* **Introduction :**

The Quartus II program allows us to design a complete digital circuit using schematic diagrams and it’s plugs, where we can simply place gates and ICs and specify their inputs and outputs, just like what we can do with a paper and a pen, and those IC’s can be not pre-installed by software, i.e. we can create our own new ones, by schematic design, or by using Verilog HDL design.

Another use of QuratusII that we can simulate and test our circuit designs, i.e. we can test our design’s response for different stimuli. To illustrate that we randomly assigned arbitrary values to each input (pin) every half interval – every 5.0 ns.

Another feature which the Quartus II program is primarily used for is that we are able to configure FPGA’s (Field Programmable Gate Array) with our designs. So we chose to download our design on Cyclone II FPGA by using programmer software that is included in Quartz II. Note that the downloading procedure is simply made using a standard USB-A which is connected to the PC computer at the lab room to USB-B cable which is connected to FPGA.

A Field Programmable Gate Array (FPGA) is an [integrated circuit](http://en.wikipedia.org/wiki/Integrated_circuit) designed to be configured by the customer or designer after manufacturing—hence "[field-programmable](http://en.wikipedia.org/wiki/Field-programmable)". The FPGA configuration is generally specified using a [hardware description language](http://en.wikipedia.org/wiki/Hardware_description_language) (HDL), FPGAs can be used to implement any logical function. FPGAs contain [programmable logic](http://en.wikipedia.org/wiki/Programmable_logic_device) components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"—somewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations. Logic blocks can be configured to perform complex [combinational functions](http://en.wikipedia.org/wiki/Combinational_logic), or merely simple [logic gates](http://en.wikipedia.org/wiki/Logic_gate) like [AND](http://en.wikipedia.org/wiki/AND_gate) and [XOR](http://en.wikipedia.org/wiki/XOR_gate). In most FPGAs, the logic blocks also include memory elements, which may be simple [flip-flops](http://en.wikipedia.org/wiki/Flip-flop_(electronics)) or more complete blocks of memory.

* **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, 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 build in our FPGAs) . Each digit is represented by a 7-segment display and if the total 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 circuit shown next page is the architecture of our system:



The circuit shown consists of the following components

1. **2x4 decoder:**

The decoder is used to let the user select which memory system is active thus

Which 7-segment display to use. Fig (a) shows the output of the 2x4 Decoder.

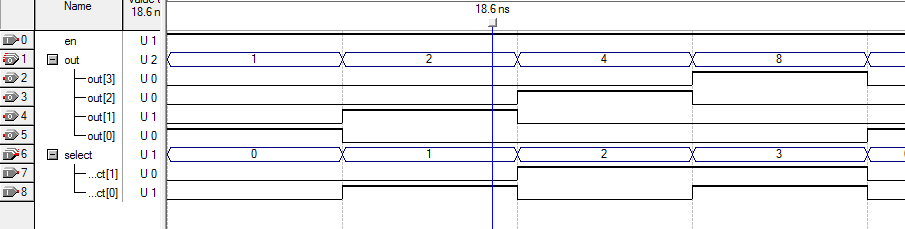


Fig (a)

1. **8x3 priority encoder:**

The priority encoder is used by the user to choose what value to view on a 7-segment display (values range from 0 to 6 in decimal) .Fig (b) next page shows the output of the priority encoder

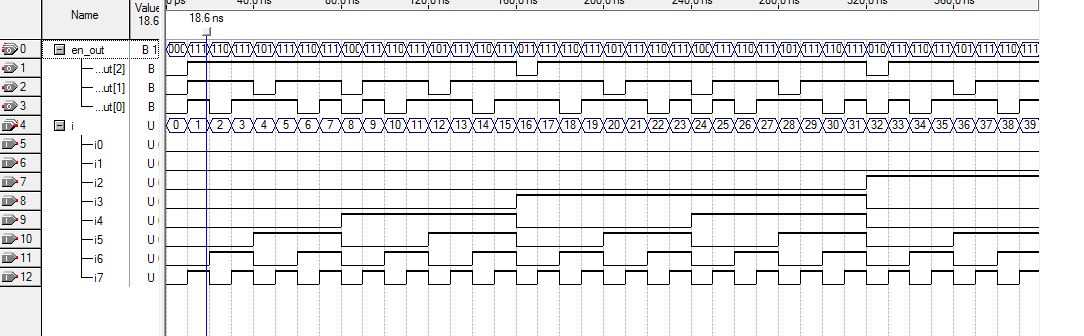
**

Fig (b)

