EMERGING TECH CONFERENCE – Edge Intelligence

Volume 02, 2023, Page 158 – 165

Proceedings of Emerging Tech Conference:
Edge Intelligence 2023

Design and implementation of a compact RISC-V based Machine Learning accelerator on Low End FPGA

Manolis Galetakis[1], Stavros Kalapothas[2], Georgios Flamis[2], Paris Kitsos[2], Fotis Plessas[1]
[1] *University of Thessaly, Volos, Greece*
[2] *University of the Peloponnese, Patras, Greece*
egaletakis@uth.gr, s.kalapothas@go.uop.gr, g.flamis@go.uop.gr,
kitsos@uop.gr, fplessas@e-ce.uth.gr

Abstract

In recent years, the use of Machine Learning (ML) algorithms on edge devices for common applications such as computer vision, speech recognition etc. is increasing rapidly. With the evolution and the complexity of the latest ML models, inference has become more computationally expensive and therefore, deployment to resource-constraint edge devices is a challenging task. A common technique to address that challenge and increase performance and efficiency, is to offload compute intensive functions from software and run them onto dedicated hardware instead. In this paper we present the complete flow of software-hardware co-design for a lightweight object detection ML model, first time implemented on a RISC-V soft processor. Then we describe software optimizations for the model and a compact hardware accelerator design. We deploy and profile, in terms of cycle count, resource utilization and power consumption of our design on a Xilinx Artix7 35T low-end Field Programmable Gate Array (FPGA) device. Our RISC-V based compact accelerator achieves almost 3.7x inference speedup @ 180 mW power consumption and consumes 5.5K slice LUTs (26.41%) – 4.3K slice registers (10.19%). Therefore, our implementation facilitates object detection for a variety of non-speed demanding embedded applications, at competent speed on small– entry-level FPGA devices.

1 Introduction

Nowadays, accelerator implementations utilizing RISC-V features have emerged rapidly. In our recent article [1] a survey on RISC-V based ML accelerators were conducted emphasizing at the various hardware cores in conjunction with the software frameworks and stacks available to implement FPGA ML inference.

Edge Impulse FOMO (Faster Objects, More Objects) [2], a novel machine learning algorithm which enables object detection on devices with limited hardware resources. FOMO, according to the author, compared to MobileNet SSD or YOLOv5, it can measure and locate objects in an image, as well as monitor numerous objects in real time while using up to 30 times less computing power and memory. The network was trained with a small subset, only two hundred images, of the PASCAL Visual Object Classes 2005 [3] dataset to perform object detection and has achieved slightly over 81% accuracy. As of this writing the authors cannot find similar implementations of FOMO neural network on a RISC-V based embedded (not SoC-based) FPGA.

When it comes to the deployment of ML accelerators, FPGA devices have emerged as the preferred choice. This is due to its low cost, high degree of reconfigurability, and most importantly, its quick time-to-market. With regard to SoC and software design and implementation, the CFU-Playground [4] is utilized, a full-stack open-source framework that aims to enable rapid prototype development of ML accelerators for FPGA systems. The selection was based on the tool simplicity of design and evaluate FPGA-based hardware accelerators combined with a soft processor, to increase the performance of ML model computation.

In this paper a complete accelerator design is presented using the aforementioned tools. The full design flow is described starting from the network selection and training, following the accelerator design methodology and deployment in a FPGA development platform.

The rest of the paper is organized as follows. In Section 2 a list of RISC-V hardware accelerators is presented. The overall proposed accelerator design architecture is described in Section 3. In Section 4 the experimental results are depicted. Finally, the conclusions and possible improvements are enclosed in Section 5.

2 Related Work

In this section, a diverse array of RISC-V-based hardware accelerator designs, which has been reported in the literature, is presented. The list can be categorized into two types, accelerators implemented as FPGA-based softcores and accelerators available as Application Specific Integrated Circuit (ASIC) designs.

The advantages of using FPGAs for accelerating AI workloads and more specifically Convolutional Neural Networks (CNN), have been extensively studied and reported in [5]–[8] in recent years, such the ones derived from the spatial computing and low-latency nature of FPGAs. RISC-V architecture has also been exploited due to its inherent extensibility with the use of custom ISA extensions, such as vector, or DSP extensions, which can be imported to increase the computation performance in an extended domain of applications. In addition, there are software frameworks that can generate RISC-V synthesizable accelerated cores featuring, for instance, the RV32V, or any other custom ISA extension.

