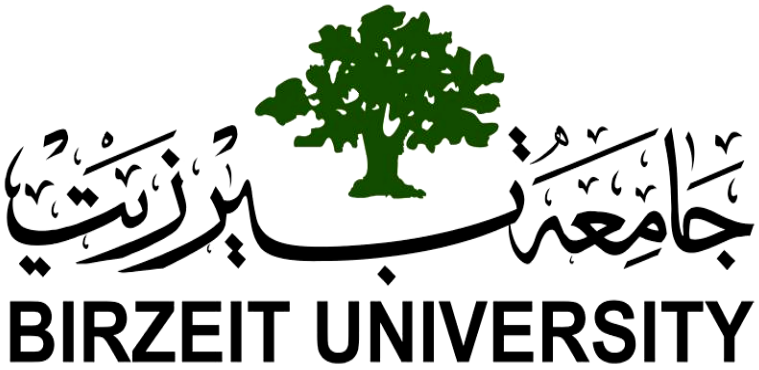



**Faculty of Engineering & Technology**
**Electrical & Computer Engineering Department**

**Digital Electronics and Computer Organization Lab**
**ENCS 2110**

**Report #9**

# A Simple Security System Using FPGA

**Prepared by:**

Maha Maher Mali 1200746

**Partners:**

Lana Thabit 1200071

Jana Herzallah 1201139

**Instructor**: Dr.Anjad Badran
**Assistant**: Eng. Mohammed Bassam

**Section:** 6

**Date: 3/6/2022**

## Abstract

The aim of this experiment is to practice building different digital components using Quartus either by building a Verilog codes or Block diagrams, and learning how to put some of the digital components, you have studied and build in pervious lab sessions, together to build useful systems. Moreover, to become more familiar with FPGA programming.

# Table of Contents

# List of figures

## 2.Theory

In this experiment, we are going to build a simple security system using Altera Quartus software. Then we will program and download this system on the FPGA board.

This security system is simply a 2-digit digital lock. The user enters a number of two digits, such that, the digit ranges from 0 to 3. Thus, every digit has a lower limit of 0 and an upper limit of 3. The number is entered using a keypad (using the 91 switch keys build in our FPGAs). Each digit is represented by a 7- segment display and if the total number entered on the displays equals to XX a green led is on allowing us to pass. Otherwise, a red LED is always on. Figure 1 represent this security system architecture.

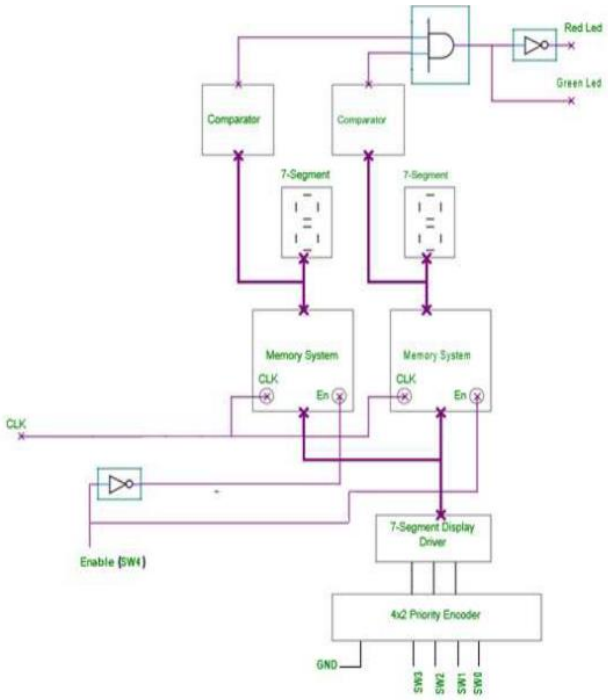



*Figure 1: security system architecture*

## 2.1 4x2 Priority Encoder

The normal digital encoder is a combinational circuit that encodes 2^n input lines by n output lines. In other words, it generates the binary code equivalent of the input line, which is active high. However, this kind of encoders has a problem. It only works when only one of the inputs is active. In other words, if there is more than one input line active, the encoder will generate the wrong code.

This issue can be resolved using the priority encoder. This kind of encoders prioritizes the level of each input. If multiple input lines are active, the output code will correspond to the input line with the highest priority.

