

RISC-V Processor Design Literature Review and Research Proposal Formulation

Matt Young
s4697249
m.young2@uqconnect.edu.au

November 2023

Contents

1	Introduction	2
2	Methodology	2
3	Initial ideas from me	3
3.1	A triple-redundancy fault tolerant RISC-V processor for safety-critical applications	4
3.2	A VLIW RISC-V processor for embedded DSP applications . .	5
3.3	A runtime reconfigurable microarchitecture for RISC-V processors	7
3.4	Design and implementation of a low-power RISC-V RV32IC processor	8
4	Literature reviews	8
4.1	Reconfigurable computing	8
5	Industry trends	10
6	Finalised research proposals	10

1 Introduction

This document is a follow on from “Project Proposal: Design, Verification and Synthesis of a RISC-V RV32IC Processor”, which we discussed in early October 2023. As we discussed at the time, that project was interesting in a practical sense, but it needed to address a specific *research question* to be viable as a thesis. This document aims to find that research question, both by detailing ideas that I myself have had over the past few years, as well as performing a relatively detailed literature review of recent papers published in the fields of RISC-V processor design and reconfigurable computing.

Before we set out to find a specific research question, we should ask ourselves: What is an acceptable thesis project? My aim is to make this project the best it can possibly be, but the timeline of one year and my relative lack of experience in processor design introduces some constraints. Specifically, we want the project to be complex enough to explore relatively novel or non-traditional CPU design ideas, but we also need to make sure the project can be completed and written up comfortably in one year. Unfortunately, this particular constraint means I have to rule out some interesting ideas such as anything out-of-order, speculative or superscalar. It does also raise into question whether we attempt more complicated things such as multi/many-core designs (even if backed by a relatively simple per-core architecture), and VLIW. Nonetheless, these complicated topics are something I’m looking forward to considering in future research, but for now, I think we are best finding a concept that is simple in theory and can be safely executed in one year (considering other coursework as well). If the project does end up being too simple, we can always add extensions.

2 Methodology

To perform the literature review, we will consider both academia as well as industry. The aim is to determine what the latest research in academia is, and to compare that with what the latest industry trends are.

The specific topics we will look into are:

- RISC-V CPU architectures, preferably simpler ones
- Low-power processor design
- Very-long-instruction-word
- Branch prediction algorithms, preferably simple, low-power ones
- Reconfigurable computing
- Custom instructions vs. Custom hardware, for domain-specific acceleration
- Applications of custom processors and/or RISC-V in cybersecurity

Additionally, we will have a look at what the latest is in the following journals:

- IEEE International Conference on Field-Programmable Technology (FPT)
- Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)
- Reconfigurable Architectures Workshop

Finally, note that we will exclude anything regarding AI and machine learning, purely because this is not a topic I'm interested in researching. That being said, a lot of AI-centric designs can be repurposed to do more interesting DSP work, so I won't completely discard AI papers if they seem to be useful for DSP.

Before we start the literature review, however, I will first briefly detail some ideas I personally have come up with over the last two years or so, which might serve as useful research questions. The goal of the literature review, then, is to see which of these ideas are reasonable, as well as to discover new topics which I might not have considered yet. Once this is done, we will end up with a list of research questions which meet all the relevant criteria for my thesis!

3 Initial ideas from me

The following section details various paper ideas I've had over the last few years. The goal is to lay these out first, and then evaluate if they hold up to scrutiny as actually good ideas by performing the literature review. The ideas

appear in the order of how much I think they are a reasonable project, with the most reasonable first and least reasonable last. Each idea has a suggested thesis title and abstract, along with pros/cons, and some additional notes, and finally a concluding verdict.

Note that these are only the ideas I, personally, have come up with. In the process of performing the literature review, we will undoubtedly come across many more good ideas that I couldn't ever think of. We will catalogue both the ideas I've had, and ideas inspired by the literature review, in a final section.

