CPEN 230L: Introduction to Digital Logic Laboratory
Lab 7: Multiplexers, Decoders, and Seven Segment Displays

Purpose

- Learn about multiplexers (MUXs), decoders and seven segment displays.
- Learn about hierarchical design in Verilog.
- Use the recommended development directory structure to perform ModelSim simulations and Quartus synthesis of the same Verilog source code.

Background, lab directory structure

- When completed your Lab 7 directory structure should be the following:

```
Lab7                        <- Top level directory for Lab 7
   mux51_3bit               <- Part 1, 3-bit wide 5 to 1 Multiplexer
      mux51_3bit.v          <- Verilog file for the multiplexer
      mux51_3bit.tcl        <- Pin assignments, I/O configuration
      mux51_3bit.sdc        <- Constraints file to suppress warnings
      <other Quartus files>
   displayDriver            <- Part 2, 7-segment Display Decoder
      displayDriver.v       <- Top level Verilog file, to drive 7-seg display
      oct7segDecoder.v      <- Decoder Verilog File (octal digit to 7-seg
                               display)
      displayDriver.tcl     <- Pin assignments, I/O configuration
      displayDriver.sdc     <- Constraints file to suppress warnings
      <other Quartus files>
   muxdisplay               <- Part 3, Combined Circuit
      sim                   <- ModelSim simulation directory
         muxdisp_top_tb.v   <- Verilog test bench code
         <other ModelSim files>
      src                   <- Source directory for both sim & synth
         muxdisp_top.v      <- Top level Verilog file, system I/O
         mux51_3bit.v       <- Multiplexer module from part 1
         oct7segDecoder.v   <- Decoder module from part 2
      synth                 <- Quartus synthesis directory
         muxdisp_top.tcl    <- Pin assignments, I/O configuration
         muxdisp_top.sdc    <- constraints file to suppress warnings
         <other Quartus files>
```

- **All 13 named files above contain only plain text.** Notice files mux51_3bit.v and oct7segDecoder.v appear twice in different directories. The copies are identical, but duplicated to keep files for the three parts of the lab completely separate.

Background, 1-bit wide 2-to-1 MUX in Verilog

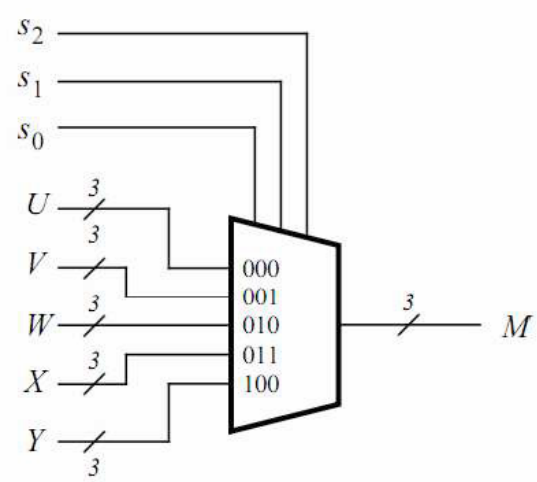
- Understand textbook section 2.8.2, pages 60 to 63, "Multiplexer Circuit". Four examples of Verilog code to implement a 1-bit wide 2-to-1 MUX follow:

```
// using assign and bitwise operators
module mux21 (input x, y, s, output m);
  assign m = (~s & x) | (s & y);
endmodule


// using assign and the conditional operator
module mux21 (input x, y, s, output m);
  assign m = s ? y : x;  // if s then y else x
endmodule


// using always and if-else
module mux21 (
  input x, y, s,
  output m);

  reg f;                  // for use in the always procedural block

  always @(x, y, s)       // could also use: always @(*)
    if (s == 1'b0)
      f = x;
    else
      f = y;
  assign m = f;           // drive output m from internal signal f

endmodule


// using always and case
module mux21 (
  input x, y, s,
  output m);

  reg f;                  // for use in the always procedural block

  always @(x, y, s)       // could also use: always @(*)
  begin
    case (s)
      0: f = x;
      1: f = y;
      default: f = x;
     endcase
  end

  assign m = f;           // drive output m from internal signal f

endmodule
```

Part 1: **3-bit wide 5 to 1 Multiplexer**

Pre-Lab

- This is a diagram of a 3-bit wide 5-to-1 multiplexer:



