

EXPERIMENT 3. INTRODUCTION TO LOGIC CIRCUITS

I. Introduction

1) Objectives

In this experiment, you will get familiar with the elementary logic gates and their usage for designing and implementing logic circuits. During this lab session you will use two different approach to implement your design. In the first part of the experiment, you will use conventional methods, IC's will be plugged on a breadboard and connected with wires. In the second part, the design will be implemented using Field Programmable Gate Arrays (FPGA). In specific, using Quartus II 14.1 schematic editor, several logic circuit designs will be drawn on computer that will then determine the connections in a single IC chip to make the desired logic function.

2) Background: Positive and Negative Logic

In a binary digital system, information is represented by the combination of ONE's and ZERO's. Each of these signal levels are indeed two non-overlapping ranges of voltages. There are two ways to assign the Boolean 0 and 1 values to these two voltage ranges: positive logic and negative logic interpretation. In positive logic, the HIGH (H) voltage level is assigned as "1" whereas the LOW (L) level is assigned as "0". On the other hand, the assignment is the vice versa in negative logic. Conventionally, unless it is specified explicitly, the logic is the positive.

A logic gate (AND, OR, NAND, etc.) is defined based on its input-output characteristic, which is determined by its internal structure. At a given time output is a function of the inputs only. It should be noted that the logical function implemented by any gate depends on the logic convention, i.e, positive or negative logic.

For further information about logic gates and positive/negative logic refer to section 2.8 of your course book titled "Digital Design" (M. Morris Mano, Prentice Hall)

II. Preliminary Work

- 1) H-L truth table of a logic gate is give in Table-1. What is the function of this gate when:
- positive logic is used?
 - negative logic is used?

INPUTS		OUTPUT
L	L	L
L	H	H
H	L	H
H	H	L

Table-1 : Truth table

- 2) What is the propagation delay of a gate? Propose a method to measure this quantity, considering the fact propagation delay is too low.
- 3) What function does the gate in Fig. 1 implement? Note that, circles indicate inversion.



Fig. 1 : The logic symbol of the gate in Part-3

- 4) Show that AND, OR and NOR functions can be implemented by only using NAND gates.
- 5) Design a two input XOR circuit using minimum number of gates given in Appendix A. (Do not use 7486 IC)
- 6) Give the explicit form of three-input XOR circuit, which is defined as:

$$Z = A \oplus B \oplus C = A \oplus (B \oplus C)$$

- 7) Consider a hypothetical voting system where there are four participants voting for a statement. Each participant expresses his idea by voting for either ACCEPT or DECLINE by sending ONE or ZERO signals to our logic circuit. The first user is dominant; regardless of the other votes, if he votes for ACCEPT the result is ACCEPT. If the first participant DECLINES the statement, the second user needs at least one more ACCEPT vote to get result. Remaining voters have equal influence on the result. No matter what they vote, if the first two users DECLINES, the result becomes DECLINE.

Considering the characteristic summarized above, first obtain the truth table. Then design this vote processing circuit, using only the gates given in the part list.

8) Fibonacci sequence is the numbers in the following integer sequence:

0 1 1 2 3 5 8 13 21 ...

Design a four-input combinational circuit with one output to determine whether a given 4-bit number is in Fibonacci sequence or not. In your design, the output should "1", if the corresponding input is a Fibonacci number and "0" otherwise.

9) Using Quartus II software tools, form the schematic of the circuit that you have designed in Part-7 and simulate the behavior and obtain the truth table by applying appropriate input combination. Refer to the first experiment manual which provides step by step instructions for schematic design and functional simulation. Take screenshot of your simulation result and schematic file and print on a single A4 paper. Don't forget to attach this paper to your preliminary work.

III. Experimental Work

A. Operation of TTL gates

- A.1) Connect the inputs of a 4-input NAND gate (74LS20) to switches (or wire a cable between input and VCC or GND) and its output to an LED. Make a truth table using the LED to determine the output level.
- A.2) Suppose you need a 3-input NAND gate. List three things you can do with the unused input pin of 74LS20 so that it does not control the output. What are the advantages and disadvantages of each connection?
- A.3) Having one of the inputs not connected, make a 3-input truth table.
- A.4) Which function does the gate perform if all inputs are tied together? What will happen if only one input is used with others not connected? Check your answers experimentally.

B. Logic Circuits

- B.1) Construct the circuit given in Fig. 2, and then obtain the truth table. Write the logic expressions for X and Y. Can this circuit be used as a three-input XOR?

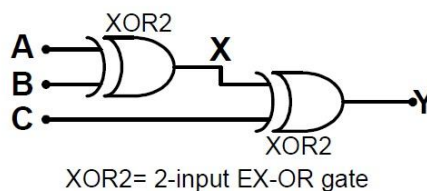


Fig. 2: Logic circuit in Part-5

- B.2) Construct the hypothetical vote analyzer that you have designed in preliminary work Part-7 and obtain the truth table.

