Experiment 4

Multiplexers in Combinational Logic

Digital Systems

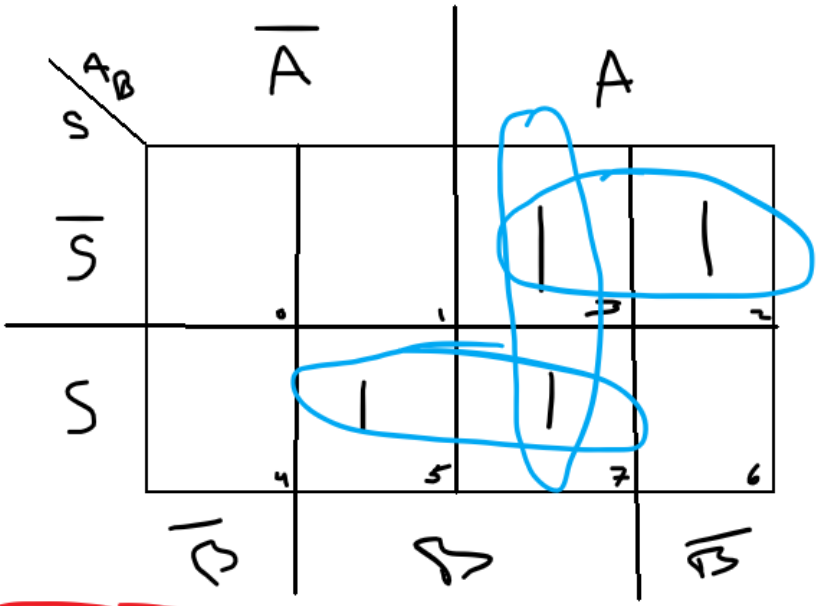
EEE3342C-20FALL 0011

Spenser Tacinelli
PID: sp498810

Pre-Laboratory Assignment:

1. Generate truth table of a 2:1 multiplexer. Determine the min-terms and write the Boolean expression for the output.

s	a	b	out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



$A\overline{S} + AB + BS$

2. Write the Verilog code in both Logic form (assign statements) and Behavioral form for the Multiplexer depending on the inputs and outputs.

```
module mux(s,a,b);

output b;
input[1:0] s;
input[3:0] a;

wire[1:0] s;
wire[3:0] a;
wire b;

assign b = a[s];

endmodule
```

```
mux4 (s, A, B, out);
(input [1:0] s, // 2-bit control signal
input A,
input B,
output reg out
);
 always @(*) begin
if (s == 2'b00) out = A;
else if (s == 2'b01) out = B;
end
endmodule
```

Objective:

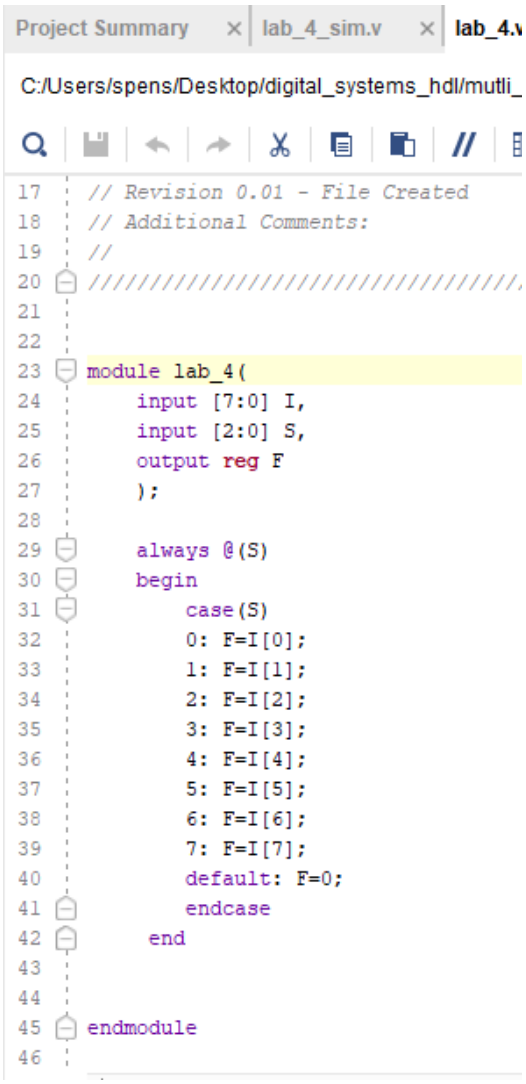
To learn about multiplexers and the implementation of combinational logic design.

Equipment:

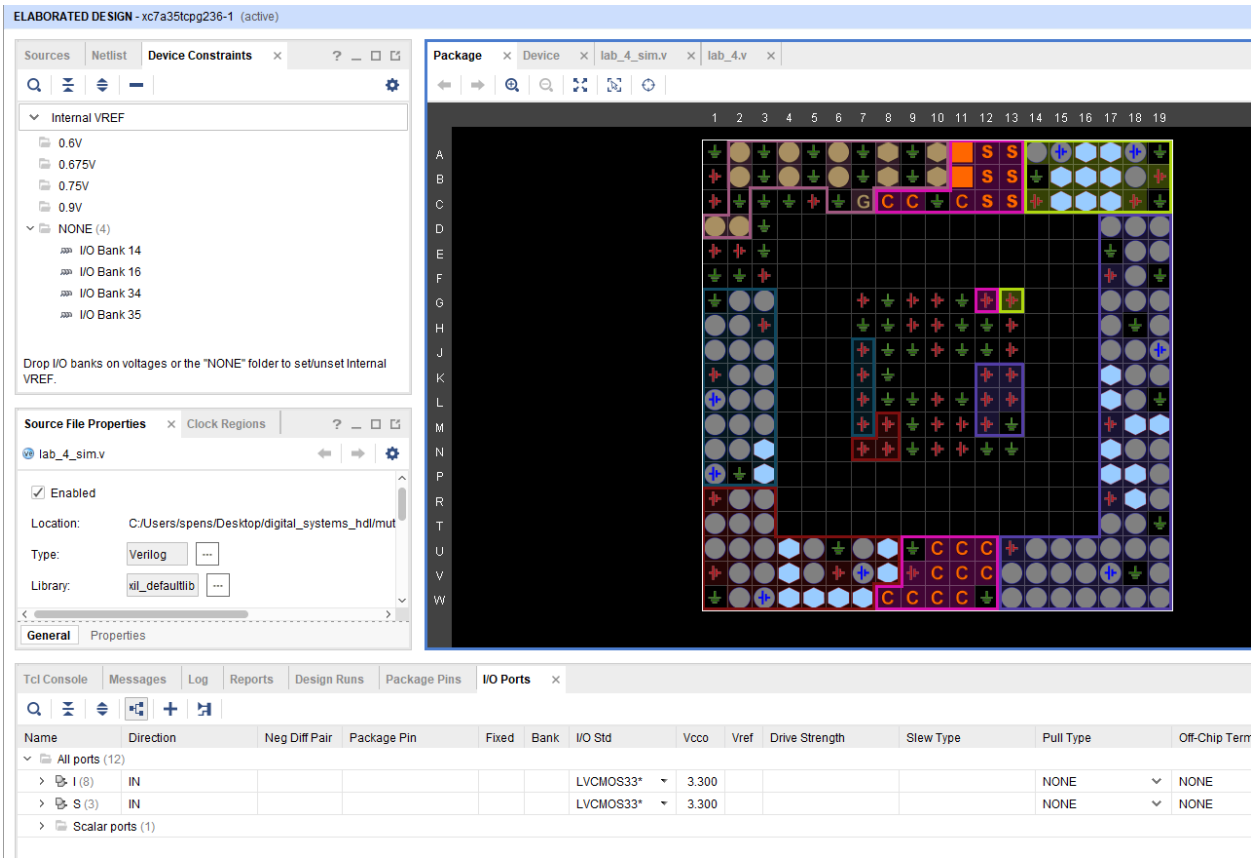
The Xilinx's FPGA VIVADO HLx Editions design tools are available in the laboratory. These tools can also be downloaded from Xilinx's web site at www.xilinx.com. The WebPack version of this tool that we use for the laboratory experiments are located under the support download section at the related website (https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloa dNav/vivado-design-tools/2017-4.html). Please take note that the file download size exceeds 1 GB and also during the installation process updates may have to be installed. It can take you up to a few hours to download and install the software on your computer. The user does not need to have the BASYS development board interface to the computer to design and simulate an FPGA.

Design Steps:

I created the project (multi_8_proj) and added the two inputs and the output register. I then programmed the logic of the 3-bit binary adder.

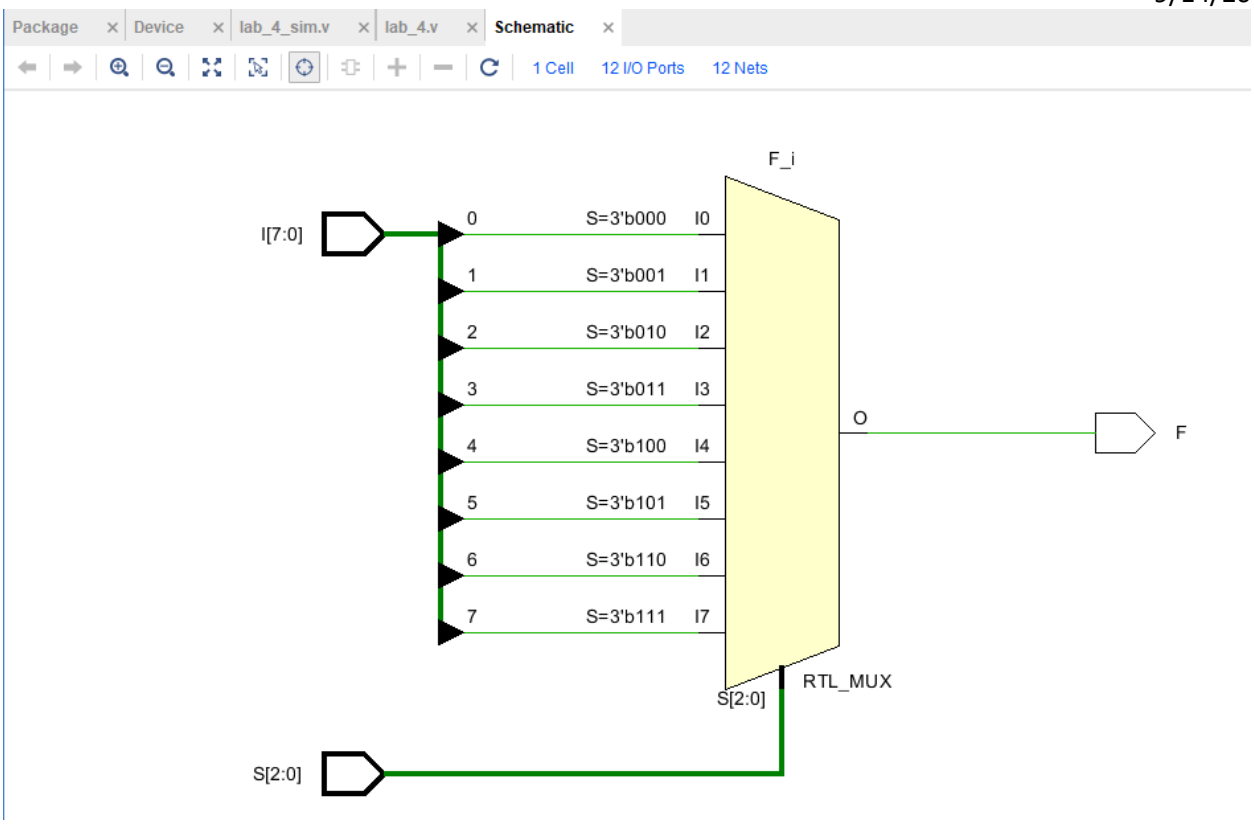


I then ran synthesis and implementation, to implement I/O Planning and Schematics. I then changed the I/O std to LVCMOS33 in the I/O Planning.



