Faculty of Engineering and Technology
Electrical and Computer Engineering Department

DIGITAL ELECTRONICS AND COMPUTER ORGANIZATION LABORATORY
ENCS2110

Experiment No.3 – Encoders, Decoders, Multiplexers, and Demultiplexers

---

Prepared by:
Rasha Mansour 1210773.

Partners:
Fatima Azazmah 1200400
Mariam Awwad 1213069

Instructor: Dr.Khader Mohammad
Teaching assistant: Haleema Hmedan

Section:7
Date: 12/12/2023

Abstract

This report presents the results of the third experiment in the Digital Electronics and Computer Organization Lab focuses on the principles and applications of Encoders, Decoders, Multiplexers, and Demultiplexers. The primary objectives include understanding the operational principles of these digital components, constructing them using basic gates and integrated circuits (ICs), and applying them to implement logic functions.

Table of Contents:

Table of Figures

List of Tables

1. Theory

1.1Objectives

- To understand the operating of Encoder, Decoder, Multiplexers and Demultiplexers.

-To construct Encoder, Decoder, Multiplexers and De- multiplexers using basic gates and ICs.

1.2 Equipment Required

- IT-3000 Basic Electricity Circuit Lab
- IT-3004 Encoder/Decoder Circuits
- IT-3005 Multiplexer/Demultiplexer Circuits

1.3 Theory

1.3.1 Decoder

Decoder is a combinational circuit that has ‘n’ input lines and maximum of 2^ n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. The outputs of the decoder are nothing but the min terms of ‘n’ input variables, lines when it is enabled.

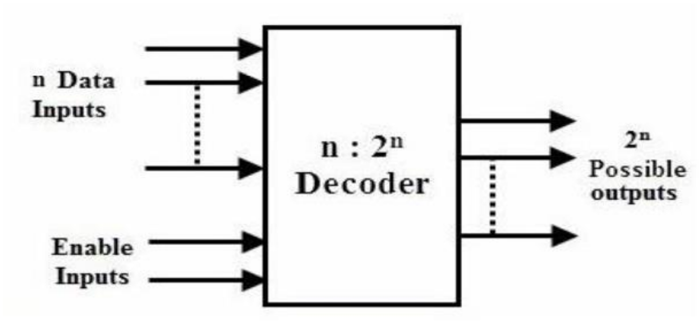


Figure1: N bit decoder with enable bit.

Enable	Inputs		Outputs			
E	A1	A0	Y3	Y2	Y1	Y0
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-to4 decoder truth table.

Constructing 2x4 line Decoder with basic gates:

Since decoder output represents minterms we can implement it using basic and gates and not gates.

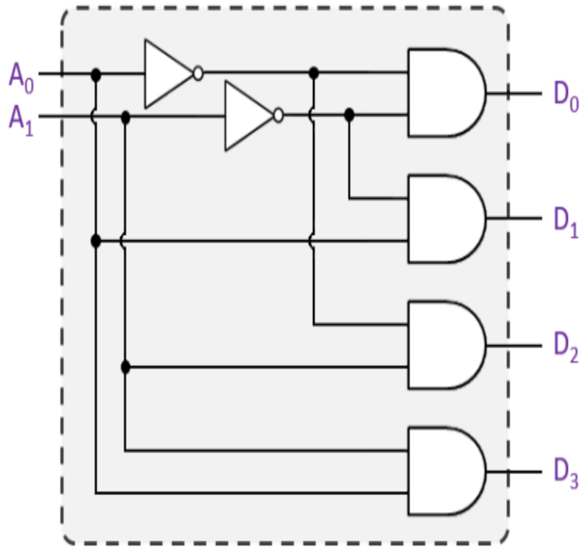


Figure 2: 2-to-4 decoder circuit.

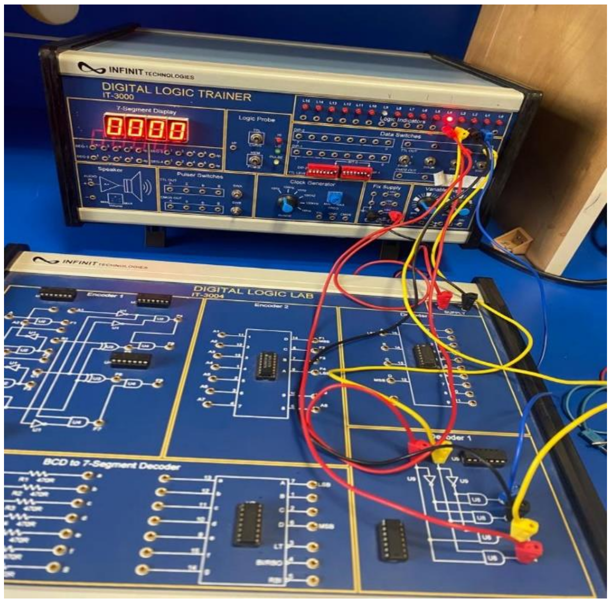


Figure 3: 2-to-4 decoder circuit implementation.

