Experiment 3

3 Bit Binary Adder

Digital Systems

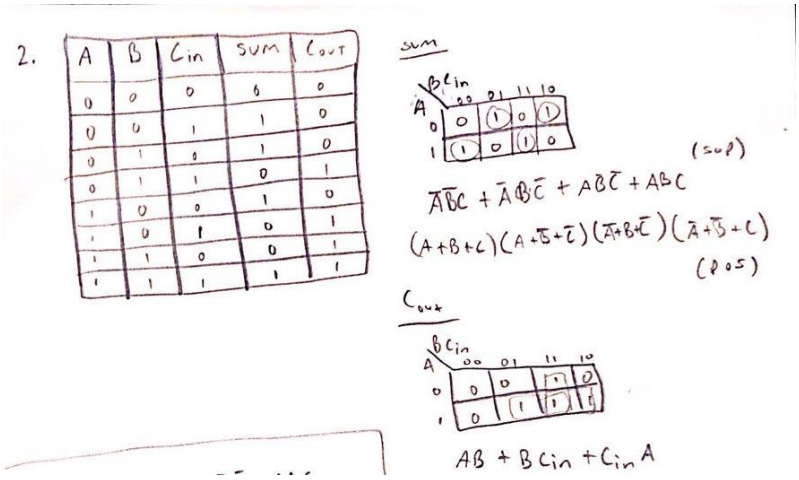
EEE3342C-20FALL 0011

Spenser Tacinelli
PID: sp498810

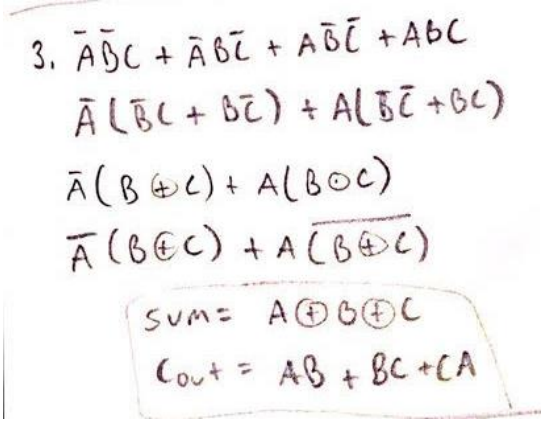
Pre-Laboratory Assignment:

1. Read this experiment carefully to become familiar with the requirements for this experiment.

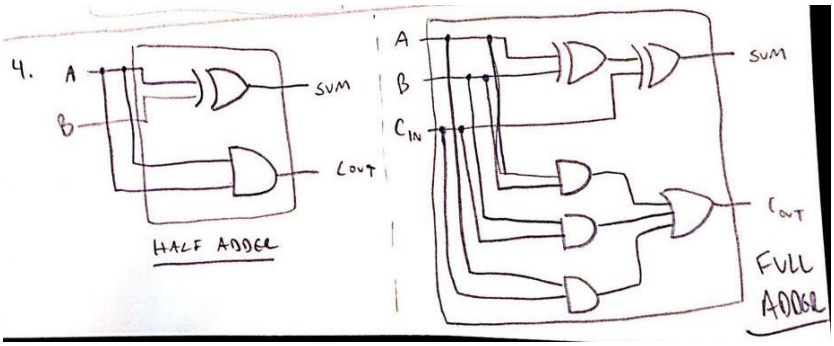
2. Prepare and complete a truth table for the full adder cell. Transfer this information to a Karnaugh Map and obtain minimum expressions in both sum of products and product of sums forms.



3. Use Boolean algebra to reduce sum of products expression to a more workable expression. (i.e. XOR's).



4. Represent the full adder and the half adder as logic diagrams.



5. Prepare a Verilog code of the complete three-bit adder circuit (your best design). It shouldnot only include pin assignments but switch and LED assignments as well. See Appendix D for pin details used by the BASYS 3 board.

```
module full_adder(A,B,Cin, SUM, Cout);
 input A,B,Cin;
 output SUM,Cout;
 assign SUM=A ^ B ^ Cin;
 assign Cout=(A & B)|(B & Cin)|(Cin & A);
end module
```

Objective:

Create a Binary Adder, that has the ability to add to binary words (3 bits), by utilizing discrete gates. And to introduce iterative cell design techniques.