Then I generated the schematic of the multiplexer circuit.

I then programmed the logic for the simulation.

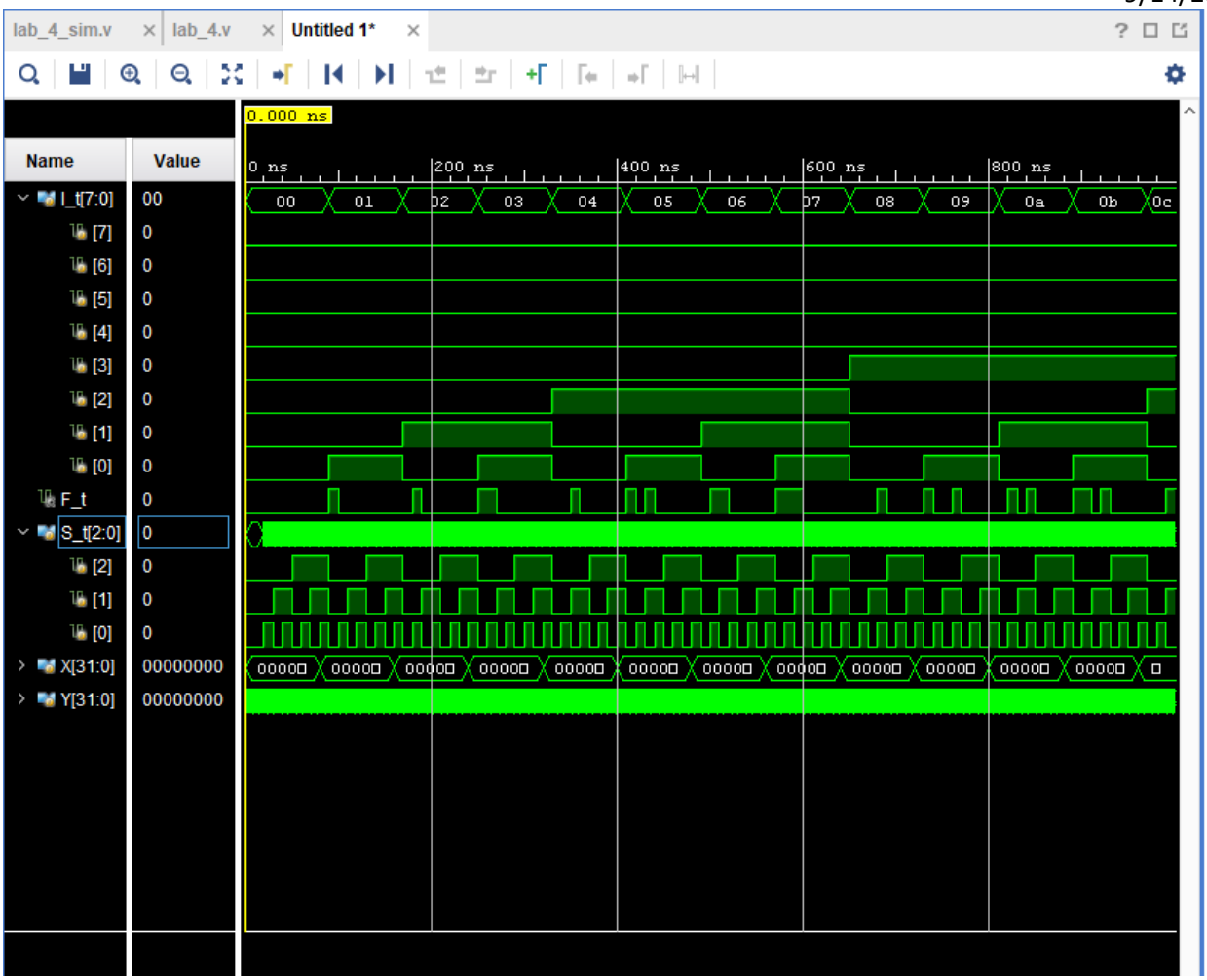

Package
Device
lab_4_sim.v
lab_4.v
Schematic
1 Cell
12 I/O Ports
12 Nets
F_i
I[7:0]
0
S=3'b000
I0
1
S=3'b001
I1
2
S=3'b010
I2
3
S=3'b011
I3
4
S=3'b100
I4
5
S=3'b101
I5
6
S=3'b110
I6
7
S=3'b111
I7
O
F
RTL_MUX
S[2:0]
S[2:0]

Package | Device | lab_4_sim.v | lab_4.v | Schematic

C:/Users/spens/Desktop/digital_systems_hdl/mutli_8_proj/mutli_8_proj.srcs/sim_1/new/lab_4_sim.v

```
22
23  module lab_4_sim(
24
25      );
26      reg [7:0] I_t;
27      reg [2:0] S_t;
28      wire F_t;
29      integer X;
30      integer Y;
31
32      lab_4 UUT(
33      .I(I_t),
34      .S(S_t),
35      .F(F_t)
36      );
37
38      initial begin
39      I_t = 3'd0;
40      S_t = 3'd0;
41      for (X = 0; X <= 255; X = X+1) begin
42      for (Y = 0; Y <= 7; Y = Y+1) begin
43      #10 I_t = X;
44      S_t = Y;
45      end
46     end
47    end
48
49  endmodule
50
```

I then ran a behavioral simulation.

