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 **4**

List of Tables **5**

1 Introduction **6**
- 1.1 Background . . . 6
  - 1.1.1 Current Implementations . . . 6
- 1.2 Objectives . . . 6
- 1.3 Legal and Ethical Considerations . . . 7
  - 1.3.1 Privacy . . . 7
  - 1.3.2 Fit for Purpose . . . 7
  - 1.3.3 Third-party Libraries . . . 7
  - 1.3.4 Generated Code . . . 7
- 1.4 Project Management . . . 7
- 1.5 Requirements . . . 7

2 The PRCO304 Processor Design **8**
- 2.1 High Level Design . . . 8
- 2.2 Registers . . . 8
- 2.3 Pipeline Architecture . . . 8
  - 2.3.1 Testing and Verification . . . 8

3 The PRCO304 Compiler **9**
- 3.1 Introduction . . . 9
- 3.2 Architecture . . . 9
- 3.3 Implementation . . . 9
- 3.4 Testing and Verification . . . 9
- 3.5 Conclusion . . . 9
- 3.6 Appendices . . . 9
- 3.7 Appendix A. PRCO Core Reference Guide . . . 9
- 3.8 Appendix B. PRCO Compiler Reference Guide . . . 9
- 3.9 Appendix C. Project Initiation Document . . . 9

List of Figures

2.1 test 8

List of Tables

1 Document revisions. 1

Chapter 1

Introduction

1.1 Background

1.1.1 Current Implementations

1.2 Objectives

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.

1.4 Project Management

1.5 Requirements

Chapter 2

The PRCO304 Processor Design

2.1 High Level Design

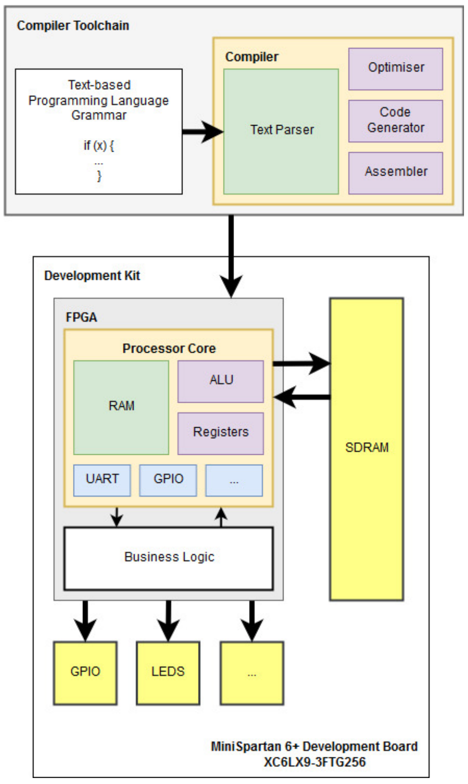

Figure 2.1: test

2.2 Registers

2.3 Pipeline Architecture

2.3.1 Testing and Verification

Chapter 3

The PRCO304 Compiler

3.1 Introduction

3.2 Architecture

3.3 Implementation

3.4 Testing and Verification

3.5 Conclusion

3.6 Appendices

3.7 Appendix A. PRCO Core Reference Guide

3.8 Appendix B. PRCO Compiler Reference Guide

3.9 Appendix C. Project Initiation Document