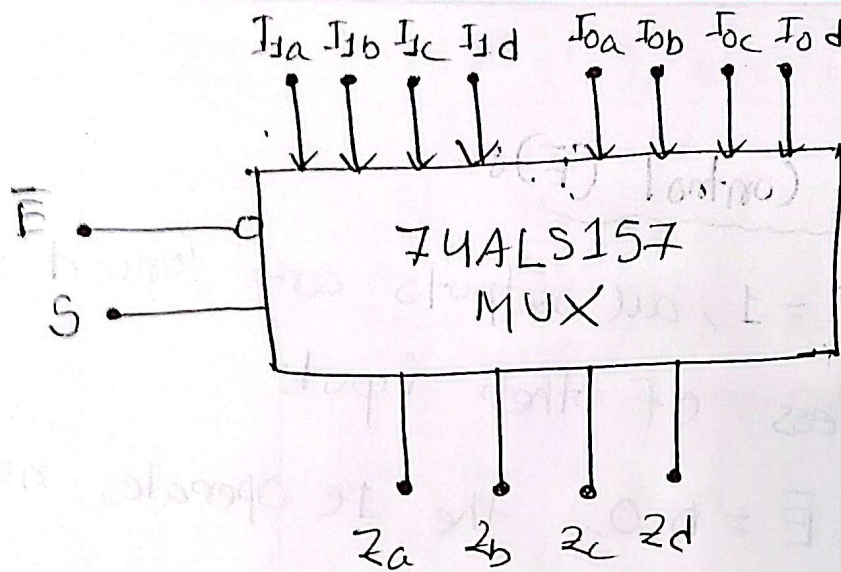


Fig: Logic Symbol.

Truth table:

$\bar{E}$	S	Z_a	Z_b	Z_c	Z_d
1	X	0	0	0	0
0	0	I_{0a}	I_{0b}	I_{0c}	I_{0d}
0	1	I_{1a}	I_{1b}	I_{1c}	I_{1d}

Operation:

1. Enable Control ($\bar{E}$):

- when $\bar{E} = 1$, all outputs are forced Low (0) regardless of other inputs
- when $\bar{E} = 0$, the IC operates normally.

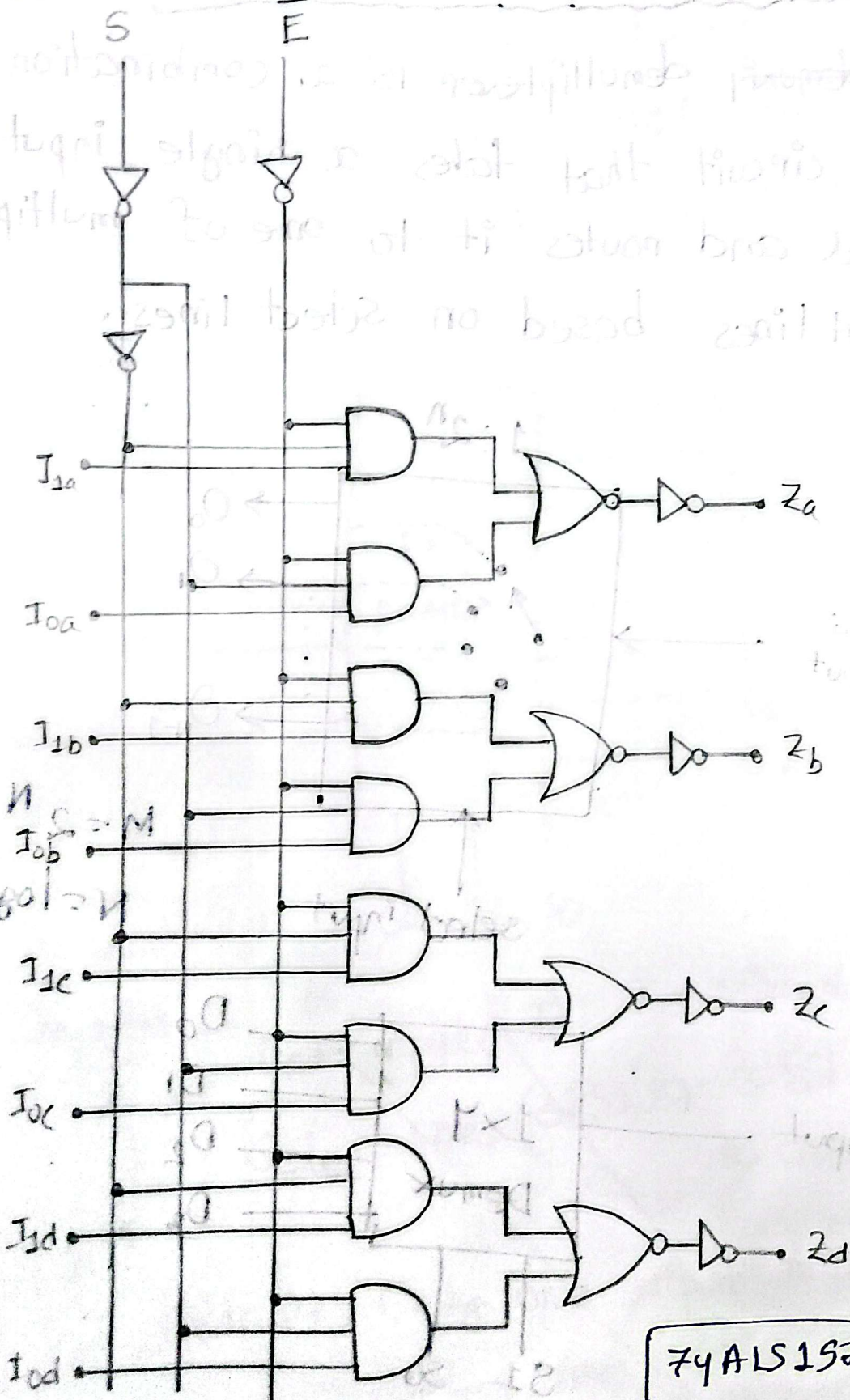
2. Data selection (S):

- when, $S = 0$, all four outputs follow their respective I_0 inputs.
- when $S = 1$, all four outputs follow their respective I_1 inputs.

3. Output Behavior:

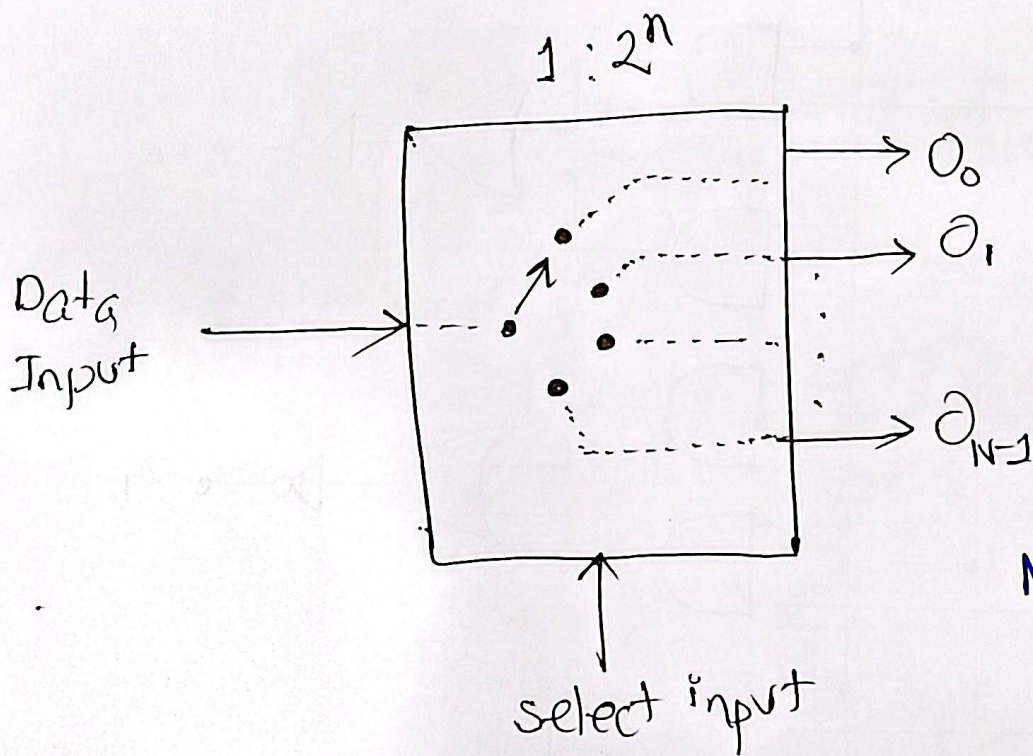
- The outputs follow the selected inputs (A or B) with minimal propagation delay.
- Outputs are non-inverted.

Logic Diagram (74ALS157) 4-to-1 Multiplexer



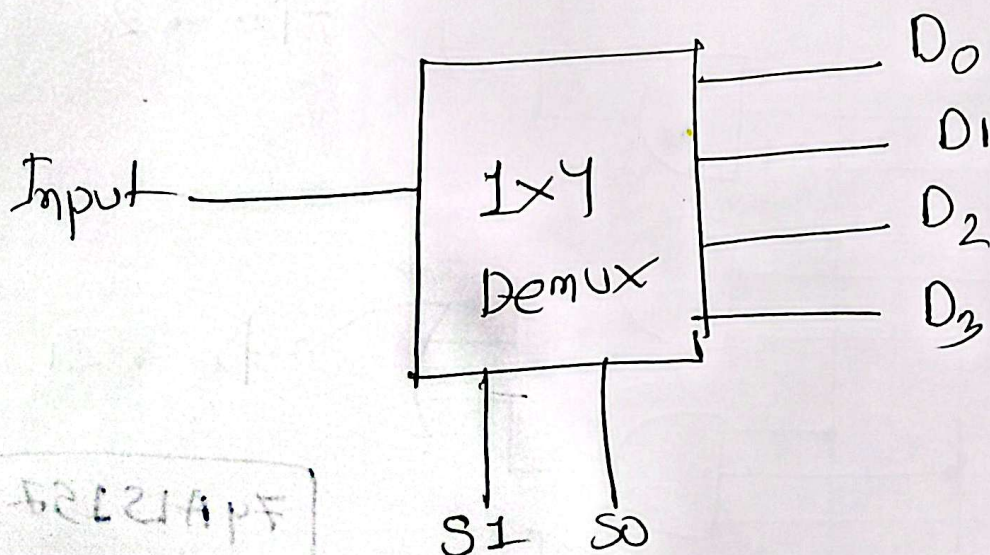
Demultiplexer (Data distributors) / DEMUX

A ~~demux~~ demultiplexer is a combinational logic circuit that takes a single input signal and routes it to one of multiple output lines based on select lines.



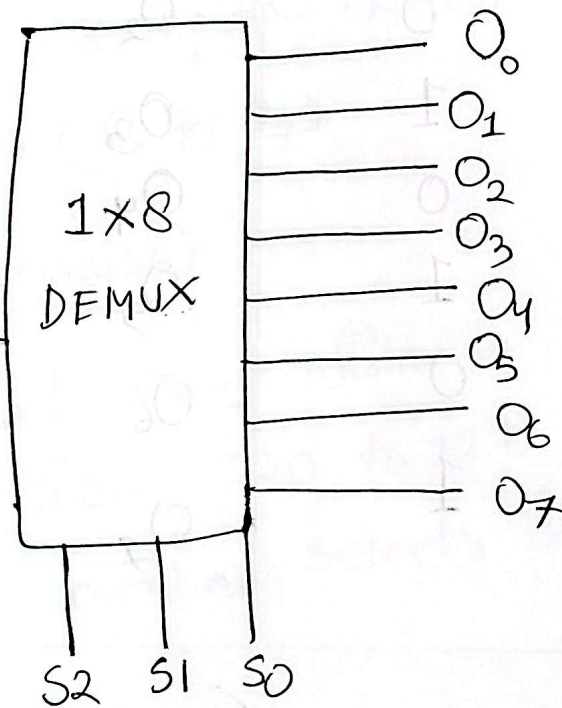
$$M = 2^N$$

$$N = \log_2(M)$$



1 to 8 Demux

A 1-to-8 demux is a digital circuit that routes a single input signal to one of eight output lines based on a 3-bit select input.



Inputs:

Data input (I): The signal to be routed.

Select lines (S_2, S_1, S_0): 3-bit binary input to choose the output.

Outputs:

O_0 to O_7 : Only one output matches the select lines at a time.

Truth table:

S2	S1	S0	Active Output	Logical Expression
0	0	0	Q_0	$Q_0 = 1 \cdot (\bar{S}_2 \bar{S}_1 \bar{S}_0)$
0	0	1	Q_1	$Q_1 = 1 \cdot (\bar{S}_2 \bar{S}_1 S_0)$
0	1	0	Q_2	$Q_2 = 1 \cdot (\bar{S}_2 S_1 \bar{S}_0)$
0	1	1	Q_3	$Q_3 = 1 \cdot (\bar{S}_2 S_1 S_0)$
1	0	0	Q_4	$Q_4 = 1 \cdot (S_2 \bar{S}_1 \bar{S}_0)$
1	0	1	Q_5	$Q_5 = 1 \cdot (S_2 \bar{S}_1 S_0)$
1	1	0	Q_6	$Q_6 = 1 \cdot (S_2 S_1 \bar{S}_0)$
1	1	1	Q_7	$Q_7 = 1 \cdot (S_2 S_1 S_0)$

Only one Output ($Q_0 - Q_7$) = 1^{active} at any time,
all others are 0 (inactive).

Working Principle:

1. Inputs:

→ 1 Data Input (I) → the signal to send to one output.

→ 3 select Inputs (S_2, S_1, S_0) → used to choose one of the 8 inputs. outputs.

→ 8 Outputs (O_0 to O_7)

2. Selection Mechanism:

→ The 3 select lines form a 3-bit binary number (from 000 to 111).

→ This binary number selects one output line.

→ Example:

* $S_2 S_1 S_0 = 011 \rightarrow$ Output O_3 is selected

3. Routing Data:

→ The selected output will follow the data input (I).

→ All other outputs stay low (0).

4. Results:

→ The value on I appears only on the selected output.

→ If $I = 1$ and $S_2 S_1 S_0 = 100$, only $O_4 = 1$
all others = 0.

Circuit:

111

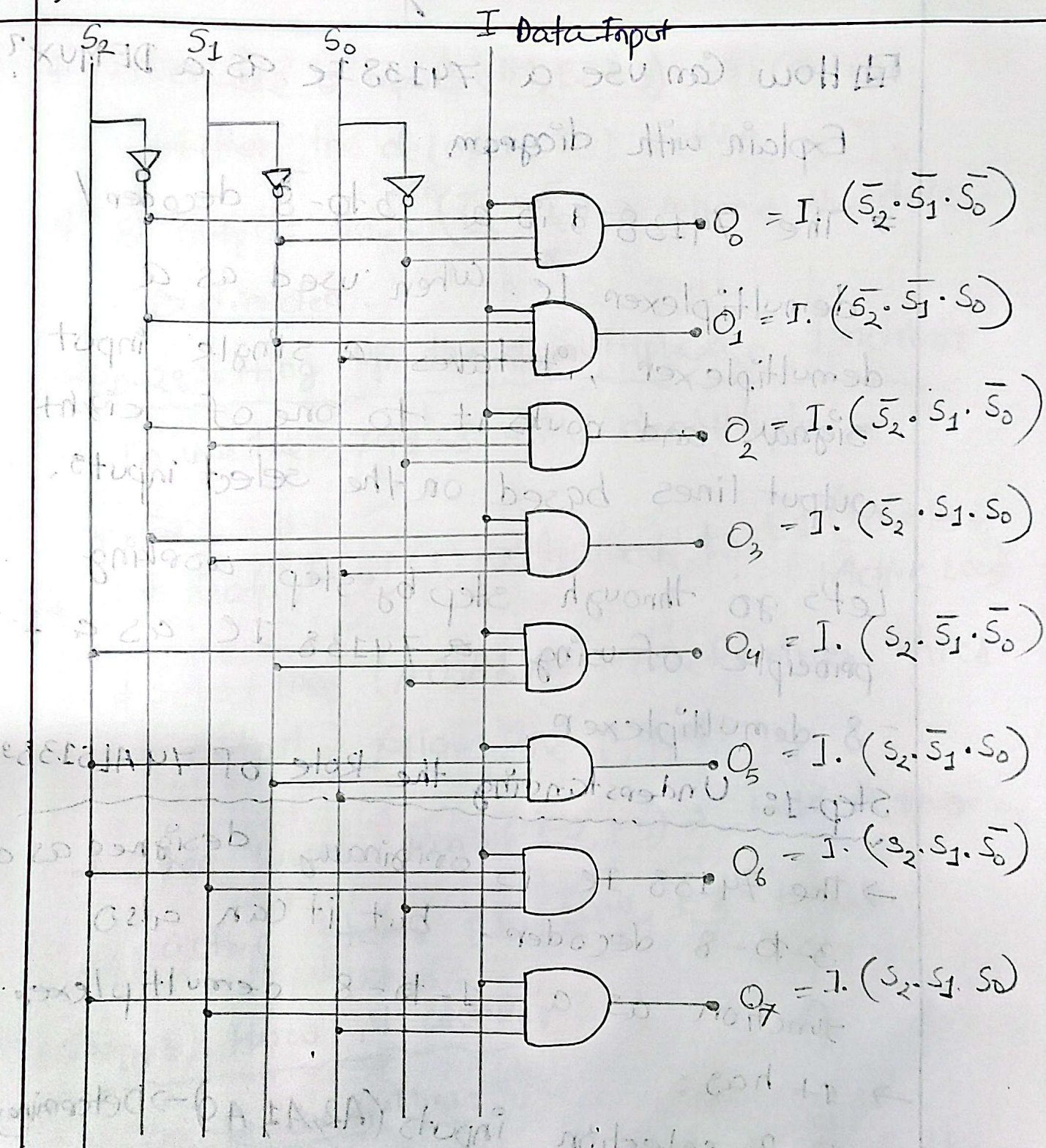


Fig. 1: 8 line demultiplexer.

How can we use a 74138 IC as a DEMUX?

Explain with diagram.

⇒ The 74138 is a 3-to-8 decoder / demultiplexer IC. When used as a demultiplexer, it takes a single input signal and routes it to one of eight output lines based on the select inputs.

Let's go through step by step working principle of using a 74138 IC as a 1-to-8 demultiplexer.

Step-1: Understanding the Role of 74ALS138

→ The 74138 IC is originally designed as a 3-to-8 decoder, but it can also function as a 1-to-8 demultiplexer.

→ It has:

+ 3 selection inputs (A_2, A_1, A_0) → Determine which output gets activated.

* 3 enable inputs ($\overline{E1}, \overline{E2}, E3$) $\rightarrow$ Control whether the decoder is active.

* 8 output lines ($\overline{O0} - \overline{O7}$) $\rightarrow$ where the data is directed.

Step-2: setting up the demultiplexer functions

$\rightarrow$ To use the 74138 as a demultiplexer:

* ~~E1~~ Data Input (I) $\rightarrow$ Applied to $\overline{E1}$ (Enable 1, Active Low)

* Select lines ($A0, A1, A2$) $\rightarrow$ Determines which output follows the input.

* Other Enable Pins ($\overline{E2}, E3$) $\rightarrow$ Held in their active states ($\overline{E2} = \text{Low}, E3 = \text{High}$).

Step-3: How it works

1. Enable Conditions:

* $E3 (\text{Gnd}) = \text{High}$ (must be high to enable the chip)

* $\overline{E2}$ (G2B) = Low (must be Low to enable the chip)

* $\overline{E1}$ (G2A) = Data Input (I) (The signal to be demultiplexed)

2. Select lines ($A2, A1, A0$):

→ Three bits determine which output ($\overline{O_0} - \overline{O_7}$) will follow the input signal.

→ Example :

* If $A2A1A0 = 000$, $\overline{O_0}$ follows $\overline{E1}$ (data input)

* If $A2A1A0 = 001$, $\overline{O_1}$ follows $\overline{E1}$, and

* If $A2A1A0 = 010$ on

3. Output Behaviours

→ The selected output will be:

* Low if $\overline{E1}$ (Data Input) = Low.

* High if $\overline{E1}$ (Data Input) = High.

→ All other outputs remain High (inactive)

Step-3: Example: A2A1A0 = 000 ($\bar{O}_0$ Selected):

* Input signal (1) is applied to $\bar{E}_1$

* $A2 = 0, A1 = 0, A0 = 0 \rightarrow \bar{O}_0$ is selected.

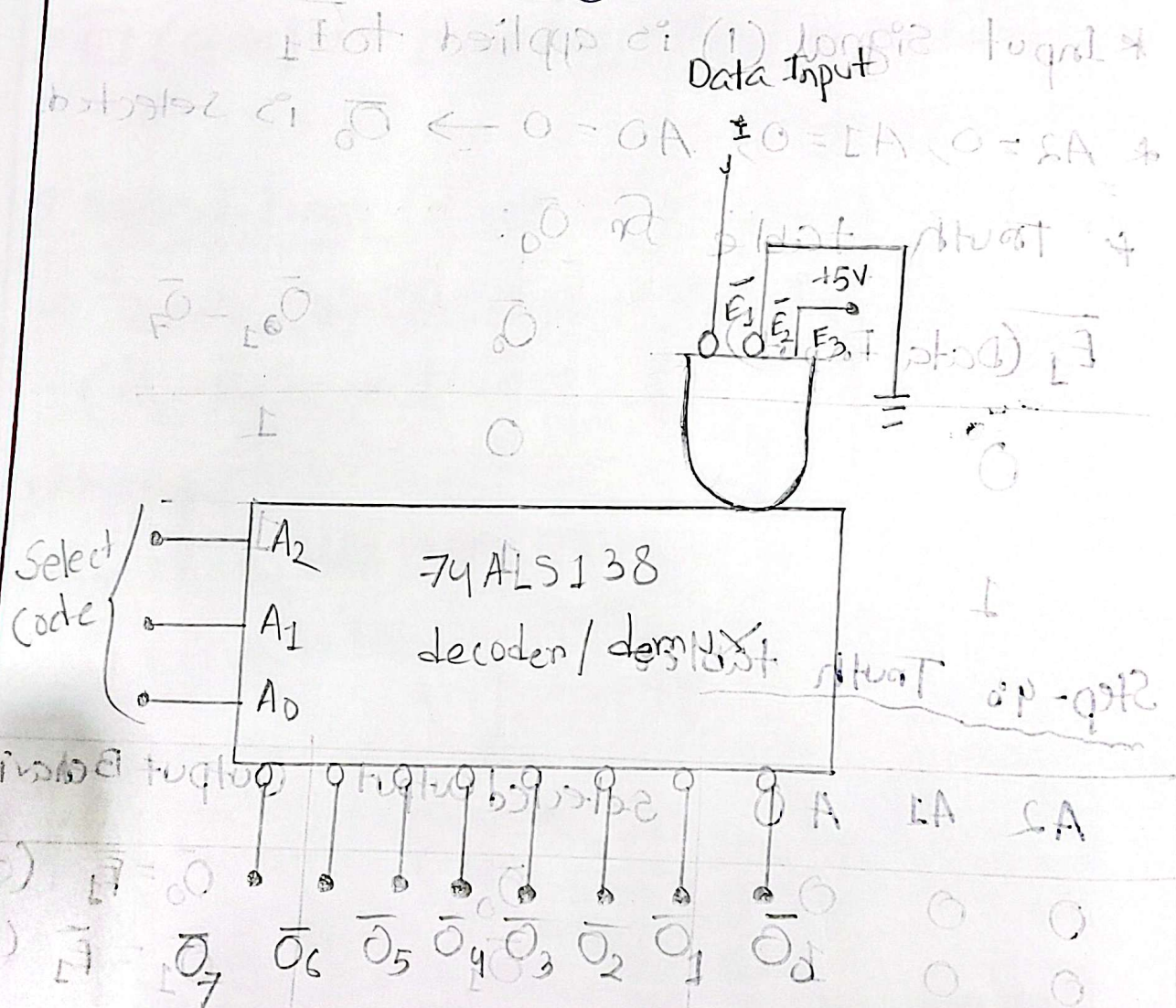
* Truth table for $\bar{O}_0$:

$\bar{E}_1$ (Data Input)	$\bar{O}_0$	$\bar{O}_0 - \bar{O}_7$
0	0	1
1	1	1

Step-4: Truth tables

A2	A1	A0	selected output	Output Behaviour
0	0	0	$\bar{O}_0$	$\bar{O}_0 = \bar{F}_1$ (Others = high)
0	0	1	$\bar{O}_1$	$\bar{O}_1 = \bar{F}_1$ (")
0	1	0	$\bar{O}_2$	$\bar{O}_2 = \bar{F}_1$ (")
0	1	1	$\bar{O}_3$	$\bar{O}_3 = \bar{F}_1$ (")
1	0	0	$\bar{O}_4$	$\bar{O}_4 = \bar{F}_1$ (")
1	0	1	$\bar{O}_5$	$\bar{O}_5 = \bar{F}_1$ (")
1	1	0	$\bar{O}_6$	$\bar{O}_6 = \bar{F}_1$ (")
1	1	1	$\bar{O}_7$	$\bar{O}_7 = \bar{F}_1$ (")

Step 5: Circuit Diagrams (Demux Mode)



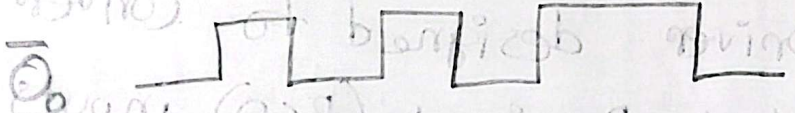
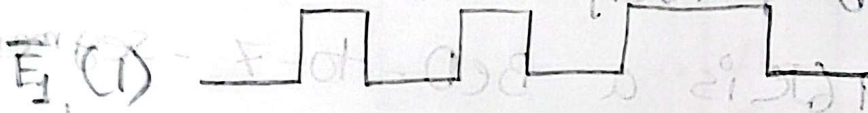
Step 6: waveform Behavior:

when $A_2 A_1 A_0 = 000 \rightarrow$ selects output $\bar{O}_0$.

