Proceedings of the 5th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/121/2025.19489

Research on Custom Algorithms and Hardware Accelerators Based on RISC-V Vector Extensions and Image Processing

Hong Chang[1,a], Kunlin Ye[2,b,*]

[1]School of Architecture, Nanjing Tech University, Nanjing, 210000, China
[2]School of Physics & Electronics, Hunan University, Changsha City of Hunan Province, 410082, China
a. 202021114106@njtech.edu.cn, b. ykl220213@hnu.edu.cn
**corresponding author*

Abstract: RISC-V is an open, modular instruction set architecture (ISA) that has gained increasing adoption in fields such as image processing, machine vision, and deep learning inference due to its scalability and flexibility. This paper provides a comprehensive review of the latest research on RISC-V, with a focus on its vector extensions, custom instruction set optimizations, and the design and implementation of related hardware accelerators. The study employs a hardware-software co-optimization approach, featuring the design of a lightweight convolutional neural network (CSANet) and an artificial intelligence image signal processing (AI-ISP) accelerator to enhance image construction and inference efficiency. The experimental methods include performance evaluations based on FPGA hardware platforms, with a quantitative analysis of computational bottlenecks in the CSANet inference process, optimizing convolution operations and data flow handling. This research provides comprehensive technical support for the application of RISC-V in high-efficiency image processing and deep learning inference, especially in low-power and edge-computing scenarios.

Keywords: Custom Algorithms, Hardware Accelerators, RISC-V Vector Extensions.

1. Introduction

With the rapid development of artificial intelligence (AI) technologies and embedded systems, tasks such as image processing, machine vision, and deep learning are placing higher demands on hardware computational performance. The traditional hardware architectures, constrained by trade-offs between power consumption, performance, and flexibility, face limitations in their application to low-power edge devices. As a result, research on hardware architectures optimized for specific applications has become a key focus. RISC-V, an open and modular instruction set architecture (ISA), is an ideal choice for achieving efficient computation in low-power embedded devices due to its high customizability and scalability. RISC-V not only supports basic general-purpose processing but also allows for the optimization of computational performance for specific tasks through custom instructions and hardware acceleration extensions.

This study focuses on hardware accelerators for image processing and deep learning inference based on the RISC-V architecture, with a particular emphasis on their applications in convolutional neural networks (CNNs), spiking neural networks (SNNs), and image signal processing (ISP). The

DOI: 10.54254/2755-2721/121/2025.19489

research explores the design and implementation of RISC-V vector extensions and custom instruction set optimizations, aiming to enhance computational efficiency and energy performance in edge computing environments.

This paper provides an overview and comparison of various research achievements in the co-optimization of RISC-V hardware and software, offering efficient solutions for image processing and deep learning inference on low-power, resource-constrained edge devices. The findings not only hold theoretical significance but also offer valuable technical guidance for practical applications.

2. Application of RISC-V in image signal processing (ISP)

Wu and his team proposed an artificial intelligence image signal processing (AI-ISP) accelerator based on RISC-V instruction set architecture (ISA) extensions, aimed at addressing the challenge of balancing performance and flexibility in deep learning inference for edge devices. The study centers around the lightweight convolutional neural network, CSANet, optimizing computational bottlenecks through the design of specialized RISC-V custom instructions and implementing it on an FPGA hardware platform. CSANet focuses on image construction and enhances image detail quality using its Double Attention Module (DAM), which leverages both channel and spatial attention mechanisms. Unlike traditional ISP pipelines that rely on multi-stage algorithms, deep learning models demonstrate the potential to generate high-quality images. In terms of model adaptation, the researchers quantized CSANet to INT8 format, making its parameters more suitable for resource-constrained edge device inference scenarios [1].

The architectural design is based on the RISC-V VexRiscv processor and integrates the CFU-Playground development framework. The researchers proposed a hardware accelerator capable of supporting conventional convolution, transposed convolution, depthwise convolution, and pointwise operations. This accelerator optimizes computation and data flow processing efficiency through specific hardware modules. Additionally, the architecture addresses the sparse data processing requirements using bitmap encoding and a zero-skip mechanism, reducing redundant computations and significantly enhancing hardware resource utilization. Through the CFU interface, these custom instructions are seamlessly integrated with TensorFlow Lite for Micro, enabling deep hardware-software co-optimization.

The experimental results show that the accelerator performed exceptionally well in CSANet inference, reducing the overall inference time from 55 million cycles in CPU mode to 6.9 million cycles, achieving a 79.7-fold performance improvement. In the optimization of conventional convolution and logic operation modules, the accelerator achieved speedups of 242x and 476x, respectively. Despite the FPGA's operating frequency being only 100 MHz, the inference performance of the accelerator is comparable to that of modern high-end smartphones. In terms of energy efficiency, although power consumption increased from 0.553 watts to 1.587 watts, the significant performance gains resulted in a 27.8-fold improvement in overall energy efficiency. Regarding resource utilization, the accelerator occupied 32.1% of the FPGA's logic units and 19.2% of its DSP resources, achieving a reasonable balance between performance and resource consumption given the scale of the design [1].

