Christopher Badolato

Experiment 3 Three-Bit Binary Adder

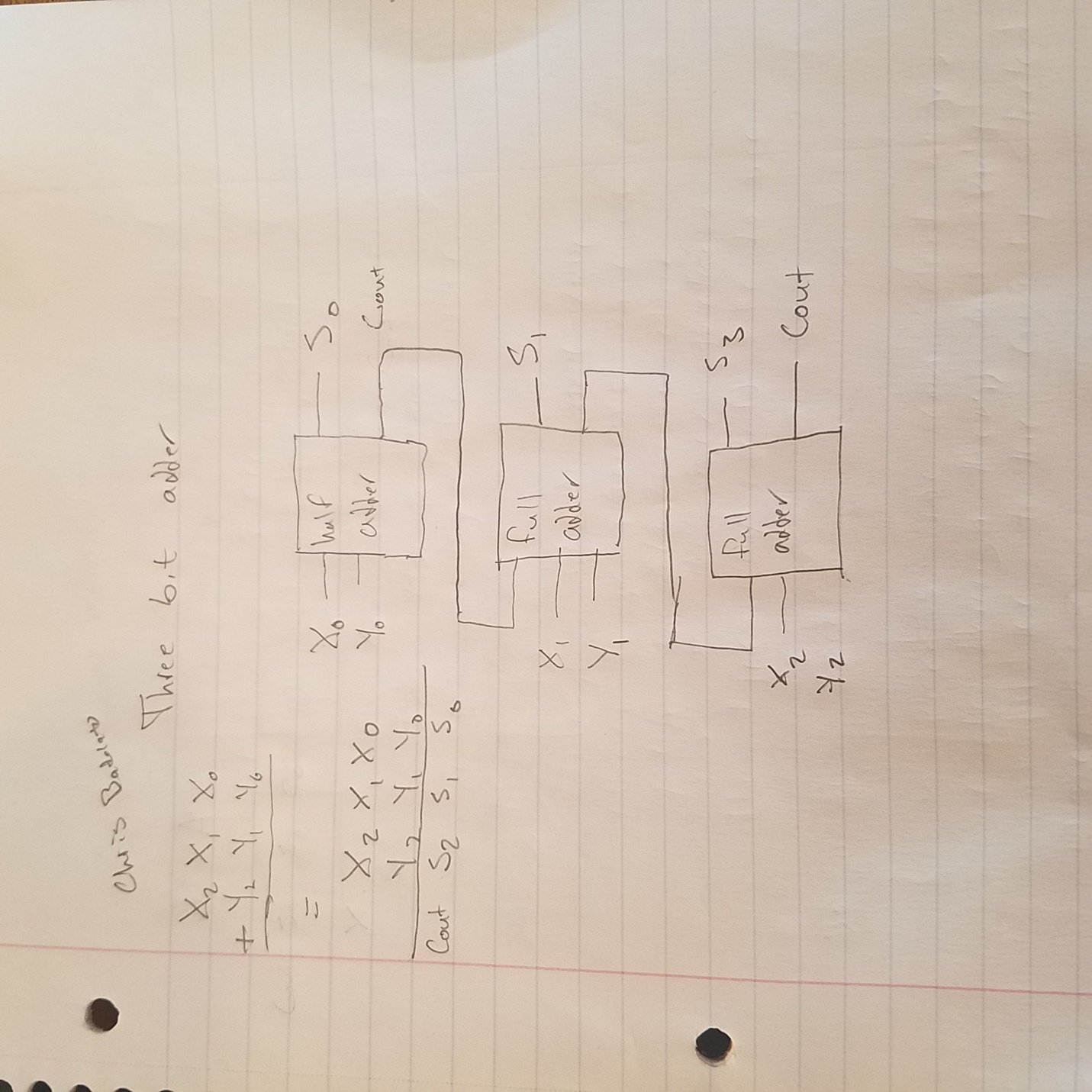
EEE 3342 LAB 0014

10/7/2018

**Objective:**

* We will design a Thee-Bit Binary adder using discrete gates
* Using a one-bit half adder and two one-bit full adders we will create this three-bit adder.

