Bir zeit University

Faculty of Information Technology

Computer Systems Engineering Department

Digital Lab ENCS 211 EXP. No. 7

Introduction to QUARTUSII Software

7.1 Objective:

Quartus II software is dedicated to program PLDs or FPGAs, in this experiment; we will study the schematic capture and HDL on Quartus II and then download simple examples on the FPGA using the Kit DE1.

7.2 Pre-lab:

Read the experiment before the lab,

1-Using Quartus II, create a schematic file for the function F=AB+C'D and simulate it.

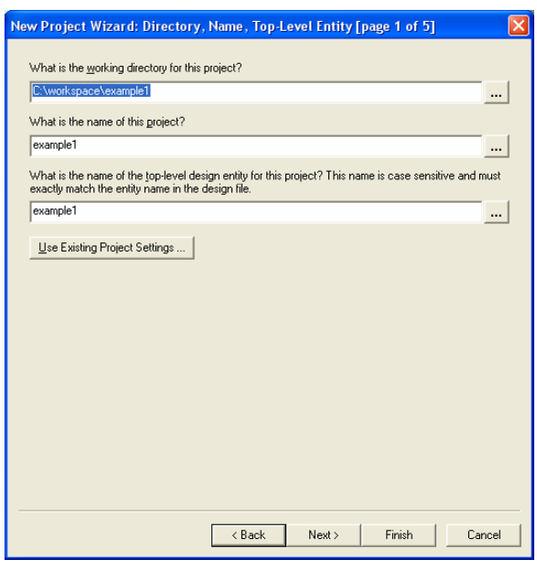
2-Using Quartus II, create Verilog file for full Adder and simulate it.

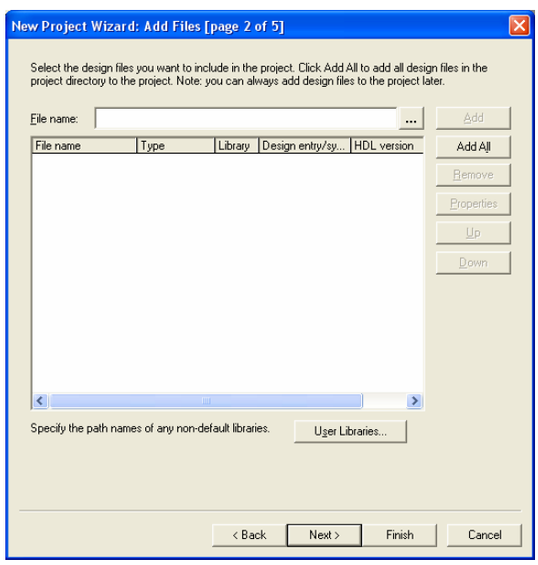
7.3 Procedure:

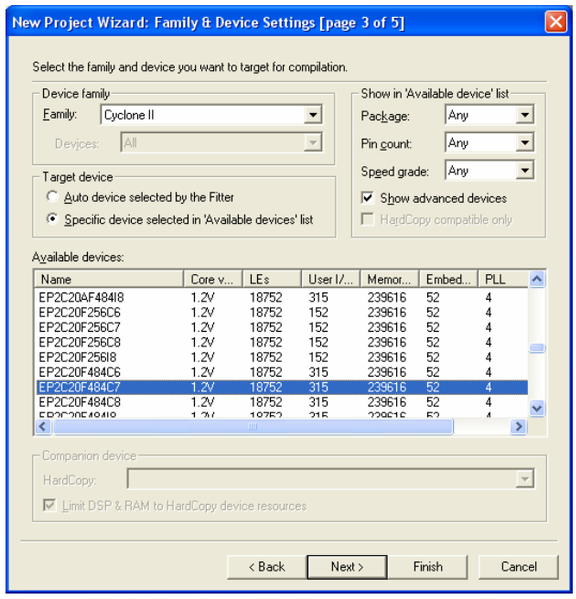
Part 1: schematic capture:

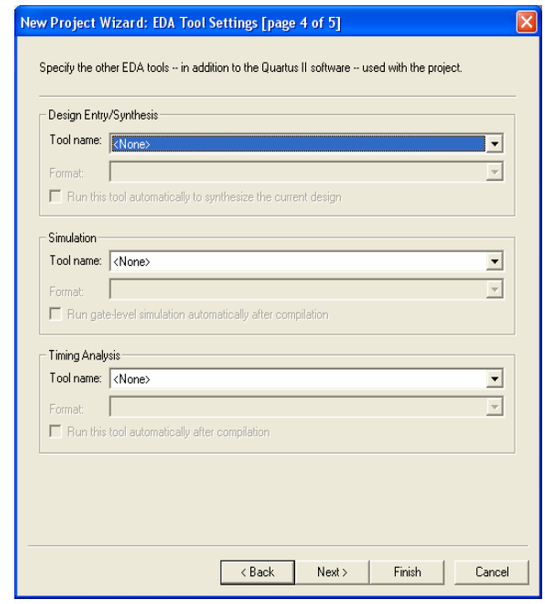
The aim of this part is to design the combinational function: F=ABC+A'D

- Run the QUARTUSII software: double click on the QUARTUS II item in the desktop.
- Define the project : file→new project wizard→then press next
- Follow the windows as shown below:

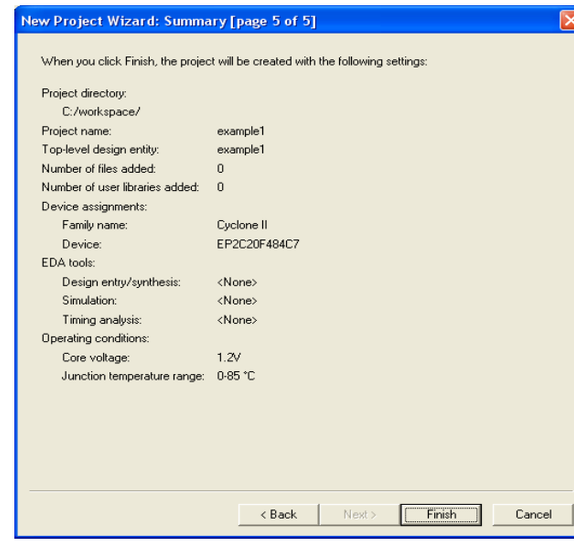








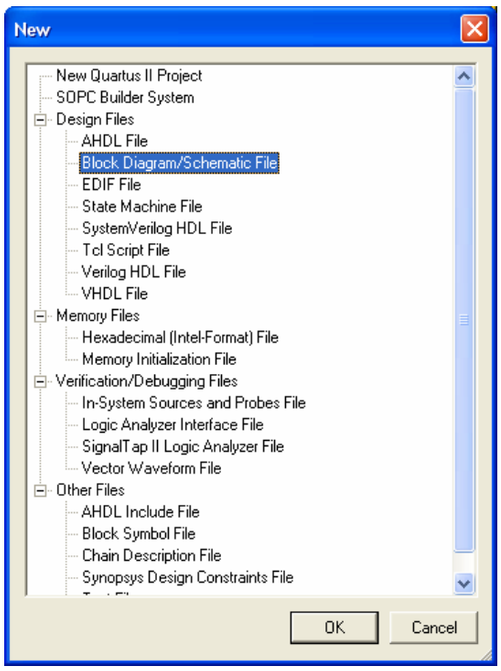
- Change the working directory to indicate your folder that will contain your project.
- Then name this project and the top level design entity, you must notice that the name of the project and the high level design entity must have the same name as the module in your program (in the case of HDL programming), then press next.
- If you have already files to add, you can add them here, otherwise (if you want to build the design now) press next.
- The next window give us a chance to use a specific device (which we will use later), choose the Family cyclone II and the device EP2C20F484C7
- Since our project is simple, and no need to add any other Tools, we just press next.
- After that the following window appears:



- Finally press finish.

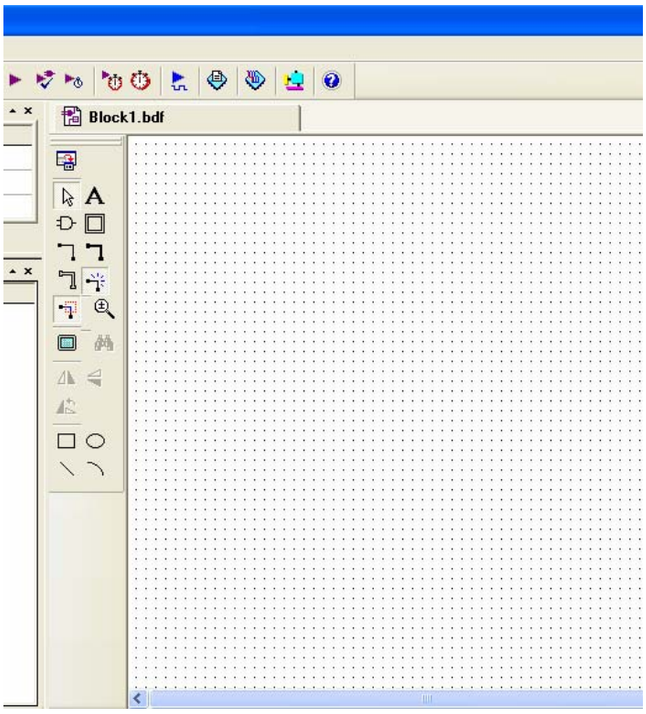
Now we will build our function a schematic file:

File → new

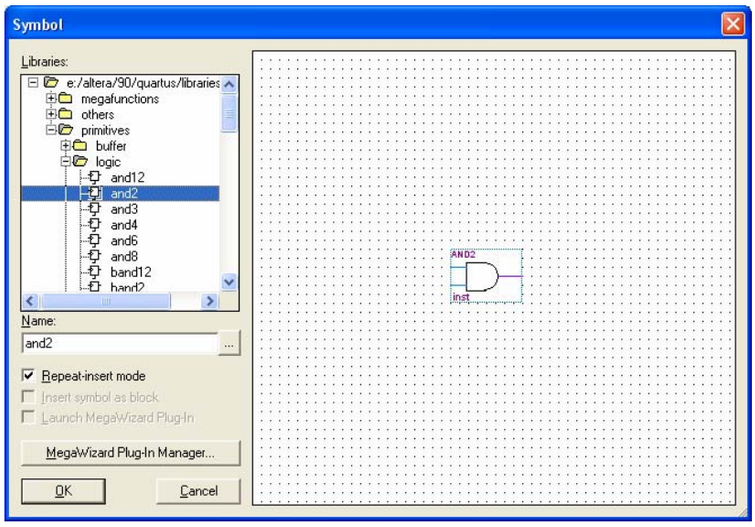


To enter components of our design double click anywhere on the schematic window, or select the symbol tool

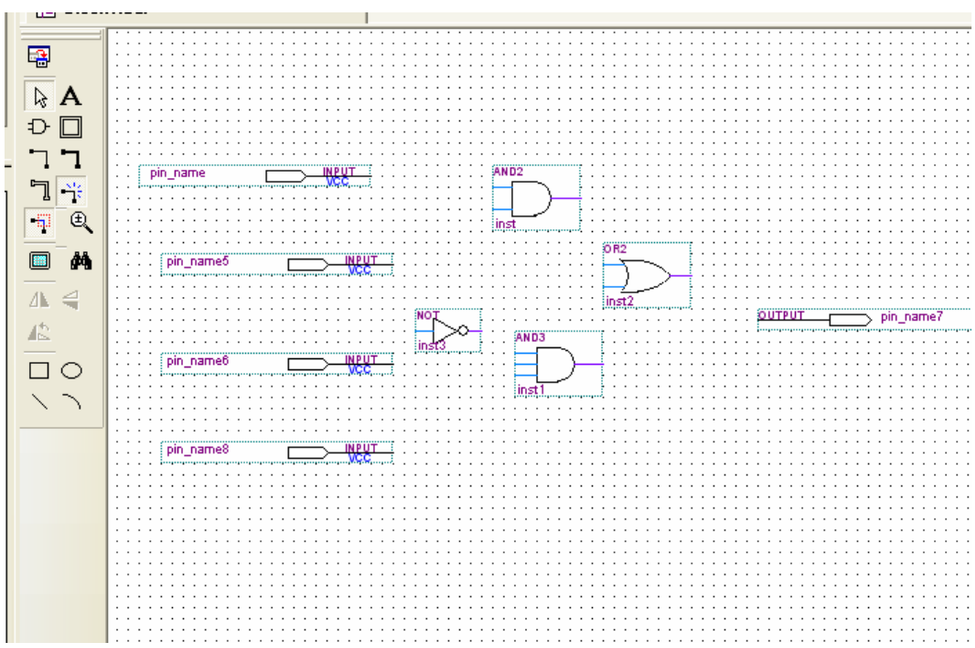
(The little and gate) as shown in the figure:



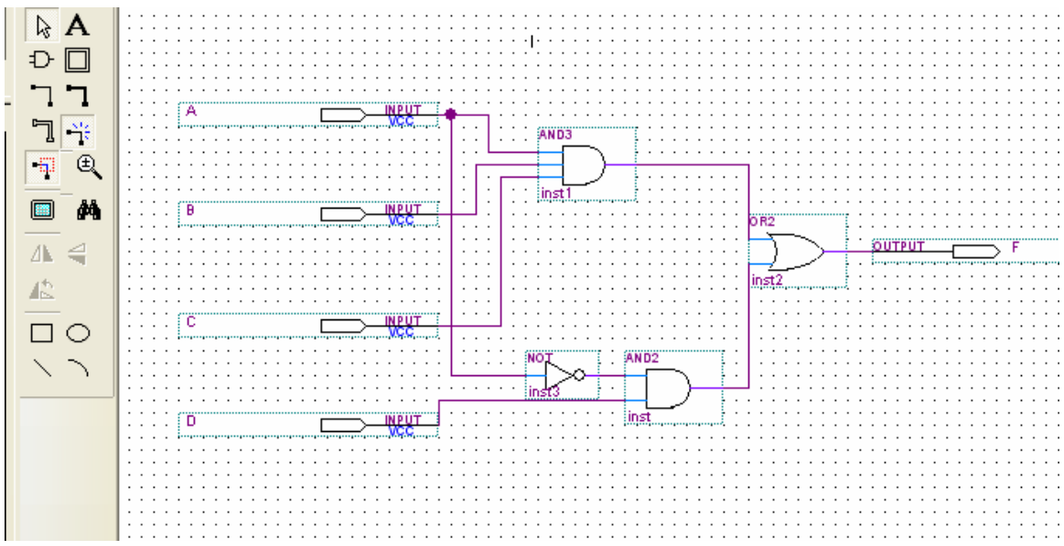
This opens the symbol window in which available libraries ,including the standard QUARTUSII library, open this library by clicking on the little plus sign next to it , then select primitives , and then select logic ,then select the gate you want in your implementation



You have to define the inputs and outputs of your implementation, and you do so by opening the symbol window→primitive→pin, the following figure shows all the design components that must now be connected:



- To connect the components of the previous figure , select the 90^{o} thin line with the dots on its ends on the tool bar, this makes your cursor a wiring tool that can be used to connect you circuit. When done, disable the wiring tool by clicking the arrow on the tool bar.
- Rename input and output ports to the variable names of our design ,to name a pin, either double click it to open its pin properties window, or right –click it , and select properties from the pull-down menu that shows up.
- Save your design , and make sure it is named the same as your project, the following figure shows the completed block diagram of our design.



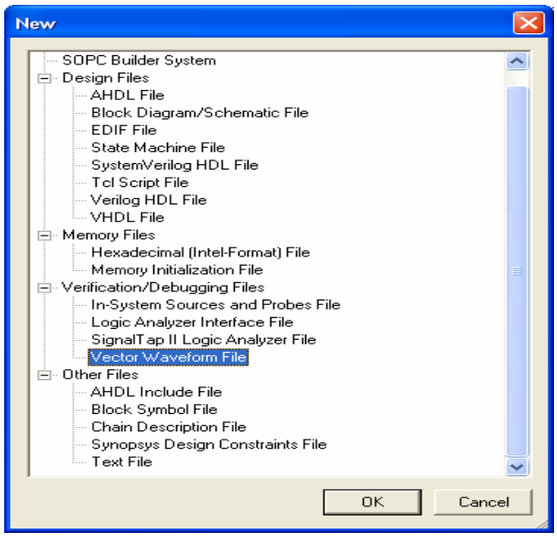
Compilation

After that we compile our design by either clicking on processing→start compilation or click on  .

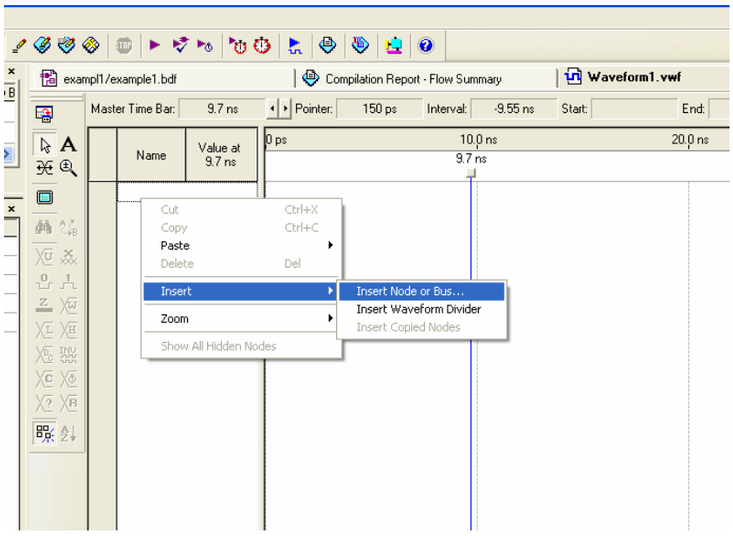
If there are errors fix them before proceeding.

