EENG2131 - Lab 7

Automated Testbenches and GitHub

Antonio Veguilla Hernandez

Student ID: P000166790

This lab builds upon your previously created 4-bit adder module, so finish that lab if you haven’t already.

For this lab, we’ll be exploring the techniques for creating automated testbenches that have been discussed in lecture this week.

# Part 1 – Combinational Circuit Testing – 4-bit Adder

Get the Verilog code and testvectors file from Canvas. Load them into a new project in Vivado (let it copy the files into your new project). Get the testbench running in the simulator. Confirm that the testbench is actually detecting errors by purposefully introducing an incorrect test pair.

# Part 2 – Combinational Circuit Testing – 4-bit Adder – New vectors

Enhance your test vector file to include more tests. Make sure you cover interesting test cases.

# Part 3 – Combinational Circuit Testing – 4-bit Adder – Exhaustive

Enhance your test vector file to include all 29 = 512 test vectors. You should use a programming language, spreadsheet, or some other method to automate the creation of this exhaustive test set.

* **Before you continue, show the instructor your Verilog code and the testbench logs. Explain how you created the exhaustive test set.**
* I choose to use Excel to generate my exhaustive test, 1st I did a normal sum using decimal notation, then for cout to be 1, I use the “If” function to give 1 one only if the sum of x+y+cin >15, otherwise is 0. For the Sum I also use the “If” function as follows: If the sum of x+y+cin > 15, then sum = x+y+cin – 16 (which is cout), otherwise, sum = x+y+cin. The next step was to convert each decimal value into binary, for that I use the “DECTOBIN” function, for x, y, and sum I specified the amount of binary places to be 4, and for cout and cin I specified the binary places to be 1. The final step was to put everything together, to do that I use the function “CONCAT” and select each cell that contains x, y, cin, cout and sum.

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated with medium confidence

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated with medium confidence

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

A picture containing text, indoor, decorated, light

Description automatically generated

**Graphical user interface

Description automatically generated**

# Part 4 – GitHub Submission

Verilog Code exists as plaintext files in your computer, and are often used like normal software source code files. To this end, we can use existing software development and collaboration techniques to enable reliably collaboration on Verilog projects. The majority of the world’s software developers use a version control system (VCS) called GIT to track the history of their code, and to let multiple developers easily collaborate on a shared codebase.

For this part of the project, we want you to learn the basics of git, and how to use the github website to track and share your code.

**My GitHub Link:** [AVeguilla/ENG2131-LAB: EENG2131-Lab repository for all of my lab projects. (github.com)](https://github.com/AVeguilla/ENG2131-LAB)