CS 223 Digital Design Laboratory Assignment 3

Digital Building Blocks: Decoders and MUXs in SystemVerilog

Preliminary Report Due: November 18, 2024 08:00

Lab Dates and Times

Section 1: November 18, 2024 Mon. 08:30-12:20 in EA-Z04
Section 2: November 19, 2024 Tue. 08:30-12:20 in EA-Z04
Section 3: November 20, 2024 Wed. 08:30-12:20 in EA-Z04
Section 4: November 18, 2024 Mon. 13:30-17:20 in EA-Z04
Section 5: November 22, 2024 Fri. 08:30-12:20 in EA-Z04
Section 6: November 19, 2024 Tue. 13:30-17:20 in EA-Z04

Location: EA-Z04 (in the EA building, straight ahead past the elevators)
Groups: Each student will do the lab individually. Group size = 1

Preliminary Work [35]

Note: This part must be completed before coming to the lab. Prepare a neatly organized report of your work and submit on Moodle before the start of the lab. Include your code as **plain text, not image!**

Today's lab needs considerable prior preparation. These prior designs and SystemVerilog models should be prepared in advance, and assembled neatly into a Preliminary Report with a printed cover page and printed pages for the schematics and SystemVerilog codes. Each part should have a proper heading. Your report should include a cover page with the following information: course name and code number, the number of the lab, your name and student ID, and date.

Multiplexer

[3] Graphical symbol, logic diagram (using AND, OR, NOT gates), and truth table of a 3-to-1 multiplexer.

[2] Behavioral SystemVerilog module for 3-to-1 multiplexer and a testbench for it.

[1] Graphical symbol and truth table of a 4-to-1 multiplexer. Include the which input corresponds to which combination of select bits in your symbol. For example denote the first input pin with "00" on the symbol.

[2] Behavioral SystemVerilog module for 4-to-1 multiplexer by using three 2-to-1 multiplexers and a testbench for it.

- Consider the truth table of the three-input function given in Table 1a.

Table 1: Three-input logic functions.

(a) Three-input logic function f.

w_1	w_2	w_3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(b) Three-input logic function g.

w_1	w_2	w_3	g
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

[2] Implement the function f given in Table 1a using a single 4-to-1 multiplexer and minimal amount of inverters. Describe the procedure with a few short sentences. Draw its logic diagram.

[2] Implement the function f given in Table 1a using minimal amount of 2-to-1 multiplexers and inverters. Describe the procedure with a few short sentences. Draw its logic diagram.

[1] Which operation is realized by the function f given in Table 1a?

- Consider the truth table of the three-input function given in Table 1b.

[2] Implement the function g given in Table 1b using a single 4-to-1 multiplexer. Describe the procedure with a few short sentences. Draw its logic diagram and modified truth table.

[2] Implement the function g given in Table 1b using a single 2-to-1 multiplexer and minimal amount of gates. Describe the procedure with a few short sentences. Draw its logic diagram and modified truth table.

[2] Behavioral SystemVerilog module for the function g given in Table 1b that you implemented with a 2-to-1 multiplexer in the previous part and a testbench for simulation.

Decoder

[3] Graphical symbol, logic diagram, and truth table of a 2-to-4 decoder. Include the "Enable" signal.

[2] Behavioral SystemVerilog module for 2-to-4 decoder and a testbench for it.

[2] Logic diagram of 3-to-8 decoder using 2-to-4 decoders and minimal amount of gates. You can draw 2-to-4 decoders as a block. Include the "Enable" signal.

[2] Structural SystemVerilog module for 3-to-8 decoder using 2-to-4 decoders you implemented in the previous part and a testbench for it.

Logic Function

[2] Schematic (block diagram) and SystemVerilog module of 8-to-1 MUX by using two 4-to-1 MUX modules, two AND gates, an INVERTER, and an OR gate. Prepare a test bench for it.

[5] Schematic (block diagram) and SystemVerilog module for $F(A,B,C,D,E) = \sum(0,1,2,3,4,10,11,16,17,18,19,25,28,29)$ function (note that there are 14 terms), using one (not two) 8-to-1 multiplexer and one 2-to-4 decoder. Do not use any other gates (AND, OR, XOR, etc.). Your design must contain only one 8-to-1 multiplexer and one 2-to-4 decoder. Complements of signals are available.

Important notes on signal assignments on FPGA:

- In this lab, you are going to name multi-bit input and output signals. If you choose to name a bits individually in your SystemVerilog code, take care to index the most significant bit with the most significant index number. For example, if you have a four bit signal Y, represent it as $y_3y_2y_1y_0$ where y_3 is the most significant bit.
- Be consistent with the ordering, otherwise you may confuse yourself into thinking that your circuit is faulty where in reality you only misconnected your correctly functioning modules.
- Follow the same convention while assigning your inputs to switches and outputs to LEDs. While experimenting, you are more likely to place the board where the switches are closer to you. In this orientation, try to physically assign the most significant bits of a signal on the left hand side of switches and LEDs. For example, if you want to input decimal "10" in binary format with switches, you should input "1010" to switches where the leftmost switch is 1. If you hold the board in the other orientation, simply flip the physical pin assignment order. Just be consistent with the way you assign your pins.
- Representing and assigning signals with this convention is more intuitive and following it will help both you while debugging and TAs while grading your circuits.

Part 1: Decoder [15]

Decoders are widely used in digital design, as a building block. Although they themselves can be built with logic gates, their function is often described (and modeled in SystemVerilog) rather than their structure.

A 2-to-4 decoder decodes a 2-bit input binary number by setting exactly one of the decoder's 4 outputs to 1. Unless it has an enable signal, one and only one output of a decoder will ever be 1 at the same time, corresponding to the current value of the inputs. With an enable signal, it is possible to make all the outputs be 0, when the decoder is disabled. When enabled, it behaves as described above. Decoder outputs are mutually exclusive, and in fact are the minterms of the inputs.