Simulation

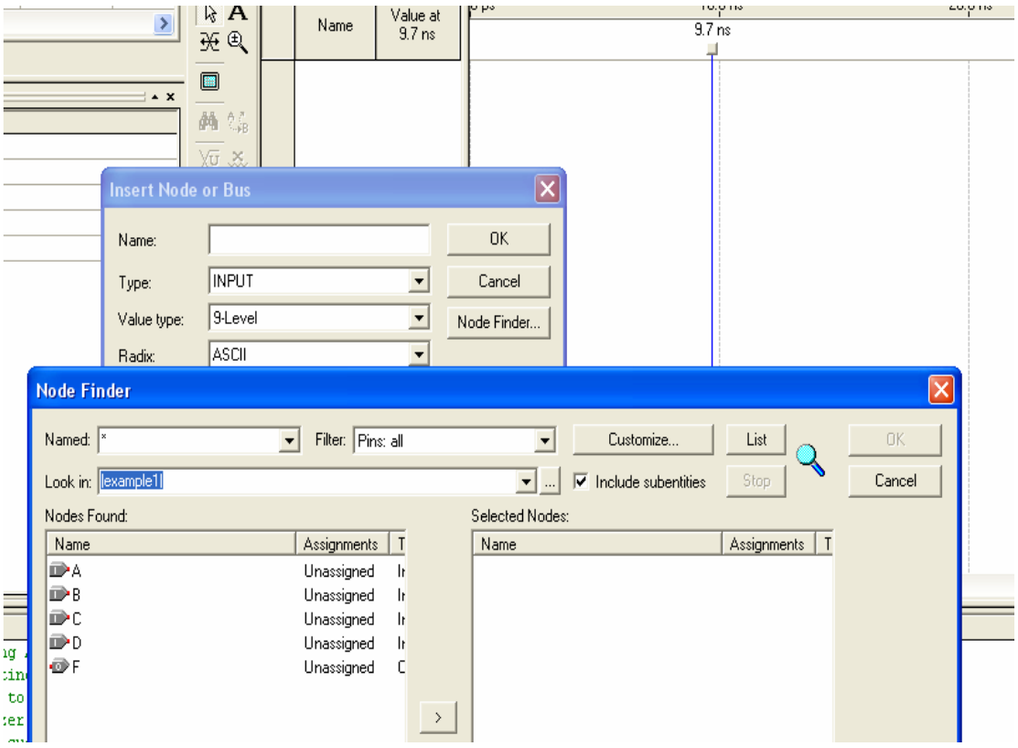
If there are no errors, File → new , select Vector waveform File



Then right-click on the leftmost side of the window (under Name), then click insert node or bus, as shown below



Click on **Node Finder** and then on List (using the Filter pins: all)



- Select all inputs and the output by clicking on (>>) (you can select one by one).
- You can select the intervals when you want the inputs to be one or zero , either by shadowing the interval ,then press the one level in the tool bar, or by write clicking the name , then select value→count value, the change the start value, the end value , and the radix as shown in the following figure:
- Now save the file with the same name as your project and in the same folder.
- Simulate your file either by clicking processes→start simulation or click on. 

Routing the Project on the FPGA:

After verifying that the design works as expected we can install the cod on the hardware (FPGA).

