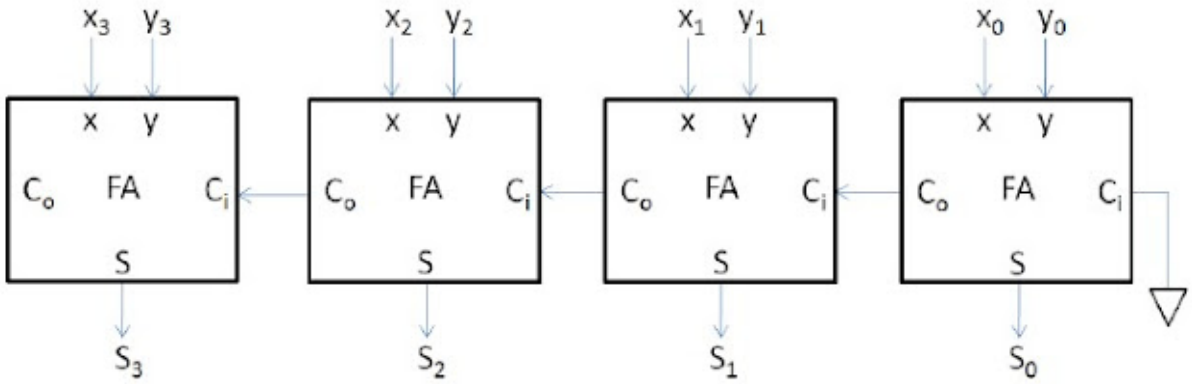
PHYS 234 Lab 6: Structural Modeling for a Ripple Carry Adder

Ethan Yang

The goal of this lab is to construct a 4-bit ripple carry adder using structural modeling in Verilog. Structural modeling describes a circuit by connecting smaller modules to build a larger system. In this case, we are using logic gates to build a ripple carry adder. A ripple carry adder is a circuit that adds 2 multi-bit binary numbers. It connects several 1-bit full adders in series, and each adder's carry output will carry into the next adder's carry input, as shown below:



The inputs for the full adder X, Y, Ci and the outputs Co and S are listed in the module parameters below. The XOR operations are used for calculating the sum, and the AND and OR operations are used for calculating the carry.

The ripple_adder connects the four full adders. Each X, Y, and S are entered to their corresponding parameters as in the figure above. Each adder passes its carry out (Co) to the next adder's carry in (Ci). The first full adder's carry in is 0, and the last full adder's carry out it the output Co.

Verilog Code:

```
`timescale 1ns / 1ps

module fulladder(
      input X,
      input Y,
      input Ci,
      output S,
      output Co
);
      wire w1, w2, w3;

      assign w1 = X ^ Y;
```

```
      assign S = w1 ^ Ci;
      assign w2 = w1 * Ci;
      assign w3 = X * Y;
      assign Co = w2 + w3;
endmodule

module rippe_adder(
      input [3:0] X,
      input [3:0] Y,
      output [3:0] S,
      output Co
);
      wire w1, w2, w3;

      fulladder u1(X[0], Y[0], 0, S[0], w1);
      fulladder u2(X[1], Y[1], w1, S[1], w2);
      fulladder u3(X[2], Y[2], w2, S[2], w3);
      fulladder u4(X[3], Y[3], w3, S[3], Co);
endmodule
```

As a result, the ripple carry adder can successfully compute the binary sum of two 4-bit numbers. For example, adding X = 0101 and Y = 0011 produces S = 1000 and Co = 0. For another instance, adding X = 1011 and Y = 0101 produces S = 0000 and Co = 1, representing a sum of 16 and overflow.

Lab 6: Structural Modeling of a Ripple Carry Adder

Verilog makes addition easy: assign y=a+b. But how do logic gates actually perform this operation, if, for example, y, a, and b are each 4 bits? In this lab, we will see how to construct an 4-bit ripple carry adder, out of logic gates. This is called structural modeling.

We could explicitly use 20 logic gates, but that's unnecessarily tedious. Instead, we'll use five logic gates to make a full adder. Then we "instantiate" the full adder four times.

Study https://www.fpga4student.com/2018/07/verilog-code-for-ripple-carry-adder.html. The webpage shows how a 4-bit ripple carry adder can be constructed from four full adders, and how a full adder can be constructed from five logic gates.

The full adder module in FPGA4student uses a different style than we're familiar with. Put it into the familiar style: put the word "input" or "output" in front of each variable in the first line instead of listing inputs and outputs in two separate lines, and use assign statements for each of the five logic gates instead of what's shown (where the output is listed first, followed by the two inputs). It's helpful to label w1, w2, and w3 in the circuit diagram.

Study the ripple carry adder module in FPGA4student. See how the full adder is instantiated four times. Each instantiation is given a different arbitrary name (u1, u2, u3, u3). Each instantiation specifies the three inputs and two outputs in the same order in which they appear in the first line of the full adder module.

There's an alternative syntax for instantiation: instead of listing the inputs and outputs in the specified order, as in fulladder u4(X[3], Y[3], w3, S[3], Co), you can map each input or output in fulladder to the correct variable: fulladder u4(.X(X[3]),.Y(Y[3]),.Ci(w3),.S(S[3]),.Co(Co)).

Rewrite and test a 4-bit ripple carry adder. You don't have to test every possible combination of inputs!