- The diagram shows that the value of {s2, s1, s0} selects which of U, V, W, X or Y appears at output M. For example, if {s2, s1, s0} = 011 then M = X. You will build this circuit on the DE2-115 board with SW[17:0] being used for s2, s1, s0, U, V, W, X and Y using respectively -- SW[17:15] = {s2, s1, s0}, SW[14:12] = U, SW[11:9]=V, SW[8:6]=W, SW[5:3]=X and SW[2:0] = Y. The 3-bit output M will appear on LEDG[2:0].
- Copy the Verilog code below into file \Lab7\mux51_3bit\mux51_3bit.v and **complete it**.
- Compile your code with Icarus Verilog to be sure there are no errors.

```
// CPEN 230L lab 7 part 1, 3-bit wide 5-to-1 MUX
// Firstname Lastname, mm/dd/yyyy

module mux51_3bit (
  input [17:0] SW,      // 18 switch inputs
  output [2:0] LEDG);  // 3 green LED outputs

  assign LEDG = (SW[17:15] == 3'd0) ? SW[14:12] :
                // replace this comment with useful code
                (SW[17:15] == 3'd4) ? SW[2: 0] :
                                      3'b000 // don't care:
                                             // set to any value
endmodule
```

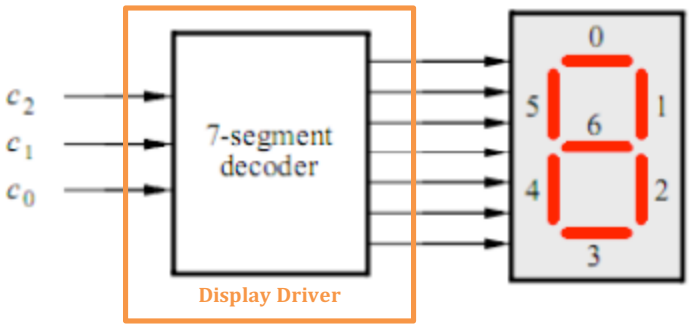
During-Lab

- Implement your 3 bit wide, 5-1 multiplexer on the DE2-115 board. In Quartus, create a Project that uses directory \Lab7\ mux51_3bit, Project name mux51_3bit, top-level design entity mux51_3bit, and containing file mux51_3bit.v.
- Put provided file mux51_3bit.sdc in directory \Lab7\mux51_3bit. This is a Synopsys Design Constraints file that will reduce warnings in Quartus.
- Put provided file mux51_3bit.tcl in directory \Lab7\mux51_3bit. This file contains pin assignments and other I/O specifications that will reduce Quartus warnings. (The pin assignments you have seen before. The I/O specifications are new. Look at the content of the file.)
- In Quartus, run mux51_3bit.tcl and compile the project.
- If you have any warnings other than "No clocks defined in design", fix them before proceeding.

- Download the circuit to the DE2-115 board.
- Verify your circuit works by setting U = 7, V = 6, W = 5, X = 4, Y = 3. Then set {s2, s1, s0} to 0, 1, 2, 3, 4 to output U, V, W, X, Y respectively to the green LEDs.
- **When you are convinced the circuit works correctly, demonstrate the test described in the previous step to your instructor.**
- Your lab report for this part should include your mux51_3bit.v Verilog code, a screen capture of the RTL Viewer image of your circuit, any other discussion that might help your target student reader understand this part of the lab.
- Save all files and close the project.

Part 2: 7-Segment Display Decoder

- Understand textbook Figure 4.21 on page 209. It will help with this part, but can't be copied directly. We will develop a 3 input decoder that will drive a display to show decimal "0" to "7". For example, an input of binary 000 displays "0" by lighting segments 0, 1, 2, 3, 4 and 5. An input of binary 111 displays "7" by lighting segments 0, 1 and 2. Segments are active-low, meaning they turn on when their control signal is a zero.



Pre-Lab

- Copy the following Verilog code into file displaydriver.v. The code provided instantiates the 7-segment decoder oct7segDecoder (octal digit to 7-segment display) and connects it to DE2-115 board switches SW[2:0] for input, and the rightmost 7-segment display HEX0 for output. The MS bit of HEX0 is HEX0[6], and it controls segment 6 in the above diagram. The LS bit of HEX0 is HEX0[0], and it controls segment 0 in the above diagram.

