

Designing and testing an adder

The main goal of this lab is to learn how to write Verilog testbench programs. Secondly, to design adders and to analyze time delays.

1. You are going to first *design a 4-bit ripple carry adder*. There is a detailed explanation (and code) of this circuit in Chapter 5 of your book. The first part of the assignment is to write a Verilog program that will implement this adder and a *Verilog testbench* to test the circuit.
2. Then create a *ripple-carry version* of the 4-bit adder, and finally a *behavioral Verilog* version (Figure 5-18).
3. Please compare the delays of each of these.

Optional: Loading to the XSA board is optional. Since you only have 4 switches on the board, you could set one of the inputs of the adder to a constant, or perhaps add some DIP switches to the board. You can use the decoder from the previous lab to see the output of the adder.

Optional: You may want to try a larger adder.

Note. If you decide to make a much larger adder, note that a limitation you will encounter is the total number of pins on the FPGA. You'll run into this even if you don't actually assign ports and load the design onto the board.

Simulation

To simulate your program, you will need to add a new source to the project and make a Verilog test fixture. This is similar to making a *Test Bench Waveform* but more convenient for testing larger designs. Add a new source and select *Verilog Test Fixture*. As you click *Next*, make sure that the new test fixture is associated with the correct Verilog code (that it is meant to test).

When you click Finish, a skeleton of the source code for a Verilog test program is generated.

Begin the test program with something similar to the code provided below. This code contains only a few of the test cases you will have to check for. Primarily, this code provides only some test cases that do not propagate the carry bit. Change this so that you can see how the carry influences the propagation delay of the addition.

Finally, make as comprehensive a testbench as possible.

```

`timescale 1ns/1ns          // Add this to the top of your file to
                             // set time scale

module testbench();

// Inputs
  reg [3:0] B;
  reg [3:0] A;
  reg C0;

// Outputs
  wire [3:0] S;
  wire C4;

// Bidirs

// Instantiate the UUT
  adder_4_v d (
    .B(B),
    .A(A),
    .C0(C0),
    .S(S),
    .C4(C4)
  );

// This just prints the results in the ModelSim text window
// You can leave this out if you want

  initial
  begin
    $monitor($time,
      "A=%b,B=%b, c_in=%b, c_out=%b, sum = %b\n",A,B,C0,C4,S);
  end

// These statements conduct the actual circuit test

  initial
  begin
    A = 4'd0; B = 4'd0; C0 = 1'b0;
    #50 A = 4'd3; B = 4'd4;
    #50 A = 4'b0001; B = 4'b0010;
    #50 A = 4'hc; B=4'h2;
  end

endmodule

```

Once you have the test file written, you can simulate it. To simulate without taking into account any gate delays, click on Simulate Behavioral Verilog Model in the Processes for Current Source Window. This is what you did last week and is a good way to test the functionality of the design.

To see the expected gate delays, double click on the Simulate Post-Place and Route Verilog Model in Processes for Current Source pane. This will place and route the design, and then open ModelSim for you.

Downloading design on the board (optional)

This is not very interesting because (a) it's difficult to add many switches, and (b) we haven't learned to use the test and measurement instruments, so can't see the real delays. However, if you want to see the fruits of your labor, you should be able to connect the inputs to appropriate ports and see (part of) the result of the addition on the seven segment display. Appropriate ports for the inputs could be the built-in DIP switches and additional switches mounted on the proto board. Alternatively, you could set one value as a constant in your Verilog code and input the other using DIP switches. DON'T FORGET TO ASSIGN PORTS.

Report

Email the text of your testbench program(s), a Zip file of the ISE project, and answer the questions below. Feel free to submit timing diagrams if they'll help you answer the questions.

Please answer the following questions.

1. Were you able to test exhaustively? How many test cases did you use? Why did you choose these?
2. Given your experience with this experiment (for example, the time it took to simulate and the time it took you to write the code), what size adder do you think could be tested exhaustively? Please justify your answer.
3. What delays did you observe, using post-place-and-route Modelsim, for the ripple-carry adder, carry-lookahead adder, and the adder generated by the synthesizer? Is this what you expected? Please explain.
4. How well do the simulated delays agree with the post-place-and-route Static Timing Report?
5. Have a look at the post-place and route simulation. What happens to the output signal (the sum) between the times the inputs change and the final value settles? Can you explain this behavior?
6. What placement and routing factors on the FPGA will affect the delays?

Things you learned:

- How to simulate the design using a Verilog program
- Simple Verilog syntax