

**Faculty of Engineering and Technology**

**Department of Electrical and Computer Engineering**

**ENCS 211**

**Digital Electronics and Computer Organization Lab**

**Experiment No. 9**

**A Simple Security System Using FPGA**

1. **Objectives:** 
   * To gain more practice in building different digital components using Quartus either by building Verilog codes or Block diagrams.
   * To learn how to put together some of the digital components you’ve studied and built in pervious lab sessions, to build useful systems.
   * To become more familiar with FPGA programming.
2. **Apparatus:** 
   * A computer with Quartus II (7.2 +) and USB driver installed.
   * Altera DE1 system with its datasheets. (For FPGA pins map).
3. **Pre Lab: (Bring a soft copy of your pre lab with you to the lab)** 
   * Prepare each part of the procedure section where it says (Pre Lab).
   * **NOTE**: It’s important that you come prepared, as this will reflect your work time during the lab plus it will be a critical variable in the evaluation of your lab report.
4. **Theory:**

In this experiment we are going to build a simple security system using Altera Quartus software, then we will program and download our system to DE1 Board (FPGA board).

Our security system is simply a 4 digit digital lock. The user enter a number of 4 digits (digit range: 0 to 6, so every digit will has a lower limit of 0 and an upper limit of 6) using a keypad (using the six switch keys built in our DE1 FPGA kits). Each digit is represented by a 7-segment display and if the number entered on the displays equals to XXXX a green led is on; allowing us to pass. Else a red led is always on; blocking us from passing.

The architecture of our system is shown in Figure 1.

As we can see in Figure 1, the system consists of the following components:

**1) 8x3 Priority Encoder:**

The user will use this priority encoder to choose what value to view on a 7-segment display (values range from 0 to 6 in decimal) , for example if the user switch SW4 to high and keep SW5 and SW6 low then the output of the encoder will be b’100.

**2) 2x4 Decoder:**

The purpose of this decoder is to let the user select which memory system is active thus which 7-segment display to use, for example if SW8 and SW7 are high then the Enable (En) pin of the 4th memory system is enabled and ready to read user input on the 8x3 priority encoder.

