FPGA-based RISC Microprocessor and Compiler (Rev. 1.00)

PRCO304 - Final Stage Computing Project

Ben Lancaster 10424877

March 14, 2018

Revision History

Table 1: Document revisions.

Date	Version	Changes
11/03/2018	1	Initial section outline.

Abstract

ben

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern computing and electronics equipment, like function generators, oscilloscopes, and spectrum analysers, use FPGAs to implement their compute intensive logic. These FPGAs are often accompanied by a small, low-cost, microprocessor to supervise and provide interfaces to external peripherals.

The aim of this project is to implement this side-microprocessor into the FPGA to save on BOM costs, PCB space, and power costs, which contribute to higher development and product costs. While savings can be made by the lack of side microprocessor, the product may need a larger FPGA to accommodate the embedded microprocessor. The project will produce a small, soft-core, CPU design and compiler. Although there is no direct client in this project, I believe this project will produce an attractive product for FPGA-based product designers wishing to employ an embedded processor solution.

1.1 Background

1.1.1 Current Implementations

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1.2 Project Overview

This project aims to provide an efficient and cost-saving alternative for board and hardware product designers utilising side-microprocessors by designing, implementing, and demonstrating, a small, portable, FPGA processor core design to be used in-place of the side-microprocessor.

The processor core will implement it's own processor and instruction set architecture and so a compiler and assembler will also be provided so that software code can easily be executed on the processor.

1.2.1 Core Deliverables

1. To improve my knowledge and experience of FPGA development, processor architecture, compilers, and embedded systems engineering.
2. To build a working and operational soft-core processor core capable of performing simple tasks.
3. Implementation of the soft-core processor on real hardware.
4. To provide product designers with an affordable alternative to a side-microprocessor in their FPGA-based products.
5. To provide a technical processor reference guide and specification for the embedded core.

1.2.2 Extended Deliverables

1. To provide embedded products a convenient solution to in-field updating.
2. To provide easy interfacing between the FPGA design and the embedded core.

1.3 Legal and Ethical Considerations

1.3.1 Privacy

The PRCO304 processor will be able to read and write to all data passing through it and control all connected peripherals (such as UARTs, SDRAMs, and SD Cards). The processor does not track or store usage behaviour, instructions and their frequency, memory contents, or timing statistics, or any other usage metric.

1.3.2 Fit for Purpose

The PRCO304 processor is not designed to run general purpose operating systems, such as Linux or embedded RTOS systems. All memory devices attached to the FPGA are fully accessible to the processor core and instructions/programs running through it, meaning that operating systems or secure applications storing private and sensitive information is not protected by modern processor features such as privilege modes and virtual memory sections. The processor lacks common components required to run modern operating systems, such as a memory management unit (MMU) and privilege modes, and so should not be run on the processor.

The PRCO304 processor is not designed to run in high-reliability or safety-critical environments that require established safety standards, such as the UK Defence Standard 00-56 (**?**) and IEC 61508 (**?**).

The PRCO304 processor, by design, should be used as a replacement for a simple micro-controller accompanying a main processing module.

1.3.3 Third-party Libraries

This project uses only 1 external library for the processor core's universal asynchronous receiver-transmitter (UART) module that does not depend on any other libraries. This allows me to guarantee that: the project rights are secure; and application behaviour is well-defined and predictable (no exploits introduced/injected from external libraries). The UART module does feature a large first-in-first-out (FIFO) buffer for temporary storage of in- and out- going messages. This FIFO is internal to the FPGA design and so is protected from external viewing/modification by probing the board in which the core is running on.

The compiler sub-project does not use any external library dependencies, does not record telemetry or usage statistics, and does not require an internet connection to run.

1.3.4 Generated Code

The code generated by the compiler is **not guaranteed** to:

- **produce constant time executable code for expressions.** For example, the compiler output for an *if* statement may implicitly vary depending on the it's, which may have been optimised out, constant-folded, or without-optimisation. This also applies for user code aiming to create reliable and accurate time delay loops; although the processor does not perform optimisations such as instruction caching or branch prediction, access to memory and ALU operations may vary in time.

- **produce code for secure-environments.** The compiler will not randomise, obfuscate, or split-up and spread, output code. Output machine code will be in a predictable format (global variables in low-memory, instruction memory in middle-memory, and stack memory in high-memory) making the binary easily subject to reverse-engineering and modification.

Chapter 2

Project Management

2.1 Time Management

2.2 Version Control

2.3 Method of Approach

2.4 Requirements

Chapter 3

PRCO304 Processor Design

3.1 High Level Design

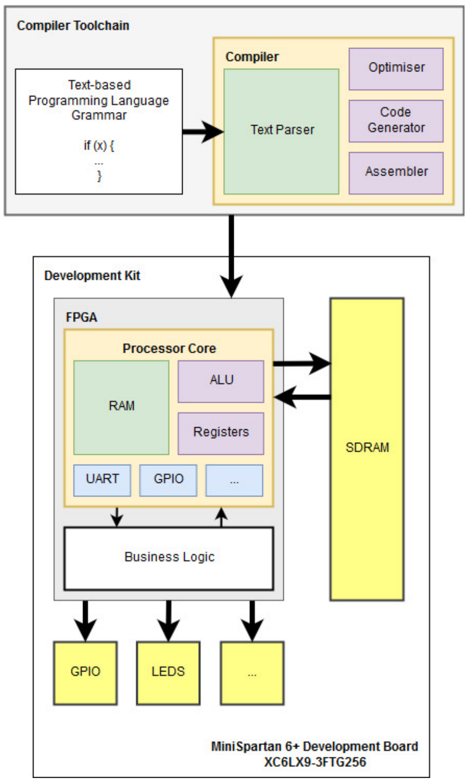

Figure 3.1: test

3.2 Registers

3.3 Pipeline Architecture

3.4 Testing and Verification

Chapter 4

PRCO304 Compiler

4.1 Introduction

4.2 Architecture

4.3 Implementation

4.4 Testing and Verification

Chapter 5

Conclusion

5.1 Project Post-mortem

Chapter 6

Appendices

6.1 Appendix A. PRCO304 Core Reference Guide

6.2 Appendix B. PRCO304 Compiler Reference Guide

6.3 Appendix C. Project Initiation Document