Faculty of Engineering and Technology

Electrical and Computer Engineering Department

DIGITAL ELECTRONICS AND COMPUTER ORGANIZATION LABORATORY

ENCS2110

Experiment No. 3

Encoders, Decoders, Multiplexers & Demultiplexers

Prepared By: **Ghaith Haj-Ali** → **1220612**

Partners: **1. Abd Al-Rahman Sia'rah** → **1230219**

2. Zaid Qamhieh → **1230069**

Instructor: **Nasser Ismail** T.A: **Raneem Al-Qaddi**

Section: **3** Date: **05-Sep.2025**

Abstract:

In this experiment, the objective was to explore the basic principles and functions of four essential combinational logic components: the encoder, decoder, multiplexer, and demultiplexer. These devices played a vital role in digital circuit design, as they were applied in data selection, code conversion, and signal routing. The circuits were constructed using both basic logic gates and integrated circuits, which provided practical understanding of their theoretical operation. Through hands-on implementation, the experiment reinforced knowledge of how these components functioned individually and how they could be applied together in larger digital systems. Furthermore, it developed skills in analyzing truth tables, converting Boolean expressions into summation form, and applying multiplexers and demultiplexers to realize logical functions. By assembling and testing these circuits in the laboratory, the theoretical concepts were validated, and practical troubleshooting techniques were improved. Overall, the experiment strengthened comprehension of combinational logic and highlighted the importance of these building blocks in achieving reliable and efficient digital circuit design.

Table of Contents

List Of Tables

Table of Figures:

Theory

1. Decoder

The decoder is a type of combinational circuit that receives an N-bit binary input and generates multiple output lines, with only one output being high (active) for each unique combination of inputs.

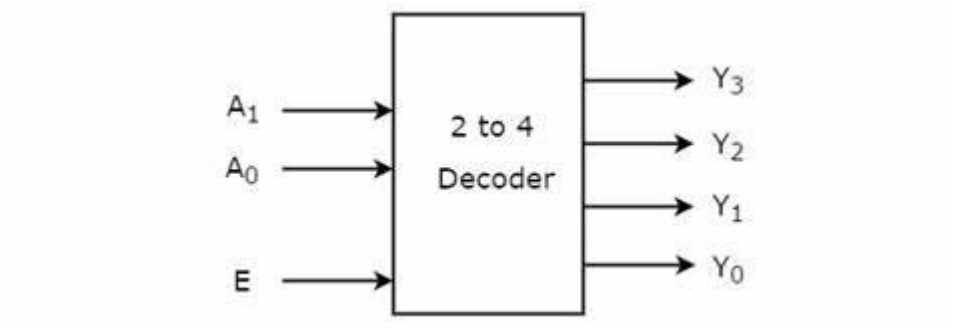

Figure 1: N to 2^n Decoder

A decoder can be implemented using only NOT (inverter) and NAND gates. Inputs are inverted as needed, and each output is generated by a NAND gate combining the original and complemented inputs. For example, in a 2-to-4 decoder, the two inputs are inverted, and four NAND gates produce the outputs, each active for a specific input combination. This method efficiently converts binary inputs into distinct output lines using only basic gates.

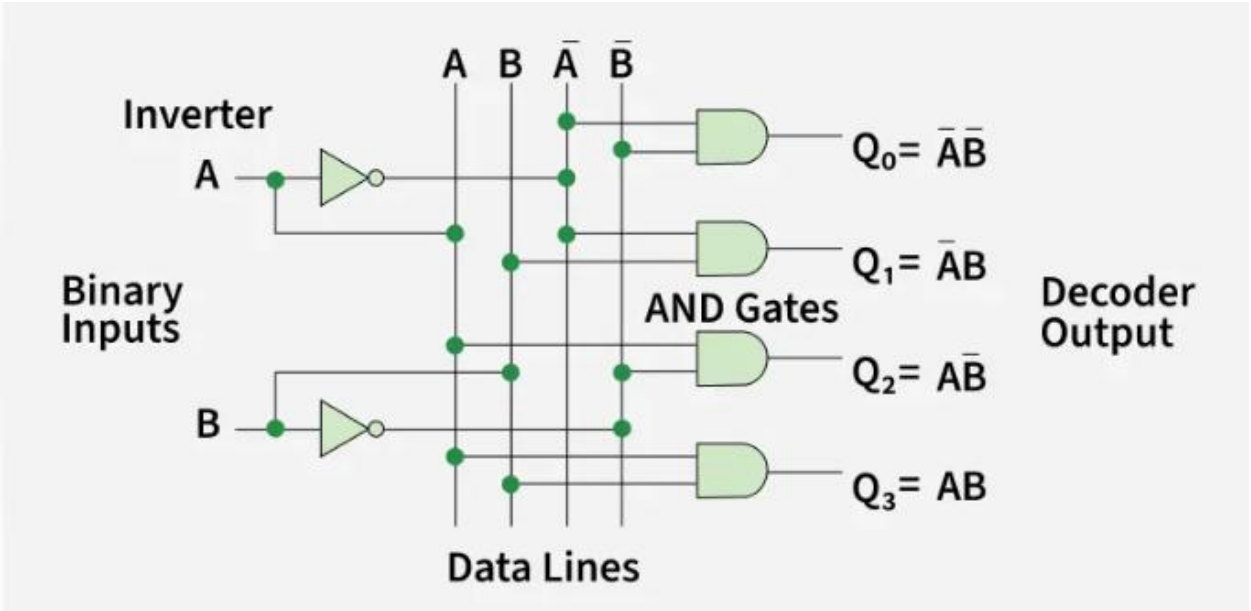

Figure 2: 2 to 4 Decoder Using Basic Gates

Inputs			Outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Table 1: 2 to 4 Decoder Truth Table

2. Encoder

The encoder is a combinational digital circuit that performs the opposite operation of a decoder. It usually has up to 2^n input lines and generates n output lines. The output lines collectively form the binary code corresponding to the active input. In this experiment, the encoder was implemented practically using only two OR gates, showing that basic gates can be used to realize its logic functions efficiently.

