ECE241 - Digital Systems

University of Toronto

Lab #2 - Fall 2008

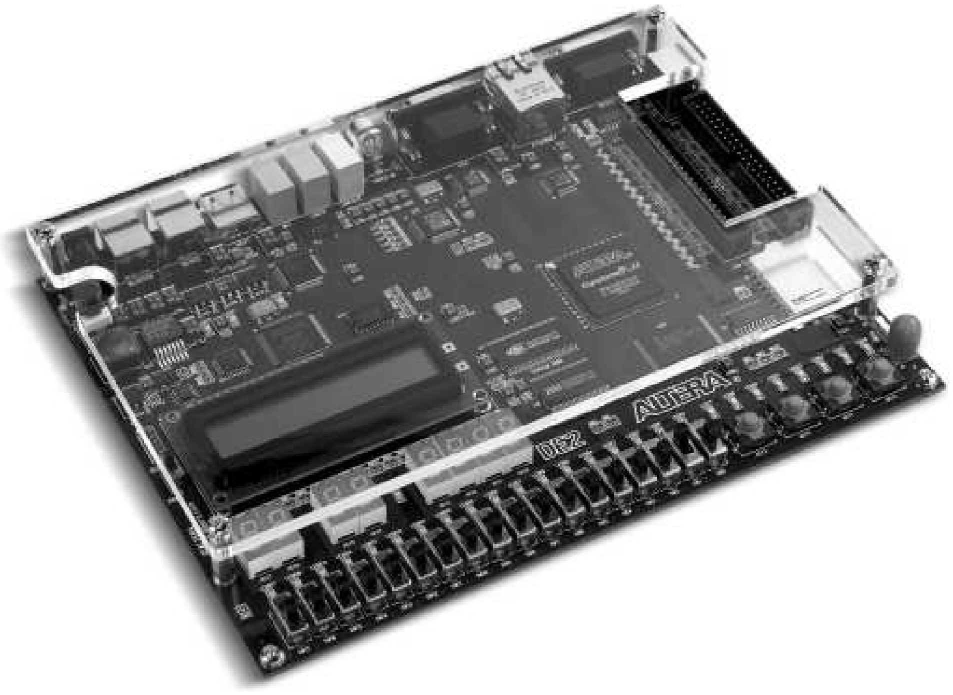
# Introduction Computer-Aided Design Software, the DE2 Board and Simple Logic

## 1. Introduction

The purpose of this exercise is to introduce you to the software tools and hardware that are used in the labs for this course. The main software tool is the Altera Quartus II Computer Aided Design (CAD) system. You will need to install that software on your computer (or use one of the University's computers) before you can start this lab. The "First Handout" given on the course web page has a section on how to download the software and obtain a license.

You will be learning to design hardware that, in the labs, will go into a kind of programmable logic chip call a Field-Programmable Gate Array, or FPGA. The FPGA chip is mounted on a board called the Altera DE2 Development and Education board, pictured below. This board will be used for all of the remaining lab exercises in the course, and for the course project.

The DE2 board contains many useful features for learning about logic circuits, including simple input and output mechanisms like switches and lights, and more complicated features like audio and video devices. This lab exercise will use only the switches and lights that are provided on the bottom edge of the board, as illustrated below, but other lab exercises and the course project will utilize more advanced features. A detailed description of the DE2 board can be found at: http://www.altera.com/education/univ/materials/boards/unv-de2-board.html.



The Altera DE2 Development and Education board.

## 2. Preparation - This must be done prior to lab period!

There are two main parts to the preparation for this exercise: learning the Quartus software basics by doing the tutorials (which we estimate will take about 2 hours) and designing some logic circuits and using the software to be ready to test them in the lab. **You must do the tutorials and the circuit design before coming to the lab. You will be required to present the results of this work for marking at the beginning of the lab.**

1. Do the tutorial called *Using Quartus II CAD Software*, which you can find in Appendix B on page 823 of the course textbook **Fundamentals of Digital Logic with Verilog Design, 2$^{nd}$ Edition**. This edition of the textbook must be used because the old edition is for a completely different version of the software.

