

PRELAB!

Read the entire lab and **complete** the prelab questions (Q1-Q3) on the answer sheet **before** coming to the laboratory.

1.0 Objectives

In the last lab, we used an adder to add two unsigned binary numbers. The objective of this lab is to understand binary number representation and basic binary arithmetic through the implementation of a 2's complement adder/subtractor. You will design and implement the circuit and explain the results to your TA to demonstrate your understanding of binary arithmetic.

NOTE: Read through the lab assignment and execute what you are asked to. Unlike previous labs, the steps will not be outlined for you, as you by now already have enough understanding of the basic tools necessary for design. Consult previous labs as necessary. It is strongly recommended that you still create a new project for each part of the lab.

2.0 Setup

2.1 Problem Definition

To build an adder/subtractor with the following specifications:

- Two 4-bit binary numbers in 2's Complement form, $X[3..0]$ and $Y[3..0]$.
- One 1-bit control input to specify the type of operation, i.e. if control = 0 then add X and Y else subtract Y from X.
- Produce a 4-bit result, 1-bit carry, and 1-bit overflow outputs.

2.2 One-bit Full Adder

A Half Adder takes two 1-bit inputs, X and Y, and generates two 1-bit outputs: the sum bit S and the carry bit C_{out} . A one-bit full adder is capable of adding two bits (X and Y) with a carry in (C_{in}) at the same time as shown in **Figure 1**, and generates two 1-bit outputs: the sum bit S and the carry bit C_{out} . Refer to your textbook for more details.

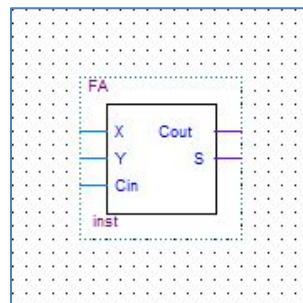


Figure 1: 1-Bit Full Adder

2.3 4-Bit Ripple Carry Adder

To build a 4-bit ripple carry adder, four 1-bit full adders can be cascaded as shown in Figure 2.

- Each 1-bit full-adder adds one bit of $X[3..0]$ to the corresponding bit of $Y[3..0]$.
- The sum bits of S form a part of the output and the generated carry bit, C_{out} , is fed into the next stage as shown in the circuit below.
- The full adder calculating the sum of the first bit (Least Significant Bit) gets a C_{in} of zero.