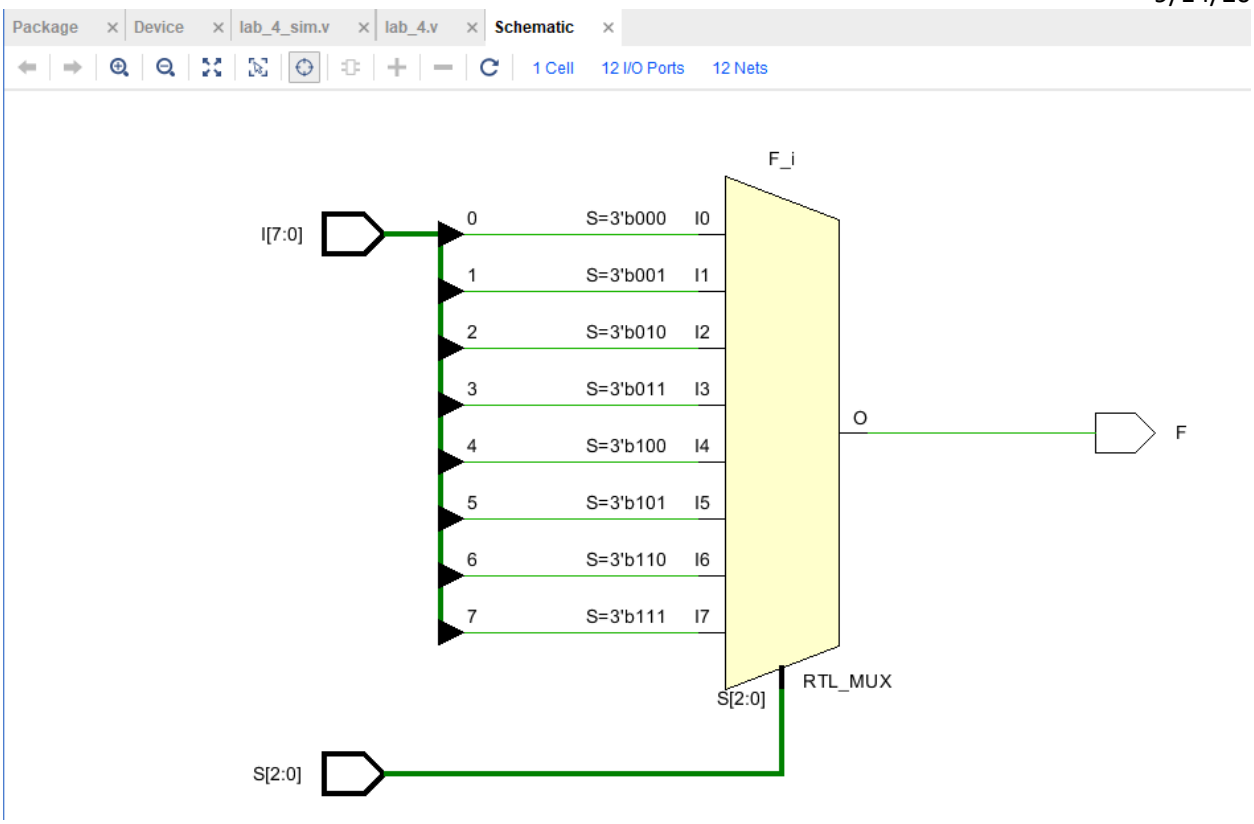


(behavioral)

Multiplexer circuit Schematic

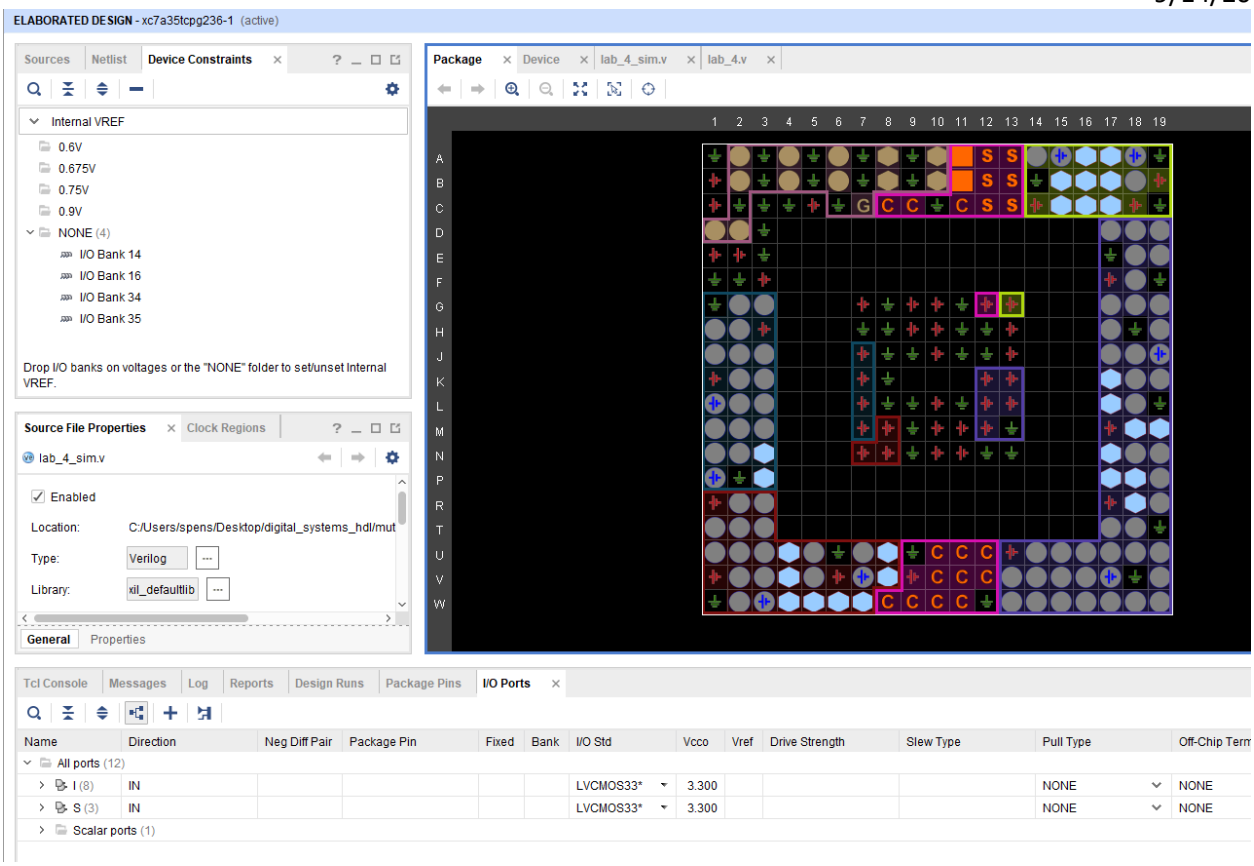
Logic Diagram:

The resulting schematic diagram for the multiplexer.



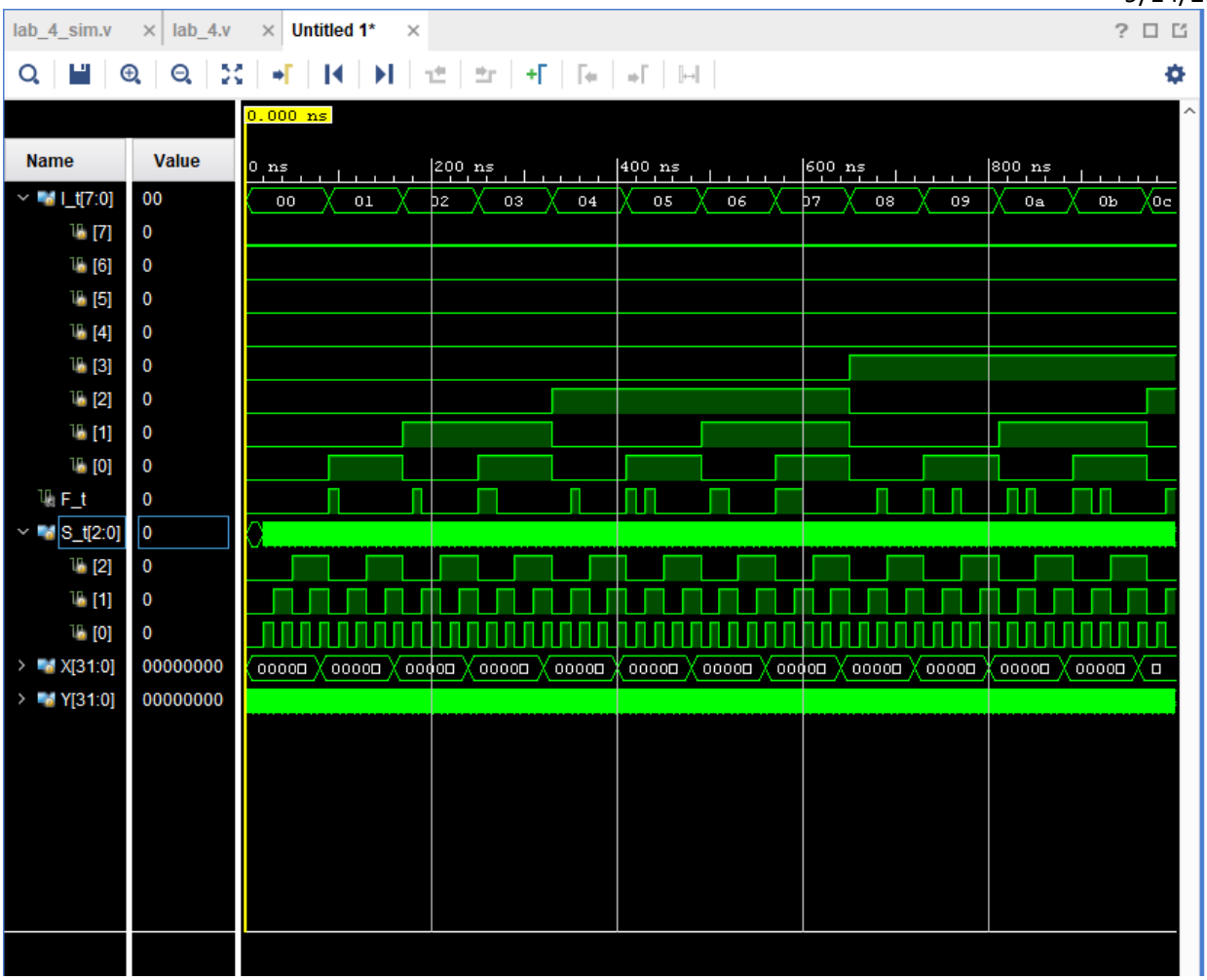
Design Specification Plan:

This is the design specification for a multiplexer. It has two inputs (I[7:0] and S[2:0]) and one output (F).



Results Statement:

Here is the behavioral simulation results for the multiplexer.