Block Diagram:

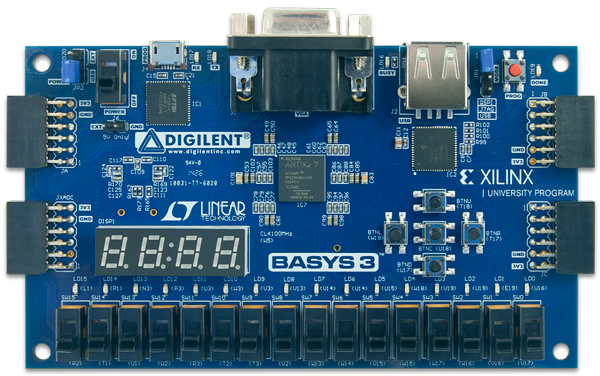


Apparatus List:

Xilinix’s FPGA VIVADO HLX edition

VIVDAO 2017.4

Basby Atrix-7



Procedure and/or Design Methodology:

First to create the half adder we just need our values for our least significant bit (X0) and (Y0)

We can combine these values in two ways for a sum and a carry.

Our sum (S0) and our Carry (C1)

S0 = X0 + Y0

C1 = X0\*Y0

We can now use our carry to retrieve our next sum value within the full adder

Our next sum S1

S1 = X1 ^ Y1 ^ C1 each term is XOR’d

C2 = (X1\*Y1) + (C1(X1^Y1))

And finally

We can get our final outputs (Most significant bit and carry)

S2 = X2 ^ Y2 ^ C2 each term is XOR’d

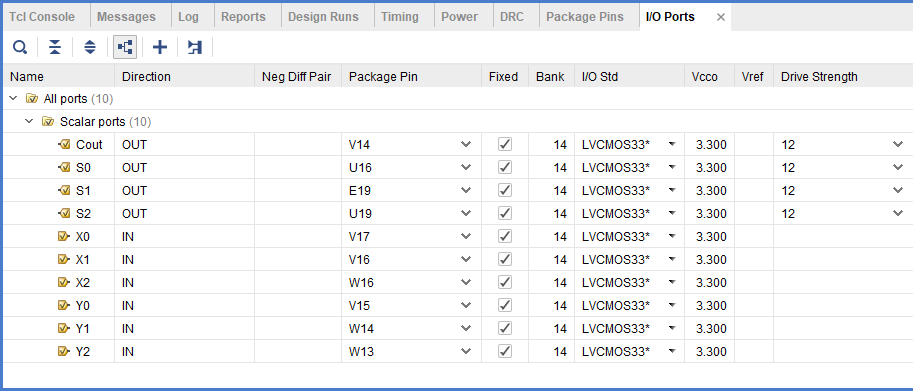
Cout = (X2\*Y2) + (C2(X2^Y2))

Design Specification Plan:

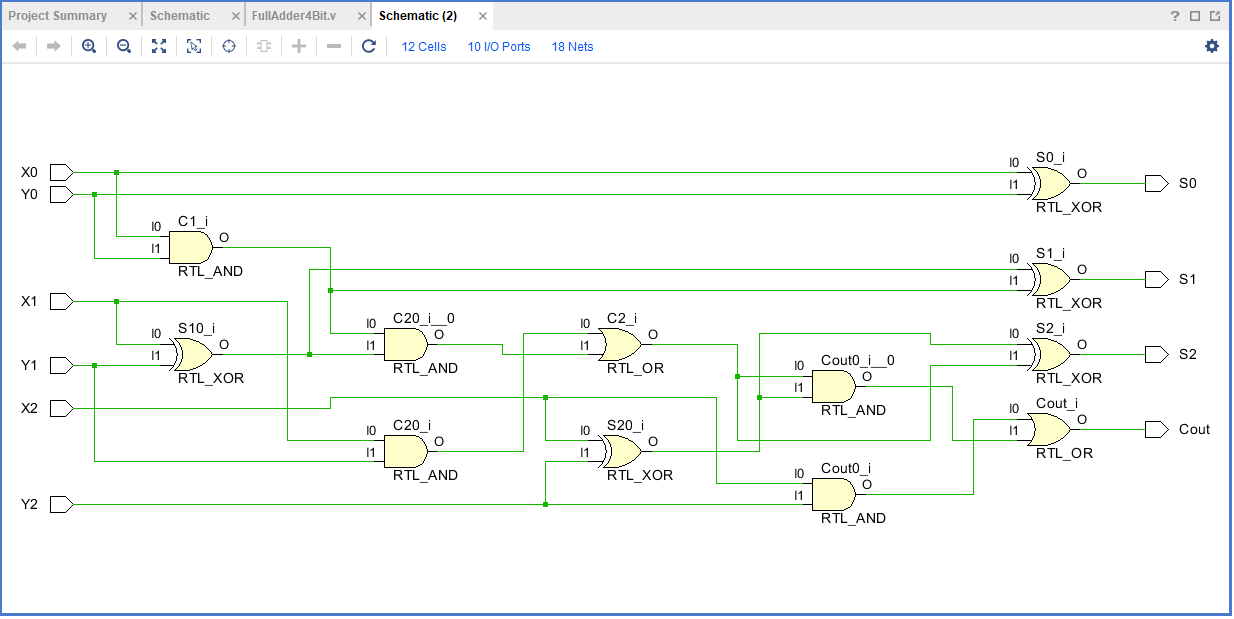
Using the below code along with our Vivado program we can create our 3 bit adder on the Atrix-7 board