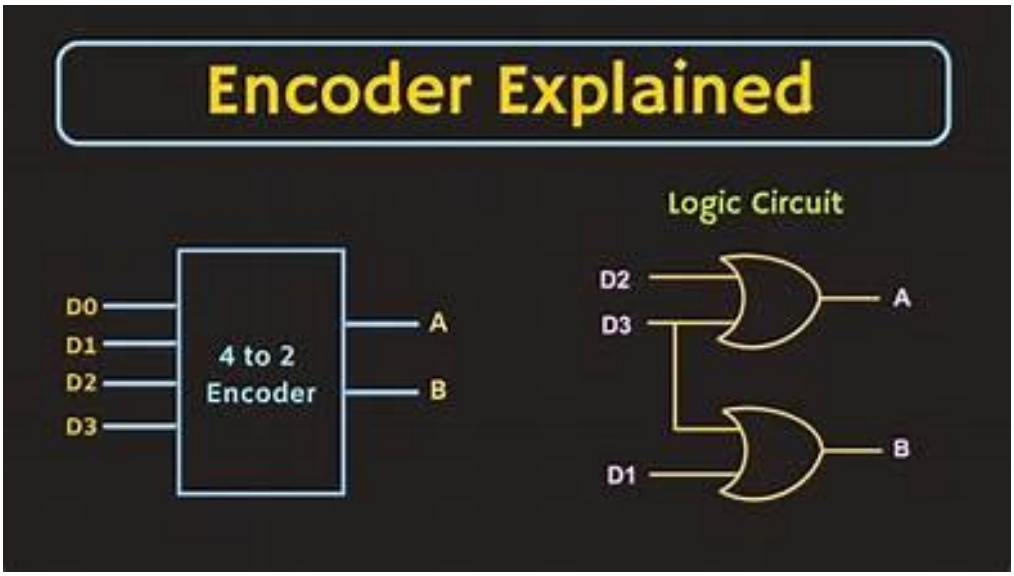

Figure 3: 4 to 2 Encoder & Using Basic Gates

Inputs			Outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	×	×	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Table 2: 4 to 2 Encoder Truth Table

Priority Encoder

A priority encoder is a digital circuit that gives precedence to its inputs and outputs the binary code of the input with the highest priority. Even if multiple inputs are active at the same time, the circuit ensures that only the highest-priority input is represented in the output.

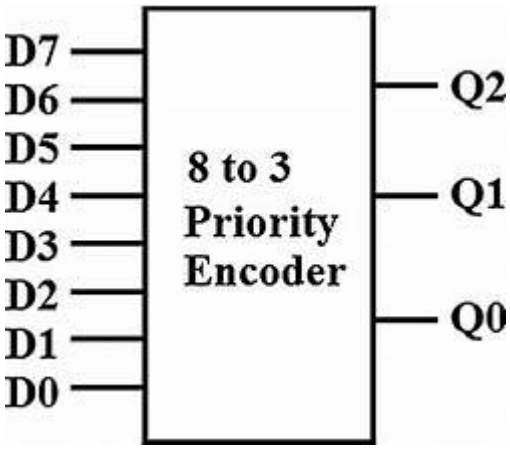

Figure 4: 8 to 3 Priority Encoder

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = Don't Care

Table 3: 8 to 3 Priority Encoder Truth Table

3. Multiplexer

A multiplexer (MUX) is a combinational logic circuit that routes one of several input signals to a single output line, depending on the combination of it's select lines. For a multiplexer with 2^n inputs, N selection lines are used to choose which input is connected to the output.

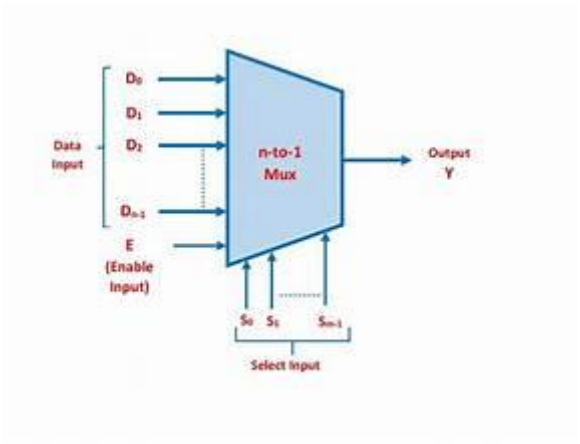

Figure 5: N to 1 Multiplexer

A 4-to-1 multiplexer can be implemented using only 2 inverters, 4 AND gates, and 1 OR gate. The two select lines are inverted as needed to create complemented signals. Each of the four AND gates combines one input with the appropriate combination of select and inverted select lines, so that only the chosen input is active. The outputs of all AND gates are then fed

into a single OR gate to produce the final output. This configuration efficiently selects one input out of four using minimal basic gates.

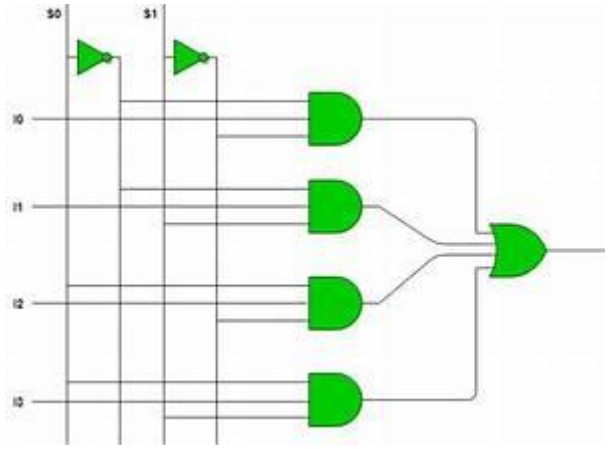

Figure 6: 4 to 1 Multiplexer using Basic Gates

Truth Table

S0	S1	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Table 4: 4 to 1 Multiplexer Truth Table

4. De-Multiplexer

A demultiplexer (DEMUX) is a combinational logic circuit that takes a single input and directs it to one of multiple outputs, based on the state of the select inputs. It functions as the opposite of a multiplexer, enabling one data signal to be distributed to multiple possible outputs, with only one active at a time.

