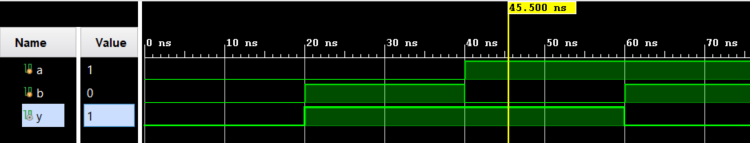
EENG 2131 - Lab 2

Basic Verilog code and simulation

# Antonio Veguilla Hernandez

Student ID: P000166790

# Resources

* Getting Started with Vivado: <https://digilent.com/reference/vivado/getting_started/2018.2>
* To do simulation:
  1. Create a testbench module that instantiates and tests the module under test.
  2. Go to Project Settings 🡪 Simulation 🡪 Simulation top module name, then use the … button to select the module name from your testbench, then click OK.
  3. Click Run Simulation 🡪 Run Behavioral Simulation
  4. Use the waveform view to explore the behavior of the circuit when exercised by the testbench:  
     

# Part 1: Design + simulate a simple AND-OR logic circuit.

1. Write down the truth table for the following logic circuit. Diagram

   Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | F |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

1. Write the Verilog design module for the logic circuit, using base gates.

Graphical user interface, text, application, email

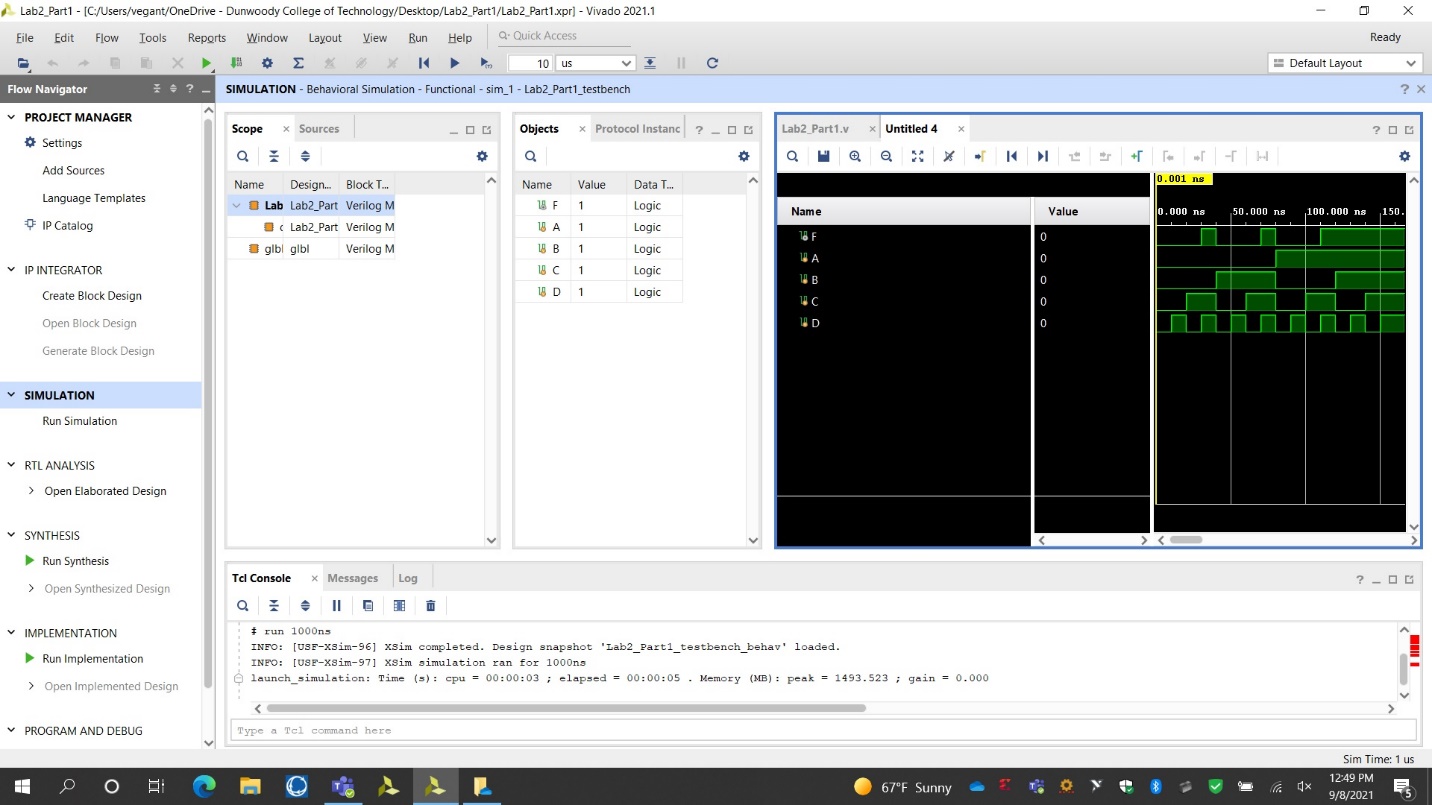
Description automatically generated

1. Write the corresponding test bench to simulate the circuit in Vivado.

Graphical user interface, application

Description automatically generated

1. Simulate for all the input combinations. Take a screenshot showing all input combinations and include it in your lab writeup.



Here’s some scaffolding code to get you started:

**module** and\_or (A, B, C, D, F);

output F;

input A, B, C, D;

// TODO – Finish module definition

// Simple gate instantiation works like this:

// xor(y, a, b);

**endmodule**

**module** and\_or\_testbench;

wire F; // output from module

reg A, B, C, D; // inputs to module

// TODO – Instantiate module under test

initial begin

// TODO – write testbench code

end

**endmodule**

# Part 2: Design + simulate a UDP based Verilog module.

1. Write the Verilog UDP for the following truth table.
2. Write a Verilog module by simply instantiating the created UDP.

Graphical user interface, application

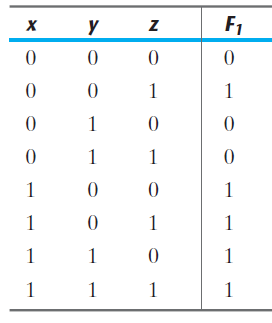
Description automatically generated

1. Write a test bench to simulate the created Verilog module.

Graphical user interface, text, application

Description automatically generated

1. Simulate for all the input combinations. Take a screenshot showing all input combinations and include it in your lab writeup.



Graphical user interface, application, PowerPoint

Description automatically generated

# Part 3: Design + simulate a FullAdder module, with no delays

1. Write down the truth table for the full adder.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | Cout | Sum |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