The user will use this priority encoder to choose what value to view on a 7-segment display (values range from 0 to 3 in decimal), for example, if the user switches SW1 to high and keeps SW2 and SW3 low then the output of the encoder will be b'01.

## 2.2 Enable Port

The purpose of this port is to allow the user to select which memory system is active, and hence which 7- segment display to use, for example, if SW4 is high then the En pin of the first memory system is enabled and ready to read the user input on the 4x2 priority encoder.

## 2.3 7-segment display driver

This driver is used to convert the output of the priority encoder to the proper input for the 7- segment displays, the output of the driver is first stored in a memory unit before it is transferred to a 7- segment (depends on which memory system is enabled using the 2x4 decoder).

## 2.4 Memory System

The purpose of such system is to ensure that the value selected by the user to display on a certain 7- segment is kept there when the user switches to select another 7-segment.

Each memory system 4 consists of seven D- flip-flops and 2x1 MUXs as shown in Figure 2.

When the enable pin equals 0, the output of each DFF becomes its input at every clock cycle. On the other hand, when the Enable pin becomes 1, the data coming from the 7-

segment driver is stored in the each DFF. The output of each DFF is sent on a data bus to a 7-segment display.

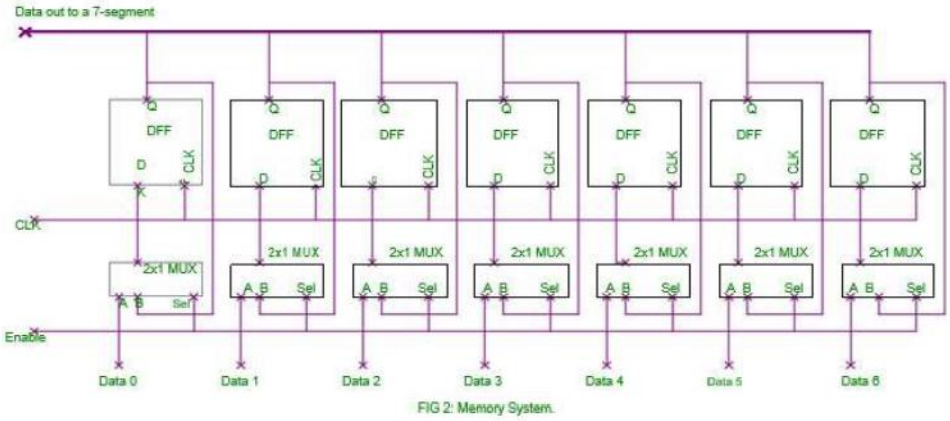



***Figure2:** Memory System*

## 2.5 Comparator

The input of each 7-segment display is connected also to a comparator. every comparator has a built-in value (reference) which is compared with the value of the 7-segment display. If both values are equal, then the output of the comparator is 1, and it is 0, otherwise. For example, if one of the comparators has a reference value equals 5, then its output will be 1 if and only if the input is equal to=7'b0100100 (which is the value of 5 in the 7-segment display). The purpose of the comparator is to lock/unlock the security system.

## 2.6 2-input AND gate

This AND gate will make sure that the two 7-segment displays have the correct combination. In other words, if each comparator output is “1”, then the AND gate output will be “1”, and the green light is ON. Otherwise, the red light will be always ON.

# 3.Procedure

## 3.1 4 x 2 priority encoder

Figure 3 show the code for 4x2 priority encoder, and figure 4 show the wave form for 4x2 priority encoder.

```
//4x2 priority encoder
// Name:Maha Maher Mali
//ID:1200746
module Maha_1200746priority(out,in);
input[3:0]in;
output reg[1:0]out;
always @(in)
begin
    casex (in)
        4'b0001:out=2'b00;
        4'b001x:out=2'b01;
        4'b01xx:out=2'b10;
        4'b1xxx:out=2'b11;
        default:out=2'b00;
    endcase
end
endmodule
```