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:

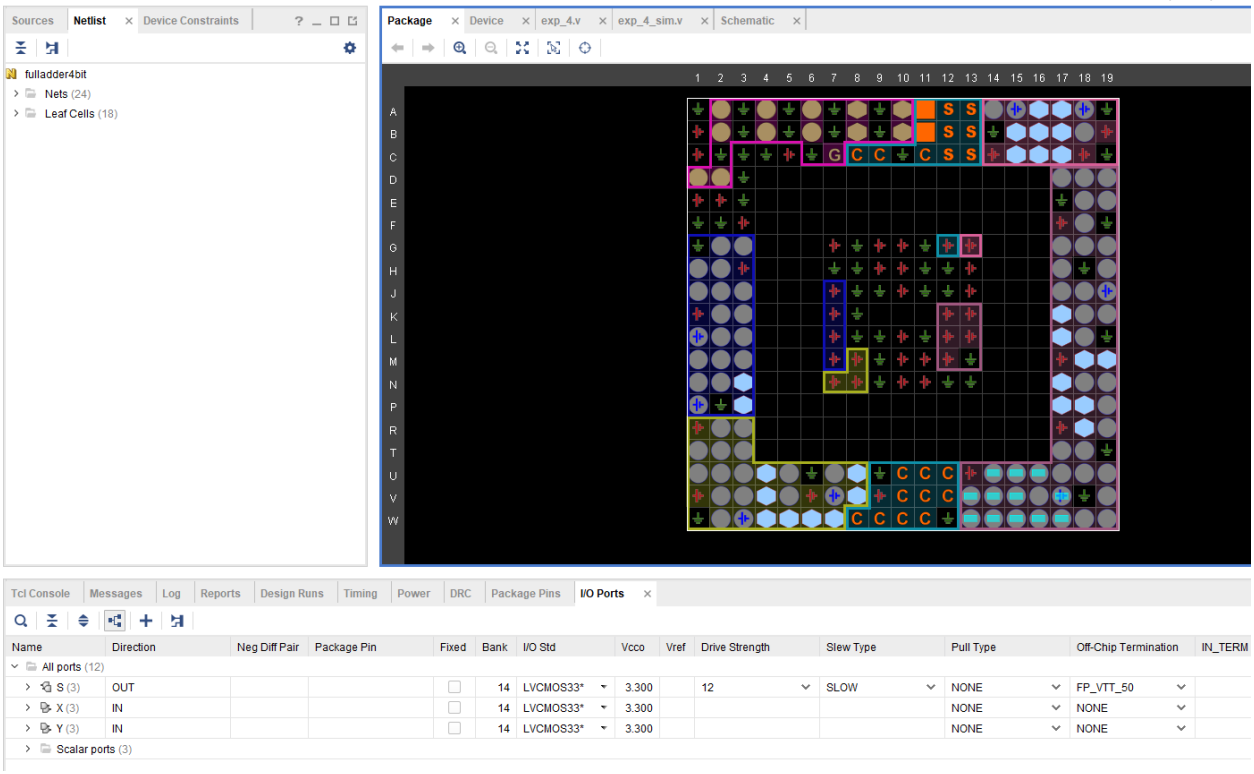
I created the project (lab_3) and added the two inputs and the four outputs. I then programmed the logic of the 3-bit binary adder.

exp_4.v | exp_4_sim.v | Untitled 1*

C:/Users/spens/Desktop/digital_systems_hdl/lab_3/lab_3.srcs/sources_1/new/exp_4.v