1.3.2 Encoder

An **Encoder** is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and 'n' output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with 'n' bits. It is optional to represent the enable signal in encoders.

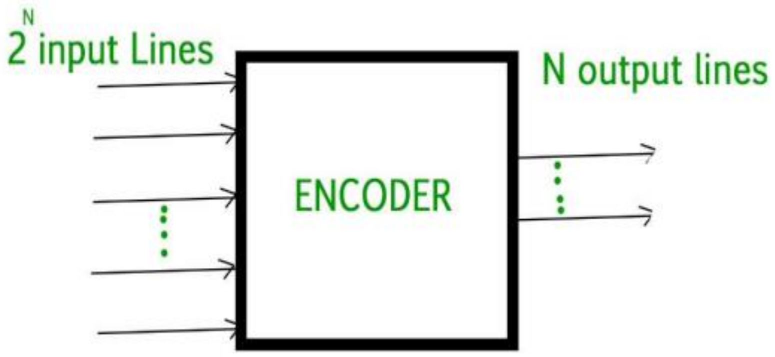


Figure 4: general Encoder.

Outputs				Inputs	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Table 2: 4-to 2 encoder truth table.

Constructing Encoder with basic gates:

Encoders can be implemented using or gates for example a 8 to 3 encoder will use 3 or gates.

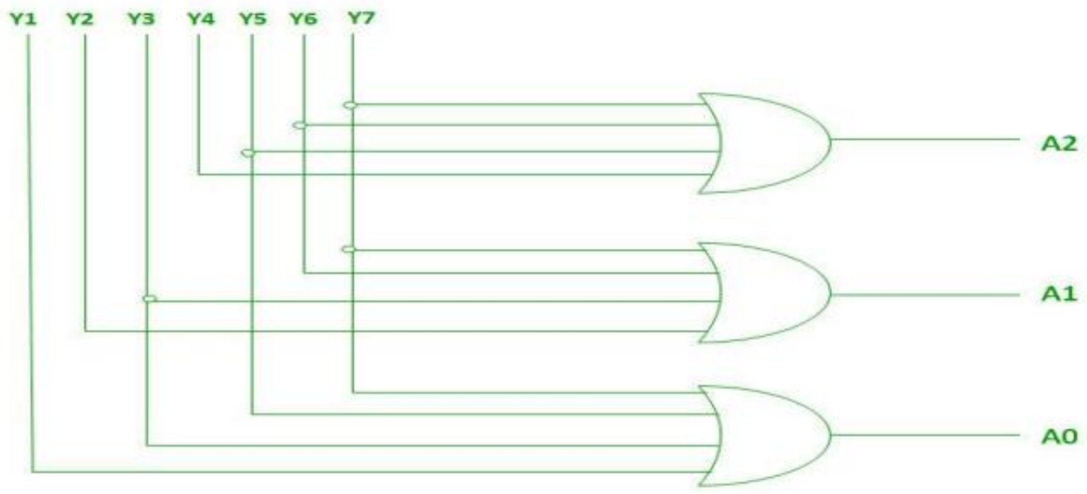


Figure 5: 8 to 3 Encoder.

The priority Encoder:

It is a type of **encoder** that was made to solve the problem that encoders are expected to have one high signal if it is active high or vice versa for active Low, if input has more than one value it may cause wrong input so to solve This priority encoder will only receive the first least significant active bit.

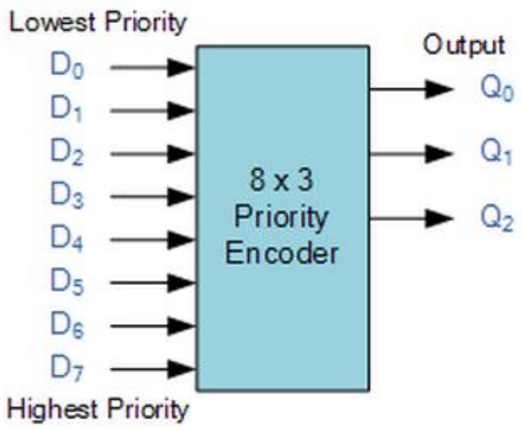
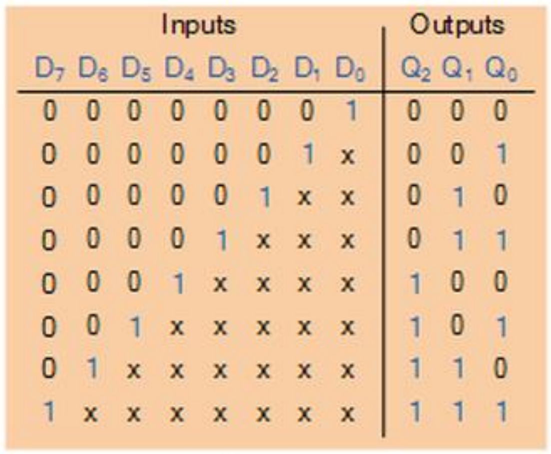


Figure 6: 8 to 3 priority encoder block diagram.



