Lab 1: Switches, Lights and Multiplexers

CSC 258, University of Toronto

September 17, 2014

About this Lab

Learning Objectives

1. To introduce you to Verilog and the Altera DE2 boards. You will gain familiarity with them over the course of the term. This week, we will use the switches SW_{17-0} on the DE2 board as inputs to the circuit. We will use light emitting diodes (LEDs) and 7-segment displays as output devices.
2. To give you practice in designing multiplexer circuits.
3. To introduce *decoders* and making decoder circuits.
4. To get familiar with testing and simulation tools.

Marking Scheme

Read and understand the Pre-lab section before the lab. All items in this document labeled **PRELAB TODO:** are to be completed before arriving to the lab, and is worth 1 mark. You will **not** be allowed to do in lab work unless you have your prelab work handed in. Items marked **INLAB TODO:** are to be completed while your are in the lab. The basic components of the lab are worth 2 marks, and the Advanced component is worth 1 mark.

1 Pre-lab Assignment

You are required to write the Verilog code for all of the following subsections. Perform the tasks marked as **PRELAB TODO:** and bring your prepared Verilog code to the lab to be marked by the TAs.

1.1 Getting Quartus II for you!

To do the pre-lab, you will need to write code in Verilog. For this, we recommend using Quartus II as your Integrated Development Environment (IDE). **PRELAB TODO:** See the instructions posted on Blackboard or go to `http://www-ug.eecg.utoronto.ca/msl/handouts/quartus.html` to find out how to get access to the software.

1.2 Other Resources

An introduction to using Quartus II is available on Blackboard under **Labs**, named **Quartus II Introduction (Verilog design)**. The DE2 User Manual is also available on Blackboard.

There is a Verilog Primer available on Blackboard, as well as a number of Verilog resources on the internet, such as `http://www.verilogtutorial.info/` and `http://www.asic-world.com/verilog/veritut.html`.

1.3 Circuit design in Verilog

1.3.1 Part I

The DE2 board provides 18 toggle switches, called SW_{17-0}, that can be used as inputs to a circuit, and 18 red lights, called $LEDR_{17-0}$, that can be used to display output values. Figure 1 shows a simple Verilog module that connects switches to red LEDs. Since there are 18 switches and lights, it is convenient to represent them as vectors in the Verilog code, as shown. We have used a single assignment statement for all 18 *LEDR* outputs, which would be equivalent to the following individual assignments:

```
assign LEDR[17] = SW[17];
assign LEDR[16] = SW[16];
...
assign LEDR[0] = SW[0];
```

The DE2 board has hardwired connections between its FPGA chip and the switches and lights. To use SW_{17-0} and $LEDR_{17-0}$ it is necessary to include in your Quartus II project the correct pin assignments, which are given in the *DE2 User Manual*. For example, the manual specifies that SW_0 is connected to the FPGA pin *N25* and $LEDR_0$ is connected to pin *AE23*. A good way to make the required pin assignments is to import into the Quartus II software the file called *DE2_pin_assignments.csv*, which has been uploaded to the BlackBoard site, in the labs section.

It is important to realize that the pin assignments in the *DE2_pin_assignments.csv* file are useful only if the pin names given in the file are exactly the same as the port names used in your Verilog module. The file uses the names *SW*[0] ... *SW*[17] and *LEDR*[0] ... *LEDR*[17] for the switches and lights, which is the reason we used these names in Figure 1.

```
// Simple module that connects the SW switches to the LEDR lights
module  part1 (SW, LEDR);
   input [17:0] SW;       // toggle switches
   output [17:0] LEDR; // red LEDs

   assign LEDR = SW;
endmodule
```

Figure 1. Verilog code that uses the DE2 board switches and lights.