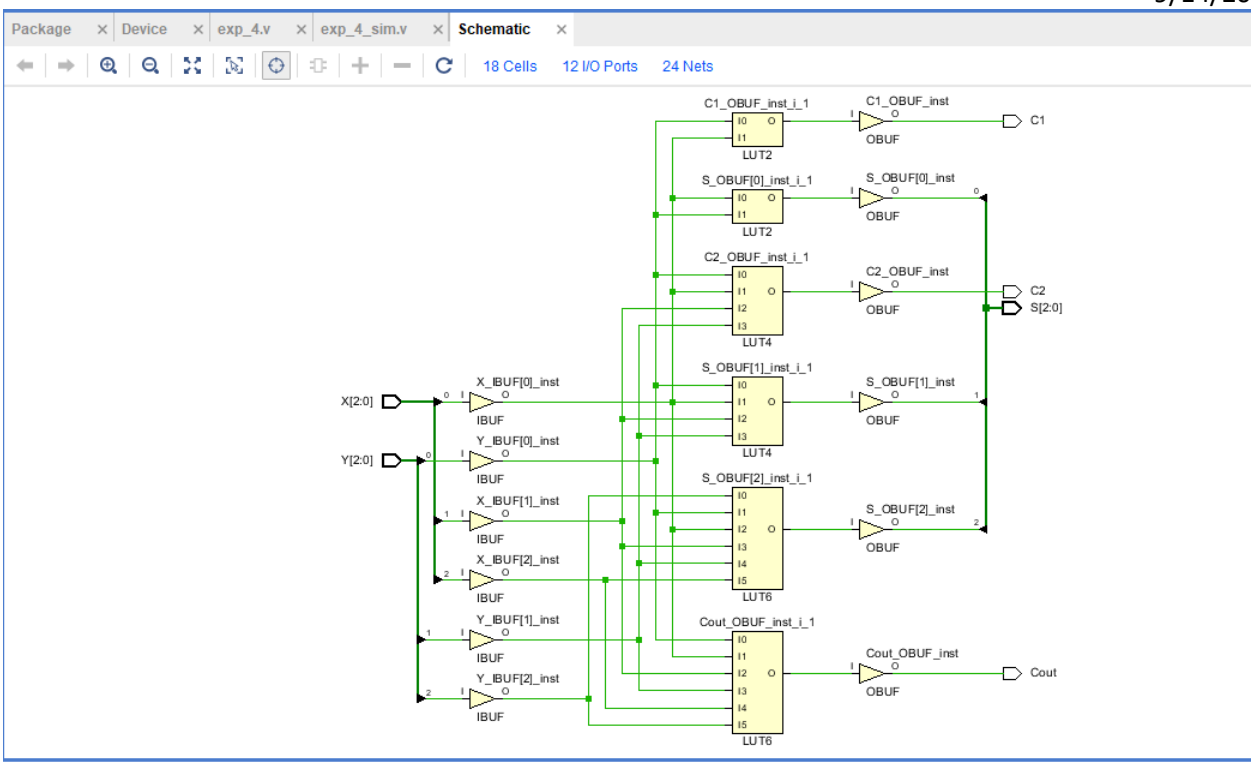
```
11  // Tool Versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////


21
22
23  module fulladder4bit(
24      input X,
25      input Y,
26      output S,
27      output Cout,
28      C1,
29      C2
30      );
31      wire [2:0] X;
32      wire [2:0] Y;
33      wire [2:0] S;
34      wire C1, C2, Cout;
35
36      assign S[0] = X[0] ^ Y[0];
37      assign C1 = X[0] & Y[0];
38      assign S[1] = X[1] ^ Y[1] ^ C1;
39      assign C2 = (X[1] & Y[1]) + (C1 & (X[1] + Y[1]));
40      assign S[2] = C2 ^ X[2] ^ Y[2];
41      assign Cout = (X[2] & Y[2]) + (C2  & (X[2] + Y[2]));
42
43  endmodule
44
```

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 3-bit binary adder circuit.

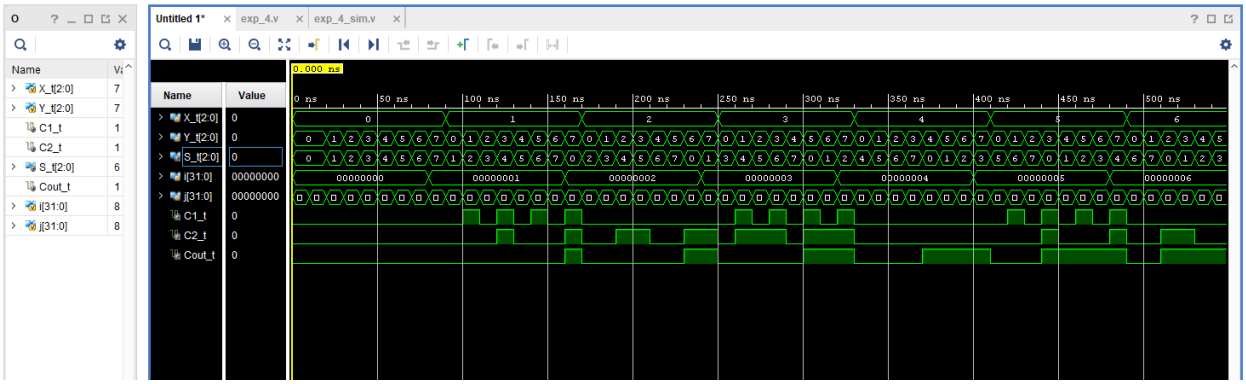
I then programmed the logic for the simulation.


Package
Device
exp_4.v
exp_4_sim.v
Schematic
18 Cells
12 I/O Ports
24 Nets
C1_OBUF_inst_i_1
LUT2
C1_OBUF_inst
OBUF
C1
S_OBUF[0]_inst_i_1
LUT2
S_OBUF[0]_inst
OBUF
C2_OBUF_inst_i_1
LUT4
C2_OBUF_inst
OBUF
C2
S[2:0]
S_OBUF[1]_inst_i_1
LUT4
S_OBUF[1]_inst
OBUF
S_OBUF[2]_inst_i_1
LUT6
S_OBUF[2]_inst
OBUF
Cout_OBUF_inst_i_1
LUT6
Cout_OBUF_inst
OBUF
Cout
X[2:0]
Y[2:0]
X_IBUF[0]_inst
IBUF
Y_IBUF[0]_inst
IBUF
X_IBUF[1]_inst
IBUF
X_IBUF[2]_inst
IBUF
Y_IBUF[1]_inst
IBUF
Y_IBUF[2]_inst
IBUF