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 = dont care

Table 3: 8-to 3 priority encoder truth table.

1.3.3 Multiplexer

- **Multiplexer** is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

- Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as **Mux**.

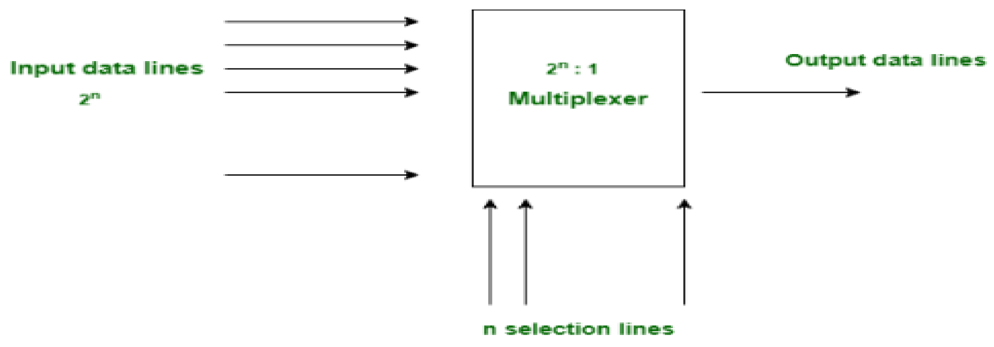


Figure 7: 4 to 1 mux diagram.

Selections		Outputs
S1	S2	Y
0	0	I_0
0	1	I_1
0	0	I_2
1	1	I_3

Table 4: 4 to 1 mux truth table.

Constructing Multiplexer with basic gates:

Multiplexers can be implemented using and & or gates (for minterms) it can also be implemented using decoders.

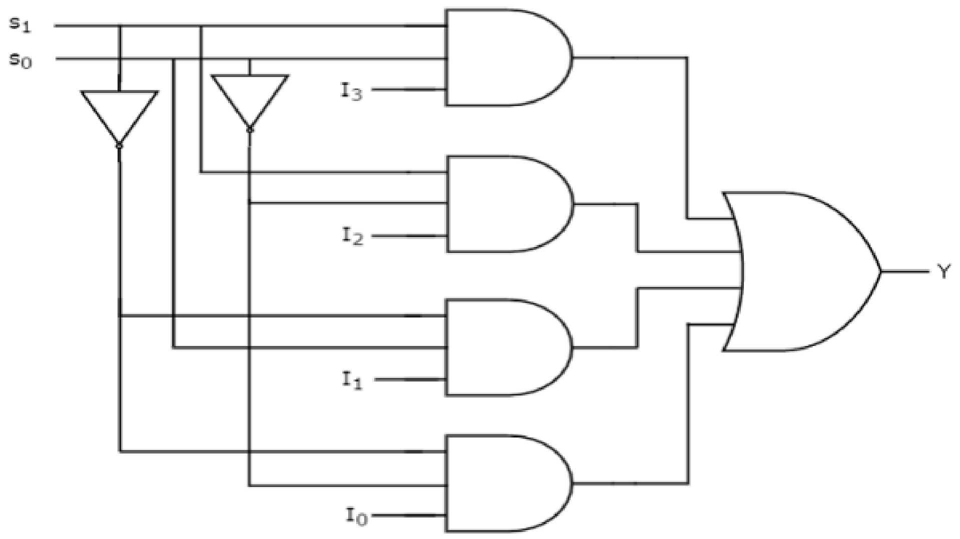


Figure 8: 4 to 1 mux circuit.

1.3.4 Demultiplexer

De-Multiplexer is a combinational circuit that performs the reverse operation of Multiplexer. It has single input, ‘n’ selection lines and maximum of 2^n outputs. The input will be connected to one of these outputs based on the values of selection lines. Since there are ‘n’ selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination can select only one output. De-Multiplexer is also called as De-Mux

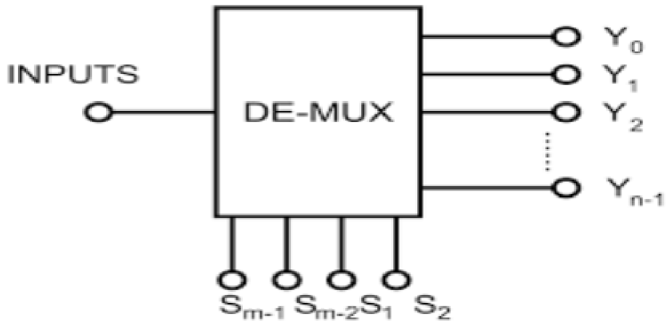


Figure 9: Demux block diagram.