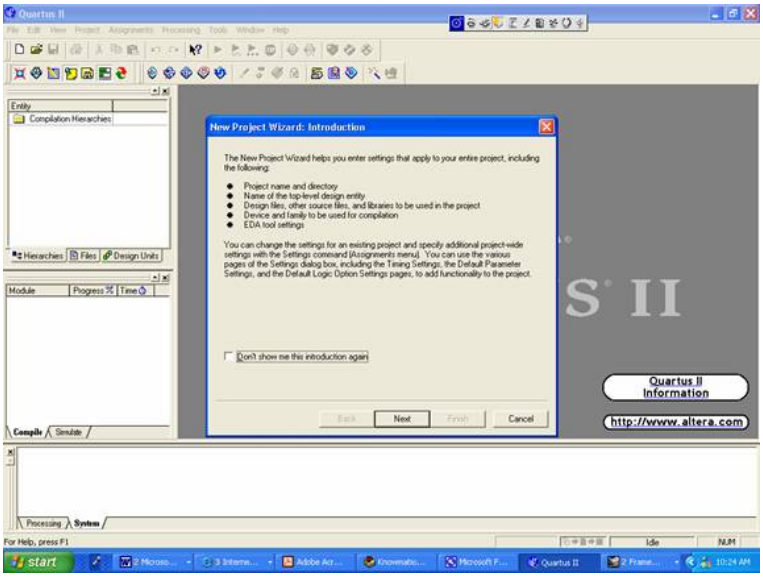
Perform the following steps to implement a circuit corresponding to the code in Figure 1 on the DE2 board.

1. Create a new Quartus II project for your circuit (see next page for detailed instructions). Select Cyclone II EP2C35F672C6 as the target chip, which is the FPGA chip on the Altera DE2 board.
2. Create a Verilog module for the code in Figure 1 and include it in your project.
3. Include in your project the required pin assignments for the DE2 board, as discussed above. Compile the project. Simulate and debug as appropriate (see pages 23 to 28 of the Quartus II Introduction document).

Your First Quartus Program (Detailed Instructions)

Here is a guide to getting your program compiled and simulated:

1. (a) Open the **File** menu, and select **New Project Wizard**. It will look like[1]:



   (b) Click Next. For the working directory of the project, select **...** and create a new folder, **lab1**. Once selected, your working directory should be **W:**
   lab1. Name the project **part1**, and the top-level design entity **part1** as well.

   (c) Click Next until you get a menu that asks for the devices. For the device family, select **Cyclone II**. In the available devices below, select **EP2C35F672C6**. Press Finish.

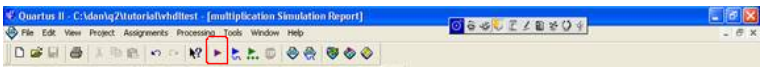
2. Now, open the **File** menu again, and select **New**. Select **Verilog HDL File** from the list. Save this file as **part1** to your lab1 directory, and select for the file to be added to your current project. It is important that this file have the same name as your top-level design entity! Now type/copy in your first program:

```
// Simple module that connects the SW switches to the LEDR lights
module  part1 (SW, LEDR);
   input [17:0] SW;       // toggle switches
   output [17:0] LEDR; // red LEDs

   assign LEDR = SW;
endmodule
```

3. Download the file **DE2_pin_assignments.csv** from the Blackboard website for CSC 258. Back in Quartus II, open the **Assignments** menu and select **Import Assignments...**. Provide the location of where you saved the pin assignment file and press OK.

4. Compile the program by pressing the purple triangle icon at the top:



5. To simulate your program, you will need to use either QSim or ModelSim to simulate the compiled version of your circuit. Instructions on how to use the simulator can be found on pages 23 to 28 of the Quartus II Introduction documents provided on Blackboard.

[1]Image credit for this section: `http://cse.spsu.edu/clo/research/vhdl/quartus2tutorial.htm`

1.3.2 Part II

Figure 2*a* shows a sum-of-products circuit that implements a 2-to-1 *multiplexer* with a select input s. If $s = 0$ the multiplexer's output m is equal to the input x, and if $s = 1$ the output is equal to y. Part b of the figure gives a truth table for this multiplexer, and part c shows its circuit symbol.

