Energy Optimization of the CVA6: an FPGA-based RISC-V Processor Running a Deep Learning Application

Justin Silver, Yara Al-Ahmad, Maxime Kahn, Jean-Baptiste Kammerer

***Abstract*—While participating in a national French student competition, the RISC Makers team has reduced the energy consumption of an open-source RISC-V processor (Ariane) by replacing its cache subsystem. A direct-mapped instruction and a direct-mapped, write-back, write-allocate data cache were developed, with an optimized energy usage. With these new caches, an energy diminution of 10% was achieved for a soft-core FPGA implementation of the CPU running the MNIST Convolutional Neural Network (CNN) deep learning classification program. Execution time also improved by 1.6%, and block RAM (BRAM) logic utilization decreased by 66.6%.**

***Index Terms*—IEEE, RISC-V, FPGA, cache, Ariane, CVA6, low-energy, embedded, Deep Learning, CNN**

I. Introduction

RISC-V is an open-source Instruction Set Architecture (ISA), which enables a wide range of different processor implementations. The OpenHW Group has leveraged this ISA to develop Ariane (CVA6): an application class 6-stage RISC-V CPU capable of booting Linux [1]. In a collaborative effort, Thales Group and others have modified the CVA6 core to be 32-bit compatible and have also developed an FPGA implementation targeting Xilinx's Zybo-Z7 device. With this compact soft-core version of Ariane, Electrical Engineering graduate students all across France were invited to partake in a competition. The objective: reduce the energy consumption of the processor executing the well-known deep learning classification program that attempts to recognize handwritten digits from the MNIST dataset.

This report details the work done by the "RISC Makers" team from the University of Strasbourg. The group's development, design process, and results obtained by the end of the competition are discussed.

Outline

To start, the "baseline" energy and performance values to which the final results will be compared are tabulated in Section II. In Section III, a more detailed analysis of the CVA6's power consumption distribution is carried out, in an effort to determine optimization targets for guiding the subsequent development. After identifying the cache subsystem as an appropriate module for improvement, its performance and the CPU's memory access patterns are explored in Section IV.

J. Silver, Y. Al-Ahmad, and M. Kahn are Electrical Engineering masters students at the University of Strasbourg in France. J. Kammerer, a professor at the same university, supervised the team.

After encountering difficulties in modifying the existing cache modules, a decision was made to develop a new cache subsystem from scratch. The design for a more energy optimized cache is presented in Section V. The final competition results obtained after integrating the new data and instruction caches are detailed in Section VI. Finally, in Section VII, the report concludes with general thoughts regarding the competition, as well as some ideas on how to improve the proposed "RISC Makers" cache.

II. Baseline

In order to properly compare results between teams at the end of the competition, "baseline" performance and energy values were obtained.

In Table I, energy usage is calculated from the number of clock cycles, (to execute the MNIST application) the average power dissipated, and the system clock frequency. Table II summarizes the FPGA logic resource utilization of the CVA6 design.

TABLE I
Baseline performance and energy values

Instructions	**Clock cycles**
1725056	2098749
Clock frequency (MHz)	**Execution time (ms)**
45	46.59
Average power (mW)	**Average energy (mJ)**
307	14.32

TABLE II
Baseline FPGA utilization values

LUTs	**FFs**	**BRAMs**	**DSP blocks**
14675	9291	36	4

Students were encouraged to concentrate their development efforts on the processor's internal core as a means to reduce the absolute energy consumption. However, as per the organizers' rules, modifications to the CVA6 core that resulted in a longer execution time or an FPGA resource utilization increase of 10% (or more, within each logic category) were disallowed.

III. Power distribution

Investigation on how to efficiently reduce the energy consumption of the CVA6 core began with a quantitative analysis,

leveraging energy reports that were generated by Xilinx's FPGA tooling. These reports detailed the dynamic and static power consumption of each instantiated HDL module. Because the competition was only concerned with absolute energy diminution, it seemed judicious to target and optimize the highest power consuming blocks.

Table III details the total power consumption distribution amongst various functional blocks within the CVA6 core[1].

TABLE III
BASELINE DISTRIBUTION OF CVA6'S POWER CONSUMPTION

Cache subsystem	Frontend	Issue stage	Other
42.9%	15.9%	30.2%	11%

The cache subsystem stands out as an obvious optimization target in Table III, with it being responsible for nearly half of the total power consumption. Having thus identified an appropriate aspect of the CVA6 core to optimize, further analysis on the default cache subsystem was carried out.