Constructing Demultiplexer with basic gates:

Demultiplexer can be build using basic gate like the not and or gate.

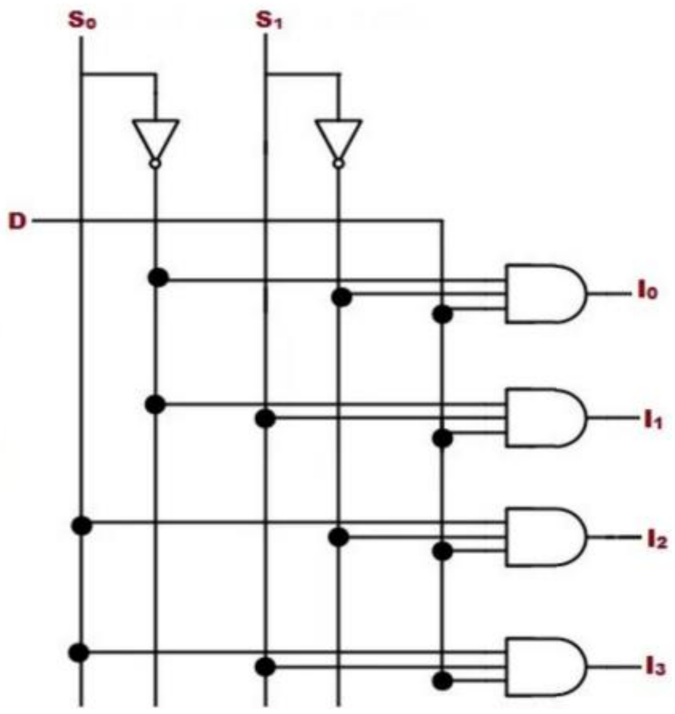


Figure 10: Demux implementation.

Selections		Outputs			
S1	**S2**	**Y3**	**Y2**	**Y1**	**Y0**
0	0	0	0	0	I
0	1	0	0	I	0
0	0	0	I	0	0
1	1	I	0	0	0

Table 5: Demux truth table.

1.4 Procedure & Discussion

1.4.1: 4 to 2 Encoder using basic gates

Using module (IT 3004 block Encoder 1) the connections should look like this:

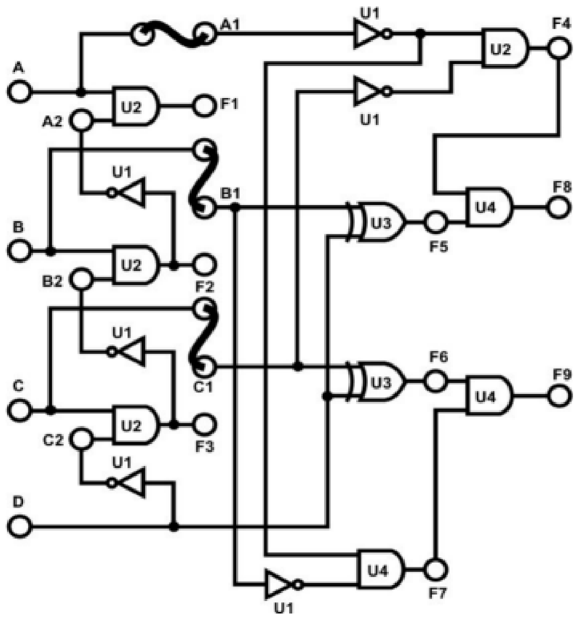


Figure 11: 4 to 2 Encoder implementation theory (IT 3004).

Depending on the inputs and connections above this is the resalted truth table:

D	C	B	A	F9	F8
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0

Table 6: experimental truth table of 4 to 2 Encoder.

From the results in the table above we notice that our encoder is an active high one, it takes the one high signal input value and the output will represent a binary code representing its value of it, notice that if there is more than one high signal in the input the output will be 0 0 which is one of the drawbacks of the normal Encoder.

1.4.2: 9 to 4 Encoder using TTL IC

This Encoder do the usual job for Encoders but instead of the usual formation of 16 inputs to 4 outputs, it will have 9 inputs only and 4 outputs.

