ELE306 Spring 2008 LAB1

ELE Account: The machines in the Computer Engineering Lab are running on Gentoo Linux with KDE (K Desktop Environment). *You must get an ELE account* to use these machines, which can be obtained from ELE System Manager in Kelley 117. A storage area is granted for each ELE account designated as the folder /u/ugrads/*your_name* where *your_name* is your login name. Each account is allocated 50 megabytes of disk space on our main ELE server. You will also automatically get an ELE email account: yourname@ele.uri.edu

Altera UP-2 is the board we use here at the lab. It runs on its own power via a power adapter. It is connected to the Linux PC via a parallel port (Altera's ByteBlasterII cable). This cable is for downloading the configuration/programming information to the FPGA. Once your design is programmed on the FPGA, this cable is no longer needed. The programming of FPGA is via JTAG (Joint Test Action Group) port on the FPGA. On the Linux side, a JTAG server must be running so the ByteBlasterII hardware can be recognized.

SPECS: (manual available in the lab and on Altera's website)

FLEX 10k70 (FPGA: up to 70,000 gates) – the device we use in this class
MAX 7128 (CPLD: up to 2,500 gates) – we don't use this device
4 Seven-Segment LEDs – only two are dedicated to FLEX10k
16 LEDs – all of them belongs to MAX and thus not use
4 Buttons – two belong to FLEX10k
24 DIP switches – one set (8 DIP switches) belongs to FLEX10k
25MHz clock on-board generator
VGA connector – connected to the analog input of LCD display
PS/2 connector – connected to the keyboard
RS-232 – URI local addition for serial communication labs

Exercise (Creating a 1-bit full adder)

The goal of the first exercise is to construct and realize a 1-bit full adder on the FLEX FPGA. This lab will take you through the design step by step explaining how to use the tools to create a schematic, simulate the circuit, and program the FLEX FPGA.

Step 1: Create and Configure the Project

- The main program used for the labs is electronic design automation (EDA) tool by *Altera* called *Quartus II*. The current version is 5.0. To start the software, click the "Start (K Menu icon)-> Run Command…". Type "**quartus**" in the window and hit return. Be sure to use all lowercase letters since Linux is case sensitive. The program may take a moment to start. NOTE: the quartus II program runs from server; meaning that the program has to be loaded to your local machine first and thus the loading time varies.

- To start a new project, click "*File->New Project Wizard"*. Once the wizard window appears, click the **Next** button past the introduction. The three fields presented on next page specify what directory, project name, and top-level entity are used. For simplicity, all the three fields should specify "lab1". In the first field (topmost), add "*lab1*" so it shows /u/ugrads/yourname/lab1. Type in "*lab1*" to the second field and the third field will be filled automatically.
- **Remember to click on "Next"** not Finish yet!!! On page two, click next since there is no file from other projects that we want to import.
- On page 3, select the followings:
    - Family: "FLEX10K"
    - Target device: Specific device selected in 'Available devices' list
    - Available devices: select EPF10K70RC240-4 (last one on the list)
    - Filters: choose 'any' for all three fields. If you do see EPF10K70R240-4 in the available devices list, ignore this field.
- Skip page 4 as we do not use additional EDA tools.
- Page 5 is a summary page of your project settings. Click Finish to complete the process of creating the project.

Step 2: Drawing the circuit

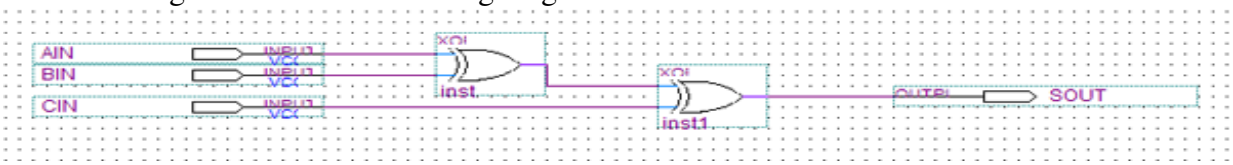
- "File->New" and select "Block Diagram/Schematic File" on the "Device Design Files" tab.
- The objective of this exercise is to create a 1-bit adder. To accomplish this task, we implement the following logic functions.