C:/Users/spens/Desktop/digital_systems_hdl/lab_3/lab_3.srcs/sim_1/new/exp_4_sim.v

```
22
23  module exp_4_sim(
24      );
25      reg [2:0] X_t;
26      reg [2:0] Y_t;
27      wire C1_t;
28      wire C2_t;
29      wire [2:0] S_t;
30      wire Cout_t;
31      integer i;
32      integer j;
33
34      fulladder4bit UUT(
35      .X(X_t),
36      .Y(Y_t),
37      .S(S_t),
38      .C1(C1_t),
39      .C2(C2_t),
40      .Cout(Cout_t)
41      );
42
43      initial
44      begin
45      X_t[2:0]=1'd0;
46      Y_t[2:0]=1'd0;
47
48        for (i = 0; i<= 7; i = i + 1) begin
49          for (j = 0; j <= 7; j = j + 1) begin
50            #10 X_t = i;
51             Y_t = j;
52
53          end
54        end
55      end
56  endmodule
57
```

I then ran a behavioral simulation.



(behavioral)

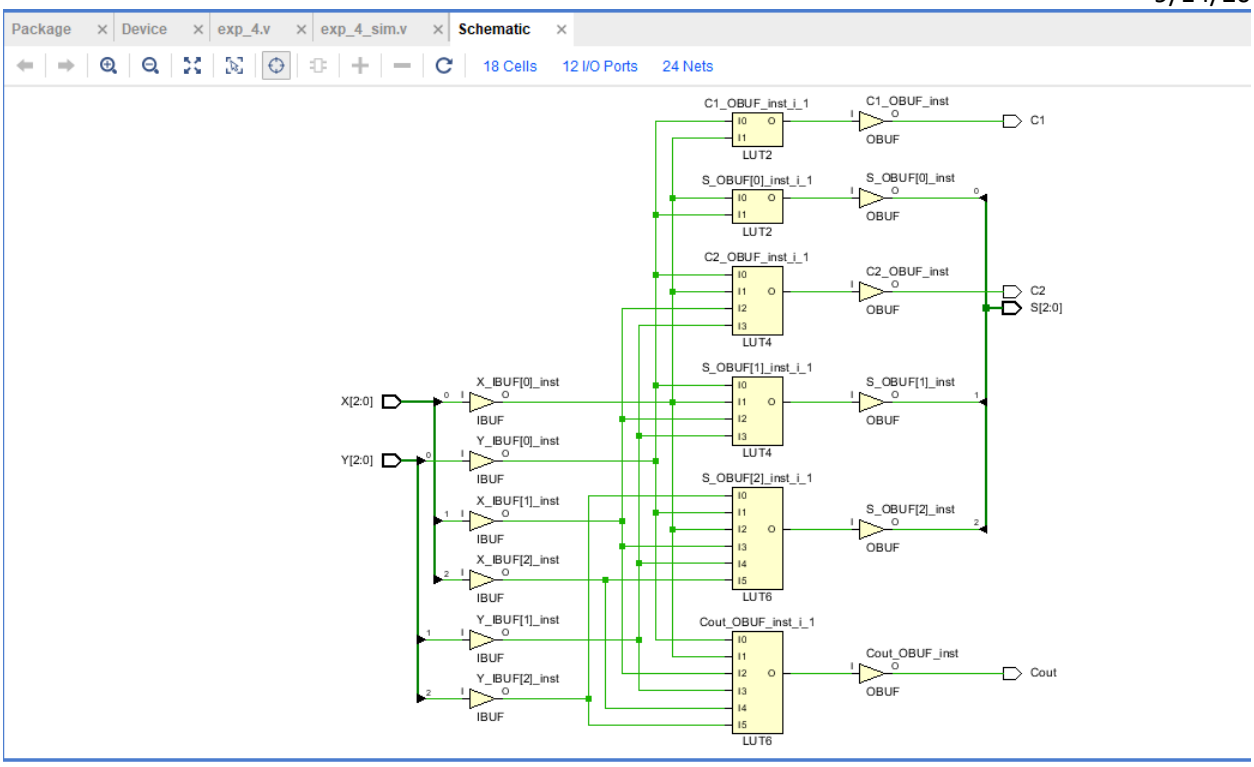
3-bit binary adder circuit Schematic

Logic Diagram:

The resulting schematic diagram for the 3-bit binary adder.