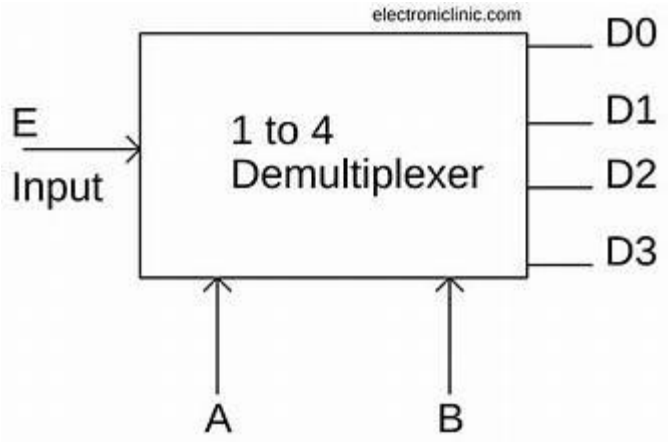

Figure 7: 1 to 4 De-Multiplexer

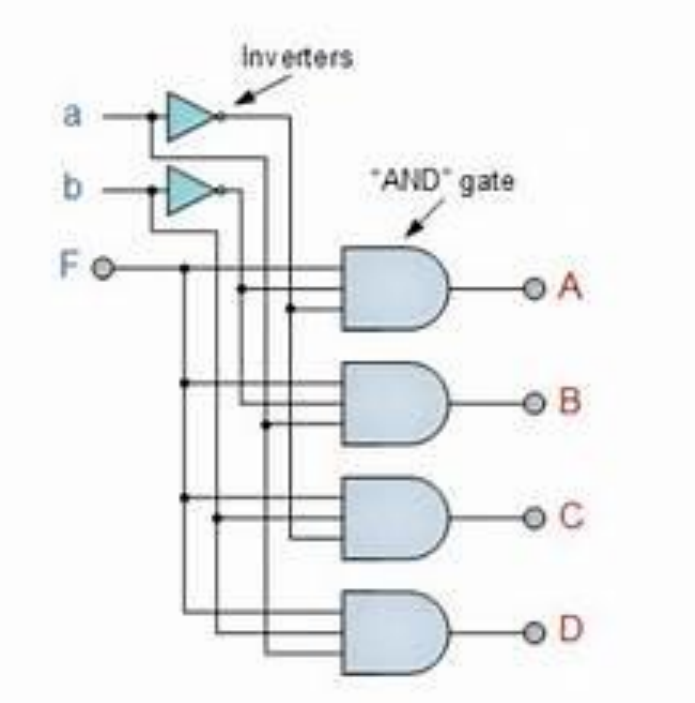

Figure 8: 1 to 4 De-Multiplexer Using Basic Gates

"F represents Data"

E	A	B	D0	D1	D2	D3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

electroniclinic.com

Table 5: 1 to 4 De-Multiplexer Truth Table

Procedure & Discussion

Constructing a 4 to 2 Encoder with Basic Gates:

Connection:

The connections were first set up as illustrated in Figure 9. We connect every wire as it shown in the experiment lab manual.

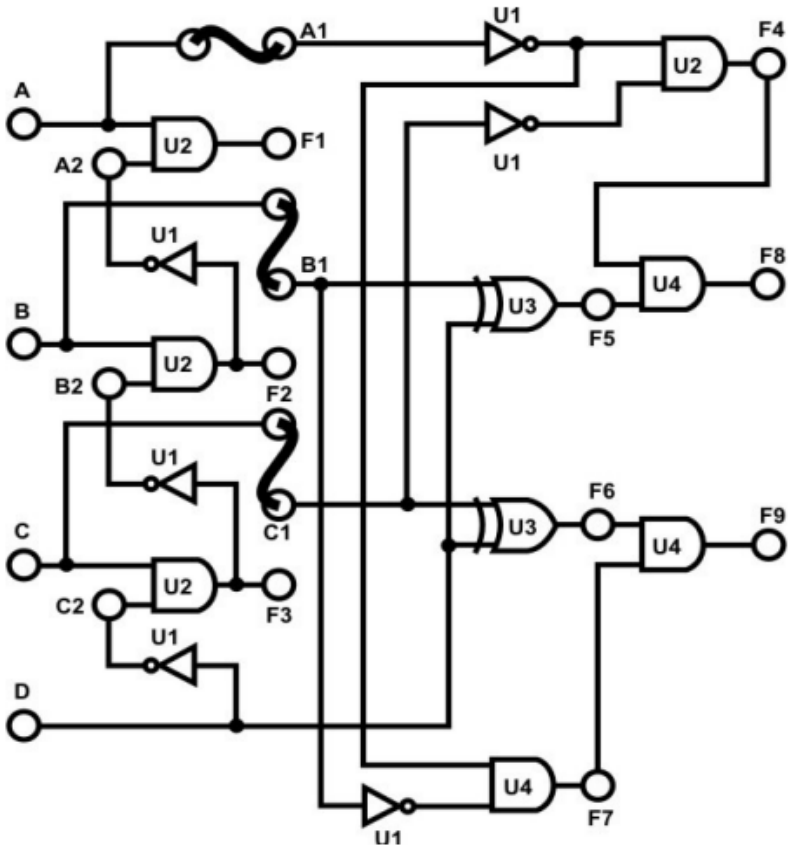

Figure 9: Wiring diagram of 4 to 2 line Encoder

The +5V output of the KL-33005 module was connected to the +5V power input of the KL-31001. The input lines A through D were assigned to switches SW0 to SW3,

while the outputs F0 and F1 were linked to logic indicators L0 and L1. By applying the input combinations for A, B, C, and D as specified in Table 6, the corresponding output states were observed and documented.

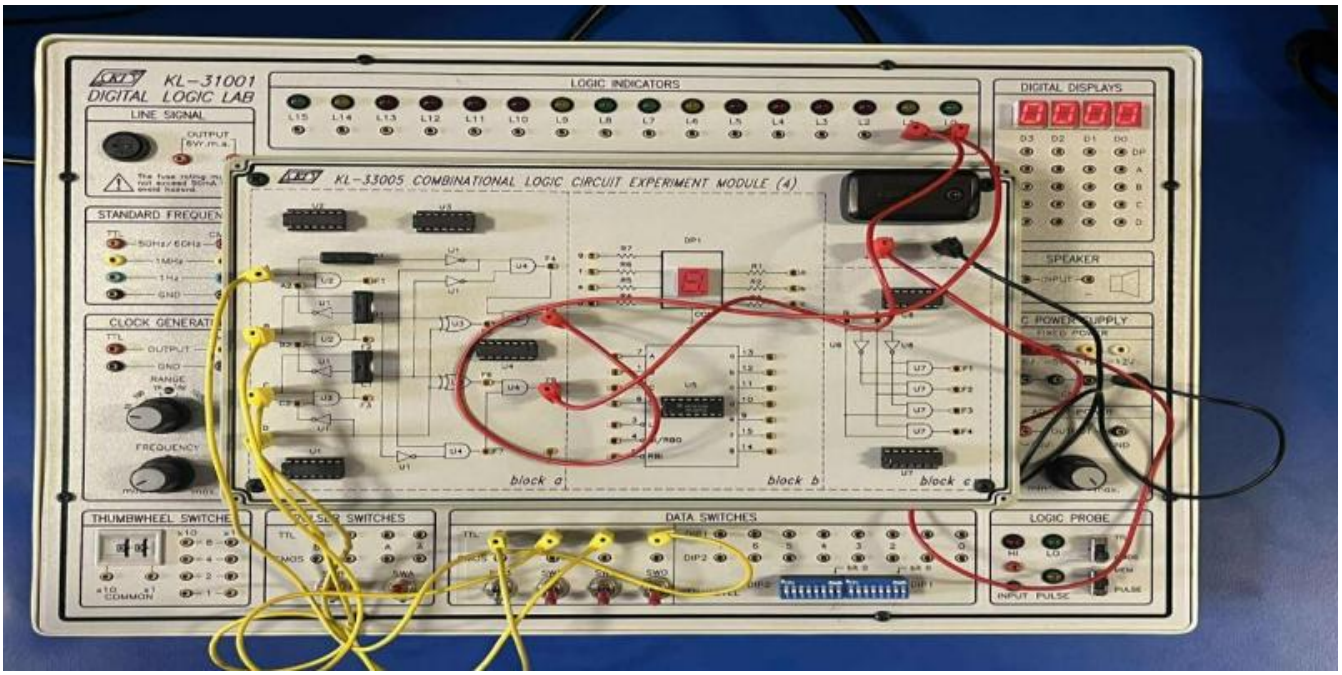

Figure 10: Our Wiring diagram for 4 to 2 Encoder

Results:

After completing the wiring connections as instructed and carefully following the given steps, we obtained the following results. The output states corresponded correctly to the applied input combinations, confirming the expected behavior of the circuit. These results demonstrate the successful implementation of the experiment and validate the theoretical concepts studied.