*Figure 3: code for 4x2 priority encoder*

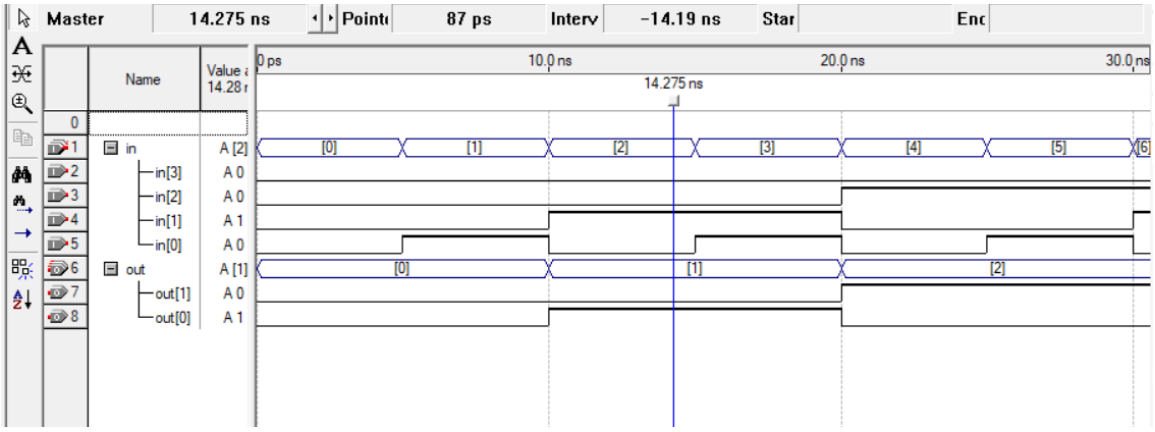



*Figure 4: waveform for 4x2 priority encoder*

## 3.2 7-segment display

Figure 5 show the code for 7-segment display, and figure 6 show the wave form for 7-segment display.

```
// seven segment display driver
// Name:Maha Maher Mali
//ID:1200746
module Maha_1200746seven(out,in);
input[1:0]in;
output reg[6:0]out;
always @(in)
begin
    casex (in)
        0:out=7'b0000001;
        1:out=7'b1001111;
        2:out=7'b0010010;
        3:out=7'b0000110;

    endcase
end
endmodule
```

*Figure 5: code for 7-segment display*

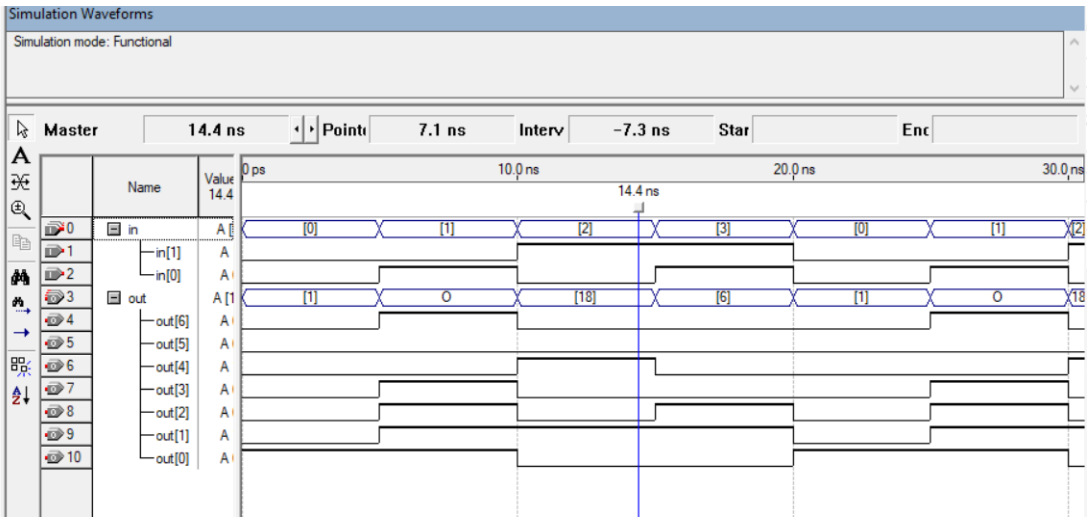



*Figure* 6: wave form for 7-segment display.