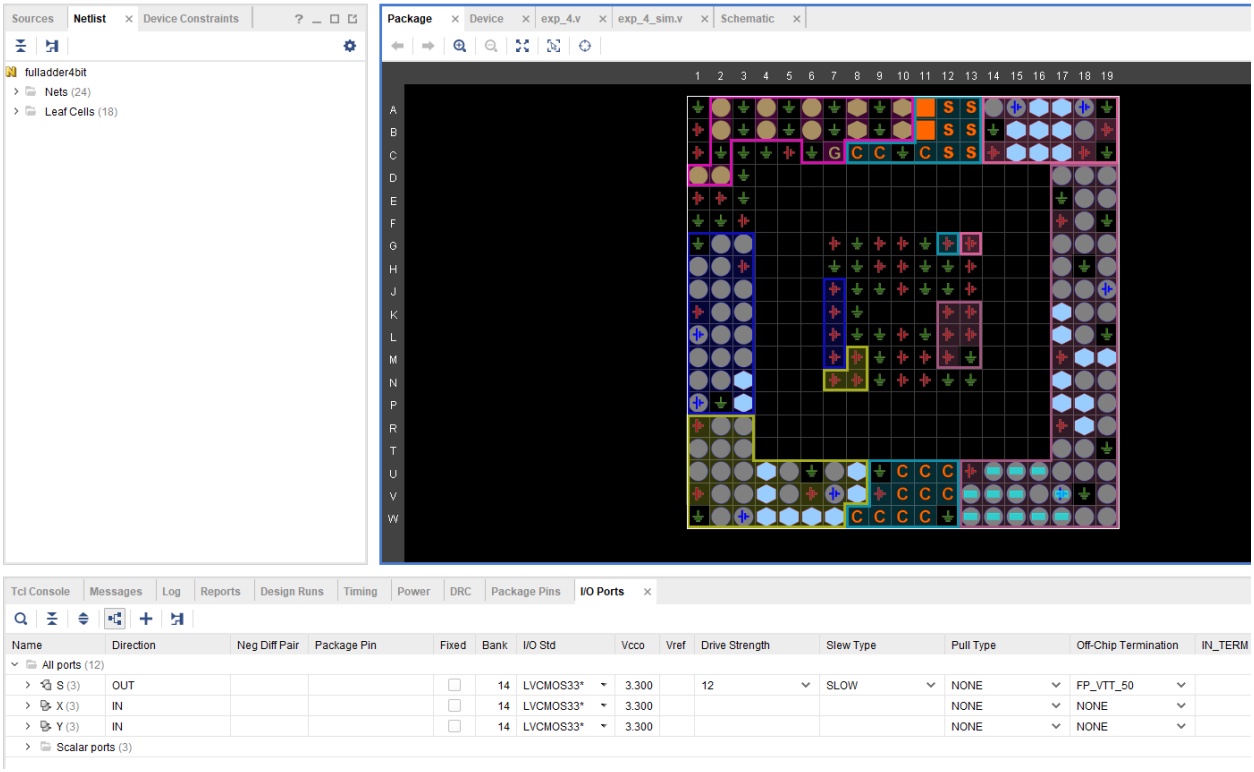
Design Specification Plan:

The resulting schematic diagram for the 3-bit binary adder.



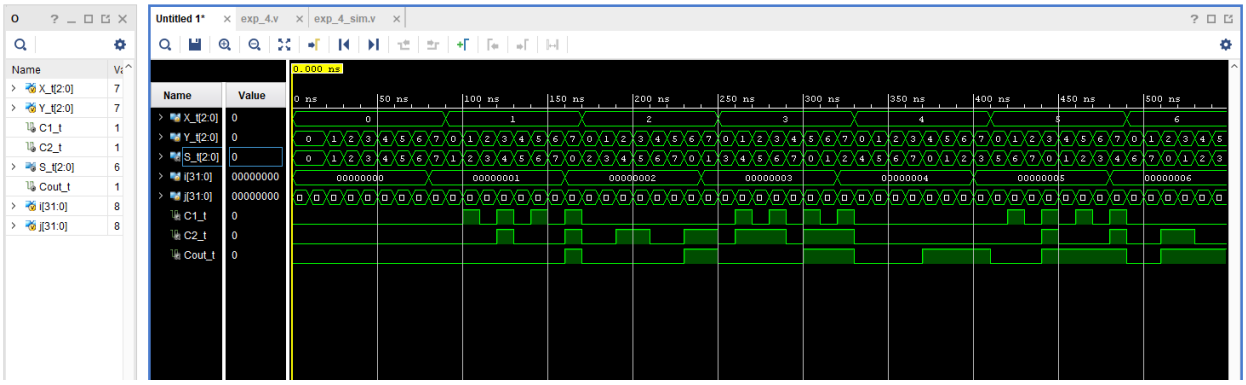
Design Specification Plan:

This is the design specification for a 3-bit binary adder. It has two inputs (X, Y) and four outputs (S, Cout, C1, C2).



