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.
- Learn about a recommended development directory structure for doing ModelSim simulation 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 for Lab 7
   mux3w_5to1               <- Part 1, 3 bit Wide 5 to 1 Multiplexer
      mux3w_5to1.v          <- Verilog file for the multiplexer
      mux3w_5to1.tcl        <- Pin assignments, I/O configuration
      mux3w_5to1.sdc        <- Constraints file to suppress warnings
      <other Quartus files>
   display_decoder          <- Part 2, 7-segment Display Decoder
      display_decoder.v     <- Top level Verilog file, decoder I/O
      oct7seg.v             <- Verilog file for the decoder
      display_decoder.tcl   <- Pin assignments, I/O configuration
      display_decoder.sdc   <- Constraints
      <other Quartus files>
   mux_display              <- Part 3, Combined Circuit
      sim                   <- ModelSim simulation directory
         mux_disp_top_tb.v  <- Verilog test bench code
         <other ModelSim files>
      src                   <- Source directory for both sim & synth
         mux_disp_top.v     <- Top level Verilog file, system I/O
         mux3w_5to1.v       <- Multiplexer module from part 1
         oct7seg.v          <- Decoder module from part 2
      synth                 <- Quartus synthesis directory
         mux_disp_top.tcl   <- Pin assignments, I/O configuration
         mux_disp_top.sdc   <- constraints
         <other Quartus files>
```

- **All 13 named files above contain only plain text.** Notice files mux3w_5to1.v and oct7seg.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

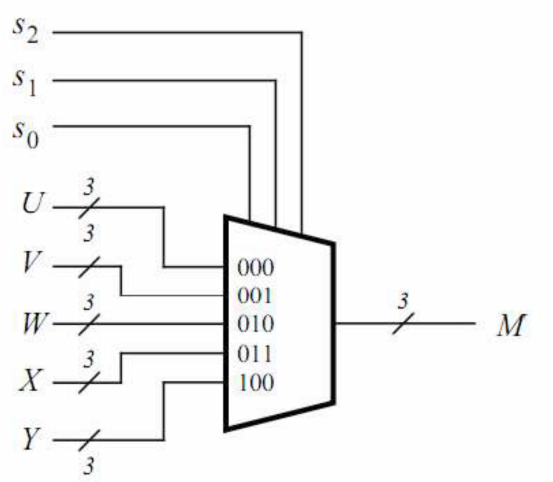
- Understand textbook section 2.8.2, pages 60 to 63, “Multiplexer Circuit”. Cedar simulation file MUXs.cdl is also available to help with understanding MUXs. Three examples of Verilog code to implement a 1-bit wide 2-to-1 MUX follow:

```
// using assign and bitwise operators
module mux_2_1 (input x, y, s, output m);
  assign m = (~s & x) | (s & y);
endmodule


// using assign and the conditional operator
module mux_2_1 (input x, y, s, output m);
  assign m = s ? y : x;  // if s then y else x
endmodule


// using always and if-else
module mux_2_1 (
  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
```

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 respectively -- SW[17:15] = {s2, s1, s0}, SW[14:12] = U, … SW[2:0] = Y. The 3-bit output M will appear on LEDG[2:0].

- Copy the Verilog code below into file \Lab7\mux3w_5to1\mux3w_5to1.v and **complete it**.
- Optional: Compile your code with Icarus Verilog to be sure there are no errors.

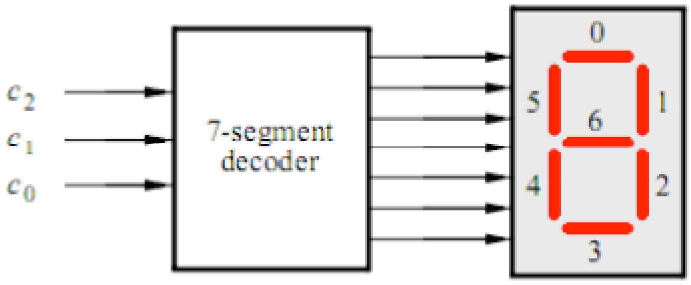
```
// CPEN 230L lab 7 part 1, 3-bit wide 5-to-1 MUX
// Firstname Lastname, mm/dd/yyyy

module mux3w_5to1 (
  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'bxxx; // don't care
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\ mux3w_5to1, Project name mux3w_5to1, top-level design entity mux3w_5to1, containing file mux3w_5to1.v.
- Put provided file mux3w_5to1.sdc in directory \Lab7\ mux3w_5to1. This is a Synopsys Design Constraints file that will reduce warnings in Quartus.
- Put provided file mux3w_5to1.tcl in directory \Lab7\ mux3w_5to1. 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 mux3w_5to1.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 mux3w_5to1.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

- Put the following Verilog code into file display_decoder.v. It instantiates 7-segment decoder oct7seg (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], controlling segment 6 in the above diagram. The LS bit of HEX0 is HEX0[0], controlling 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 display_decoder (
  input  [2:0] SW,      // 3 input toggle switches
  output [6:0] HEX0);   // 7-seg display element

  oct7seg char0 (       // instantiate oct7seg
    .c_i    (SW),       // c_i and SW are 3 bits each
    .disp_o (HEX0));    // disp_o and HEX0 are 7 bits each

