Microarchitectural Design-Space Exploration of an In-Order RISC-V Processor in a 22nm CMOS Technology

Max Doblas*[1], Andrew Wright†[2],
Nehir Sonmez*[1], Miquel Moreto*[1],
Arvind†[2]

* *High Performance Domain-Specific Architectures, Barcelona Supercomputing Center*
† *Computer Science & Artificial Intelligence Lab,Massachusetts Institute of Technology*

ABSTRACT

The purpose of this paper is to explore the trade-offs between IPC and maximum clock frequency in an in-order processor design. This work evaluates the impact on the performance and frequency of different pipeline optimizations. We target ASIC implementation using an advanced synthesis tool-flow with modern technology libraries. As a result, we can analyze the processor's critical paths in a representative environment. In this paper, we analyze and modify *Riscy*, an in-order processor, taking into account the consequences of considering the ASIC target for this design. We have achieved a frequency of 1.3GHz and 2.03 CoreMark/MHz in the EEMBC CoreMark.

1 Introduction

Computer architecture research in the academic community has generally focused on reducing the number of clock cycles it takes to execute a program in a processor design. They use instructions-per-cycle (IPC) as a performance metric but generally ignore the clock cycle duration. In the end, the real performance of a processor is measured in terms of instructions-per-second, which is the direct multiplication of the IPC and the clock frequency on a single-core processor. The right balance of these two parameters gives the maximum performance. Sometimes this is the most challenging part since the frequency is highly dependent on the technology chosen and the low-level optimizations applied by proprietary tools [AVLO17]. Also, access to the tools and technology libraries is essential and hard to get. This paper aims to improve an in-order processor called *Riscy*, a 5-stage in-order core developed at MIT, by taking ASIC hardware synthesis into account. Many versions of *Riscy* have been synthesized for FPGAs and have been used in teaching.

[1]E-mail: {max.doblas, nehir.somez, miquel.moreto}@bsc.es
[2]E-mail: {acwright, arvind}@mit.edu

2 Original In-Order Riscy Processor

Riscy in-order is a single issue 5-stage core. It implements the RV64I integer ISA along with the complete implementation of the G and C extensions. It also implements the privileged ISA achieving to boot the Linux kernel successfully. Riscy in-order is written entirely in Bluespec SystemVerilog (BSV), taking advantage of the modularity and atomicity given by the properties of this language. The 5-stage pipeline is shown in Figure 1. The pipeline is composed of: a fetch stage, the decoder stage, the execution stage, the memory stage, and finally, the write-back.

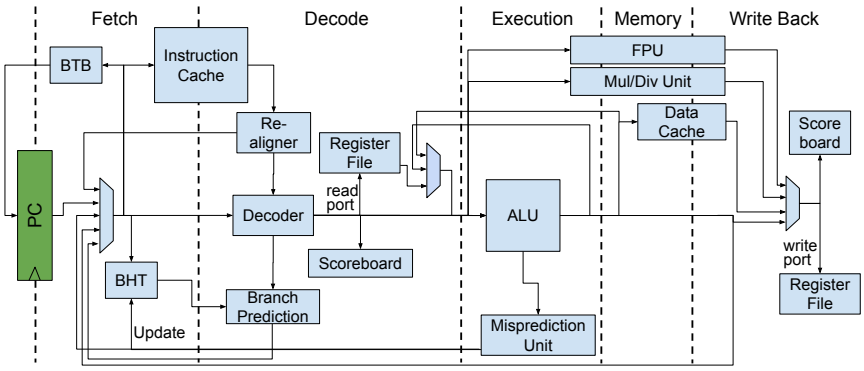

Figure 1: Diagram of the original Riscy v1 pipeline

If we take a more in-depth vision of the core, Riscy implements BHT and BTB branch predictors to improve PC generation. Also, all the forwarding has been implemented to reduce the data dependencies. The caches are 16K Bytes, 4-way associative, VIPT with one cycle of latency. All the memory arrays inside the caches are implemented using SRAMs.