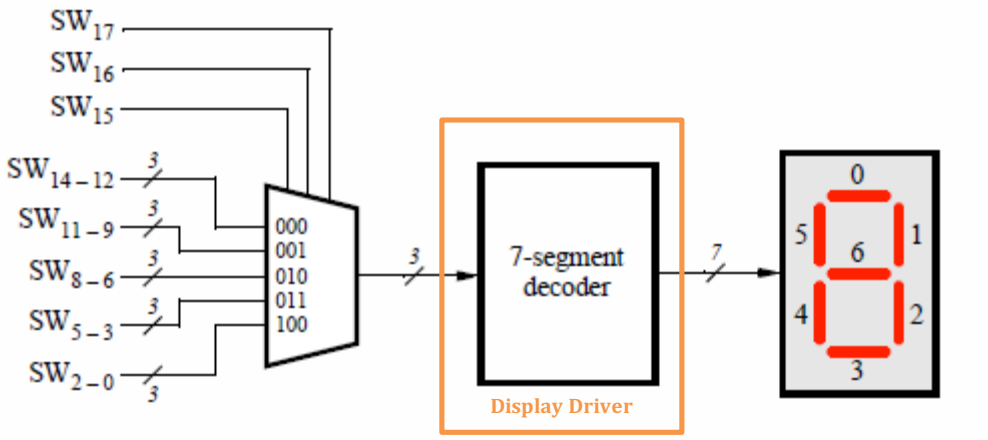
```
// CPEN230 lab 7 part 2, top level DE2-115 board connections
// Firstname Lastname, mm/dd/yy
//
// Input is an octal digit. Output is 0 to 7 on a 7-segment
// display.

module displayDriver (
  input  [2:0] SW,     // 3 input toggle switches
  output [6:0] HEX0);  // 7-seg display element

  oct7segDecoder dec_inst ( // instantiate oct7segDecoder
    .c_i    (SW),           // c_i and SW are 3 bits each
    .disp_o (HEX0));        // disp_o and HEX0 are 7 bits each

endmodule
```

- Create file oct7segDecoder.v containing module oct7segDecoder, that implements the 7-segment decoder block in the above diagram. Its input should be a 3-bit vector called c_i (c input). Its output should be a 7-bit vector called disp_o (display output). The body of the module should be an always statement with an embedded case statement similar to the fourth implementation of the mux21 example provided in page 2 of this document. (When done, ponder how much easier this is than going from a 3-input 7-output truth table to 7 expressions to 7 K-maps to an all-NAND (or whatever implementation) as you did in the past.
- Put provided files displayDriver.sdc and displayDriver.tcl in directory Lab7\displayDriver.
- Compile your code oct7segDecoder.v and displayDriver.v with Icarus Verilog to be sure there are no errors.

During-Lab

- Synthesize your 7-segment display decoder. In Quartus, create a Project using directory \Lab7\displayDriver.
- Run displayDriver.tcl and compile the project.
- If you have any warnings other than "No clocks defined in design", fix them.
- Download the configuration to the DE2-115 board.
- **When you are convinced the circuit works as it should, demonstrate it to your instructor.**
- Your lab report for this part should include your oct7segDecoder.v Verilog code, a screen capture of the RTL Viewer image of your circuit (expanded to show the details of module oct7segDecoder), and any other discussion that might help your target student reader.
- Save all files and close the project.

Part 3: Combined Circuit



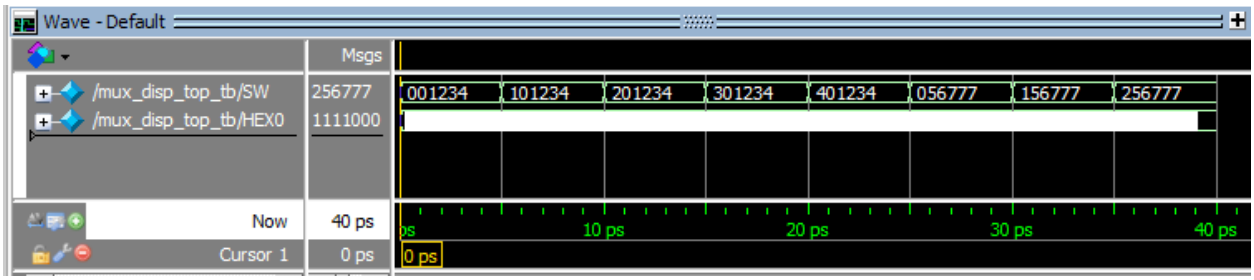
Pre-Lab

- Create all files necessary to simulate the above circuit using ModelSim, and synthesize it using Quartus and the DE2-115 board. **See "Background, lab directory structure" at the top of this file, sub-directory mux_display. We will use this same separation of source, simulation, and synthesis files in future labs.** The idea is to the keep the common Verilog files used by ModelSim and Quartus in a source (src) directory, but keep the files generated by ModelSim (sim) and Quartus (synth) separate.
- Create Verilog file muxdisp_top.v that defines module muxdisp_top, with input SW[17:0], output HEX0[6:0], and instantiations of modules mux51_3bit and displayDriver connected as indicated in the above diagram.
- Create muxdisp_top.tcl and muxdisp_top.sdc files from the content of similar files in Parts 1 and 2.
- Create Verilog file muxdisp_top_tb.v that instantiates module muxdisp_top, drives the inputs (stimulus) and monitors the outputs (response). Partially completed test bench code follows:

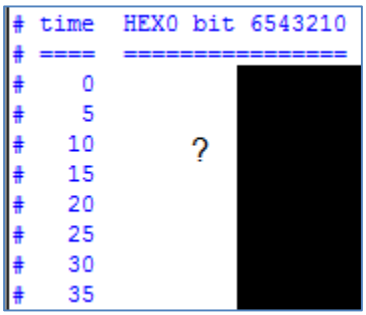
```
// CPEN230L lab 7 part 3, MUX/Decoder test bench
// Firstname, Lastname mm/dd/yy

`timescale 1ns / 100ps

module muxdisp_top_tb;  // testbench top level, no inputs/outputs
  reg [17:0] SW_sim;    // simulated input switches
  wire [6:0] HEX0_sim;  // simulated output 7-segment display segments

  muxdisp_top DUT(      // instantiate the DUT
    .SW   (SW_sim),
    .HEX0 (HEX0_sim));

    initial $timeformat(-9, 1, " ns", 10); // specify time format to be ns

    initial begin                   // test bench header
      $display("time  HEX0 bit 6543210");
      $display("====  ================");
    end

    // Trick to trigger $monitor on SW_sim changes without putting SW_sim in
    // its parameter list
    always @ (SW_sim) begin
      $monitor("%4d           %7b", $time, HEX0_sim);
    end

    initial begin
         SW_sim[14: 0] = 15'o01234; // use octal to deal with 3-bit groups
         SW_sim[17:15] =  3'o0;     // @t=0  output = SW[14:12] = 0 = 1000000
      #5 SW_sim[17:15] =  3'o1;     // @t=5  output = SW[11: 9] = 1 = 1111001
         // Replace this comment with t=10,15,20 to output 2, 3, 4
      #5 SW_sim[14: 0] = 15'o56777;
         // Replace this comment with remaining lines to display 5,6,7, 7, 7
      $finish; // end of simulation
    end

endmodule
```

During-Lab

- **Simulate** your circuit using ModelSim. Project name is muxdisp_top_tb. Use directory \Lab7\muxdisp_top\sim. Add muxdisp_top_tb.v and all \Lab7\muxdisp_top\src\ Verilog files to the ModelSim project.
- Verify your circuit works by getting pictures similar to those following, with content that shows your oct7segDecoder module produces the expected output. If needed, change the time scale to ps (pico-seconds) by right-clicking on the values → Grid, Timeline, and Cursor Control → Time units → ps. Change the SW waveform radix to Octal by right-clicking on its name and choosing Radix → Octal.
- ModelSim Wave pane for muxdisp_top_tb:



- ModelSim Transcript pane for muxdisp_top_tb:



- When you are satisfied that your simulation works correctly, save a ModelSim screen capture for use in your lab report. Realize that if you proceed and your circuit doesn't work correctly you will have to re-compile and re-do this simulation process.

- **Synthesize** your circuit using Quartus in directory \Lab7\muxdisplay\synth with Project name and top-level design entity muxdisp_top.
- In the \Lab7\muxdisp_top\synth directory, create files muxdisp_top.tcl and muxdisp_top.sdc using content from the TCL and SDC files in parts 1 and 2.
- Compile the design.
- If you have any warnings other than "No clocks defined in design", fix them.
- Download the configuration to the DE2-115 board FPGA.
- Test the circuit by manually doing the same procedure done by the test bench code, then **demonstrate that test procedure to your instructor.**
- Your lab report for this part should include
    - muxdisp_top_tb.v and muxdisp_top.v Verilog code
    - images of the ModelSim Wave and Transcript pane output, with a description of how this information proves your circuit is working correctly
    - an RTL Viewer image of your circuit (non-expanded to show how simple it is)
    - any other discussion that might help your target student reader
- Save all files, close ModelSim, close Quartus, and take a copy of all your files with you.

Pre-Lab Deliverables:

The penalty for not handing in the prelab deliverables is 50% of the lab grade

Part 1:

- mux51_3bit.v

Part 2:

- oct7segDecoder.v, displayDriver.v

Part 3:

- muxdisp_top.v, muxdisp_top_tb.v
- Test the correct operation of your design with Icarus Verilog and GTKwave and provide a snapshot of the table generated by muxdisp_top_tb