endmodule
```

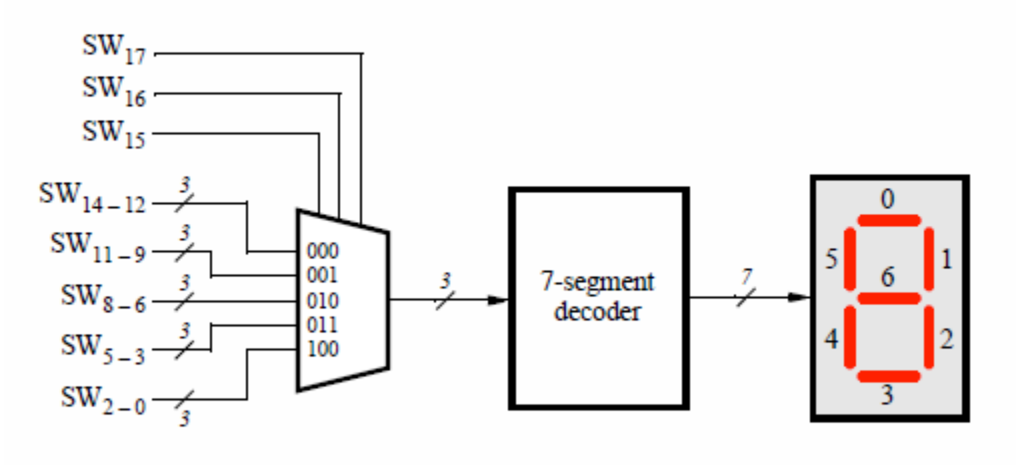
- Create file oct7seg.v containing module oct7seg, that implements the 7-segment decoder block in the above diagram. Its input should be a 3-bit vector called c_i (c input). Its output should be a 7-bit vector called disp_o (display output). The body of the module should be a single assign statement with nested conditional operators similar to that in file mux3w_5to1.v. (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 display_decoder.sdc and display_decoder.tcl in directory Lab7\display_decoder.
- Optional: Compile your code 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\display_decoder.
- Run display_decoder.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 oct7seg.v Verilog code, a screen capture of the RTL Viewer image of your circuit (expanded to show the details of module oct7seg), and any other discussion that might help your target student reader.
- Save all files and close the project.

Part 3: Combined Circuit



Pre-Lab

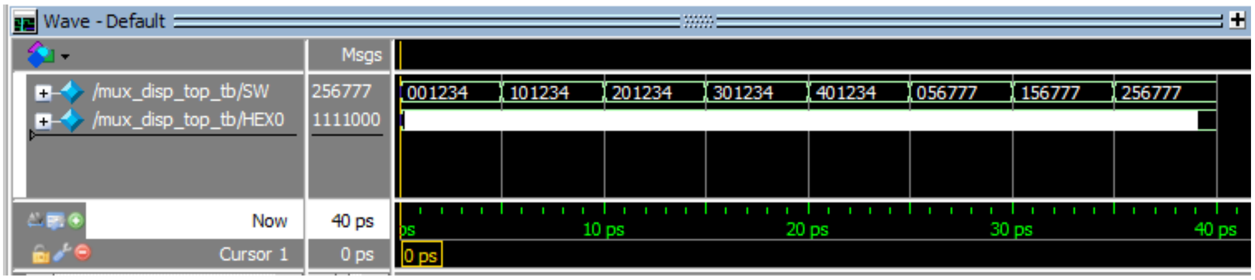
- Create all files necessary to simulate the above circuit using ModelSim, and synthesize it using Quartus and the DE2-115 board. **See "Background, lab directory structure" at the top of this file, sub-directory mux_display. We will use this same separation of source, simulation, and synthesis files 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 mux_disp_top.v that defines module mux_disp_top, with input SW[17:0], output HEX0[6:0], and instantiations of modules mux3w_5to1 and oct7seg connected as indicated in the above diagram.
- Create mux_disp_top.tcl and mux_disp_top.sdc files from the content of similar files in Parts 1 and 2.
- Create Verilog file mux_disp_top_tb.v that instantiates module mux_disp_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

module mux_disp_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

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

  initial begin                     // test bench stimulus
      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 t=25,30,35 to output 5,6,7
    #5 $finish;                    // @t=40, end simulation
  end

  initial begin                 // test bench response
    $display("time  HEX0 bit 6543210");
    $display("====  ================");
    $monitor("%4d           %7b", $time, HEX0_sim);
  end

endmodule
```

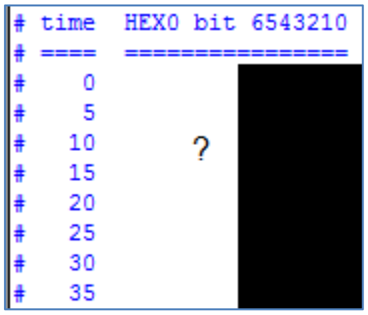
During-Lab

- **Simulate** your circuit using ModelSim. Project name is mux_disp_top_tb. Use directory \Lab7\mux_disp_top\sim. Add mux_disp_top_tb.v and all \Lab7\mux_disp_top\src\ Verilog files to the ModelSim project.
- Verify your circuit works by getting pictures similar to those following, with content that shows your oct7seg 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 mux_disp_top_tb:



- ModelSim Transcript pane for mux_disp_top_tb:



```
# time   HEX0 bit 6543210
# ====   ================
#    0
#    5
#   10        ?
#   15
#   20
#   25
#   30
#   35
```

[redacted]

- 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\mux_display\synth with Project name and top-level design entity mux_disp_top.
- In the \Lab7\mux_disp_top\synth directory, create files mux_disp_top.tcl and mux_disp_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
  - mux_disp_top_tb.v and mux_disp_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)
  - a photograph of the DE2-115 board with an explanation of what {s2, s1, s0} are set to, what switches are thus routed to the output, what those switches are set to, and why the display is thus correct.
  - 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.