Results Statement:

Here is the behavioral simulation results for the 3-bit binary adder.

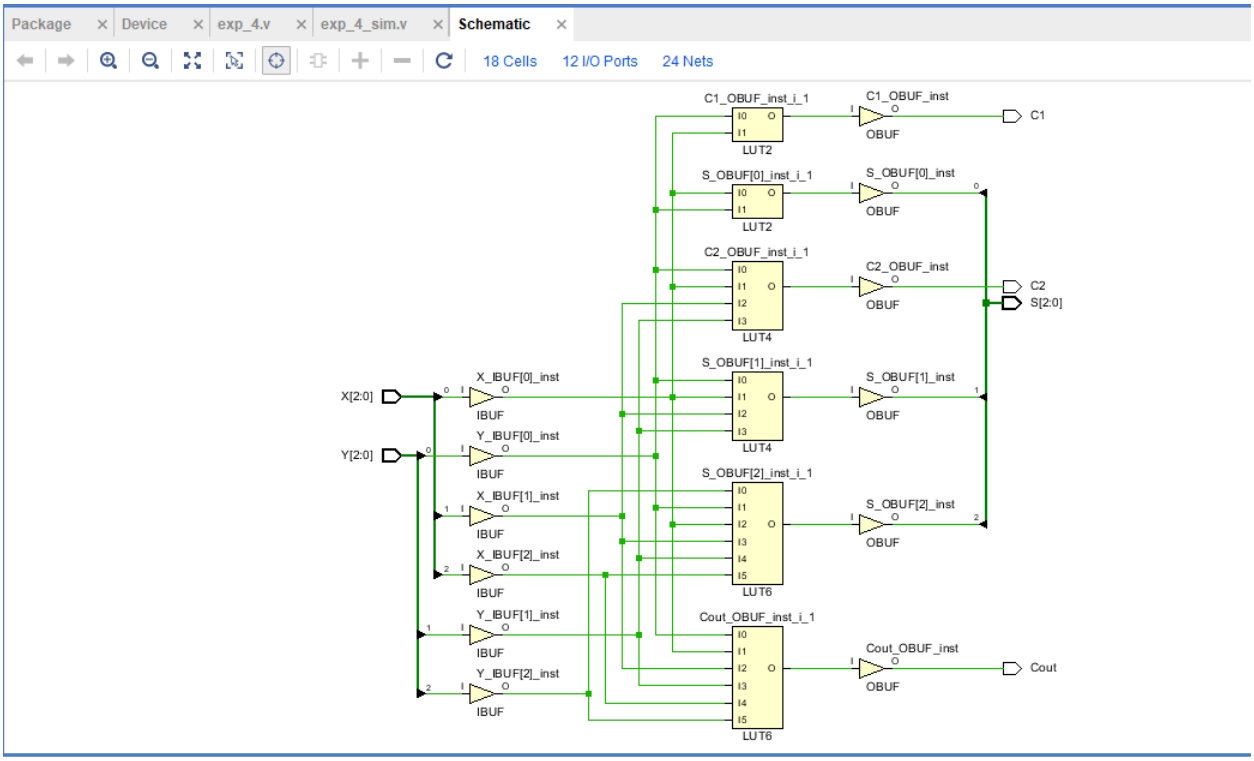


A	B	Cin	SUM	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

3-bit binary adder circuit Verilog

Logic Diagram

The results from the simulation testing for the simulation for 3-bit binary adder circuit.