In [9], the authors have demonstrated a 5-stage RV32I barrel processor for matrix-matrix product operations with little less than 7K LUTs utilization in FPGA, in its minimum 8-thread configuration. Also, in [10], a RISC-V software processor based on VexRiscv core and synthesized on a FPGA with 9K LUTs utilization for multiplication operations is presented. In [11], a cluster of RISC-V cores as an accelerator overlay based on the Parallel Ultra Low Power (PULP) platform is implemented on a FPGA with more than 20K LUTs utilization in most configurations. Moreover in [12], a scalable vector processor in RTL synthesizable core files is introduced, that uses novel techniques for high-performance and cost-effective implementation.

Arnold in [13], is a fully programmable RISC-V microcontroller unit (MCU) fabricated in 22nm as a System-on-Chip (SoC) for better performance and energy efficiency. GAP-8 in [14], is a ultra-low power RISC-V SoC for Edge-AI applications with a focus on battery operated scenarios. The Kendryte K210 SoC [15], is a low-cost ASIC with a dual-core 64-bit capable RISC-V that implements the RV64GC ISA and has demonstrated 0.25 TOPS performance on a fixed-point precision CNN model at 400MHz. Lastly, another fabricated and commercially available hardware accelerator is the MAX78000 MCU [16], which incorporates a 32-bit RISC-V coprocessor. The main role of the RISC-V is to act as a smart direct memory access (DMA) and move data between the CNN and on-chip memory.

3 Proposed Accelerator Design

To profile and analyze the FOMO model "The Step-by-Step Guide to Building an ML Accelerator" from CFU-Playground documentation [17], has been followed. The final trained model has a total of 27 layers, with the majority being convolutional and depthwise convolutional, fifteen and six respectively, and the remaining layers being added, pad and softmax. At first, we compiled the model and run inference (object detection) on 96 by 96 pixel grayscale images. As expected, the majority of time 95.4% is consumed in the two main convolutional layers. The convolution function is presented in Algorithm 1 in the form of pseudocode. The inner loops, lines four to thirteen hereinafter called for simplicity "the inner loop", also described as fetch data from memory – calculate quantized dot product – store result back in memory. On the other hand, the loops in lines one to three hereinafter also called for simplicity "the outer loop" simply takes care that the correct data are fetched from the memory for the algorithm to operate correctly. To further profile this inner loop the RISC-V performance counters, are utilized. Our observations at the cycle count of the outer loop compared to the total spend in the convolution function depict that almost 90% of the time is spent in the inner loop. In conclusion the portion of code, i.e., the inner loop, is the best candidate for optimization and therefore we should focus on it. The same conclusion derives from the analysis of the code for depthwise convolution layer. The next two paragraphs quote a short description of both software and hardware optimization that were applied.

3.1. Software Optimizations

Prior any hardware accelerator has been developed, the ML model can be accelerated by further parameterizing in software.

After inspection of the two convolution functions, it is observed that the possible convolution parameters (padding, filter width, height etc.) are constant. However, these parameters can be replaced, with literal values, directly in the source code.

Algorithm 1 Original 2D Software Convolution in CFU Playground

Parameter: X, Y, Z = *output matrix dimensions*
D, C = *weights matrix dimensions*
Input: I = *image data matrix*, W = *weights matrix*, B = *bias*,
S = *scaling vector*, O = *offsets vectors*
Output: R = *dot product output*

```
1  for x = 0 to X do
2    for y = 0 to Y do
3      for z = 0 to Z do
4        for h = 0 to D do
5          for w = 0 to D do
6            for c = 0 to C do
7              R^z_{x,y} += W^{z,c}_{h,w} * (I^c_{x+h,y+w} + O_i)
8            end for
9          end for
10       end for
11       R^z_{x,y} += B^z
12       R^z_{x,y} *= S^{z^{-1}}
13       R^z_{x,y} += O^z_r
14     end for
15   end for
16 end for
```

Algorithm 1: Convolution Software Function

There are some variables' value checking inside the code that is predetermined for our model for convolution or depth wise convolution layers. Some examples of such checks include the existence of a bias value, or the input data is inside image that both are always true. These checks can also be stripped away. It is also feasible to eliminate loops controlled by parameters with fixed values, such as batches or the width and height of filters, without impacting the outcome. With respect to Algorithm 1 for convolutional layer, in the last loop in lines six to eight, the value of filter depth is always a multiple of four, except the first layer. Taking into consideration that the greatest common factor of these layers is eight and the values of filter and input data are continuous in memory, we can unroll this loop by a factor of eight. The effect of these optimizations is summarized in Fig. 2 for convolution and depthwise convolution (orange column) showing speed up improvement by a factor of 2.3x.

