CSC258-Lab #2 Pre-Lab report

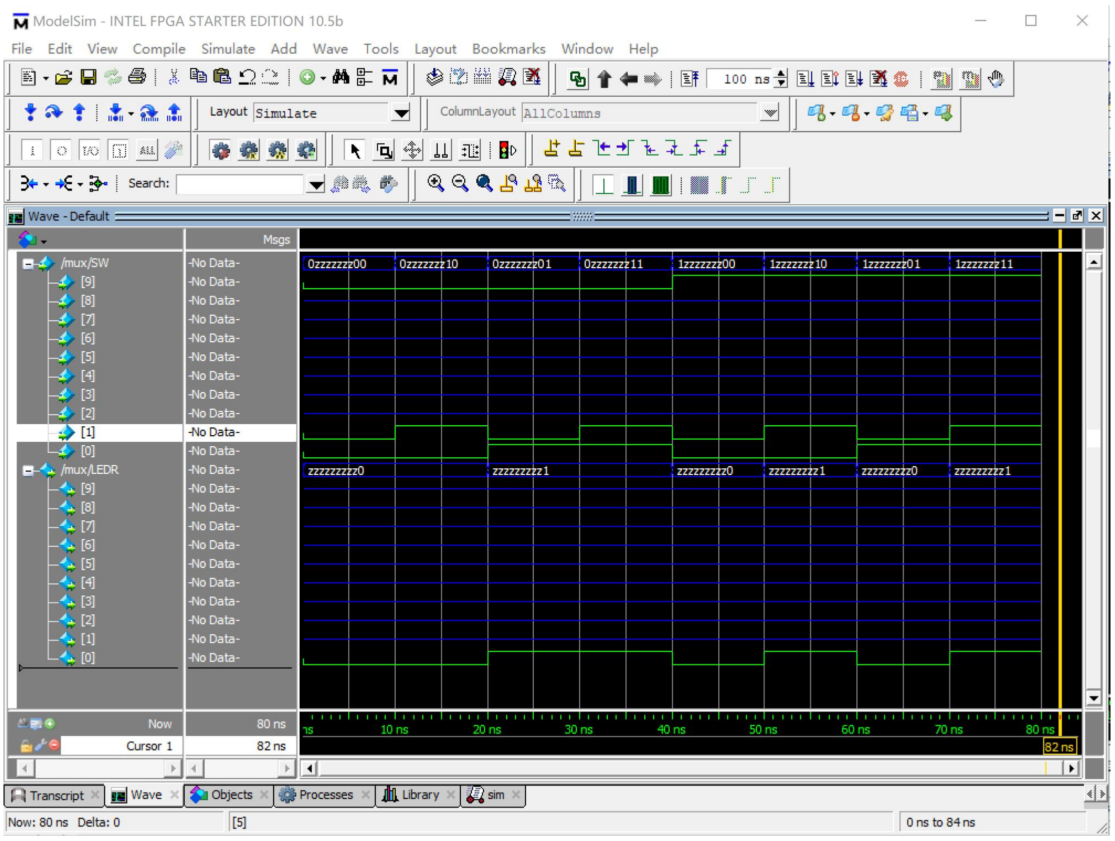
Multiplexers, Design Hierarchy, and HEX Displays

Last Name: Shi
First Name: JIakai
Student Number: 1003986760

Pre-Lab Part I

1. Look at the generated waveform , which is the simulation output, and verify that the provided test-cases work as expected.

The waveform is shown below:



So, the output is represented by LEDR[0], which meets all the result of the test-cases. It can be verified by creating a truth table for this Boolean expression.

Pre-Lab Part II

1) **How many rows that the truth table in Table2 would have if that truth table is full?**

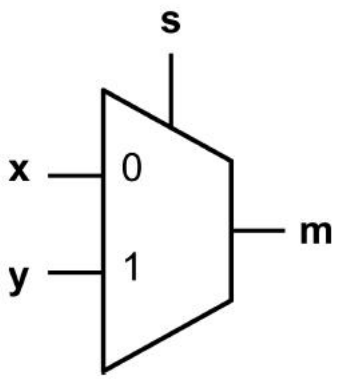
There are 6 inputs (s0, s1, v, x, u, w) in a 4-to-1 mux, so there will be $2^6 = 64$ rows.