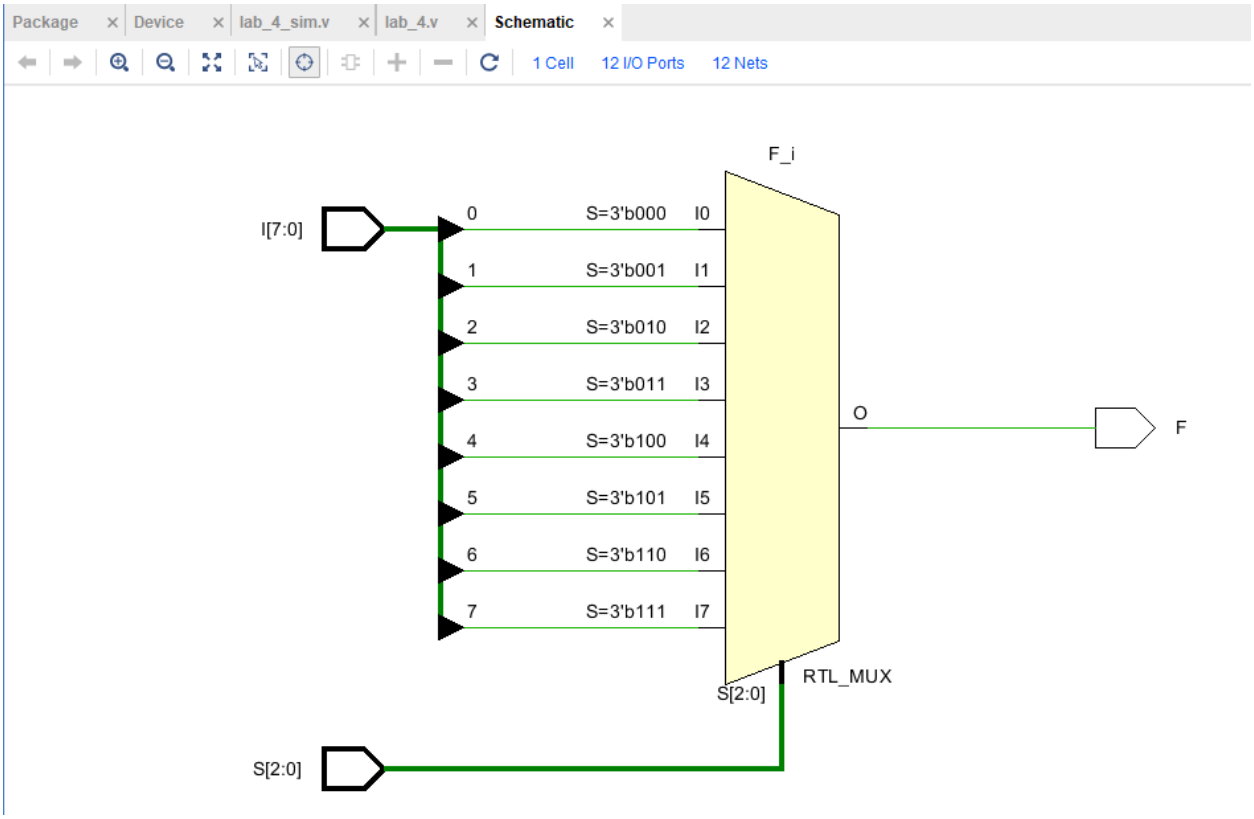
s	a	b	out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1

1	1	0	0
1	1	1	1

Multiplexer circuit Verilog

Logic Diagram

The results from the simulation testing for the simulation for multiplexor circuit.



