Creating a RISC-V heterogeneous SoC

Matthew Knight
Department of Computer Science
University of Warwick

26 March 2023

Abstract

This project aims to design and analyse heterogeneous RISC-V SoCs implemented in FPGAs. Many devices have varying performance needs during use, and always aim to minimise power usage. Heterogeneous architectures aim to provide a solution that delivers high performance and high efficiency by combining highly efficient CPU cores (S cores) and highly powerful CPU cores (B cores) in a single processor. When performing background, low priority tasks the S cores are used and the B cores are idle/off, resulting in low power usage. When there are demanding, high priority tasks the B and S cores are used, resulting in higher performance than if just the S cores existed.

The CPU cores implement the RISC-V ISA, an open-standard ISA with support for Linux. There is minimal research into general purpose RISC-V processors implementing a heterogeneous architecture, so the project will perform some novel research. The S and B cores will be designed using the RocketChip library, that generates HDL for a configurable RISC-V SoC.

The designed heterogeneous SoC has been be benchmarked and compared to homogenous designs in terms of size, power draw and performance. Based on these metrics, heterogeneous RISC-V SoCs implemented in FPGAs have limited benefits over homogenous, with a B and S processors providing small decreases in power and area usage for significantly less performance compared to a two B processor, but giving much increased performance for very small power and area increases compared to a two S processor.

Contents

Chapter 1

Introduction

Write around four paragraphs establishing the context and motivating your project.

1.1 Related work

Discuss related work.

1.2 Objectives

One sentence summary of your project. Followed by a short list of concrete objectives:

- Objective 1
- Objective 2
- Objective 3

Chapter 2

Background

Your literature review goes here. **?**] discusses the advantages of functional programming by exploring *e.g.* lists, trees, and noughts and crosses in a functional programming language.

Chapter 3

Design

In this chapter, we describe the overall design of our solution to the problem identified in Chapter 1, building on work described in Chapter 2.

Chapter 4

Implementation

In this chapter, we describe the implementation of the design we described in Chapter 3. You should **not** describe every line of code in your implementation. Instead, you should focus on the interesting aspects of the implementation: that is, the most challenging parts that would not be obvious to an average Computer Scientist. Include diagrams, short code snippets, etc. for illustration.

Chapter 5

Evaluation

Describe the approaches you have used to evaluate that the solution you have designed in Chapter 3 and executed in Chapter 4 actually solves the problem identified in Chapter 1.

While you can discuss unit testing etc. you have carried here a little bit, that is the minimum. You should present data here and discuss that. This might include *e.g.* performance data you have obtained from benchmarks, survey results, or application telemetry / analytics. Tables and graphs displaying this data are good.

Chapter 6

Conclusions

The project is a success. Summarise what you have done and accomplished.

6.1 Future work

Suggest what projects might follow up on this. The suggestions here should **not** be small improvements to what you have done, but more substantial work that can now be done thanks to the work you have done or research questions that have resulted from your work.

Bibliography