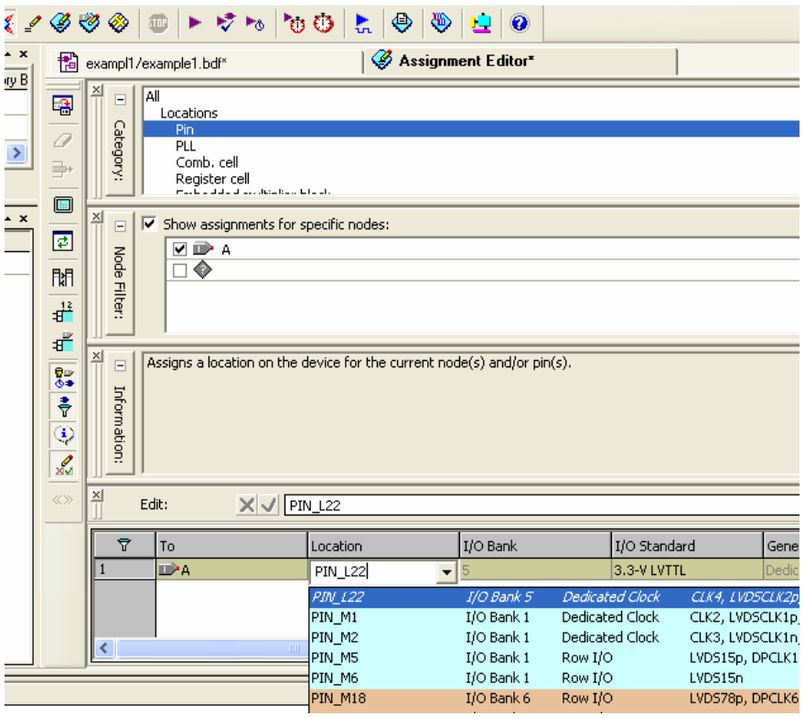
- Double check that the selected device is the one available on the Kit DE1. This can be done using Assignments→ Device
- We need 4 Inputs and one output , so we can use 4 switches and a LED . We have to use the user manual of DE1 to figure out the location of the switches and the LEDs . the Table below show the location of the switches

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_L22	Toggle Switch[0]
SW[1]	PIN_L21	Toggle Switch[1]
SW[2]	PIN_M22	Toggle Switch[2]
SW[3]	PIN_V12	Toggle Switch[3]
SW[4]	PIN_W12	Toggle Switch[4]
SW[5]	PIN_U12	Toggle Switch[5]
SW[6]	PIN_U11	Toggle Switch[6]
SW[7]	PIN_M2	Toggle Switch[7]

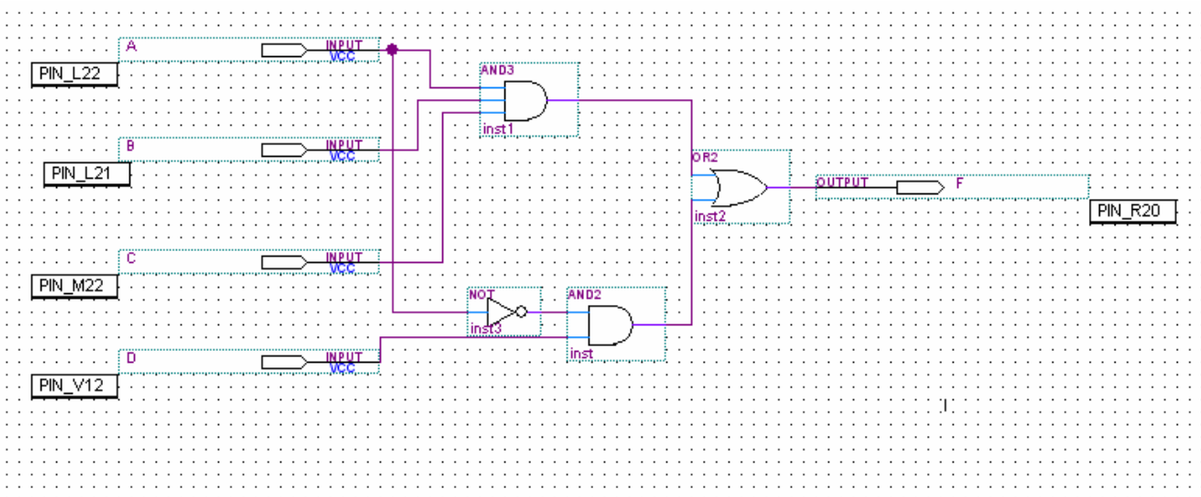
Assign PINs for the Inputs and Outputs. You can assign by right click the pin then choose

- Locate→locate in assignment editor

- Select Pin in the category and select a switch for input.