1. Write the Verilog design module for the full adder, using gate instantiations

Graphical user interface, text, application, email

Description automatically generated

1. Write the corresponding test bench to simulate the circuit in Vivado.

Graphical user interface, text, application, email

Description automatically generated

1. Simulate for all the input combinations. Take a screenshot showing all input combinations and include it in your lab writeup.

Graphical user interface, application, PowerPoint

Description automatically generated

# Part 4: Design + simulate a FullAdderWithDelay module, with #10 delays

1. Write the Verilog design module for the full adder, using gate instantiations, but add “#10” to each instantiation (like **XOR #10 g1(Y, A, B);**) to introduce a 10 ns propagation delay to each gate.

Graphical user interface, text, application, email

Description automatically generated

1. Re-use the same test bench to simulate the circuit in Vivado.

Graphical user interface, text, application, email

Description automatically generated

1. Simulate for all the input combinations. Take a screenshot showing all input combinations and include it in your lab writeup.

Graphical user interface, application

Description automatically generated

# Part 5: Design + simulate a 4-bit Ripple Carry Adder (RCA)

1. This adder will add together two 4-bit binary numbers and a carry-in bit, producing a 4-bit sum and a carry-out bit.
2. How many rows would this truth-table have? \_\_\_\_\_512\_\_\_\_\_\_\_\_\_\_
3. Write the verilog design module for the 4-bit adder, by instantiating the FullAdderWithDelay module written for the previous exercise.

Graphical user interface, text

Description automatically generated

1. Write the corresponding test bench to simulate the circuit in Vivado.

Graphical user interface, text, application, email

Description automatically generated

1. It would be difficult to simulate the circuit for all the input combinations, so you should think carefully about which test cases to run and verify. What sort of “corner cases” should you test? Take a screenshot showing your test bench simulation cases and include it in your lab writeup.
2. What single-bit input change would result in the longest delay to the last output to change? How long would this delay be? Simulate this case and confirm your calculations, including a screenshot of the simulation output. Why do we care about this longest delay through the multi-bit adder?

Graphical user interface, application

Description automatically generated