IV. DEFAULT CACHE ANALYSIS

Quantitative cache analysis

As per the RISC-V ISA specification [2], the CVA6 CPU tracks certain events using hardware registers called "performance counters". The values present within the memory related registers by the end of the program's execution are shown in Table IV.

TABLE IV
MEMORY RELATED PERFORMANCE COUNTERS

Loads	Stores	I$ misses[2]	D$ misses
535829	32243	868	4687

By comparing Table IV with Table I, 33% of the total number of CPU instructions were found to be memory related, with CPU loads making up 31% and CPU stores, 2%. The I$ and D$ hit ratio was calculated to be 99.9% and 99.2% respectively.

Although they pale in comparison with the total number of memory requests, it seemed logical to reduce the baseline number of cache misses in an effort to improve CVA6's energy consumption. Generally speaking, these misses are not only energetically costly due to long main memory transactions, but they also have the potential to result in CPU stalls (pipeline flushes). For a pipelined processor with many stages, stalls will likely result in other unrelated units contributing more to the overall consumption. In addition, reduction of the number of stalls typically reduces the overall program execution time, which could in turn reduce the total energy usage.

It is important to mention that the CVA6 is an out-of-order execution processor. This means that not all cache misses are guaranteed to generate costly pipeline stalls, so long as there is other work to be done within the core. However, with a third of the total number of instructions being memory related, a significant number of pipeline stalls resulting from data cache misses seemed likely.

[1]Values reference solely the instantiated Ariane CVA6 module (63 mW) and thus ignore peripheral and clock generation power consumption.

[2]I$ and D$ signify the CPU's L1 instruction and data cache respectively.

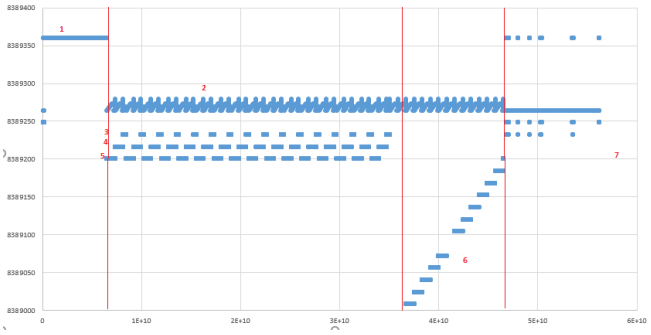

Fig. 1. Address of data memory access (hexadecimal) vs. time (ps)

Memory access patterns

In an effort to better understand the nature of the aforementioned loads, stores, and cache misses, the MNIST program's memory accesses were investigated with the help of CVA6's instruction trace modules. The goal was to spot stride patterns, such as array looping, which would ideally give a clue on how to optimize several key parameters of the cache: size, associativity, and cache block bit width for example.

Fig. 1 illustrates the data memory access locations vs. execution time of the MNIST program. Sequential memory access patterns are observed, which should not come as a surprise, since the present CNN application performs convolution on images using weighted filters, and thus requires the CPU to loop through pixel arrays and perform multiply-accumulate (MAC) operations at various moments.

In light of the discoveries presented in this section, a group decision was made to focus on optimizing the existing cache subsystem[3], starting with the data cache.

Unfortunately, attempts to modify the existing modules were largely unsuccessful. This was in part due to the lack of documentation on CVA6's cache unit and on the interfaces between the CPU and main memory request ports. The default cache's codebase also proved to be somewhat inflexible, with changes resulting in a multitude of compilation errors, or a hanging processor. Several of the failed modifications include:

- Converting the write-through cache to write-back
- Replacing the pseudorandom way replacement strategy with a Pseudo Least Recently Used (PLRU) algorithm
- Removing the speculative all-way parallel data store lookup which occurs in the event of a cache miss

After much frustration, it became apparent that in order to freely implement and test out several energy optimization ideas, the cache subsystem needed to be better understood. It was thus decided that a new cache will be designed from scratch, despite the quickly approaching competition deadline. The objective was to develop a simplified, compact, and energy-aware cache that could be easily parameterized and tailored to the MNIST application.