Design: Give the System Verilog code which models a 2-to-4 decoder in behavioral style. (This means modeling with Boolean equations, using continuous assignment statements.)

[5] **Simulation:** Using the System Verilog testbench code, verify in simulation that your 2-to-4 decoder with enable is working correctly. (Be sure to compare the order of the ports in your module with the order of the ports in the instantiation of your decoder in the testbench to make sure they match 1-to-1.)

Implementation: Now, follow the Xilinx design flow to synthesize, create programming file, and download your 2-to-4 decoder bitstream to your BASYS-3 FPGA board.

[10] **Test:** Using the switches and LEDs on BASYS-3 that you have assigned now, test your decoder. When you are convinced that it works correctly, show the physical implementation results to the TA. Be prepared to answer questions that you may be asked.

Part 2: Multiplexer [20]

A multiplexer ("MUX" for short) is another higher-level building block, used widely in digital design. A M-to-1 multiplexer has M data inputs and 1 data output, and allows only one input to pass through to the output. A set of select inputs determines which input to pass through. MUXs can be composed into larger MUXs, as you will see in this part of the lab.

Refer to Table 1b for the truth table of function g. You implemented this function with one 2-to-1 multiplexer and some logic gates in the preliminary work.

[5] **Simulation:** Simulate your function g circuit implemented with 2-to-1 multiplexer and some gates and show it to your TA.

[5] **Test:** Now, follow the Xilinx design flow to synthesize, create programming file, and download your function g implementation bitstream to your BASYS-3 FPGA board. Using the switches and LEDs on BASYS-3 that you have assigned, test your circuit. When you are convinced that it works correctly, show the physical implementation results to the TA. Be prepared to answer questions that you may be asked.

Design: Write structural System Verilog code for an 8-to-1 multiplexer using two 4-to-1 multiplexers, two AND gates, an OR gate and an inverter.

[5] **Simulation:** Simulate 8-to-1 multiplexer and show it to your TA.

[5] **Test:** Now, follow the Xilinx design flow to synthesize, create programming file, and download your 8-to-1 multiplexer bitstream to your BASYS-3 FPGA board. Using the switches and LEDs on BASYS-3 that you have assigned now, test your 8-to-1 multiplexer. When you are convinced that it works correctly, show the physical implementation results to the TA. Be prepared to answer questions that you may be asked.

Part 3: Logic Function [15]

Set up the circuit you designed in the preliminary work for $F(A, B, C, D, E) = \sum(0, 1, 2, 3, 4, 10, 11, 16, 17, 18, 19, 25, 28, 29)$ in a new module, using a single 8-to-1 multiplexer and a 2-to-4 decoder. Using inverses of signals is allowed.

[5] **Simulation:** Simulate your implementation and show it to your TA.

[10] **Test:** Implement your circuit on FPGA using switches as inputs and a LED as output. Show your circuit to TA. Be prepared to answer questions that you may be asked.

Part 4: Bit Shifter [15]

In digital systems it is often necessary to have circuits that can shift the bits of a vector by one or more bit positions to the left or right. Design a circuit that can shift a four-bit vector $W = w_3w_2w_1w_0$ one bit position to the right when a control signal Shift is equal to 1. Let the outputs of the circuit be a four-bit vector $Y = y_3y_2y_1y_0$ and a signal k, such that if `Shift` $= 1$ then $y_3 = 0, y_2 = w_3, y_1 = w_2, y_0 = w_1,$ and $k = w_0$. If `Shift` $= 0$ then $Y = W$ and $k = 0$.

Design and implement the bit shifter circuit described above.

[5] **Simulation:** Simulate bit shifter circuit and show it to your TA.

[10] **Test:** Now, follow the Xilinx design flow to synthesize, create programming file, and download your bit shifter circuit bitstream to your BASYS-3 FPGA board. Use four switches to input your vector, and another switch to input your `Shift` signal. Use four LEDs to represent your output vector and another LED to represent k. When you are convinced that it works correctly, show the physical implementation results to the TA. Be prepared to answer questions that you may be asked.

Hint: Think about which component you can use to shift bits of a vector.

Clean Up

1. Clean up your lab station, and return all the parts, wires, the Beti trainer board, etc. Leave your lab workstation for others the way you would like to find it.
2. CONGRATULATIONS! You are finished with Lab #3 and are one step closer to becoming a computer engineer.

Notes

- Advance work on this lab, and all labs, is strongly suggested.
- Be sure to read and follow the Policies and other related material for CS223 labs, posted in Moodle.

Lab Policies

1. There are three computers in each row in the lab. Don't use middle computers, unless you are allowed by lab coordinator.
2. You borrow a lab-board containing the development board, connectors, etc. in the beginning. The lab coordinator takes your signature. When you are done, return it to his/her, otherwise you will be responsible and lose points.
3. Each lab-board has a number. You must always use the same board throughout the semester.
4. You must be in the lab, working on the lab, from the time lab starts until you finish and leave. (bathroom and snack breaks are the exception to this rule). Absence from the lab, at any time, is counted as absence from the whole lab that day.
5. No cell phone usage during lab. Tell friends not to call during the lab hours–you are busy learning how digital circuits work!
6. Internet usage is permitted only to lab-related technical sites. No Facebook, Twitter, email, news, video games, etc–you are busy learning how digital circuits work!
7. If you come to lab later than 30 minutes, you will lose that session completely.
8. When you are done, DO NOT return IC parts into the IC boxes where you've taken them first. Just put them inside your Lab-board box. Lab coordinator will check and return them later.