This design has two well-known spots that cause a considerable delay if the FPU and the MULT/DIV units are ignored. These two critical points are located in the two caches. In particular, in the tag comparison on the second cycle of the access. It is due to two factors; the considerable access time of the SRAM itself and the comparison of the SRAM output tags with the physical tag from the TLB. Thus, these two actions, which are performed in series, introduce a huge delay that can be propagated if the cache's output is not registered.

To verify the hypothesis made, we have performed 22nm ASIC syntheses of the Riscy. The results show a critical path that starts on the instruction cache tag array as we expected. The maximum frequency is 667 MHz and 633 MHz with the bypasses implemented.

3 7-stage High Frequency Riscy pipeline

One of the most critical points in the pipeline is the front-end of the design. The SRAM inside the instruction cache has an output delay that consumes a huge part of the clock cycle. It is necessary to add a register at the output of the instruction cache to avoid further propagation of this delay. Also, we decided to break the decoder stage in two to reduce the number of operations done in series in this stage and the control logic of this stage. The new front-end is shown on Figure 2. With the last modifications, all the major critical paths of the Front-End have been removed. Although, now the pipeline has seven stages. The increment of stages has an impact on performance because the PC redirections have a more significant cycle penalty.

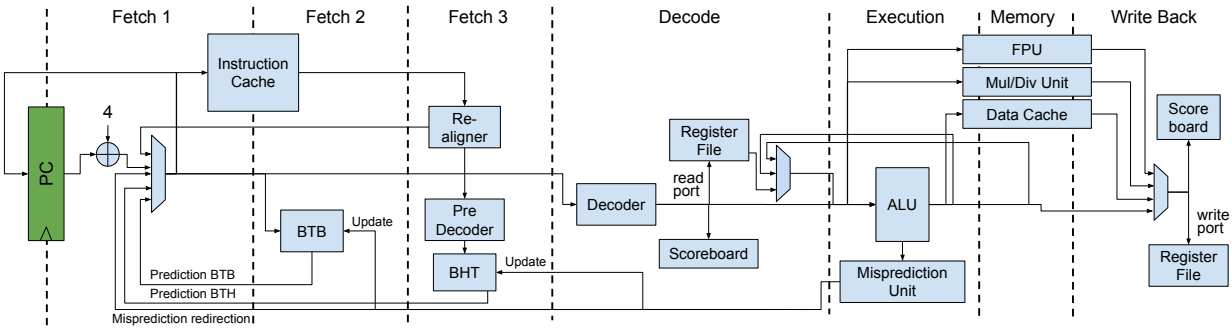

Figure 2: Block diagram of the new Riscy back-end

In a similar way to the instruction cache, it is necessary to avoid all the data paths generated for the data cache. We have decided to register the output of the SRAMs. We modified the data cache by adding a register on the output of each SRAMs to cut the delay propagation. The critical paths should be removed. However, all the memory and atomic instructions have a penalty of one cycle, which significantly impacts performance since each memory or atomic instruction is stalling the pipeline one cycle. With these changes, the critical path was solved. However, we have noticed that the extra latency on the data cache had a significant negative impact on performance. However, we can hide this extra latency if the access to the data cache is started in the execution stage.

4 Evaluation and Results

In this section, we have evaluated the different versions of the design: **Riscy v1:** Original version of the core; **Riscy v1-B:** Riscy v1 with bypasses implemented.**Riscy FE:** it has the new front-end without bypasses implemented; **Riscy FE-B:** Riscy FE with the bypasses implemented; **Riscy FE-B-DC:** the Riscy FE-B with the two-cycle latency data cache; **Riscy v2:** It is the final version of Riscy proposed in this project.