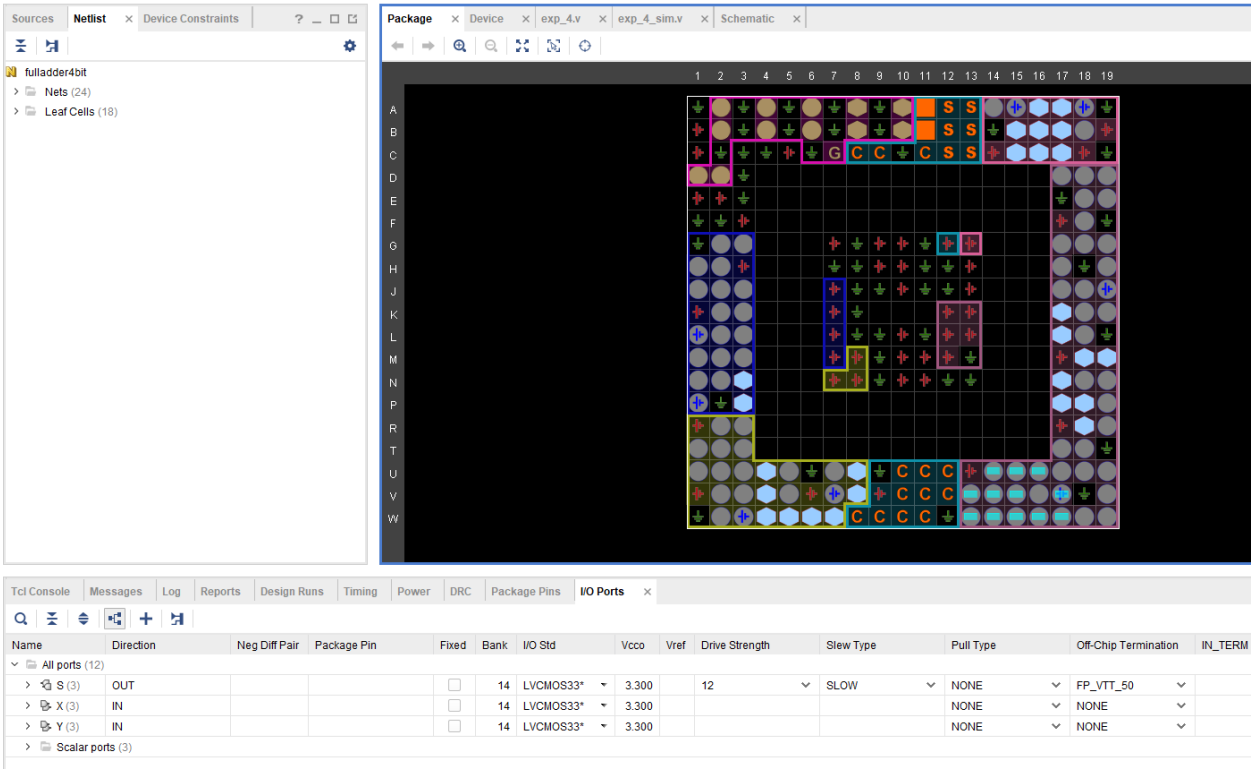


Design Specification Plan

This is the design specification for a 3-bit binary adder. It has two inputs (X, Y) and four outputs (S, Cout, C1, C2).

C:/Users/spens/Desktop/digital_systems_hdl/lab_3/lab_3.srcs/sim_1/new/exp_4_sim.v

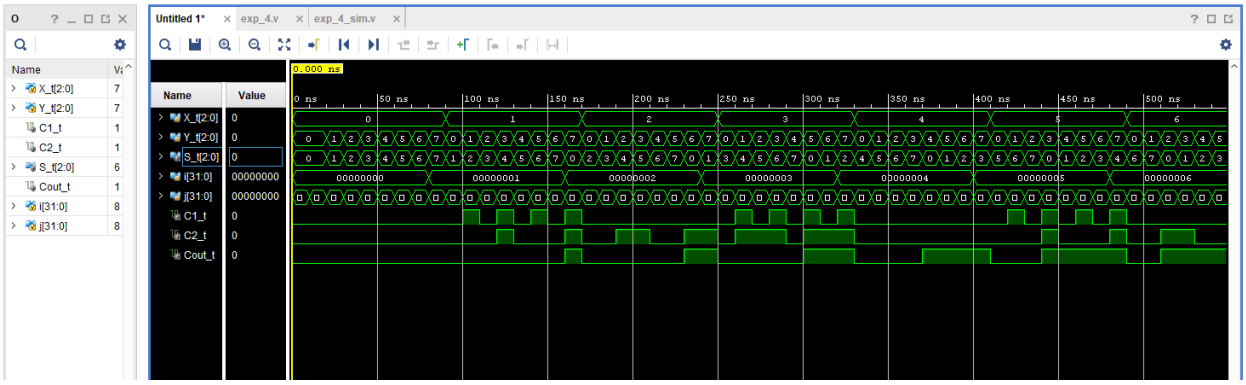
```
22
23  module exp_4_sim(
24      );
25      reg [2:0] X_t;
26      reg [2:0] Y_t;
27      wire C1_t;
28      wire C2_t;
29      wire [2:0] S_t;
30      wire Cout_t;
31      integer i;
32      integer j;
33
34      fulladder4bit UUT(
35      .X(X_t),
36      .Y(Y_t),
37      .S(S_t),
38      .C1(C1_t),
39      .C2(C2_t),
40      .Cout(Cout_t)
41      );
42
43      initial
44      begin
45      X_t[2:0]=1'd0;
46      Y_t[2:0]=1'd0;
47
48        for (i = 0; i<= 7; i = i + 1) begin
49          for (j = 0; j <= 7; j = j + 1) begin
50            #10 X_t = i;
51             Y_t = j;
52
53          end
54        end
55      end
56  endmodule
57
```



Results Statement

Explanation of pictures and tables.