   This tutorial describes the basics of how Quartus II helps a designer describe circuits and check them for correctness. It shows two ways of creating circuits: either using schematic capture, in which you "draw" a picture of the circuit, or describing a circuit in language form. We ask that you do the same circuit in both ways so that you understand the schematic for and textual form describe exactly the same thing. We will use the textual form often because it is far more powerful and quicker, but it is easy to confuse with software, and so we repeat: the language is a way to describe hardware circuits. By using both in this tutorial, hopefully this will be clear.
2. Do the first part of the tutorial called *Implementing Circuits in Altera Devices*, which is in Appendix C.1 of the textbook on page 855.
3. Do the tutorial *Physical Implementation in an FPGA*, which is found in Appendix D on 885.
4. Do the circuit design work indicated in each of Parts II to V in Section 3 below. Create these circuits using the Verilog method of description. Bring a printed copy (pasted into your lab book) for all four circuits. Warning: this is quite a bit of work, so be sure to leave about at least 2 hours for this part alone.

All of the following material **must be ready for marking at the beginning of the lab, in your lab book:**

1. A print-out of the schematic diagram *and* simulation results for the circuit described in Appendix B that you created.
2. A print-out of the Quartus II project file (this file has the extension *.qpf*) for your schematic-based project
3. Your Verilog source code for the circuit in Appendix B
4. A print-out of the Quartus II *project file* for your Verilog-based project in Appendix B
5. A print-out of the Quartus II *project file* corresponding to section C.1
6. A print-out of the Quartus II *settings file* (this file has the extension *.qsf*) for the project created in section D.2

**It is strongly suggested that you simulate all of the circuits that you design, to ensure that they are correct. A key learning in this course is the skill of debugging hardware.**

## 3. Lab Work (This requires the preparation indicated above)

In the lab you will have to implement and test circuits that you created in the preparation. The in-lab component of your lab grade will be based only on the circuits you create in parts IV and V of this exercise. However, to be able to do those parts, you'll have to have done parts I through III. Please inform your TA when you have completed **both** parts IV and V to obtain your grade.

To simplify some of the steps, a *starter kit* is provided on course website, located in the directory:

`http://www.eecg.utoronto.ca/~jayar/ece241_08F/Lab2_starterkit.zip`

The starter kit is a ZIP archive containing a Quartus II project that you will need for each part of the lab. Unzip the archive into a working directory called *lab2*. If you use Windows Explorer to view the contents of the *lab2* directory, you should see the folders shown in Figure 2.

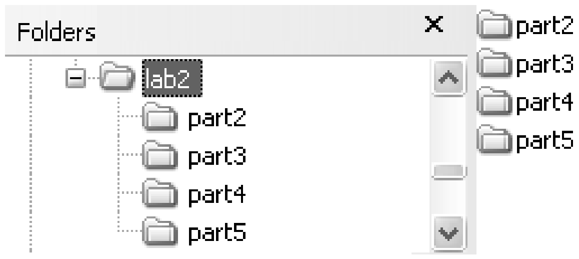


Figure 2. The contents of the starter kit.

**Part I**

Download the circuits you designed in the tutorial preparation onto the Altera DE2 board and test to see that they work. Remember to set the device to EP2C35F672C6 (this is the device on the Altera DE2 board).

**Part II**

Create a simple circuit to connect four switches to four lights on the Altera DE2 board, by extending the following Verilog code:

```
// Simple module that connects the SW switches to the LEDR lights
module part2 (Switch_1, Switch_2, Switch_3, Switch_4,
              Light_1, Light_2, Light_3, Light_4);
   input Switch_1, Switch_2, Switch_3, Switch_4; // toggle switches
   output Light_1, Light_2, Light_3, Light_4; // lights

   // Your code goes here
endmodule
```

Figure 3. Verilog code for Part II.