$\rightarrow$ For this case, the data signal applied to $\bar{E}_1$ will be transmitted to $\bar{O}_0$.

→ All other outputs will remain in their inactive high state.

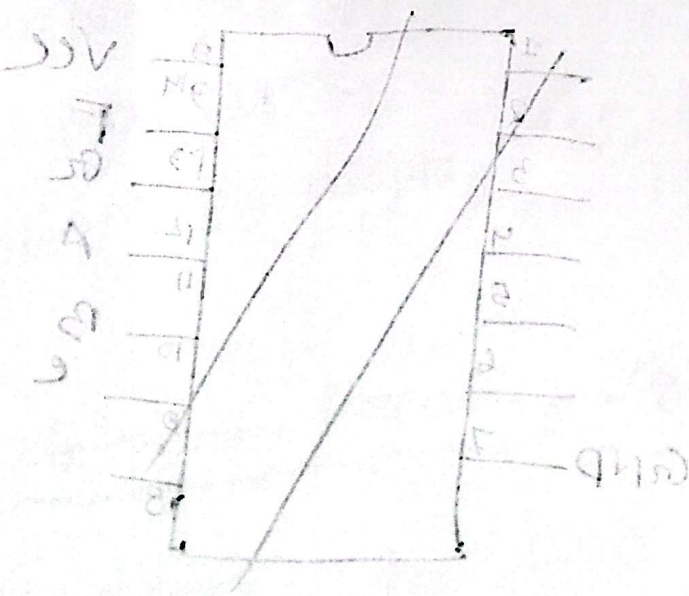
(a) Use an example of a 74138 decoder.



$\overline{O}_7 - \overline{O}_0$

Logic 1

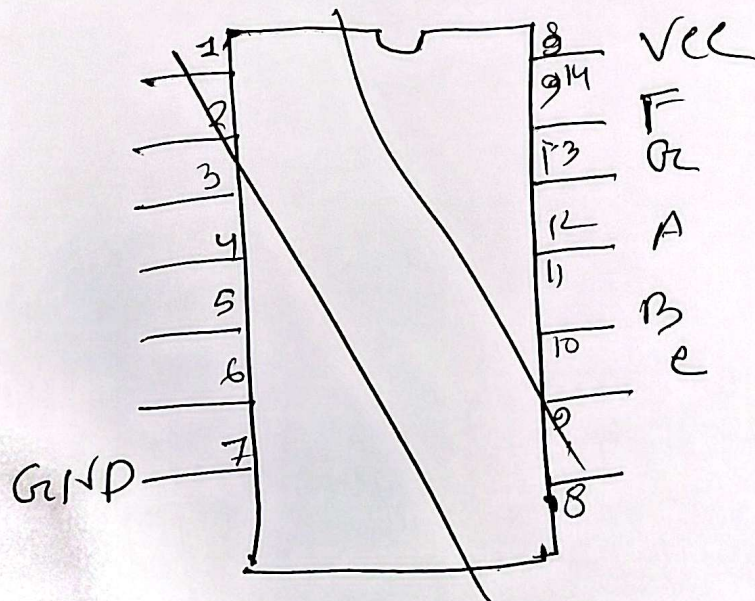
wave forms for $A_2 A_1 A_0 = 000$.



Final (21-22)

7.(a) Use an example to explain the working principle of a 7446 IC.

⇒ The 7446 IC is a BCD-to-7-segment Decoder/Driver designed to convert 4-bit Binary-Coded Decimal (BCD) inputs into signals that drive a 7-segment LED display. It can sink higher current and includes features like ripple blanking and lamp test.



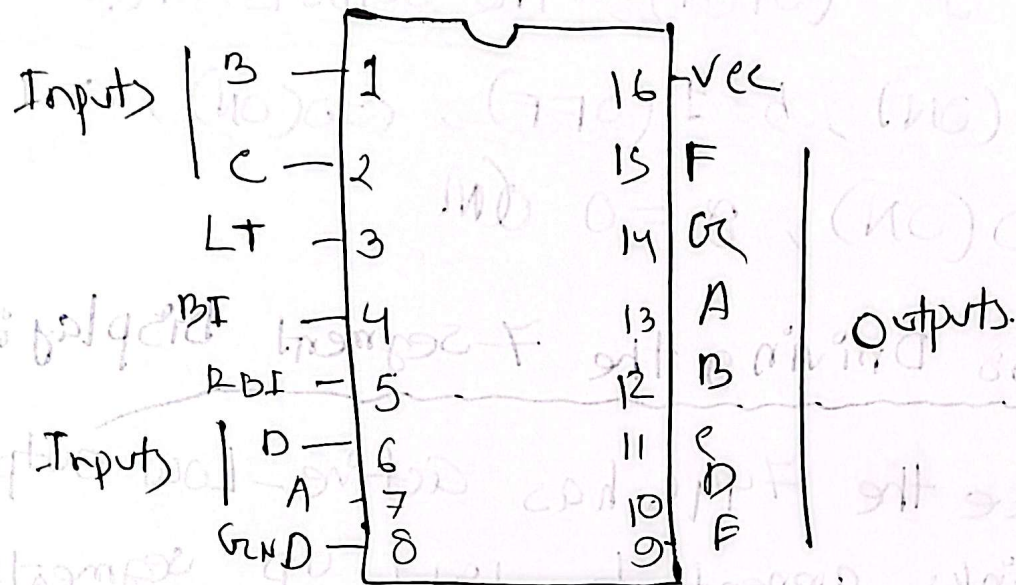


Fig. 7446 IC Pin Diagram

Working Principles

1. Step-1: BCD input Applied:

→ The 4-bit BCD code (A, B, C, D) is applied to the input pins.

→ Example: To display "5", input = 0101
(A = 0, B = 1, C = 0, D = 1)

2. Step-2: Internal Decoding Logic:

→ The IC decodes the BCD input and activates the correct segment outputs (a-g) based on a built-in truth table.

→ For "5" (0101), the outputs are:

$a=0$ (ON), $b=1$ (OFF), $c=0$ (ON), $d=1$ (OFF)
 $f=0$ (ON), $g=0$ (ON)

Step 3: Driving the 7-segment Display:

→ Since the 7446 has active-low outputs, it sinks current to light up segments.

→ A common anode display is used,

where:

* Anode (positive side) is connected to $V_{CC}(+5V)$

* Cathode (Segment pins) connect to 7446

outputs via resistors.

→ when an output is Low (0), current flows

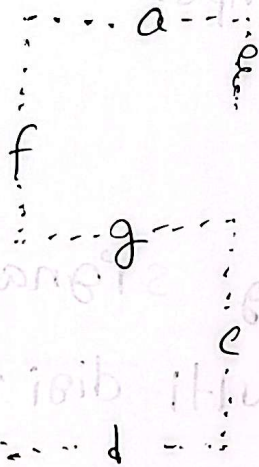
→ segment lights up

→ when an output is high (1), no current

flows → segment stays off.

Step 4: Displaying the Digits

→ For "5", the segment a, c, d, f, g light up.



Circuit Diagrams Working

Special Functions Explained

(a) Lamp Test ($\overline{LT} = \text{Pin } 3$)

→ when $\overline{LT} = 0$ (Low), all segments turn ON
(displays 8)

(b) Blanking Input ($\overline{BI} = \text{Pin } 4$)

→ when $\overline{BI} = 0$ (Low), all outputs turn OFF, blanking the display.

(c) Ripple Banking ($\overline{RBI}$, $\overline{RBO}$)

+ $\overline{RBI}$ (Pin 5)

→ If $\overline{RBI} = 0$ and input = 0000 (0), the display blanks.

+ $\overline{RBO}$ (Pin 4)

→ Outputs a blanking signal to the next digit's $\overline{RBI}$ in multi-digit displays.

7(b) →

8. Form a 1-of-48 decoder using 74138 IC.

⇒ To build a 1-of-48 decoder

we need 7x 74138 IC.

→ Explanation

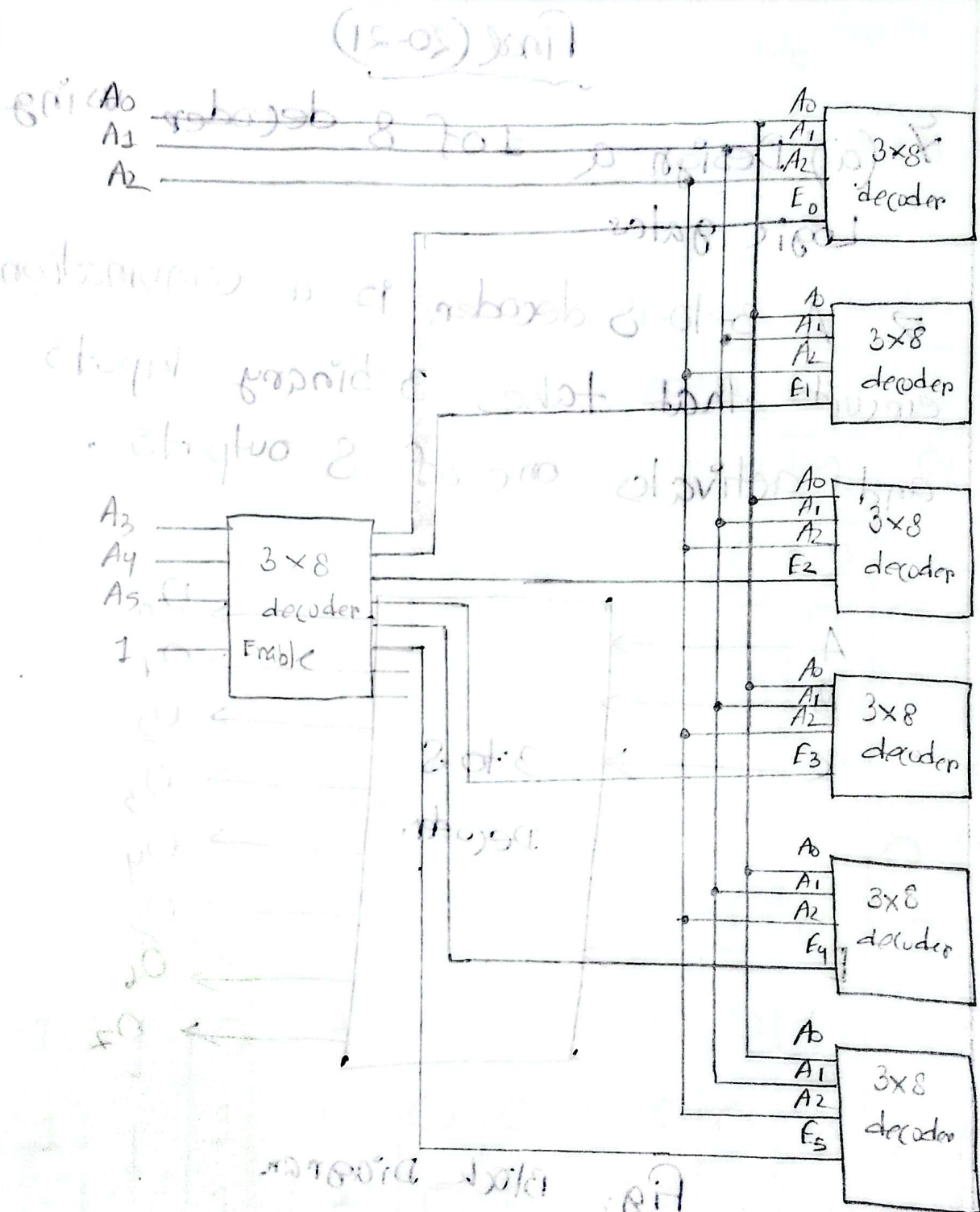


fig. 1-of-48 decoder.

Final (20-21)

Q. (a) Design a 1 of 8 decoder using Logic gates

⇒ A 3-to-8 decoder is a combinational circuit that takes 3 binary inputs and activates one of 8 outputs.

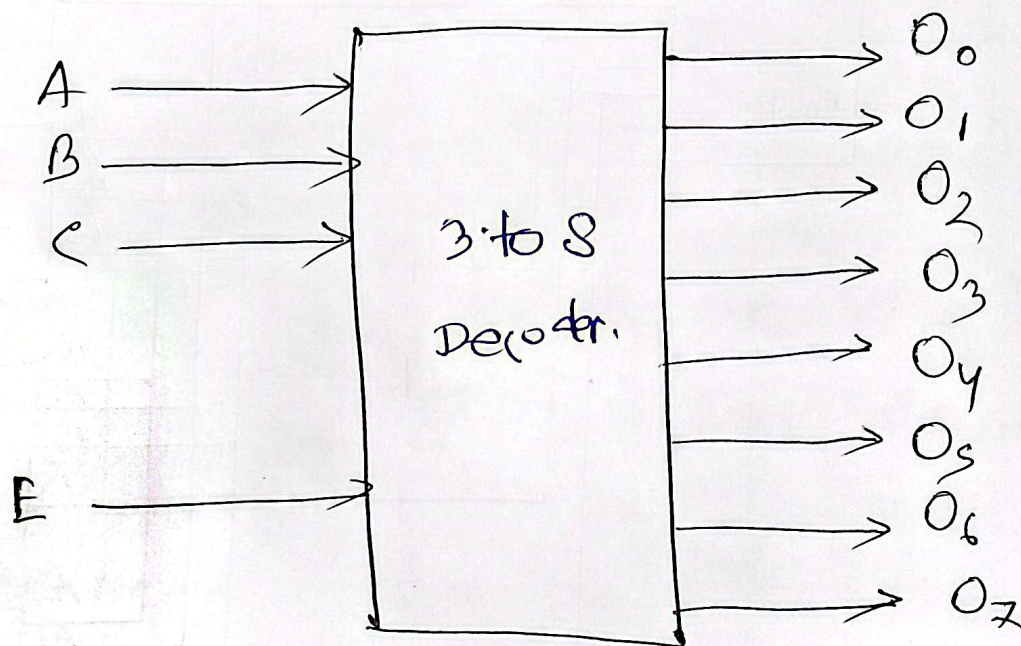


Fig. Block Diagram

Fig. 1 of 8 decoder

Inputs: A, B, C (MSB), E (enable)
(LSB)

Outputs: $Q_0 - Q_7$ (only one is active high at a time)

Truth tables

Inputs				Outputs							
E	C	B	A	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

Circuit:Logical Expressions

$$O_0 = E \bar{A} \bar{B} \bar{C}$$

$$O_1 = E \bar{A} \bar{B} C$$

$$O_2 = E \bar{A} B \bar{C}$$

$$O_3 = E \bar{A} B C$$

$$O_4 = E A \bar{B} \bar{C}$$

$$O_5 = E A \bar{B} C$$

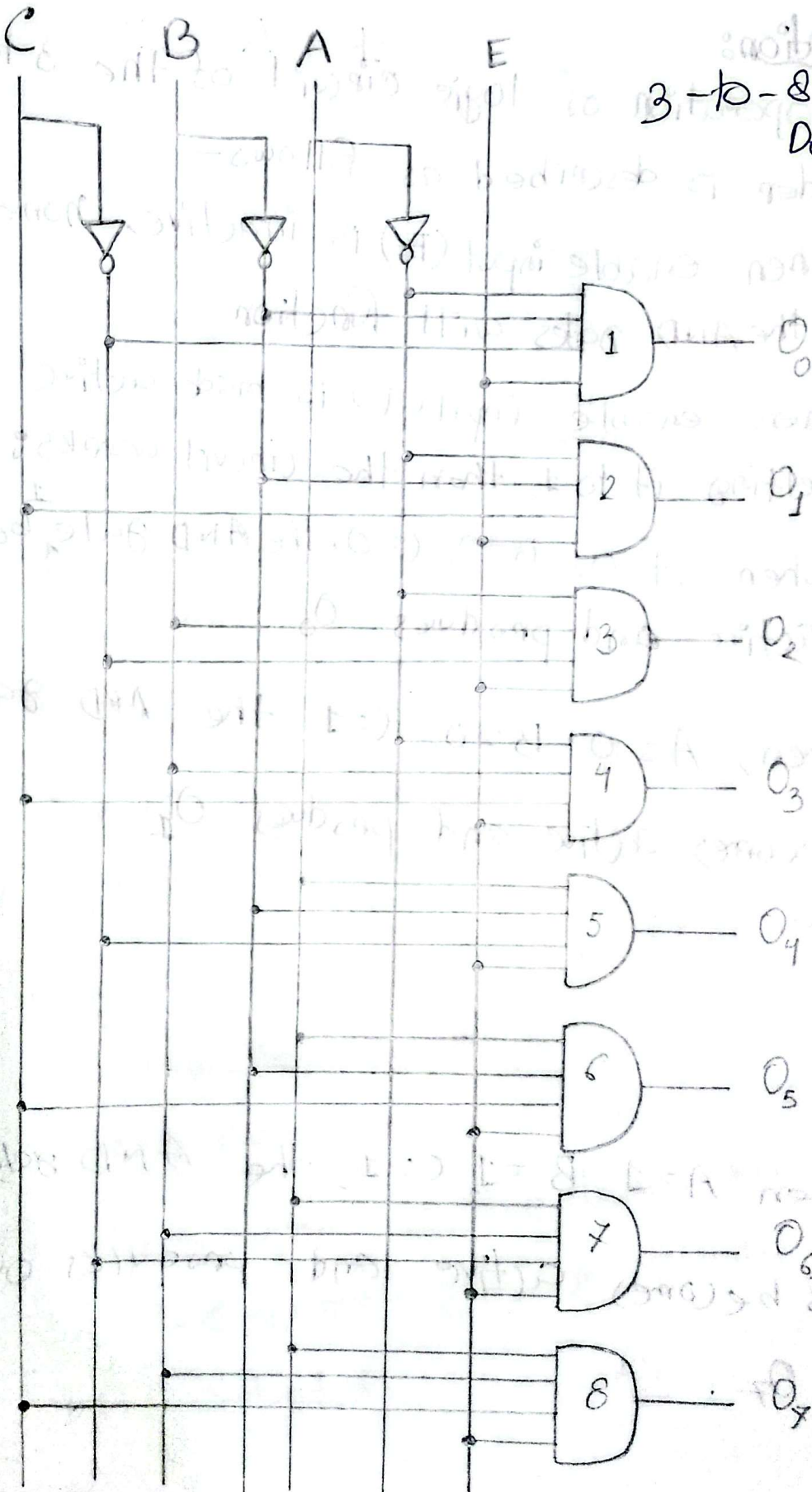
$$O_6 = E A B \bar{C}$$

$$O_7 = E A B C$$

Circuit.

127

3-to-8
Decoder



Operation:

The operation of logic circuit of the 3 to 8 decoder is described as follows -

→ when enable input (E) is inactive, none of the AND gates will function

→ when enable input (E) is made active by setting it to 1, then the circuit works

* when, $A=0$, $B=0$, $C=0$, the AND gate₁ becomes active and produces O_0 .

* when, $A=0$, $B=0$, $C=1$, the AND gate 2 becomes active and produces O_1

...

* when $A=1$, $B=1$, $C=1$, the AND gate

8 becomes active and produces output

O_7 .

7(b) → আর কটা আছে

8 7(c), 8(a), (b) → আর কটা আছে

8(c) 7442 BCD to Decimal Decoders

Working Principles

1. Inputs:

- IC has 4 binary inputs: D, (MSB), C, B, A (LSB)
- These represent a BCD number from 0000 (0) to 1001 (9).

2. Decoding Functions:

- The IC activates one of the 10 output lines ($\bar{O}_0$ to $\bar{O}_9$) corresponding to BCD input value.
- Only one output goes Low (0) at a time, the rest remain High (1).

3. Output Behavior:

- Outputs are active Low, meaning the selected output goes to 0 volts, indicating it's selected.

For example:

→ If input = 0100 (decimal 4), output $\bar{O}_4 = 0$, all others High.

Final (19-20)

7. (a) Draw the logic Diagram of a 1-of-16 Decoder and explain its operation. Consider an enable input in your design.

⇒ A 1-of-16 decoder converts a 4-bit binary input (A_3, A_2, A_1, A_0) into a single active output ($O_0 - O_{15}$) among 16 possible outputs.

Additionally, an Enable (E) input ensures that the decoder functions only when E is active.

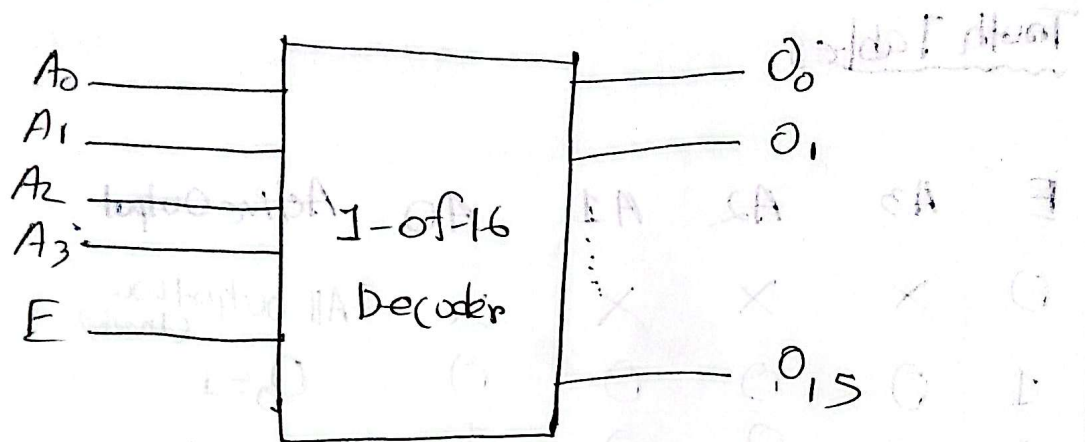


Fig: Block Diagram

Inputs:

- $A_0 - A_3$ (4-bit binary input, $A_0 = \text{LSB}$),
- Enable (Active High, turn decoder ON/OFF)

Outputs:

- $O_0 - O_{15}$ (Active High, only one is High at a time.)

Truth Table

E	A ₃	A ₂	A ₁	A ₀	Active Output
0	X	X	X	X	All output low (Inactive)
1	0	0	0	0	O ₀ = 1
1	0	0	0	1	O ₁ = 1
1	0	0	1	0	O ₂ = 1
1	0	0	1	1	O ₃ = 1
1	0	1	0	0	O ₄ = 1
1	0	1	0	1	O ₅ = 1
1	0	1	1	0	O ₆ = 1
1	0	1	1	1	O ₇ = 1
1	1	0	0	0	O ₈ = 1
1	1	0	0	1	O ₉ = 1
1	1	0	1	0	O ₁₀ = 1
1	1	0	1	1	O ₁₁ = 1
1	1	1	0	0	O ₁₂ = 1
1	1	1	0	1	O ₁₃ = 1
1	1	1	1	0	O ₁₄ = 1
1	1	1	1	1	O ₁₅ = 1

Logical Expressions

$$O_0 = E \cdot \bar{A}_3 \cdot \bar{A}_2 \cdot \bar{A}_1 \cdot \bar{A}_0$$

$$O_1 = E \cdot \bar{A}_3 \cdot \bar{A}_2 \cdot \bar{A}_1 \cdot A_0$$

$$O_2 = E \cdot \bar{A}_3 \cdot \bar{A}_2 \cdot A_1 \cdot \bar{A}_0$$

$$O_3 = E \cdot \bar{A}_3 \cdot \bar{A}_2 \cdot A_1 \cdot A_0$$

$$O_4 = E \cdot \bar{A}_3 \cdot A_2 \cdot \bar{A}_1 \cdot \bar{A}_0$$

$$O_5 = E \cdot \bar{A}_3 \cdot A_2 \cdot \bar{A}_1 \cdot A_0$$

$$O_6 = E \cdot \bar{A}_3 \cdot A_2 \cdot A_1 \cdot \bar{A}_0$$

$$O_7 = E \cdot \bar{A}_3 \cdot A_2 \cdot A_1 \cdot A_0$$

$$O_8 = E \cdot A_3 \cdot \bar{A}_2 \cdot \bar{A}_1 \cdot \bar{A}_0$$

$$O_9 = E \cdot A_3 \cdot \bar{A}_2 \cdot \bar{A}_1 \cdot A_0$$

$$O_{10} = E \cdot A_3 \cdot \bar{A}_2 \cdot A_1 \cdot \bar{A}_0$$

$$O_{11} = E \cdot A_3 \cdot \bar{A}_2 \cdot A_1 \cdot A_0$$

$$O_{12} = E \cdot A_3 \cdot A_2 \cdot \bar{A}_1 \cdot \bar{A}_0$$

$$O_{13} = E \cdot A_3 \cdot A_2 \cdot \bar{A}_1 \cdot A_0$$

$$O_{14} = E \cdot A_3 \cdot A_2 \cdot A_1 \cdot \bar{A}_0$$

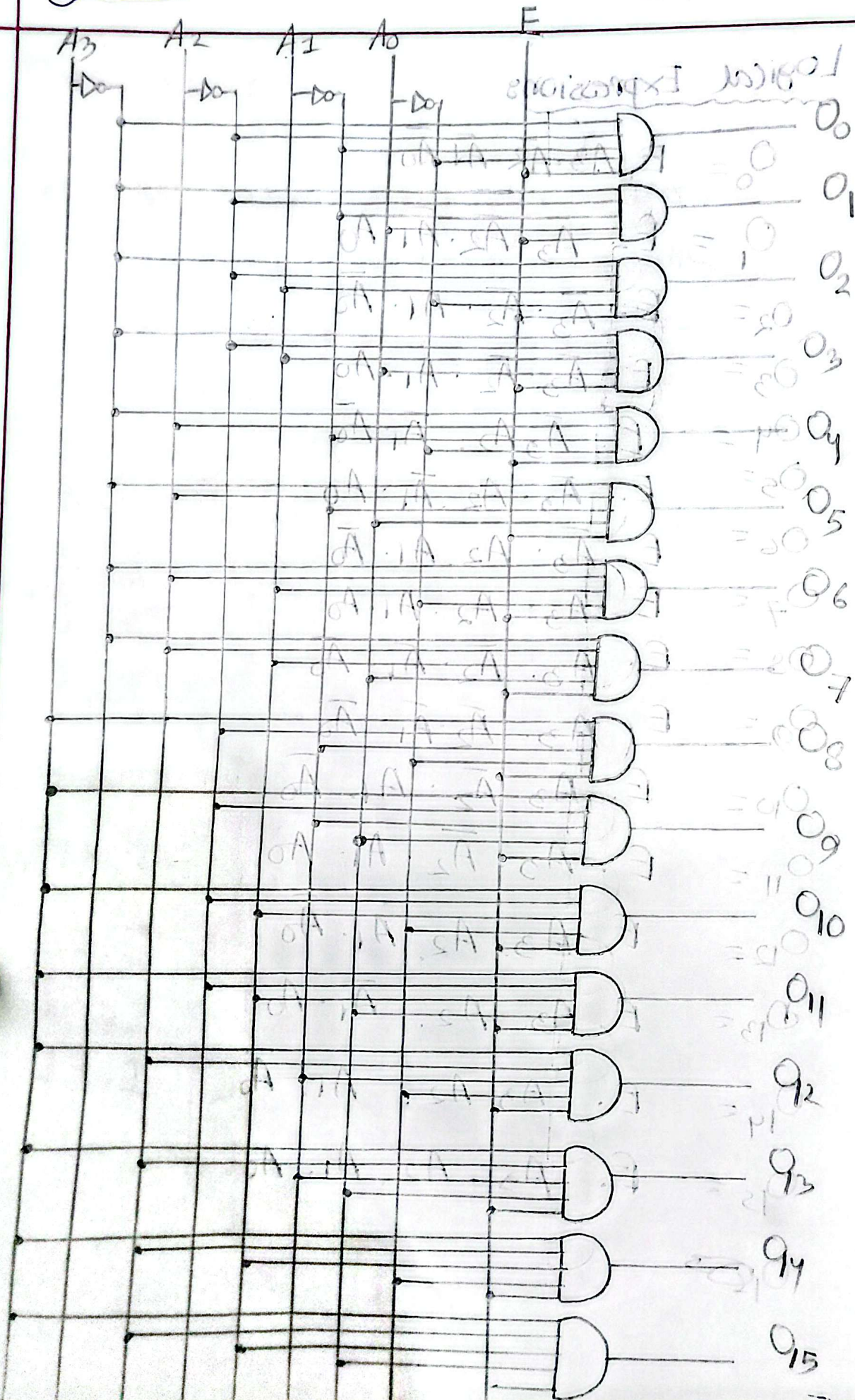
$$O_{15} = E \cdot A_3 \cdot A_2 \cdot A_1 \cdot A_0$$

$$O_{16}$$

Circuit:

131

4x16 Decoder.