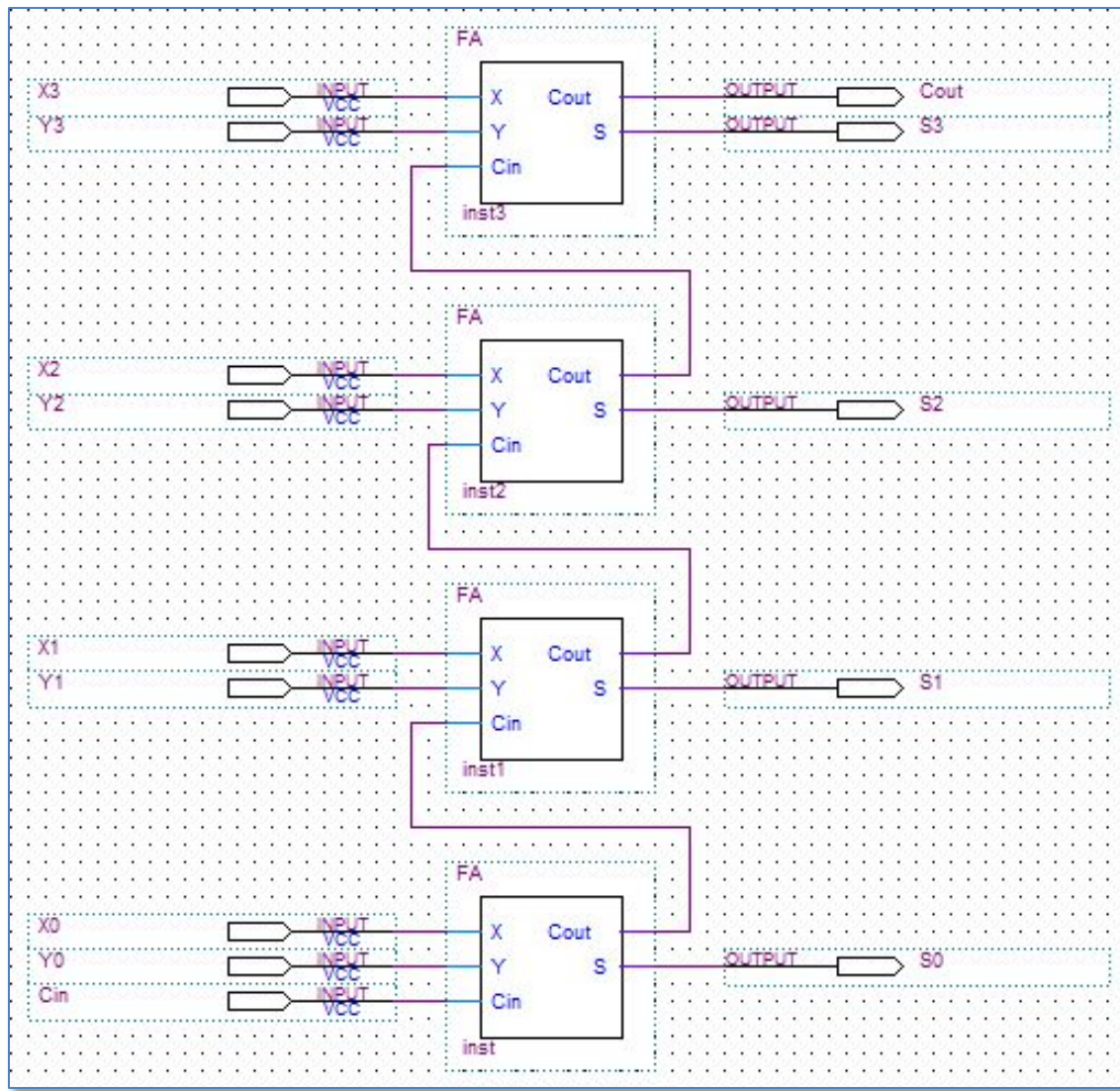


Figure 2: 4-Bit Ripple Carry Adder

2.4 Subtraction

Subtraction is, with the use of 2's complement, the same as addition. To compute $X - Y$, all we need to do is add the 2's complement for $-Y$ to X . The Ripple Carry Adder is codified to accomplish this by adding the 1's complement for $-Y$ input with a carry in $C_{in} = 1$. A control input is used to specify the subtract operation. The 1's complement for $-Y$ for subtraction is achieved by XORing each bit of Y with the control input ($=1$). The Control input is also connected to C_{in} of the 4-bit adder. This is shown in **Figure 3**. When Control (C_{in}) is 0, the circuit adds $X[3..0]$ to $Y[3..0]$, and when Control is 1, it subtracts $Y[3..0]$ from $X[3..0]$.

2.5 Overflow:

The result of arithmetic operations should fit in the bits available for output. If the result of an arithmetic operation **doesn't fit** into the bits provided, then an arithmetic overflow occurs. Overflow can be detected by comparing the carry-in and carry-out of the sign bit in the numbers.

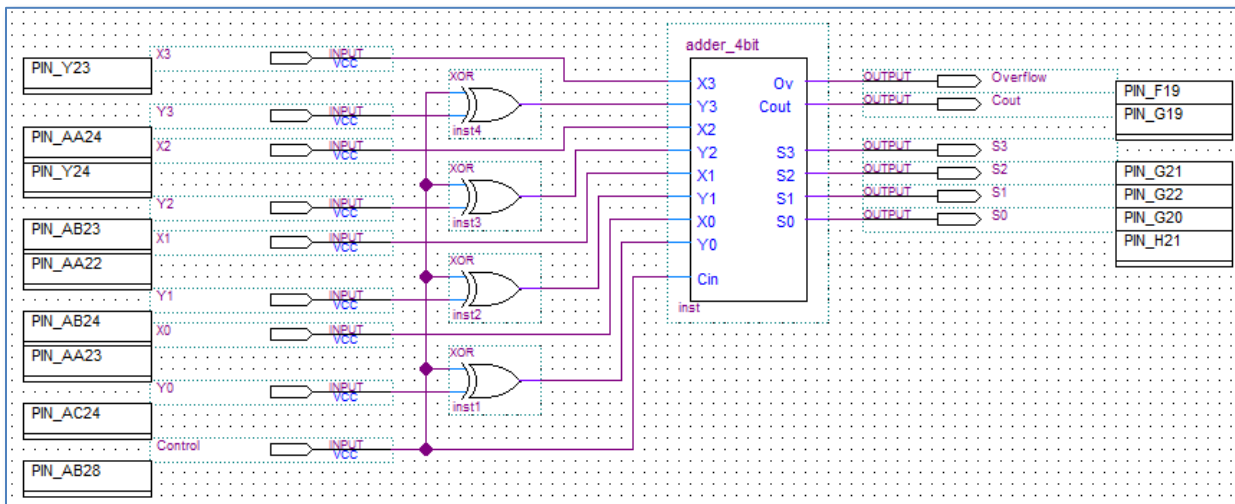


Figure 3: Adder/Subtractor

3.0 The Design

3.1 Implementation Procedure:

- Implement a full adder in Verilog (FA.v) as shown below.


```

module FA (Cin, X, Y, S, Cout);
input Cin, X, Y;
output Cout, S;
assign S = (expression for S);
assign Cout = (expression for Cout);
endmodule

```

- Compile this code and generate a default symbol for this.
- Open a new **.bdf** file (*adder_4bit.bdf*) and construct the 4-bit adder using the symbol for the Full Adder. The connections are shown in the diagram for the ripple carry adder. Add the XOR gate required for calculation of overflow.
- Generate a default symbol for this adder.
- Now in a different **.bdf** file (*add_sub.bdf*) make the final Adder/Subtractor.
- The overall schematic you should have at the end should look similar to the one shown in **Figure 3**.

3.2 Testing

Demonstrate your design to the TA by showing the results in ModelSim. Generate Verilog files for each component of your project and import these three Verilog files to ModelSim. To test this lab, you will create your own DO file. At a minimum, you should perform the six additions in the prelab Q1 to test your circuit.

4.0 Complete

You are done with this lab. Make sure to close all lab files, exit Quartus Prime, log off the computer, power down the DE2-115 board, and hand in your answer sheet. **Don't forget to write down your name and your lab section number.**