Implementation of Logic Circuits with FPGA

In this part of the experiment, you will implement your designs using a different approach. Specifically, computer aided digital design and verification tools provided by Altera will be used. First, you will draw your design using Schematic Editor tool of Quartus II. Afterwards, functional simulation will be performed to validate your design. Finally, the design will be compiled and load on physical hardware.

General Design Flow

In this part of the experiment you will implement the logic circuits that you have designed in preliminary work, i.e., **Vote Analyzer** and **Fibonacci Number Checker**.

C. Starting a New Project

- C.1) Open Quartus II software and create a new project using Project Wizard (**File>New Project Wizard**) and name it as *EE314_EXP3* as shown in Fig. 3. You can create a new directory on Desktop and set it as the working directory.

Note: make sure the PC is started on Windows 7 64-Bit.

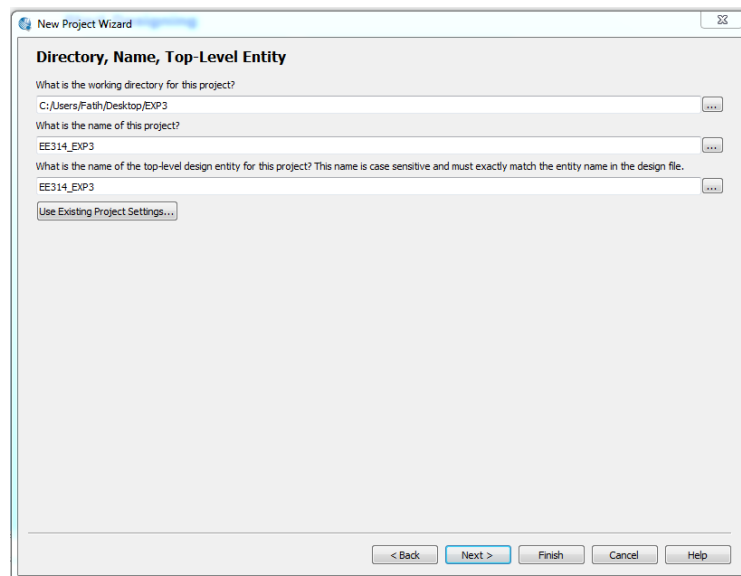


Fig. 3: Assigning working directory and project name

- C.2) Click **Next**.
- C.3) Choose **Empty Project** and click **Next** twice.
- C.4) As shown in Fig. 4, choose Device Family – Device configuration as **Cyclone V** and **5CSEMA5F31C6**, respectively. Then click **Next**.

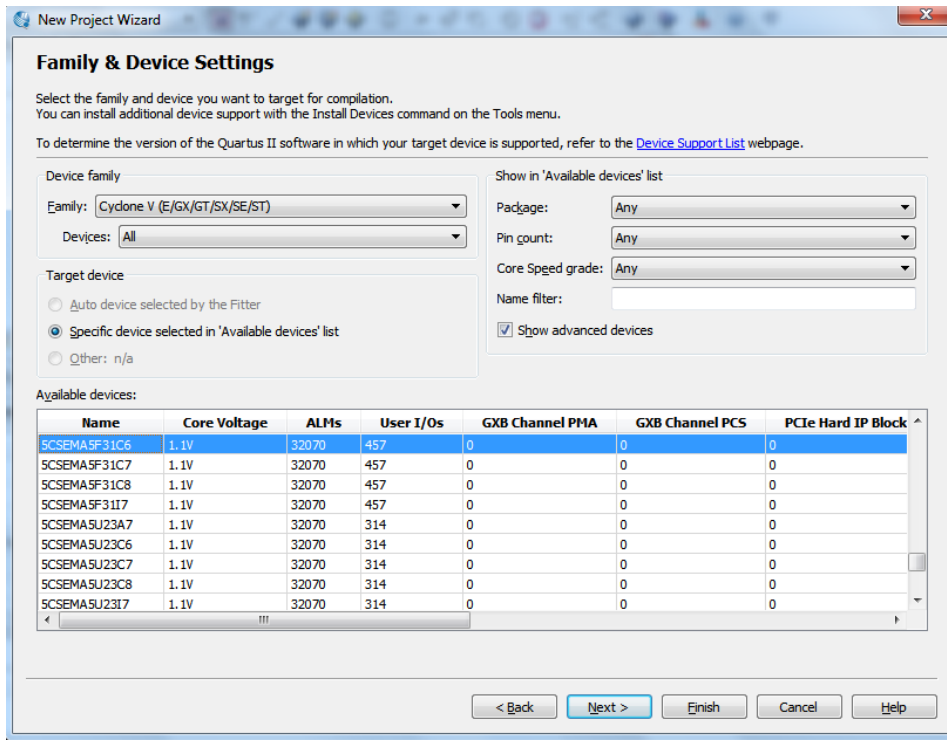


Fig. 4: Device selection window

C.5) On the upcoming window click **Next**.