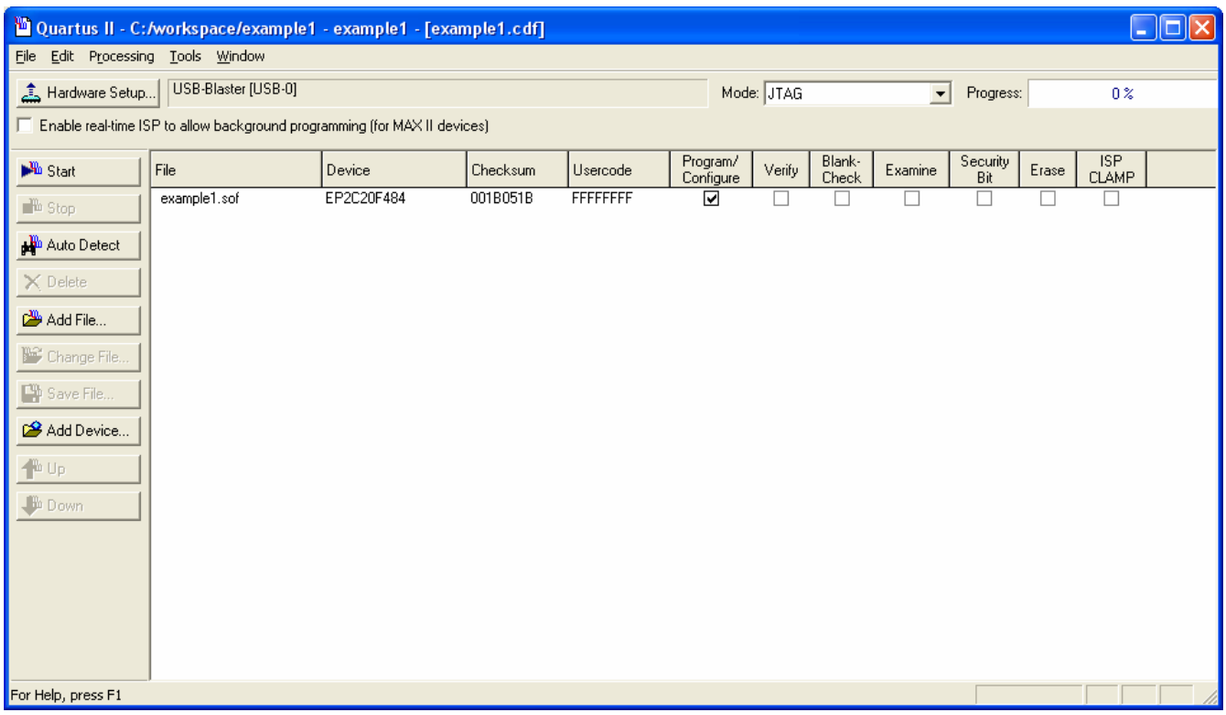


The Figure below shows the circuit with PIN assignments.



Re-compile the Project so that the new changes in the PINs take place.

Download the Program on the FPGA Tools→ Programmer



Make sure that the currently selected hardware is USB-Blaster, and then click **Start.**

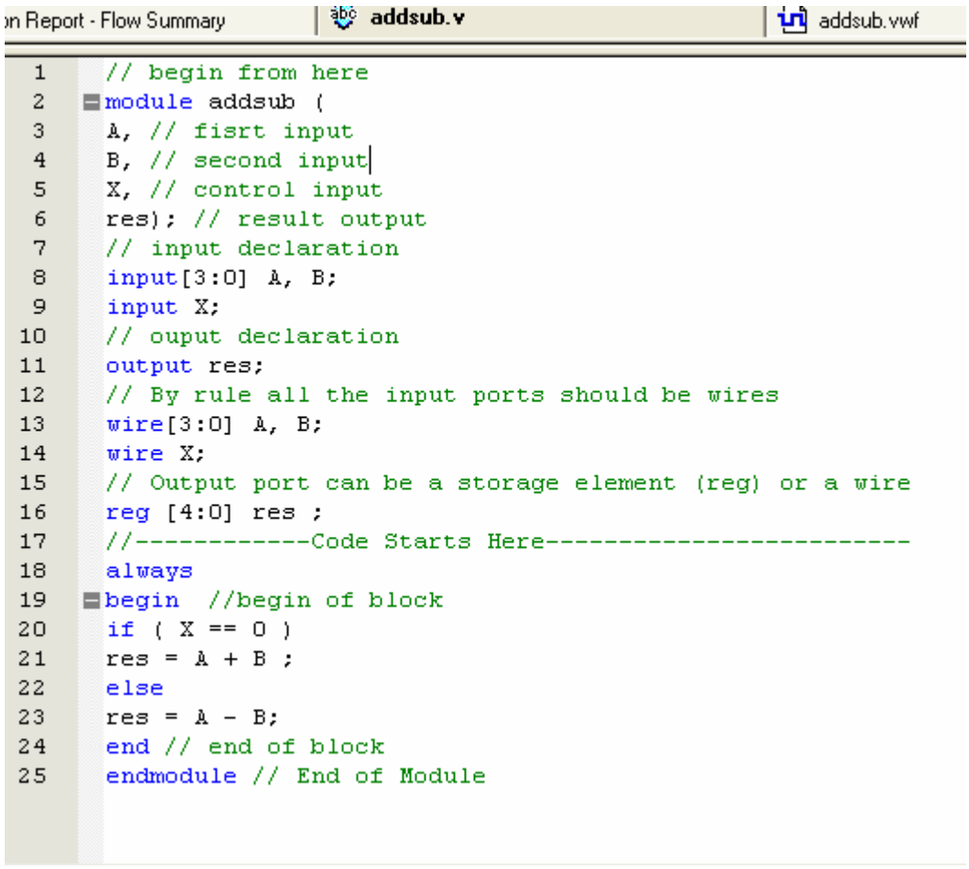
(Show the results to the instructor)