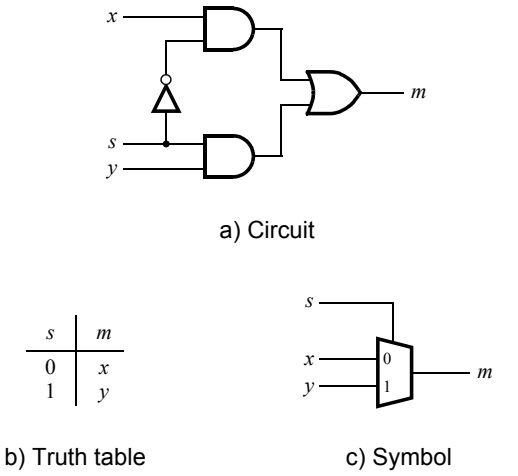


Figure 2: A 2-to-1 multiplexer.

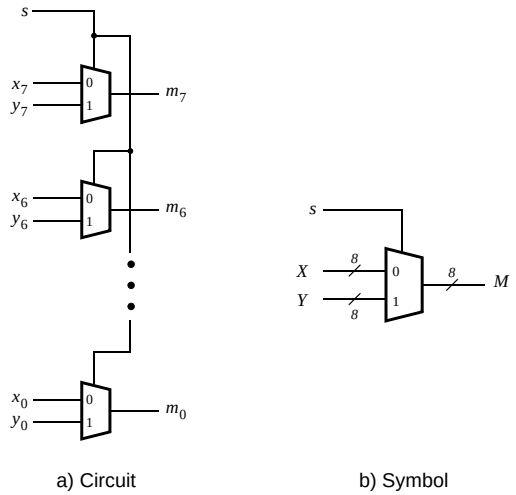


Figure 3: An eight-bit wide 2-to-1 multiplexer.

The 2-to-1 multiplexer can be described by the following Verilog statement:

assign m = (~s & x) | (s & y);

PRELAB TODO: Write a Verilog module that includes eight assignment statements like the one shown above to describe the circuit given in Figure 3*a*. This circuit has two eight-bit inputs, X and Y, and produces the eight-bit output M. If $s = 0$ then $M = X$, while if $s = 1$ then $M = Y$. We refer to this circuit as an eight-bit wide 2-to-1 multiplexer. It has the circuit symbol shown in Figure 3*b*, in which X, Y, and M are depicted as eight-bit wires. Perform the steps shown below.

1. Create a new Quartus II project for your circuit.
2. Include your Verilog file for the eight-bit wide 2-to-1 multiplexer in your project. Use switch SW_{17} on the DE2 board as the s input, switches SW_{7-0} as the X input and SW_{15-8} as the Y input. Connect the SW switchesto the red lights $LEDR$ and connect the output M to the green lights $LEDG_{7-0}$.
3. Include in your project the required pin assignments for the DE2 board. As discussed in Part I, these assignments ensure that the input ports of your Verilog code will use the pins on the Cyclone II FPGA that are connected to the SW switches, and the output ports of your Verilog code will use the FPGA pins connected to the $LEDR$ and $LEDG$ lights.
4. Compile the project. Simulate and debug as appropriate.

1.3.3 Part III

In Figure 2 we showed a 2-to-1 multiplexer that uses the value on input s to select which of the two inputs x and y will be transmitted through output m. For this part of the lab, consider a circuit in which the output m has to be selected from *five* inputs: u, v, w, x, and y. Part a of Figure 4 shows how we can build the required 5-to-1 multiplexer by using four 2-to-1 multiplexers.

In order to specify which input will be selected as an output to the multiplexer, we need more than one select bit. The 5-to-1 circuit uses a 3-bit select input $s_2s_1s_0$ and implements the truth table shown in Figure 4*b*. A circuit symbol for this multiplexer is given in part c of the figure.

