DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF NEBRASKA—LINCOLN

CSCE 230 Semester Project

Team 2

[Nathan Doherty, Molly Lee, Shea Winkler]
11/3/2017
[Version 1]

The project consists of the design of a simple processor (and additional components, as time allows) for a NIOS II Architecture using primarily VHDL.

Revision History

[This table documents the various major changes to this document]

Version	Description of Change(s)	Author(s)	Date
1.0	First Draft of Design-Doc. – Phase II: Datapath Implementation and R-Type functionality	Winkler, Shea; Lee, Molly; Doherty, Nathan	2017/11/03
2.0	Phase III: Memory and instruction implementation	Winkler, Shea; Lee, Molly; Doherty, Nathan	2017/11/13

Contents

Introduction

[Provide a short introduction to this document, the project and the context in which it is being developed]

This document provides a description of Team 2's efforts to create a NIOS II processor architecture using primarily Very High Speed Integrated Circuit Hardware Description Language (VHDL) for design. Implemented will be a substantial subset of a Reduced Instruction Set Computer (RISC) instruction set architecture (ISA). The ISA to be implemented will resemble a subset of the NIOS II architecture and includes some features that are unique to Advanced RISC Machines (ARM). A goal of this project is to reduce the time needed for processes and to efficiently use hardware resources. A basic processor will result from this design and implementation, and there will be opportunities to extend its functionality (….details here coming later….).

1.1 Purpose of this Document

[Describe the purpose of this document; the goal(s) that its content are intended to achieve]

This document should provide a guide for explaining the design decisions Team 2 used in creating its simple processor. It will also elaborate on complicated details and explain operation of the different parts of the processor and any additional parts Team 2 includes.

1.2 Scope of the Project

[Describe the scope of the project, what features and functionality it covers (at a high-level). Describe the problem statement and context in which this project is being developed. Who is it for, what is it for, etc.? You may also explicitly indicate what is *not* within the scope—other potential pieces of the overall project that are not covered by this document]

This project is to be completed in order to successfully complete CSCE 230 at UNL. The processor will be able to do basic functions (simple math, Boolean logic, sequential and conditional analysis). Additional functions may be added (e.g., replace polling with interrupting) as time allows. The processor will have nowhere near the functionality nor complexity of a modern processor found on a computer chip.

In phase I, Team 2 worked to create the control unit for the processor. The control unit is responsible for setting certain flag values that affect the way in which the processor functions. Phase II will implement the control unit to initialize a functioning processor.

In phase II, Team 2 created the Data Path File for the project and compiled a basic processor for the first time. In this phase, it was only necessary to implement Register-type (R-type) instructions. This was the simplest version of the processor that functions. The successive phases have been built from the working product of this phase.

In phase III, Team 2 modified the Data Path and the Control Unit to add new R-type, D-type and B-type instructions, including JR, CMP, LW, SW, ADDI, B and BAL. Team 2 also integrated the memory interface and instruction address generator into the existing processor in this phase. Phase IV will build off Phase III and implement I/O.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

[Define any terms that require a definition—domain specific terms, non-standard terms, or terms that are used in non-standard ways]

R-type instruction – an instruction for the processor that is passed to the processor from a register memory location where the instruction is stored.

D-type instruction – (definition goes here)

B-type instruction – (definition goes here)

1.3.2 Abbreviations & Acronyms

[Define all abbreviations and acronyms used in this document here. This relieves you of the need to define such things within the context of the document itself and provides an easy reference for the reader.]

RISC – Reduced Instruction Set Computer
VHDL - Very High Speed Integrated Circuit Hardware Description Language
ISA – Instruction Set Architecture
ARM – Advanced RISC Machines
NIOS – Netware Input/Output System
R-type – Register Type
D-type – Data Type
B-type – Branch Type
JR – Jump Register
CMP – Compare
LW – Load Word
SW – Store Word
ADDI – Add Immediate
B – Branch
BAL – Branch and Link
I/O – Input/Output

2. Overall Design Description

[Provide an overall summary/description of the project. Identify the major design components, technologies, etc.]

This project is divided into five main parts. Phase I consisted of building a 16 X 16-bit register file, a 16-bit ALU and a control unit. Phase II included integrating the register file, ALU, control unit and other components into a basic data path for executing R-type operations, including Add, Sub, And, Or, and Xor. Phase III included adding additional R-type, D-type, and B-type instructions, as well as implementing the memory interface and the instruction address generator. Phases IV-VII have yet to be completed.