## 3.3 D- Flip Flop

Figure 7 show the code for D- Flip Flop, and figure 8 show the wave form for D- Flip Flop.

```
// Name:Maha Maher Mali
//ID:1200746

module Maha_1200746dff(Q,D,CLK);
output reg Q;
input D,CLK;
always @ (posedge CLK)
    begin
            Q=D;
        end
endmodule
```

*Figure7: code for D- Flip Flop*

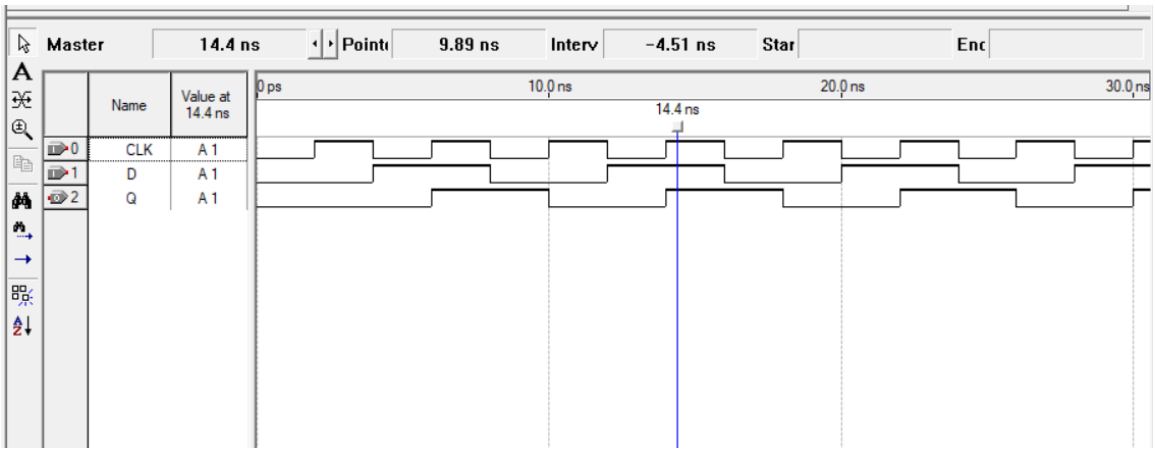



*Figure8: wave form for D- Flip Flop*

## 3.4 2x1 MUX

Figure 9 show the code for 2x1 MUX, and figure 10 show the wave form for 2x1 MUX.

```
// Name:Maha Maher Mali
//ID:1200746
module Maha_1200746mux(X,Y,S,OUT);
input S;
input X,Y;
output OUT;
reg OUT;
always @ (X or Y or S)
begin
    if(S==0)
        OUT=X;
    else
        OUT=Y;
    end
endmodule
```

*Figure9: code for 2x1 MUX*

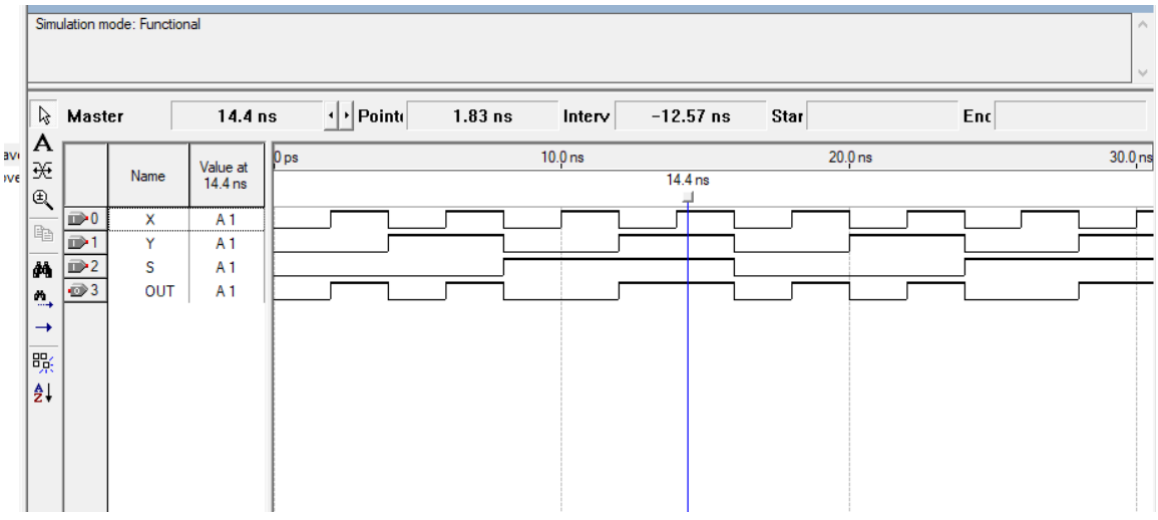



*Figure10: wave form for 2x1 MUX*

## 3.5 comparator

Figure 11 show the code for comparator, and figure 10 show the wave form for comparator.

```
// Name:Maha Maher Mali
//ID:1200746
module comparator(out,in);
input [6:0]in;
output reg out;
always@(in)
    begin
        if(in==7'b0100100)
            out=1'b0;
        else
            out=1'b0;
        end
endmodule
```