This circuit is fine when the output is a single bit, but what if each of the 5 inputs are 3-bit binary numbers? Recall from Figure 3 that an eight-bit wide 2-to-1 multiplexer can be built by using eight instances of a 2-to-1 multiplexer. Figure 5 applies this concept to define a three-bit wide 5-to-1 multiplexer. It contains three instances of the circuit in Figure 4*a*. **PRELAB TODO:** Write in Verilog and simulate a 3-bit wide 5-to-1 multiplexer.

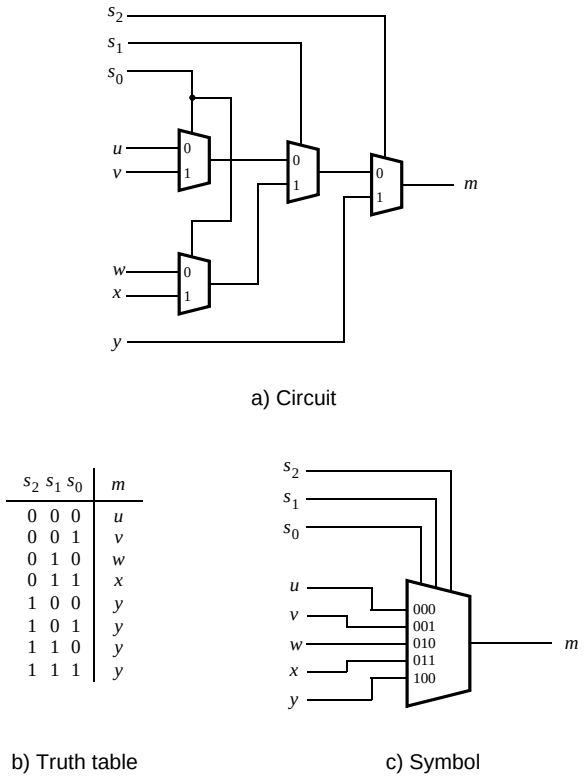


Figure 4: A 5-to-1 multiplexer.

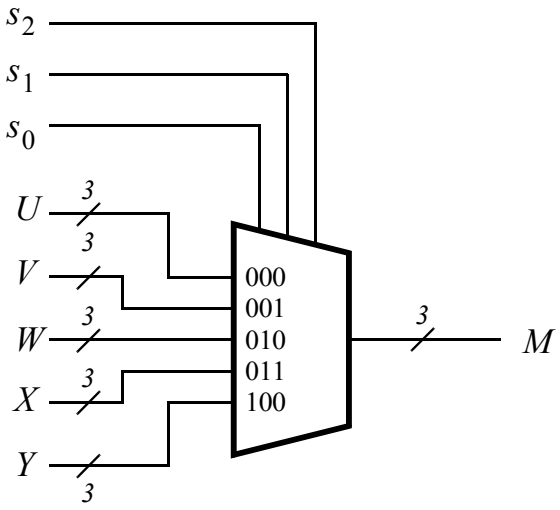


Figure 5: A three-bit wide 5-to-1 multiplexer.

Perform the following steps to implement the three-bit wide 5-to-1 multiplexer.

1. Create a new Quartus II project for your circuit.
2. Create a Verilog module for the three-bit wide 5-to-1 multiplexer. Connect its select inputs to switches SW_{17-15}, and use the remaining 15 switches SW_{14-0} to provide the five 3-bit inputs U to Y. Connect the SW switches to the red lights $LEDR$ and connect the output M to the green lights $LEDG_{2-0}$.
3. Include in your project the required pin assignments for the DE2 board. Compile the project. Simulate and debug as appropriate.

1.3.4 Part IV

Figure 6 shows a *7-segment* decoder module that has the three-bit input $c_2c_1c_0$. This decoder produces seven outputs that are used to display a character on a 7-segment display. Table 1 lists the characters that should be displayed for each valuation of $c_2c_1c_0$. To keep the design simple, only four characters are included in the table (plus the 'blank' character, which is selected for codes $100 - 111$).