Inputs				Outputs	
A	B	C	D	F0	F1
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Table 6: Our 4 to 2 Encoder Truth Table

Discussion:

As the result show, it was observed that the encoder receives multiple inputs and translates them into a binary code. The circuit generates a valid output only

when a single input is active. In cases where more than one input is activated simultaneously, the output becomes 0, which represents an undefined condition.

Construct 9 to 4 Line Encoder:

Connection:

The connections were first set up as illustrated in Figure 11. We connect every wire as it shown in the experiment lab manual.

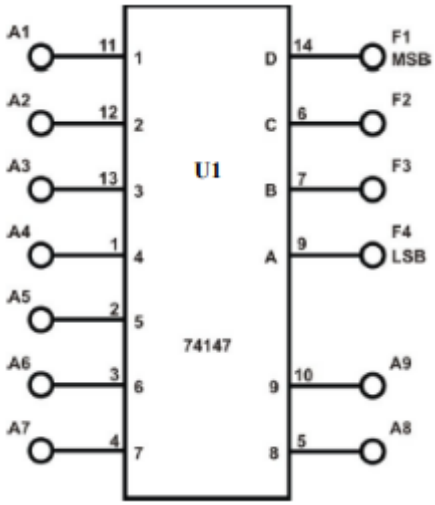

Figure 11: BCD Priority Encoder

In this part of the experiment, the KL-33006 module was used together with the 74147 (U1) blocks. The +5V supply of the KL-33006 was first connected to the +5V output of the fixed power source, and the remaining wiring was completed as illustrated in Figure 12. The input pins of U1 (A1–A8) were connected to DIP switches 1.1–1.8, while A9 was connected to switch S1. The outputs F1–F4 were then linked to logic indicators L1–L4, respectively. After completing the setup, the input sequences provided in Table 7 were applied, and the corresponding output states were recorded, with careful consideration of both active LOW and active HIGH polarities when analyzing the results.

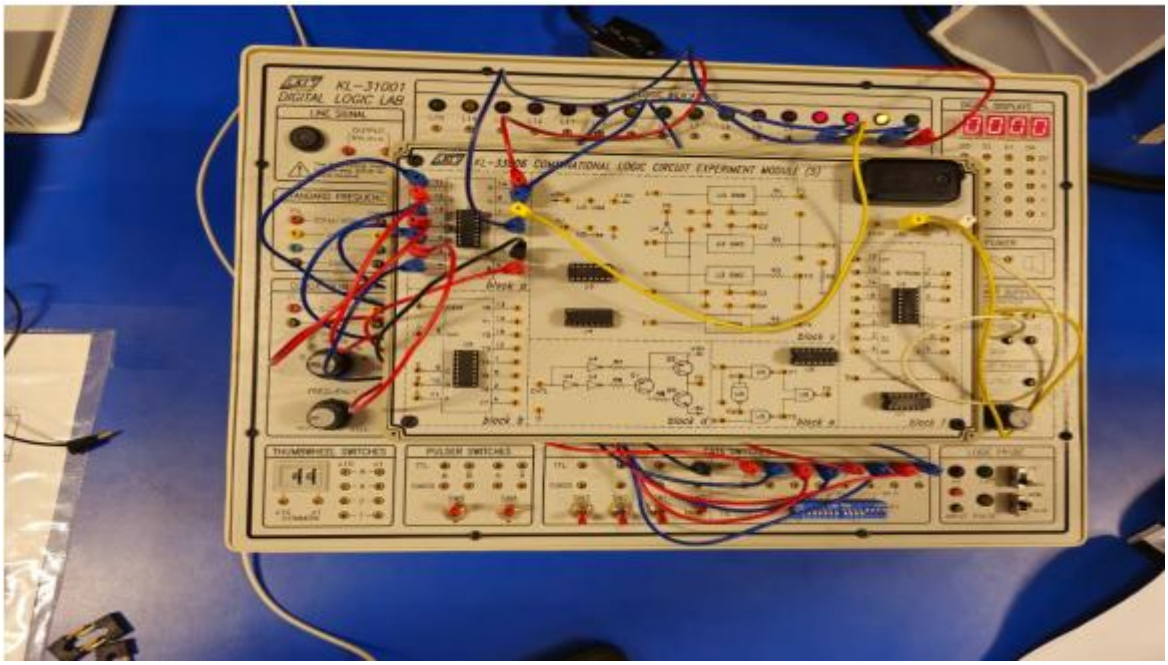

Figure 12: Our 74147 BCD Priority Encoder

Results:

Inputs									Outputs			
A9	A8	A7	A6	A5	A4	A3	A2	A1	F1(MSB)	F2	F3	F4(LSB)
0	1	1	1	1	1	1	1	1	0	1	1	0
0	0	1	1	1	1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	0	1	1	0	1
1	1	1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	0	0	0	0	1	0	1	1
1	1	1	1	0	1	1	1	1	1	0	1	0
1	1	1	1	0	0	0	1	1	1	0	1	0
1	1	1	0	1	1	1	0	0	1	0	0	1
1	1	0	1	1	0	1	1	0	1	0	0	0
1	1	0	0	0	1	1	1	1	1	0	0	0
1	0	0	0	0	1	1	1	1	0	1	1	1

Table 7: Our BCD Priority Truth Table

Discussion:

It was observed that the circuit functions as a BCD priority encoder with both inputs and outputs operating in active LOW logic. The priority sequence is arranged from A9 down to A1. As indicated in Table 7, when input A9 is set to 0, the output

lines F1 through F4 take the value 0110, which corresponds to the digit 9 in active-LOW binary form.

Construct 2 to 4 Line Decoder with basic gates:

Connection:

We used Block Decoder 1 of the KL-33005 module. The +5V pin of the module was connected to the +5V output of the fixed power supply. Inputs A and B were assigned to switches SW0 and SW1, while outputs F1–F4 were linked to indicators L0–L3. After applying the input combinations for A and B from Table 8, the corresponding output states were observed and recorded.