Operation:

1. When Enable = 0,

→ All AND gates are disabled → All output ($O_0 - O_7$) = 0.

2. when Enable = 1,

→ The decoder activates one output based on

$A_3 A_2 A_1 A_0$:

+ Each output O_n is an AND combination of:

→ Enable (1)

→ Appropriate True / complemented inputs.

+ Example:

→ For $A_3 A_2 A_1 A_0 = 0101$ (5):

→ $O_5 = \bar{A}_3 \cdot A_2 \cdot \bar{A}_1 \cdot A_0 \cdot \text{Enable}$

→ Only this AND gate has all 1s → $O_5 = 1$

Others = 0.

Example: Decoding 1100 (12)

1. Inputs:

$$\rightarrow A_3 = 1, A_2 = 1, A_1 = 0, A_0 = 0$$

$$\rightarrow \text{Enable} = 1$$

2. Logic:

$$\rightarrow O_{12} = A_3 \cdot A_2 \cdot \bar{A}_1 \cdot \bar{A}_0 \cdot \text{Enable}$$

$$= 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

$\rightarrow$ All other outputs remain 0.

3. Result: Only $O_{12} = 1$ (High).

7(b),

8(a) $\rightarrow$ Using 74HC151 ICs, design a 16 input MUX and explain its operation.

$\Rightarrow$ To create a 16:1 MUX, we cascade two 74151 ICs using a 4th select line for chip selection.

16:1 MUX designComponents required:

- Two 74151 ICs (for 16 inputs)
- 1 NOT gate (for 4th select line)
- 1 OR gate (for output combining)

Circuit Connections:

1. Inputs: + First 74151 (IC1): $I_0 - I_7$
 + second 74151 (IC2): $I_8 - I_{15}$

2. Select lines: + common S_0, S_1, S_2 : (connected to both ICs).

+ S_3 (4th select bit): + IC1 Enable ($\bar{E}$): (connected to S_3 via NOT gate)
 + IC2 Enable ($\bar{E}$): (connected to S_3 directly)

+ This ensures only one IC is active at a time.

3. Output: ~~2~~ Outputs of both ICs fed into an OR gate.

Circuit Diagram - (not drawn)

Truth Table:

S_3	$S_2 S_1 S_0$	Active IC	Output
0	0 0 0	IC1	I_0

Operation:Step-1: chip selection (S_3):

* If $S_3 = 0$:

* IC1 Enabled ($\bar{E} = 0$), IC2 Disabled ($\bar{E} = 1$).

* IC1 selects from $I_0 - I_7$ (based on S_2, S_1, S_0)

* IC2 outputs 0 (disabled)

* OR gate passes IC1's output.

* If $S_3 = 1$:

* IC1 disabled ($\bar{E} = 1$), IC2 enabled ($\bar{E} = 0$).

* IC2 selects from $I_8 - I_{15}$

* IC1 outputs 0 (disabled)

* OR gate passes IC2's output.

Step-2: Input Selection (S_2, S_1, S_0):

→ The 3 LSBs (S_2, S_1, S_0) select 1 of 8 inputs in the active IC.

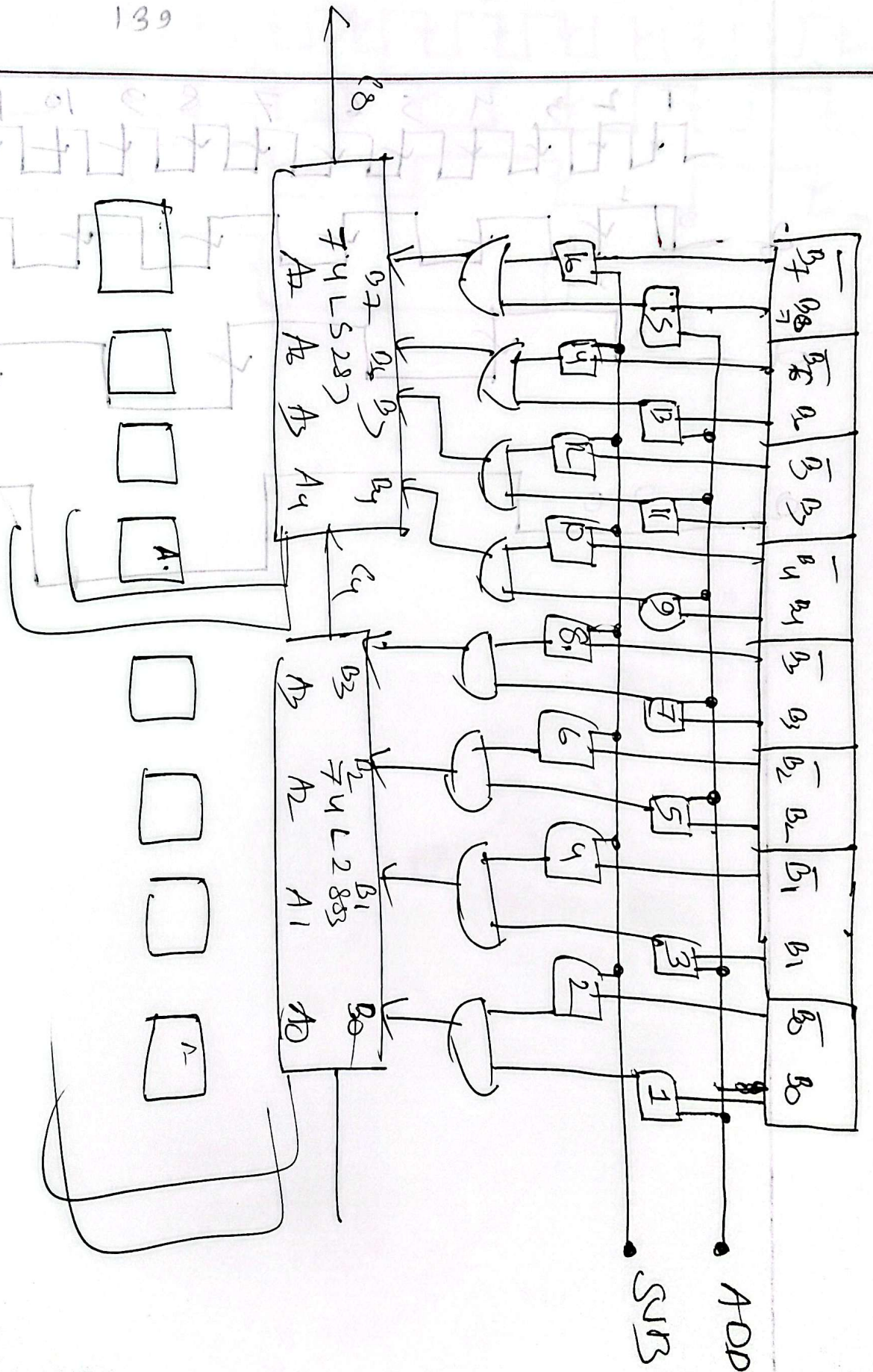
Example:

→ $S_3 S_2 S_1 S_0 = 1011$ (11)

* $S_3 = 1 \rightarrow$ IC2 active

* $S_2 S_1 S_0 = 011 \rightarrow$ select I_{11} in IC

* final output = I_{11}



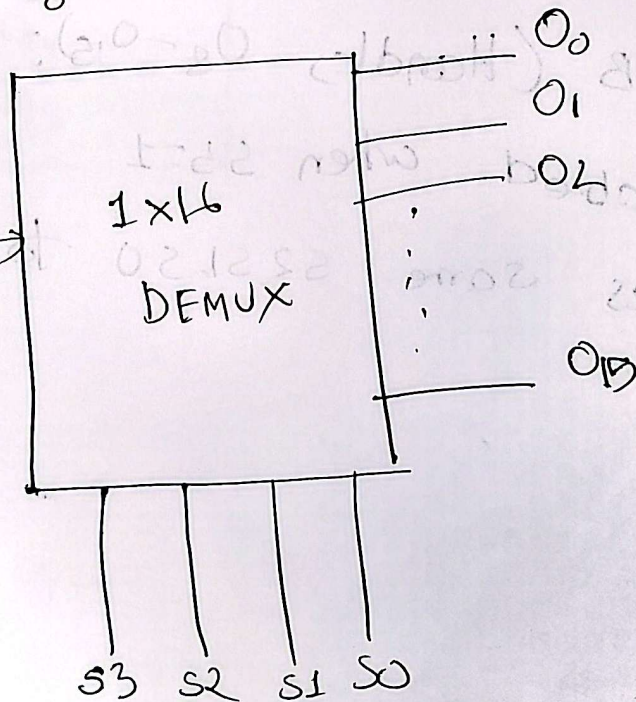
Final (19-20)

8. (b) with appropriate diagram, explain the working principle of a 1 line to 16 line Demultiplexer.

⇒ A 1-to-16 DEMUX can be designed using two 1-to-8 DEMUX units by using an extra control signal to determine which DEMUX is active at a given time.

Components Required:

- 1 data Input $\rightarrow I$
- 4 select lines $\rightarrow s_3, s_2, s_1, s_0$
- 2 $\times$ 1:8 demultiplexers
- 1 $\times$ 1:2 demultiplexers
- 16 outputs $\rightarrow O_0$ to O_{15}



Detailed

Working Principles

Step 1: 1:2 DEMUX (chip Selection)

* Input: Signal Data Signal I

* Select: S_3 (MSB of 4-bit Control)

* Outputs:

→ when $S_3 = 0$:

Routes 1 to 1:8 DEMUX A

→ when $S_3 = 1$: Routes 1 to 1:8 DEMUX B

Step 2: 1:8 DEMUX (channel Selection)

* DEMUX A (Handles $O_0 - O_7$):

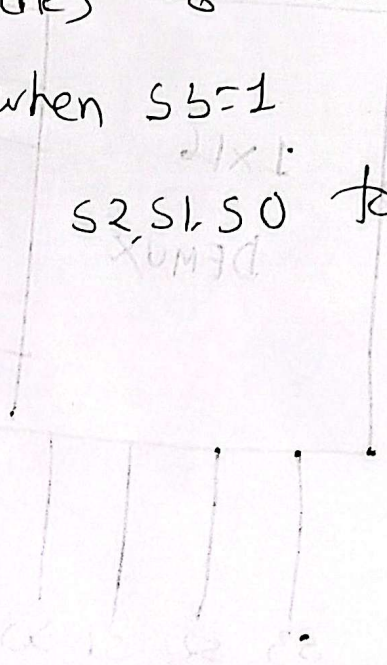
→ Enabled when $S_3 = 0$

→ Uses S_2, S_1, S_0 to select one of 8 outputs

* DEMUX B (Handles $O_8 - O_{15}$):

→ Enabled when $S_3 = 1$

→ Uses same S_2, S_1, S_0 to select one of 8 outputs



Truth table:

③ 8/10/21

S3	S2	S1	S0	A selected Output
0	0	0	0	O ₀
0	0	0	1	O ₁
0	0	1	0	O ₂
0	0	1	1	O ₃
0	1	0	0	O ₄
0	1	0	1	O ₅
0	1	1	0	O ₆
0	1	1	1	O ₇
1	0	0	0	O ₈
1	0	0	1	O ₉
1	0	1	0	O ₁₀
1	0	1	1	O ₁₁
1	1	0	0	O ₁₂
1	1	0	1	O ₁₃
1	1	1	0	O ₁₄
1	1	1	1	O ₁₅

DEWA X 1x1 8/10/21

Examples

Let's say $I = 1$:

$$I = 1$$

$$S_3 S_2 S_1 S_0 = 1011 \rightarrow (11)$$

Then, $S_3 = 1 \rightarrow 1:2$ DEMUX route) $I = 1$ to DEMUX B
($O_8 - O_{15}$)

$$S_2 S_1 S_0 = 011 \rightarrow \text{DEMUX B selects } O_{11}$$

Output: $O_{11} = 1$, all others = 0.

8(c) $\rightarrow$ ~~67C~~ ~~67Ch~~.

Final (17-18)

7(a) $\rightarrow$ Explain the working principle of 74138 decoder IC.

$\Rightarrow$ The 74138 IC is a 3-to-8 line decoder with active low outputs and three enable inputs. It takes 3 binary inputs and activates one of 8 outputs based on the combination of inputs — but only when enabled.

2. Apply 3 bit Binary Inputs (7)

→ The select lines $A_2 A_1 A_0$ form a 3-bit

binary number.

→ This binary number selects one of the 8 outputs.

3. Decoding Action:

→ The output line corresponding to the input binary code goes Low (0) and all other output lines remain high (1).

4. Example:

→ If $A_2 A_1 A_0 = 101$ (5) $\overline{O_0}, \overline{O_4}, \overline{O_6}, \overline{O_7} = 1$

→ Then $\overline{O_5} = 0$ and

যদি truth table, circuit — $\overline{O_5}$ যাওয়ার বা (৬)

Chapter-8

(2)

TTL $\Rightarrow$ Transistor-Transistor-Logic

Characteristics of Logic Family / Digital IC Properties of Digital IC / TTL characteristics:

1. Fan-Out:

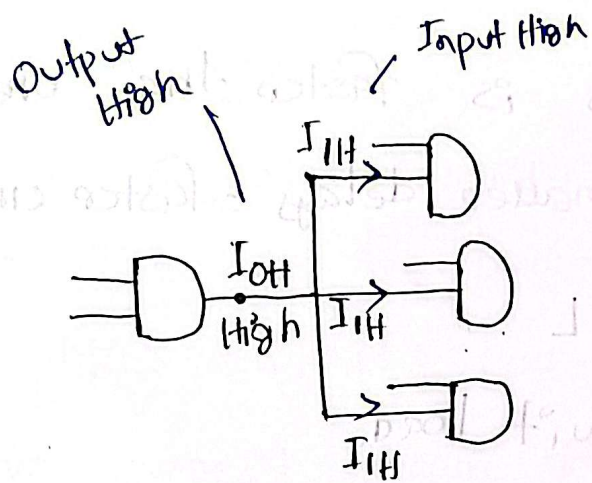
$\rightarrow$ Fan-out is the maximum number of logic gate inputs that a single logic gate output can drive reliably. It is also called loading factor.

$\rightarrow$ For example, if a logic gate has a fan-out of 10, it means:

* It can drive up to 10 other gate inputs without problems.

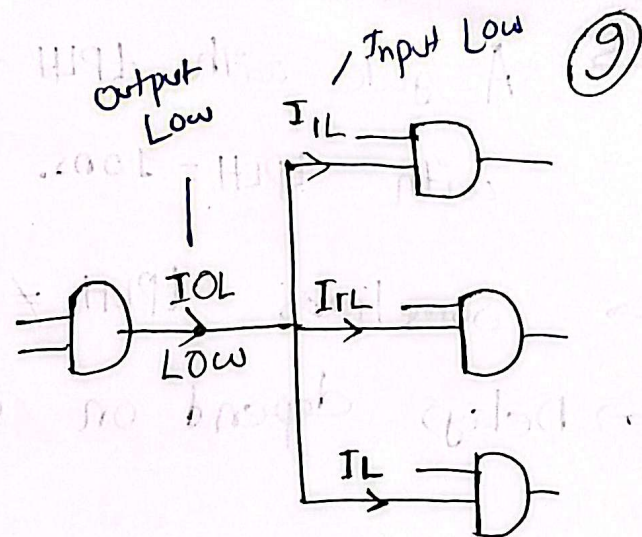
* But if it connects more than 10 inputs, it may cause logic errors.

$\Rightarrow$ It defines current supply by given gate.



- High state fan-out:

$$= \frac{I_{OH}(\text{Max})}{I_{IH}}$$



- Low state fan-out:

$$= \frac{I_{OL}(\text{max})}{I_{IL}}$$

2. Propagation Delays

→ When a signal passes through a logic gate (like an inverter), there is a small delay before the output changes, this is called propagation delay.

→ There are two types of propagation delays:

* t_{PLH} → Delay time in going from logical 0 to logical 1

(Low to High)
state

* t_{PHL} → Delay time in going from logical 1 to logical 0

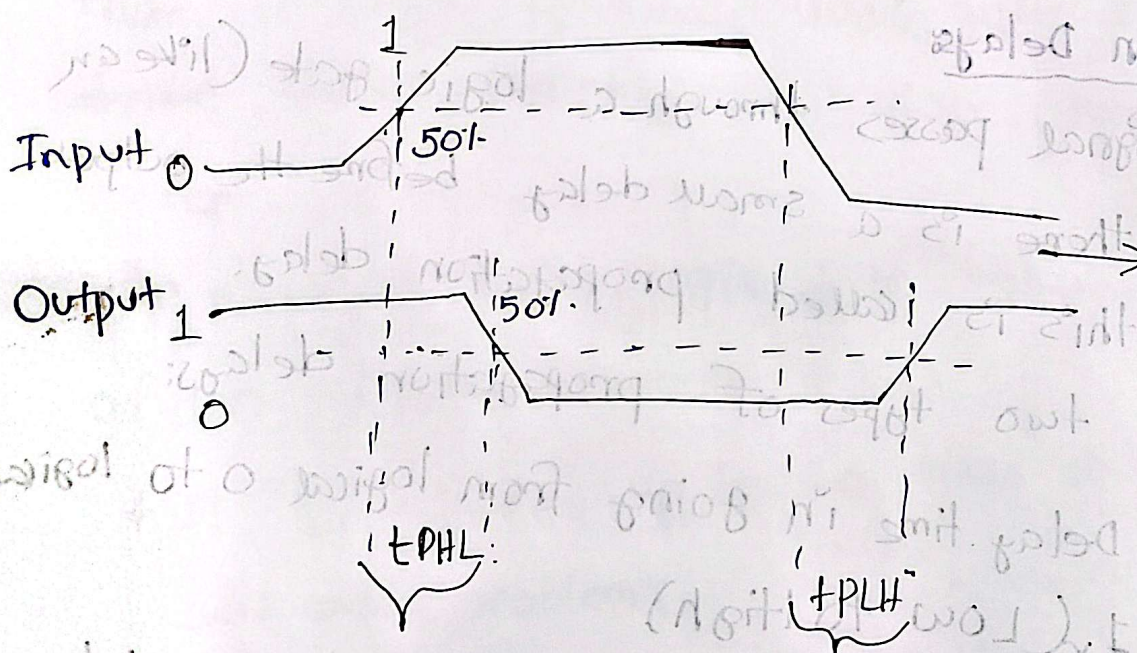
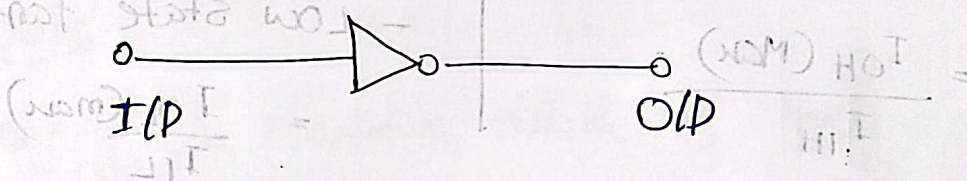
(High to Low)
state.

→ It is measured between the 50% voltage points of input and output signals.

→ A gate with $t_{PLH} = 5\text{ns}$ is faster than one with $t_{PLH} = 10\text{ns}$. Smaller delays = faster circuit. (10)

→ Sometimes $t_{PLH} \neq t_{PHL}$

→ Delays depend on circuit load



$$\therefore t_p = t_{PLH} + t_{PHL}$$

→ It is measured between the output and input signals.

3. Power Requirements

(11)

→ Every Integrated circuit (IC) needs electrical power to work.

+ For TTL ICs, the power supply is called VCC.

+ For MOS ICs, it's called VDD.

→ The current drawn from the supply varies depending on the logic gates of the circuits on the chip.

+ I_{CCH} = current when all outputs are High.

+ I_{CCL} = current when all outputs are Low.

→ For example:

A NAND gate chip draws:

+ More current (I_{CCL}), when outputs are Low.

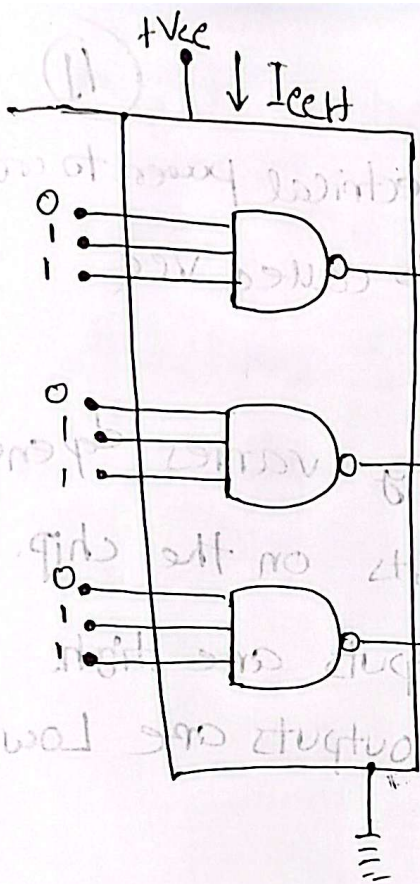
+ Less current (I_{CCH}), when outputs are High.