3.2. Hardware Optimizations

Following the software optimizations applied previously, when the last loop unrolled by a factor of eight this equals to eight discrete multiply and accumulate operations in each iteration. In the original code, the algorithm operates on data and filter values stored in byte arrays. But our soft core uses 32- bit wide registers which are contiguous in memory. This means that we can perform single instruction multiple data (SIMD) multiply and accumulate calculations thus having the same results with two discrete operations per iteration. Focusing more on the convolution function from the perspective of computation, the result from the accumulation is multiplied by a quantized multiplier, after bias addition and finally limited between a constant min and max value. All this functionality is realized in hardware with the final accelerator architecture shown in Fig. 1.

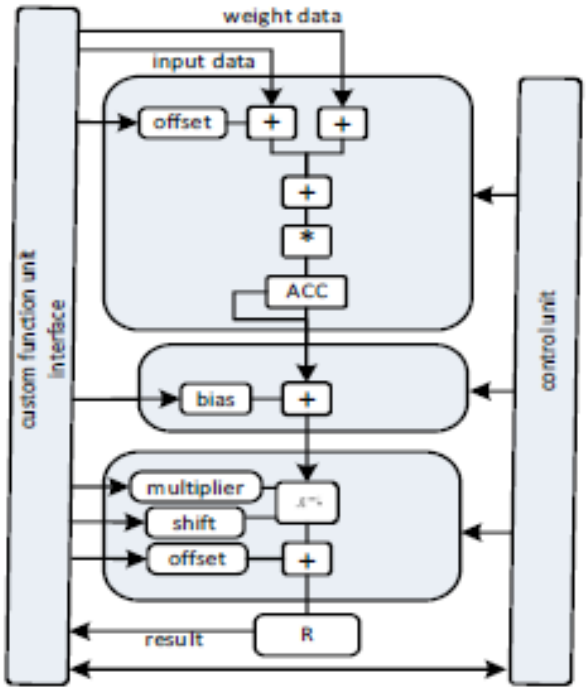

Figure 1: Accelerator Architecture

The accelerator is connected to the soft processor through the custom function unit (CFU) interface using

RISC-V custom command a process thoroughly described in the documentation of CFU Playground Framework. Three units namely mac, bias and quantizer are responsible to implement the SIMD, bias addition and quantizer multiplication – limitation respectively. A simple control unit is used to realize the CFU protocol communication and synchronize units while a simple register is utilized to buffer results. To integrate the hardware accelerator functionality the following changes were made to original convolution function, in Algorithm 1:

1. Two initialization commands were implemented and used to transfer configuration parameters (offset, bias, min and max values) into the appropriate accelerator unit.
2. Multi-byte SIMD accumulation is done using two sequential accumulations commands whose operands are four-byte input and filter values.
3. Bias addition and quantization is realized through a simple read command which also transfers the result back to CPU.

Fig. 2 summarizes the impact of the hardware accelerator showing speed up improvement by a factor of 3.7x.

4 Experimental Results

Our experiments targeted the Arty A7 development board from Digilent [18]. This is primarily because of two reasons a) the availability of the specific board on the lab and b) the ability to measure the power consumption. The Arty board includes circuitry for monitoring the main (5V) and part (only 0.95V) of the FPGA core voltage supplies as well as the current consumed from those supplies. The first represents the total system (board level) power consumption, and the second represents the power consumption of the digital logic and block ram of our design in the FPGA core.