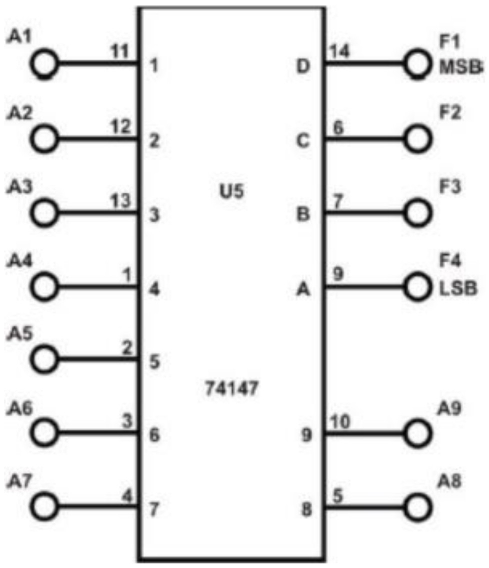


Figure 12: 4147 BCD Priority Encoder.

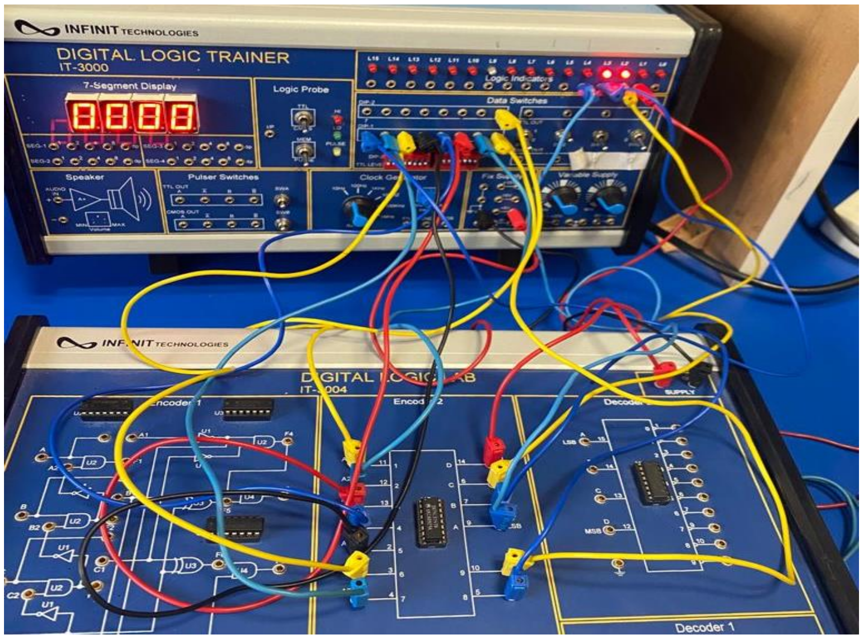


Figure 13: Real implementation of 9 to 4 Encoder.

Depending on the inputs and connections above this is the resalted truth table:

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

Table 7: experimental truth table of 9to 4 Encoder.

From the result above we notice that this Encoder is an active low one because it takes the values of the first low signal and presents the output depending on it, we also notice that it is a priority encoder that takes the first most significant low signal and ignores any other low signal (Note the output is from the least significant to the most).

1.4.3: 2 to 4 Decoder using basic gates

Using module, IT 3004 Decoder block the connection should look like this:

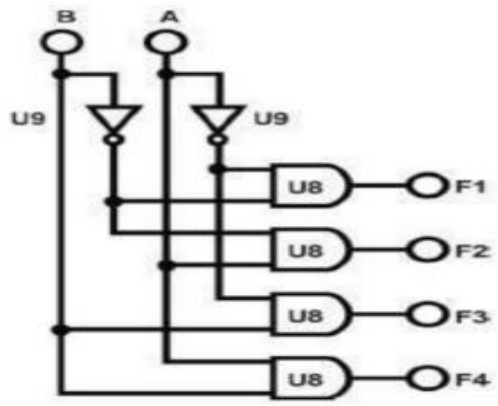


Figure 14: 2 to 4 Decoder block in IT 3004.

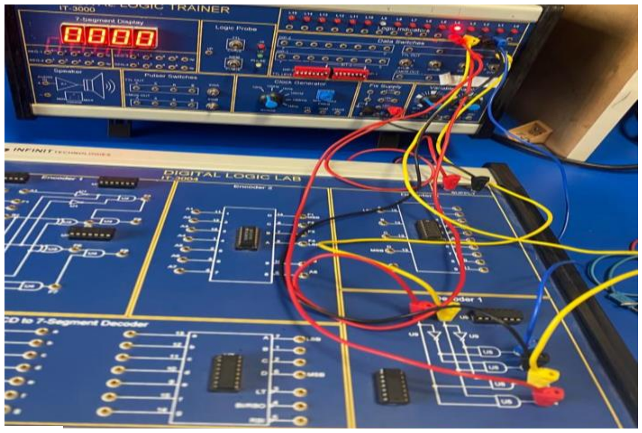


Figure 15: Real implementation of 2 to 4 Decoder.