→ Since outputs may be High or Low half of the time, we take the average current:

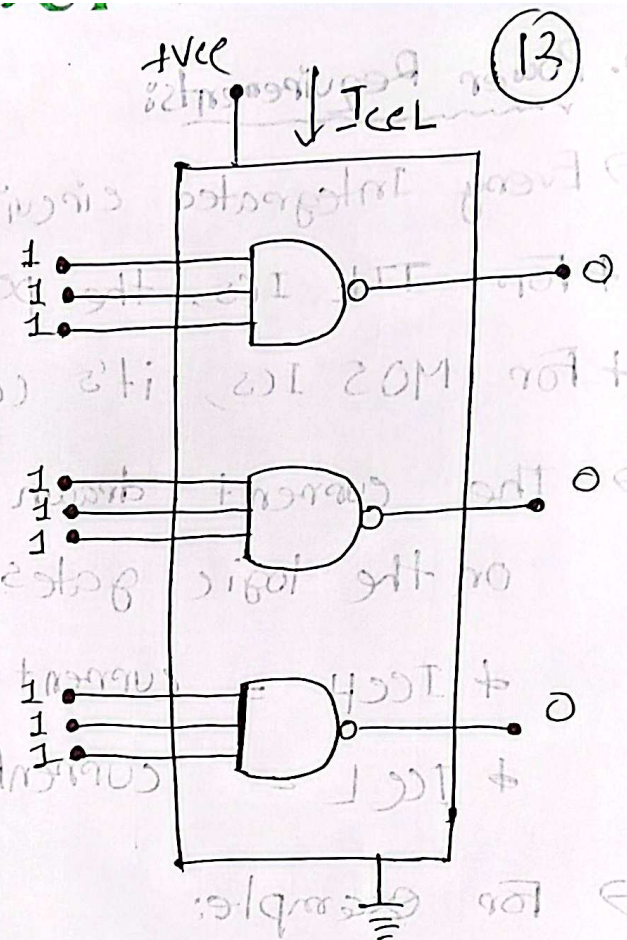
$$I_{CC}(\text{avg}) = \frac{I_{CCH} + I_{CCL}}{2}$$

→ Average power is

$$P_D(\text{avg}) = I_{CC}(\text{avg}) \times V_{CC}$$



(a)



(b)

4. Noise Immunity:

- Noise is an unwanted signal caused by stray electric and magnetic fields.
- the noise immunity of a logic circuit refers to the circuit's ability to tolerate noise without errors in its output.
- A quantitative measure of noise immunity is called noise margin. There are two types:

$V_{OH}(\min) \rightarrow$ It is min O/P volt for logic 1 $I_{OH} \rightarrow$ high level O/P current

$V_{OL}(\max) \rightarrow$ It is max O/P volt for logic 0 $I_{OL} \rightarrow$ low level O/P current

$V_{IH}(\min) \rightarrow$ It is min I/P volt for logic 1 $I_{IH} \rightarrow$ high level I/P current

$V_{IL}(\max) \rightarrow$ It is max I/P volt for logic 0 $I_{IL} \rightarrow$ low level I/P current

$\rightarrow$ There are two types of noise margin:

1. High state Noise Margin:

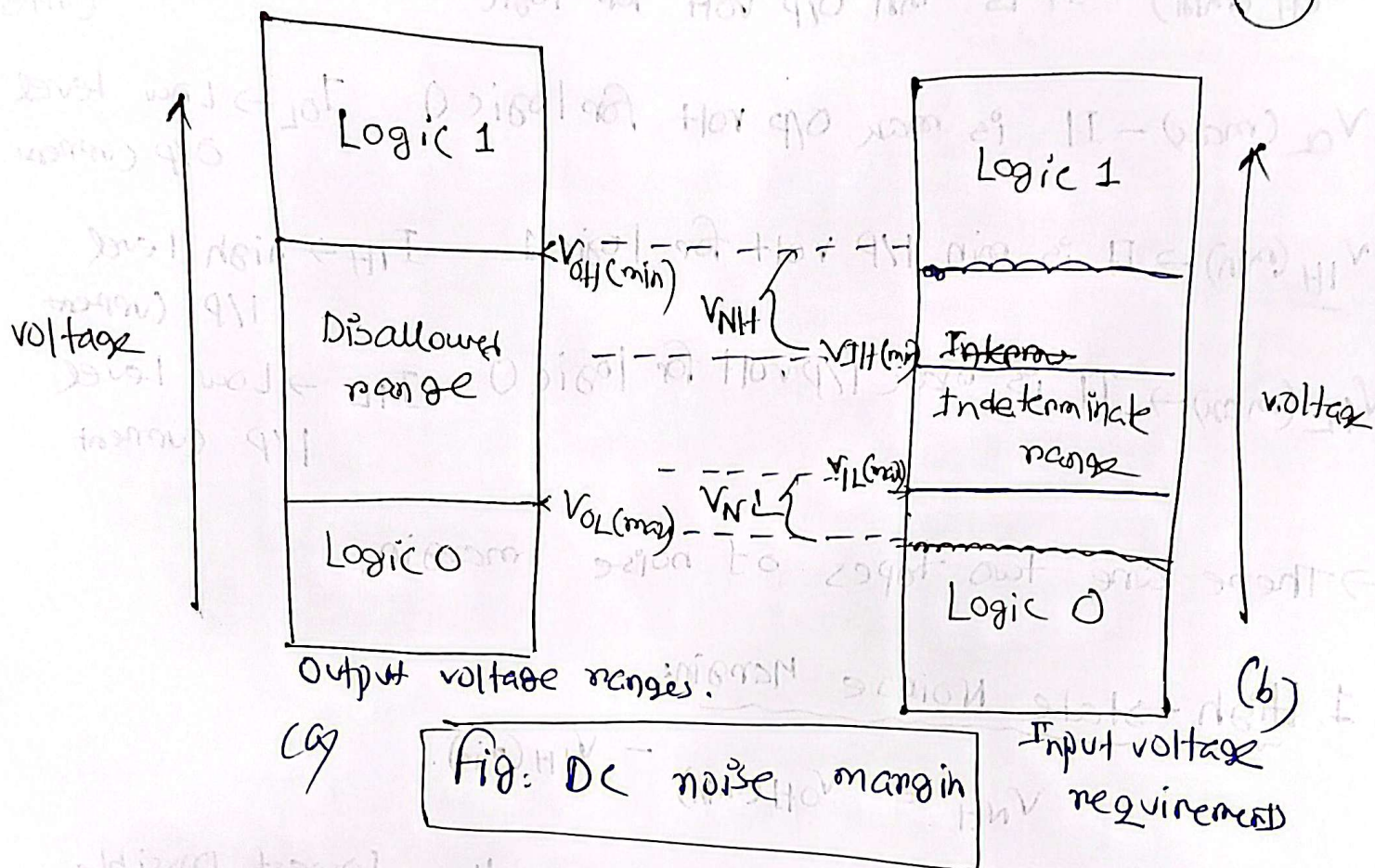
$$V_{NH} = V_{OH}(\min) - V_{IH}(\min)$$

V_{NH} is the difference between the lowest possible high output and the minimum input voltage required for a high.

2. Low state Noise Margin:

$$V_{NL} = V_{IL}(\max) - V_{OL}(\max)$$

V_{NL} is the difference between the largest possible low output and the maximum input voltage required for a low.



5. Invalid Voltage Levels

→ For a logic circuit to work correctly, the input voltage must be:

• Less than 0.8V (for logic 0) or

• Greater than 2.0V (for logic 1).

→ Voltages between 0.8V and 2.0V are invalid and cause unpredictable output.

→ These invalid voltages can happen if:

* A logic gate is malfunctioning.

* Its fan-out is exceeded.

* The power supply is unstable.

(15)

6. Current Sourcing and Current Sinking Actions

* Current Sourcing: When the output of gate 1 is in High (1) state, it provides a current I_{OH} to the input of the next gate 2. Thus, the output of gate 1 is acting as a source of current for the gate 2 input.

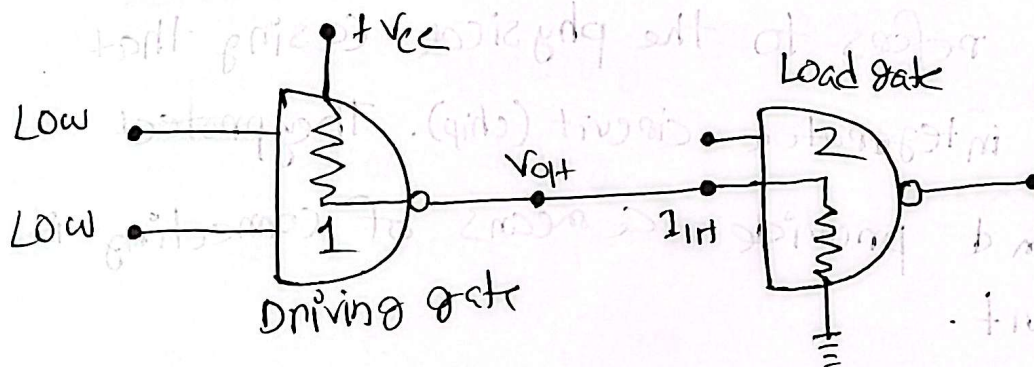


Fig. Current Sourcing: Driving gate supplies current to load gates in High state.

* Current Sinking: When the output of gate 1 is in Low (0) state, it receives current from the input circuit of gate 2. Thus, the input of gate 2 is acting as a sink and must be able to sink a current I_{IL} .

3. SOIC (Small Outline IC):

(17)

→ Smaller version of DIP, used in SMT.

4. QFP (Quad Flat Package)

→ Leads on four sides.

5. BGA (Ball Grid Array)

→ Uses solder balls instead of pins for high density.

Characteristics of TTL logic gates

→ Operates typically at +5V DC.

→ Propagation delay is around 10ns to 30ns.

→ Consumes more power than CMOS; about 10mW per gate.

→ Standard TTL gates can drive up to 10 inputs of other TTL gates.

→ Can't sink more current than it can source.

→ Commonly found in DIP.

→ Noise Margin:

High = $\approx 0.4V$

Low = $\approx 0.4V$

→ Voltage Levels:

Logic Low: 0V to 0.8V

Logic High: 2V to 5V

Invalid: 0.8 to 2V

Ques: A certain TTL IC output is rated at $I_{OH}(\text{max}) = 800 \mu\text{A}$ and $I_{OL} = 48 \text{ mA}$. Express the IC fan-out in terms of unit loads. [Consider $1 \text{ UL} = 40 \mu\text{A}$ in High state and 1.6 mA in the Low state].

Given,

$$I_{OH}(\text{max}) = 800 \mu\text{A}$$

$$I_{OL} = 48 \text{ mA}$$

$$1 \text{ UL} = (\text{Unit Load}) = 40 \mu\text{A} (\text{High})$$

$$1 \text{ UL} = 1.6 \text{ mA} (\text{Low})$$

1. Fan-out for High state:

$$\text{Fan-out High} = \frac{I_{OH}}{1 \text{ UL High}} = \frac{800 \mu\text{A}}{40 \mu\text{A}}$$

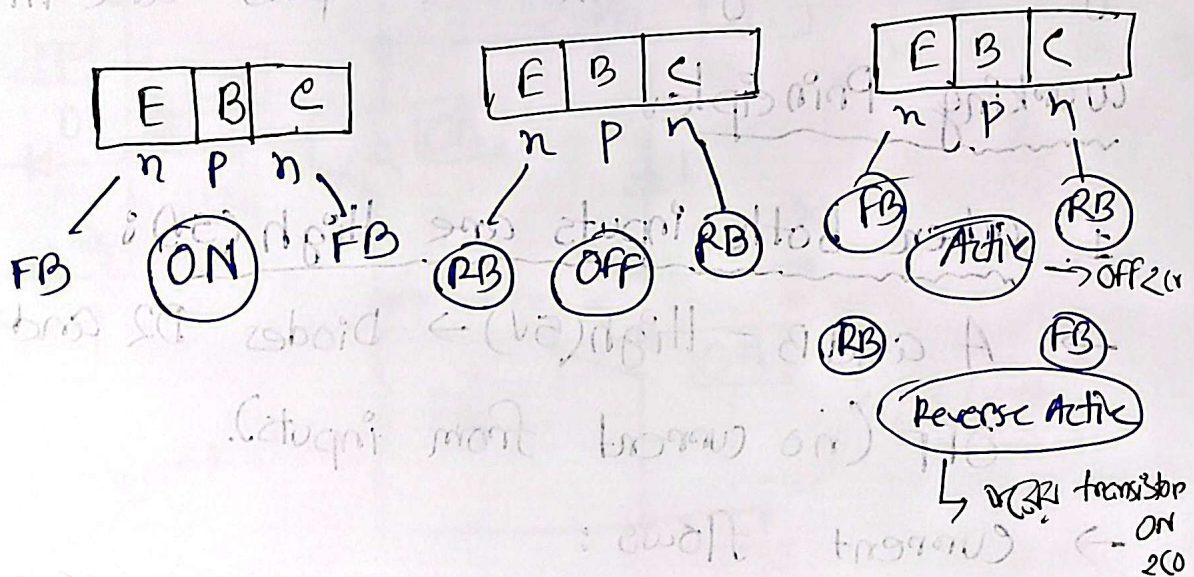
$$= 20$$

2. For Low state:

$$Fan-out = \frac{I_{OL}}{I_{IL Low}} = \frac{48 \text{ mA}}{1.6 \text{ mA}} = 30$$

Ans.

Basic:



TTL NAND Gates

(20)

A TTL NAND gate uses bipolar junction transistors to implement the NAND function. It operates based on transistors switching, where the output goes Low only when all inputs are high.

Working Principles

1. When both inputs are High (5V):

→ $A \text{ and } B = \text{High}(5V) \rightarrow$ Diodes $D2$ and $D3$ are OFF (no current from inputs).

→ Current flows:

$+5V \rightarrow R1 \rightarrow D4 \rightarrow \text{Base of } Q_2 \rightarrow Q_2 \text{ turns ON.}$

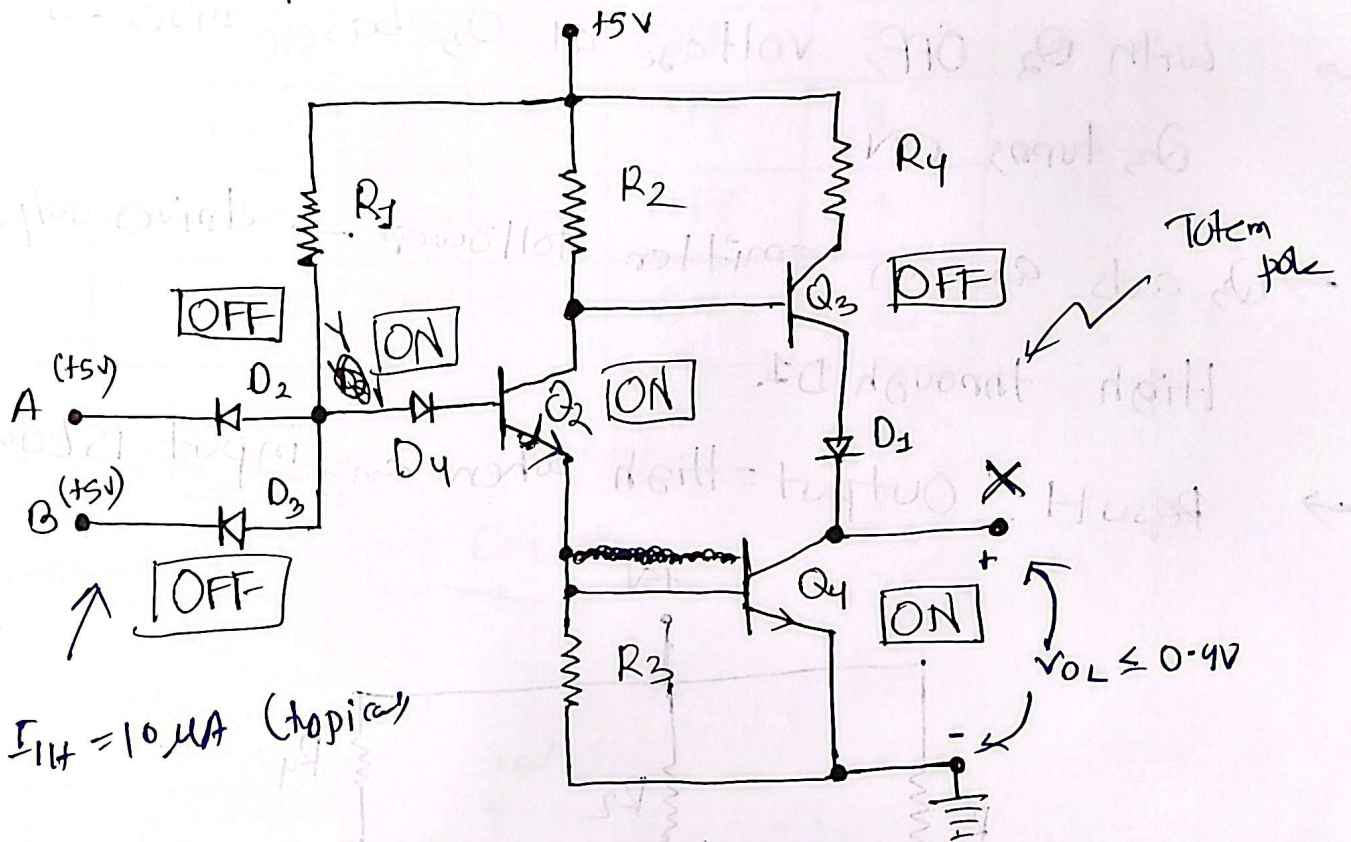
→ Q_2 turns ON $\rightarrow$ sends base current to Q_4 , turning Q_4 ON.

→ Q_4 (a saturated transistor) pulls output $\times$ Low.

→ Q_2 's collector voltage drops to $\approx 0.8V \rightarrow$ not enough to turn ON Q_3 , especially with $D1$ blocking current.

→ Q_3 stays OFF output remains Low. (2)

→ Result : Output = Low when all inputs are High.



(a) Low output

2. When either A or B is Low:

→ Example : B = Low (0V) → D_3 turns ON, current flows:
 $+5V \rightarrow R_1 \rightarrow D_3 \rightarrow GND$

→ D_3 forward-bias holds point Y at $\sim 0.7V \rightarrow$

NOT enough to turn ON Q_2 !!

→ 3. Q_2 is OFF → no base current for Q_4 → Q_4 is OFF

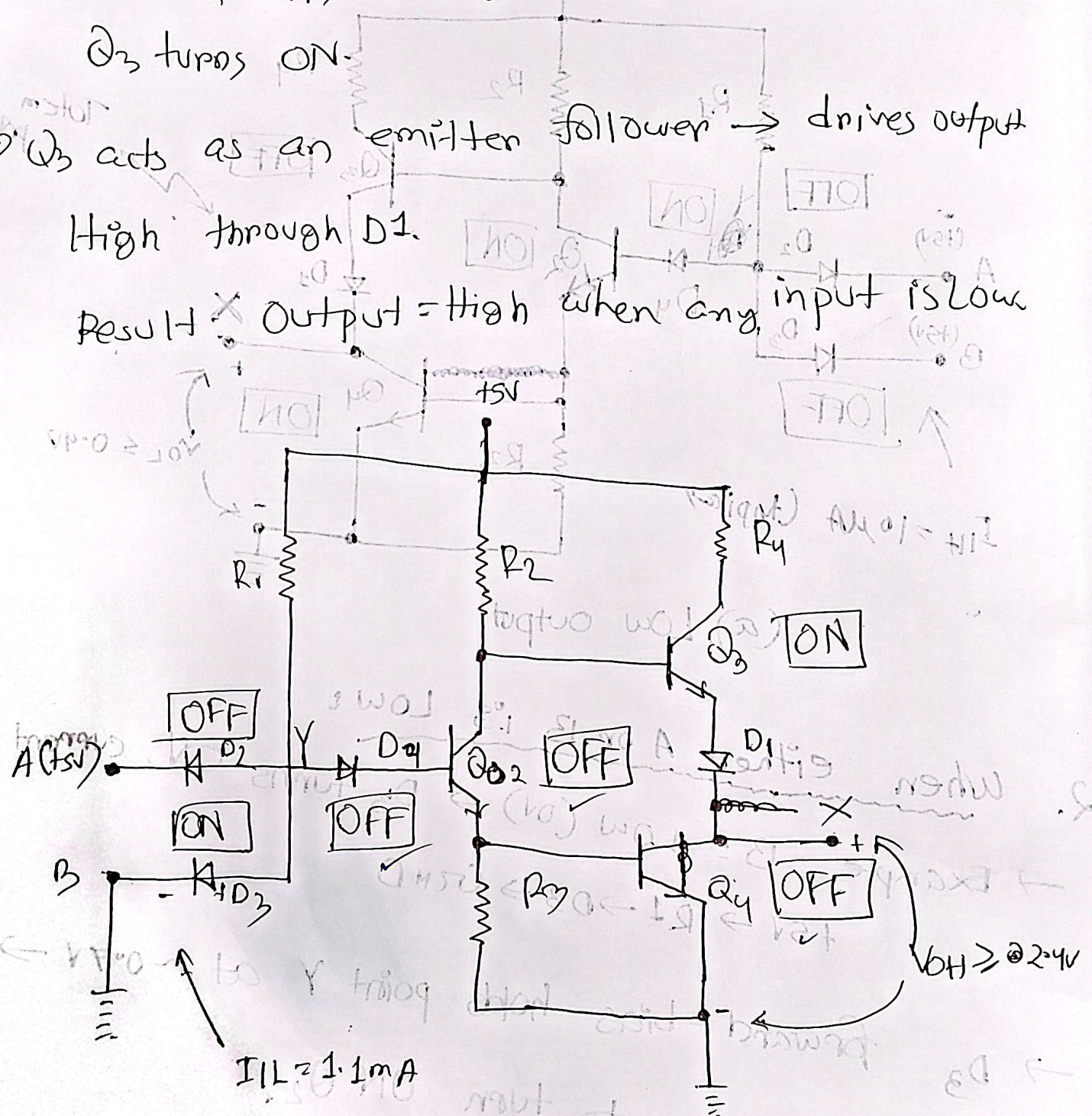
→ With Q_2 OFF, voltage at Q_3 base rises →

Q_3 turns ON.

→ Q_3 acts as an emitter follower → drives output

High through D_1 .

→ Result: Output = High when any input is Low



(b) High Output

Truth table:

(23)

A	B	Q ₁	Q ₂	Q ₃	Q ₄	Output
0	0	ON	OFF	ON	OFF	1
0	1	ON	OFF	ON	OFF	1
1	0	ON	OFF	ON	OFF	1
1	1	OFF	ON	OFF	ON	0

TTL NOR gates

Totem-Pole Output Circuit:

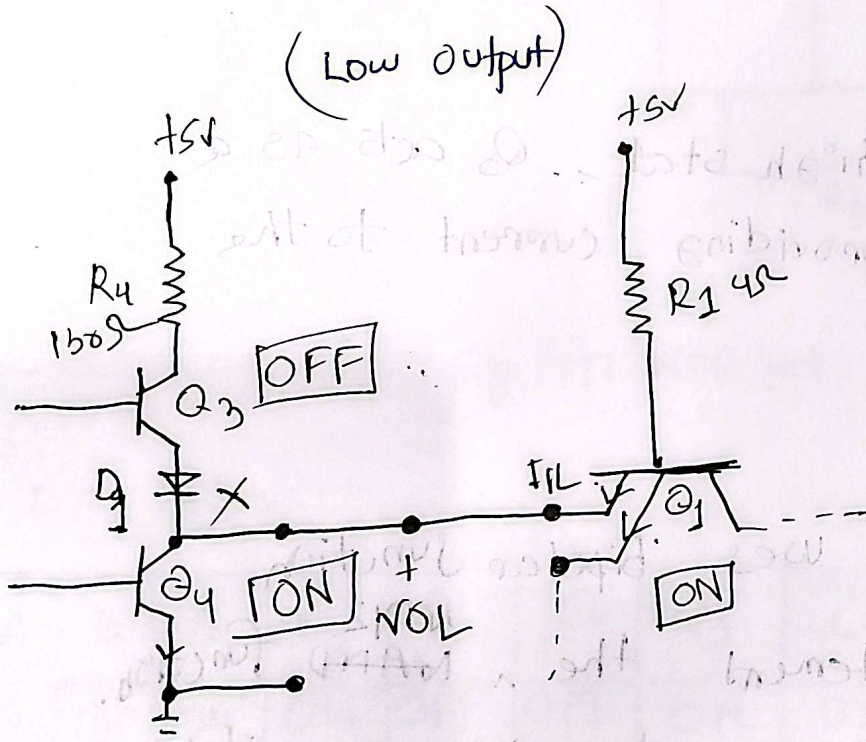


Fig: When the TTL output is in the LOW state, Q₄ acts as a current sink, deriving its current from the load.

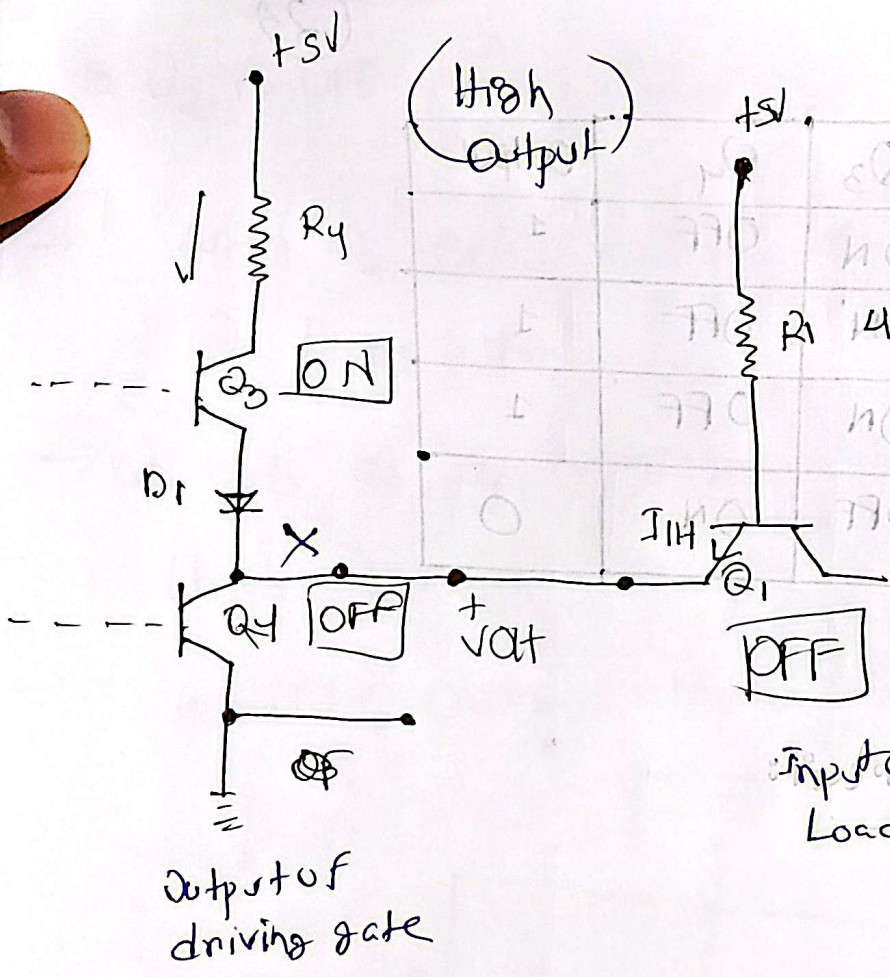


Fig: In the output high state, Q_3 acts as a current source, providing current to the load gate.

TTL NOR gate:

A TTL NOR gate uses bipolar junction transistors to implement the NOR function. It operates based on transistors switching where the output ~~at~~ goes High only when all inputs are Low otherwise Low outputs.