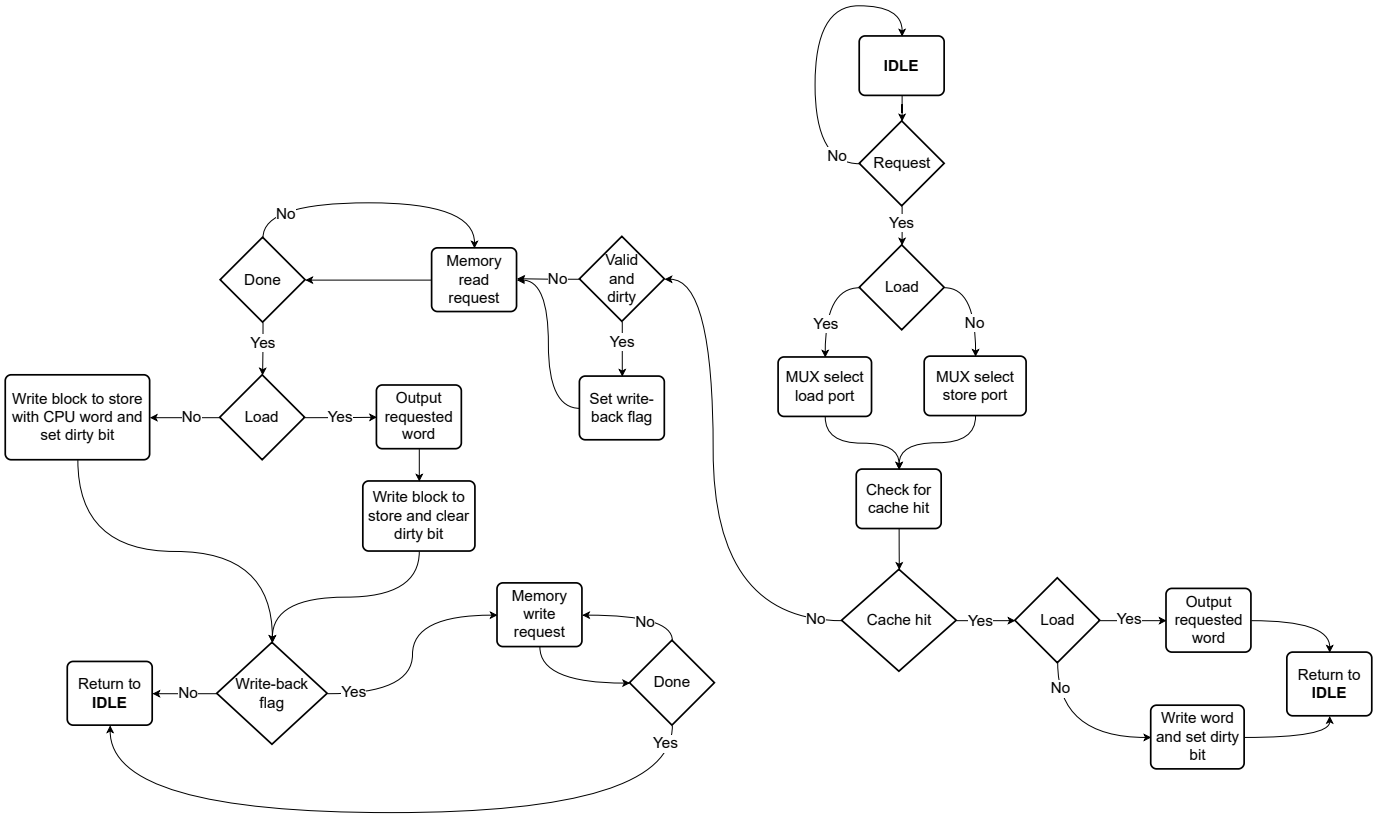

Fig. 2. FSM controller diagram for the RISC Makers data cache

V. New cache design

To aid in the development of a new cache control unit, a finite state machine (FSM) diagram was constructed. Fig. 2 illustrates the simplified control flow of the D$. The I$'s control flow is similar to the D$'s, minus the store/write mechanisms and write-back functionality.

CVA6's default caches are both set associative, with the data cache being write-through. The new proposed caches are both direct-mapped, with the data cache being write-back, write-allocate. In terms of capacity, the default data cache size is 32 KiB, which is also the case for the new data cache. By experimenting with the cache size parameters, it was found that this size resulted in an optimal energy/latency trade-off. However, the new data cache has an increased cache block width size of 256 bits compared to the default cache's 128 bits; once again, the optimal value was obtained through experimentation. The default instruction cache's size is 16 KiB, whereas the new instruction cache is only 2 KiB. Although the total number of instruction cache misses increased by more than 50% due to the size decrease, the total number of clock cycles did not significantly increase.