For the evaluation, we used the EEMBC CoreMark. For the clock frequency and critical path analysis, we used an ASIC target synthesis done with the Genus 19.11-s087_1 from Cadence. We are using the 22FDX GlobalFoundries technology node. The standard cell libraries used in all the analysis has 8-tracks, low Vt, a 104 CPP with the typical corner at 25°C, and a power supply of 0.8 V.

4.1 Frequency and IPC Evolution of Riscy

First of all, we comment and analyze the evolution of the performance in the different versions of the Riscy core. Figure 3 shows the results of IPC (Coremark/MHz) and the maximum frequency of each version of the Riscy core. Also, it shows the performance (Coremark, much is better) of each design version.

Riscy v2 improves the maximum clock frequency by 39%, and the IPC by 15% compared to the original Riscy v1, making the Riscy v2 60% faster than Riscy v1. However, directly comparing Riscy v1 with Riscy v2 is not fair because they have different IPC optimizations. To make an apple to apple comparison, we need to compare the Riscy v1-B with Riscy v2 as both implement the bypass optimization. Riscy v2 has a clock frequency that is 47% faster, but with a 9% IPC reduction, resulting in a 34% overall performance improvement. Also, we synthesize Riscy v2 using different combinations of 8-track cells with low and superlow Vt.

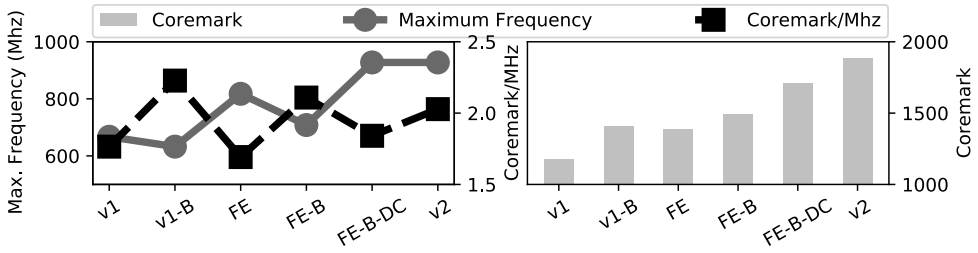

Figure 3: Evolution of the performance of the Riscy core designs. On the left side, there are the IPC and frequency results. On the right side there is the overall performance results.

In the 8-track super-low Vt configuration, Riscy v2 achieves 930 MHz, 1353 MHz, and 1658 MHz in slow, typical, and fast corners.

5 Conclusions

In this project, we have explored the design space of the Riscy core for a modern 22nm ASIC technology target. The final design has a performance improvement of around 60% to the original Riscy design. Compared to the original Riscy design with the same IPC optimizations, around 34% performance improvements are achieved. This improvement is mainly achieved thanks to the improved maximum clock frequency of the design. Riscy v2 has a more mature 7-stage pipeline taking into account the SRAM timing limitations. It scores a 2.03 CoreMark/MHz in the EEMBC CoreMark benchmark, staying very close to the performance achieved by some of the best academics cores as Rocket [Asa16] and Ariane [ZB19]. Moreover, it can achieve more than 1.3 GHz in a typical PVT corner using a technology node of 22nm from GlobalFoundries.

It would also be interesting to implement a superscalar issue in Riscy and apply the same analysis performed in this project. Since the complexity in the core increases considerably, the new critical paths probably would not be related to the caches. Moreover, it will be interesting to apply the same analysis to the RiscyOO out-of-order design [ZWBA18].

References

[Asa16] Krste AsanoviÄ. The rocket chip generator. Technical Report UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016.

[AVLO17] L. AmarÃº, P. Vuillod, J. Luo, and J. Olson. Logic optimization and synthesis: Trends and directions in industry. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 1303–1305, 2017.

[ZB19] F. Zaruba and L. Benini. The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fd-soi technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2629–2640, Nov 2019.

[ZWBA18] S. Zhang, A. Wright, T. Bourgeat, and A. Arvind. Composable building blocks to open up processor design. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 68–81, 2018.