C.6) A summary window will be visible. The summary window should be similar to the Fig. 5. Click **Finish**.

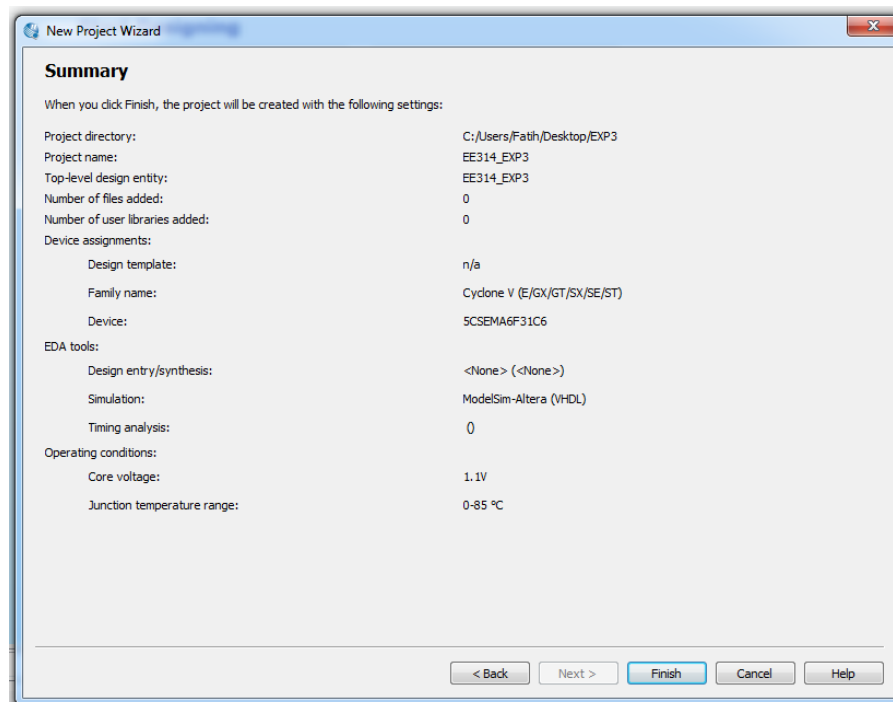



Fig. 5: Summary window

D. Entering The Design

- D.1) After you set a new project and working space, create a **Schematic File** by choosing **File>New>Block Diagram/Schematic File** and click **OK**. Then choose **File>Save as** and enter the name of the file as *EE314_EXP3*. Make sure **“Add the file to current project”** is selected and then click **Save**.
- D.2) On schematic editor window, click  on icon or double click on a blank space to open symbol library.
- D.3) Based on your **Vote Analyzer** design in preliminary work Part-7, insert appropriate gates and input/output pins on schematic window. As an example, if you have used 2-input AND gate in your design, expand the **library** directory and then expand **primitives** directory. In primitives, expand **logic** directory and choose **and2** gate as shown in Fig. 6

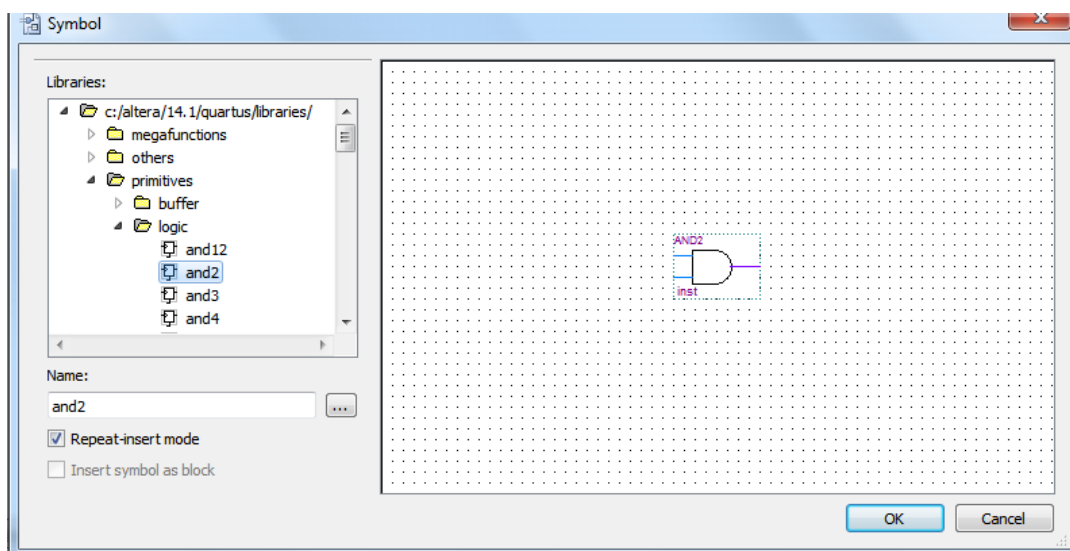



Fig. 6: Choosing a symbol from library

- D.4) Double click on **pin_name** field of input/output pins to give proper names, as shown in Fig. 7. (e.g., a,b,c,d as input pins and Y as output pin.) It is also possible to right click on pin name and choosing properties to make the assignment on the upcoming window, as you have done in the first experiment.