The seven segments in the display are identified by the indices 0 to 6 shown in the figure. Each segment is illuminated by driving it to the logic value 0. **PRELAB TODO:** Write a Verilog module that activates every segment of the 7-segment display based on the 3-bit input, according to Table-1. Use only simple Verilog **assign** statements in your code to specify each logic function using a Boolean expression.

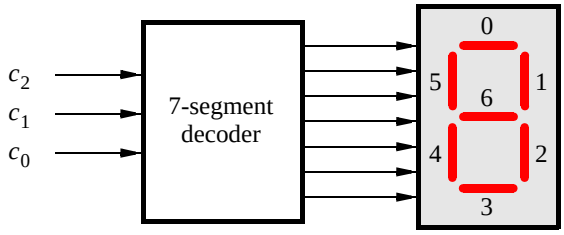


Figure 6: A 7-segment decoder.

$c_2c_1c_0$	Character
000	H
001	A
010	P
011	P
100	Y
101	
110	
111	

Table 1. Character codes.

Perform the following steps:

1. Create a new Quartus II project for your circuit.
2. Create a Verilog module for the 7-segment decoder. Connect the $c_2c_1c_0$ inputs to switches SW_{2-0}, and connect the outputs of the decoder to the *HEX0* display on the DE2 board. The segments in this display are called *HEX0*[0], *HEX0*[1], ..., *HEX0*[6], corresponding to Figure 6. You should declare the 7-bit port

   output [0:6] HEX0;

   in your Verilog code so that the names of these outputs match the corresponding names in the *DE2 User Manual* and the *DE2_pin_assignments.csv* file.
3. After importing the required DE2 board pin assignments, compile the project. Simulate and debug as needed.

1.3.5 Part V

Consider the circuit shown in Figure 7. It uses a three-bit wide 5-to-1 multiplexer to select one out of five characters specified by Switches, and displays it on a single 7-segment. Using the 7-segment decoder from Part IV this circuit can display any of the characters H, A, P, Y and 'blank'. The character codes are set according to Table 1 by using the switches SW_{14-0}, and a specific character is selected for display by setting the switches SW_{17-15}.

An outline of the Verilog code that represents this circuit is provided in Figure 8. Note that we have used the circuits from Parts III and IV as subcircuits in this code. **PRELAB TODO:** You are to complete the code in Figure 8 so that it uses one 7-segment display, and a 5-to-1 multiplexer to select which input to display.

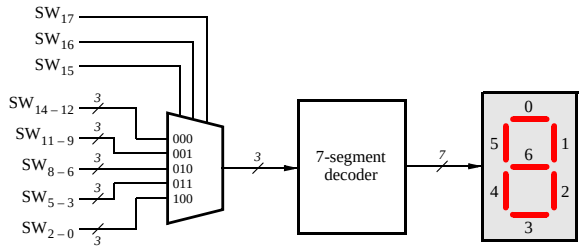


Figure 7: A circuit that can select and display one of five characters on a single 7-segment display.

```
module  part5 (SW, HEX0);
   input [17:0] SW;       // toggle switches
   output [0:6] HEX0;   // 7-seg displays

   wire [2:0] M;

   mux_3bit_5to1 M0 (SW[17:15], SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], M);
   char_7seg H0 (M, HEX0);
endmodule

// implements a 3-bit wide 5-to-1 multiplexer
module mux_3bit_5to1 (S, U, V, W, X, Y, M);
   input [2:0] S, U, V, W, X, Y;
   output [2:0] M;

   ... code not shown

endmodule

// implements a 7-segment decoder for H, A, P, Y and 'blank'
module char_7seg (C, Display);
   input [2:0] C;           // input code
   output [0:6] Display; // output 7-seg code

   ... code not shown

endmodule
```

Figure 8. Verilog code for the circuit in Figure 7.

Perform the following steps.

1. Create a new Quartus II project for your circuit.

2. Include your Verilog module in the Quartus II project. Connect the switches SW_{17-15} to the select inputs of the three-bit wide 5-to-1 multiplexer. Also connect SW_{14-0} to the multiplexers as required. Connect the output of the multiplexer to the 7-segment decoder and subsequently to the 7-segment display, *HEX0*.