2.1 Alternative Design Options

[If applicable, describe and discuss alternative design options that you considered and discuss why they were not chosen. What advantages and disadvantages do the alternatives provide and what advantage/disadvantages do the chosen design elements provide. Provide some justification for why the chosen elements' advantages/disadvantages outweighed the alternatives]

3. Detailed Component Description

In this section, we will discuss the components used in each phase of the project. (more to come)

3.1 Register File

The register file was designed in Phase I….(more to come)

3.1.1 Component Testing Strategy

[This section will describe your approach to testing this particular component.]

3.2 Control Unit

The control unit was also designed in Phase I, and updated in subsequent phases. In Phase II, most of the R-type instructions were added. In Phase III, the remainder of the R-type instructions, as well as D-type and B-type instructions were added.

3.2.1 Component Testing Strategy

[This section will describe your approach to testing this particular component. Describe any test cases, unit tests, or other testing components or artifacts that you developed for this component. What were the outcomes of the tests? Did the outcomes affect development or force a redesign?]

3.3 Data Path

[This section should detail your Data Path—the components and how they relate to each other. It is highly recommended that you document these elements using tables, UML diagrams, and other visually-informative methods. Figures and tables should have proper captions and be referenced in the main text just like in **Error! Reference source not found.**. You should provide subsections to organize your presentation as applicable.]

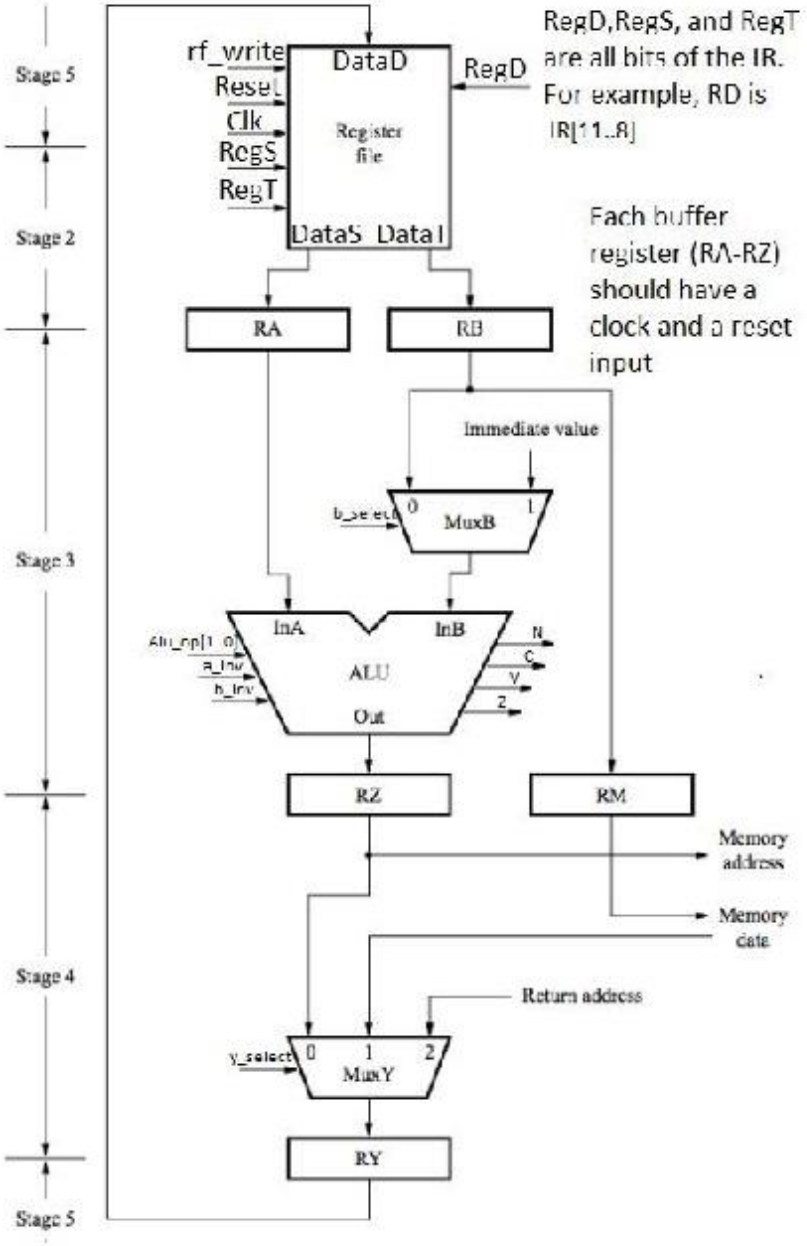

Figure 1: This is the Data Path that was implemented for Phase II of the CSCE 230 Project.