    SOUT = AIN xor BIN xor CIN
    COUT = (AIN and BIN) or (AIN and CIN) or (BIN and CIN)

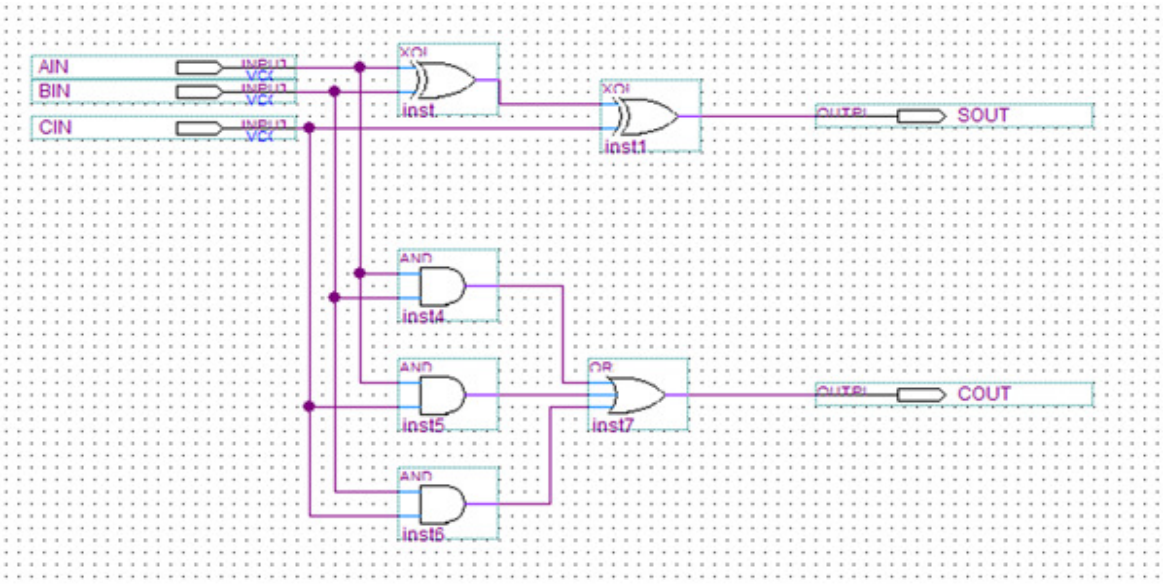
For this logic to be created, we will start by implementing the first function to generate the logic for output SOUT.

- Click on the "AND gate" symbol (third from top) on the vertical tool bar or double click anywhere in the schematic window. A window titled "Symbol" appears. Double click on the library directory to see the entire directory tree. For now you may go directly to "primitives/logic" directory where you will find all the primitive logic gates you'll need. Alternatively, you may type the name in the "Name" field if you know the name of the primitive you want.
- Type "xor" in the Name field and click OK to insert the XOR gate.
- Follow the same process from the previous step except, instead of the XOR gate, insert two inputs (symbol "input" under primitives/pin). Right click on one of the inputs, select *Properties,* and type AIN in the Pin name field. (Or simply highlight "pin_name" and replace it with the name AIN) Repeat the same process for the other input but name it BIN. You may also use the copy and paste functions to rapidly populate the schematic with the same symbol.
- Now the wiring of the devices will be described. Left click and drag from the endpoint of the input component to the input lead of the XOR gate. Repeat for the second input to the XOR gate.

- To complete the circuit of SOUT, another input gate named CIN, a XOR gate, and output gate (symbol "output") named SOUT must be inserted. Connect the output to the second XOR by creating a wire between the devices. The second XOR inserted should have one input connected to the first XOR and the input CIN. The circuit should appear to be something similar to the following diagram.



- Repeat the same process of inserting, naming, and connecting objects to create the logic for the COUT signal. Be careful not to insert duplicates of the inputs AIN, BIN, and CIN. More than one wire can be connected to each of the inputs. (Hint: three "and2" and a "or3" can be used.) A complete diagram is given below to assist in the creation of the circuit.



Step 3: Assigning to Pins of the device

For the logic to be useful, connections to external devices for I/O are needed. On the Altera board, there are many pins that connect directly to the CPLD devices. These pins are available on the outer edge of the PCB board labeled as expansion ports. To simplify some designs, elements are built onto the board. Details of the UP-2 board can be found in the manual, "University Program Design Laboratory Package," available in the lab. As can be seen on the board, there are 24 DIP switches, 4 push-buttons, 16 LEDs, and 4 Seven-segment LEDs, a VGA connector, and a PS2 port available for usage. Two Seven-segment displays are wired to each CPLD. The Flex chip has 8 DIP switches, 2 push-buttons, the VGA connector, and the PS2 port connected.

- Before assigning inputs and outputs to the I/O pins of the Altera board, the project must be compiled by selecting *Processing->Start Compilation* from the menu. A name for the

schematic will need to be specified to save it. (Leave the name at default) While the compilation process is running, the lower left corner will show the progress of the four processes: "Analysis & Synthesis", "Fitter", "Assembler" and "Timing Analyzer".

- To assign pins to the inputs and outputs of the device, choose "*Assignments -> Pins"* from the menus. In the All pins window area, you will see a table with several rows and columns. Rows start with node name (for lab1, it is AIN, BIN, etc.). There are 9 columns: Node Name, Direction, Location, I/O Bank, Vref Group, I/O Standard, Reserved and Group. Select row that starts with AIN. Double click on the "Location" entry and a list of all available pins will show up. (The FPGA has 240 pins) Select or type in "PIN_41" or simply "41" and the system will automatically show "PIN_41" in the entry. Complete the same process for the rest of the elements in the table given below.
- Click *File->Save* to save the drawing.

