University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

Lab 3
FPGA CAD Tool Flow

1 Motivation

In this lab you will take a simple design through the FPGA Computer Aided Design (CAD) tool-flow, from design entry to programming the hardware. However, because we don't yet have the new 150 boards yet, we'll stop just short of physically downloading the bit file. This lab will give you experience with the software that you'll be using for the rest of the semester.

2 Introduction to the Design Tool Flow

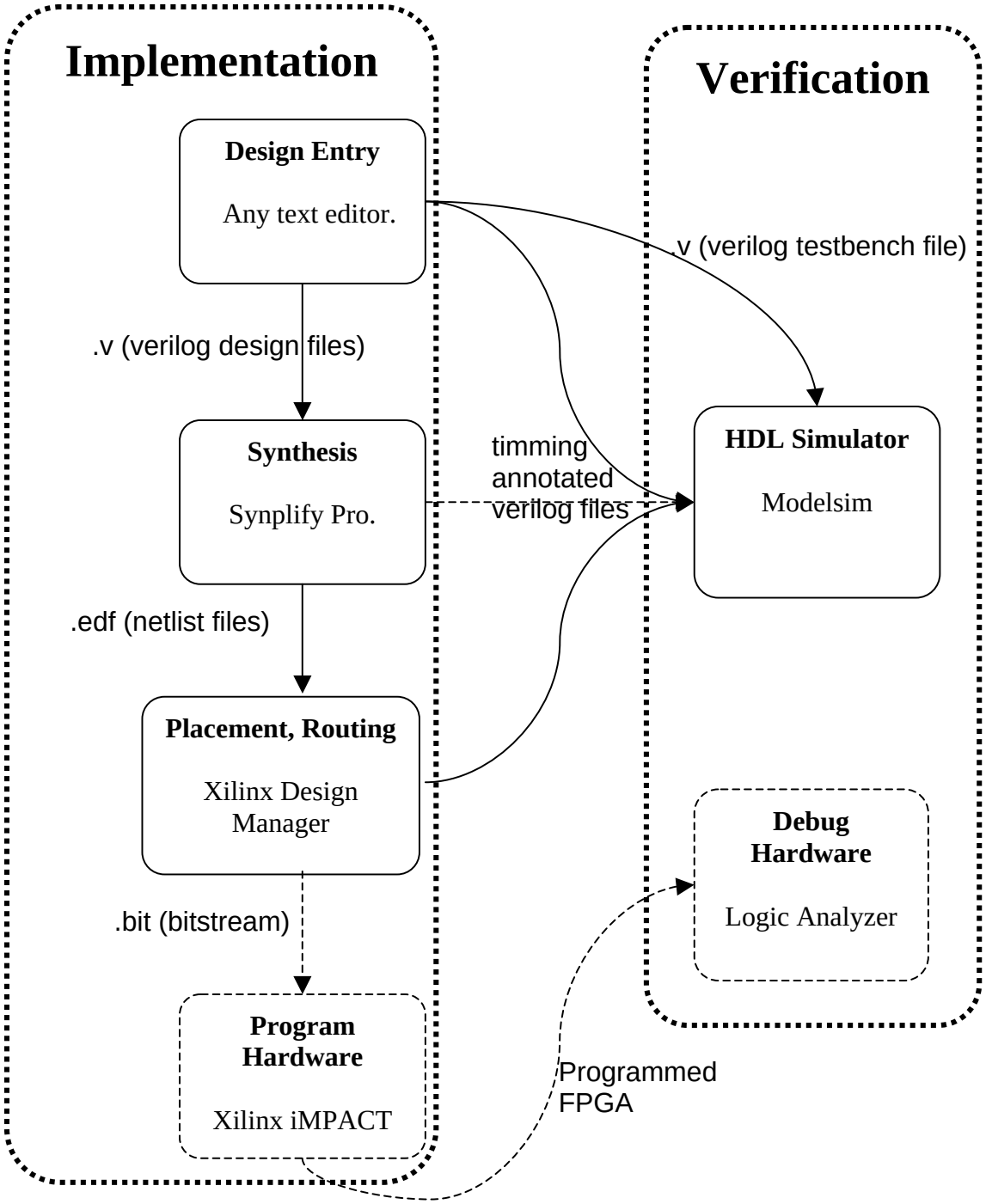
Refer to the below illustration for the steps involved in the CAD tool flow we will use.

