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

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. This is because more power has to be supplied to achieve the same performance that a smaller processor could achieve while using less power, due to the reduced size.

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, the first 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.

3 Existing Solutions

https://ieeexplore.ieee.org/document/8702538 - Low Energy Heterogeneous Computing with Multiple RISC-V and CGRA Cores. This implements only a single type of RISC-V core which then uses a CGRA (Coarse-grained reconfigurable array) as an accelerator in certain tasks. This is dissimilar to what the project is proposing.

https://www.mdpi.com/1424-8220/21/19/6491 - A Heterogeneous RISC-V Processor for Efficient DNN Application in Smart Sensing System. This is similar to the above, where a RISC-V core is used as the host to an accelerator designed for specific workloads. This is also dissimilar to what the project is proposing.

https://arxiv.org/abs/2206.01901 - Enabling Heterogeneous, Multicore SoC Research with RISC-V and ESP. An extension to the ESP (Embedded Scalable Platform) project to enable the creation of an SoC with multiple RISC-V cores. This does not allow the creation of an SoC with different types of RISC-V core, but does implement some of the functions required for this and so could potentially be extended to allow for dissimilar RISC-V cores implemented in a single SoC.

https://open-src-soc.org/2022-05/media/posters/4th-RISC-V-Meeting-2022-05-03-Luca-Valente-poster-abstract.pdf - 64-bit RISC-V host core running Linux that offloads tasks to a PMCA (Programmable ManyCore Accelerator) made of 8 32-bit RISC-V cores. This is the most similar work I could find to the aim of the project, but differs by having the smaller RISC-V cores as accelerators as opposed to linux-capable, indepedent cores.

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. Multicore RISC-V SoCs that implement SMP in Linux are most similar to my objectives - this is essentially what the aims are, with the extra of the cores being two different types and performance levels.

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.

1. Design a heterogeneous RISC-V SoC containing 2 dissimilar cores, capable of executing at least the RV64I instruction set.
2. Run Linux on the previously designed SoC and connect to it via SSH.
3. Allow processes to execute on both cores inside of Linux.
4. Intelligently select which core the process will run on, depending on factors such as process priority and resource usage.
5. Allow both cores to simultaneously execute processes when required.
6. Have comparable or better performance per watt than an SoC with only 1 of the larger cores.

5 Methods and Resources

https://esp.cs.columbia.edu/ - open source SoC platform, can be used to develop heterogeneous CPU architectures (not RISC-V specific). This allows accelerators and host cores to be implemented together, but does not allow the creation of an SoC with more than 1 type of core, or even multi-core RISC-V SoCs without the extension listed previously. I believe this platform is therefore unsuitable for use in my project due to the large number of changes required and the limited time available for my project.

https://github.com/ucb-bar/chipyard - Chipyard is an open source framework for the development of chisel-based SoC's. It allows the creation of multi-core RISC-V SoCs with various core types, including allowing multiple types of core to be implemented

together on the same SoC. Chipyard designs that meet the minimum specifications for the Linux kernel are also able to run Linux once loaded onto an FPGA, but only certain types of board are supported. It is also unclear whether the heterogeneous designs made using Chipyard would be able to run Linux, as I have been unable to find any examples of heterogeneous RISC-V CPU running Linux.

https://github.com/eugene-tarassov/vivado-risc-v - Scripts and sources to design a various RISC-V SoCs that can run Linux. Allows for various designs of multicore RISC-V systems, featuring Rocket cores (https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html) and BOOM cores (https://github.com/riscv-boom/riscv-boom). The base configuration options given do not support heterogeneous RISC-V SoC, but the project is based on the Rocket Chip generator. This directly supports heterogenous processing at various levels. The Rocket Custom Coprocessor (RoCC) interface is used primarily for application specific coprocessors. Tile and TileLink are generators multiple processor agents, with options for coherence, shared memory and shared networks.

The development of the Linux-capable heterogeneous RISC-V CPU will most likely be done by modifying

6 Risks and Ethical Considerations

Risk	
FPGA is too small	As the design for the SoC grows larger and more complex
Student is unable to work	Either by illness or other
Concept is flawed/impossible/too difficult	As the project progresses, it appears th

There are no ethical considerations for this project, allowing development to be unhindered by any bureaucracy that may have otherwise been needed.

7 Timetable

I'll make a gantt chart and put it here.

References