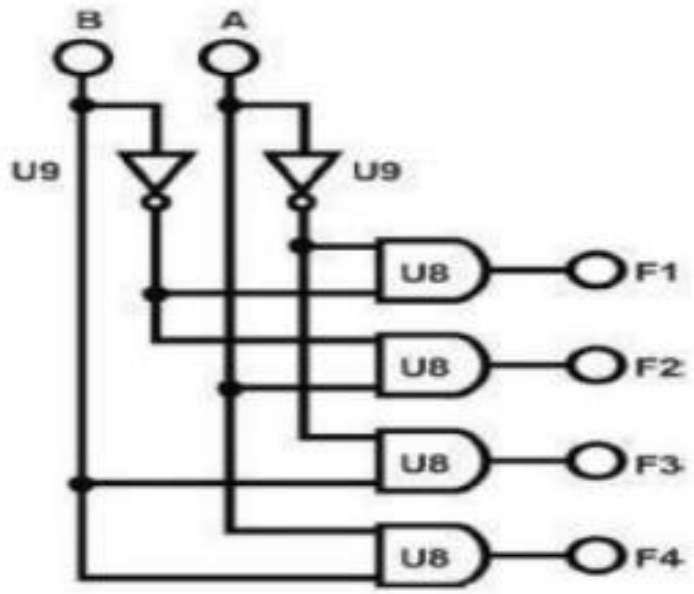

Figure 13: 2 to 4 Decoder

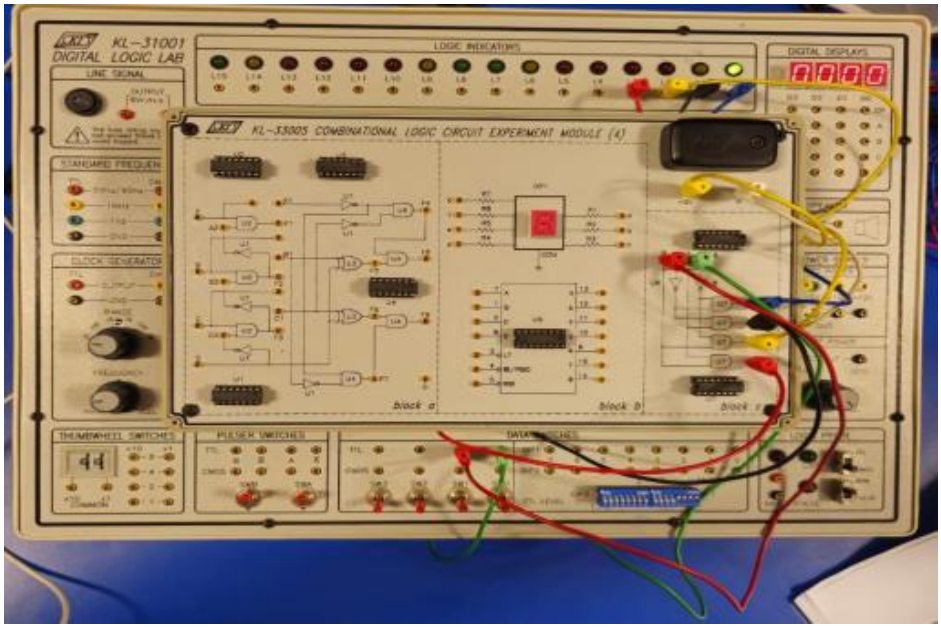

Figure 14: Our 2 to 4 Decoder

Results:

Inputs		Outputs			
B	A	F1	F2	F3	F4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Table 8: Our 2 to 4 Decoder Truth Table

Discussion:

Table 8 shows that the decoder functions with active HIGH outputs. Its operation is realized through a combination of AND gates and NOT gates.

Construct 4 to 10 Line Decoder:

Connection:

For this part of the experiment, we used the U10 (7442) IC on Block C of the KL-33004 module, as it functions as a BCD-to-decimal decoder. The +5V pin of the module was connected to the +5V output of the fixed power supply. Inputs A–D were connected to switches SW0–SW3, and outputs 0–9 were linked to indicators L0–L9. After setting the switches according to Table 9, the output states on L0–L9 were observed and recorded in the same table.

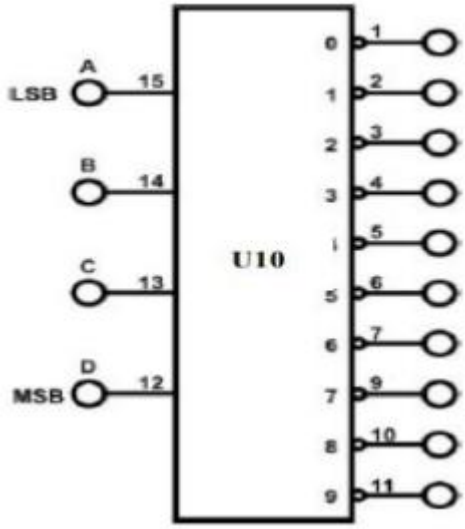

Figure 15: 4 to 10 Line Decoder

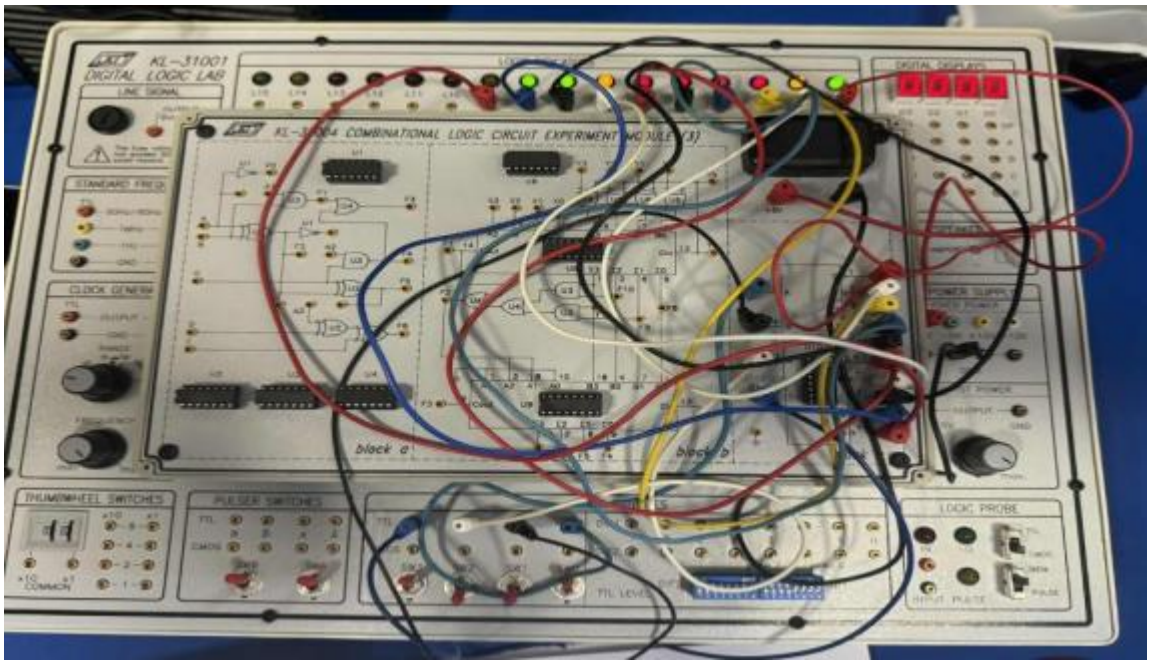

Figure 16: Our 4 to 10 Decoder

Results:

	Inputs				Outputs									
	D(MSB)	C	B	A(LSB)	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0

Table 9: Our 4 to 10 Decoder Truth Table

Discussion:

The decoder has active HIGH inputs and active LOW outputs, where a logic 0 indicates an active line. With 4 inputs, 16 combinations are possible, but only 10

represent valid BCD values; the remaining 6 combinations produce all outputs HIGH, indicating invalid states.

Construct 2 to 1 Line Multiplexer with basic gates:

Connection:

Block E of the KL-33006 module was used as a 2-to-1 multiplexer. The +5V of the module was connected to the fixed power supply. Inputs A and B were linked to switches SW0 and SW1, while the selection line C was connected to SW2. The output F3 was connected to indicator L0. Input sequences from Table 10 were applied, and the resulting states of F3 were recorded.

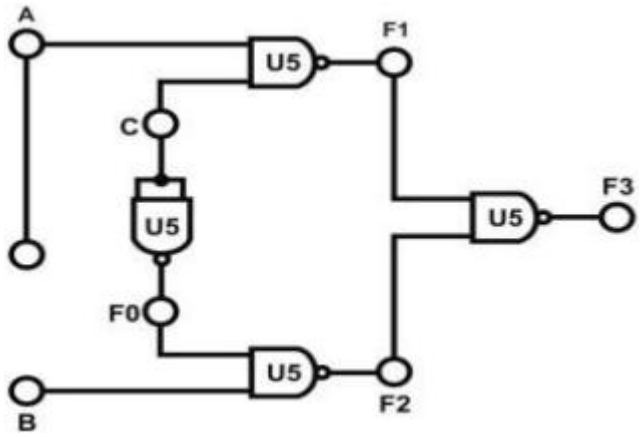

Figure 17: 2 to 1 Mux

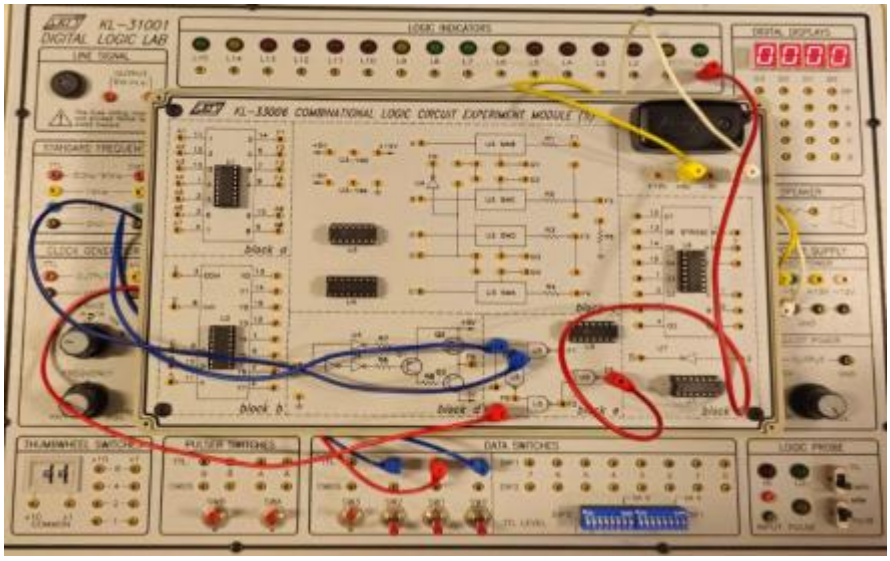

Figure 18: Our 2 to 1 Mux

Results:

Inputs			Output
C	B	A	F3
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Table 10: Our 2 to 1 Mux Trith Table

Discussion

The truth table shows that the multiplexer selects one input to pass to the output based on the selection line. As seen in Table 10, when C = 0, the output follows input B, and when C = 1, it follows input A.

Construct 8 to 1 Line Multiplexer with IC:

Connection:

For this section, we used the 74LS151 multiplexer IC located on Block F of the KL-33006 module. The module's +5V was connected to the fixed power supply. Data inputs D0 through D7 were assigned to DIP switches 1.0–1.7, while the selection inputs C, B, and A were connected to switches SW2, SW1, and SW0. The multiplexer output Q was linked to indicator L0. To operate the MUX correctly, the STROBE input was set LOW (0), enabling normal functionality.