3.1 A triple-redundancy fault tolerant RISC-V processor for safety-critical applications

Abstract: In recent years, there has been increasing interest in designing RISC-V processors for safety-critical applications. For example, NASA selected SiFive to provide a RISC-V processor for their High-Performance Spaceflight Computing project. One way to design a fault tolerant processor is by using one or more redundant cores operating in lockstep. In this thesis, I design a RISC-V RV32ICM processor with three cores in a triply-redundant lockstep configuration. The design includes attractive features for embedded computing such as a single-cycle 32x32 multiplier and a barrel shifter. The design is synthesised for the Lattice ECP5 FPGA using open-source tools, and the resulting physical implementation is benchmarked using CoreMark. Combined with a rad-hardened CMOS fabrication node, this processor may be an attractive offering for mission-critical computing in defence or space applications, or as an FPGA soft-core for industrial applications.

Pros:

- Interesting research topic, as I am interested in multi-/many-core designs
- Definite real-world applications, probably more so as a softcore rather than an ASIC
- The same processor can effectively be duplicated 3 times, it's just the redundancy and potentially correction logic that is hard

Cons:

- Redundancy logic may be difficult to verify, difficult to prove it’s actually redundant (I imagine a lot of edge cases)
- We will probably encounter significant difficulties with verification, especially with fault-tolerance.
- We may encounter significant time-pressures as designing a RV32 core is complex enough, and the fault tolerant logic adds extra verification work.

Notes:

- We need to *very* carefully scope out this project and define *exactly* what we mean by “fault tolerant”. This will most likely involve designing to be immune to certain types of single event upsets.
- We need to research ways we can reliably introduce single event upsets into both our simulated and real-world system to prove fault-tolerance. This may be extremely challenging.
- A quick search reveals that there are many existing papers on fault tolerant RISC-V design - we could learn a lot from these.

Verdict: I think this is a good project, it’s achievable and has backup options. We should explore research and strongly consider this idea. Specifically, we need to research the why and how of fault tolerant hardware design, see if anyone’s made a fault-tolerant RISC-V processor (and if so, how they did it), and also particularly understand triple-redundancy.

3.2 A VLIW RISC-V processor for embedded DSP applications

Abstract: Digital signal processing (DSP) has become an increasingly important workload for edge computing devices, such as microcontrollers and low-power microprocessors. Yet, maintaining high DSP performance while managing power remains a difficult challenge. VLIW processors have a long history in the DSP segment of the semiconductor industry for their ability to achieve high performance while requiring only simple, low-power hardware. In this thesis, I explore designing a very-long-instruction-word (VLIW) RISC-V processor with specific extensions for fixed-point calculations, ideal for DSP workloads. The processor uses a special VLIW encoding based on

the RISC-V RV32IM ISA, and processes 128-bit words for a maximum of 4 issues per clock. Each of the 4 execution unit supports a single-cycle 32x32 multiplier. I synthesise this design for the Lattice ECP5 FPGA using open-source tools, and measure its power, performance and area (PPA) against competing RISC-V processors on a simple DSP workload.

Pros:

- Very interesting research: combines bits of my two favourite interests, processor design and high-performance computing
- Some potential real-world applications, although not as much as other topics
- Relatively novel research, there are not many papers on RISC-V VLIW encodings
- Strictly speaking, VLIW hardware is not “that” hard (it’s just the writing the decoder and verifying it that’s the hard part - the functional units can be easily duplicated)

Cons:

- Harder to verify: unlike a traditional RV32IC core, the encoding is semi-custom, so we can’t use any existing RISC-V verification suites or instruction fuzzers
- May encounter significant difficulties with the register file, especially since the ECP5 only supports a maximum of dual-port RAM, where we may need quad-port or higher for VLIW
- No RISC-V VLIW compilers (and writing a compiler out of scope for this thesis), so we would have to write an assembler and then program DSP algorithms in assembly - ouch!
- Although VLIW is cool (in my opinion), it’s not as applicable in industry as a more standard RV32 design

Notes:

- We need to clarify exactly what is meant by “fixed point extensions” - what hardware is involved in this? Do we want a CORDICS module?

Verdict: Explore more research. This may be too hard for us right now.