2.1 Design Entry

The first step in logic design is to conceptualize your design. Once you have a good idea about the function and structure of your circuit, you can start the implementation process by specifying your circuit. In this class we will use a Hardware Description Language (HDL) called Verilog. HDLs have several advantages over other methods of circuit specification: ease of editing (files can be written using any text editor), ease of management when dealing with large designs, and the ability to use a high-level behavioral description of a circuit. If you are familiar with emacs, you may find it convenient for writing and editing Verilog code. *In Lab 4 you will write Verilog; for lab 3 this week, we will provide you the appropriate Verilog descriptions.*

2.2 Synthesis

Once your design is entered, the next step in the implementation path is synthesis. In our case, the function of the synthesis program is to translate the Verilog description of the circuit into an equivalent circuit comprising a set of primitive circuit components that can be directly implemented on an FPGA. In a way, the synthesis tool is almost like a compiler. Where a compiler translates to a sequence of primitive commands that can be directly executed on a processor, synthesis translates to primitive circuit components that can be directly implemented in FPGA. The final product of a synthesis tool is a netlist file, a text file that contains a list of all the instances of primitive components in the translated circuit and a description of how they are interconnected.



CS150 CAD tool flow. Optional links are showed as dashed lines.

2.3 Placement, Routing

The next step in the implementation flow is to take the netlist of components generated by the synthesis tool and turn it into bits that are need to configure the LUTs, Switchboxes, Flip-flops, and other configurable resourses in the FPGA. To do that, first the primitive circuit components in the netlist need to be assigned to a specific *place* on the FPGA. For example, a 4LUT implementing the function of a 4 input NAND gate in a netlist could be implemented with any of the about 40,000 4LUTs in a Xilinx Virtex 2000E FPGA chip. Clever choice of placement will make the subsequent routing easier and result in a faster overall circuit. Once the components are placed, the proper connections must be made according to the netlist description. That step is called *routing*. Unlike synthesis, which only requires a set of primitive components to translate to; placement and routing are dependent upon the specific size and structure of the target FPGA chip. Due to this reliance, the FPGA vendor usually provides the placement and routing programs. The end product after placement and routing is a bit file containing the stream of bits used to configure the FPGA. Note: placement and routing are NP hard optimization problems and the provided software uses heuristics to solve them. There are cases where humans can do a better job by hand.

2.4 Program Hardware

The last step in the implementation flow is the simple act of transporting the configuration bits to the FPGA. There are also many ways of doing this. For this class we will be mostly using the Parallel Cable IV along with the iMPACT software to program the board.

2.5 Verification

As you should have learned from experience, a significant part of the time and effort used on any sizable project will be spent on debugging, and logic design is no exception. There are two ways to verify the correctness of a design: to program the FPGA with the design and check if the circuit is behaving correctly, or to run simulations of the design in software. To program the FPGA and physically check the functionality sounds simple and is in fact the final testing that a design must pass. However, the whole tool flow takes a while to run and repeatedly tweaking the input design to fix errors would require running the flow repeatedly. In addition it can be difficult to physically observe the causes for an error on a FPGA. For these reasons, software simulation is also needed in the verification process. There are many places along the tool flow where you can use simulation to verify the correctness of your design. For this class we will use a HDL simulator for all the different simulations. The first place to simulate your design is right after design entry. At this point you can only test functionality. Because there is no information available about the actual implementation on the FPGA, there is no way to accurately predict delay. As you progress down the tool flow and more information about the physical implementation on the FPGA becomes available, more accurate timing simulations can be performed. The CAD tool at each step along the implementation flow is capable of producing Verilog files annotated with timing values that can be used in simulation.

3 Prelab

1. Read and understand the introduction to Design Tool Flow (above).

2. Take a look at the provided Verilog files to see if you can decipher them. (If you have not had experience with Verilog in the past, you will probably not understand everything in this files, but should be able to understand the basic function of the specified circuit and the operation of the tester.)
   http://inst.eecs.berkeley.edu/~cs150/handouts/4/Lab/lab3_cir.v
   http://inst.eecs.berkeley.edu/~cs150/handouts/4/Lab/lab3_cir_testbench.v

3. Read section 1,3,7,11 of the Modelsim tutorial.
   http://inst.eecs.berkeley.edu/~cs150/handouts/4/Lab/qk_guide.pdf

4 Procedure

4.2 Functional simulation

1. Download `lab3_cir.v` and `lab3_cir_testbench.v` from the course website.
2. Start the Modelsim simulator.
3. Create a new project and add the two provided files as existing files.
4. Compile both files.
5. Start a simulation of the **lab3_cir_testbench** module.
6. Select **view->all windows** in the menu bar.
7. Add the signals IN, E, R, CLK, OUT in the wave window by dragging them from the signals window.
8. Advance the simulation time by 1u sec by typing “run 1us”

Look at the waveform from the wave window. Can you tell what the function of this circuit is from here? (You can change the display from binary to hex on a group of signals by right-clicking the signal then choosing **Radix->Hexadecimal**.)