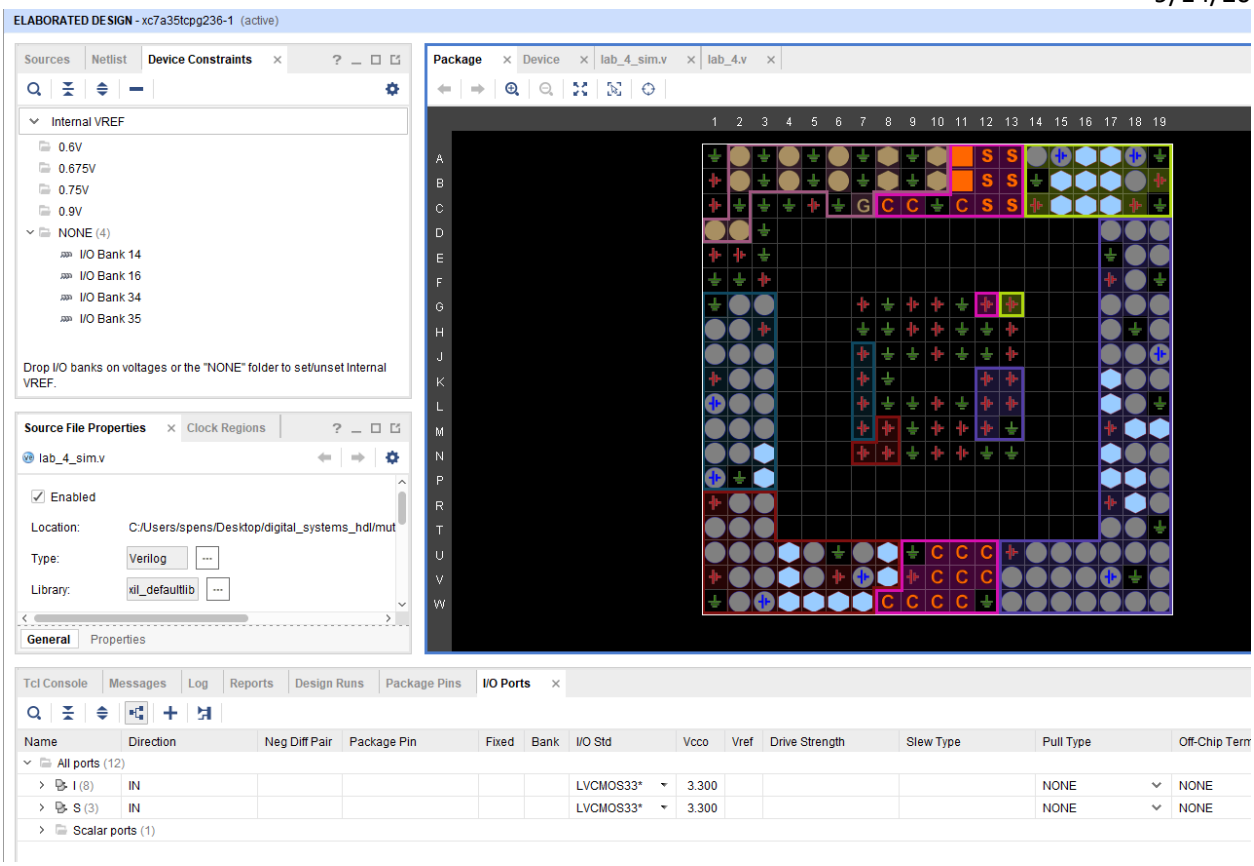
Design Specification Plan

This is the design specification for a multiplexer. It has two inputs (I[7:0] and S[2:0]) and one output (F).

Package | Device | lab_4_sim.v | lab_4.v | Schematic

C:/Users/spens/Desktop/digital_systems_hdl/mutli_8_proj/mutli_8_proj.srcs/sim_1/new/lab_4_sim.v

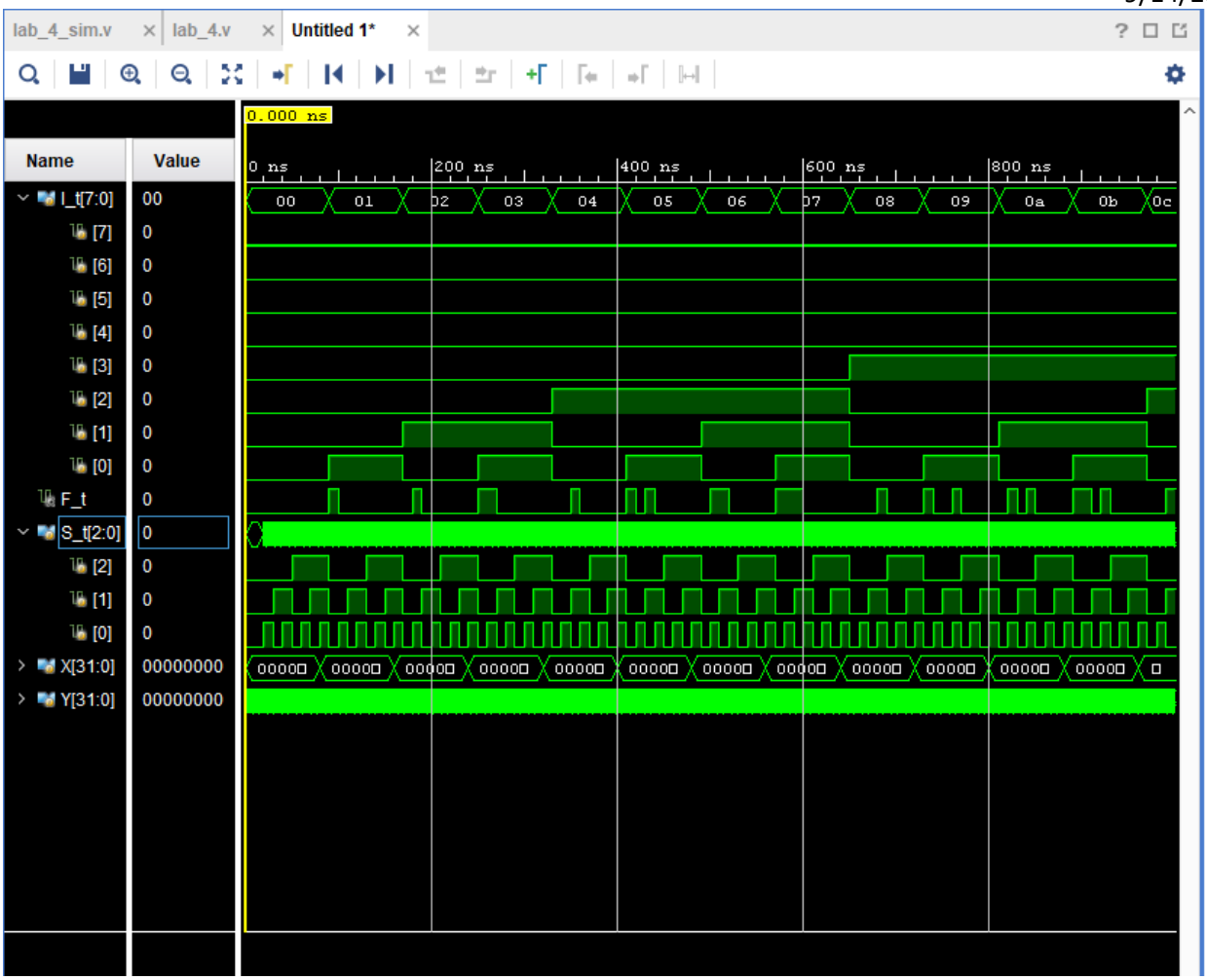
```
22
23  module lab_4_sim(
24
25      );
26      reg [7:0] I_t;
27      reg [2:0] S_t;
28      wire F_t;
29      integer X;
30      integer Y;
31
32      lab_4 UUT(
33      .I(I_t),
34      .S(S_t),
35      .F(F_t)
36      );
37
38      initial begin
39      I_t = 3'd0;
40      S_t = 3'd0;
41      for (X = 0; X <= 255; X = X+1) begin
42      for (Y = 0; Y <= 7; Y = Y+1) begin
43      #10 I_t = X;
44      S_t = Y;
45      end
46     end
47    end
48
49  endmodule
50
```



Results Statement

Explanation of pictures and tables.