3.3 A runtime reconfigurable microarchitecture for RISC-V processors

Abstract: Modern processors are designed with power, performance and area (PPA) in mind. However, at times, these goals can be conflicting: a low-power design naturally reduces performance, and a high-performance design increases power. In this thesis, I introduce a RISC-V RV32IC processor with a runtime reconfigurable microarchitecture. End-users can configure the processor’s microarchitecture at runtime to dynamically target the goals of high-performance or low-power, with a great deal of flexibility. This is achieved by installing both a “high-performance” and “low-power” microarchitecture on the same chip, and using clock-gating and a re-routing system based on writes to a specific register. The design is synthesised for the Lattice ECP5 FPGA using open-source tools, and real-world power and performance metrics are analysed.

Pros:

- Interesting research topic
- Potential real-world applications and future research options

Cons:

- Unclear if my ideas for the low-power side of things will actually work
- In particular, the multi-cycle design from Harris & Harris may actually draw more power than a pipelined design. This would need some serious analysis.
- Requires essentially implementing two processors: high performance and low power

Notes:

- Instead of implementing two CPUs from scratch, we could borrow existing implementations - this is less fun though
- Following Harris & Harris might naturally allow us to come up with the “low power” and “high performance” designs
- How does this compare to big.LITTLE style designs (i.e. heterogeneous computing)? I discarded the idea of making a big.LITTLE style CPU

because, in my opinion, it doesn't make sense to do for a microcontroller. Does a "reconfigurable microarchitecture" make any sense on a microcontroller either? We may not see much power difference unless you compare something like an OoO design with a in-order design, and of course OoO is *way* out of our league.

Verdict: This is an interesting question, but it may be hard both to implement, and we may not see any real power savings. It also may not make sense to do for a microcontroller style CPU anyway.

3.4 Design and implementation of a low-power RISC-V RV32IC processor

Abstract: With the ever increasing number of battery-powered smart devices in the world today, low-power processor design is becoming a very active field in computer architecture research. Traditionally, architectures like Arm and MIPS have been the main targets of low-power designs - yet, in recent years, the open-source RISC-V architecture has become a driving force in the industry. In this thesis, I design a low-power RISC-V RV32ICM processor by exploring common low-power design methodologies used in industry and research. The resulting design is synthesised for the Lattice ECP5 FPGA using open-source tools, and is extensively benchmarked, both for power and performance. I show how details like pipeline depth and ALU design can create interesting trade-offs between a processor's performance and its power consumption.

4 Literature reviews

This brief literature review aims to determine what the latest research in both reconfigurable computing and processor design is. The reconfigurable computing journals I will be analysing are the IEEE FPT and FCCM, as well as the RAW workshop (not from IEEE). On the processor design side, the journals that will be analysed are:

4.1 Reconfigurable computing

As might be expected, accelerating compute-intensive workloads remains one of the principal applications of reconfigurable computing. However, the most

“trendy” workload to accelerate at the moment appears to be machine learning, specifically deep learning. In my opinion, I believe that there are more devices actually deployed in the real-world that need DSP accelerating,¹ however, undoubtedly the more popular research topic is machine learning. Although DSP remains an incredibly important workload for FPGAs, machine learning is being added to more and more devices, so will become an increasingly important workload as years go on. Already, we’ve seen manufacturers like Xilinx offer AI-tuned SoCs and IP cores.

Other workloads that remain popular for FPGA acceleration include data compression, as shown in [1]. The authors in this particular paper demonstrate performance improvements of up to 25x a fast AMD CPU on a challenging compression algorithm. FPGAs are a great target for data compression due to their hardware parallelism. We could potentially investigate adding data compression instructions to a RISC-V processor, in particular regarding zlib, bzip2 or zstd. The algorithm tested in [1] seems to be too niche, and is complicated to accelerate compared to something like zlib or DEFLATE. We should also consider comparing whether an FPGA-SoC like Zynq, or processor with custom compression functions included, is faster. There is some existing literature on this concept: In [2], researchers from IBM demonstrate their hardware data compression accelerator for the taped-out IBM POWER9 processor. This circuit is designed into the main silicon of the chip, and the entire processor itself is manufactured on GlobalFoundries’ 14nm process node. Their results indicate that the compressor circuit uses only 0.5% chip area, but can speed up zlib workloads 388x compared to a single IBM POWER9 core. This is extremely impressive, as POWER9 is a modern, speculative out-of-order processor [3].