Depending on the inputs and connections above this is the resalted truth table:

A	B	Y0	Y1	Y2	Y3
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: experimental truth table of 2 to 4 Decoder.

The output above is correct because it represent the output of the basic 2 to 4 Decoder , note that f1 represent 0 and f2 1, f3 2 and f4 4, output is Equal to the binary code from the input.

1.4.4: 4 to 10 Decoder using TTL IC

This Encoder do the usual job for Decoders but instead of the usual formation of 4 inputs to 16 outputs, it will have 10 outputs for the usual 4 inputs.

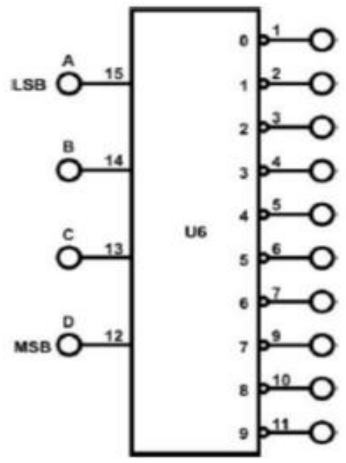


Figure 16: 4-to-10-line Decoder.

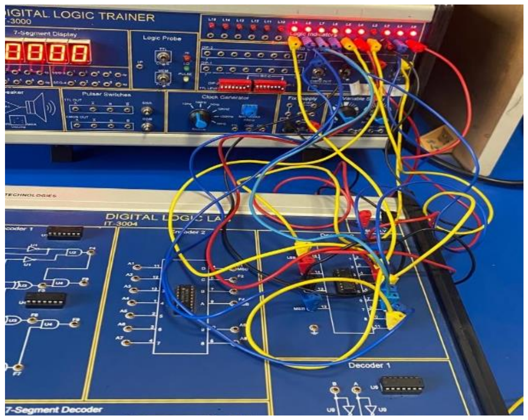


Figure 17: 4 to 10 Decoder real life implementation.

Depending on the inputs and connections above this is the resalted truth table:

D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1

Table 9: experimental truth table of 4 to 10 Decoder.

According to the values above we realize that it is an active low Encoder and the highest value it can take values from 0 (0000) to 9(1001) (ten values), the low signal in the output denotes the value of the binary code from the input.

1.4.5: 2-to-1 multiplexer using basic gates

We implement it using module IT-3005 that have a circuit that represent a mux made from nor gates.

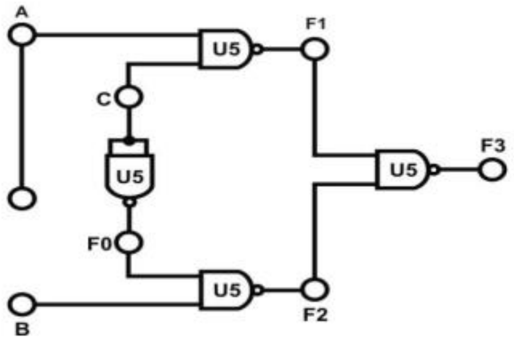


Figure 18: 2 to 1 multiplexer.

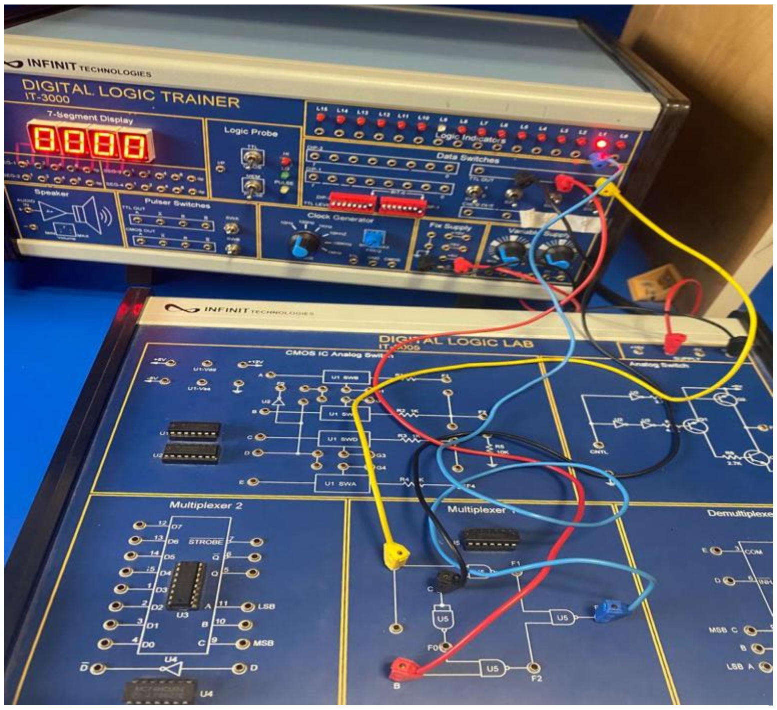


Figure 19: Real implementation of 2 to 1 multiplexer.

