Creating a RISC-V heterogeneous CPU architecture

Project specification

Matthew Knight
Department of Computer Science
University of Warwick

1 Introduction

The aim of this project is to design a heterogeneous CPU using the RISC-V open source instruction set. Once designed, the CPU should be implementable on an FPGA and be able to run a Debian Linux distro.

1.1 Definitions

Heterogeneous CPU A CPU that contains 2 different types of core, mismatched in performance and power consumption.

RISC-V ISA An open-source CPU instruction set architecture.

SoC System-on-Chip, a complete computing system within a single circuit board or design.

FPGA A field programmable gate array, an integrated circuit that can be reprogrammed using a hardware description language (HDL).

2 Background

2.1 Motivations for a heterogeneous CPU

Designing a CPU is a complex challenge. They need to be power efficient to reduce the cost of running the system, as well as providing adequate processing power when required. Often one is sacrificed to improve the other: for instance, energy efficiency is often reduced due in larger CPU designs. Due to the increased size and complexity, more power has to be supplied to achieve the same performance as a smaller processor.

A heterogeneous CPU attempts to solve this issue by combining dissimilar core designs; often a powerful core, or p-core, and an efficient core, or e-core. When only light processing is required, the p-core can be effectively shutdown and the e-core will do all processing, resulting in less power used. For heavy processing, the p-core is then used to increase peak performance. Depending on the exact implementation, the p-core can be used either individually or in tandem with the e-core, but both result in greater performance than just the e-core.

This has been used extensively in many mobile devices. ARM released the big.LITTLE in 2011(1), a mobile heterogeneous architecture. The architecture provides the low power usage needed in mobile devices the majority of the time when idle, only running tasks like checking for new messages. It also provides the high performance that can be demanded from phones when used to browse the internet, play mobile games, etc. Most big.LITTLE designs use HMP (heterogeneous multiprocessing) where all cores are available to have processes assigned to them at all times. The alternatives to this are: clustered switching, where either the big cores or the LITTLE cores are in use, and in-kernel switching, where big and LITTLE cores are paired to form a virtual core and only one performs the tasks assigned to the virtual core at any one time.

2.2 The RISC-V ISA

The RISC-V ISA(2) is an open-source instruction set for the design of a CPU, defining the behaviour of the CPU. By defining just behaviour, the actual implementation of the instructions are left to the designer and allow for characteristics like performance, size and more to vary from CPU to CPU. In addition to this, the ISA has extensions and different base options so that a CPU can be tailored to fit an exact use case where only necessary instructions are implemented.

Some of the RISC-V implementations using correct base and extension options are also able to run versions of Linux. The RV64GC configuration is an example of this, with the Debian distribution(3) offering versions that directly support systems using RISC-V processors that meet that minimum configuration. In addition to this, around 95% of the packages available on the Debian OS can be built on a RISC-V machine, allowing such a system to be used for general computing to some extent.

The majority of use cases for RISC-V are currently embedded and highly specialised systems such as compute accelerators for tasks like artificial intelligence, neural networks and image processing. General purpose use of RISC-V is not currently widespread, and design of general purpose RISC-V processors is a developing area.

3 Existing Solutions

3.1 Rocket Chip generator

Rocket Chip(4) is a generator for SoCs, capable of producing many different configurations and designs of RISC-V based SoCs for use in simulation and implementation in FPGAs and VLSI (Very-large-scale integration, creation of an IC of a design). The Rocket Chip generator creates RTL code in Chisel, a hardware description language based on Scala. To create a new SoC, a configuration is written in Scala to describe the features of the SoC, such as cores, cache, tiling (organisation) and peripheral connections.

Cores and their organisation are the main interest of this project, and Rocket Chip provides a large range of options. Rocket Chip can generate two types of RV64GC core: BOOM(5) and Rocket. The BOOM cores contain out-of-order execution pipelines, allowing for more efficient processing of instructions. The Rocket cores are simpler, with in-order pipelines. Both BOOM and Rocket cores have optional FPUs, configurable branch predictors and options for extra ISA extensions. The base Rocket Chip generator allows for combinations of BOOM and Rocket cores to be implemented on a single SoC, including heterogeneous designs such as a BOOM and Rocket core together.

3.2 Other Designs

Most existing heterogeneous solutions appear to comprise of a 'host' processor and an accelerator, where the host processor is constantly running and offloads tasks to the accelerator when needed. This is different to the aim of this project, where the cores are of equal standing and both could be used individually. A brief summary of these works can be seen below.

Title	Description
A RISC-V Heterogeneous SoC for Embedded Devices(6)	A 64-bit RISC-V host core that offloads tasks to a PMCA (Programmable ManyCore Accelerator) made from 8 32-bit RISC-V cores. Implements two types of RISC-V core in a single SoC, but one is used as an accelerator for the other as opposed to them being of equal standing.
Muntjac multicore RV64 processor(7)	A open-source component collection used to build multicore, Linux-capable SoCs designed to be extendable for future development. This uses a single type of RV64 core and uses the Rocket Chip TileLink system.
BlackParrot(8)	A linux-capable multicore RISC-V system, aimed at being a host for accelerator chips. Allows for SMP (Simultaneous Multi-Processing) inside of Linux with the cores 'tiled' such that they can be swapped out and the amount easily changed.
Chipyard(9)	An alternative to Rocket Chip that also provides the functionality to create custom SoCs with RISC-V cores. Chipyard provides more options for adding accelerators, as well as more options for types of RISC-V core and directly allows different types of core to be implemented in one SoC.