In terms of applying this concept to RISC-V processors, there is exceedingly little research. The only recent paper which does so was published by Google researchers in [4], in which, as part of their benchmarking suite, they introduce the concept of a “compression and decompression processing unit”, or “CDPU”. They develop a “CDPU” RTL generator in Chisel [5], and extend the famous out-of-order BOOM core [6] from Berkeley with their new CDPU. They report that their CDPU accelerator is 10x to 16x faster than a single Xeon core at various compression algorithms, including some new ones like zstd. If we were to implement a data compression accelerator, we would most

¹I have no actual evidence to back this up other than a gut feeling.

certainly not be able to target anything this complicated. We could instead focus on simpler algorithms, such as Huffman coding or even just run-length encoding, to demonstrate the principle that compression can be accelerated in hardware.

On the cybersecurity side of things, some interesting research in [7] shows how reverse engineering FPGA bitstreams is rapidly progressing to be achievable. In this work, the authors were actually able to restore high-level HDL information just from a bitstream. This isn't particularly useful for my research - I just included it out of interest, since I imagine many manufacturers may have distributed unencrypted bitstreams under the impression they were impossible to reverse engineer.

5 Industry trends

6 Finalised research proposals

7 References

- [1] Dongdong Tang et al. “pLPAQ: Accelerating LPAQ Compression on FPGA”. In: *2022 International Conference on Field-Programmable Technology (ICFPT)*. 2022, pp. 1–6. DOI: [10.1109/ICFPT56656.2022.9974593](https://doi.org/10.1109/ICFPT56656.2022.9974593).
- [2] Bulent Abali et al. “Data Compression Accelerator on IBM POWER9 and z15 Processors : Industrial Product”. In: *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 2020, pp. 1–14. DOI: [10.1109/ISCA45697.2020.00012](https://doi.org/10.1109/ISCA45697.2020.00012).
- [3] Satish Kumar Sadasivam et al. “IBM Power9 Processor Architecture”. In: *IEEE Micro* 37 (2017), pp. 40–51. URL: <https://api.semanticscholar.org/CorpusID:31208323>.
- [4] Sagar Karandikar et al. “CDPU: Co-Designing Compression and Decompression Processing Units for Hyperscale Systems”. In: *Proceedings of the 50th Annual International Symposium on Computer Architecture. ISCA '23*. Orlando, FL, USA: Association for Computing Machinery, 2023. ISBN: 9798400700958. DOI: [10.1145/3579371.3589074](https://doi.org/10.1145/3579371.3589074). URL: <https://doi.org/10.1145/3579371.3589074>.

- [5] Jonathan Bachrach et al. “Chisel: Constructing Hardware in a Scala Embedded Language”. In: *Proceedings of the 49th Annual Design Automation Conference*. DAC '12. San Francisco, California: Association for Computing Machinery, 2012, pp. 1216–1225. ISBN: 9781450311991. DOI: [10.1145/2228360.2228584](https://doi.org/10.1145/2228360.2228584). URL: <https://doi.org/10.1145/2228360.2228584>.
- [6] Krste Asanović, David A. Patterson, and Christopher Celio. “The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor”. In: 2015. URL: <https://api.semanticscholar.org/CorpusID:16283995>.
- [7] Reilly McKendrick et al. “Leveraging FPGA Primitives to Improve Word Reconstruction during Netlist Reverse Engineering”. In: *2022 International Conference on Field-Programmable Technology (ICFPT)*. 2022, pp. 1–5. DOI: [10.1109/ICFPT56656.2022.9974401](https://doi.org/10.1109/ICFPT56656.2022.9974401).