(image from CSCE 230 Project Part II Datapath implementation)

3.3.1 Component Testing Strategy

[This section will describe your approach to testing this particular component. Describe any test cases, unit tests, or other testing components or artifacts that you developed for this component. What were the outcomes of the tests? Did the outcomes affect development or force a redesign?]

Phase I: A (ControlUPDATE.do) testing file was provided and the control unit was tested using the do file to ensure it was working properly.

Phase II: A (PhaseTwo.do) testing file was created by Team 2 to test the Data Path file (and functioning processor) to ensure it is working properly.

Phase III: A (PhaseThree.do) testing file was created by Team 2 to test the new Data Path file after implementing the Memory Interface and Instruction Address Generator. Team 2 also created a (MemoryInitialization.mif) instruction file to provide the instructions to memory.

3.4 Memory Interface

[This section will be used to detail phase V where you design an original data structure and integrate it into your application. In earlier phases this section may be omitted or a short note indicating that details will be provided in a subsequent revision of this document?]

In Phase III, the Memory Interface was implemented into the Data Path.

3.4.1 Component Testing Strategy

[This section will describe your approach to testing this particular component. Describe any test cases, unit tests, or other testing components or artifacts that you developed for this component. What were the outcomes of the tests? Did the outcomes affect development or force a redesign?]

3.5 Instruction Address Generator

[During the development lifecycle, designs and implementations may need to change to respond to new requirements, fix bugs or other issues, or to improve earlier poor or ill-fitted designs. Over the course of this project such changes and refactoring of implementations (to make them more efficient, more convenient, etc.) should be documented in this section. If not applicable, this section may be omitted or kept as a placeholder with a short note indicating that no major changes or refactoring have been made.]

In Phase III, the Instruction Address Generator was implemented into the Data Path.

3.5.1 Component Testing Strategy

[This section will describe your approach to testing this particular component. Describe any test cases, unit tests, or other testing components or artifacts that you developed for this component. What were the outcomes of the tests? Did the outcomes affect development or force a redesign?]

3.6 I/O

[This section will be used to detail phase IV where you … This section will detail the I/O that you designed—how it conformed to the requirements, how it worked, other tools or methods that you designed to assist, how it handles corner cases and the expectations or restrictions that you've placed on the user of the I/O. In earlier phases this section may be omitted or a short note indicating that details will be provided in a subsequent revision of this document.]

I/O will be implemented in Phase IV.

(Insert caption here)

(Insert table here)

3.6.1 Component Testing Strategy

[This section will describe your approach to testing this particular component. Describe any test cases, unit tests, or other testing components or artifacts that you developed for this component. What were the outcomes of the tests? Did the outcomes affect development or force a redesign?]

4. Additional Material

[This is an optional section in which you may place other materials that do not necessarily fit within the organization of the other sections.]

Coming soon…

5. Bibliography

[This section will provide a bibliography of any materials, texts, or other resources that were cited or referenced by the project and/or this document. You *must* consistently use a standard citation style such as APA or MLA (good reference: http://www.cws.illinois.edu/workshop/writers/citation/).]

[1] *CSCE 230 Project Overview*. (2017). Retrieved 25 October, 2017, from https://canvas.unl.edu/courses/19800/files/folder/Project/Overview%20and%20Helpful%20docs?preview=1285596

[2] *CSCE 230 Project Part II Datapath implementation*. (2017). Retrieved October 25, 2017, from https://canvas.unl.edu/courses/19800/files/folder/Project/Part%202?preview=1245925

[3] *CSCE 230 Project Part III Memory and instruction implementation*. (2017). Retrieved November 5, 2017, from https://canvas.unl.edu/courses/19800/files/folder/Project/Part%203?preview=1285586

(Do we need to list all the provided files that were on Canvas?)

Book Example:

[3] Eckel, B. (2006). *Thinking in Java* (4th ed.). Prentice Hall.