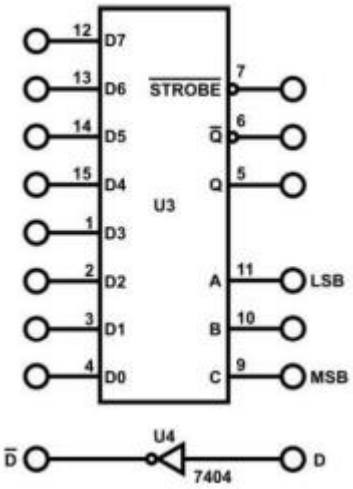

Figure 19: 8 to 1 Mux

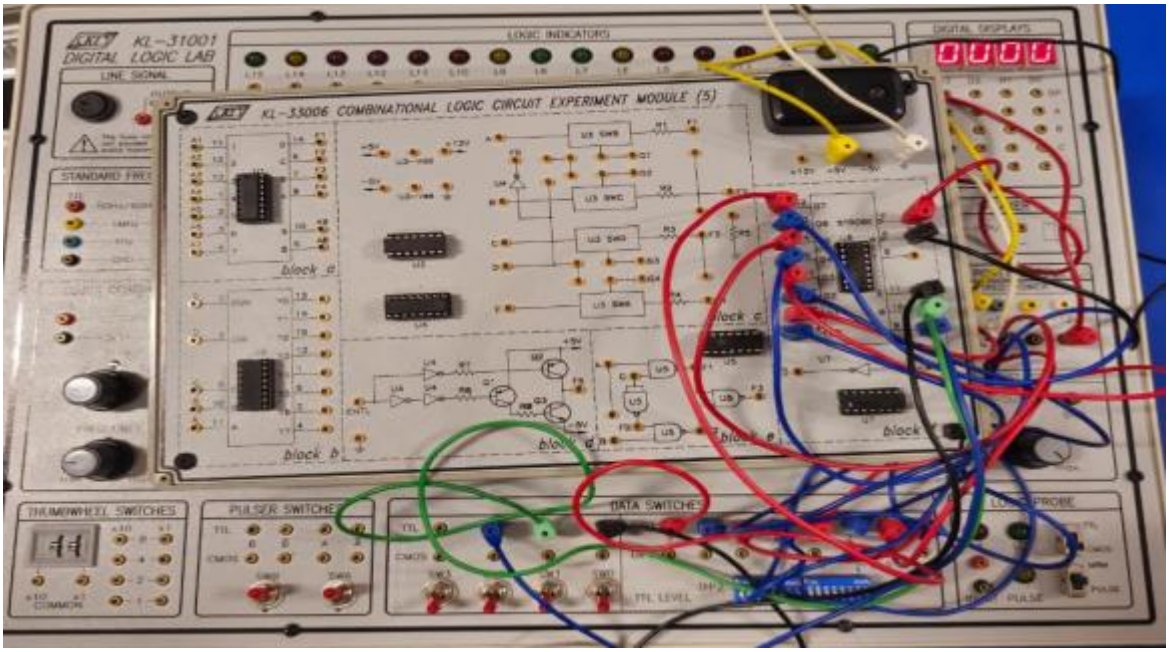

Figure 20: Our 8 to 1 Mux

Results:

Inputs			Output
C	A	B	Q
0	0	0	D0 = 1
0	0	1	D1 = 0
0	1	0	D2 = 1
0	1	1	D3 = 0
1	0	0	D4 = 1
1	0	1	D5 = 0
1	1	0	D6 = 1
1	1	1	D7 = 0

Table 11: Our 8 to 1 Mux

Discussion:

This multiplexer uses three control inputs, A, B, and C, to select one of the eight data inputs (D0–D7). For instance, when CBA = 000, the output Q follows D0, and when CBA = 010, Q follows D2. Table 11 details these cases, with the input sequence 01010101 applied to D0–D7. It was also noted that the IC operates correctly only when the STROBE input is LOW (0); if STROBE is HIGH (1), the output Q remains at logic 1 regardless of the selected input.

Using Multiplexer to implement a Logic Function:

Connection:

In this part of the experiment, we used the same 8-to-1 multiplexer to implement the logic function F(A, B, C, D) = Σ(0, 2, 4, 5, 7, 8, 10, 11, 15). The +5V supply of the KL-33006 module was connected to the fixed power source. Since the function has four variables, we created a truth table using D as the data input and A, B, C as the selection lines. Inputs A–D were wired to switches SW3–SW0, and the selection lines A, B, C were connected to the multiplexer. The output Q was linked to indicator L2, while the MUX inputs D0–D7 were assigned values {0, 1, D, D′} according to Table 12, matching each output Y to input D for every combination of CBA.

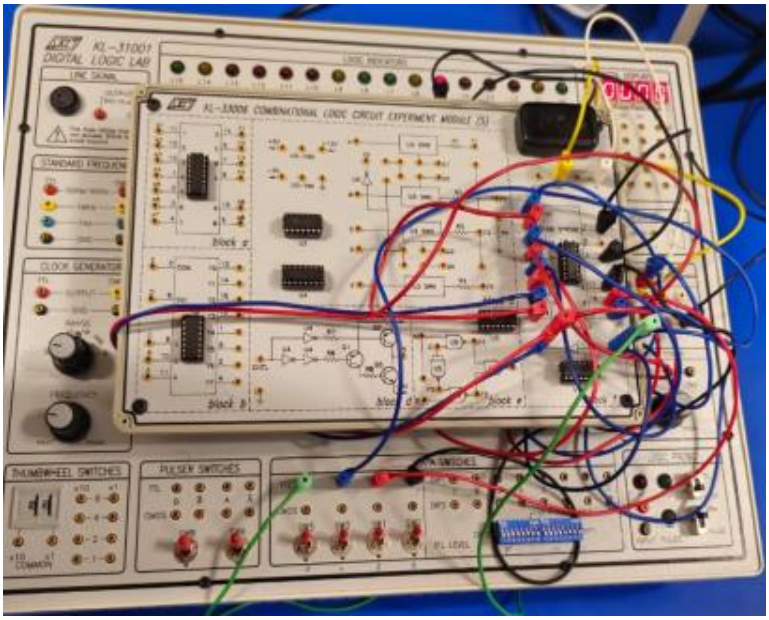

Figure 21: Our Implementation of Logic Function using Mux

Results:

Inputs				Outputs	
A	B	C	D	Y	Y
0	0	0	0	1	
0	0	0	1	0	D′
0	0	1	0	1	
0	0	1	1	0	D′
0	1	0	0	1	
0	1	0	1	1	1
0	1	1	0	0	
0	1	1	1	1	D
1	0	0	0	1	
1	0	0	1	0	D′
1	0	1	0	1	
1	0	1	1	1	1
1	1	0	0	0	
1	1	0	1	0	0
1	1	1	0	0	
1	1	1	1	1	D

Table 12: Our Implementation of Logic Function using Mux Truth Table

Discussion:

Finally, the outputs of the constructed circuit were tested and found to match the values in the function's truth table, confirming that the connections were correct.

Constructing 1-to-2 Line Demultiplexer with Basic Logic Gates:

Connection:

In this part, Block E of the KL-33006 module was used. The +5V of the IT3005 module was connected to the fixed power supply. Connections were made

as shown in Figure 3.17, with input A linked to switch SW0, input C to SW3, and outputs F1 and F2 connected to indicators L1 and L2. First, C was set to 0, and the response of F1 and F2 was observed as input A was varied. Then, C was set to 1, and changes at input A were applied again while recording the outputs of F1 and F2.