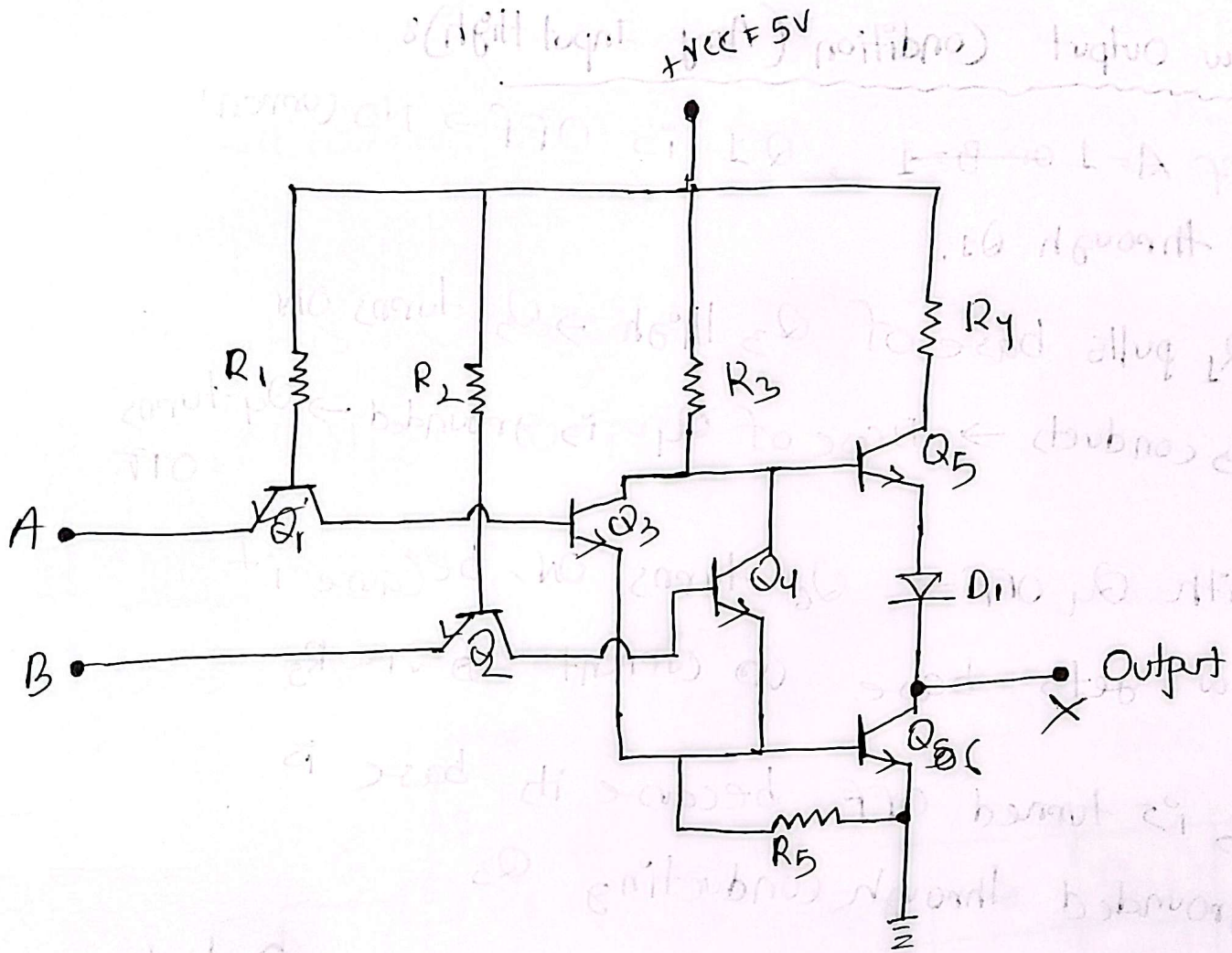


Fig. TTL NOR gate.

Truth tables

A	B	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	Q ₆	Output	D1
0	0	ON	ON	OFF	OFF	ON	OFF	1	ON
0	1	ON	OFF	OFF	ON	OFF	ON	0	OFF
1	0	OFF	ON	ON	OFF	OFF	ON	0	OFF
1	1	OFF	OFF	ON	ON	OFF	ON	0	OFF

Working Principles

(26)

1. Low Output Condition (Any Input High):

- If $A=1$ or $B=1$, Q_1 is OFF → NO current through Q_1 .
- R_1 pulls base of Q_3 high → Q_3 turns ON
- Q_3 conducts → Base of Q_4 is grounded → Q_4 turns OFF
- With Q_4 OFF → Q_6 turns ON, because it now gets base current via R_5
- Q_5 is turned OFF because its base is grounded through conducting Q_3
- Q_6 connects the output to ground → Output is LOW.

2. High Output Condition ($A=0, B=0$):

- Q_1 and Q_2 turn ON → They conduct to ground.
- This Q_1 and Q_2 in saturation pull their

collectors Low $\rightarrow$ so Q_3 and Q_4 turn OFF. (27)

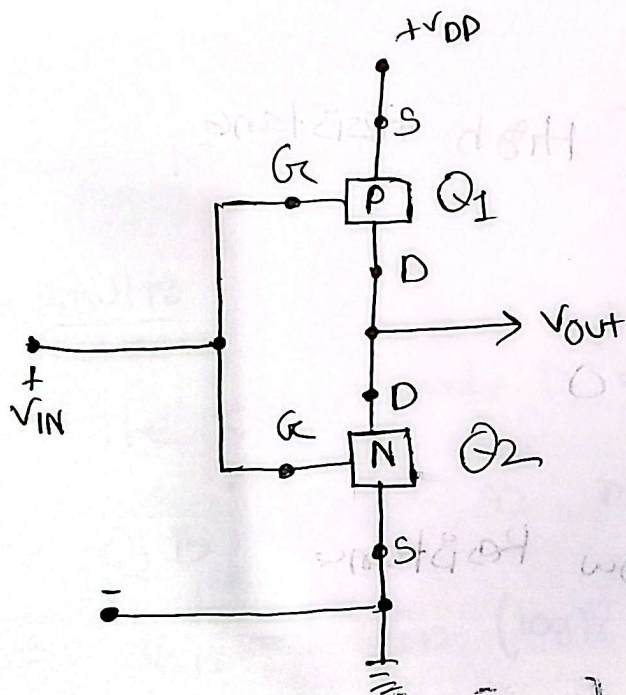
$\rightarrow Q_3$ and Q_4 OFF means Q_{56} remains OFF, which allows Q_5 to turn ON, pulling the output High.

$\rightarrow$ Output is connected to V_{CC} via Q_5

$\rightarrow$ Result: High Output.

CMOS Inverter

CMOS $\rightarrow$ Complementary Metal Oxide Semiconductor



Input	Q_1	Q_2	Output
1	0	1	0
0	1	0	$1/V_{DD}$

Here,

$$G_i = 1, P = 0, N = 1$$

$$G_i = 0, P = 1, N = 0$$

Working Principles

(28)

A CMOS inverter consists of two MOSFETs:

→ Q1: P-channel MOSFET (PMOS) → on top.

→ Q2: N-channel MOSFET (NMOS) → on bottom.

Both are connected in series between V_{DD} and GND.

Case: 1: $V_{IN} = +V_{DD}$ (logic 1):

• PMOS (Q1):

→ Gate = $+V_{DD}$, Source = $+V_{DD}$

→ SO, $V_{GS} = 0$

→ PMOS is OFF → very High Resistance

• NMOS (Q2):

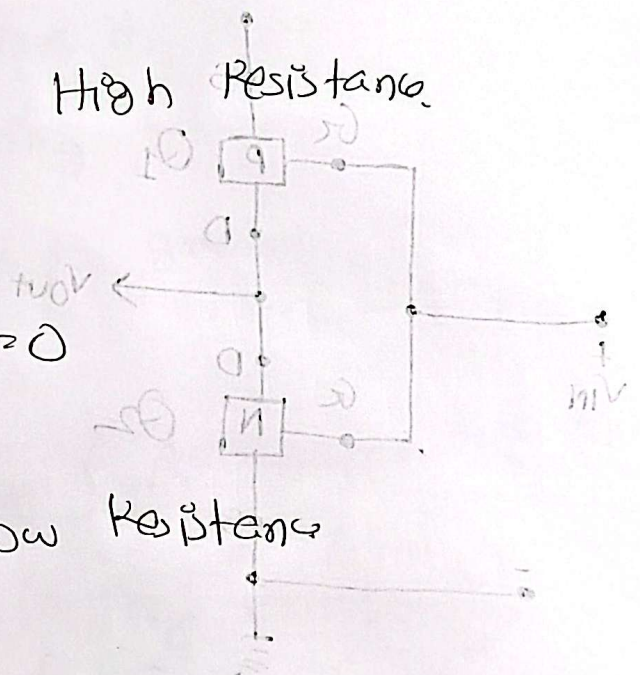
→ Gate = $+V_{DD}$, Source = 0

→ SO, $V_{GS} = +V_{DD}$

→ NMOS is ON → Low Resistance

• Results:

→ The output node is connected to ground through Q2 (ON).



→ Q_1 is OFF, so no connection to V_{DD} . (2)

→ $V_{out} = 0V$ (logic 0).

Case: 2 $V_{IN} = 0V$ (logic 0):

• PMOS (Q_1):

→ Gate = $0V$, source = $+V_{DD}$

→ So, $V_{GS} = -V_{DD}$

→ PMOS is ON → Low Resistance

• NMOS (Q_2):

→ Gate = $0V$, source = $0V$

→ So, $V_{GS} = 0V$

→ NMOS is OFF → Very high Resistance

• Results

→ The output node B connected to V_{DD} through Q_1 (ON)

→ Q_2 is OFF, so no connection to ground

→ $V_{out} = +V_{DD}$ (logic 1).

CMOS NAND gates

(30)

The CMOS NAND gate is constructed by modifying the basic CMOS inverter. It consists of:

- * Two P-channel MOSFETs (PMOS) at the top, connected in parallel between V_{DD} and the output.

- * Two N-channel MOSFETs (NMOS) at the bottom, connected in series between the ground and the output.

- * Inputs A and B are connected to the gates of both PMOS and NMOS pairs.

Key Concepts

- 0V (Logic 0) input $\Rightarrow$ turns ON PMOS, OFF NMOS.
- $+V_{DD}$ (Logic 1) input $\Rightarrow$ turns OFF PMOS and ON NMOS.

CMOS NAND gate

(31) i. short

Circuit:

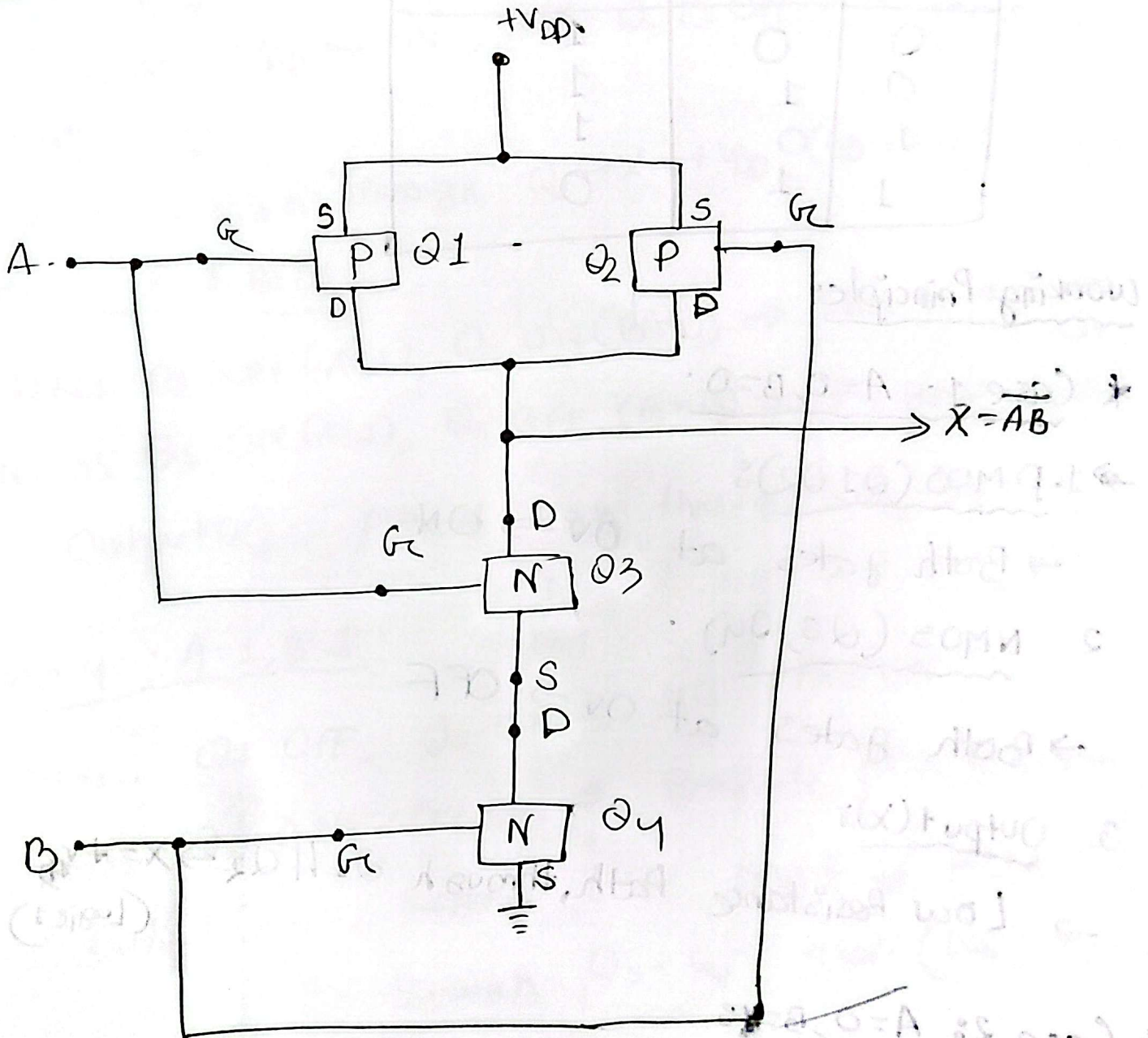


Fig. CMOS-NAND gate.

Truth table:

(32)

A	B	Output-X
0	0	1
0	1	1
1	0	1
1	1	0

Working Principles:

* Case 1: $A=0, B=0$:

→ 1. PMOS (Q_1, Q_2):

→ Both gates at 0V → ON

2. NMOS (Q_3, Q_4):

→ Both gates at 0V → OFF

3. Output (X):

→ Low Resistance Path through $Q_1 || Q_2 \rightarrow X = +V_{DD}$
(Logic)

* Case 2: $A=0, B=1$:

1. PMOS (Q_1):

→ Gate at 0V → ON

2. PMOS (Q_2):

→ Gate at $+V_{DD}$ = OFF

3. NMOS (Q_3)

→ Gate at 0V → OFF

4. NMOS (Q_4)

→ Gate at $+V_{DD}$ → ON, but Q_3 is OFF → no ground path.

5. Output (X):

→ pulled high through Q_1 → $X = +V_{DD}$ (Logic 1).

* Case-3 : $A=1, B=0$

1. PMOS Q_1 OFF ($A=1$), Q_2 ON ($B=0$) → still at least one ON

2. NMOS Q_3 ON ($A=1$), Q_4 OFF ($B=0$) → NO ground path

3. Output (X): pulled high through Q_2 → $X = +V_{DD}$ (Logic 1)

* Case 4 : $A=1, B=1$

1. PMOS Q_1 OFF, Q_2 ON → NO PMOS ON

2. NMOS Q_3 ON, Q_4 ON → complete path to ground

3. Output (X): pulled Low Resistance path through $Q_3 + Q_4$ → 0V (Logic 0).

CMOS NOR gate

(34)

The CMOS NOR gate is constructed by modifying the basic CMOS inverter. It consists of

→ Two P-channel MOSFETs (PMOS) at the top connected in series between V_{DD} and the output.

→ Two N-channel MOSFETs (NMOS) at the bottom, connected in parallel between output and the ground.

→ Inputs A and B are connected to the gates of both PMOS and NMOS pairs.

Key Concepts

→ $0V$ (Logic 0) input → turns on PMOS, OFF NMOS

→ V_{DD} (Logic 1) input → turns OFF PMOS, ON NMOS.

Circuit:

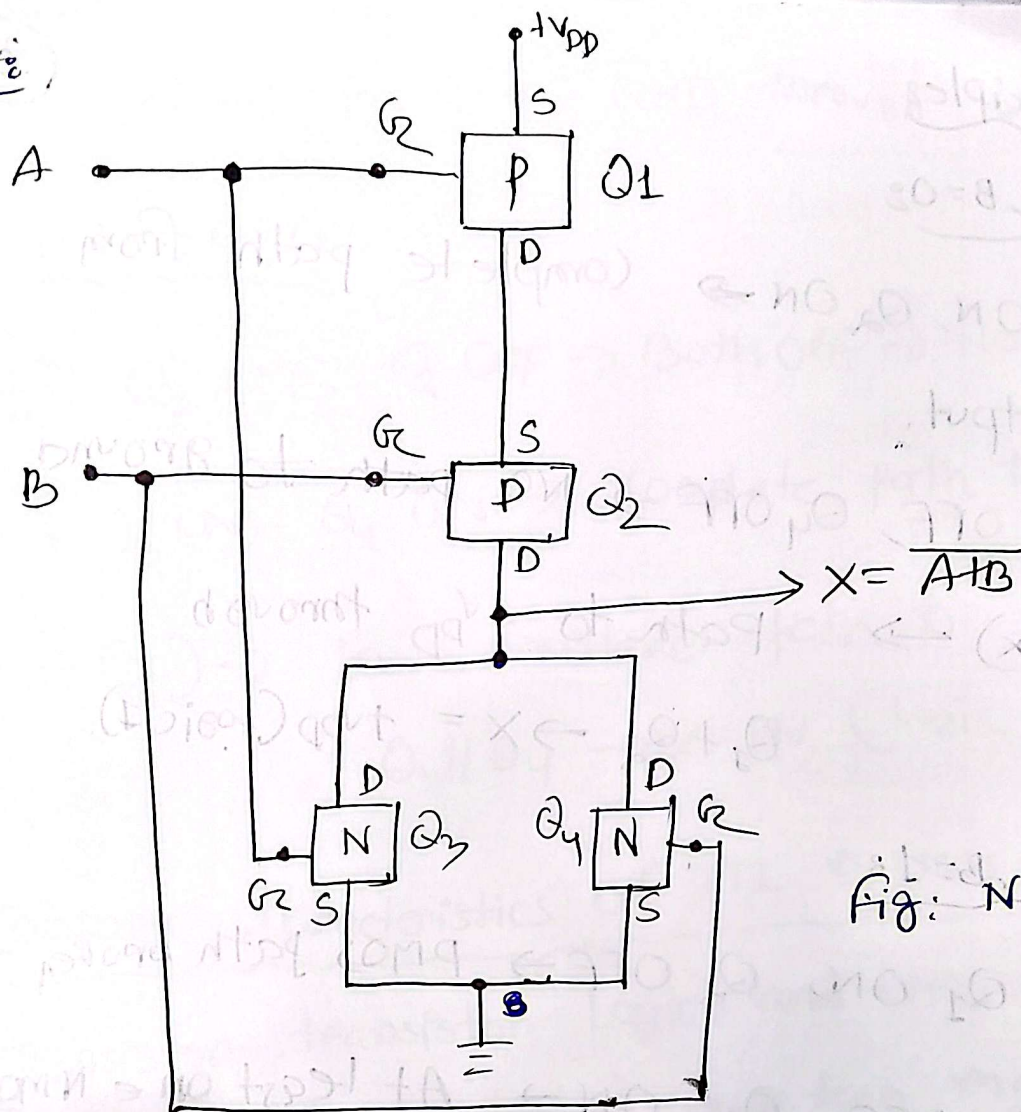


Fig: NOR gate CMOS.

Truth Table:

A	B	Output, X
0	0	1
0	1	0
1	0	0
1	1	0

Working Principles

(36)

* Case 1: $A=0, B=0$

* PMOS Q_1 ON, Q_2 ON $\Rightarrow$ complete path from V_{DD} to output.

* NMOS Q_3 OFF, Q_4 OFF $\Rightarrow$ NO path to ground.

* Output (X) $\rightarrow$ Path to $+V_{DD}$ through $Q_1 + Q_2 \rightarrow X = +V_{DD}$ (Logic 1).

* Case 2: $A=0, B=1$

$\rightarrow$ PMOS Q_1 ON, Q_2 OFF $\Rightarrow$ PMOS path broken.

$\rightarrow$ NMOS Q_3 OFF, Q_4 ON $\Rightarrow$ At least one NMOS ON $\rightarrow$ GND path.

$\rightarrow$ Output (X) $\rightarrow$ path to GND through $Q_4 \rightarrow X = 0V$ (Logic 0).

* Case 3: $A=1, B=0$

$\rightarrow$ PMOS Q_1 OFF, Q_2 ON $\Rightarrow$ PMOS path broken.

$\rightarrow$ NMOS Q_3 ON, Q_4 OFF $\Rightarrow$ At least one NMOS ON.

→ Output (X) → path to GND through $Q_3 \rightarrow X = 0V$ (Logic 0) (57)

* Case 4) $A=1, B=1$:

→ PMOS Q_1 OFF, Q_2 OFF → Both OFF → No path to V_{DD}

→ NMOS Q_3 ON, Q_4 ON → complete path to GND

→ Output (X) → low resistance path to GND through $Q_3 \parallel Q_4 \rightarrow X = 0V$ (Logic 0).

Ex Compare characteristics of TTL vs CMOS Logic Gates:

TTL (Transistor-transistor Logic) and CMOS (Complementary Metal Oxide - Semiconductor) are two major Logic Families used in digital circuit. These gates mainly differ in power usage, speed, noise resistance. TTL has higher power consumption and faster switching but limited noise immunity and fan-out. CMOS uses less power, has higher noise immunity, and can work over a wider voltage range.

Characteristic	TTL (Transistor-transistor Logic)	CMOS (Complementary Metal Oxide Semiconductor) (30)
1. Logic Levels	0V (Low), 5V (High) / V_{cc}	0V (Low), V_{DD} (High)
2. Power Consumption	High (specially when idle)	Very Low (almost zero when idle)
3. Speed (Propagation Delay)	Faster in older versions.	Slower initially, but now fast.
4. Input Impedance	Low	Very high
5. Noise Margin	Lower	Higher
6. Fan-out	Typically 10	Very high
7. Power-Supply Requirement	Fixed at 5V	3V - 15V
8. Temperature Sensitivity	Less	More
9. Cost	Cheaper Generally	Slightly costlier in some applications
10. Integration Density	Moderate	Very High
11. Output Drive Capability	Stronger	Weaker output drive.

Final (21-22)

(39)

1. Design a parity bit generator and checker circuit. Explain their working principle using a sample bit stream.

⇒ The parity bit method is primarily used for error detection, not error-correction. It can detect single-bit errors in a transmitted data stream but cannot correct them.

(Even Parity Example)

1. Parity Bit Generator: Adds an extra bit (parity)

to a binary data stream to make the total number of 1s even (even parity) or odd (odd parity).

① Even parity: Total no of 1s in the data bits including the parity bit should be even.

② Odd parity: Total no of 1s in the data bits including the parity bit should be Odd.

Circuit Design (4 bit data + 1 parity bit) (40)

→ Input : 4-bit Data ($D_3 D_2 D_1 D_0$).

→ Output: 5-bit data ($D_3 D_2 D_1 D_0 P$), where P = parity bit.

→ Logic : $P = D_3 \oplus D_2 \oplus D_1 \oplus D_0$.

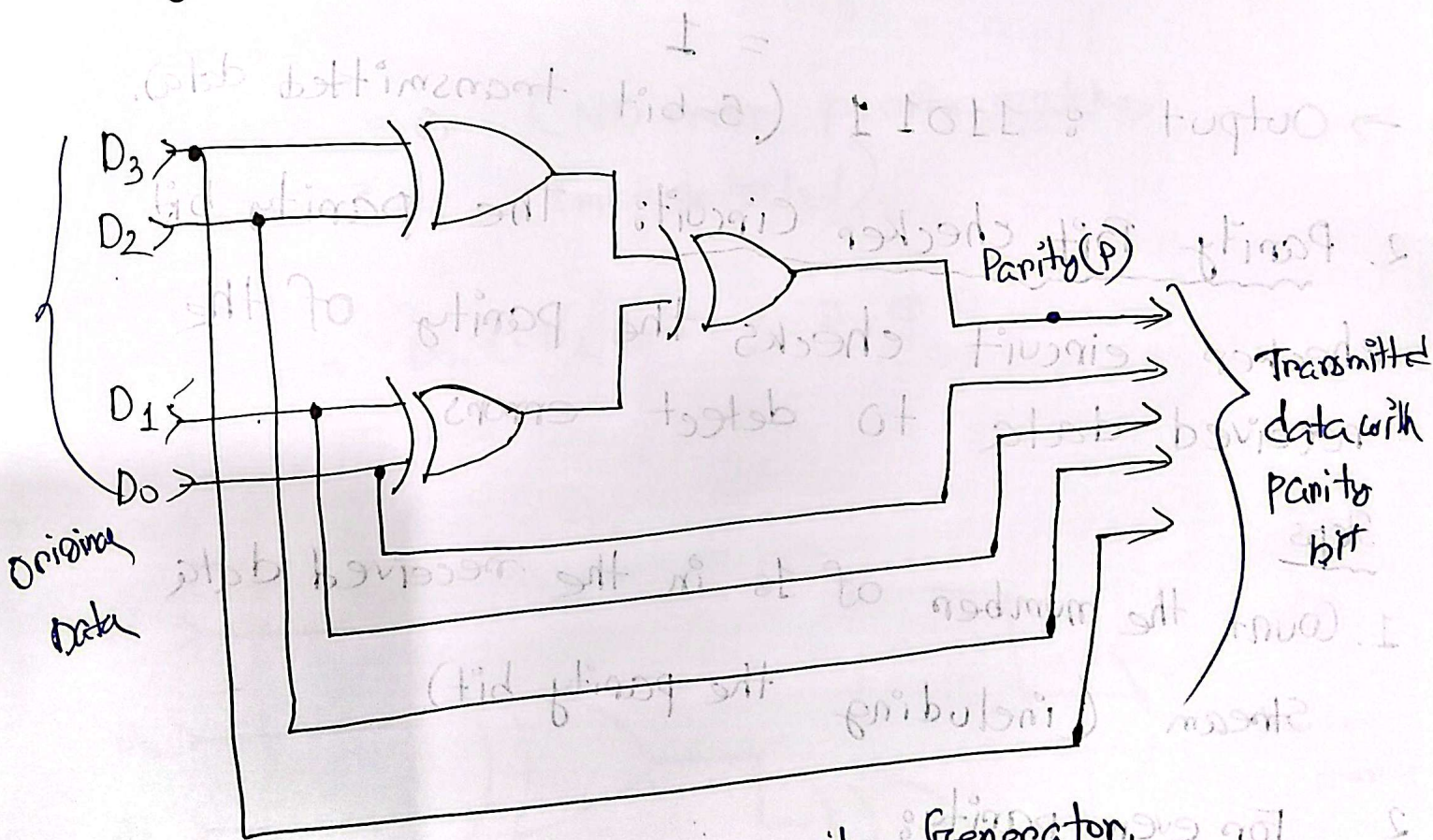


Fig: Even-parity Generator.

