Laboratory Exercise 1
Switches, Lights, and Multiplexers

TCES 330 Digital Systems Design
Spring 2009

Authors:

Robert Gutmann _____

Larry Wear _____

Submission Date: April 7, 2009

Table of Contents

Laboratory Assignment 1

The purpose of this exercise is to learn how to connect simple input and output devices to an FPGA chip and implement a circuit that uses these devices. 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.

Requirements:

The objectives of each part of the lab assignment are described in this section of the report.

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 uses these switches and shows their states on the 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 is equivalent to the 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 is provided on the *DE2 System CD* and in the University Program section of Altera's web site. The procedure for making pin assignments is described in the tutorial *Quartus II Introduction using Verilog Design*, which is also available from Altera 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. 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.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by toggling the switches and observing the LEDs.

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.

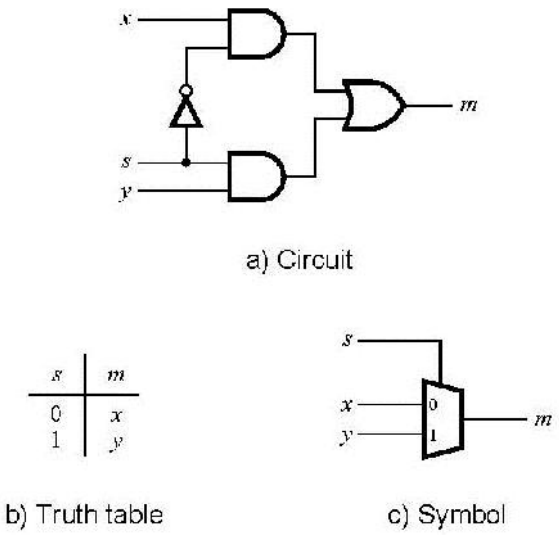

Figure 2. A 2-to-1 Multiplexer.

.

The multiplexer can be described by the following Verilog statement:

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

You are to 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.

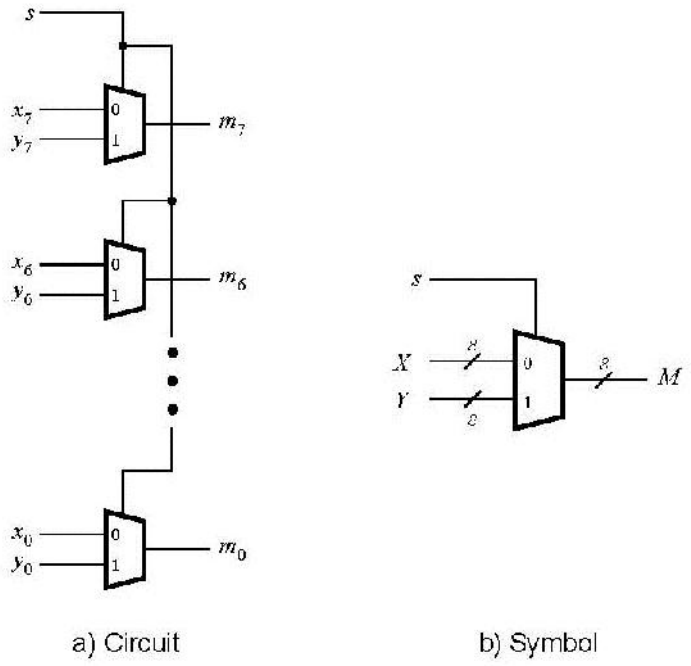

Figure 3. An eight-bit wide 2-to-1 multiplexer

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* switches to 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.
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 III

…

Part IV

…

Part V

…

Part VI

…

Design

This section of the report describes our analysis of the requirements for this laboratory exercise and the resulting project design.

Part I

…

Part II

The first sub-task is to write a Verilog module with the following properties which have been derived from the Requirements section:

1. Has two eight-bit inputs, X and Y.
2. Produces an eight-bit output, M.
3. If the additional input S = 0 then M = X, while if S = 1 then M = Y.
4. Is implemented by eight **assign** statements.