2) **Draw a schematic (not in Quartus) showing how to connect the mux2to1 modules to build the 4-to-1 multiplexer.**

The Boolean expression for 2-to-1 multiplexer is

$$m = xs' + ys$$

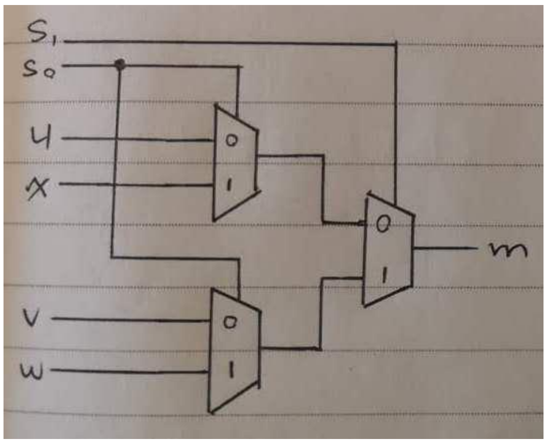
and symbol for a 2-to-1 multiplexer is



The Boolean expression for 4-to-1 multiplexer is

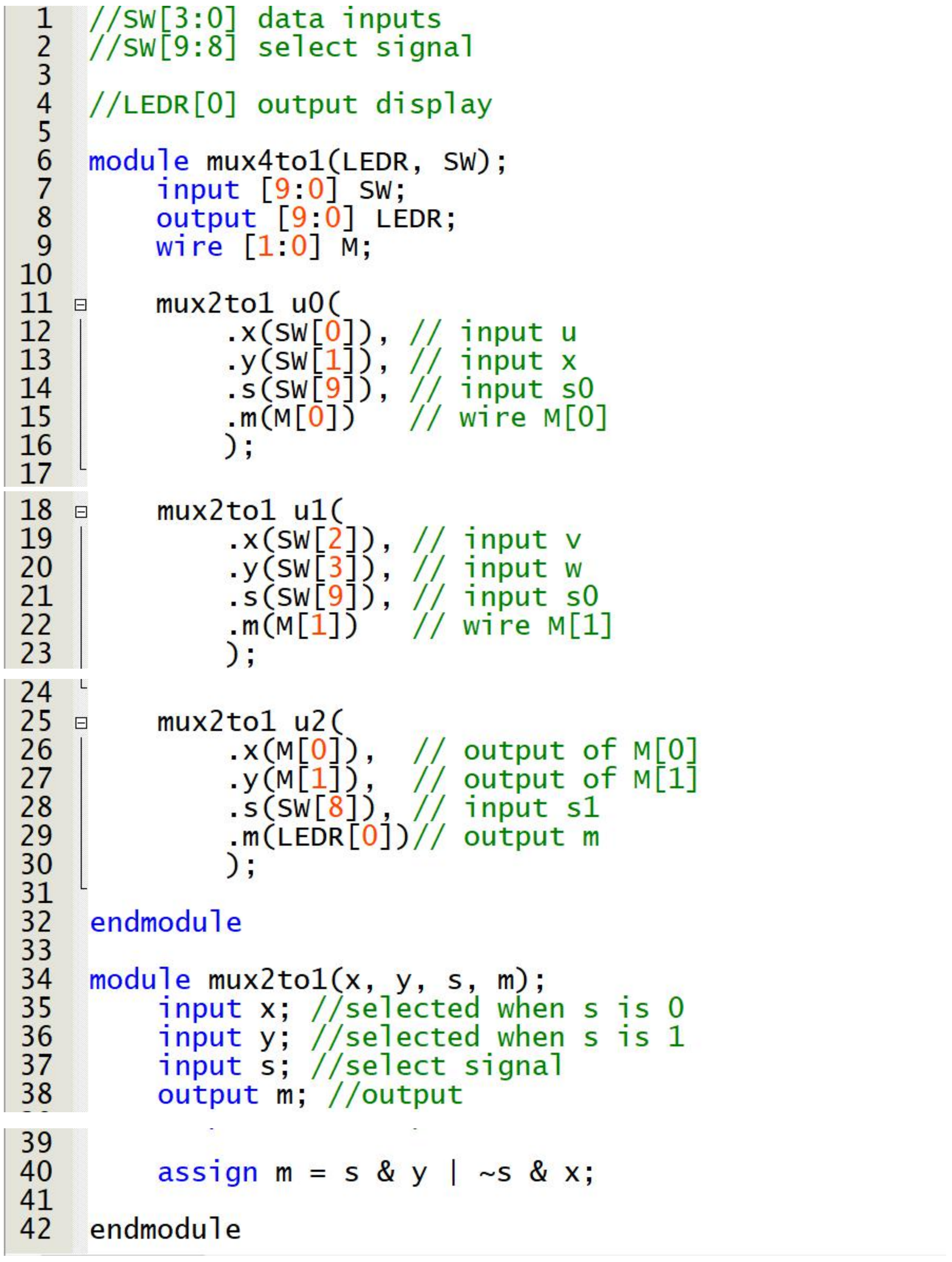
$$\begin{aligned} m &= us_0's_1' + vs_0's_1' + ws_0's_1' + xs_0s_1 \\ &= (us_0' + xs_0)s_1' + (vs_0' + ws_0)s_1 \end{aligned}$$