Working Example:

→ Data Stream: 1101 (3 bits are '1' → odd → parity bit must be '1' for even parity).

→ Calculation: $P = 1 \oplus 1 \oplus 0 \oplus 1$
 $= 1$

→ Output: 11011 (5-bit transmitted data).

2. Parity Bit checker Circuit: The parity bit

checker circuit checks the parity of the received data to detect errors.

Steps

1. Count the number of 1s in the received data stream (including the parity bit).

2. For even parity:

→ If the number of 1s is even, the data is correct.

→ If the number of 1s is odd, there is an error.

3. For Odd Parity:

→ If the number of 1s is odd, the data is correct

→ If the number of 1s is even, there is an error

Circuit Design: (5-bit received data)

* Input : 5-bit received data ($D_3 D_2 D_1 D_0 P$)

* Output :

→ Error = 0 (NO error, parity matches)

→ Error = 1 (Error detected)

* Logic:

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

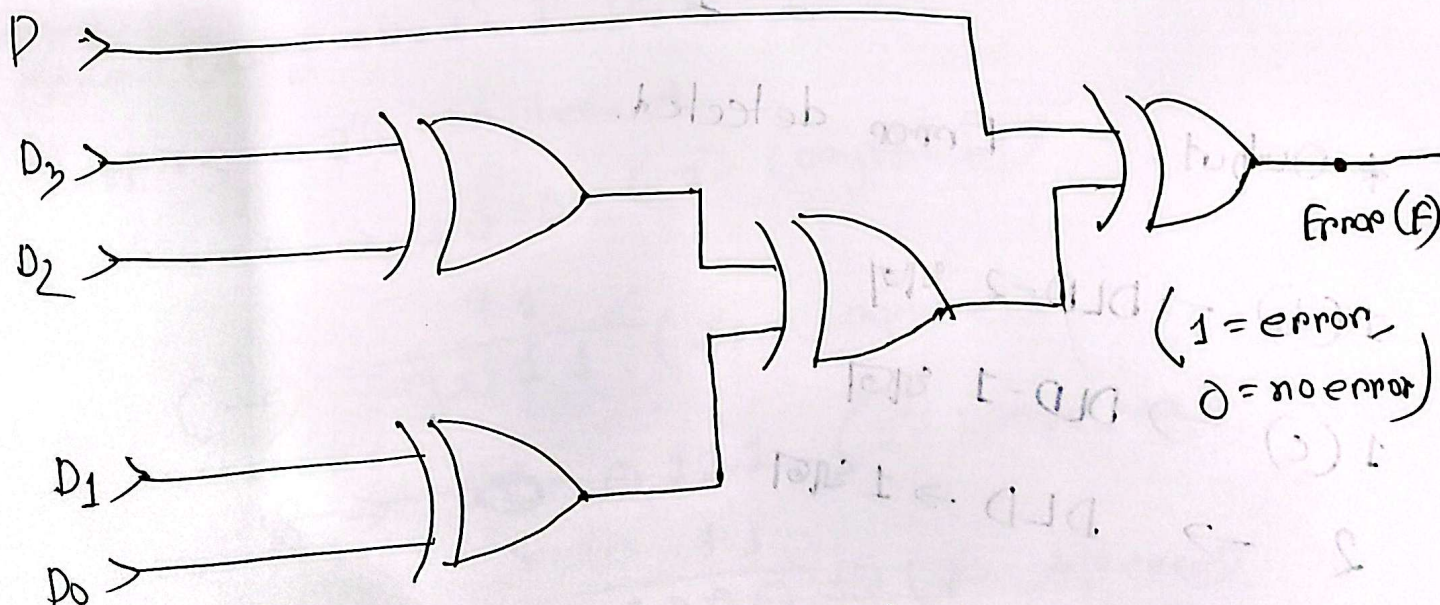


Fig: Even parity checker

Working Principles:

1. Correct Transmission: (No error)

* Received : 11011 (Even parity 4 '1's).

* Check :

$$\text{Error} = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$

* Output : NO error.

2. Incorrect Transmission (Error):

* Received : 11001 (Bit flipped: now 3 '1's - Odd).

* Check : $\text{Error} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 1$

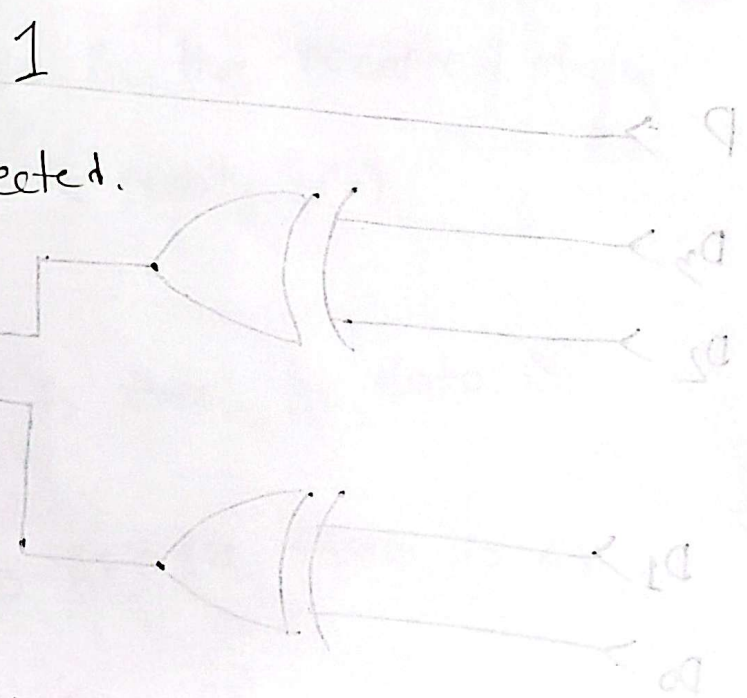
* Output :

Error detected.

1(b) → DLD-2

1(c) → DLD-1

2 → DLD → 1



3(a) $\rightarrow$ DLD -1

3(b) $\rightarrow$ Circuit for subtraction DLD-1.

now we have to compute $(-5+8)$.

Steps:

1. Represent $+5$ and 8 in 4-bit 1's complement.

$$5 = 0101$$

$$\text{Invert} = 1010$$

$$8 = 1000$$

$$\text{Invert} = \cancel{0111}$$

2. The inverted levels pass through the OR gates into the adder to be added to the bits A_0 to A_3 . Here carry in is 1. Thus the B register converted to it's 2's complement.

$$5 \rightarrow 1010 \text{ (1's complement)}$$

$$\begin{array}{r} +1 \\ \hline (-5) \quad 1011 \text{ (2's complement)} \end{array}$$

$$\begin{array}{r} \cancel{1000} + \cancel{0111} \text{ (1's complement)} \\ \hline 8 \Rightarrow \end{array}$$

$$\begin{array}{r} +1 \\ \hline 1000 \rightarrow \text{(2's complement)} \end{array}$$

3. Perform Binary Addition $(-5+8)$.

$$(-5) = 1011$$

$$(+8) = 1000$$

$$\begin{array}{r} 1011 \\ + 1000 \\ \hline 10011 \end{array}$$

Carry bit

(correct answer,

4. (a) DLD-2 $\rightarrow$ 28 page.

(b) DLD-2 $\rightarrow$ 17 page.

5. DLD-2 $\rightarrow$ 35 page

Step-1: $n=4$ bits FF = JK-FF

Step-2: Excitation table of JK-FF

Q_n Q_{n+1}

0 0
0 1

1 0

1 1

0 X
1 X

X 1
X 0

$M=0$, up counting

$M=1$, down counting

Step 4: Circuit Excitation table

M	$Q_3 Q_2 Q_1 Q_0$	$Q_3^+ Q_2^+ Q_1^+ Q_0^+$	$J_3 K_3 J_2 K_2 J_1 K_1 J_0 K_0$
0 0	0 0 0 0	0 0 0 1	0 X 0 X 0 X X 1 X
1 0	0 0 0 1	0 0 1 0	0 X 0 X 1 X X 1
2 0	0 0 1 0	0 0 1 1	0 X 0 X X 0 1 X
3 0	0 0 1 1	0 1 0 0	0 X 1 X X 1 1 X
4 0	0 1 0 0	0 1 0 1	0 X X 0 0 X X 1
5 0	0 1 0 1	0 1 1 0	0 X X 0 X 0 X 1
6 0	0 1 1 0	0 1 1 1	0 X X 1 X 1 1 X
7 0	0 1 1 1	1 0 0 0	1 X X 1 1 X 1
8 0	1 0 0 0	1 0 0 1	X 1 0 X 0 X 0 X 1 X
9 0	1 0 0 1	1 0 1 0	X 1 0 X 0 X 1 X 1 X
10 0	1 0 1 0	1 0 1 1	X 1 0 X 0 X 1 X 1 X
11 0	1 0 1 1	1 1 0 0	X 1 0 1 X 1 X 1 X
12 0	1 1 0 0	1 1 0 1	X 1 1 X 1 X 1 X
13 0	1 1 0 1	0 0 0 0	X 1 1 X 1 X 1 X
14 1	0 1 1 1		
15 1	0 1 1 1		
16 1	0 0 0 0	0 0 0 0	
17 1	0 0 0 1	0 0 0 1	
18 1	0 0 1 0	0 0 0 1	
19 1	0 0 1 0	0 0 0 1	

9	1	0	0	1	1	0	0	1	0
10	1	0	1	0	0	0	0	1	1
21	1	0	1	0	1	0	1	0	0
22	1	0	1	1	0	0	1	0	1
23	1	0	1	1	1	0	1	1	0
24	1	1	0	0	0	0	0	1	1
25	1	1	0	0	1	1	0	0	0
26	1	1	0	1	0	1	0	0	1
27	1	1	0	1	1	0	1	0	0
28	1	1	1	0	0	0	1	1	1
29	1	1	1	0	1	1	0	0	0

The minimal Expression.

By using Boolean Algebra Rules.

$$J_0 = K_0 = 1$$

$$J_0 J_1 = K_1 = Q_0 \bar{M} + \bar{Q}_0 M$$

$$J_2 = K_2 = Q_0 Q_1 \bar{M} + \bar{Q}_0 \bar{Q}_1 M$$

$$J_3 = K_3 = Q_0 Q_1 Q_2 \bar{M} + \bar{Q}_0 \bar{Q}_1 \bar{Q}_2 M$$

অতঃপর সব প্রকরণ

6(a) $\rightarrow$ DLD-3

(b) $\rightarrow$ $x + x'y + \overline{x}y + \overline{x}\overline{y}$

7(a) DLD-2 $\rightarrow$ 118 page

(b) DLD-2 $\rightarrow$ 96 page

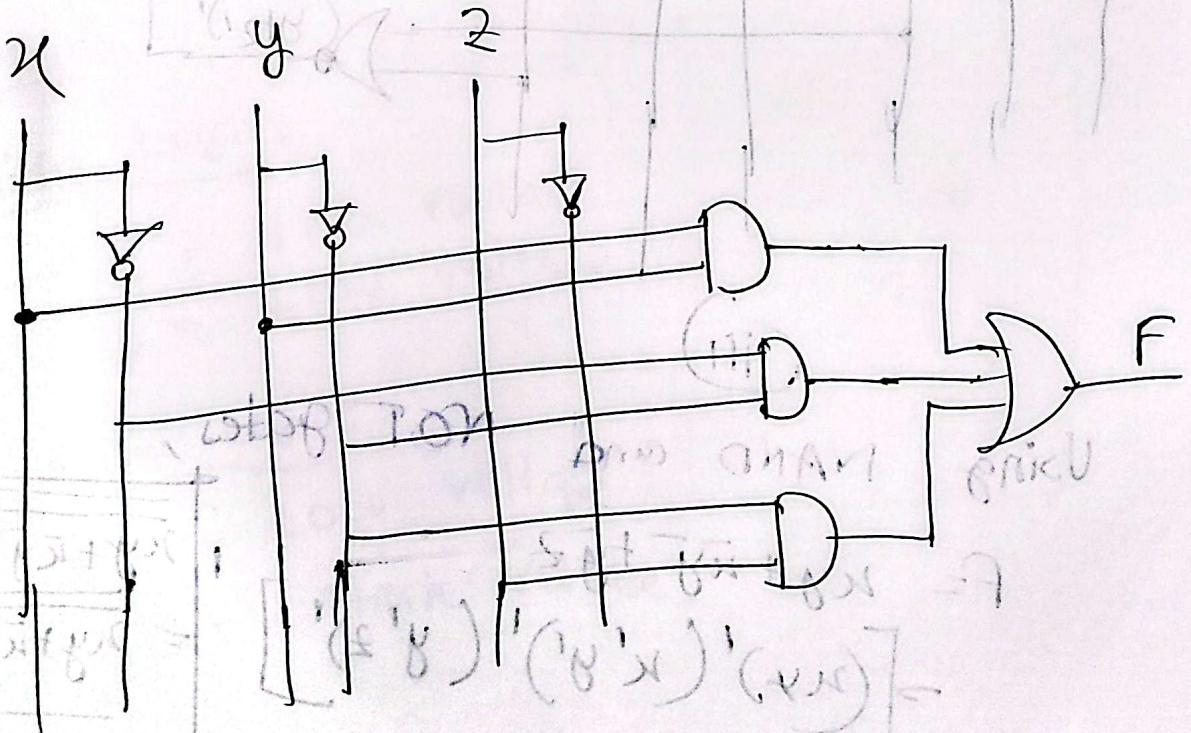
8. ~~DLD~~ DLD-2 $\rightarrow$ 122 page

Final (20-21)

1(a)

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

① $\rightarrow$ AND, OR, NOT



2

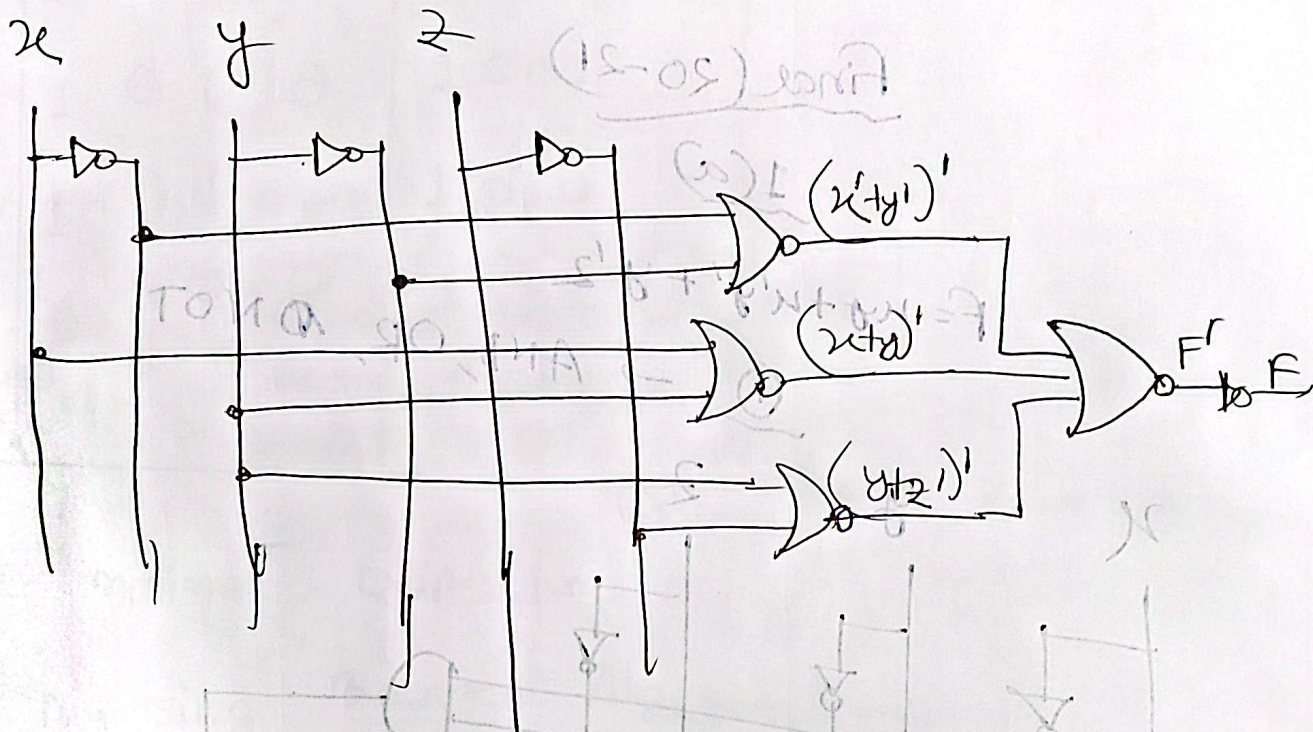
(ii)

Using NOR and NOT gates

$$F = xy + \bar{x}\bar{y} + \bar{y}z$$

$$= (\overline{\bar{x} + \bar{y}}) (x' + y')' + (x + y)' + (y + z)'$$

$$\begin{aligned} xy + \bar{x}\bar{y} + \bar{y}z &= \overline{\bar{x}\bar{y}} + \overline{\bar{x}\bar{y}} + \overline{\bar{y}z} \\ &= \overline{\bar{x} + \bar{y}} + \overline{x + y} + \overline{y + z} \end{aligned}$$



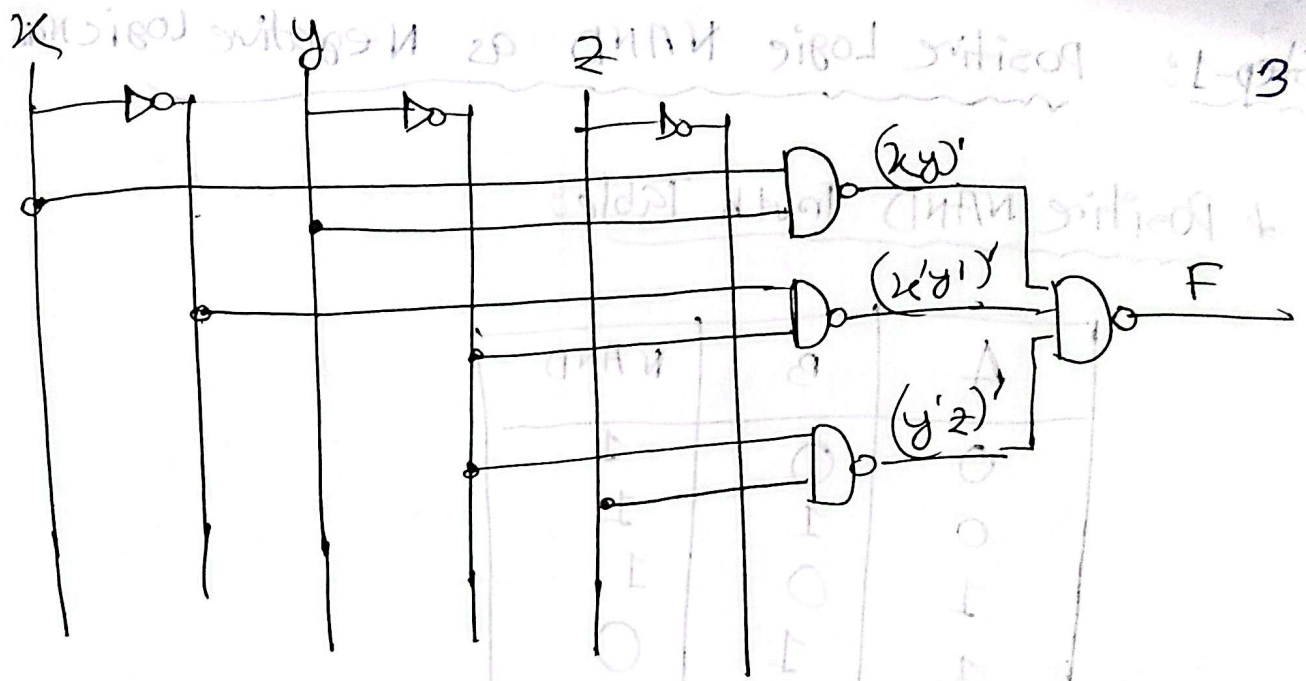
(iii)

Using NAND and NOT gates,

$$F = xy + \bar{x}\bar{y} + \bar{y}z$$

$$= \left[(xy)' (x'y')' (y'z)' \right]$$

$$\begin{aligned} \overline{\overline{xy + \bar{x}\bar{y} + \bar{y}z}} &= \overline{\overline{xy + \bar{x}\bar{y}} \cdot \bar{y}z} \\ &= \overline{\bar{x}\bar{y} + \bar{y}z} \end{aligned}$$



(b) To show that a positive logic NAND gate is a negative logic NOR gate and vice versa.

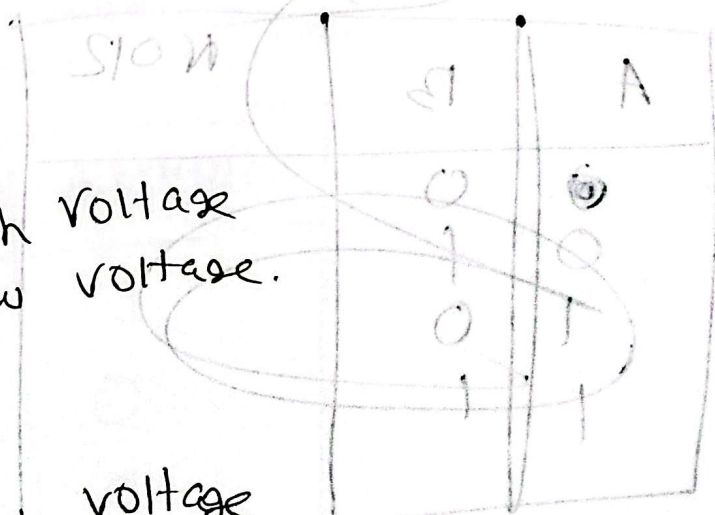
Key Concepts:

1. Positive Logic:

→ 1 = High voltage
→ 0 = Low voltage.

2. Negative Logic:

→ 1 = Low voltage
→ 0 = High voltage



A
Step-1: Positive Logic NAND as Negative Logic NOR

* Positive NAND Truth Table

A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

→ Reinterpret in ^{negative} Positive Logic

* Swap 1 $\leftrightarrow$ 0 again for inputs/output

* Negative NOR Truth Table

A	B	NOR
0	0	1
0	1	1
1	0	1
1	1	0

1. Positive Logic
 High = 1
 Low = 0

2. Negative Logic
 High = 0
 Low = 1

* Negative NOR Truth Table:

A	B	NOR
1	1	0
1	0	0
0	1	0
0	0	1

* Observations:

→ The reinterpreted table matches a

NOR gate in

negative Logic.

Step-2: Negative Logic NOR as positive Logic NAND

* Negative NOR Truth Table

A	B	$A \oplus B$ NOR
0	0	1
0	1	0
1	0	0
1	1	0

→ Reinterpret in positive logic

→ Swap $1 \leftrightarrow 0$ again.

Positive NAND Truth Table:

A	B	NAND
1	1	0
1	0	1
0	1	1
0	0	1

A	B	NOR
1	1	0
1	0	1
0	1	1
0	0	1

+ Observation:

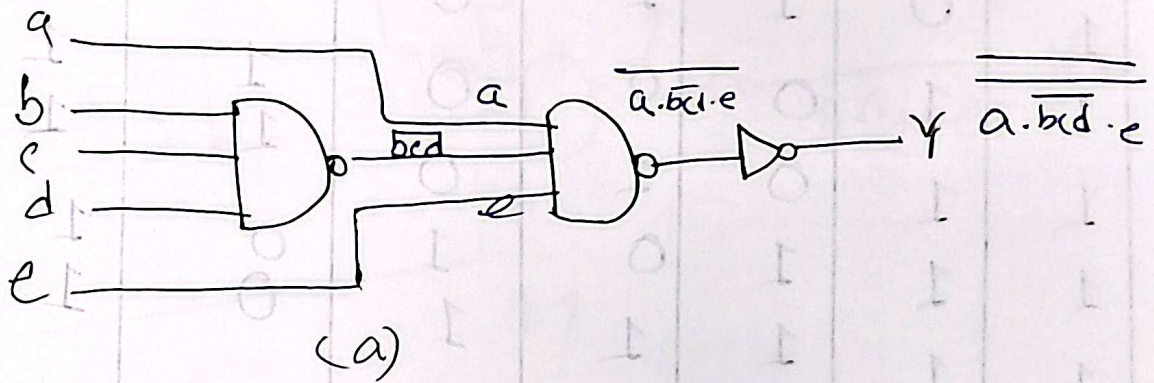
→ The reinterpreted table matches a NAND gate in positive logic.