Here is the behavioral simulation results.



A	B	Cin	SUM	Cout
0	0	0	0	0
0	0	1	1	0

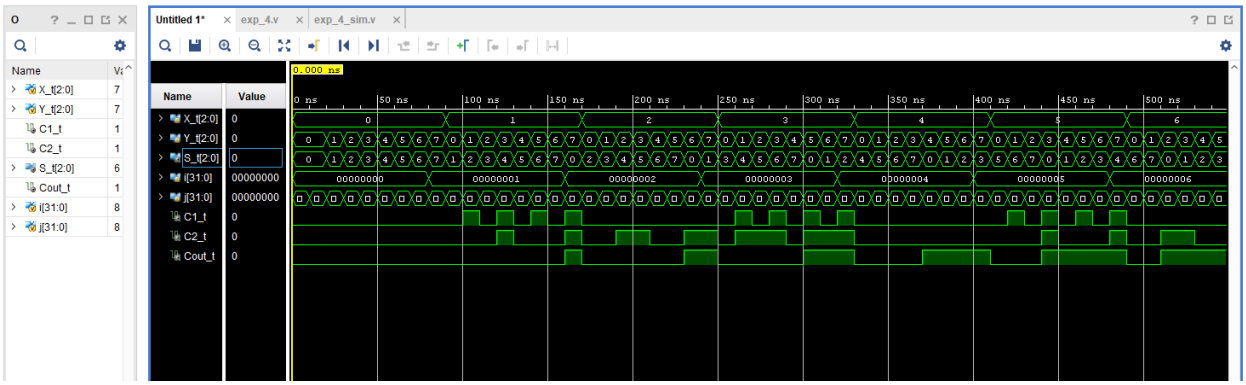
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Test Plan

The design was tested through a behavioral simulation that looped through every possible input, which would give every output.

<u>Burglar alarm controller circuit</u>

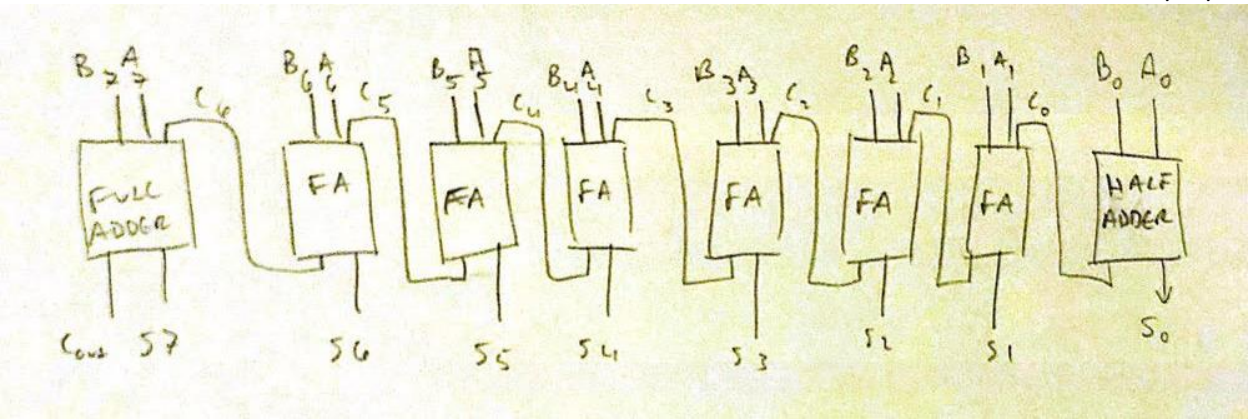
Here is the behavioral simulation results.



Conclusion

My results were what I expected, and all the test and simulation results were correct and agreed with the logic that I had programmed. By running the gate through a simulation and then analyzing it was interesting, and lead to further understanding of the logic gates functionality.

1. Using full adder and half adder block diagrams, draw an 8-bit adder diagram.



2. Comment on the feasibility of designing an 8-bit adder using the brute force method.

With this method, we will have to derive eight equations, and this would be quite hard to do. It is possible, but it would be tedious and error prone.

3. Identify the advantages and disadvantages of the brute force method.

The advantage would be that the circuit would be very fast. The disadvantage is it would be difficult to make.

4. Identify the advantages and disadvantages of the iterative cell method.

One advantage is that the design is quite simple when compared to other designs. This is because a full adder is utilized when adding the significant bits from two numbers. The major disadvantage is that it is extremely slow. This is because of the carry from LSB to MSB.

5. Have you met all the requirements of this lab (Design Specification Plan)?

Yes.

6. How should your design be tested (Test Plan)?

The design was tested through a behavioral simulation that looped through every possible input, which would give every output by utilizing the programming technique of a for loop.