*Figure11: code for comparator*

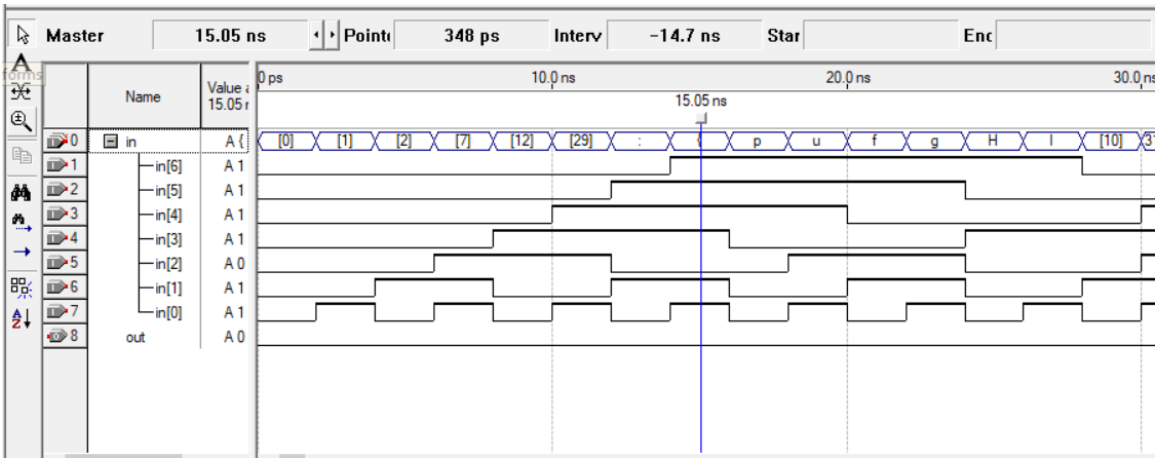
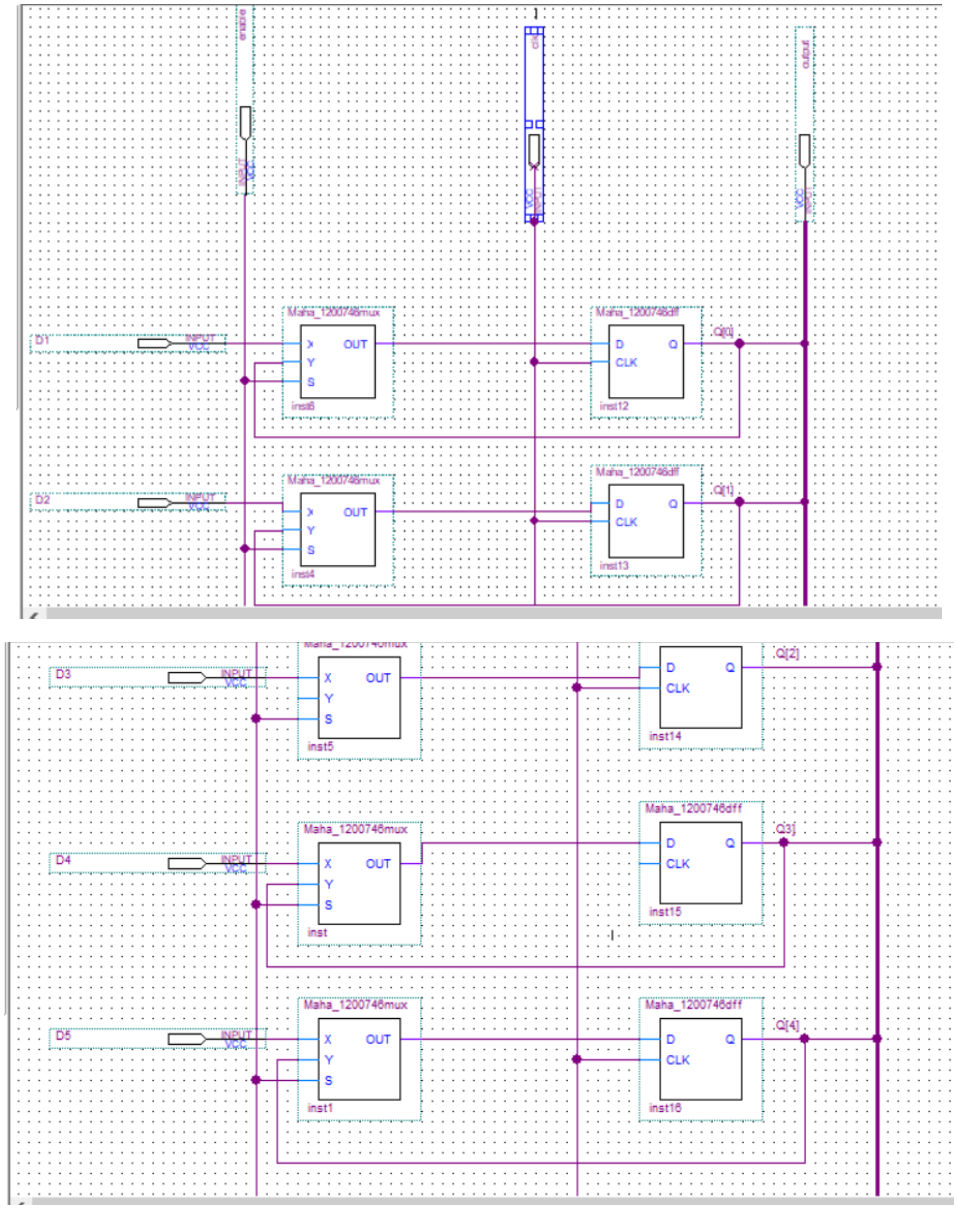