Fig. 7: Assigning pin names

- D.5) Wire each component using **Orthogonal Node**  **Tool**. Position the mouse pointer over the right edge of the one of your input pins as an example. Click and hold the mouse button and drag the mouse to the right until the drawn line reaches the pinstub of the gate, which we will

connect the input. Release the mouse button when you see a box appear, which leaves the line connecting the two pinstubs. A successful wiring operation can be seen in Fig.8. Apply the same procedure for remaining wires and construct the circuit. After wiring, **Save** the schematic.

Hint: If you make any mistake, you can use right click of your mouse and select delete option. It is possible to delete any component on schematic editor, i.e., logic gates, wires and input/output pins. It is also possible to use **undo (ctrl+z)** option from edit menu. In addition, if you need multiple number of copies, you can use **copy** and **paste** options from edit menu after selecting the component you want to copy.

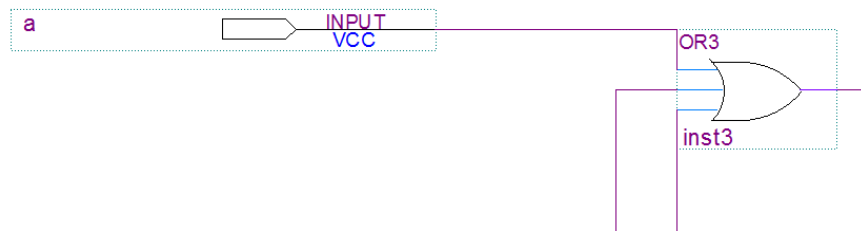



Fig. 8: Wiring components

E. Compiling the Designed circuit

So far, we have transferred our design to the computer space using schematic editor tools. Now we are ready to test our design. Whether you would like to test your design using simulation tools or on real hardware, first thing you should do is compilation. The entered schematic file, *EE314_EXP3.bdf*, is processed by several Quartus II tools that analyze the file, synthesize the circuit, and generate an implementation of it for the target chip. These tools are controlled by the application program called the Compiler.

- E.1) Run the compiler selecting **Processing > Start Compilation** or click  icon. You are supposed to save your project before compiling. As the compilation is continuing, the progress is reported at the left of the schematic. When the compilation finishes, compilation report is produced. A tab showing the report is opened automatically and you can close it. To open it whenever you want, click **Processing>Compilation Report**. There are several messages in the message window. When compilation becomes unsuccessful, appropriate messages will be given in there. If everything goes fine, the interface looks like Fig. 9.

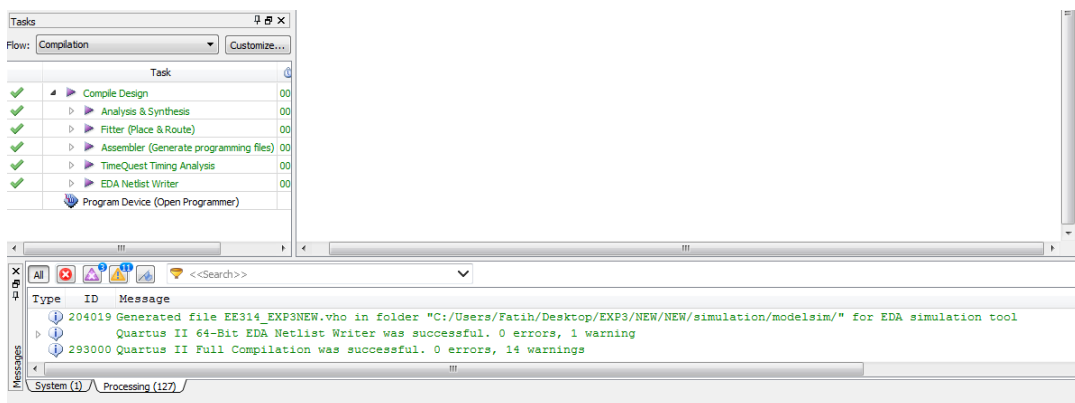



Fig. 9: Compilation result

There shouldn't be any error after compilation. If you've received any error, click  on icon to display error message. Most of the time, errors occur due to wiring issues. Another problem might occur due to missing top-level entity. If you have named your schematic file different than the project name or if you forget to click "Add file to current project" option, you will receive this error. If this is the case for you, first make sure, the schematic file is added to your project (**Project>Add Current File to Project**). Afterwards, set the schematic as the Top-Level Entity of your project. (**Project>Set as Top-Level Entity**)

- E.2) Fix errors if any and then **Save** the schematic file and **Recompile**

F. Simulating the Logic Circuit

Now, we will perform a functional simulation to make sure the designed circuit performs as required.

