

Lab Assignment 2

ECE/CS 3700

Spring 2009

Assigned during the week 2/2-2/6 onwards, Due date: your respective lab sessions during the week 2/16-2/20.

The objective of this lab assignment is to learn the techniques of “Schematic Entry”, “Verilog Design”, “Simulation” and “Synthesis” in the context of digital logic design. Moreover, this lab will introduce to you an elementary 2-bit computer. Your job is to: i) design the logic using a CAD tool called the “Schematic Editor”; ii) “Simulate” your design using a Verilog testbench; iii) Re-design the same circuit using Verilog description and simulate using the same test bench; and finally, iv) “Synthesize” the netlist and download the design onto the FPGA and demonstrate its correct operation. This way, you will learn the top-down design flow and prototyping using FPGAs.

Note: On the class website, I’m also going to upload (soon) a few tutorials on each of the above exercises. These tutorials are created so that you can use the step-by-step procedures to get acquainted with the CAD tools and associated resources. My suggestions would be to first recreate the (very simple) tutorial to see that you can invoke these tools readily from your accounts and then proceed to solve the problem given below. Hope you’ll have fun with this lab.

I. THE TWO-BIT COMPUTER

You are to design a circuit depicted in the black-box (block) diagram shown in Fig. 1. The circuit takes three (word-level) inputs $A[1 : 0]$, $B[1 : 0]$, $I[1 : 0]$; where A, B, I are each 2-bit vectors. In other words, $A[1 : 0] = \{a_1, a_0\}$, are binary variables¹. $A[1 : 0]$, $B[1 : 0]$ are the “data” inputs of the design and $I[1 : 0]$ corresponds to the “control” or the “instruction” input. The output is also a 2-bit vector, $F[1 : 0] = \{f_1, f_0\}$.

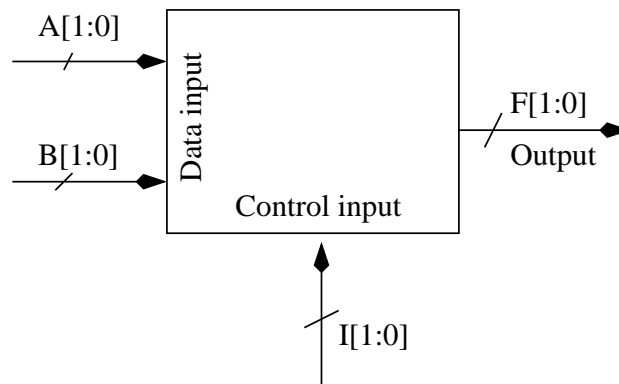


Fig. 1. Block Diagram of a two-bit Computer

¹For all practical purposes, your design has 6 binary variables as inputs: $a_1, a_0, b_1, b_0, i_1, i_0$. It is, however, customary to represent data as a group or a vector of bits, giving rise to a notation involving bit-vectors.

Circuit functionality: The values of the input control signals ($I[1 : 0]$) decide the operation of this circuit. The instructions are:

- When $I[1 : 0] = (i_1, i_0) = (0, 0)$, the output $F[1 : 0]$ is a bit-wise AND of the inputs, i.e. $f_1 = a_1 \cdot b_1$; $f_0 = a_0 \cdot b_0$.
- When $I[1 : 0] = (i_1, i_0) = (0, 1)$, the output $F[1 : 0]$ is a bit-wise OR of the inputs, i.e. $f_1 = a_1 + b_1$; $f_0 = a_0 + b_0$.
- When $I[1 : 0] = (i_1, i_0) = (1, 0)$, the output $F[1 : 0]$ performs an *equality check* of the corresponding bits; i.e. $f_0 = 1$ if $a_0 = b_0$, otherwise $f_0 = 0$. Similarly, $f_1 = 1$ when $a_1 = b_1$, otherwise $f_1 = 0$.
- When $I[1 : 0] = (i_1, i_0) = (1, 1)$, the output $F[1 : 0]$ perform the complement of $A[1 : 0]$; i.e. $f_0 = \overline{a_0}$ and $f_1 = \overline{a_1}$.

II. THE ASSIGNMENT

You are asked to design a circuit that implements the above function. You are free to use any technique that you wish for circuit optimization. Spend a few minutes to think about the design, perhaps you may want to draw a circuit diagram on paper...

Once your design is ready, do the following:

- **Schematic entry:** Using the ISE Webpack tool, you will draw the schematic corresponding to your design, using AND, OR, NOT, XOR, etc. gates. Using a Verilog testbench, you will simulate the schematic and verify the correctness of your design. Your testbench should apply ALL possible inputs to the design. A tutorial on how to use ISE/Schematics/Testbench/etc. is available on the class website. The TAs will also give you a demo of the same.
- **Structural Verilog:** Once your schematic entry and simulation is complete, you will now write a structural Verilog description of the circuit. Structural description means that you will use AND/OR/NOT/XOR/etc. gate instantiations (Verilog primitives, as we covered in the class, also as shown in Fig. 2.31 in textbook) so that your Verilog corresponds to your schematic. Using the same testbench from above, you will now simulate the Verilog code.
- **Functional Verilog:** As the third exercise, you will now re-design the same circuit using continuous assign statements (as shown in Figs. 2.34 and 2.35 in the textbook). Simulate this circuit again with the same testbench. This is often called “functional” Verilog as we are describing Boolean functions.
- In this lab, you are **not** allowed to use the **behavioural** construct, such as the ‘always’ statement, for the design. We will do that in later labs.
- **Synthesis:** Finally, you will synthesize the circuit, generate the schematics, place and route the hardware and generate the bit-file to program the FPGA. Using the XESS tools, you will download the circuit on the FPGA and show its correct operation. You may choose any type of switches/LEDs from your lab kits.

A. Lab report submissions

You are expected to document your labs in a professional manner. Show your optimizations (if any), turn in the schematic of your circuit – CAD tools can print the schematics for you. Also, turn in your Verilog code, simulation

testbenches and the simulation results.

Take care to draw the schematics neatly and professionally. Organize your layout, the wires and components should be spaced as evenly as possible. Eliminate jogs in wires, line-up the components, etc. You get the idea.

This is a two week lab. In the first week, make sure you are able to invoke the tools properly, and you learn the design flow. The TAs will help you getting started with the CAD tools. Once that hurdle is crossed, then the design assignment will not be too much of a problem. For final check-offs, show your design, schematic and the code, simulations, as well as a functioning circuit on the FPGA, to the TAs during the week 2/16-2/20.

Once you learn the design and synthesis process in this lab, you'll be ready to start working on some really interesting and non-trivial digital design applications (next lab onwards).