The following Verilog module accomplishes this sub-task:

```
Module Mux8( S, X, Y, M );
        input S;
        input [7:0] X, Y;
        output [7:0] M;

        // the mux
        assign M[0] = (~S & X[0] ) | (S & Y[0] );
        assign M[1] = (~S & X[1] ) | (S & Y[1] );
        assign M[2] = (~S & X[2] ) | (S & Y[2] );
        assign M[3] = (~S & X[3] ) | (S & Y[3] );
        assign M[4] = (~S & X[4] ) | (S & Y[4] );
        assign M[5] = (~S & X[5] ) | (S & Y[5] );
        assign M[6] = (~S & X[6] ) | (S & Y[6] );
        assign M[7] = (~S & X[7] ) | (S & Y[7] );
```

endmodule

This module is stored in the file Mux8.v.

The next sub-task is to create a Quartus II project and associated Verilog file that contains the module Mux8. This Verilog file (module) has the following properties:

1. Includes the Mux8 module.
2. Uses SW_{17} as the S input.
3. Uses SW_{7-0} as the X input.
4. Uses SW_{15-8} as the Y input.
5. Connects the SW switches to the lights LEDR
6. Connects the output M to the lights $LEDG_{7-0}$.

The file Lab1_Part2.v contains the module Lab1_Part2() that satisfies these requirements. A listing can be found in Appendix A. Also note that the necessary pin assignments can be found in the file Lab1_Part2.csv, also listed in Appendix A.

All of the above is in the Quartus II project named Lab1_Part2 contained in a folder of the same name.

Part III

…

Part IV

…

Part V

…

Part VI

…

Test Procedures

The following test procedures will be used to verify that each part of this laboratory exercise satisfies the requirements given in the Requirements section, above.

Part I

…

Part II

The following test procedure will be used to verify that the Quartus II project Lab1_Part2 satisfies the requirements for this part.

1. The eight-wide 2 to 1 multiplexer circuit will be tested using ModelSim and the testbench shown in Figure 9, Appendix A. This testbench generates the test vectors shown in Table 1 and outputs the multiplexer output M. Simulations will be run in order to verify the behavior shown in Table 1.
2. Open the project and verify that compilation produces no errors or un-allowed warnings.
3. Verify that the RTL Viewer shows a multiplexer circuit.
4. Load the project onto the DE2 board without errors.
5. Verify that the switches SW are connected to the lights LEDR by turning each switch ON then OFF, verifying that the corresponding LED turns ON then OFF.
6. Generate the test vectors shown in Table 1and verify the corresponding outputs, M.

Table 1. Part 2 Text Vectors

$X = SW_{7-0}$	$Y = SW_{15-8}$	$S = SW_{17}$	$M=LEDG_{7-0}$
00000000	11111111	0	00000000
00000000	11111111	1	11111111
11111111	00000000	0	11111111
11111111	00000000	1	00000000
10101010	01010101	0	10101010
10101010	01010101	1	01010101

Part III

…

Part IV

…

Part V

…

Part VI

…

Test Results

Part I

…

Part II

The following observations correspond to the numbers in the Test Procedures section for this part:

1. The simulation using ModelSim produced the output shown in Figure 4. This output corresponds to Table 4 for each row, indicating the simulation passes.
2. Compilation was successful, with the Flow Summary shown in Figure 5. Flow Summary for Lab1_Part2Figure .
3. The RTL View produced the result shown in Figure . This is a reasonable result as it depicts eight repetitions of the circuit shown in Figure 2a.
4. The project was downloaded to the DE2 board without errors.
5. The lights LEDR turned ON and OFF with the corresponding switches as specified in the test procedure.
6. All test vectors as specified in Table 1 produced the corresponding outputs M.

```
#   X            Y            S        M
# 00000000    11111111      0       00000000
# 00000000    11111111      1       11111111
# 11111111    00000000      0       11111111
# 11111111    00000000      1       00000000
# 10101010    01010101      0       10101010
# 10101010    01010101      1       01010101
```