Note: In order to perform a simulation you don't have to make pin assignments. However, it is possible make pin assignments before or after the simulation if you need to implement the design on an FPGA. In this experiment, we will make the assignments after the simulation phase.

- F.1) Click on **File>New>Verification/Debugging Files>University Program VWF** to open the **Simulation Waveform Editor**. The simulation window is opened. Now save it as **VoteTest.vwf**.
- F.2) In the simulation window, set the desired simulation from 0 to 320 ns selecting **Edit>Set End Time** and entering **320 ns** in the dialog box that pops up. Then select **View>Fit in Window** to display the entire simulation range of 0 ns to 320 ns in the window shown in Fig. 10.

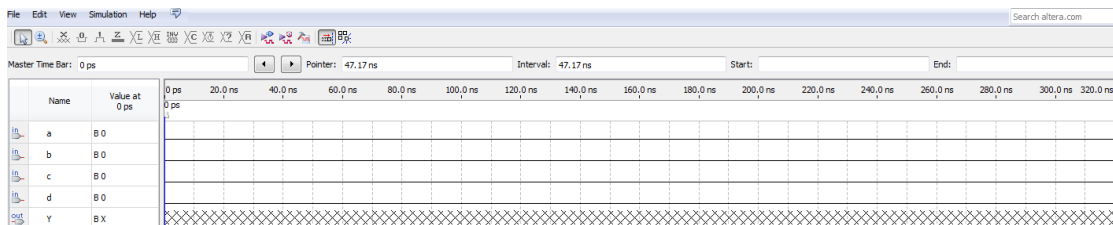
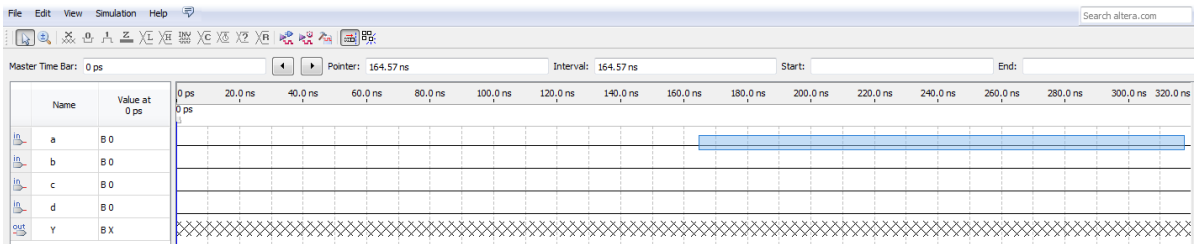
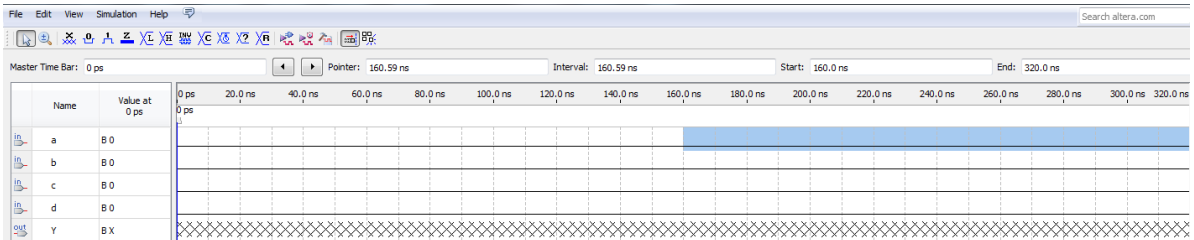


Fig. 10: The augmented waveform editor window

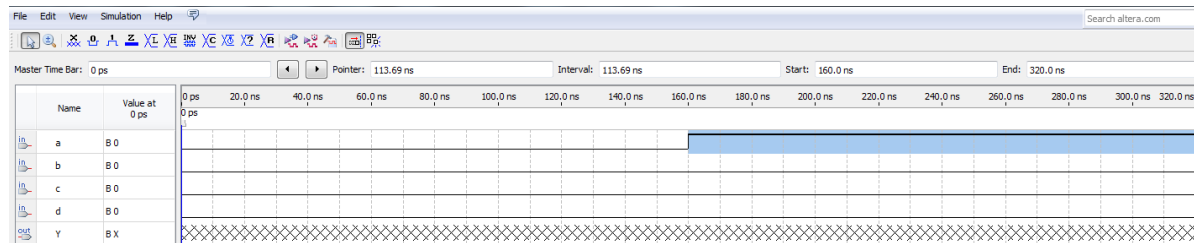
- F.3) Now input and output nodes will be included into the simulation. Click **Edit>Insert>Insert Node or Bus** and then click **Node Finder** on the opened window. In Filter menu choose Pins: all and click on the **List** button. Then click on **>>** and then **OK**. Then click **OK** again on the opened window.
- F.4) Now, input waveforms will be arranged. To make it easy to draw waveforms, waveform editor displays vertical guidelines and provides a drawing feature that snaps on these lines which can be invoked clicking **Edit>Snap to Grid** (a tick sign is needed to be seen). As default all nodes are in LOW state. In order to verify the design, reasonable values for each input signal must be arranged. As an example, click on waveform for a node and choose the time interval between 160 and 320 ns clicking on mouse when the pointer is near 160 ns and drag it until 320 ns as shown in Fig. 11a. Release the mouse button and make sure, interested portion of the waveform is selected as shown in Fig. 11b. Finally, choose **Edit>Value>Force High**. Resulting waveform, should be similar to Fig. 11c. Apply similar procedure to the remaining nodes so that all possible input combinations can be evaluated.