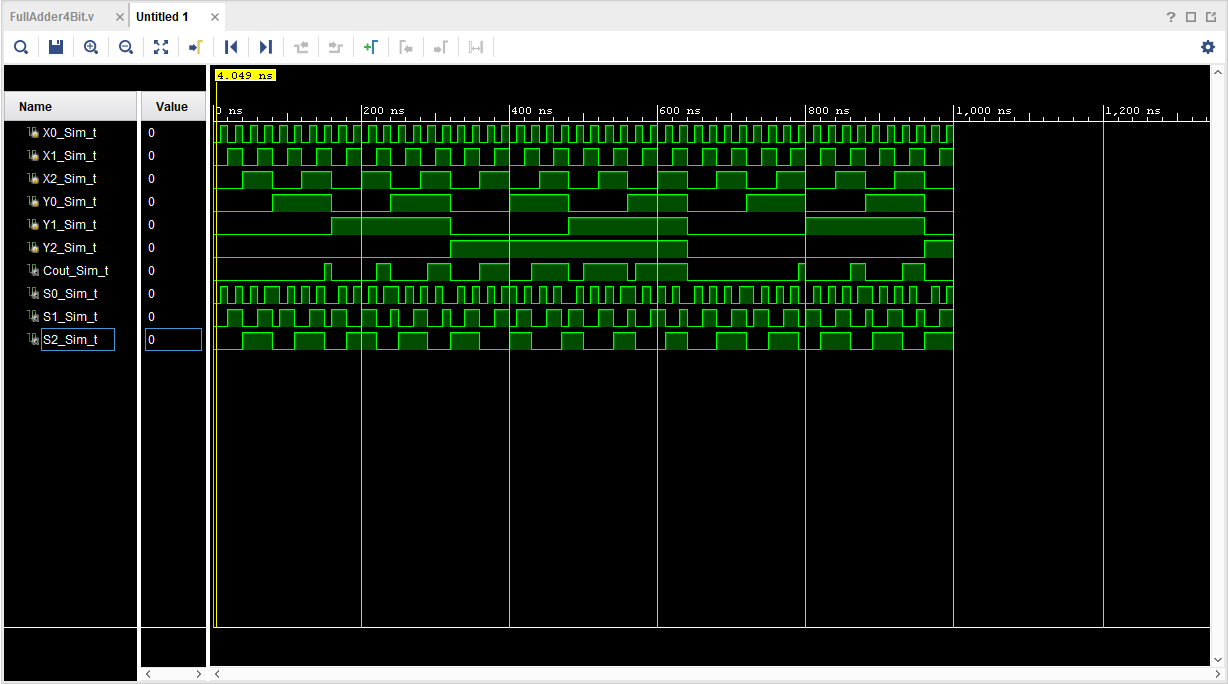
FullAdder4Bit.v testbench.v

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  module FullAdder4Bit(  input wire X0,  input wire X1,  input wire X2,  input wire Y0,  input wire Y1,  input wire Y2,  output wire S0,  output wire S1,  output wire S2,  output wire Cout  );  wire C1;  wire C2;  assign S0 = X0 ^ Y0;  assign C1 = X0 & Y0;  assign S1 = X1 ^ Y1 ^ C1;  assign C2 = (X1 & Y1) | C1&(X1^Y1);  assign S2 = X2 ^ Y2 ^ C2;  assign Cout = (X2 & Y2) | C2&(X2^Y2);  endmodule | `timescale 1ns / 1ps  module testbench(  );  reg X0\_Sim\_t;  reg X1\_Sim\_t;  reg X2\_Sim\_t;  reg Y0\_Sim\_t;  reg Y1\_Sim\_t;  reg Y2\_Sim\_t;  wire Cout\_Sim\_t;  wire S0\_Sim\_t;  wire S1\_Sim\_t;  wire S2\_Sim\_t;  FullAdder4Bit UTT (  .X0(X0\_Sim\_t),  .X1(X1\_Sim\_t),  .X2(X2\_Sim\_t),  .Y0(Y0\_Sim\_t),  .Y1(Y1\_Sim\_t),  .Y2(Y2\_Sim\_t),  .Cout(Cout\_Sim\_t),  .S0(S0\_Sim\_t),  .S1(S1\_Sim\_t),  .S2(S2\_Sim\_t)    );  initial begin  X0\_Sim\_t = 1'b0;  X1\_Sim\_t= 1'b0;  X2\_Sim\_t= 1'b0;  Y0\_Sim\_t= 1'b0;  Y1\_Sim\_t= 1'b0;  Y2\_Sim\_t= 1'b0;  end    always #10 X0\_Sim\_t = ~X0\_Sim\_t;  always #20 X1\_Sim\_t = ~X1\_Sim\_t;  always #40 X2\_Sim\_t = ~X2\_Sim\_t;  always #80 Y0\_Sim\_t = ~Y0\_Sim\_t;  always #160 Y1\_Sim\_t = ~Y1\_Sim\_t;  always #320 Y2\_Sim\_t = ~Y2\_Sim\_t;    endmodule |



Detailed Schematic Diagram:





Test Plan:

Results:

The results of the experiment are a three-Bit binary Adder that will include the fourth carry digit to the left of the results.