Figure 4. ModelSim Output

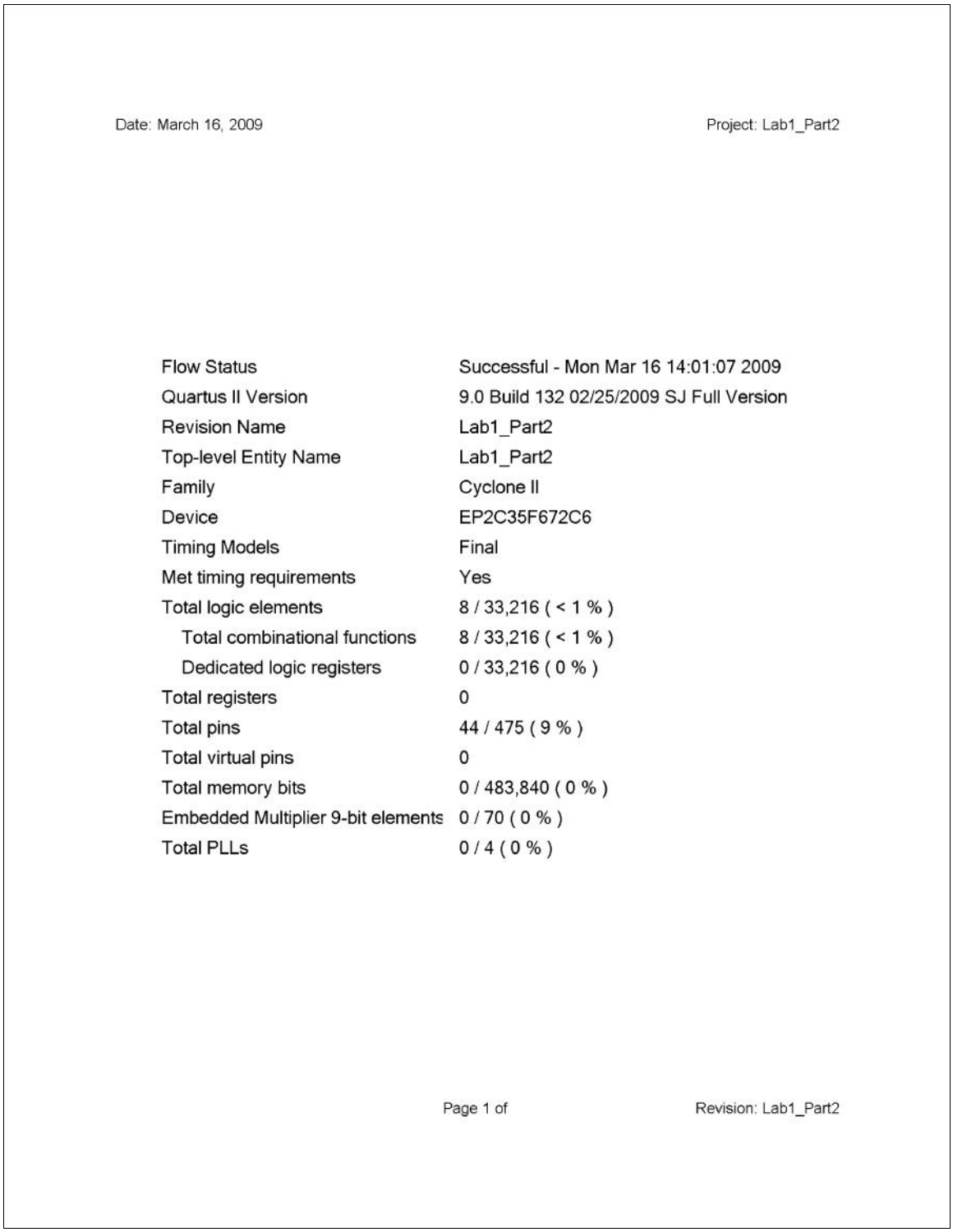

Figure 5. Flow Summary for Lab1_Part2

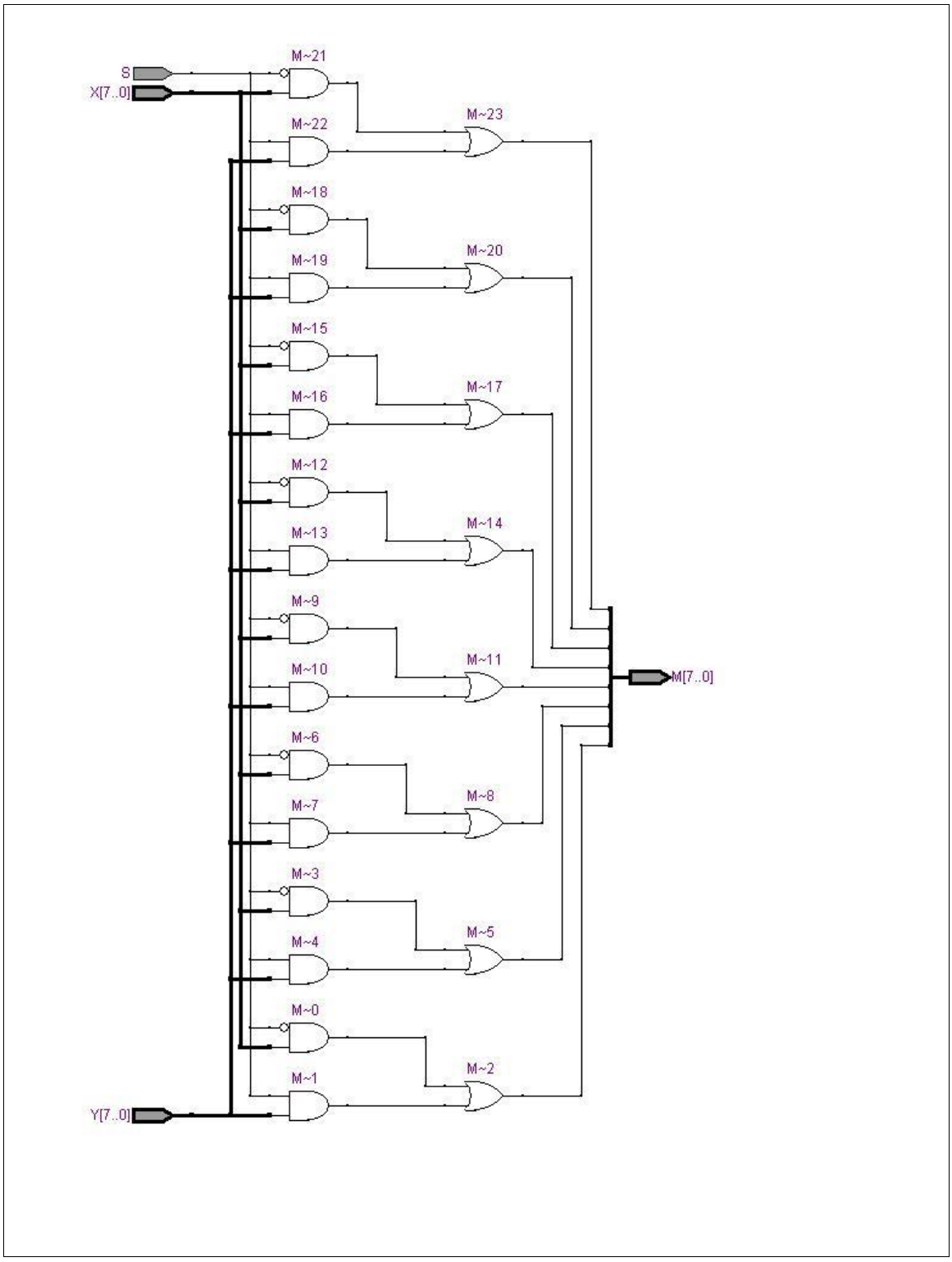

Figure 6. RTL View for Lab1_Part2

Part III

…

Part IV

…

Part V

…

Part VI

…

Observations

Part I

…

Part II

Several capitalization errors had to be corrected in order for Mux8.v to compile, but otherwise there were no issues in satisfying the test procedures. All tests were successful as defined by the Test Procedures section.