Hence, the schematic can be drawn like this:



3) **Write a Verilog code for your circuit.**

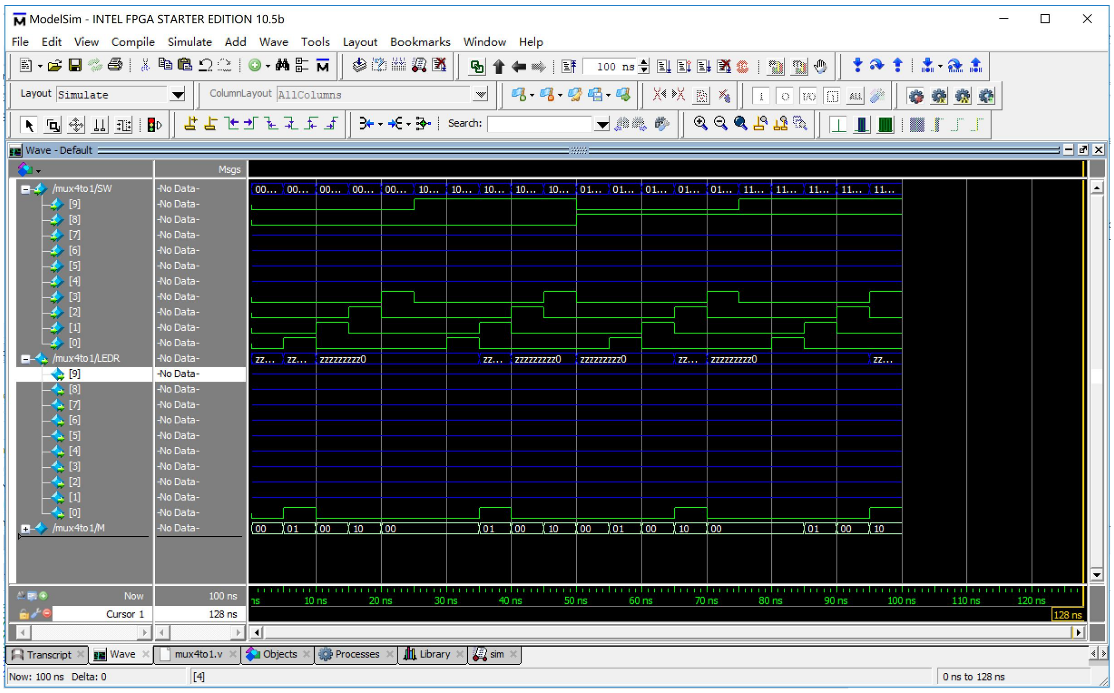
The Verilog code is represented below:



```
//SW[3:0] data inputs
//SW[9:8] select signal

//LEDR[0] output display

module mux4to1(LEDR, SW);
    input [9:0] SW;
    output [9:0] LEDR;
    wire [1:0] M;

    mux2to1 u0(
        .x(SW[0]), // input u
        .y(SW[1]), // input x
        .s(SW[9]), // input s0
        .m(M[0])   // wire M[0]
        );

    mux2to1 u1(
        .x(SW[2]), // input v
        .y(SW[3]), // input w
        .s(SW[9]), // input s0
        .m(M[1])   // wire M[1]
        );


    mux2to1 u2(
        .x(M[0]),  // output of M[0]
        .y(M[1]),  // output of M[1]
        .s(SW[8]), // input s1
        .m(LEDR[0])// output m
        );

endmodule

module mux2to1(x, y, s, m);
    input x; //selected when s is 0
    input y; //selected when s is 1
    input s; //select signal
    output m; //output


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

endmodule
```