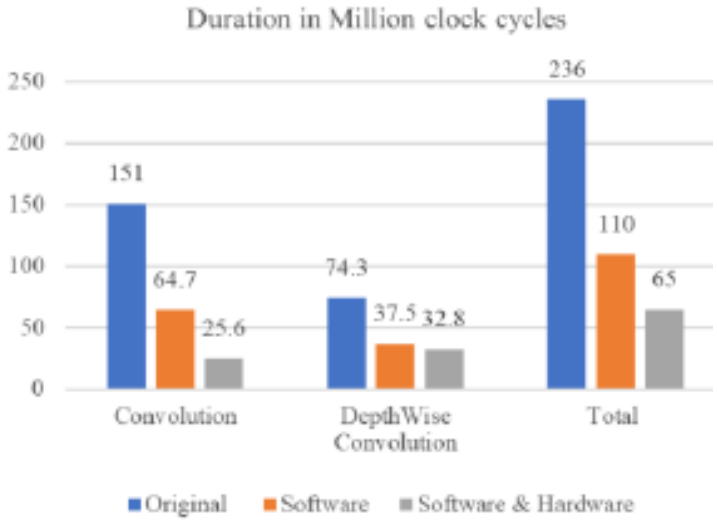

Figure 2: Speed Optimization

The experiment configuration includes the VexRiscv full variant with 8KB data and 8KB instruction cache, complemented with the compact accelerator presented in Section 3 and we measure speed (cycle count), area utilization and power consumption. The results are presented in Fig. 2, Fig. 3 and Fig. 4 respectively. When deemed necessary three discrete experiment configurations were executed for a) original code, b)

software only optimizations and c) software and hardware optimizations.

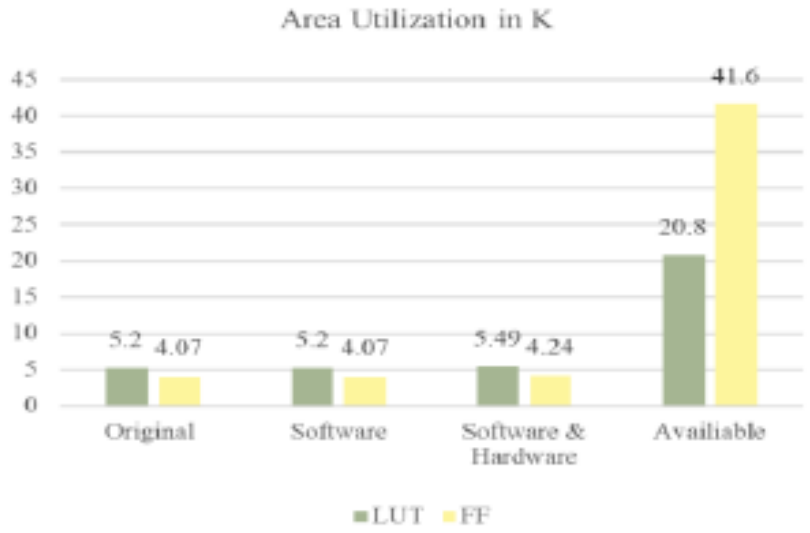

Figure 3: Area Utilization

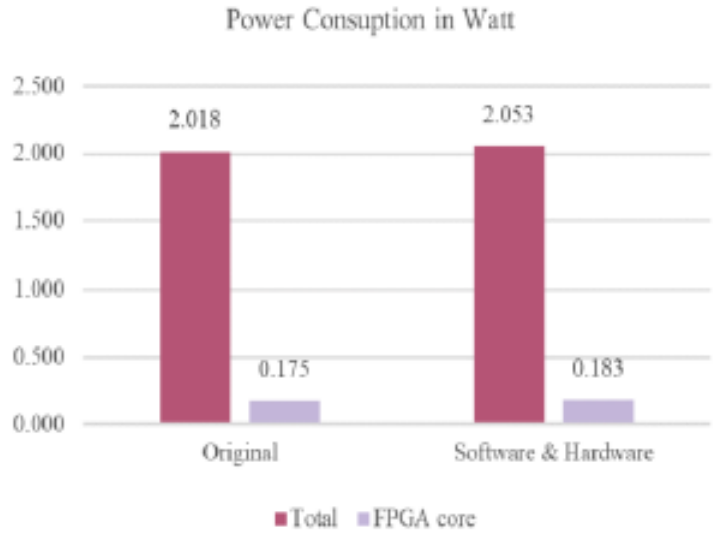

Figure 4: Power Consuption

The results show overall performance improvement by a factor of 3.7x in terms of execution speed. The time for one frame inference taking into consideration that our clock speed is 100MHz reduced from 2.5 seconds to 0.65 seconds. The design consumes 5.5K slice LUTs (26.41%) – 4.3K slice registers (10.19%) thus it can fit in most of the entry level FPGAs of all major manufactures (Intel, Xilinx, Lattice, etc.). The power consumption of the FPGA chip is about 180 mW, which corresponds to the maximum value measured over one hundred images dataset. This result is acceptable if we consider that, according to a thorough study on tinyML conducted on diverse applications and systems in [19], the typical power consumption is between 25mW and 200mW.