On the DE2 board the FPGA chip that your designs will be programmed into has hardwired connections between the pins of the FPGA chip and the switches and lights on the board. Note that these lights and switches are connected to circuits that allow them to generate 1s and 0s as inputs to your FPGA circuit (for the switches) and to turn on and off in response to digital 1s and 0s that are outputs from your circuit. Later in this course we will describe how those kinds of circuits work.

To use switches and lights you have to tell Quartus II which of the input/output signals in your Verilog code should be connected to which pins on the FPGA chip and which are connected to the switches or lights. The procedure for doing this is called *pin assignment* and was covered in the tutorial in Appendix D of the textbook.

Table 1 below indicates which pins (which are referred to by names such as PIN_N25) of the FPGA are connected to which switches and lights on the board. There are 18 total switches and lights on the board, but the table only lists 4 of each.

| Function | Altera FPGA Cyclone II Pin on the DE2 |
|---|---|
| Switch_1 | PIN_N25 |
| Switch_2 | PIN_N26 |
| Switch_3 | PIN_P25 |
| Switch_4 | PIN_AE14 |
| Light_1 | PIN_AE23 |
| Light_2 | PIN_AF23 |
| Light_3 | PIN_AB21 |
| Light_4 | PIN_AC22 |

Table 1: Pin assignment table for lights and switches in Part II.

Do the following steps to describe, download and test the circuit that connects the 4 lights and switches:

1. The project for this part is provided in the starter kit that you downloaded from the website given above. Open in Quartus II (using the command File > Open Project) the project named *part2.qpf* in the *part2* subdirectory to begin your work.

2. Create a Verilog module named *part2* for the code in Figure 3 and include it in your project. Make sure to complete the code by adding the assignment statements for the lights.

3. Use Quartus II to make the pin assignments shown in the above table as described in Appendix D. Compile the project.

4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by flipping the switches and observing the lights.

**Part III**

The DE2 board provides 18 toggle switches, which we will now refer to as $SW_{17-0}$, that can be used as inputs to a circuit, and 18 red lights, referred to as $LEDR_{17-0}$, that can be used to display output values. Figure 4 gives a Verilog module that uses these switches and connects them to the lights. Since there are 18 switches and lights it is much easier to represent them as "vectors" in the Verilog code, desclared as, for example: **input [17:0]W** as shown. This allows us to use just one single assignment statement: **assign LEDR = SW** to connect all 18 *LEDR* outputs to all 18 switches, and is equivalent to the following 18 individual assignments:

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

To use $SW_{17-0}$ and $LEDR_{17-0}$ in this way, it is still necessary to assign pins on the FPGA chip that connect to these input and output signals, as we did for the pins used in Part II. Although we could enter these pin assignments manually (18 for the switches plus 18 for the lights), we have provided a pin assignment file that already has these names assigned to the correct pins. To make use of these assignments, you must import into the Quartus II software a file called *DE2_pin_assignments.csv*, which is included in the starter kit you downloaded above. The procedure for making pin assignments by importing the *.csv* file are included in the tutorial in Appendix D of the textbook on pages 890 and 891.

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 this file are exactly the same as the input and output names used in your Verilog module. The *.csv* file uses the names *SW*[0] . . . *SW*[17] and *LEDR*[0] . . . *LEDR*[17] for the switches and lights, which is the reason we have used these names in Figure 4.

```
// Simple module that connects the SW switches to the LEDR lights
module part3 (SW, LEDR);
    input [17:0] SW;        // toggle switches
    output [17:0] LEDR;     // red LEDs

    assign LEDR = SW;
endmodule
```

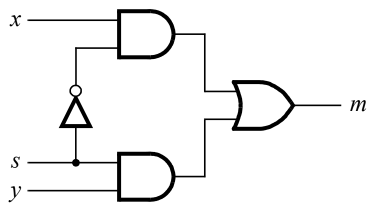
Figure 4. Verilog code that uses the DE2 board switches and lights.

