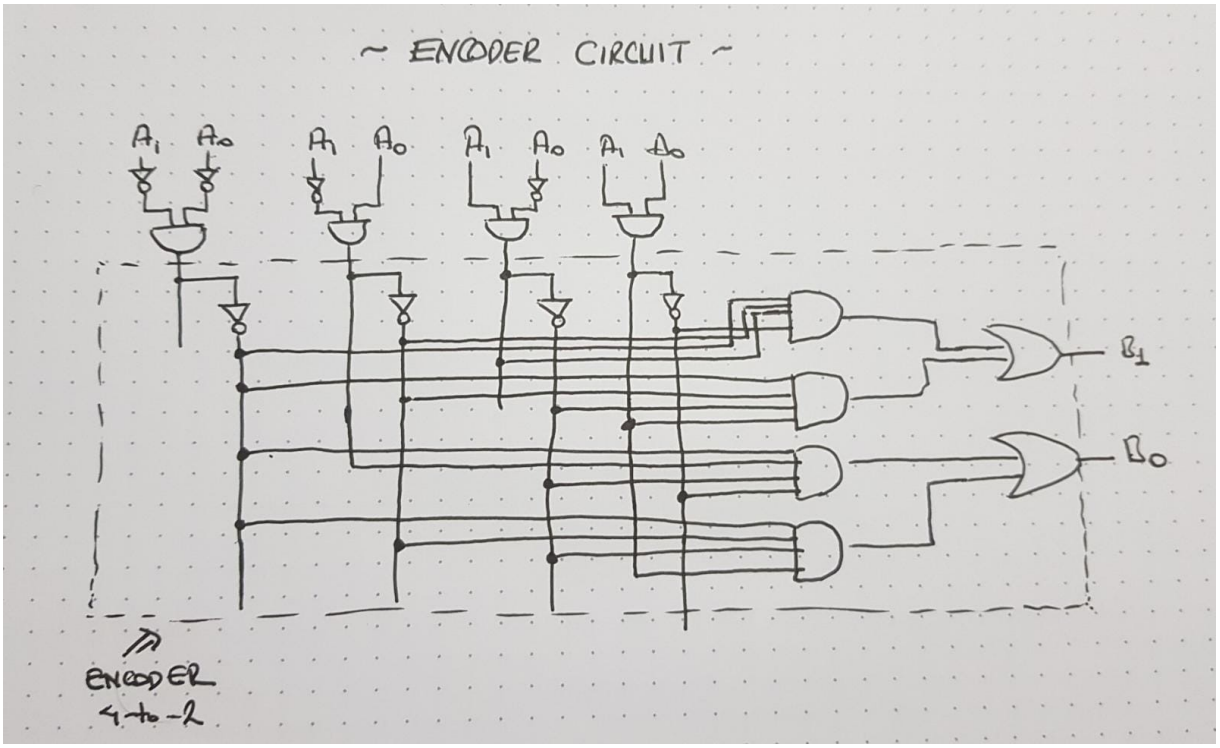
ELEC 204 Digital Design Preliminary Lab Report
Preliminary Lab 2
Name: Mehmet Enes Erciyes
Date: 10/27/2019

QUESTION 4.1

When we examine the truth table for this question, we can see that $A_1A_0 = B_1B_0$
To implement this function using an encoder, we can make the following circuit:

Figure.1 – Encoder Circuit for the Given Truth Table in Question 4.1



In this encoder circuit, truth table looks like:

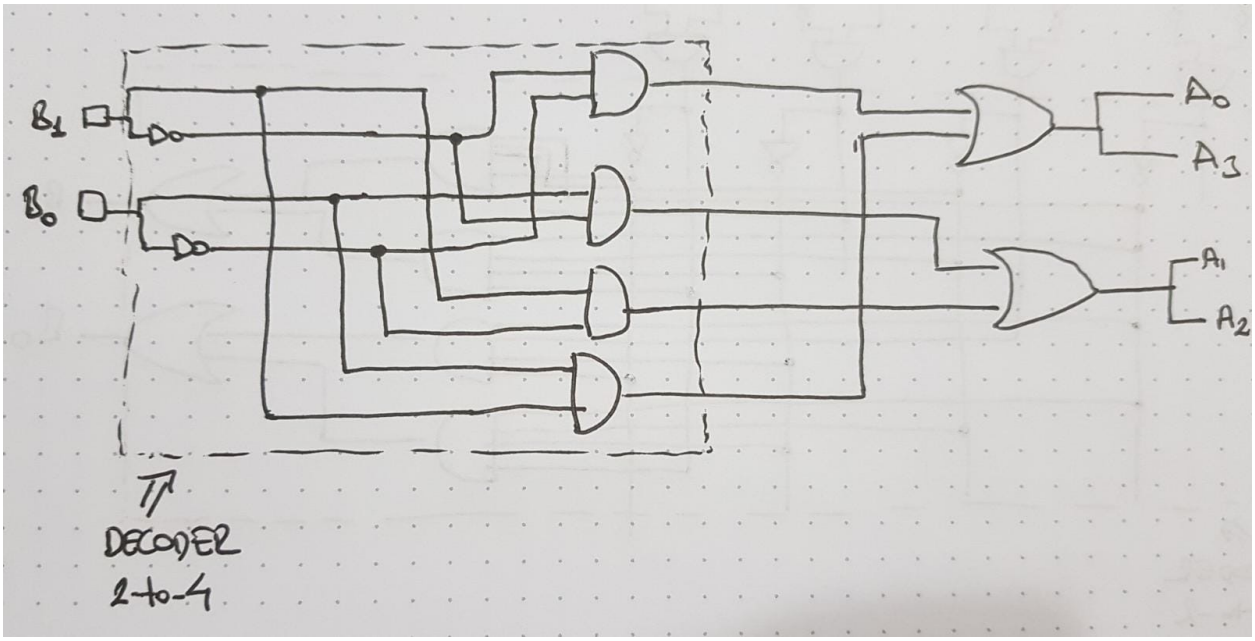
Figure.2 – Truth table for the encoder circuit

Q0	Q1	Q2	Q3	B1	B0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Q0,Q1,.. represent the compound inputs of the encoder.