5 Conclusions

In this paper we presented a complete hardware-software RISC-V based ML accelerator design using open-source tools. We described a full design workflow, starting from the network selection and training of the model, carried on with the accelerator design methodology and deployment in a FPGA development platform. Our implementation achieved an overall speed increase by a factor of 3.7x compared to running inference natively without the software hardware optimizations available in CFU illustrating that is feasible to achieve object detection at decent speed on low-resource and entry-level FPGA devices.

This is an exploratory work which is presented as an attempt to exploit the CFU framework on resource-constrained devices and therefore, further work is required to proliferate results. Finally, from the power consumption perspective, we believe that by eliminating the necessity of external memory, or replacing it with internal Block-RAM, will have a huge positive impact.

The source code is available in the form of a CFU Playground project at: https://github.com/ECSAlab/fomo-object-detection.

6 References

[1] *A Survey on RISC-V-Based Machine Learning Ecosystem.* **S. Kalapothas, M. Galetakis, G. Flamis, F. Plessas, and P. Kitsos.** 2, s.l. : Information, 2023, Vol. 14.

[2] FOMO. *EDGE IMPULSE.* [Online] https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/object-detection/fomo-object-detection-for-constrained-devices.

[3] The PASCAL Visual Object Classes. [Online] http://host.robots.ox.ac.uk/pascal/VOC.

[4] *CFU Playground: Full-Stack Open-Source Framework for Tiny Machine Learning (tinyML) Acceleration on FPGAs.* Prakash, S., et al. s.l. : IEEE ISPASS, 2023.

[5] *Best Practices for the Deployment of Edge Inference: The Conclusions to Start Designing.* **Flamis, G., Kalapothas, S. and Kitsos, P.** 16, s.l. : Electronics, 2021, Vol. 10.

[6] *A systematic literature review on hardware implementation of artificial intelligence algorithms.* **Talib, M.A., Majzoub, S., Nasir, Q**. s.l. : Supercomput, 2021.

[7] *Domain-specific hardware accelerators.* **W.J. Dally, Y. Turakhia, and S. Han.** 7, s.l. : Communications of the ACM, 2020, Vol. 63.

[8] *FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review.* **A. Shawahna, S. M. Sait and A. El-Maleh.** s.l. : IEEE Access, 2019, Vol. 7.

[9] *RISC-V barrel processor for deep neural network acceleration.* **M. AskariHemmat, O. Bilaniuk, S. Wagner, Y. Savaria, and J. P. David.** s.l. : In 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021.

[10] *Accelerated RISC-V for post-quantum SIKE.* E**lkhatib, Rami, B. Koziel, R. Azarderakhsh, and M. M. Kermani.** 6, s.l. : IEEE Transactions on Circuits and Systems, Vol. 69.

[11] *A RISC-V-based FPGA Overlay to Simplify Embedded Accelerator Deployment.* **G. Bellocchi, A. Capotondi, F. Conti and A. Marongiu.** s.l. : 24th Euromicro Conference on Digital System Design (DSD), 2021.

[12] *RISC-V2: A Scalable RISC-V Vector Processor.* **K. Patsidis, C. Nicopoulos, G. C. Sirakoulis and G. Dimitrakopoulos.** s.l. : IEEE International Symposium on Circuits and Systems (ISCAS), 2020.

[13] *Arnold: An eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End Nodes.* **Pasquale Davide Schiavone, Davide Rossi, Alfio Di Mauro, Frank Gurkaynak, Timothy Saxe, Mao Wang, Ket Chong Yap, Luca Benini.** 4, s.l. : IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, Vol. 29.

[14] *GAP-8: A RISC-V SoC for AI at the Edge of the IoT.* **Flamand, E.** s.l. : IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2018.

[15] Kendryte home page. [Online] https://www.canaan-creative.com/product/kendryteai.

[16] *Benchmarking the MAX78000 artificial intelligence microcontroller for deep learning applications.* M. Clay, C. Grecos, M. Shirvaikar, and B. Richey. s.l. : Real-Time Image Processing and Deep Learning, 2022, Vol. 12102.

[17] CFU-Playground documentation. [Online] https://cfu-playground.readthedocs.io/en/latest/index.html.

[18] Artix-7. [Online] digilent. https://digilent.com/reference/programmable-logic/arty-a7/start.

[19] *A Comprehensive Survey on TinyML.* **Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki and A. S. Hafid,.** s.l. : IEEE Access, 2023, Vol. 11.