*Figure12: wave form for comparator*

## 3.6 Memory system Block Diagram

Figure 13 show the block diagram for the system.



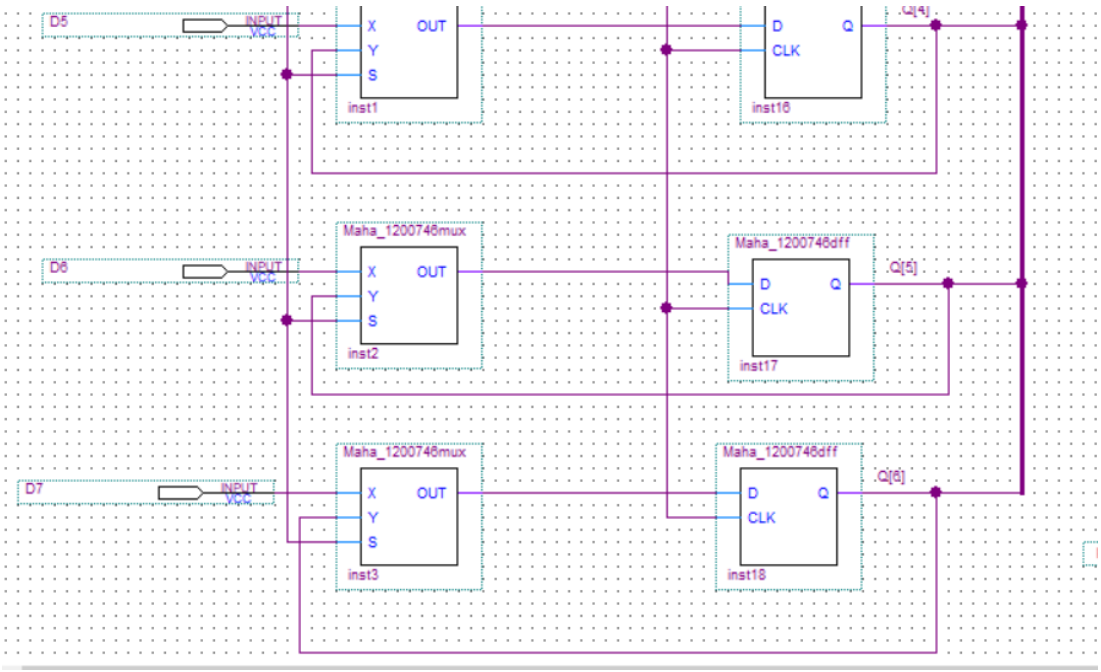



*Figure13:system block diagram*

## 3.7 The security system final block diagram

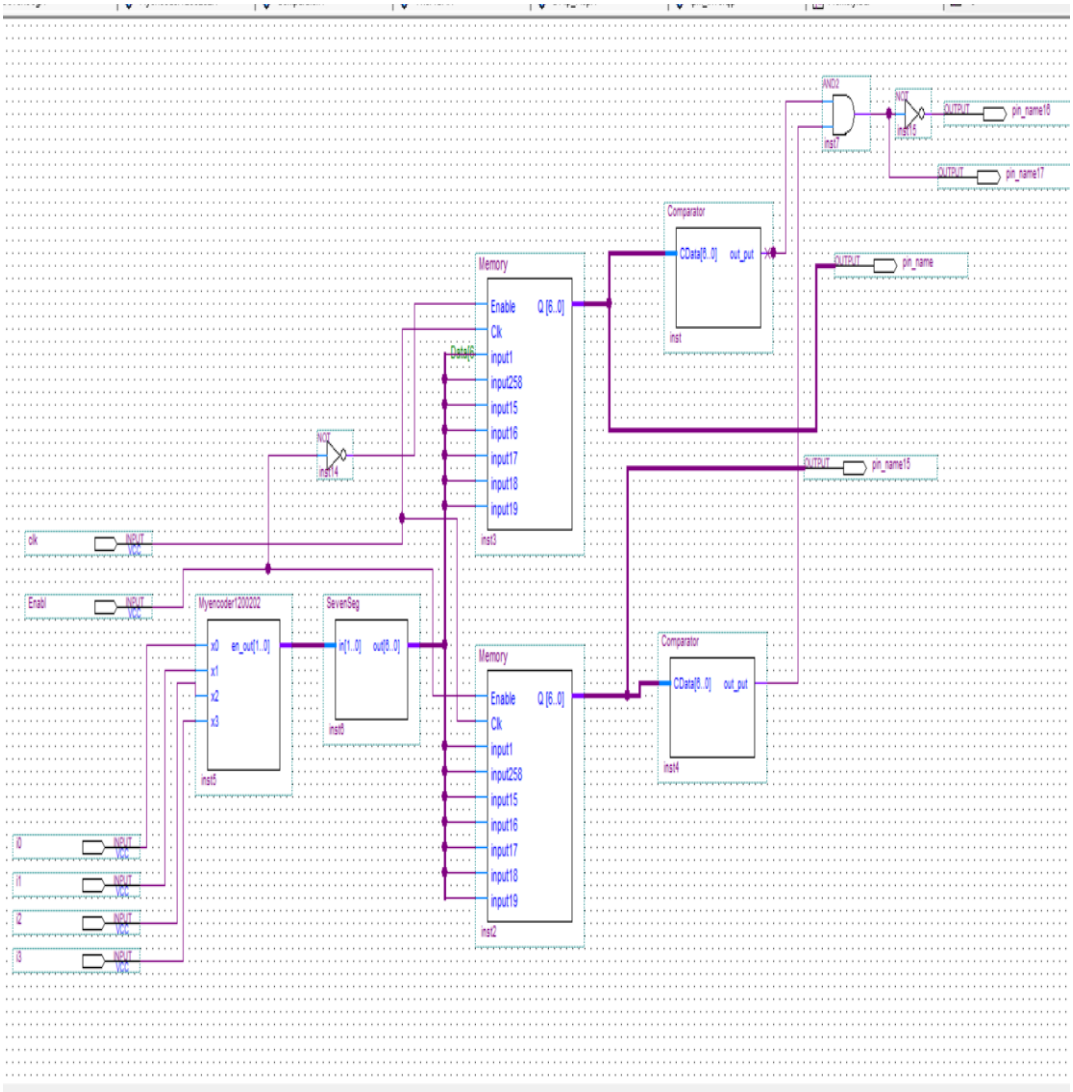



*Figure14: The security system*

# Conclusion

In conclusion, we understood the in this experiment we learn how to build a security system by using digital systems and memory. Also, how to download the system to the FPGA Board, and to putting everything together. In addition to that, we became able to deal with Verilog code using Quartus.

# References

[1] https://technobyte.org/verilog-multiplexer-2x1 . Accessed on 3-06-2022 at 1:45PM.

[2] https://www.geeksforgeeks.org/verilog-priority-encoder . Accessed on 3-06-2022 at 2:17PM.

[3] https://www.rfwireless-world.com/source-code/VERILOG/1-bit-comparator-4-bit-comparator-verilog-code.html . Accessed on 3-06-2022 at 3:20PM.