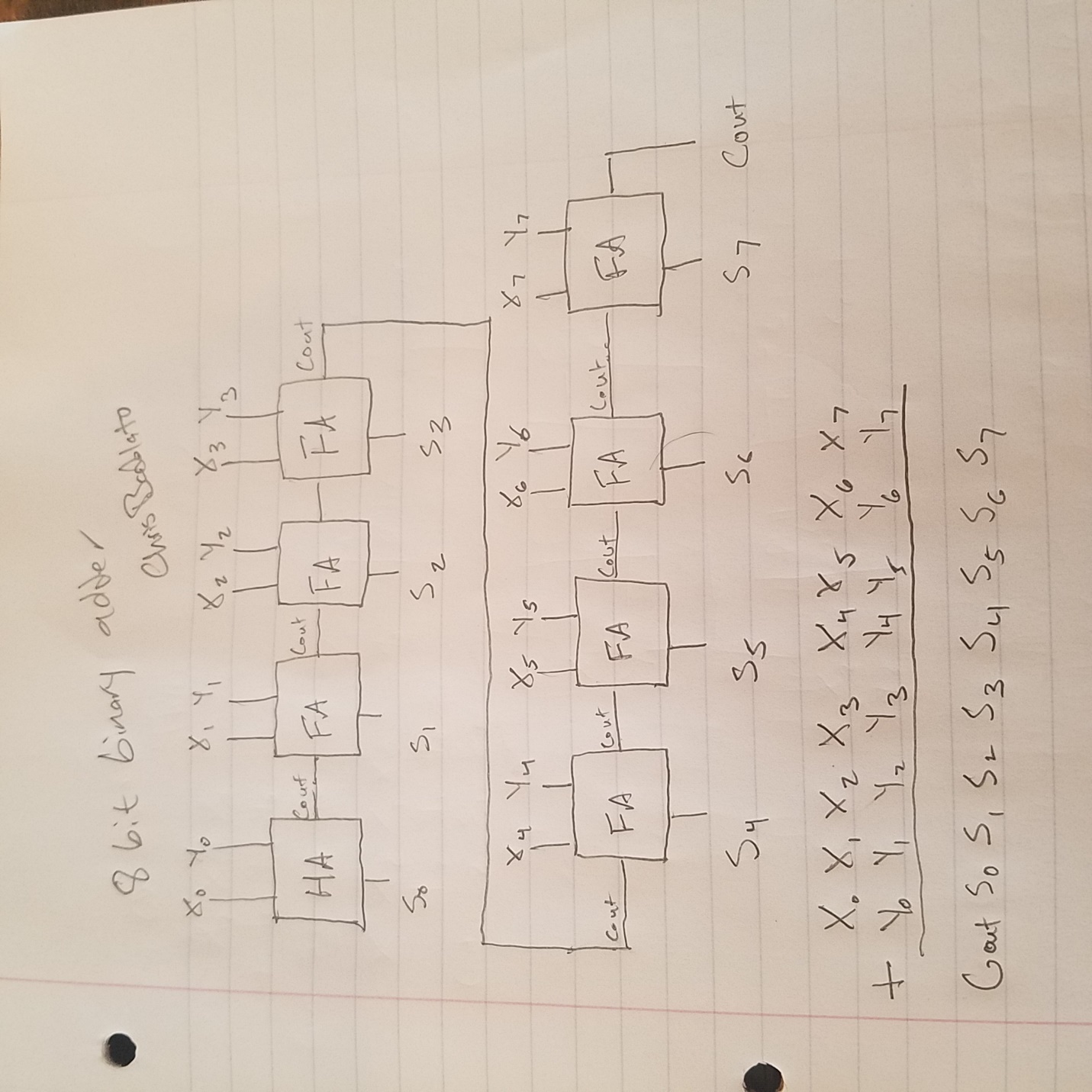
Conclusion

Using a half adder along with two full adders we can do binary addition of three numbers including a carry value. The half adder which does not take in a carry bit is used to add the first two least significant bits. The half add has a carry output that will be used in the addition of the next bit. The full adder includes a carry input allowing us to do the addition with the carry until the most significant bit is reached.

Questions:

(To be incorporated in the Conclusion section of your laboratory report.)

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



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

With brute force method we would have to derive an equation with 8 different variables is difficult but it is feasible and fast!

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

The advantages of using the brute force method is that it is fast, but to its disadvantage it is difficult to design.

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

With iterative with have the advantage of the design being very simple since each significant bit is parallel to the bit to be added. The first half adder adding X0 to Y0 to form our Sum (S0) and Carry (C0). The next Full adder adds the carry bi t(C0) to X1 and Y1 to form S1 and C1. Finally C1 adds to Y2 and X2 to form S2 and Cout.

The disadvantage is that it is very slow since we need to carry from least significant bit to the most significant bit.

1. Have you met all the requirements of this lab (Design Specification Plan)?
2. How should your design be tested (Test Plan)?

To test this design, we must first upload our code to Vivado and upload the software to our board with the selected pins from the IO ports shown before.

We can test the software by adding 110 + 110. This will mean, switch for X0, X1, Y0 and Y1 will all be switched up (The bit is one). X2 and Y2 will but left down. The board will do the addition and light up the corresponding lights to the sum with the carry bit as the most significant bit.

1100 will be our answer means Cout, and S0 will turn on and S1 and S2 will be shut off.