Do the following steps:

1. The project for this part is provided in the starter kit. Open the project named *part3.qpf* in the *part3* subdirectory to begin your work.
2. Create a Verilog module called *part3* for the code in Figure 4 and include it in your project.
3. Include in your project the required pin assignments for the DE2 board, as described above. Compile the project.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by toggling the switches and observing the lights (which are actually LEDs, Light Emitting Diodes).

## Part IV

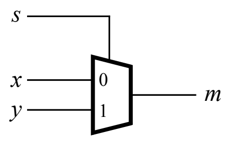
Figure 5$a$ shows a sum-of-products circuit that implements a 2-to-1 *multiplexer* of inputs $x$ and $y$ select input $s$ and output $m$. If $s = 0$ the multiplexer's output $m$ is equal to the input $x$, and if $s = 1$ the output is equal to $y$. Part $b$ of the figure gives a truth table for this multiplexer, and part $c$ shows the schematic circuit symbol.



a) Circuit

| $s$ | $m$ |
|---|---|
| 0 | $x$ |
| 1 | $y$ |

b) Truth table



c) Symbol

Figure 5. A 2-to-1 multiplexer.

The multiplexer can be described by the following Verilog statement:

```
assign m = (∼s & x) | (s & y);
```

You are to design a circuit, using Verilog, that is a more complex version of a multiplexer. Rathern that select between two signals, your circuit is to select between two sets of *eight* signals, as illustrated in Figure 6$a$. 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 6$b$, in which $X$, $Y$, and $M$ are depicted as eight-bit wires. Perform the steps shown below.

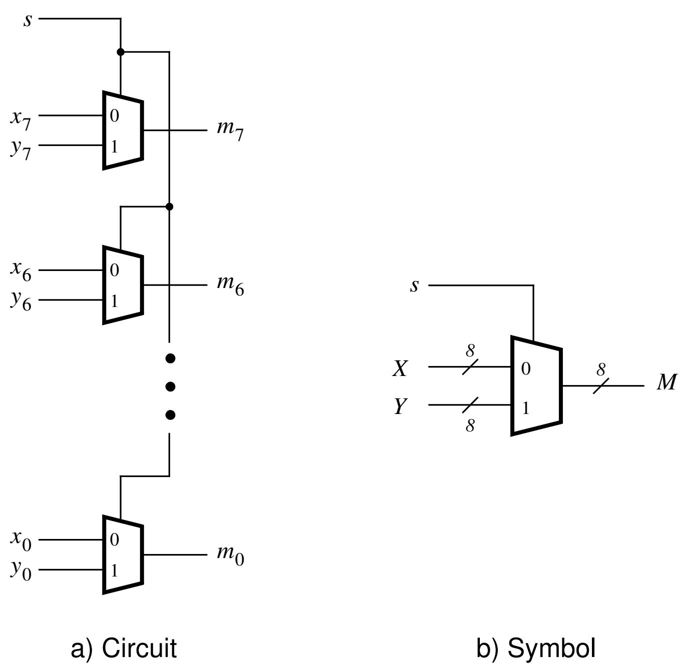


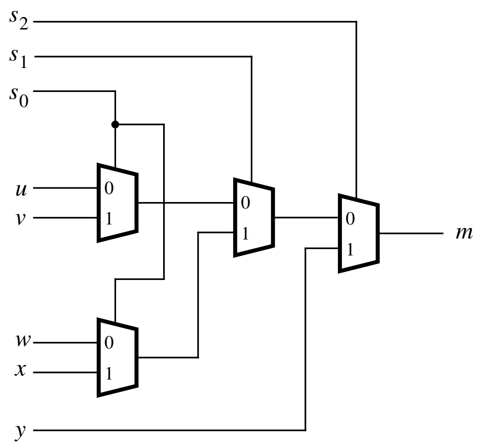
Figure 6. An eight-bit wide 2-to-1 multiplexer.