**Note**: The Enable (En) pin of the decoder must be active low while switching between selection lines of the decoder.

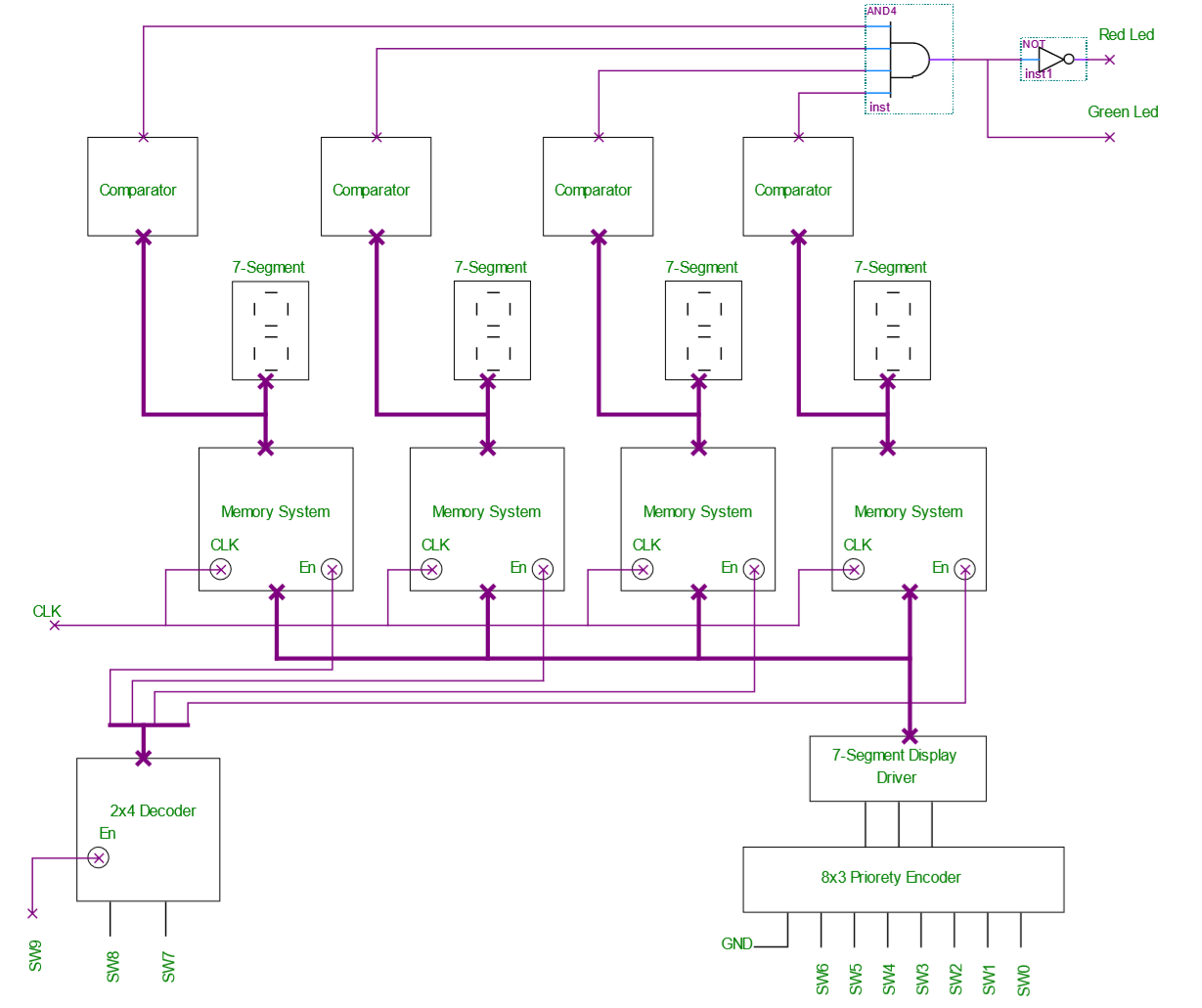


Figure 1: Overall System Design

**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 transferring to a 7- segment (depends on which memory system is enabled using the 2x4 decoder).

**4) Memory System:**

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

Each memory system is consisting of seven D- flip flops and 2x1 MUXs as seen in Figure 2.

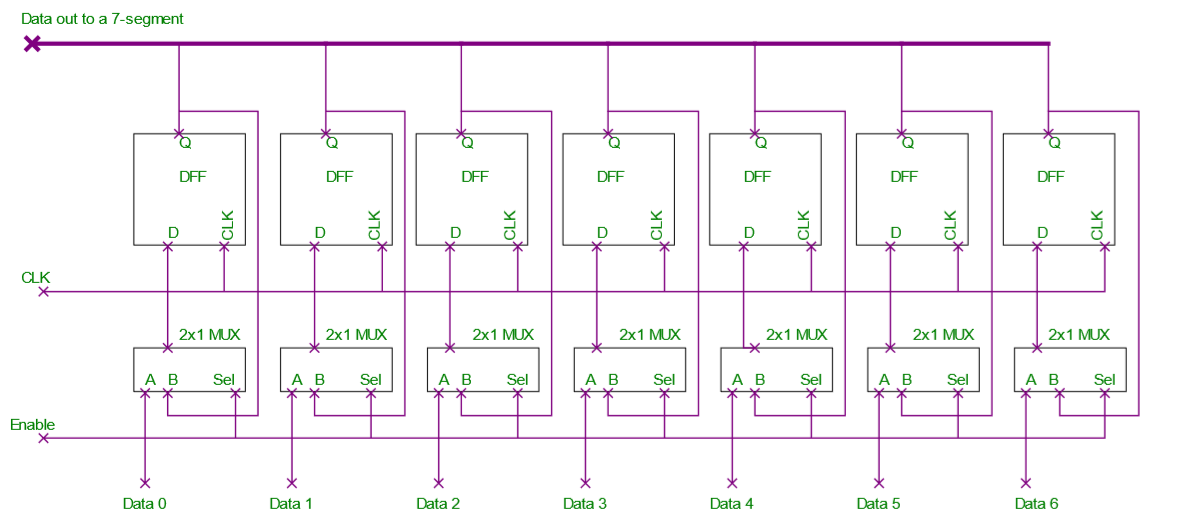


Figure 2: Memory System

When the Enable pin =0, the output of each DFF becomes its input at every clock cycle, when the Enable pin becomes 1 the data coming from the 7-segment driver is then stored in the each DFF. The output of each DFF is sent as a data bus to a 7-segment display.