1. **7-Segement 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).fig (c) shows the output of the seven segment driver.

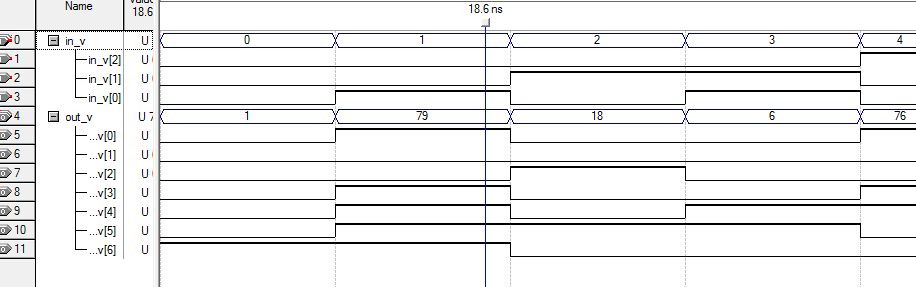


Fig (c)

1. **Memory System:**

The memory system is used to keep the selected value by the user to select another 7-segment. Each memory system is consisting of seven D- flip flops and 2x1 MUXs as seen in the following figure fig (d).

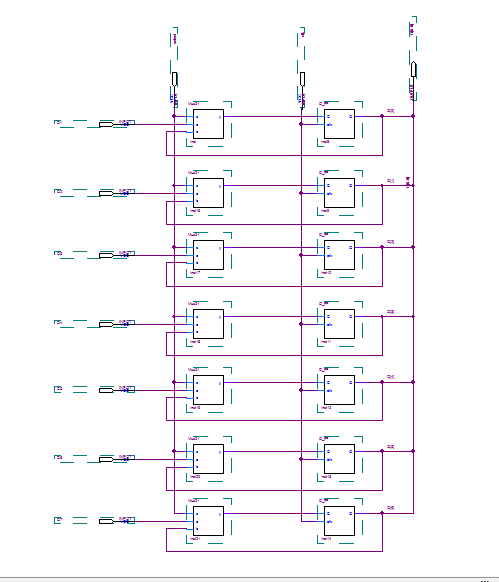


Fig (d)

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

1. **Comparator:**

The purpose of the comparator is to lock/unlock our security system.

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.

We wrote in the code of the comparator the reference we want:

*module Mycomp(Cdata,out);*

*input [6:0] Cdata;*

*wire [6:0] Cdata;*

*output out;*

*reg out;*

*always @(Cdata)*

*begin*

*if (Cdata == 7'b0100100)*

*out<=1'b1;*

*else*

*out <=1'b0;*

*end*

*endmodule*

1. **4-input AND gate:**

The 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,

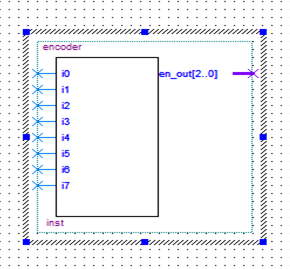
Else a red light will be always on.

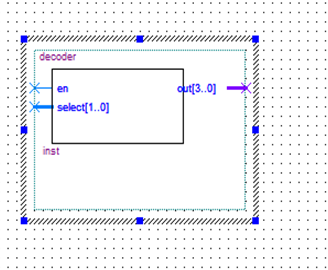
* **Procedure:**

We have made a block diagram for each component and then we have connected the components in such a way to get the system we want.