Depending on the inputs and connections above this is the resalted truth table:

A	B	C	F3
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 10: experimental truth table of 1 to 2 MUX.

Based on the inputs above C represent the selection bit and A and B are the inputs, if C is 0 the output will equal the input A, and if c equals 1 the output will equal the B, the C input represents a binary code that for every bit it changes the output will equal one of the input values. depending in our output our results match the basic mux.

1.4.6: 8 to 1 Multiplexer using IC

Using U3 (74LS151) on block Multiplexer 2 of module IT-3005 will be used to construct this Mux.

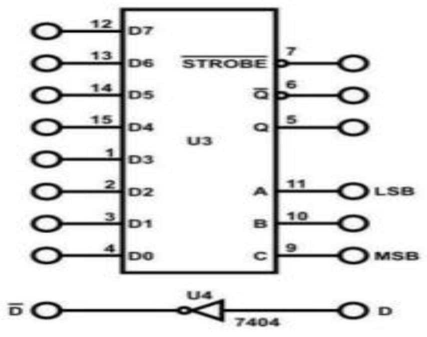


Figure 20: 8 to 1 Mux.

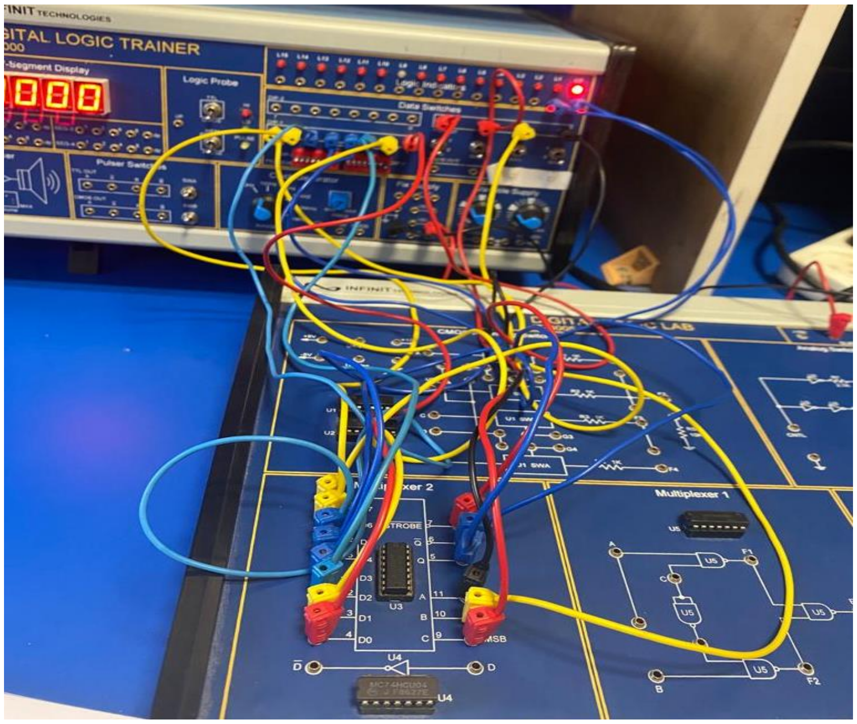


Figure 21: Real implementation of 8 to 1 Mux.

Depending on the inputs and connections above this is the resalted truth table:

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

Table 11: experimental truth table of 8 to 1 MUX.

In the table above the A, B and C represent the selection bits for the Mux Depending on the binary code from the selection bits the output is the name of the input that will be selected, in an 8 to 1 Mux the inputs are numbered from 0 to 7, for example, if the selection equals (1 1 1) the output will equal the seventh input line as shown above.

1.4.7: Implementing a function using Multiplexer

A Mux can be used to implement different Boolean functions and minterms This can be done by determining the function that satisfies the output values, and use this function as an input, this can help us to also help us to implement a function that usually need a big Mux using smaller mux and Input functions for example lets implement this function.

F (A, B, C, D) =∑ (0,2,4,5,7,8,10,110,15)

The truth table will look like this:

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1

Table 12: truth table of ∑ (0,2,4,5,7,8,10,110,15).

Notice that if we want to implement this function using 16 to 1 Mux the input will be 1 for every minterm and 0 for every maxterm, but if we want to implement it in an 8 to 1 Mux for Example, we divide the table from every two lines of output and the input will be a function of the last selection bit D As shown next to the table, it can also be reduced to 4 to 1 mux but the function inputs will be bigger and will depend in the last two selection bits, and so on this can be done until we reach 2 to 1 Mux.