Part2: Verilog

Adder subtractor:

A and B are bytes. The circuit provides A + B when X = 0, and A - B if X = 1.

1. Create a new project as in part1
2. Make new verilog file: File --> New --> "Verilog HDL file"
3. Study, write, and compile and simulate the following program.



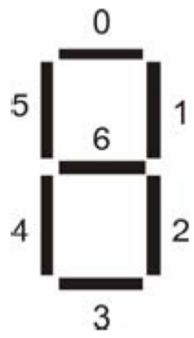
Modify the program to work as follows:

A and B are bytes. The circuit provides zero output when X=0, A + B when X = 1, and A - B if X = 2, A*B when X=3, .

(Simulate it and show the results to the instructor)

Seven segment display decoder (driver):

```
module sevenseg ( in_v, out_v);
input[3:0] in_v; // input declaration
output[6:0] out_v; // output declaration
wire[3:0] in_v; // By rule all the input ports should be wires
reg [6:0] out_v ; //Output port can be a storage element (reg) or a wire
always @ (in_v)
begin
case (in_v)
0 : out_v = 7'b1111110;
1 : out_v = 7'b0110000;
2 : out_v = 7'b1101101;
3 : out_v = 7'b1111001;
4 : out_v = 7'b0110011;
5 : out_v = 7'b1011011;
6 : out_v = 7'b1011111;
7 : out_v = 7'b1110000;
8 : out_v = 7'b1111111;
9 : out_v = 7'b1111011 ;
10 : out_v = 7'b1110111 ;
11 : out_v = 7'b0011111 ;
12 : out_v = 7'b1001110 ;
13 : out_v = 7'b0111101 ;
14 : out_v = 7'b1001111 ;
15 : out_v = 7'b1000111 ;
endcase
end // end of block
endmodule // End of Module counter
```



Make sure that you have the file sevenseg.v in the same directory you work and it is compiled Correctly, and then create symbol by File→create/update→create symbol files for current file

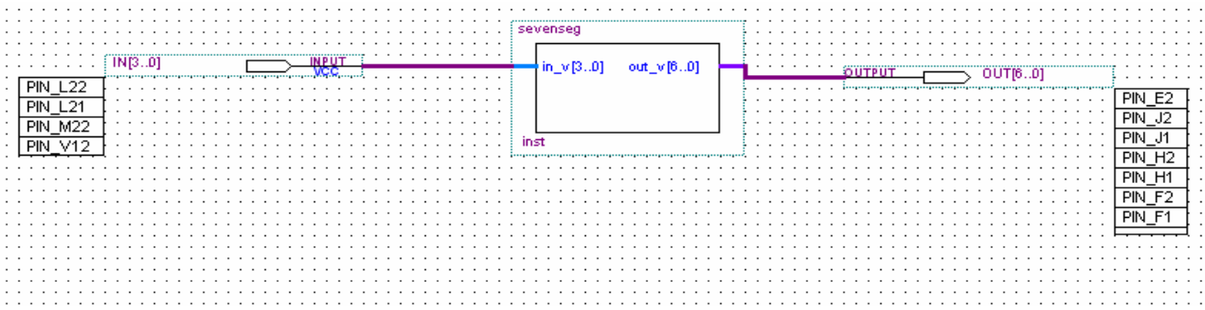
Now create a new project and a schematic file. Insert the created symbol in this file

Knowing that the seven-segment displays are active- low. Modify the above code and then

1-Assign four switches as input

2- Assign one of the sevens segments to the output

Re-compile and then rout it on the FPGA **(show the results to the instructor)**



Keep the working code; you will need it in the next experiments