(a)




(b)



(c)

Fig. 11: a- Selecting a portion of the waveform, b- Active selection of the waveform
c- Editing the waveform


F.5) **Save** the waveform file (**File>>Save**).



F.6) Choose **Simulation>>Run Functional Simulation** or click on  icon. Then output waveform will be observed in a new window. Compare the result with the expected truth table. If you observe an unexpected behavior, go back to the Schematic File and check your design.

G. Pin Assignments

If everything has gone well, you are ready to implement your design on FPGA board. To this end, we will first make proper pin assignments and then program the physical hardware. In order to evaluate your design, 4 switches will be used as input elements and 1 LED as output.

G.1) Select **Assignments>Assignment Editor**.

G.2) In the **Category** drop-down menu, select All. Click on the **<<new>>** button located near the top left corner to make a new item appear in the Table 2. Double click the box under the column labeled **To** so that Node Finder button  appears.

G.3) Click on  the button to reach the window in Fig. 12. Click on  to reach more search options. In Filter drop down menu select **Pins: all**. Then click the List button to display output and input pins to be assigned: (Y, a, b, c, d). Click on **>>** button and click OK. Now all pins are labeled under the column **To**.

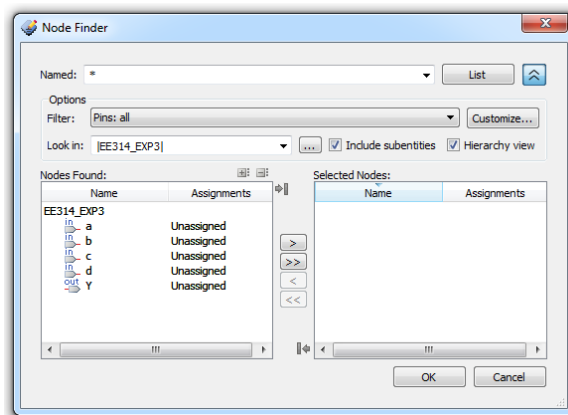


Fig. 12: Node finder window

G.4) Now open the drop down menu under **Assignment Name** column and select **Location(Accepts wildcards/group)** as shown in Fig. 13.

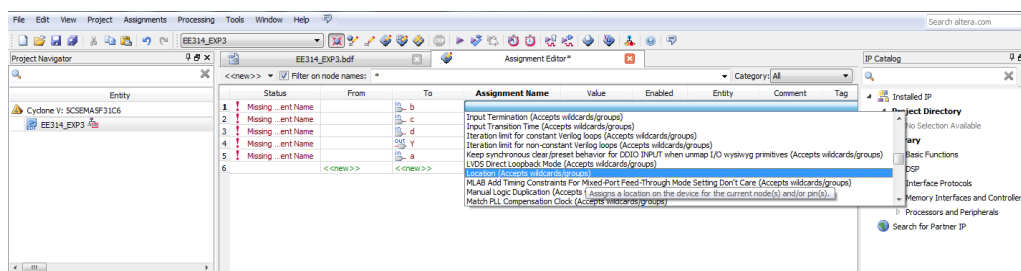



Fig. 13: Assignment name selection window

G.5) We will assign **LEDR[0]** as output and **SW[0], SW[1], SW[2], SW[3]** as a, b, c, d inputs, respectively. Write the FPGA Pin No. expression in Appendix B to the column under the Value. For example to assign SW[0] switch to a, PIN_AB12 should be written to the row under the Value column. The resulting window should be similar to the window in Fig. 14.

	Status	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
1	✓ Ok		in a	Location	PIN_AB12	Yes			
2	✓ Ok		in b	Location	PIN_AC12	Yes			
3	✓ Ok		in c	Location	PIN_AF9	Yes			
4	✓ Ok		in d	Location	PIN_AF10	Yes			
5	✓ Ok		out Y	Location	PIN_V16	Yes			
6		<<new>>	<<new>>	<<new>>					

Fig. 14: Pin assignments

G.6) Save the assignment and start compilation by  clicking.