Question 1 What is the clock to output delay in this simulation?

4.3 Synthesis

Now we start to map the design to an FPGA:

1. Start the synthesis program Synplify Pro.
2. Start a new project from **File->New**, project file.
3. Add the source file `lab3_cir.v` to the project. Note: Don’t add the testbench file because it is there only to provide input for the design in simulation.
4. Change target device in the implementation option to Xilinx Virtex-E XCV2000E FG680

Note: You can also get the tool to output a Verilog netlist for simulation by choosing **Implementation Results->Write Mapped Verilog netlist**. But we will not use this option for this lab.

5. Click **RUN**
6. Click and examine RTL view and Gate View

Question 2 Can you now figure out the function of this circuit?

Question 3 What are the exact module names of the different types of primitive components this circuit maps to? What are their functionalities? Goto the Xilinx website http://toolbox.xilinx.com/docsan/xilinx4/manuals.htm
And look for their functionality in the **Librarys Guide**.

4.4 PlaceRoute

1. Start the Xilinx Design manager by choosing from Synplify Pro, **Options->Xilinx->Start Design Manager.**
2. Choose **Design->Options.**
3. Choose **Modelsim Verilog** for simulation.
4. Click **implement**.

Question 4 Choose the report browser and find the number of slices, 4Luts, and Flipflops used in this design, and the minimum clock period for this design.

#Slices_____ #4LUTS_____ #Flops_____

Min Period_____

4.5 Floor planner, FPGA editor

The floor planner is a program that lets the designer do some manual placement. Run the floor planner from Design manager. The floor planner should open some windows. The window on the left shows the pieces of components that need to be placed. The gray window shows the result of the automatic placer algorithm. And the white window is an area for the designer to do his own placement. *Optional: Can you come up with a better placement than the automatic*

placer? Note: you can start out with the result of the auto placer by choosing ***Floorplan -> Replace all with placement****.*

After you are done with the floor planner, run the FPGA editor from Design manager. This tool shows you an even more detailed view of the FPGA. You can see all the occupied slices and even the detailed routing from this tool. You *could* also make changes to the design with this tool, however this is not recommended as it is essentially equivalent to modifying the bit stream directly, and thus very prone to errors.

4.6 Timing Simulation

The last thing we will do is to simulate the design after it has been placed and routed. This process is a little tricky since it involves the output Verilog file of the Design Manger, a Xilinx library that contains primitive module delays, and a .sdf file that contains the wire delay between the different components.

1. Start Modelsim
2. Download http://inst.eecs.berkeley.edu/~cs150/handouts/4/Lab/libs.zip
3. Extract the contents to C:\temp
4. Add the provided file lab3_cir_testbench.v to the project.
5. Fnd time_sim.v, time_sim.sdf in your project directory from Synplicity and Xilinx and add them to your project. Make sure you check the **Copy to project** box for the .sdf file
6. Remove the two lines:
   wire GSR = glbl.GSR;
   wire GTS = glbl.GTS;
   From time_sim.v
7. **Compile** the two Verilog files
8. Choose **Simulate**
9. Under the tab **libraries** add the simprims directory that you created from lib.zip
10. Then under design Load the testbench.
11. Repeat the rest of the simulation steps from earlier.

Question 5 What is the clock to output delay of the circuit in this simulation? What part of the circuit is associated with the delay you observed? Is this the critical path? Why or why not?

Question 6 Modify the `lab3_cir_testbench.v` file so that the clock period is 2ps shorter than the minimum clock period indicated by the report from design manager and repeat the simulation. Does the circuit still function? Why?

J. Wawryznek, N. Zhou, Fall 2002

5 Checkoffs

Name: _____Name: _____
Lab Section: **M**: PM **Tu**: AM PM **W**: AM PM
Th: AM

Prelab Questions

Functional Simulation **TA: _____(10%)_____(05%)**

Timing Simulation **TA: _____ (20%)_____(10%)**

Timing Simulation at different Period **TA: _____ (10%)_____(05%)**

Questions **TA: _____ (60%)_____(30%)**