4) **Simulate your circuit with ModelSim for different values of s , u , v , w and x .**

The waveform is shown below:



So my do file is trying to show that for each control signals (set s_1s_0 to 00, 01, 10, 11), which SW control LEDR[0]. The result of the waveform is that, if s_1s_0 is 00, SW[0] (input u) control LEDR[0]; if s_1s_0 is 01, SW[2] (input v) control LEDR[0]; if s_1s_0 is 10, SW[3] (input w) control LEDR[0]; and if s_1s_0 is 11, SW[1] (input x) control LEDR[0]. Therefore, the result is exactly the same as the information showing on the truth table, which means the circuit is working.

Pre-Lab Part III

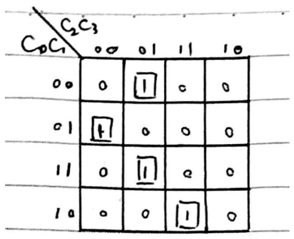
1. Write an expression and draw a Karnaugh map for each segment of the 7-segment decoder.

Since the 7-segment display uses common anode, the logic "0" on the K-map is where the light segment is lighting up.

HEX0:

$$F' = C_0'C_1'C_2'C_3 + C_0'C_1C_2'C_3' + C_0C_1C_2'C_3 + C_0C_1'C_2C_3$$

The Karnaugh map for HEX0:

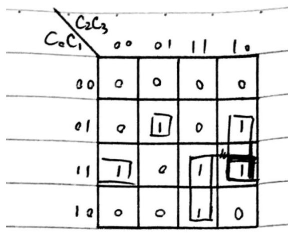


C_0C_1 \ C_2C_3	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	0	1	0	0
10	0	0	1	0

HEX1:

$$F' = C_0'C_1C_2'C_3 + C_1C_2C_3' + C_0C_1C_3' + C_0C_2C_3$$

The Karnaugh map for HEX1:

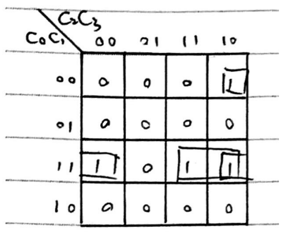


C_0C_1 \ C_2C_3	00	01	11	10
00	0	0	0	0
01	0	1	0	1
11	1	0	1	1
10	0	0	1	0

HEX2:

$$F' = C_0'C_1'C_2C_3' + C_0C_1C_3' + C_0C_1C_2$$

The Karnaugh map for HEX2:

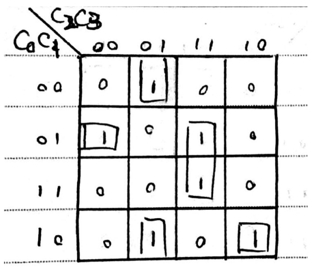


C_0C_1 \ C_2C_3	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	1	0	1	1
10	0	0	0	0

HEX3:

$$F' = C_1'C_2'C_3 + C_0'C_1C_2'C_3' + C_1C_2C_3 + C_0C_1'C_2C_3'$$

The Karnaugh map for HEX3:

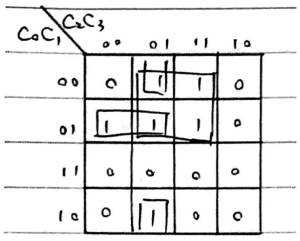


$C_0C_1 \backslash C_2C_3$	00	01	11	10
00	0	1	0	0
01	1	0	1	0
11	0	0	1	0
10	0	1	0	1

HEX4:

$$F' = C_0'C_3 + C_0'C_1C_2' + C_1'C_2'C_3$$

The Karnaugh map for HEX4:

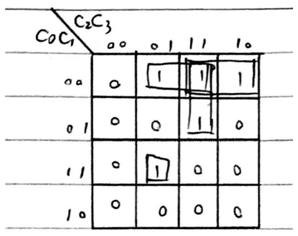


$C_0C_1 \backslash C_2C_3$	00	01	11	10
00	0	1	1	0
01	1	1	1	0
11	0	0	0	0
10	0	1	0	0

HEX5:

$$F' = C_0'C_1'C_3 + C_0'C_1'C_2 + C_0'C_2C_3 + C_0C_1C_2'C_3$$

The Karnaugh map for HEX5:

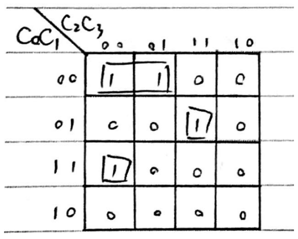


$C_0C_1 \backslash C_2C_3$	00	01	11	10
00	0	1	1	1
01	0	0	1	0
11	0	1	0	0
10	0	0	0	0

HEX6:

$$F' = C_0'C_1'C_2' + C_0'C_1C_2C_3 + C_0C_1C_2'C_3'$$

The Karnaugh map for HEX6:



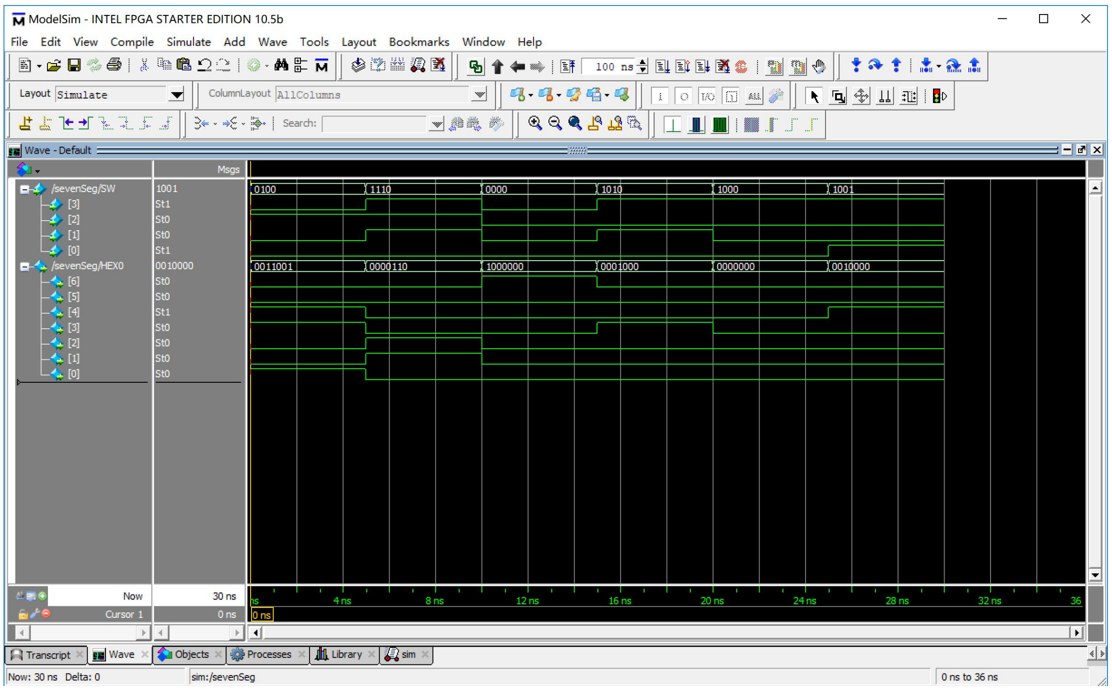
$C_0C_1 \backslash C_2C_3$	00	01	11	10
00	1	1	0	0
01	0	0	1	0
11	1	0	0	0
10	0	0	0	0

2. Write a Verilog module for the 7-segment decoder.

Here is the Verilog code have been provided in the file sevenSeg.v.

3. You will provide input signals for 2, then 7, then 0, then 5, then 1, and finally 9. You must include your simulation waveforms as part of your prelab.

The waveform is shown below:



ModelSim - INTEL FPGA STARTER EDITION 10.5b

```
/sevenSeg/SW     1001      0100     1110     0000     1010     1000     1001
  [3]            St1
  [2]            St0
  [1]            St0
  [0]            St1
/sevenSeg/HEX0   0010000   0011001  0000110  1000000  0001000  0000000  0010000
  [6]            St0
  [5]            St0
  [4]            St1
  [3]            St0
  [2]            St0
  [1]            St0
  [0]            St0

Now       30 ns
Cursor 1  0 ns

Now: 30 ns  Delta: 0     sim:/sevenSeg     0 ns to 36 ns
```

The test cases deriving the waveform have been provided in the file display.do.