1. The project for this part is provided in the starter kit. Open the project named *part4.qpf* in the *part4* subdirectory to begin your work.
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* switches to the red lights *LEDR* and connect the output $M$ to the green lights on the DE2 board, called *LEDG*$_{7-0}$.
3. Include in your project the required pin assignments for the DE2 board using the *DE2_pin_assignments.csv* as described above. As discussed in Parts II and III, these assignments ensure that the inputs declared in your Verilog code will use the pins on the Cyclone II FPGA that are connected to the *SW* switches, and the outputs of your Verilog code will use the FPGA pins connected to the *LEDR* and *LEDG* lights.
4. Compile the project.
5. Download the compiled circuit into the FPGA chip. Test the functionality of the eight-bit wide 2-to-1 multiplexer by toggling the switches and observing the LEDs.

**Part V**

In Figure 5 we showed a 2-to-1 multiplexer that selects between the two inputs *x* and *y*. For this part 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 7 shows how we can build the required 5-to-1 multiplexer by using four 2-to-1 multiplexers. The circuit uses a 3-bit select input $s_2s_1s_0$ and implements the truth table shown in Figure 7$b$. A circuit symbol for this multiplexer is given in part $c$ of the figure.

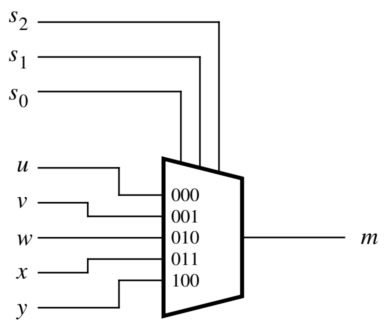
Recall from Figure 6 that an eight-bit wide 2-to-1 multiplexer can be built by using eight 2-to-1 multiplexers. Figure 8 applies this concept to define a three-bit wide 5-to-1 multiplexer. It contains three instances of the 5-to-1 multiplexer circuit in Figure 7.



a) Circuit

| $s_2$ | $s_1$ | $s_0$ | $m$ |
|---|---|---|---|
| 0 | 0 | 0 | $u$ |
| 0 | 0 | 1 | $v$ |
| 0 | 1 | 0 | $w$ |
| 0 | 1 | 1 | $x$ |
| 1 | 0 | 0 | $y$ |
| 1 | 0 | 1 | $y$ |
| 1 | 1 | 0 | $y$ |
| 1 | 1 | 1 | $y$ |

b) Truth table



c) Symbol

Figure 7. A 5-to-1 multiplexer.

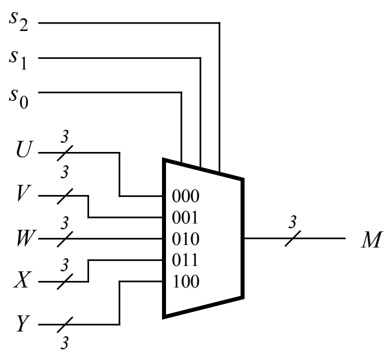


Figure 8. A three-bit wide 5-to-1 multiplexer.

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

1. The project for this part is provided in the starter kit. Open the project named *part5.qpf* in the *part5* subdirectory to begin your work.
2. Create a Verilog module for the three-bit wide 5-to-1 multiplexer. Connect its select inputs to switches $\text{SW}_{17-15}$, and use the remaining 15 switches on the Altera DE2 board ($\text{SW}_{14-0}$) to provide the five 3-bit inputs $U$ through $Y$. Connect the *SW* switches to the red lights *LEDR* and connect the output $M$ to the green lights $\textit{LEDG}_{2-0}$.
3. Include in your project the required pin assignments for the DE2 board. Compile the project.
4. Download the compiled circuit into the FPGA chip. 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 $M$.