Positive NAND $\equiv$ Negative NOR (shown).

A	B	NOR
1	1	0
1	0	1
0	1	1
0	0	1

1(c)

Construct truth table & Boolean Expression.



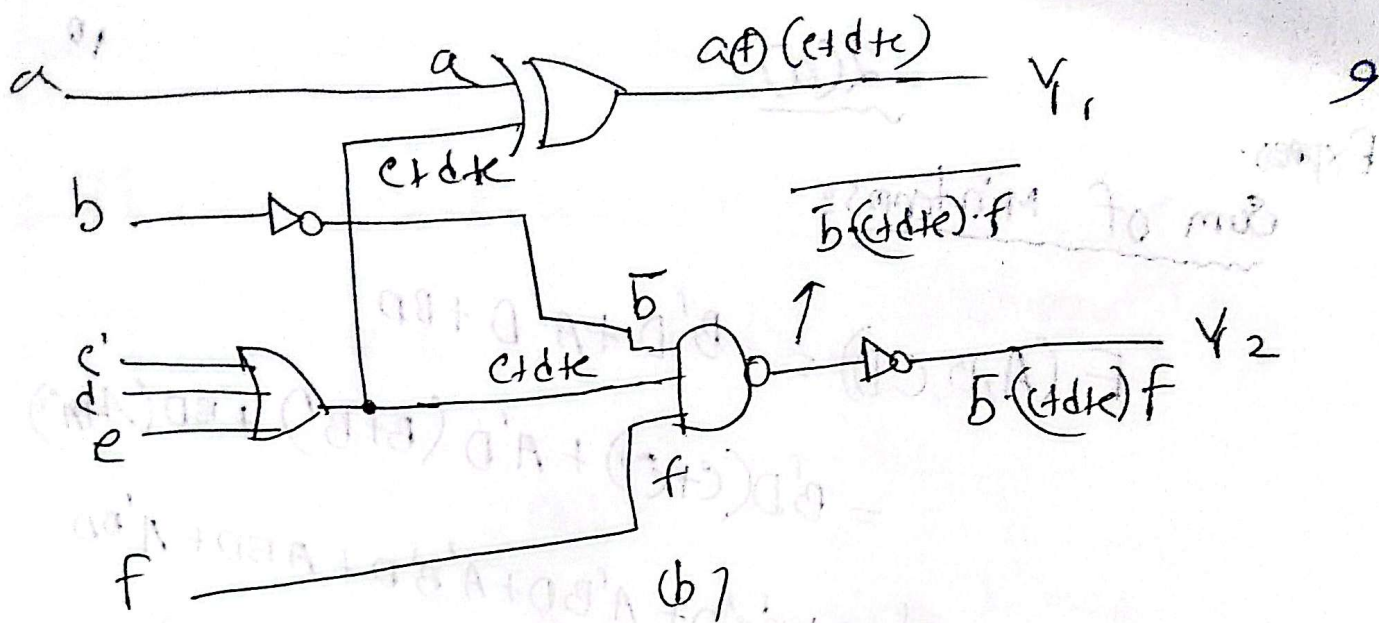
Boolean Expression:

$$Y = \overline{a.bcd.e} = a.bcd.e$$

Truth Table:

a	b	c	d	e	a.bcd.e bcd	a.bcd.e
0	0	0	0	0	0	1
0	0	0	0	1	0	1
0	0	0	1	0	0	1
0	0	0	1	1	0	1
0	0	0	0	0	0	1
0	0	1	0	1	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	0	1	1	1	0	0

[illegible]



Boolean expression

$$Y_1 = a \oplus (c+de)$$

$$Y_2 = \bar{b} \cdot (c+de) \cdot f$$

Simplify $\rightarrow Y_1 = a \cdot (c+de) + \bar{a} \cdot (c+de)$

$$= a \cdot \bar{c} \cdot \bar{d} \cdot \bar{e} + \bar{a} \cdot (c+de)$$

$\rightarrow$ table of truth

Express:

2(a)

Sum of Min terms:

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

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

$$= B'CD + B'C'D + A'BD + A'B'D + ABD + A'BD$$

$$= B'CD(A+A') + B'C'D(A+A') + A'BD(C+C')$$

$$+ A'B'D(C+C') + ABD(C+C') + A'BD(C+C')$$

$$= A'B'CD + A'B'CD + A'B'CD + A'B'CD + A'B'CD$$

$$+ A'CD + A'CD + A'CD + A'CD + A'CD$$

$$+ A'BCD + A'BCD + A'BCD + A'BCD + A'BCD \rightarrow \text{Standard form}$$

$$= \sum (11, 3, 9, 1, 7, 5, 15, 13) \rightarrow \text{Canonical form}$$

$$F(A, B, C, D) = \sum (1, 3, 5, 7, 9, 11, 13, 15)$$

Express Product of maxterm

//

$$= \prod (0, 2, 4, 6, 8, 10, 12, 14)$$

$$= \prod (0, 2, 4, 6, 8, 10, 12, 14) \rightarrow \text{minterms}$$

2(b)

Simplify boolean functions, using k-map:

$$F(w, x, y, z) = \sum (2, 3, 12, 13, 14, 15)$$

$\bar{w}x$ $\bar{y}z$ $\bar{y}\bar{z}$ $y\bar{z}$ yz
 $\bar{w}\bar{x}$ $\bar{w}x$ $w\bar{x}$ wx

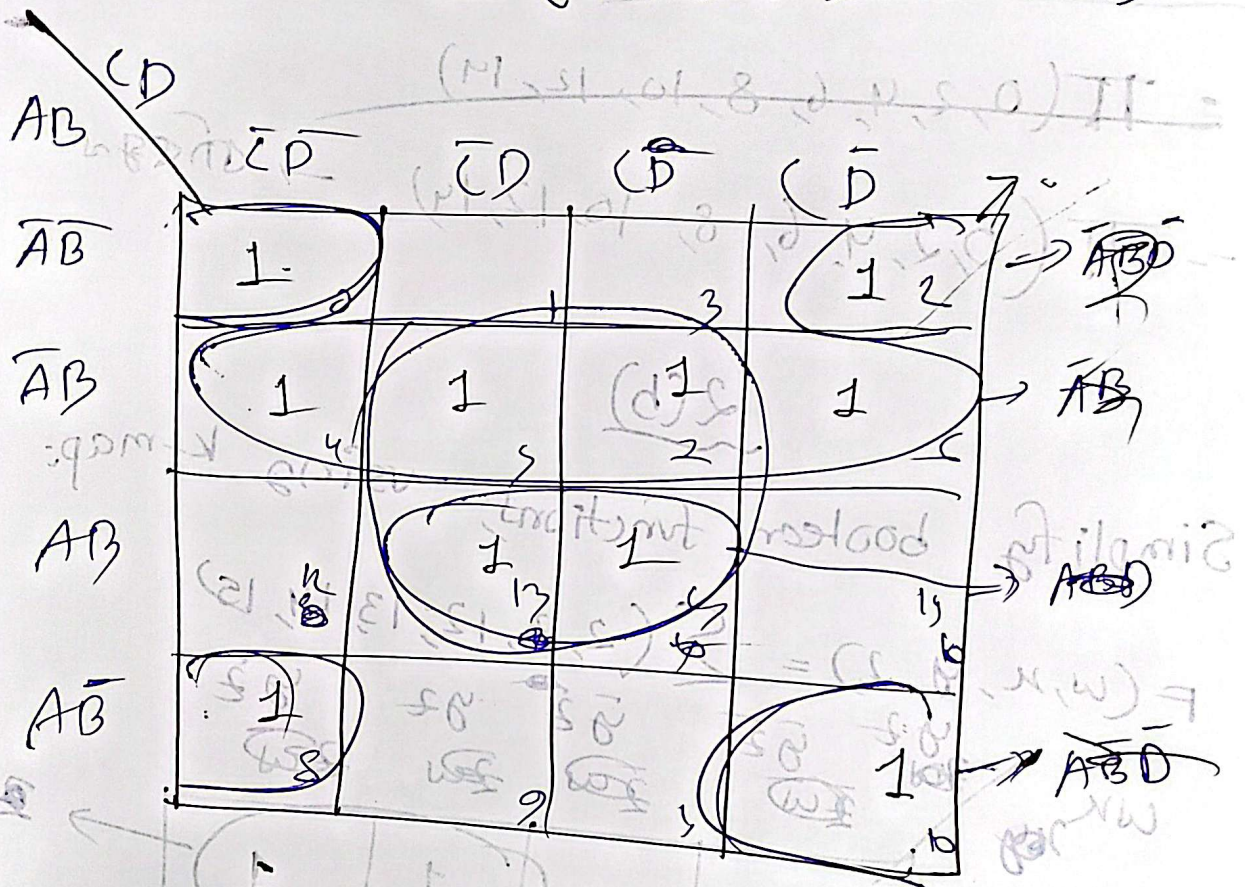
0	1	3	2
4	5	7	6
8	12	13	14
9	11	10	

$\bar{w}\bar{x}\bar{y}z$ (1) $\bar{w}\bar{x}y\bar{z}$ (1) $\rightarrow \bar{w}\bar{x}\bar{y}$
 $\bar{w}x\bar{y}z$ (1) $\bar{w}x\bar{y}\bar{z}$ (1) $\bar{w}xy\bar{z}$ (1) $\bar{w}xyz$ (1) $\rightarrow \bar{w}x\bar{y}$
 $w\bar{x}\bar{y}z$ (1) $w\bar{x}y\bar{z}$ (1) $wx\bar{y}z$ (1) $wxyz$ (1) $\rightarrow wx$

~~$F = \bar{w}\bar{x}\bar{y}$~~

$F = wx + \bar{w}\bar{x}\bar{y}$

$$F(A, B, C, D) = \sum (0, 2, 4, 5, 6, 7, 8, 10, 13, 14) \quad 12$$



$$F = \bar{A}\bar{B}\bar{D} + \bar{A}B + ABD + A\bar{B}\bar{D}$$

$$F = \bar{B}\bar{D} + \bar{A}B + BD$$

2(c)

13

$$F(A, B, C, D) = \sum (2, 4, 6, 10, 12)$$

$$d(A, B, C, D) = \sum (0, 8, 9, 13)$$

Step-1: K-map

	CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	$\bar{X}_0$	1	3	1	2
$\bar{A}B$	1	5	7	1	6
AB	1	$\bar{X}_{13}$	14		14
$A\bar{B}$	$\bar{X}_8$	$\bar{X}_9$	11	1	10

Final (19-20)

Ans 1(a)

Duality Principle: The duality principle states that:

Every algebraic expressions in Boolean Algebra remains valid if we interchange:

1. "AND(.) with OR(+)"

2. "0 with 1"

Key Implication of Duality:

1. Dual of an Expression:

→ Replace all AND with OR, all OR with AND, all 0 with 1, all 1 with 0.

* Example:

→ Original: $A + (B \cdot C) = 1$

→ Dual: $A \cdot (B + C) = 0$

2. Dual of Theorems

→ Original (Idempotent Law): $A + A = A$

→ Dual: $A \cdot A = A$

→ Original (De Morgan's Law): $\overline{A+B} = \bar{A} \cdot \bar{B}$

→ Dual: $\overline{A \cdot B} = \bar{A} + \bar{B}$

3. Dual of Gates:

(02-01) Janil

15

→ $NAND \leftrightarrow NOR$: A positive logic NAND is a negative logic NOR (vice-versa)

→ $AND \leftrightarrow OR$: Their truth tables are duals.

Example: Duality in Action

Original : $F = A \cdot (B + \bar{C}) + 0$

Dual : $f_{dual} = A + (B \cdot \bar{C}) \cdot 1$

Verification:

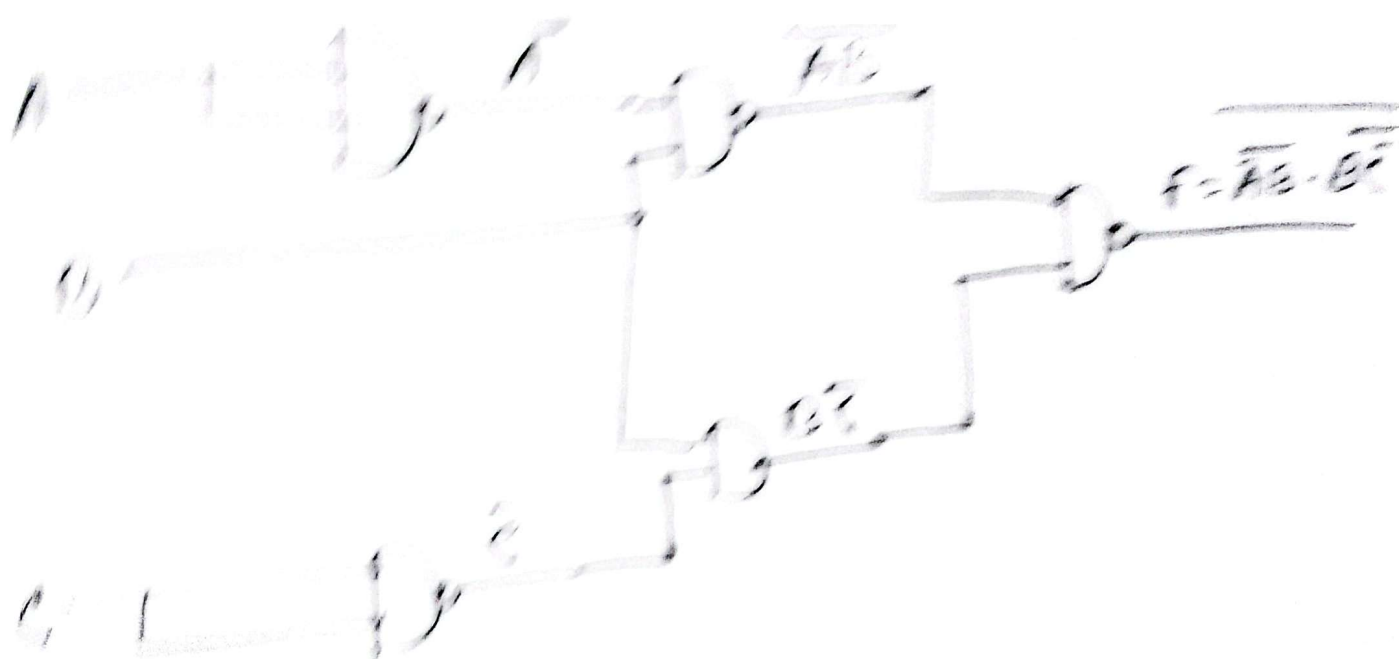
* Original : If $A=1, B=0, C=1, F=1 \cdot (0+0) + 0 = 0$

* Dual : $f_{dual} = 1 + (0 \cdot 0) \cdot 1 = 1$

1(0)

Implement function using NAND gate

$$F = A'B + BC'$$



1(c)

Simplifying the Boolean Expression function F
using don't care condition:

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

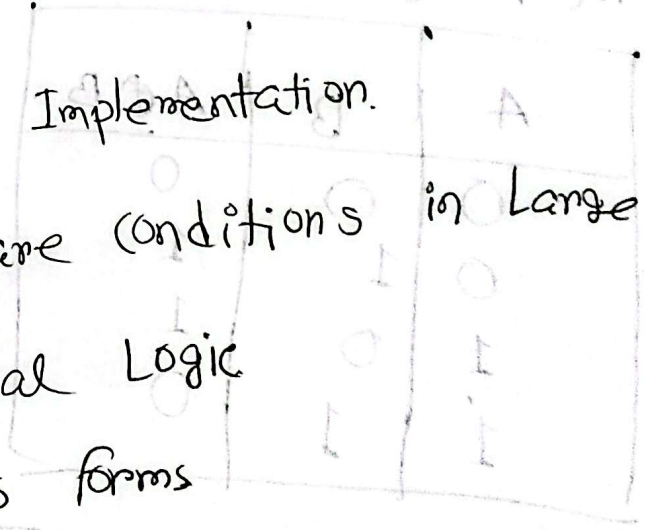
1(0)

18

Karnaugh Maps are a powerful tool for simplifying Boolean functions, but they have several limitations:

Limitation of K-maps

1. Not suitable for large variable Functions like as more than 5.
2. Error-Prone Manual use.
3. NO Systematic Procedure.
4. Not ideal for computer Implementation.
5. Difficult with don't care conditions in Large Maps.
6. Limited to Combinational Logic.
7. Limited to SOP/POS forms.
8. NO guarantee of minimality.
9. Static Representation.
10. Manual complexity.



2(a)

19

Show that the dual of the exclusive-OR is also its complement.

⇒ To show that the dual of the exclusive-OR is also its complement, we'll follow these steps:

Step-1: Definition of XOR and its complement:

* XOR ($A \oplus B$):

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

Truth Table:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

* Complement of XOR ($\overline{A \oplus B}$):

$$\overline{A \oplus B} : AB + \bar{A}\bar{B}$$

This is the XNOR operation.

Truth Table:

20

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Step 2: Dual of XOR

The dual of a Boolean expression is obtained by:

- Swapping AND(.) and OR(+) operators
- Swapping 0 and 1.

* Original XOR : $A \oplus B = \bar{A}B + A\bar{B}$

* Dual of XOR : $\text{Dual}(A \oplus B) = (\bar{A} + B) \cdot (A + \bar{B})$

Step 3: Simplify the Dual Expressions

Let's simplify $(\bar{A} + B) \cdot (A + \bar{B})$:

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

Using Boolean algebra:

$$\bar{A}A = 0$$

$$B\bar{B} = 0$$

So :

$$0 + \bar{A}\bar{B} + AB + 0 = AB + \bar{A}\bar{B}$$

This is exactly the complement of XOR (XNOR):

Step 4: Verification via Truth Table:

A	B	$\bar{A} + B$	$A + \bar{B}$	$(\bar{A} + B) \cdot (A + \bar{B})$	$\overline{A \oplus B}$
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	1	1	1	1

Conclusion:

We've shown that:

$$\begin{aligned} \text{Dual}(A \oplus B) &= \overline{A + B} \cdot \overline{A \oplus B} = \overline{A + B} \cdot (A \oplus B) \\ &= \overline{A \oplus B} \end{aligned}$$

Thus, the dual of XOR is its complement (XNOR).

2(b)

22

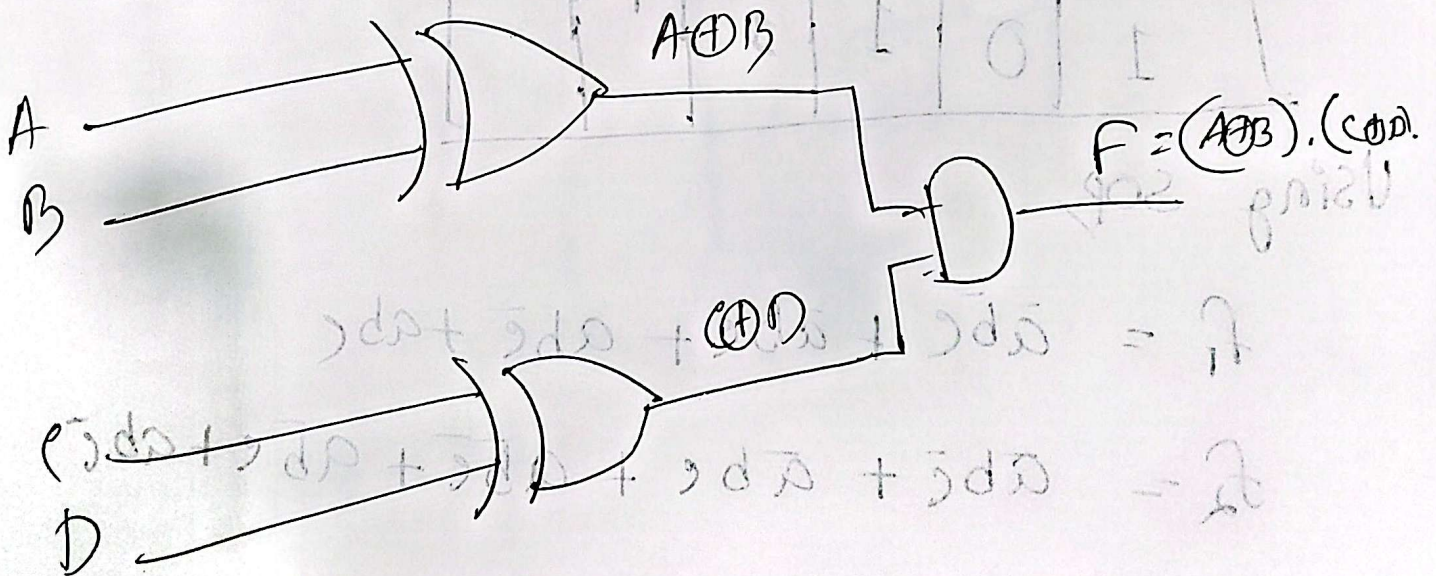
Implement Boolean function with XOR and AND gates

$$F = A\bar{B}c\bar{D} + \bar{A}Bc\bar{D} + A\bar{B}\bar{c}D + \bar{A}B\bar{c}D$$

$$= A\bar{B}c\bar{D}(\bar{A} + A) + \bar{A}Bc\bar{D}(\bar{A} + A) + A\bar{B}\bar{c}D(\bar{A} + A) + \bar{A}B\bar{c}D(\bar{A} + A)$$

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

$$F = (A \oplus B) \cdot (c \oplus D)$$



23

2(c)

(d)

Boolean Expression & draw logic circuit from truth table:

f_1	f_2	a	b	c
1	0	0	0	0
0	0	0	0	1
0	1	0	1	0
1	1	0	1	1
0	1	1	0	0
0	1	1	0	1
1	1	1	1	1
1	0	1	1	0

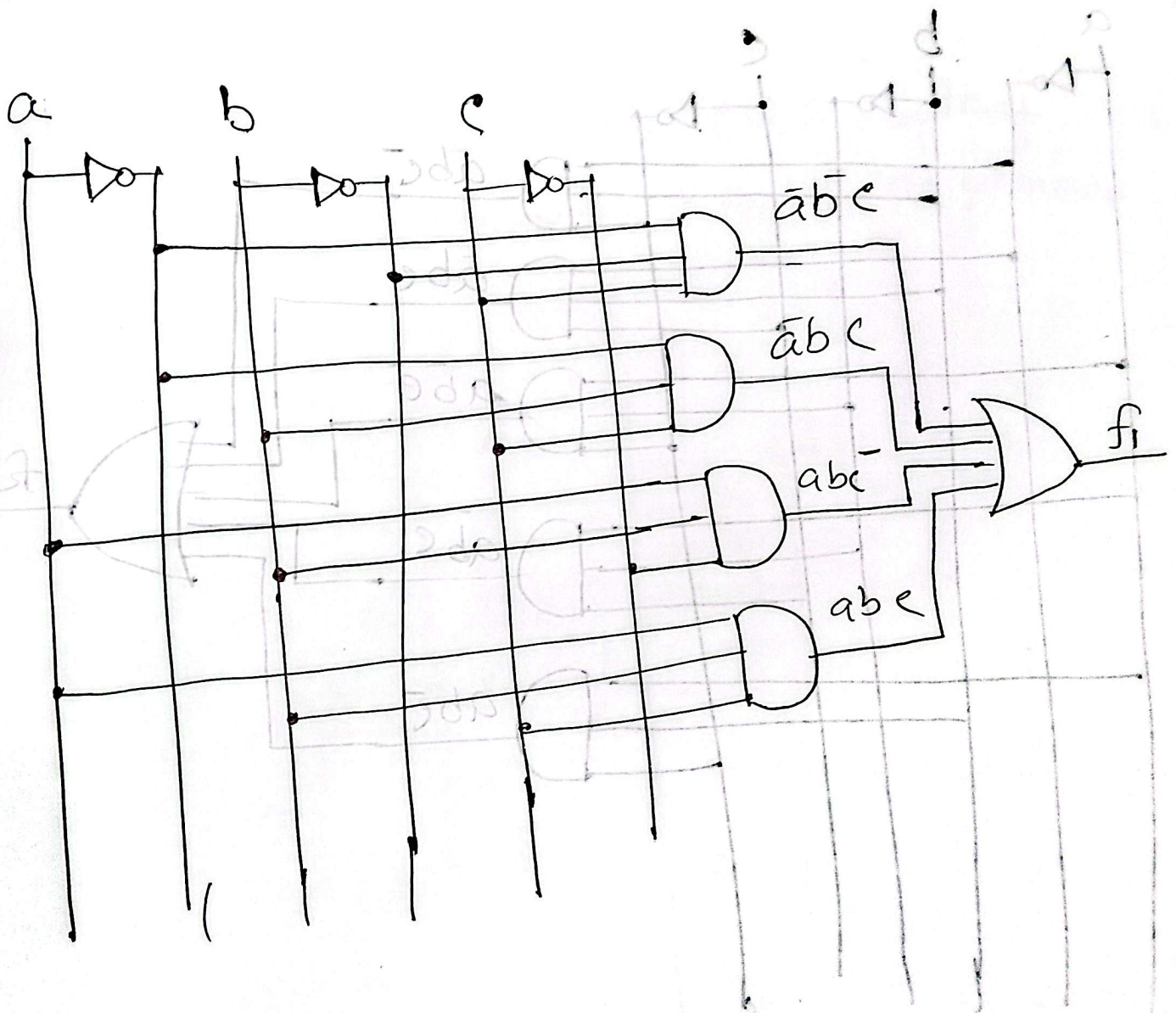
Using SOP

$$f_1 = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$

$$f_2 = a\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}c + a\bar{b}c + abc$$