**Note**: For each 7-segment display we need a memory system block.

**5) Comparator:**

The input of each 7-segment display is connected also to a comparator, every comparator has a build in value (reference) which is compared with the value of the 7-segment display , if both are equal then the output of the comparator is 1 else the output will be 0 ; for example if one of the comparators has a reference value = 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 7-segment display). The purpose of the comparator is to lock/unlock our security system.

**6) 4-input AND gate:**

This AND gate will make sure that all 4 7-segment displays have the correct combination ; if each comparator output = 1 , then the AND gate output will be 1 , thus a green light is on, else a red light will be always on.

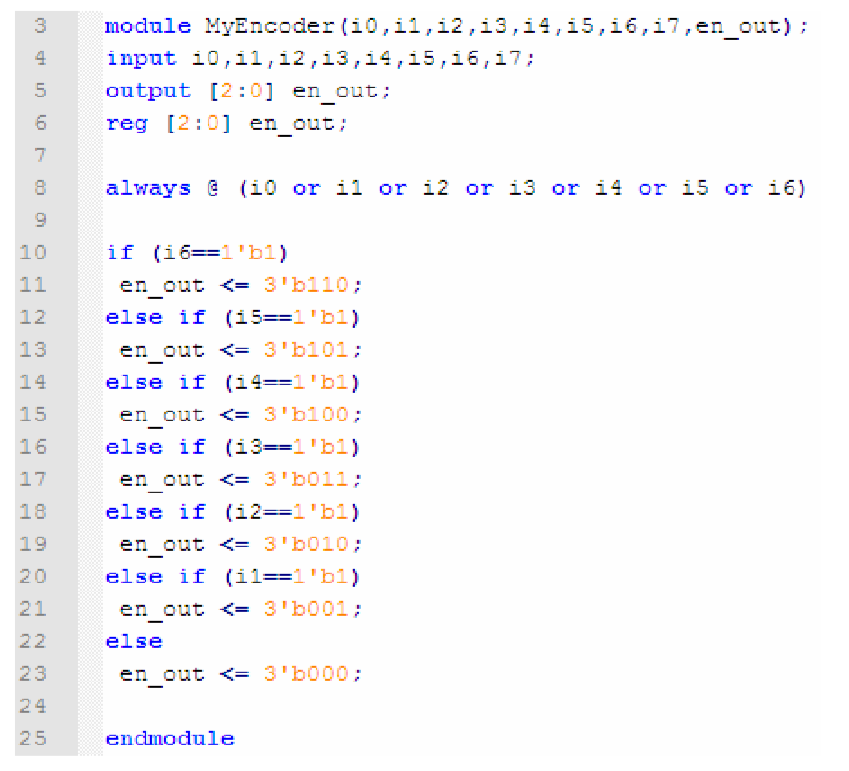
1. **Procedure:**

After we understood the architecture of our security system it’s time to start programming and designing it.

**NOTE: *Create a Symbol for each component you build.***

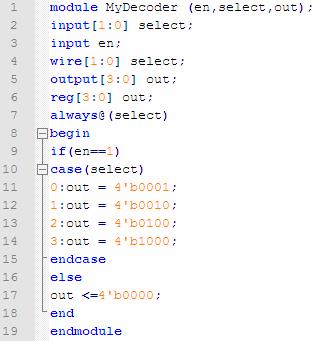
* 1. **8x3 priority encoder.**

Write down the following code, compile and simulate it **(Pre Lab).**



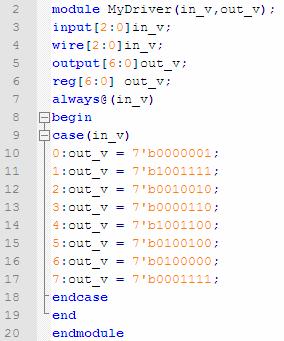
1. **2x4 Decoder:**

Write down the following code, compile and simulate it **(Pre Lab).**



1. **7-segement driver:**

Write down the following code, compile and simulate it **(Pre Lab).**



1. **Memory System:**

1- Write the code of a D- Flip Flop, compile and simulate it.