1.4.8: 1 to 2 Demultiplexer using basic gates

to implement it we use the demux block in the IT-3005 as the figure below:

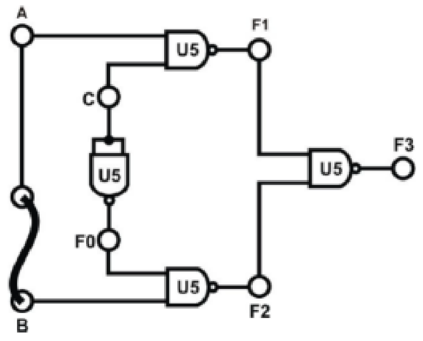


Figure 22: 1 to 2 Demultiplexer.

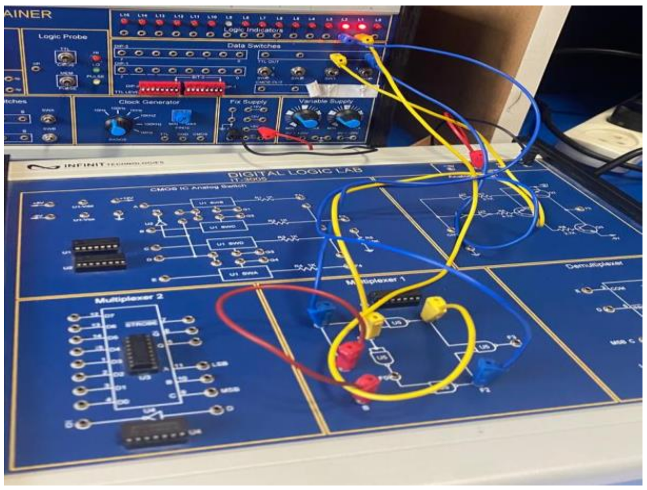


Figure 23: Real implementation of 1 to 2 Demultiplexer.

Depending on the inputs and connections above this is the resalted truth table:

C	A	F1	F2
0	0	1	1
0	1	1	0
1	0	1	1
1	1	0	1

Table 13: truth table of 1-2 demux

the selection bit c determines in which output the value of input will appear. note that this demux as an active low one, we also noticed that if c =0 f2= A complement and if c=1 f1=a complement.

1.4.9: 1 to 8 Demultiplexer with CMOS IC

We implement it using Demultiplexer blook on module IT-3005

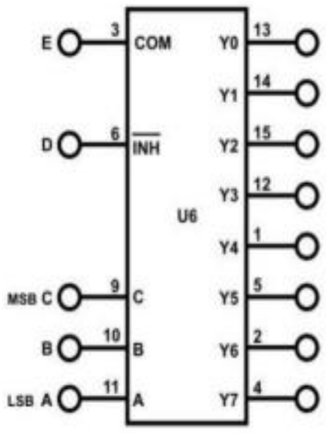


Figure 24: 1 to 8 Demultiplexer.

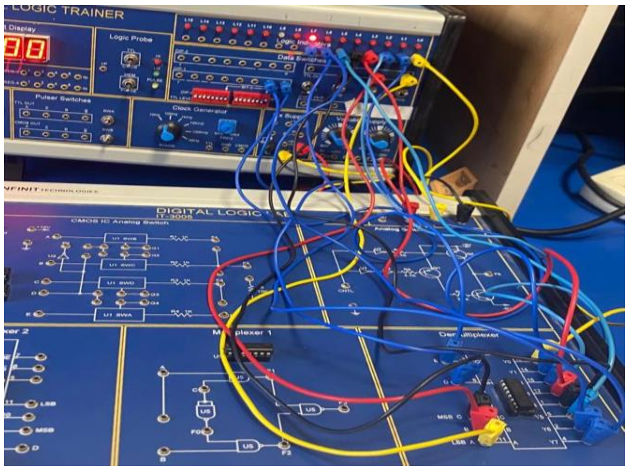


Figure 25: Real implementation of 1 to 8 Demultiplexer.

Depending on the inputs and connections above this is the resalted truth table:

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

Table 14: truth table of 1-8 demux

the 1 to 8 demux works like the regular 1 to 2 demux, a, b, c are the selection bits of it, the binary code in the selection bit will determine which output line will carry the high signals, and the rest will equal low signal, our results are true because it is the same truth table as the original truth table.

2. Conclusion

In conclusion, the experiment successfully achieved its objectives of understanding and constructing encoders, decoders, multiplexers, and demultiplexers. The practical implementation of these circuits provided valuable insights into their functionalities and limitations. The results obtained from testing aligned with theoretical expectations, confirming the accuracy and reliability of the constructed circuits.

3. References

- [1] Manual for Digital Electronics and Computer Organization Lab, 2023, Birzeit University.

https://ritaj.birzeit.edu/bzu-msgs/attach/2375682/Digital+lab+manual+2023.pdf