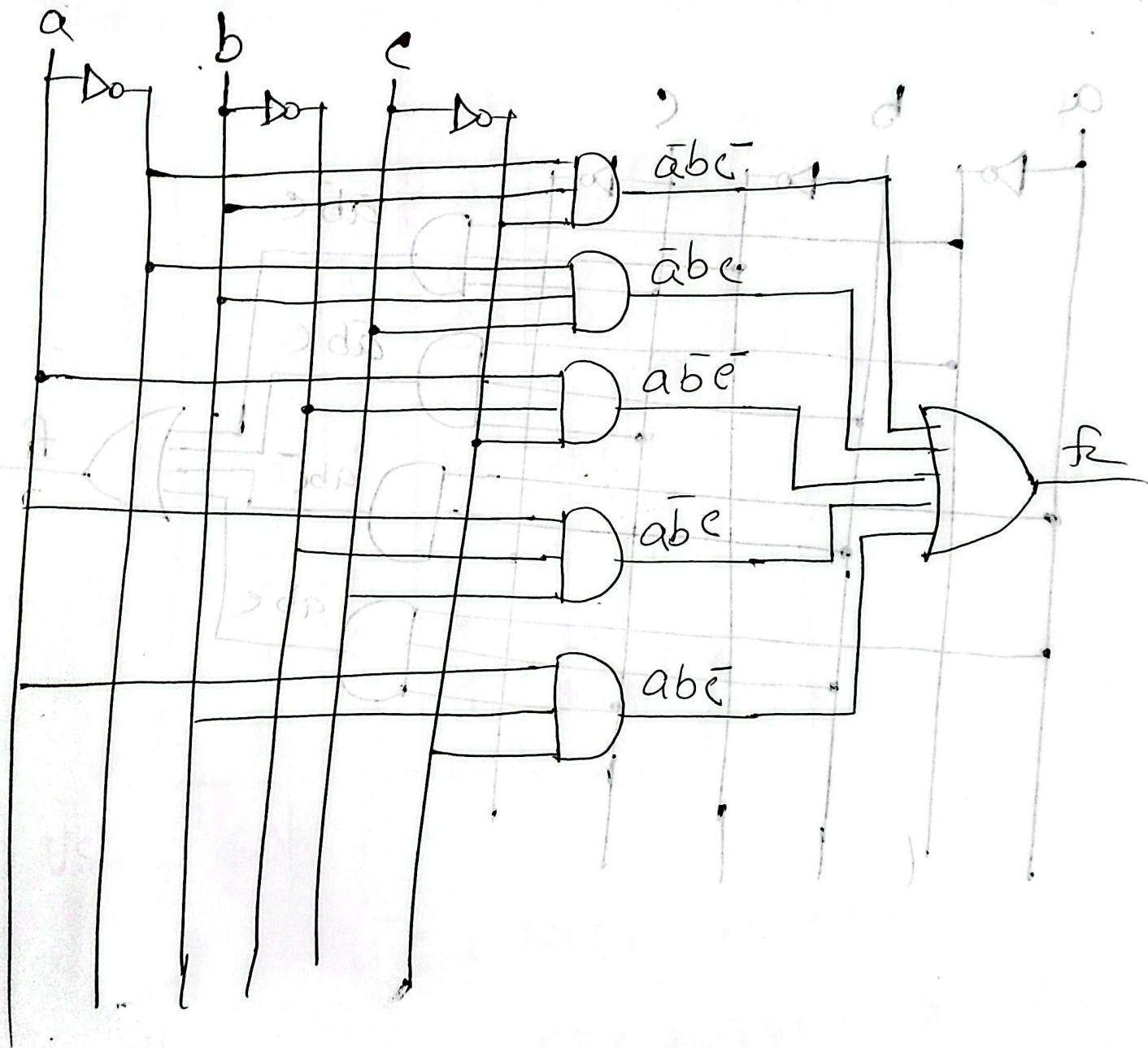
Logic Diagram.

$$f_1 = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$



$$f_2 = \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + abc$$

25



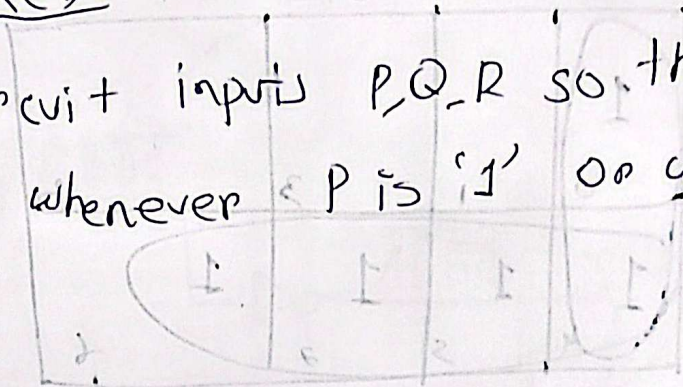
Final (18-19)

26

2(a)

Design a logic circuit inputs P, Q, R so that output S is high whenever P is '1' or whenever

$Q = R = 0$.



P	Q	R	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

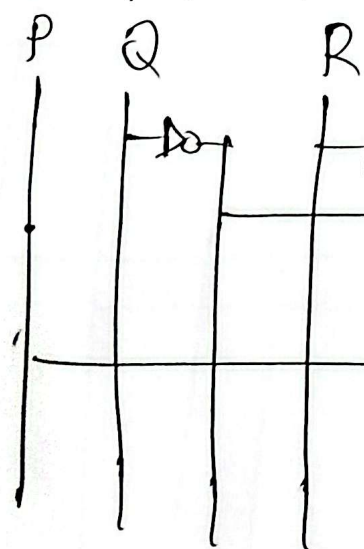
$$S = \bar{P}\bar{Q}\bar{R} + P\bar{Q}\bar{R} + P\bar{Q}R + P\bar{Q}\bar{R} + PQR$$

Simplify using k-map: (01-21) 2017

27

	$\bar{P}\bar{Q}\bar{R}$	$\bar{P}Q\bar{R}$	$\bar{P}QR$	$P\bar{Q}\bar{R}$	$PQ\bar{R}$	PQR
$\bar{P}$	1	0	0	0	0	0
P	1	1	1	1	1	1

$$S = P + \bar{Q}\bar{R}$$



	$\bar{P}\bar{Q}\bar{R}$	$\bar{P}Q\bar{R}$	$\bar{P}QR$	$P\bar{Q}\bar{R}$	$PQ\bar{R}$	PQR
1	1	0	0	0	0	0
2	0	1	0	0	1	0
3	0	0	1	0	0	1
4	1	1	1	1	1	1
5	1	0	0	1	0	0
6	1	1	0	1	1	0
7	1	0	1	1	0	1
8	1	1	1	1	1	1

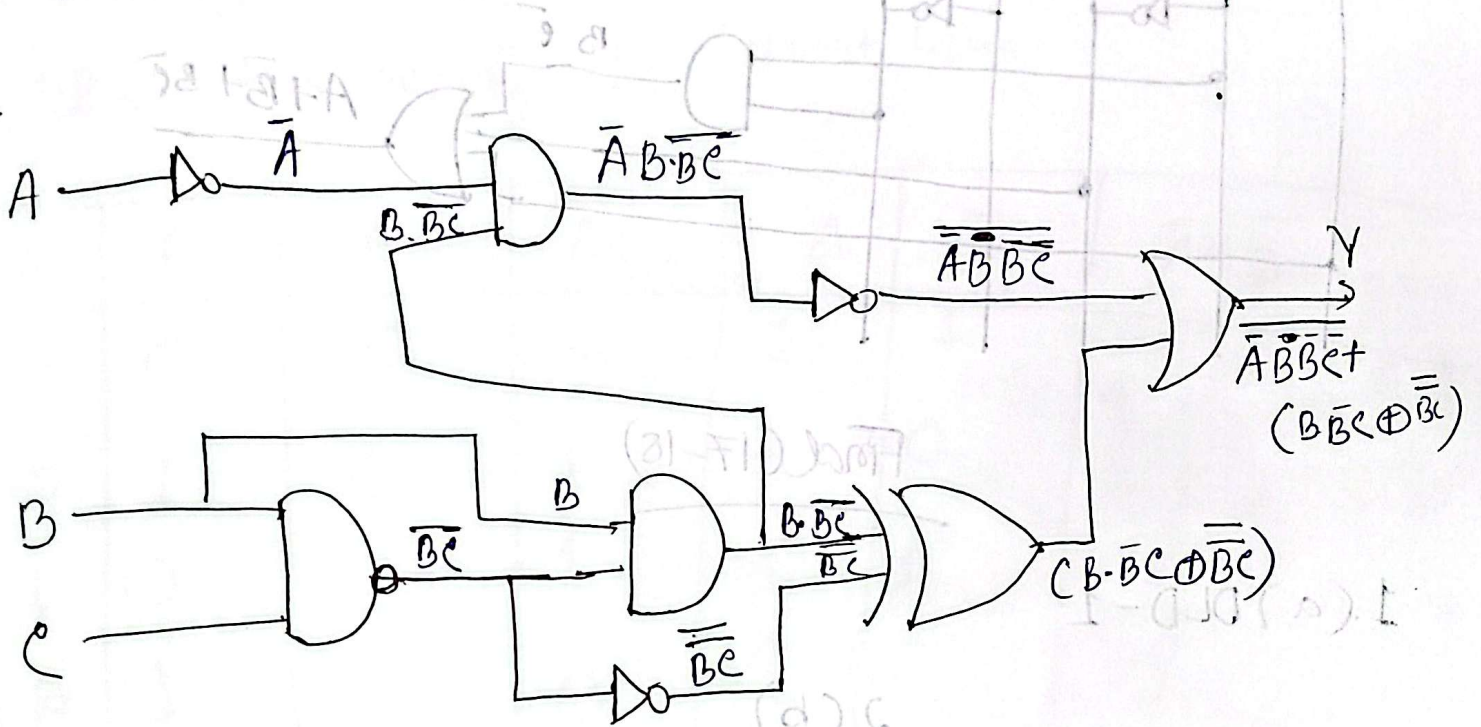
$$S = P + \bar{Q}\bar{R}$$

$$S = P + \bar{Q}\bar{R} = P + \bar{Q}\bar{R} + P\bar{Q}\bar{R} + PQR + P\bar{Q}R + PQR + PQR = P + \bar{Q}\bar{R}$$

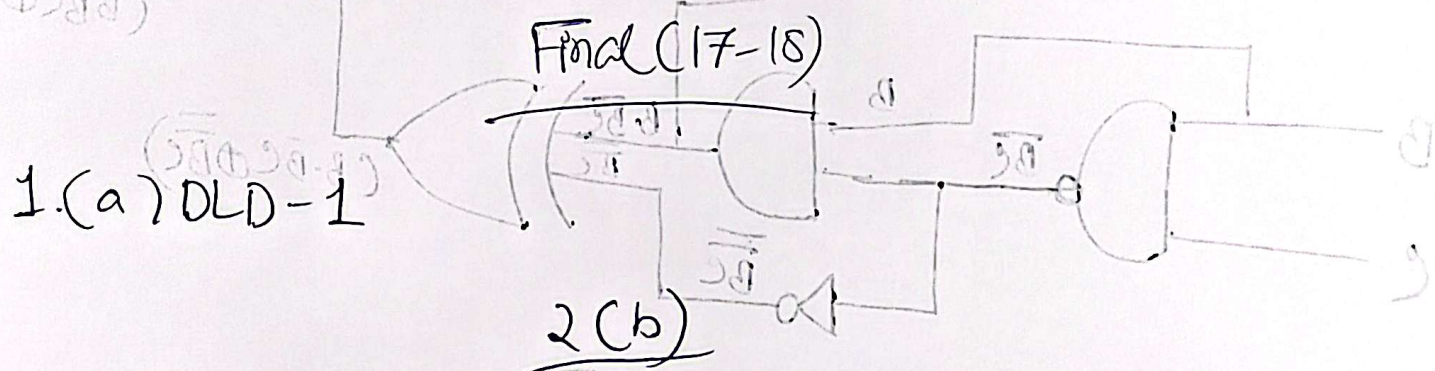
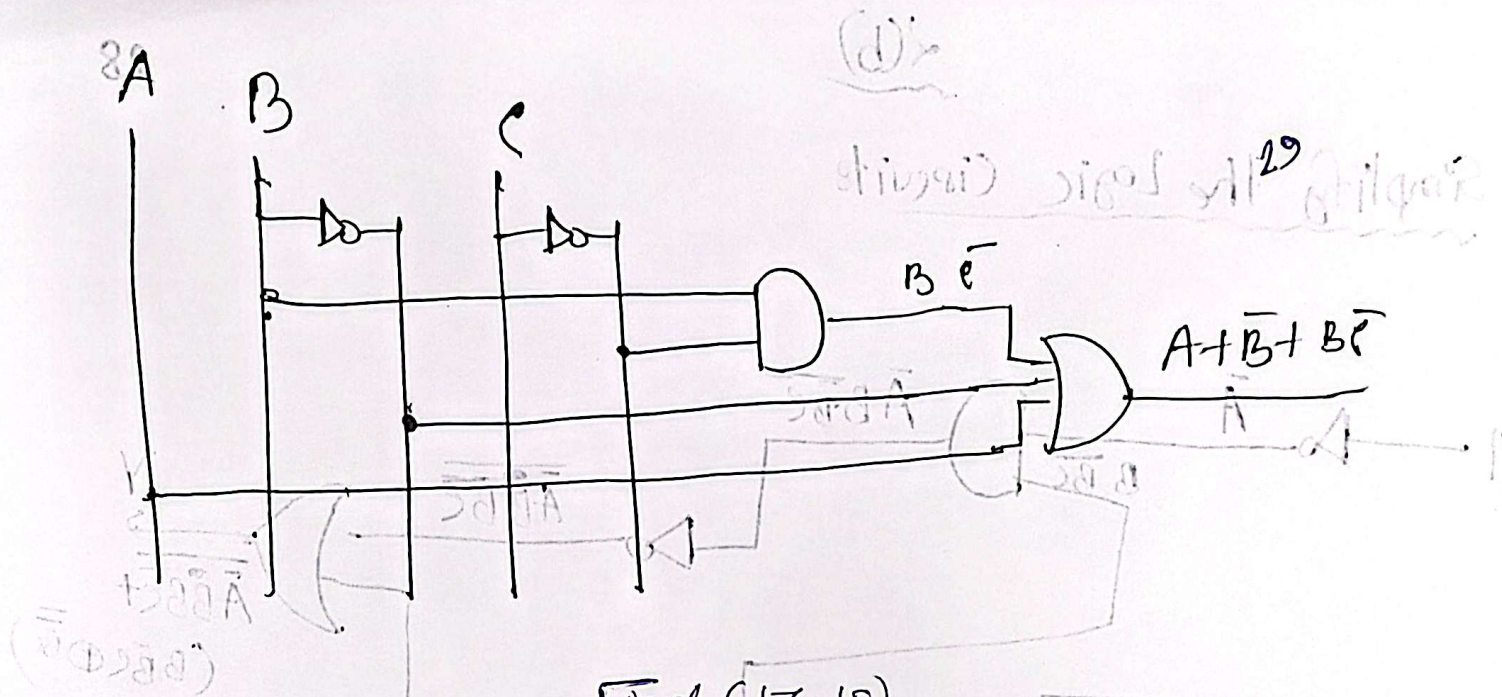
2(b)

28

Simplify the Logic Circuit



$$\begin{aligned}
 Y &= \bar{A}\bar{B}\bar{C} + (B\bar{B}\bar{C} + \bar{B}C) \\
 &= \bar{A} + \bar{B} + \bar{B}\bar{C} + B(\bar{B} + C) \\
 &= \bar{A} + \bar{B} + \bar{B}\bar{C} + B\bar{B} + BC \\
 &= \bar{A} + \bar{B} + \bar{B}\bar{C} + 0 + BC \\
 &= \bar{A} + \bar{B} + \bar{B}\bar{C} + BC \\
 &= \bar{A} + \bar{B} + \bar{B}\bar{C} + BC
 \end{aligned}$$



De-Morgan's Theorem:

De-Morgan's theorem states:

① First Law:

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

The complement of a product is equal to the sum of the complements.

② Second Law:

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

The complement of a sum is equal to the product

of the complements 30

Proof Using Truth Tables:

Step-1: we verify the first Law.

A	B	$A \cdot B$	$(\overline{A \cdot B})$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Since $(\overline{A \cdot B}) = \overline{A} + \overline{B}$, the first Law is proven true.
 For the second Law:

A	B	$A + B$	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Since $(\overline{A+B}) = \overline{A} \cdot \overline{B}$, the second Law is proven true.

2(c)

What are the different methods to write a logical Expression from truth-table? Explain with an example.

⇒ There are three primary methods to derive a logical expression from a truth table:

1. Sum of Products (SOP)
2. Product of Sums (POS)
3. Karnaugh Map (K-map) Simplification

Let's explore each method with an example.

① Sum of Products (SOP):

- Identify rows where the output is 1.
- Write AND terms for each row using all input variables.

→ Combine these terms using OR operation.

Example: Given the truth table for $F(A, B)$:

A	B	$F(A, B)$
0	0	0
0	1	1
1	0	1
1	1	0

Minterms with $F=1 \rightarrow (\bar{A}B + A\bar{B})$

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

(This is the XOR function).

② Product of Sums (POS):

→ Identify the rows where the output is 0.

→ Write OR terms for each row using negated input variables.

→ Combine these terms using AND operation.

Example: Same table

A	B	F(A,B)
0	0	0
0	1	1
1	0	1
1	1	0

Rows with $F=0 \rightarrow (A+B)(\bar{A}+\bar{B})$

$$\therefore \text{POS} = (A+B)(\bar{A}+\bar{B})$$

③ Karnaugh-map (K-map):

- Place 1s and 0s from the truth table into a K-map
- Group adjacent 1s (for SOP) or 0s (for POS)
- Extract simplified expressions

2(d)

34

Explain K-Map method to simplify a logic circuit.

⇒ A Karnaugh Map (K-map) is a visual way to simplify Boolean expressions by minimizing logic gates used in a circuit. It helps eliminate redundant variables, making circuits more efficient and faster.

Step-by-Step K-Map Simplification

Step-1: Draw the K-map Grids

→ the number of cells in the K-map $= 2^n$ (where n is the number of variables).

→ for 2 variables $\rightarrow$ 4 cells.

→ for 3 variables $\rightarrow$ 8 cells.

→ for 4 variables $\rightarrow$ 16 cells.

Each cell represents a minterm corresponding to input combinations.

②: Fill in the Minterms:

- Identify where the function is True (1) in the truth table.
 - All those positions with 1s in the K-map
- For example, if given $F(A, B) = \sum (1, 2)$, we place 1s in positions 1 and 2

③ Group the 1s:

- Combine adjacent 1s into groups of powers of 2 (1, 2, 4, 8, ...).
- Bigger groups eliminate variables, simplifying the expressions.
- Groups can wrap around the edges.

For example, if we have four 1s in a square shape, we simplify the term by reducing variables.

④ Extract the Simplified Expression:

36

→ Identify common variables in each group.

→ write the reduced Boolean expression.

Example: If grouped is eliminate B , then $F(A,B)$ simplifies to just A .

Example of Kmap simplification:

Given the truth table for $F(A,B)$.

A	B	$F(A,B)$
0	0	0
0	1	1
1	0	1
1	1	0

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

K-Map Representation:

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

Simplify : $F(A,B) = \bar{A}B + A\bar{B} = A \oplus B$

□ Explain the operation of an Asynchronous up/down MOD-8 counter with an appropriate diagram

⇒ ~~How it~~ An Asynchronous up/down MOD-8 counter is a 3-bit binary counter that counts ~~from 0 to 7~~ or either up (0 to 7) or down (7 to 0) based on a control signal.

Here 3 bit up counter: 000-111

3 bit down counter: 111-000

Let, if $M=0$, then up counting occur.

if $M=1$, then down counting occurs.

Truth Tables How it Works:

1. Up Counting Mode ($M=0$)

→ The external clock triggers FF0 → toggles on every clock edge.

→ FF1 is triggered by Q_0 → toggles when Q_0 falls.

→ FF2 is triggered by $Q_1 \rightarrow$ toggles when Q_1 falls.
 counting sequence: 0-1-2-3-4-5-6-7-0
 (repeats)

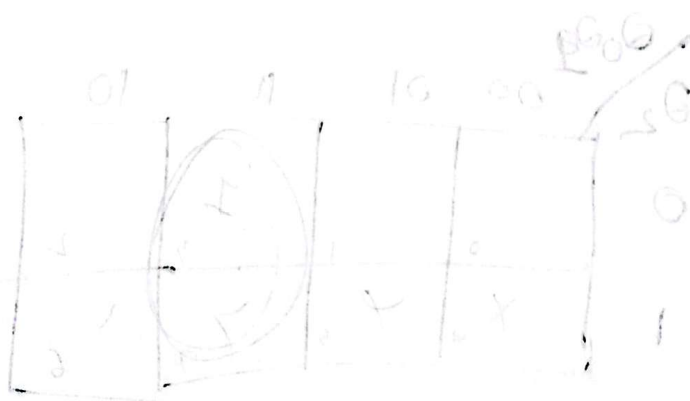
2. Down Counting Mode (M=1):

→ The external clock still triggers FF0
 → FF1 is now triggered by $\bar{Q}_0 \rightarrow$ toggles when Q_0 rises.
 → FF2 is now triggered by $\bar{Q}_1 \rightarrow$ toggles when Q_1 rises.

Counting Sequence: 7-6-5-4-3-2-1-0.

Truth table

Logic Diagram



□ Explain the operation of MOD-8 Synchronous Up Counter.

⇒ A mod-8 counter counts from 0 to 7. Synchronous means all flip-flops are triggered at the same time by a common clock signal. After reaching ~~sa~~ 7, it resets back to 0.

→ Step-1: Determine the number of flip-flops.

⇒ Number of flipflops needed $(n) = 3$

∴ Number of states $(N) = 2^n = 2^3 = 8$

∴ Maximum count = $N-1 = 8-1 = 7$

Step-2: JK flip-flop used to design mod-8 Synchronous up counter.

Step-3: State diagram.

Step 4: Working principle:

FF Configuration:

- 3 flip flops (A, B, C) represent the 3-bit binary count (Q_2, Q_1, Q_0)
- All the flip-flops receive the same ~~signal~~ clock signal.

State Transition Logic:

- For a binary up counter, each flip-flop toggles based on the previous state.

* LSB (Q_0) → Toggles on every clock pulse

* Q_1 : Toggles only when $Q_0 = 1$

* Q_2 : Toggles only when ^{both} Q_0 & $Q_1 = 1$

The logic is implemented using AND gate

to control the flip-flop inputs.

Implementation using J K FF.

→ If $J = K = 0$, the output remain unchanged

→ If $J = K = 1$, the output toggles.

* LSB : $J_0 = K_0 = 1$

* Q_1 : $J_1 = K_1 = Q_0$

* Q_2 : $J_2 = K_2 = Q_0 Q_1$

Truth Table.

Circuit.

Down 2 (2P)

1. FF Control Logic:

* LSB (Q_0) → Toggles on every clock pulse

* Q_1 : Toggles only when $Q_0 = 0$

* Q_2 : Toggles only when both Q_0 & $Q_1 = 0$

2. Implementation of JK FF.

$$\text{For } Q_0 : J_0 = K_0 = 1$$

$$\text{For } Q_1 : J_1 = K_1 = Q_0$$

$$\text{For } Q_2 : J_2 = K_2 = Q_0 \cdot Q_1$$

4. Synchronous up/down counter structures

কিছু বৈশিষ্ট্য:

Key features

* MOD 16 $\rightarrow$ 16 states (0000 to 1111 in binary,
0 to 15 in decimal)

* 4 flip-flops (JK) with a common clock.

* Control input (Up/down):

$\rightarrow M=0$, up counting (Increments count 0000 $\rightarrow$
1111 - 0000)

$\rightarrow M=1$, down counting (Decrements count
1111 - 0000 - 1111)

* Synchronous Operation: All the flip-flops change
at the same time. (no ripple delay)

Working Principles & Design

1. FF Configuration:

- 4 FFs ($Q_3 Q_2 Q_1 Q_0$) represent the 4-bit binary count.
- All FFs share the same clock signal.
- A control line (Up/down) determines counting direction.

$M = (Up) \Rightarrow 0 \rightarrow \text{increment}$

$M = (\text{down}) \Rightarrow 1 \rightarrow \text{Decrement}$

2. Toggles Conditions:

FF	Up Mode ($M=0$)	Down Mode ($M=1$)
Q_0 (LSB)	Always toggles	Always toggles
Q_1	Toggles when $Q_0 = 1$	Toggles when $Q_0 = 0$
Q_2	Toggles when $Q_0 \& Q_1 = 1$	Toggles when $Q_0 = 0$ and $Q_1 = 1$
Q_3 (MSB)	Toggles when $Q_0 = Q_1 = Q_2 = 1$	Toggles when $Q_0 = Q_1 = Q_2 = 0$

3. Logic Implementation (Using JK):

- If $J=k=0$, output remains unchanged
- If $J=k=1$, output toggles.

Input Logic (Combined Up/down Control).

* $Q_0 = J_0 = K_0 = 1$ (always toggles)

* Q_1 :

→ ~~up mode: $J_1 = K_1 = 1$~~

→ Up mode ($\bar{M}=0$): $J_1 = K_1 = Q_0$

→ Down Mode ($M=1$): $J_1 = K_1 = \bar{Q}_0$

→ Combined: $J_1 = K_1 = \bar{M}Q_0 + M\bar{Q}_0$

* Q_2 :

→ Up mode ($\bar{M}=0$): $J_2 = K_2 = Q_0 \cdot Q_1$

→ Down mode ($M=1$): $J_2 = K_2 = \bar{Q}_0 \bar{Q}_1$

→ Combined: $J_2 = K_2 = (\bar{M}Q_0Q_1) + (M\bar{Q}_0\bar{Q}_1)$

* Q_3 :

→ Up mode ($\bar{M}=0$): $J_3 = K_3 = Q_0Q_1Q_2$

→ Down mode ($M=1$): $J_3 = K_3 = \bar{Q}_0\bar{Q}_1\bar{Q}_2$

→ Combined: $J_3 = K_3 = (\bar{M}Q_0Q_1Q_2) + (M\bar{Q}_0\bar{Q}_1\bar{Q}_2)$