H. Programming the Device

- H.1) Open programmer by clicking **Tools>>Programmer**.
- H.2) Turn the DE1-Soc board on by pushing the red power button.
- H.3) If DE1-SoC is not chosen default, click on **Hardware Setup** and choose **DE1-SoC** on the opened window then click on Close.
- H.4) Click on Auto Detect and then click on **5CSEMA5** and click on OK. Click **Yes** on the pop up window. After clicking **Yes** the screen should be seen like Fig. 15.

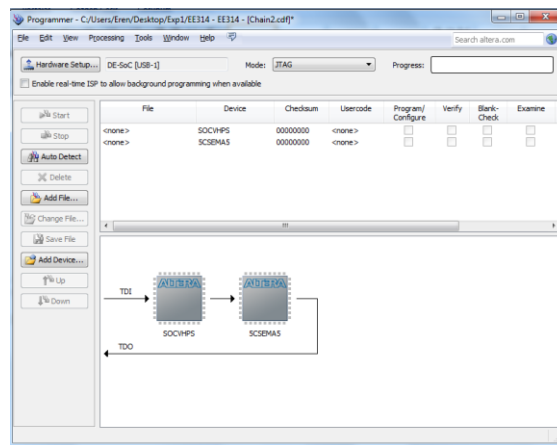


Fig. 15: Programmer window

- H.5) Double Click the **column under the file** on **5CSEMA5** row and click on **output_files** directory and then click on **EE314_EXP3.sof** file, which is your programming file.
- H.6) Click on OK. Then click on **Program/Configure**. Programmer window should look as seen in Fig. 16.

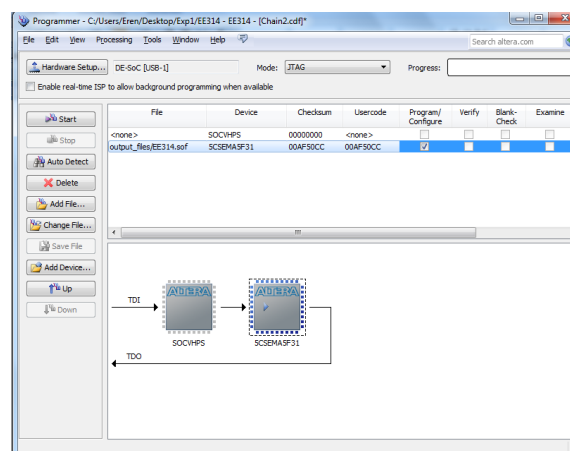


Fig. 16: Programmer window after step 5

- H.7) Finally, click on **Start** to program the device. After a successful programming operation, the progress bar should be seen as Fig. 17.

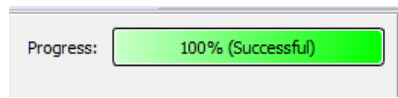


Fig. 17: Progress bar after programming the device

- H.8) Test the designed circuit applying different input combinations and compare your results with the truth table and simulation results. If you observe a different behavior as compared to your simulation results, it is most likely there has been a mistake during pin assignment phase. If this is the case, go back to the pin assignment window and fix the wrong assignments. Don't forget recompile your design before programming.

I. Fibonacci Number Checker

In this part, you will present your understanding of proper Altera usage with minimally guided study.

- H.1) Create a new schematic file and save it as **FibonacciChecker**. Don't forget to click "**add file to current project**" option.
- H.2) Based on the logic circuit design in your preliminary work Part-8, add appropriate components, input/output pins and wiring.
- H.3) **Save** the design.
- H.4) As we have named the Schematic File other than the project name itself, we need to indicate our intention to make current schematic file as the Top-Level Entity of the project. Click **Project>>Set as Top-Level Entity**. If you miss this step, the compiler will consider *EE314_EXP3.bdf* file as the Top-Level Entity.
- H.5) First, simulate the design and then program the FPGA by following the same procedure in the previous steps. You are encouraged to do it without checking the manual as much as possible. It will be a good practice to get familiar with the Quartus II development environment.