QUESTION 4.2

Figure.3 – Decoder Circuit for the Given Truth Table in Question 4.2

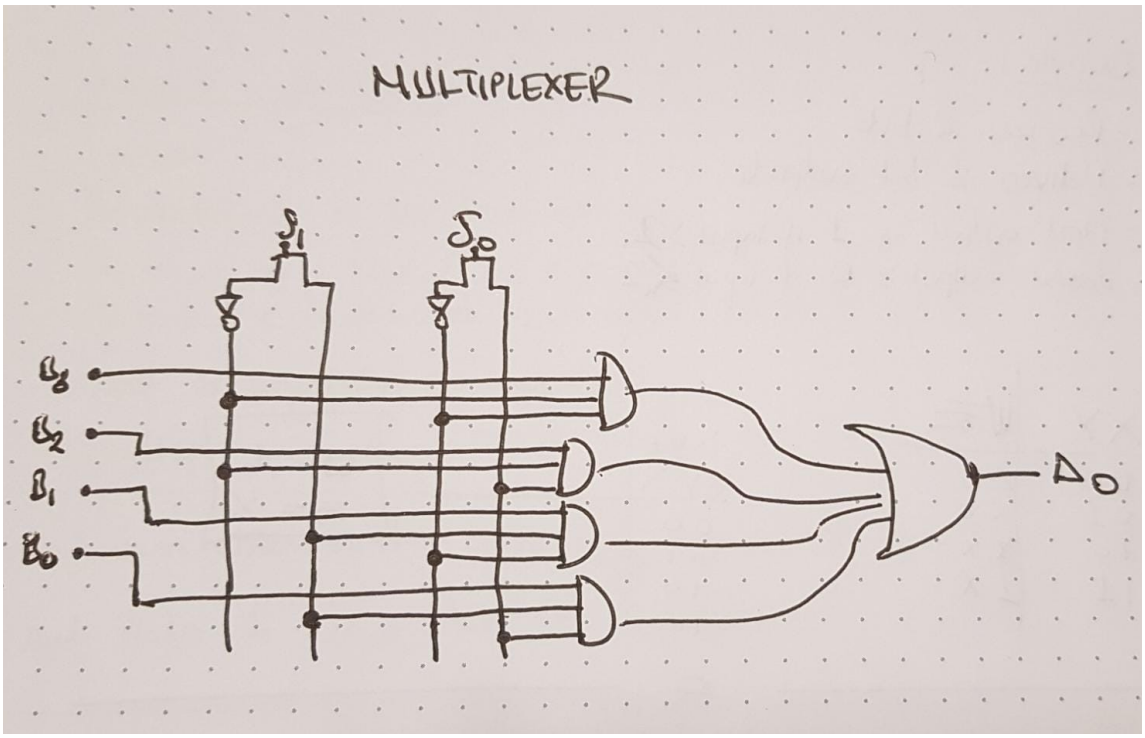


QUESTION 4.3

Figure.4 – Truth Table for Designed Multiplexer

S1	S0	I
0	0	B3
0	1	B2
1	0	B1
1	1	B0

Figure.5 – The Multiplexer Design

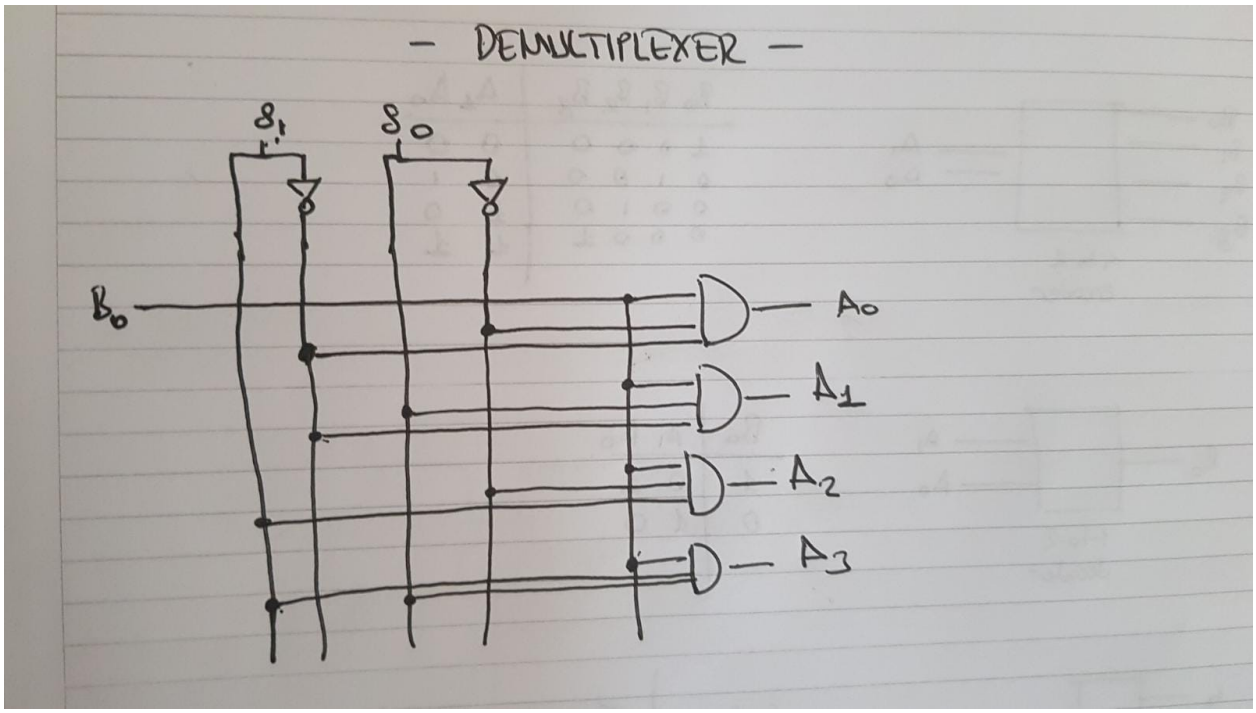


QUESTION 4.4

Figure.5 – Truth Table for Demultiplexer

S1	S0	A0	A1	A2	A3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Figure.6 – Demultiplexer Design



ELEC 204 Digital Design Lab Report

Lab 2

Name: Mehmet Enes Erciyes
Lab Collaborator: Bulut Boru

Date: 10/27/2019

*Please delete the highlighted lines and write your own parts for the report.
Reminder: Your lab grade is a weighted average of your performance before, during and after the lab: **Total lab grade = Preliminary work* (30%) + Lab interview and demo* (40%) + Lab report (30%)
***Please make sure the lab report does not exceed 4 A4 pages.**
****Please make sure you indicate the name of your lab collaborator (if there is any) who you worked together to solve the lab questions. Do not change your lab collaborator throughout the semester.