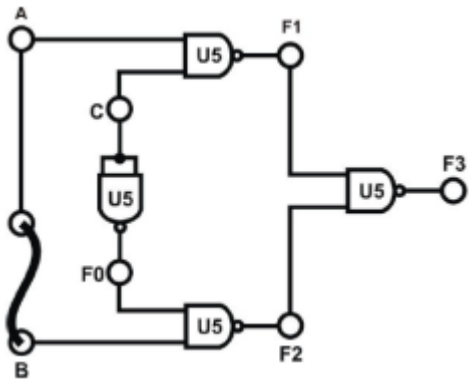

Figure 22: 1 to 2 De-Mux

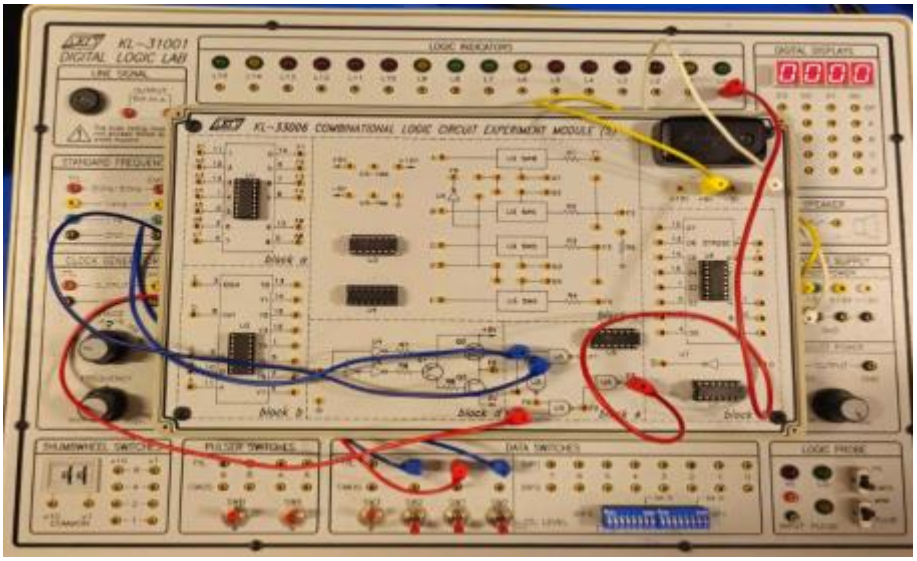

Figure 23: Our 1 to 2 De-Mux

Results:

Inputs		Outputs	
C	A	F1	F2
0	0	1	1
0	1	1	0
1	0	1	1
1	1	0	1

Table 13: Our 1 to 2 De-Mux Truth Table

Discussion

As shown in the truth table that the circuit uses active-HIGH inputs and outputs. The demultiplexer routes the input data to one of its outputs depending on the N-bit select line. For a DEMUX with input A and selection line C, the output is determined directly by the value of C.

Construct 2 to 4 Line Decoder with basic gates:

Connection:

For this part, we used the U2 (4051) demultiplexer IC on Block B of the KL-33006 module. The module's +5V was connected to the fixed power supply. Input lines E and D were assigned to DIP switches 1.0 and 1.1, while the selection inputs A, B, and C were connected to switches SW0, SW1, and SW2. Outputs Y0 through Y7 were connected to logic indicators L0–L7 to monitor the results.

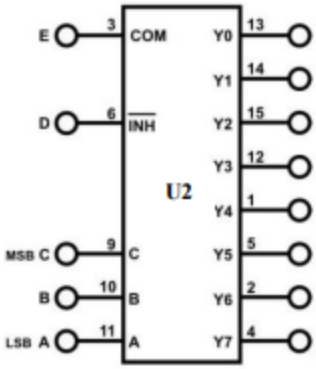

Figure 24: 1 to 8 De-Multiplexer

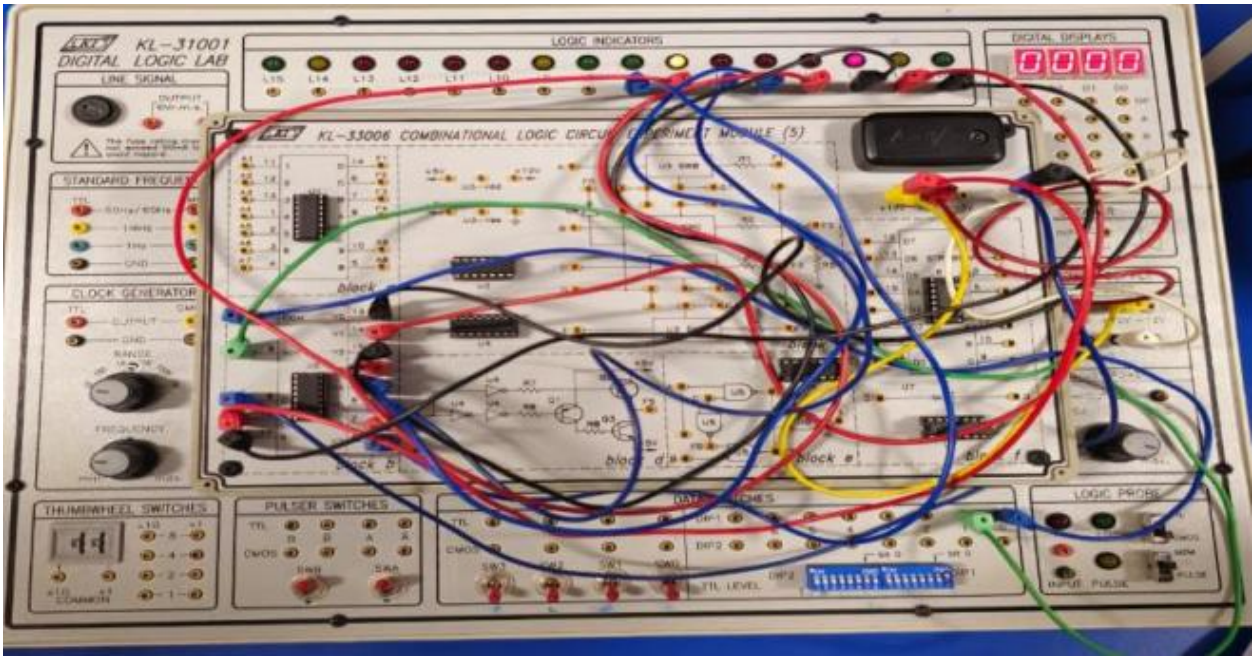

Figure 25: Our 1 to 8 De-Mux

Results:

Inputs			Outputs							
C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Table 14: Our 1 to 8 De-Mux Truth Table

Discussion

The 1 to 8 line demultiplexer functions like a decoder with an enable input. The selection lines C, B, and A determine which output receives the input data. When the enable input D is 0 (active LOW), the circuit operates correctly. For example, if CBA = 011 (decimal 3), the input is directed to Y3 while all other outputs remain at 0. The same behavior applies to all other input combinations.

Conclusion:

This experiment provided hands-on experience with digital combinational circuits, enhancing our understanding of how encoders, decoders, multiplexers, and demultiplexers operate. We learned to implement logic functions using multiplexers and practiced constructing circuits with both basic gates and ICs, which strengthened our theoretical knowledge and practical skills. The observed results matched the expected outcomes, verifying the correct functioning of the circuits. Overall, the experiment improved our ability to analyze and design digital systems and boosted our confidence in using ICs for building more complex circuits.

References:

- ENCS2210 Lab-Manual
- Electronicclinic.com
- Geeks for Geeks.com