Appendix A: Part List and Pin Diagrams

7400 IC: 2 input NAND gates

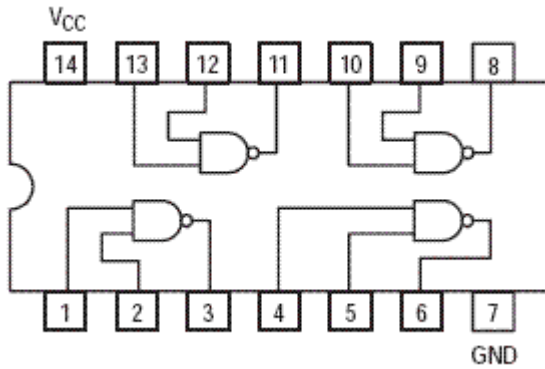
7404 IC: Inverter gates

7410 IC: 3 input NAND gates

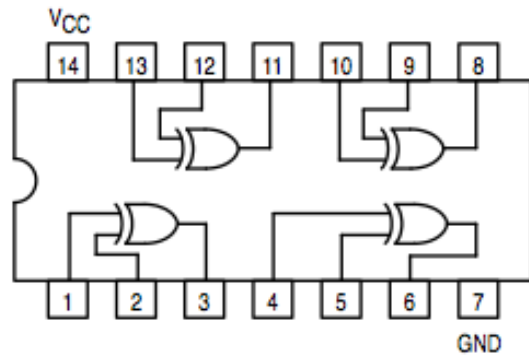
7486 IC: 2 input XOR gates

7420 IC: 4 input NAND gates

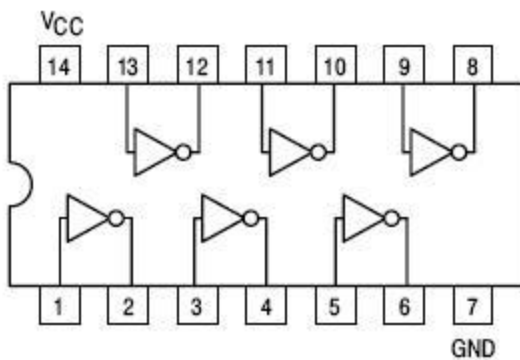
7400:



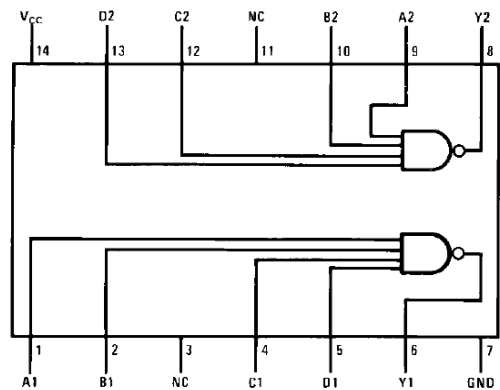
7486:



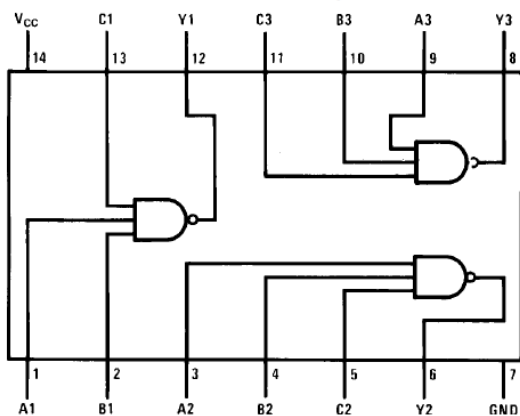
7404:



7420:



7410:



Appendix B: FPGA switch, button and LED description table

<i>Signal Name</i>	<i>FPGA Pin No.</i>	<i>Description</i>	<i>I/O Standard</i>
SW[0]	PIN_AB12	Slide Switch[0]	3.3V
SW[1]	PIN_AC12	Slide Switch[1]	3.3V
SW[2]	PIN_AF9	Slide Switch[2]	3.3V
SW[3]	PIN_AF10	Slide Switch[3]	3.3V
SW[4]	PIN_AD11	Slide Switch[4]	3.3V
SW[5]	PIN_AD12	Slide Switch[5]	3.3V
SW[6]	PIN_AE11	Slide Switch[6]	3.3V
SW[7]	PIN_AC9	Slide Switch[7]	3.3V
SW[8]	PIN_AD10	Slide Switch[8]	3.3V
SW[9]	PIN_AE12	Slide Switch[9]	3.3V
<i>Signal Name</i>	<i>FPGA Pin No.</i>	<i>Description</i>	<i>I/O Standard</i>
KEY[0]	PIN_AA14	Push-button[0]	3.3V
KEY[1]	PIN_AA15	Push-button[1]	3.3V
KEY[2]	PIN_W15	Push-button[2]	3.3V
KEY[3]	PIN_Y16	Push-button[3]	3.3V
<i>Signal Name</i>	<i>FPGA Pin No.</i>	<i>Description</i>	<i>I/O Standard</i>
LEDR[0]	PIN_V16	LED [0]	3.3V
LEDR[1]	PIN_W16	LED [1]	3.3V
LEDR[2]	PIN_V17	LED [2]	3.3V
LEDR[3]	PIN_V18	LED [3]	3.3V
LEDR[4]	PIN_W17	LED [4]	3.3V
LEDR[5]	PIN_W19	LED [5]	3.3V
LEDR[6]	PIN_Y19	LED [6]	3.3V
LEDR[7]	PIN_W20	LED [7]	3.3V
LEDR[8]	PIN_W21	LED [8]	3.3V
LEDR[9]	PIN_Y21	LED [9]	3.3V