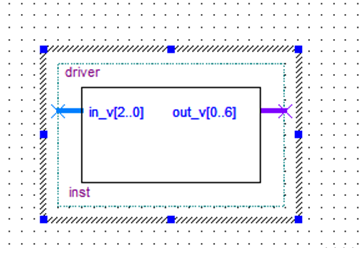
The block diagram for each component next page:

**2x4 Decoder**

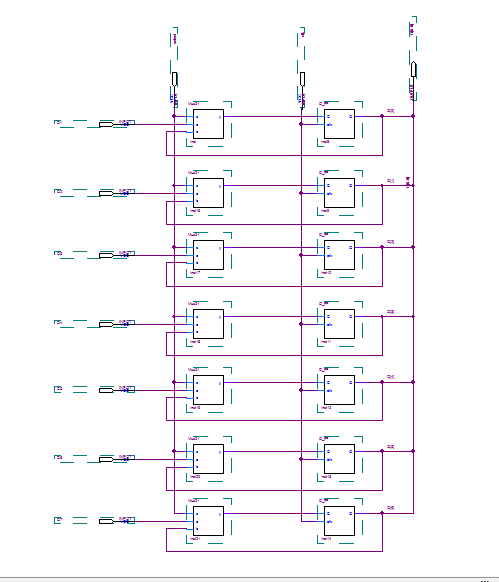
**

**8x3 Priority Encoder** **

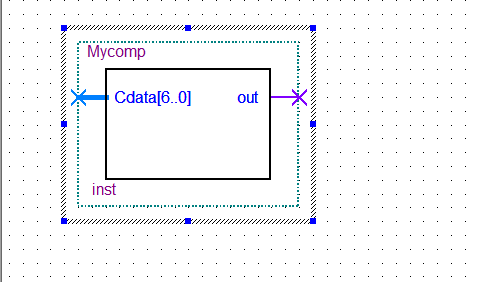
**7-segement driver**

**

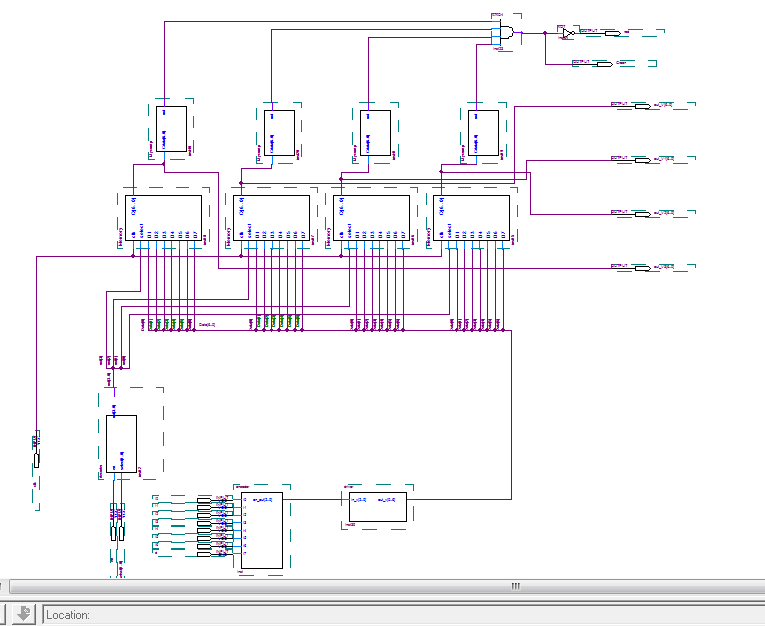
**Memory system**

**

**Comparator**

**

**The final design**

**

After connecting everything we assigned pins values to the security system design and then download the system to the FPGA board.

* **Conclusion:**

We noticed that dealing with software is much better than hardware , since we done some changes of our circuit by changing the software but if we use the hardware we must doing a big changes in the hardware design. We leant how to put some of the digital components to build useful systems.

* **References:**

* Pictures from other reports.
* Our experiment Pdf sheet.