Pin Name	**Pin Value**
AIN	41
BIN	40
CIN	39
COUT	14
SOUT	25

**note – Due to way the devices are wired, the circuit is considered to be active low for the inputs and outputs.*

Step 4: Timing Simulation

Since there are three binary inputs, the simulation is pretty basic with an end result of 8 possible sets of input states.

- Click *File->New* and choose the *Other Files* tab. Select *Vector Waveform File* and click *OK*.
- Now that a file is created, the nodes that are being analyzed need to be inserted. To do this click *Edit->Insert Node or Bus*.
- Push the *Node Finder* button to assist in selecting the available nodes. Verify that the field Named contains "*" and Filter contains *Pins: all*. Click *List* to have the available nodes listed. Once filtered, all available nodes are listed in the *Nodes Found* field. To transfer a node to the waveform editor, highlight the name and click the ">" button. Since we want to test all nodes in this lab, click the ">>" button and then *OK*. As many nodes were selected, the *Name* field of the *Insert Bus or Node* window will contain "**Multiple Items**" to represent the many names. Click *OK* to proceed to add the nodes to the waveform editor file.
- *Edit->End Time* and select a time of 1000ns (or 1μs). This will display the entire length of the 8 possible states of input. Since each state has a length of 10ns by default, you will have to change that to 100ns. *The adder will have a delay time of 20-25ns.
- Highlight a row by clicking on the name. Then click *Edit->Value->Count Value* or the fifth button up on the vertical toolbar on the left side (the button with a waveform and letter "C" for a symbol). This button is used for inserting a waveform that counts through the possible values. In the case of this lab, the counting will be performed at different rates for the 3 signals to view all possible states of the machine.

- For the first signal AIN, select the *Timing* tab and fill in the multiply by field in with a value of 1. Following the same procedure for the other two inputs, set a count signal with a multiply by of 2 and the other 4. The other buttons' functions can be seen by holding the mouse over them and reading the text at the bottom of the Quartus II window. Another method of setting the test waveform that is simulated, select a portion of time and hit the wanted signal type from the left toolbar.
- Save the waveforms by choosing *File->Save*. Verify that it saves the waveform under the proper directory and filename. If the project's name is Lab1, then a good name for the waveform might be lab1.vwf.
- As can be seen from the waveforms, the output signals have no determinable value. To have the output generated, the input must be simulated. Simulation is done by clicking *Processing->Start Simulation* from the drop-down menus.
- Verify that the desired output is generated. This will be part of the lab report. To do this, use arrow keys, <- and ->, to navigate through the waveforms. The time will be shown on top of the cursor (a light blue vertical line).
- ***What has caused the glitches that occur in SOUT?***

Step 5: Programming

After developing the logic of the circuit and verifying the correct logical output, the information can be transmitted to the CPLD for realization. To program the CPLD, follow the next few steps.

- Click *Tools->Programmer*.
- Click the *Hardware Setup* button.
- Select the programming hardware from the "*Available hardware list*"
    - Under the *Hardware Type* field, select "*ByteBlasterII*" which contains */dev/parport0* for the *Port* field.
- Click the *Select Hardware* button. Once the hardware is made current, click the *Close*.
- Once returned to the programming window, a list of the available files that can be programmed from the current project are shown. For this particular project, there is currently only one file to program. Select the file and check the "*Program/Configure*" checkbox associated with it. The configuration file has an extension of ".sof".
- The configuration is now complete so save it by hitting the save button or by clicking "*File -> Save"* from the drop-down menus.
- Now, programming the device can be done by choosing *Processing->Start Programming* from the drop-down menus or clicking the top button of the vertical toolbar with the label *Start Programming*.
- The programming is complete and the switches states can be changed to cause the output to change. IT IS TIME TO PLAY!

Assignment (4-bit binary adder)

Each lab will contain an exercise part and an assignment part. The assignment part of this lab is for you to complete a 4-bit adder. You should create a new project calls "lab1a" for this assignment. It is always a good idea to create a different project in a different directory (folder) for different sections of a lab. You will perform the simulation just as described above. Finally, program the UP-2 board and make demonstration to the TA or the Instructor.

Lab 1 Report:

From the Exercise Part:

1. Record the delay times with respect to all possible input patterns from the simulation waveforms of the 1-bit adder.
2. Report your take on the condition(s) in which a glitch maybe provoked.
3. Demonstrate the operation of the 1-bit adder on UP-2 to the TA or the Instructor.

From the Assignment part:

4. Include the schematic of the 4-bit adder.
5. Include the simulation waveform of the 4-bit adder.
6. Demonstrate the operation of the 4-bit adder on UP-2 to the TA or the Instructor.

The Laboratory report is always due at the beginning of the next lab session.