4 Objectives

The overall objective is design a heterogeneous RISC-V CPU to be used in an SoC that can run Linux and use both cores to improve peak performance and power efficiency. While this is the overarching goal, it can be broken down into smaller objectives. These objectives have been labelled with each labelled as Must, Should or Could to indicate their importance and expected completion.

1. Design a heterogeneous RISC-V SoC containing 2 dissimilar cores, each capable of executing at least the RV64GC instruction set (Must).
2. Execute instructions on both cores when implemented in an SoC on an FPGA (Must).
3. Have comparable performance to an SoC with only 1 of the larger cores. (Must)
4. Run Linux on the previously designed SoC and connect to it via SSH (Should).
5. Allow processes to execute on both cores inside of Linux - SMP (Could).
6. Intelligently select which core the process will run on, depending on factors such as process priority and resource usage (Could).

5 Risks and Ethical Considerations

Risk	Risk Description	Mitigation	Risk level
FPGA is too small	As the design for the SoC grows larger and more complex, it will require a greater amount of LUTs. This may result in a larger design than the FPGA to be used can implement.	This risk could be mitigated by purchasing a larger FPGA, or by reducing the complexity of the design and objectives.	High
Student is unable to work	Either by illness or other matters, the student is unable to continue work on the project for an extended period.	Mitigating this risk is difficult as illness cannot be predicted, but the likelihood of this occurring is very low.	Low
Concept is flawed/ impossible/ too difficult	As the project progresses, it appears that the main objectives of the project cannot be completed either at all, or in the timeframe provided.	The objectives and scope of the project can be changed/reduced in order to produce a full piece of work by the deadline.	Low

There are no ethical considerations for this project.

6 Project Management and Resources

Weekly meetings will be held with the Project Supervisor to discuss current progress and any issues that may arise during the project. A project work board will be made and kept updated with tasks to complete, completed work and a log of what is currently in progress. This will be shared with the Project Supervisor and assist in tracking the project over it's lifetime, as well as helping in the creation of the Progress Report.

An Artix A7 FPGA prototyping board is the target FPGA to be used during development, and where the final implementation of the SoC will be tested. `git` will be used as version control for the continuous development of the hardware description, along with `GitHub` to store a remote copy of all work completed to ensure the risk of losing progress by hardware malfunction is minimal.

6.1 Timetable

Date	Objective
13/10/22	Finalisation and submission of Project Specification
13/10/22 - 21/10/22	Complete research and testing on the capabilities of Rocket Chip generation
21/10/22 - 20/11/22	Create initial design of the heterogeneous CPU and SoC
20/11/22 - 10/12/22	Finalise design of CPU and SoC, begin testing and attempt Linux boot
20/11/22 - 28/11/22	Write and then submit the combined Project Specification and Progress Report
10/12/22 - 15/02/23	Complete testing and attempted Linux boot, potential research into applications of the designed SoC
15/02/23 - 6/03/23	Write project presentation
6/03/23	Present the project outcomes and demonstrate work
6/03/23 - 15/04/23	Completion of first draft of Final Report
15/04/23 - 2/05/23	Finalisation and submission of the Final Report

References

[1] A. Ltd., "big.little processing." `https://web.archive.org/web/20121022055646/http://www.arm.com/products/processors/technologies/bigLITTLEprocessing.php`, 2011.

[2] R.-V. Foundation, *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2*, 2017.

[3] K. M. Manuel A. Fernandez Montecelo, "Risc-v - debian." `https://wiki.debian.org/RISC-V`, 2021.

[4] K. AsanoviÄ, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, "The rocket chip generator," Tech. Rep. UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016.

[5] J. Zhao, B. Korpan, A. Gonzalez, and K. Asanovic, "Sonicboom: The 3rd generation berkeley out-of-order machine," May 2020.

[6] L. Valente, M. Sinigaglia, Y. Tortorella, D. Rossi, and L. Benini, "A risc-v heterogeneous soc for embedded devices." `https://open-src-soc.org/2022-05/media/posters/4th-RISC-V-Meeting-2022-05-03-Luca-Valente-poster-abstract.pdf`.

[7] X. Guo, D. Bates, R. Mullins, and A. Bradbury, "Muntjac multicore RV64 processor: introduction and microarchitectural guide," Tech. Rep. UCAM-CL-TR-972, University of Cambridge, Computer Laboratory, June 2022.

[8] M. W. D. C. J. S. D. P. G. C. Z. Z. A. S. C. B. V. T. G. A. J. J. M. O. M. B. T. D. Petrisko, F. Gilani, "Blackparrot: An agile open source risc-v multicore for accelerator socs," *IEEE Micro*, 2020.

[9] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanovic, and B. Nikolic, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, 2020.