3. Include the required pin assignments for the DE2 board.

4. Compile, simulate and debug the project.

1.4 Part VI: Eight Character Rotator (Advanced)

For the advanced part of this lab, extend the code of Part VI so that it uses eight 7-segment displays rather than just one. You will need to use eight instances of the subcircuit. The purpose of your circuit is to rotate five characters specified by SW_{14-0} over the eight 7-segment displays in a circular fashion. Rotation is controlled by the switches SW_{17-15}. As an example, if the word specified is HAPPY, then your circuit should produce the output patterns illustrated in Table 2.

SW_{17} SW_{16} SW_{15}	Character pattern							
000				H	A	P	P	Y
001			H	A	P	P	Y	
010		H	A	P	P	Y		
011	H	A	P	P	Y			
100	A	P	P	Y				H
101	P	P	Y				H	A
110	P	Y				H	A	P
111	Y				H	A	P	P

Table 2. Rotating the word HAPPY on eight displays.

Compile, download, test and debug your code for this part. Test the rotation functionality of the circuit by toggling the select signals $SW17-15$ and setting the characters of the word HAPPY with switches $SW14-0$.

2 In lab work

When you come to the lab, upload the designs from the prelab and verify them on the DE2 board. Make sure each part works before you verify the next part, and show your working implementation to the TAs where indicated. If you are unable to complete the entire lab within the allotted time, make sure that you show the parts that you were able to complete to the TA before the end of the lab session.

2.1 Part I: Switches to LEDs

1. Log into the computer at your station using your utorid.
2. Open up Quartus II 11.1. You will find the icon in: Start menu - All Programs - Altera 11.1 Build X - Quartus II 11.1 - Quartus II 11.1

   (a) Open up your Quartus project for Part 1 of the pre-lab assignment. Compile it (see page 3).
   (b) Now, we want to download our compiled file to the DE2 board. To do so, go to "Tools" menus, then click "Programmer".

      i. You will see a pop-up window when you do this. Ensure it says **USB-Blaster [USB-0]** at the top. If it says no hardware, select the Hardware Setup button and select USB-Blaster there. If that option is not available, then check that your board is turned on, and that a grey USB cable is plugged into the "Blaster" port on the upper left of the board.
      ii. Once the file is downloaded to the device, test the device. Turn the switches on and off. For swiches in the 'on' position, the LED above it should be on, and vice versa.

INLAB TODO: Show your board to your TA once you have completed and verified this part.

2.2 Part II: Eigth-bit wide 2-to-1 Mux

Compile and download your program for Part-II of your prelab to your board. Test and debug it until you are certain it behaves as you expect.

2.3 Part III: Three-bit wide 5-to-1 Mux

Compile, download, test and debug your code for a three-bit wide 5-to-1 mux from Part-III of your prelab. Test the functionality of the three-bit wide 5-to-1 multiplexer by toggling the switches and observing the LEDs. Ensure that each of the inputs U to Y can be properly selected as the output M.
INLAB TODO: Show your board to your TA once you have completed and verified this part.

2.4 Part IV: Three-bit 7-Segment Decoder

Compile, download, test and debug your code for a three-bit input decoder from Part-IV of your prelab. Test the functionality of the circuit by toggling the SW_{2-0} switches and observing the 7-segment display. You must be able to display all four characters of H, A, P, Y and blank.

2.5 Part V: Five inputs, one 7-Segment Display

Compile, download, test and debug your code from Part-V of your prelab. Test the functionality of the circuit by setting different character codes on the switches SW_{14-0} and then selecting different inputs to display on the 7-Segment.

2.6 Part VI: Eight Character Rotator (Advanced)

Compile, download, test and debug your code from Part-VI of your prelab. Test the functionality of the circuit by setting different binary values on switches SW_{17-15} and confirming that the correct output appears on the 7-Segment display.
INLAB TODO: Show your board to your TA once you have completed and verified this part.