1. **Introduction and objectives**

In this lab, we designed a 4-to-2 encoder module, 2-to-4 decoder, a 4-to-1 multiplexer and an 1-to-4 de-multiplexer module. These modules are the foundational combinational logic circuits.

Encoders take 4 bits and maps them to 2 bits. There is 1-to-1 correspondence between each input and output. Encoders are used for compressing data without loss, turning rotary or linear motion into a digital value and much more.

Decoders do the reverse. They take 2 bits and map them to 4 bits. Decoders can decode the encoded data. They are also used in many display devices.

Multiplexer and demultiplexers are used to select between different inputs. They are used to send multiple communication signals over one channel and read them one by one.

When implementing these circuits, we used separate modules in Xilinx for each module.

Our code took inputs and returned outputs separately for each module. We will look at how our code works and how our inputs and outputs are defined in the next section.

2. **Methods**

Encoder Module:

```
entity encoder is
    Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
           B : out  STD_LOGIC_VECTOR (1 downto 0));
end encoder;

architecture Behavioral of encoder is
```

```
begin

B(0) <= ((not A(0)) and A(1) and not A(2) and (not A(3)))
         or ((not A(0)) and (not A(1)) and (not A(2)) and A(3));

B(1) <= ((not A(0)) and  not A(1) and  A(2) and (not (A(3))))
        or ((not A(0)) and (not A(1)) and (not A(2)) and A(3));

end Behavioral;
```

Encoder module has four inputs : A(3), A(2), A(1) and A(0) and it has two outputs: B(1) and B(0). Our code should map from four input to two output according to the following truth table:

A0	A1	A2	A3	B1	B0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

When we construct the K-Map and derive the expressions. We get the expression for B(1) and B(0) as seen in the code.

Our code does the mapping by assigning the correct expressions to output bits and represent them on the FPGA board as instructed by its implementation constraints file.

Decoder Module:

```
entity decoder is
    Port ( B : in  STD_LOGIC_VECTOR (1 downto 0);
           A : out  STD_LOGIC_VECTOR (3 downto 0));
end decoder;

architecture Behavioral of decoder is

begin
```

```
A(0) <= (not B(1)) and (not B(0));
A(1) <= (not B(1)) and (B(0));
A(2) <= B(1) and (not B(0));
A(3) <= B(1) and B(0);

end Behavioral;
```

Decoder module has two inputs : B(1) and B(0), and it has two outputs: A(3), A(2), A(1) and A(0). Our code should map from two input to four output according to the following truth table:

B0	B1	A0	A1	A2	A3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Again, when we derive the expressions, we can easily see that we get the expressions in the code.

Our code does the mapping by assigning the correct expressions to output bits and represent them on the FPGA board as instructed by its implementation constraints file.

Multiplexer Module

```
entity multiplexer is
    Port ( I : in  STD_LOGIC_VECTOR (3 downto 0);
           s : in  STD_LOGIC_VECTOR (1 downto 0);
           Y : out  STD_LOGIC);
end multiplexer;

architecture Behavioral of multiplexer is

begin

Y <= (I(0) and not s(1) and not s(0)) or
        (I(1) and not s(1) and s(0)) or
        (I(2) and s(1) and not s(0)) or
        (I(3) and s(1) and s(0));
```

```
end Behavioral;
```

Multiplexer module has six inputs : s(1) and s(0) – select bits; I(3),I(2),I(1) and I(0) – data bits and it has one output: Y

Our code should select one output from four input and return it according to the following truth table:

s0	s1	Y
0	0	I(0)
0	1	I(1)
1	0	I(2)
1	1	I(3)

Our code makes use of two boolean algebra identities:

- X * 1 = X
- X + 0 = X

When s0 and s1 are 1, first three parts of the expression for Y becomes 0 and the last part becomes “I(3) and 1”. According to the two identities above, this expression results in I(3). Same kind of reasoning applies for every output.

Our code, after assigning correct expressions, represent outputs on the FPGA board as instructed by its implementation constraints file.