Here is the behavioral simulation results.



s	a	b	out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1

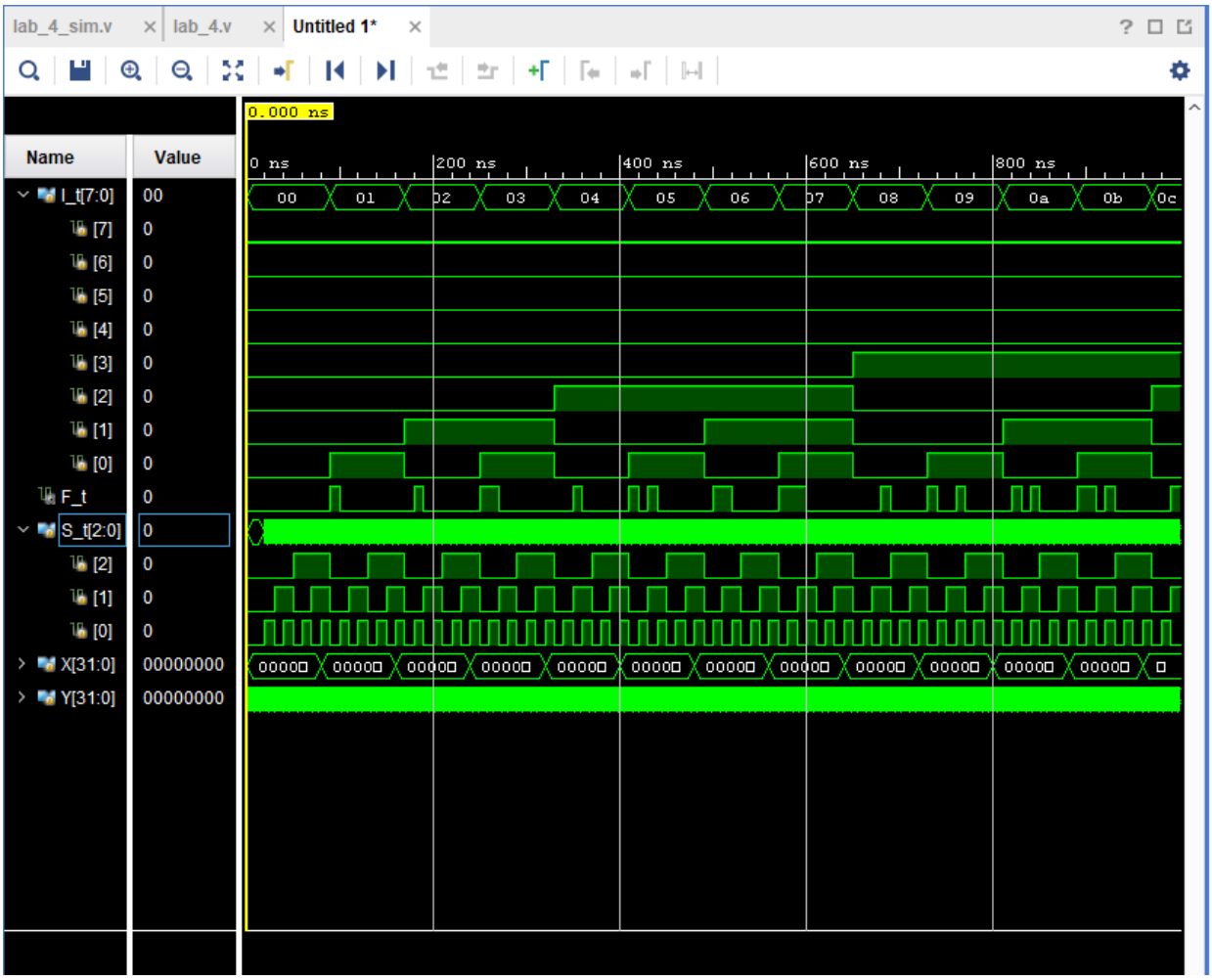
1	1	0	0
1	1	1	1

Test Plan

The design was tested through a behavioral simulation that looped through every possible input within an embedded loop to go through each selector option paired with each input option, which would give every output.

Burglar alarm controller circuit

Here is the behavioral simulation results.



Conclusion

All of my results came out as expected and agreed with each other, including the simulation. By analyzing the simulation and lookup table, with some trial and error with the programming logic, it really helped with understanding how the multiplexor works.

a) Investigate the function of a lookup table and describe how one works.

A lookup table basically is a table of values describing the functionality of a logical circuit. It will show you every combination of inputs and what the output(s) are. It's very handy for both simple and complex circuits, and a lookup table makes it very easy to calculate the min and max terms along with the function (by making a Karnaugh map and calculating it that way).

b) Consider a 16 word by 1 bit lookup table. Give the values stored in each location 0000 binary (word zero) to 1111 binary (word fifteen) for the function F(w, x, y, z). The truth table that was generated in the pre-laboratory will help here.

w	x	y	z	F
1	1	1	1	1
1	1	1	0	0
1	1	0	1	1
1	1	0	0	1
1	0	1	1	1
1	0	1	0	1
1	0	0	1	0
1	0	0	0	0
0	1	1	1	0
0	1	1	0	0
0	1	0	1	0
0	1	0	0	0

0	0	1	1	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1