2- Write the code of a 2x1 MUX, compile and simulate it **(Pre Lab).**

**Note: *It’s important that your MUX behaves as explained in the memory system* *section (check back the theory).***

3- Use a block diagram to build the following design:

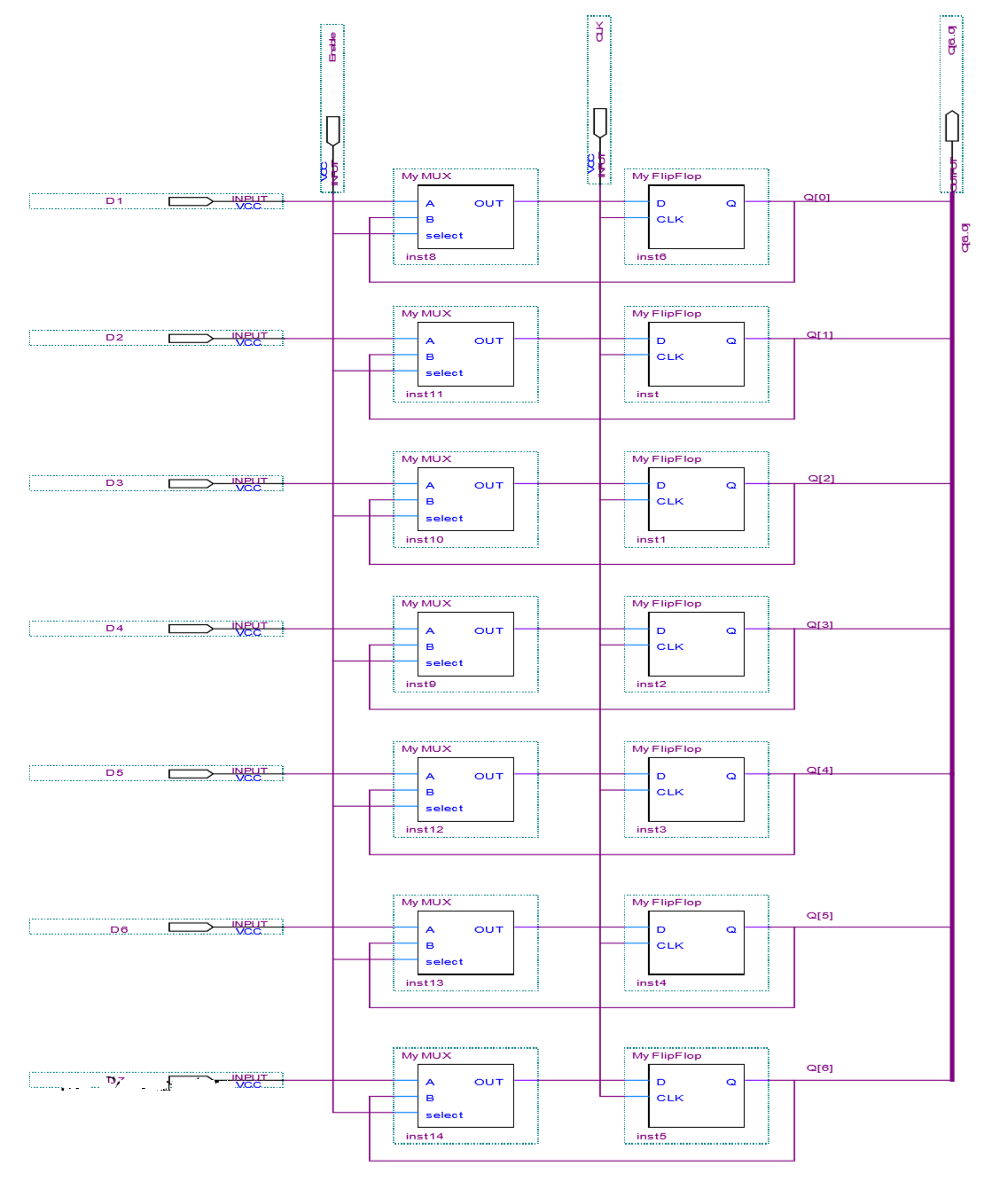
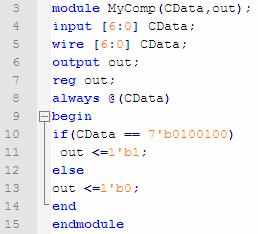


Figure 3: Memory System

1. **Comparator:**

Write down the following code, compile and simulate it **(Pre Lab).**



**Note: *For simplifying reasons we will build one comparator based on a reference* *value X (in this example X = 5), you can build four different comparator with four different values to compare with.***

1. **Putting everything together:**

Build and design the security system using the components you build during the previous sections.

The final block design should look like the schematic on the next page.

1. **Assign pin values to the security system design you just built and then download the system to the FPGA board.**