Demultiplexer Module:

```
entity demultiplexer is
    Port ( I : in  STD_LOGIC;
           s : in  STD_LOGIC_VECTOR (1 downto 0);
           Y : out  STD_LOGIC_VECTOR (3 downto 0));
end demultiplexer;

architecture Behavioral of demultiplexer is

begin

Y(3) <= I and s(1) and s(0);
Y(2) <= I and s(1) and not s(0);
Y(1) <= I and not s(1) and s(0);
```

```
Y(0) <= I and not s(1) and not s(0);

end Behavioral;
```

Demultiplexer module has three inputs : s(1) and s(0) – select bits; I – input bit and it has four output: Y(3),Y(2),Y(1) and Y(0).

Our code should select which output should be returned according to the following truth table:

s0	s1	Y(0)	Y(1)	Y(2)	Y(3)
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Input I behaves like the switch that opens and closes the entire module.

When we look at the truth table, we can easily see that we will get the expressions in the code.

Our code, after assigning correct expressions, takes its inputs from switches and represent outputs with the LEDs on the FPGA board as instructed by its implementation constraints file.

3. **Problems encountered, errors and warnings resolved**

During the programming, I encountered several syntax errors. Having gained an experience from previous lab project, I was comfortable at finding and debugging the problems.

The problem that I had most trouble with was the miswritten expression in the encoder module. “not” operators were used in wrong places, therefore the encoder had not worked properly even if the code was synthesized perfectly. After a search, I found the problem and corrected it.

4. **Conclusion**

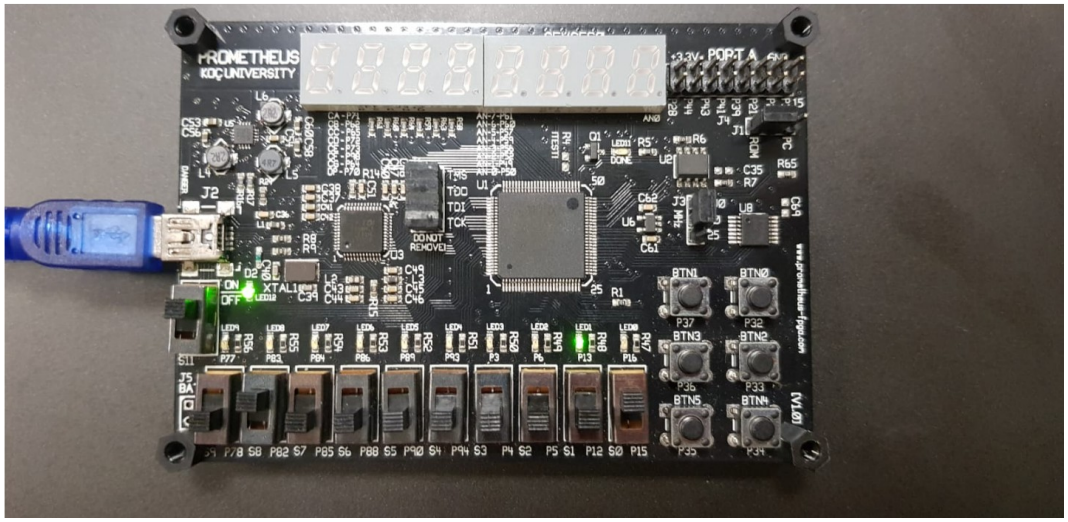
In this lab, we implemented foundational combinational logic circuits. We learned the working principles of an encoder, decoder, multiplexer and demultiplexer. We created separate modules for every circuit and implemented them from the logic gate level. Since every combinational logic circuit can be implemented with these foundational logic circuits, it was a quite important lab for next weeks.