VI. New cache integration results

The final results obtained after integrating the new cache subsystem are summarized below.

[3]For the interested reader, the architectural details of CVA6's default data cache have been treated by [3].

Final results: energy and performance

Compared to the baseline values, an energy diminution of 10% was achieved. Execution time also improved by 1.6%.

TABLE V
Energy and performance values with new caches

Instructions	**Clock cycles**
1725056	2063787
Clock frequency (MHz)	**Execution time (ms)**
45	45.86
Average power (mW)	**Average energy (mJ)**
281	12.89

Final results: utilization

Compared to the baseline values, the logic resource utilization decreased in each category, apart from the number of DSP blocks used. The largest improvement concerned BRAM utilization, which decreased by 66.6%.

TABLE VI
FPGA utilization values with new caches

LUTs	**FFs**	**BRAMs**	**DSP blocks**
11592	7808	12	4

Final results: power distribution

After integrating the new caches, the power distribution amongst the different functional blocks shifted drastically. Namely, cache subsystem usage decreased by nearly 50% and the issue stage's power consumption increased by 12.6%.

TABLE VII
DISTRIBUTION OF CVA6'S POWER CONSUMPTION WITH NEW CACHES

Cache subsystem	Frontend	Issue stage	Other
21.4%	21.4%	42.8%	14.4%

Final results: new cache subsystem performance

The new caches' performance is summarized below.

TABLE VIII
NEW CACHE PERFORMANCE

Loads	Stores	I$ misses	D$ misses
541842	32243	2147	2513

VII. CONCLUSION

As part of a national French student competition, the RISC Makers team managed to reduce the energy consumption of CVA6, an open-source RISC-V processor, by replacing its cache subsystem with a more simplified and energy-aware solution. While running a deep learning classification program, the processor's absolute energy consumption was reduced by 10%, execution time improved by 1.6%, and the FPGA's BRAM logic utilization decreased by 66.6%.

Achieving these results during the competition was difficult. While there were technical obstacles due to CVA6's limited documentation and the competition's various constraints, communication and relational challenges were also present within the team. It proved quite difficult to juggle final year internships, end of year exams, and personal obligations while simultaneously maintaining an open line of communication within the team. This was exceptionally more challenging with all team members only being able to work together remotely.

The authors are confident that further optimization of the new and proposed "RISC Makers" cache subsystem is a viable pathway towards significantly reducing the overall energy consumption. An initial energy improvement has been achieved during the competition by first developing a configurable and modifiable architecture. With this base subsystem in place, more exotic changes can easily be envisaged in the future. Such examples include cache prefetching techniques [4] and/or non-blocking cache architectures [5].

ACKNOWLEDGMENT

Thanks to Jean-Baptiste Kammerer for being extremely supportive throughout the development process, and for encouraging collaboration and an open line of communication between team members. Thanks to Freddy Anstotz, the head of the second year Electrical Engineering master's program at the University of Strasbourg, for bringing the existence of the competition to the authors' attention and for organizing the RISC Makers team's participation. Thanks to Denis Muller, who helped order necessary material. Finally, thanks to the Thales Group and the CNFM for organizing the event.

REFERENCES

[1] F. Zaruba, S. Member, and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-ready 1.7GHz 64bit RISC-V Core in 22nm FDSOI Technology," Tech. Rep. [Online]. Available: https://github.com/pulp-platform/ariane

[2] A. Waterman and K. Asanovic, "The RISC-V Instruction Set Manual Volume I: Unprivileged ISA - Document Version 20191213," *RISC-V Foundation*, vol. I, 2019. [Online]. Available: https://riscv.org/technical/specifications/

[3] V. Martinoli, Y. Teglia, B. Abdellah, and R. Leveugle, "CVA6's Data cache: Structure and Behavior," Tech. Rep. [Online]. Available: https://arxiv.org/abs/2202.03749

[4] N. P. Jouppi, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers," Tech. Rep.

[5] D. Kroft, "LOCKUP-FREE INSTRUCTION FETCH/PREFETCH CACHE ORGANIZATION," Tech. Rep.