**EE480 Assignment 2: Multi-Cycle “Gr8B0nd”**

Implementor’s Notes

Tyler Cultice  
Department of Electrical and Computer Engineering  
 University of Kentucky  
 Lexington, KY, USA  
 tacu229@uky.edu

Chace Ritchie  
 Department of Electrical and Computer Engineering  
 University of Kentucky  
 Lexington, KY, USA  
chace.ritchie@uky.edu

Preston Watts  
 Department of Electrical and Computer Engineering  
 University of Kentucky  
Lexington, KY, USA  
 pcwa226@g.uky.edu

ABSTRACT

This project involved our team building a multi-cycle implementation of **Gr8B0nd**. However, for this assignment, we were not required to implement the posit arithmetic aspect of Gr8B0nd. Conference Name:ACM Woodstock conference

**GENRAL APPROACH**

This assignment required us to build a functional processor in Verilog. In order to accomplish this, we needed to make some important decisions up-front.

First, we discussed general design approaches that best suited our style. Initially, we intended to modularize our design with separate ALUmodules for posit arithmetic and integer arithmetic. After discussions, we then decided that this could cause more issues and chose to implement each instruction in a single **processor** module.

Next, we discussed which Assembler to use from the previous project to assemble the instruction set. We decided to use an unaltered version of Dr. Dietz design for the Assembler and found that his design was the best for how we wanted to complete this project.

Next, we defined a significant number of constant variables in the top of our **gr8b0nd.v** file, shown as *`define*. Using this feature in Verilog allowed us to uniquely define the bit-size of fields, values of instruction op-codes, and even short-hand operations that were often used. These are important for the clarity of our code and for the ability to quickly debug issues while testing.

Next, we decided on a **testbench** module that would be used to run our design. We used the testbench provided to us by Dr. Dietz as a starting point and altered it slightly in order to work with our design.

Finally, we began implementing the instructions. This was done in 3 parts. First, we set up the environment for testing our design, namely the logistical aspects of the **processor** module and decoding the instructions. Next, we would design the implementation of one instruction at a time and add it to the processor design. Finally, we would design a sequence of test instructions and test data to make sure the instruction design was coded properly. This was repeated for all instructions.

Found within our submission *tarball* is **gr8b0nd.aik**, our assembler, **gr8b0ndTest.asm**, our file used to test each instruction, **vmem0.txt**, the text file used in *VMEM0* in the CGI interface for Verilog, **vmem1.data**, the data file used in *VMEM1* in the CGI interface for Verilog,and **gr8B0nd.v,** the processor design file.

**ISSUES**

There were no known issues when completing this project. Every instruction was correctly implemented, and we found **100% coverage** when completing testing.

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00