References

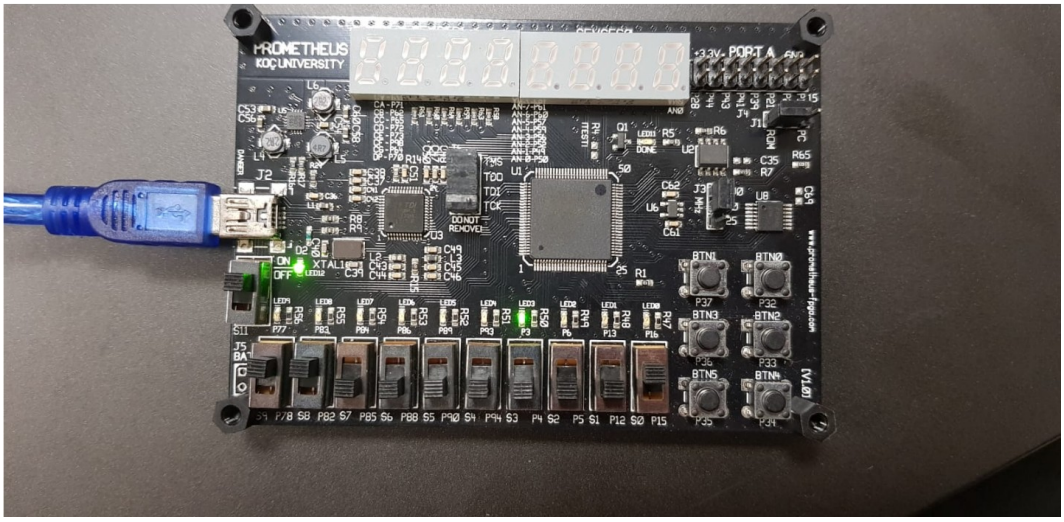
1. Lab 2 Instructions Sheet
2. Design of decoders and encoders by Engineer2009Ali on YouTube.

Appendixes are placed inside the report.

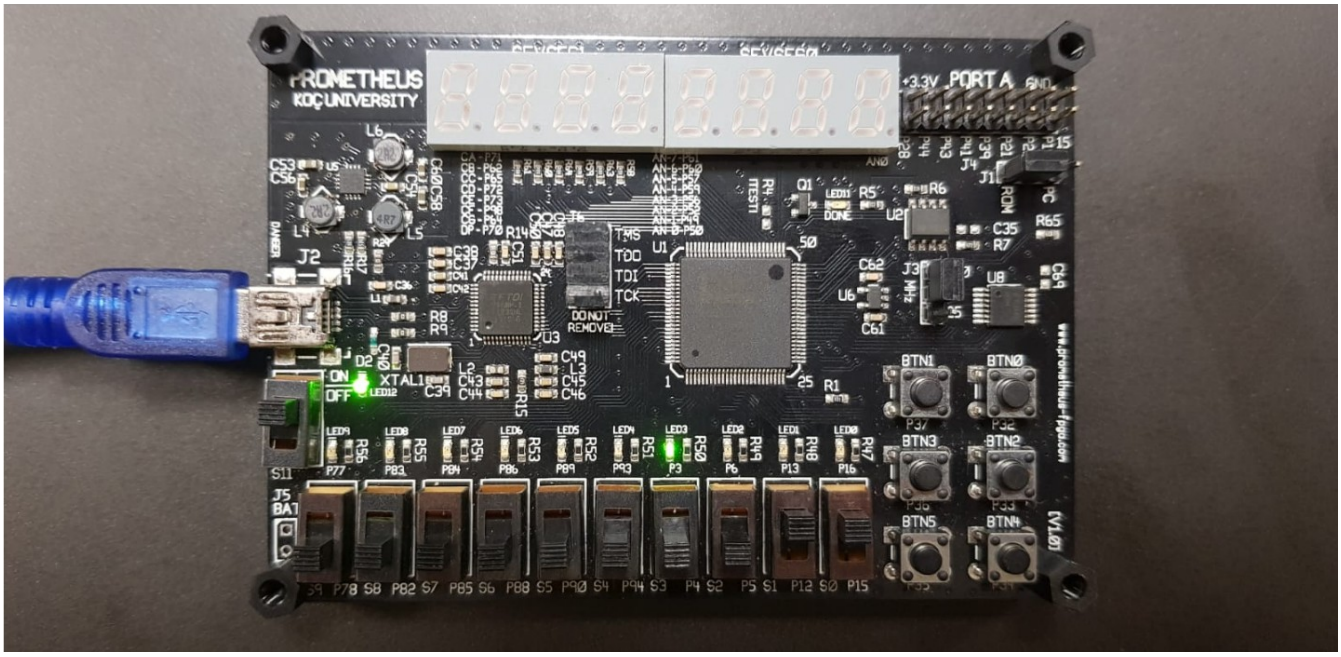
Multiplexer



Demultiplexer



Decoder



Encoder