Part III

…

Part IV

…

Part V

…

Part VI

…

Conclusion

…

Appendix A

This Appendix contains listings of the files used in Part II of this laboratory exercise.

```
// TCES 330, Spring 2009
// Date: March 16, 2009
// Author; R. Gutmann

// Mux8 implements a 2-to-1 eight-bit wide mux
// Originally written for Lab 1, Part 2.

module Mux8( S, X, Y, M );
  input S;           // mux select line in
  input  [7:0] X, Y; // mux inputs
  output [7:0] M;    // mux output

  // the mux:
  assign M[0] = (~S & X[0]) | (S & Y[0]);
  assign M[1] = (~S & X[1]) | (S & Y[1]);
  assign M[2] = (~S & X[2]) | (S & Y[2]);
  assign M[3] = (~S & X[3]) | (S & Y[3]);
  assign M[4] = (~S & X[4]) | (S & Y[4]);
  assign M[5] = (~S & X[5]) | (S & Y[5]);
  assign M[6] = (~S & X[6]) | (S & Y[6]);
  assign M[7] = (~S & X[7]) | (S & Y[7]);

endmodule
```

Figure 7. Listing of Mux8.v

```
// TCES 330, Spring 2009
// Date: March 16, 2009
// Author; R. Gutmann

// Lab1_Part2 instantiates a 2-to-1 eight-bit wide mux
// and hooks it up to switches for inputs and
// lights for output.
// Originally written for Lab 1, Part 2.

module Lab1_Part2( SW, LEDR, LEDG );
input  [17:0] SW;    // toggle switches
output [17:0] LEDR;  // red LEDs
output [7:0]  LEDG;  // green LEDs
wire S;              // the select line

assign S = SW[17];   //
assign LEDR = SW;

// the mux:
Mux8 U1( S, SW[7:0], SW[15:8], LEDG[7:0] );

endmodule
```

Figure 8. Listing of Lab1_Part2

```
// TCES 330, Spring 2009
// Date: March 16, 2009
// Author; R. Gutmann
//
// This module tests a 2-to-1 eight-bit wide mux
// Originally written for Lab 1, Part 2.

module TestMux8;

  reg S;           // mux select line in
  reg  [7:0] X, Y; // mux inputs
  wire [7:0] M;    // mux output

  // the mux:
  Mux8 DUT( S, X, Y, M );

  initial begin
        X = 0;
        Y = 8'hFF;
        S = 0;
    #50 S = 1;
    #50 X = 8'hFF; Y = 0; S = 0;
    #50 S = 1;
    #50 X = 8'hAA; Y = 8'h55; S = 0;
    #50 S = 1;
    #50 $stop;
  end  // initial

  initial  begin
    $display( "  X\t\t  Y\t\tS\t  M" );
    $monitor( "%b\t%b\t%b\t%b", X, Y, S, M );
  end

endmodule
```

Figure 9. Listing of Testbench for Mux8

```
# Quartus II Version 5.1 Internal Build 160 09/19/2005 TO Full Version,
# File: D:\de2_pins\de2_pins.csv,
# Generated on: Wed Sep 28 09:40:34 2005,

# Note: The column header names should not be changed if you wish to import
this .csv file into the Quartus II software.,

To,Location
SW[0],PIN_N25
SW[1],PIN_N26
SW[2],PIN_P25
SW[3],PIN_AE14
SW[4],PIN_AF14
SW[5],PIN_AD13
SW[6],PIN_AC13
SW[7],PIN_C13
SW[8],PIN_B13
SW[9],PIN_A13
SW[10],PIN_N1
SW[11],PIN_P1
SW[12],PIN_P2
SW[13],PIN_T7
SW[14],PIN_U3
SW[15],PIN_U4
SW[16],PIN_V1
SW[17],PIN_V2
LEDR[0],PIN_AE23
LEDR[1],PIN_AF23
LEDR[2],PIN_AB21
LEDR[3],PIN_AC22
LEDR[4],PIN_AD22
LEDR[5],PIN_AD23
LEDR[6],PIN_AD21
LEDR[7],PIN_AC21
LEDR[8],PIN_AA14
LEDR[9],PIN_Y13
LEDR[10],PIN_AA13
LEDR[11],PIN_AC14
LEDR[12],PIN_AD15
LEDR[13],PIN_AE15
LEDR[14],PIN_AF13
LEDR[15],PIN_AE13
LEDR[16],PIN_AE12
LEDR[17],PIN_AD12
LEDG[0],PIN_AE22
LEDG[1],PIN_AF22
LEDG[2],PIN_W19
LEDG[3],PIN_V18
LEDG[4],PIN_U18
LEDG[5],PIN_U17
LEDG[6],PIN_AA20
LEDG[7],PIN_Y18
```

Figure 10. Lab1_Part2_pin_assignments