The accelerator offers several key advantages. First, the lightweight design of the quantized CSANet model is well-suited to the storage and computational constraints of edge devices, while the accelerator significantly enhances the model's inference efficiency. Second, the integration with the CFU-Playground development framework facilitates hardware-software co-optimization with TensorFlow Lite, streamlining the development process and providing developers with flexible access to hardware resources. Third, the introduction of bitmap encoding and the zero-skip mechanism drastically reduces redundant computations in sparse data processing, improving hardware utilization. Lastly, the accelerator is versatile, not only optimized for CSANet but also

DOI: 10.54254/2755-2721/121/2025.19489

supporting core computational modules in other deep learning models such as MobileNet, demonstrating substantial application potential.

3. RISC-V optimizations in deep learning inference

RV-SCNN is a custom processor specifically designed for edge devices, with its key features based on the RISC-V general-purpose instruction set. Introducing multiple SIMD (Single Instruction Multiple Data) custom instruction extensions, significantly enhances the inference performance of spiking neural networks (SNNs) and convolutional neural networks (CNNs). The core feature of RV-SCNN is the reuse of key operation modules required for both SNN and CNN inference, allowing it to support both computation modes. Additionally, the architecture optimizes memory access overhead through hardware loop control units, address calculation units, and inter-layer fusion units, while incorporating an IM2COL unit to improve the efficiency of 3×3 convolution operations.

In the experiments, the researchers evaluated the inference performance of RV-SCNN using CNN networks such as LeNet, VGG11, and MobileNetV1, as well as SNN networks based on fully connected and convolutional layers. The results showed a significant acceleration in matrix multiplication and convolution operations with RV-SCNN. For matrix multiplication, RV-SCNN achieved up to 18.72 GOPS (for SNN) at a clock frequency of 300 MHz, reducing latency by more than 95%. In convolution operations, RV-SCNN optimized memory layout through the IM2COL unit, reaching up to 3.29 GOPS in CNN mode and 26.46 GOPS in SNN mode. Furthermore, for full network inference, RV-SCNN achieved speedups of 88.3%, 95.15%, and 93.75% on LeNet, VGG11, and MobileNetV1, respectively [2].

In terms of hardware resources, RV-SCNN's resource consumption increased compared to the baseline RI5CY architecture, particularly in the use of logic units and flip-flops, which grew by 180% and 140%, respectively. This increase is primarily attributed to the complex control logic of the SCNN array unit and the IM2COL unit. However, due to the highly optimized design, the overall power consumption only increased by 23.3%. In the ASIC implementation, RV-SCNN was fabricated using a 55nm CMOS process, consuming 3.09 mW at 300 MHz, demonstrating strong energy efficiency. For SNN inference, RV-SCNN achieved an energy efficiency of 9.88 pJ/SOP, while for CNN inference, it reached 679 GOPS/W.

Compared to other mainstream SNN and CNN accelerators, such as Eyeriss and ODIN, RV-SCNN stands out with its versatility and flexibility, rooted in its RISC-V-based architecture. It demonstrates exceptional acceleration and energy efficiency in both SNN and CNN inference tasks. The design of RV-SCNN offers an efficient, flexible, and low-power solution for neural network inference on edge devices, presenting significant potential for a wide range of applications [2].

4. LLVM RV32X extension in graphics rendering

A framework supporting the RISC-V RV32X graphics extension instructions has been built based on LLVM. The RV32X instruction set defines a set of graphics processing instructions designed for efficient handling of 3D images and media data. The RV32X architecture integrates both CPU and GPU functionalities, divided into four subsets: pixel processing, graphics pipeline processing, texture processing, and vertex shading.

To validate the performance of the RV32X extension instruction set, Wang and his team designed a benchmark ray tracing program and performed compilation and feature analysis of RISC-V, X86, and ARM architectures using the LLVM and Clang toolchains. The experiments utilized the MICA feature mining tool to analyze instruction-level parallelism (ILP), instruction mix ratios, and register dependencies and memory reuse distances across seven ray tracing programs. By comparing the ray

DOI: 10.54254/2755-2721/121/2025.19489

tracing programs on RISC-V, X86, and ARM architectures, the study highlighted the characteristic advantages of RV32X in high-performance, low-power graphics rendering.

The experiment employed various rendering features of ray tracing, such as anti-aliasing, material textures, and metallic reflections, which were run on RISC-V and other mainstream architectures. The functionality of the RV32X extension instruction set was validated through compilation and assembly, with all 56 custom instructions passing the llvm-lit tests, confirming that the extension is correctly and comprehensively implemented in the LLVM backend.

The experimental results demonstrated that the RV32X architecture exhibits high instruction-level parallelism (ILP) in ray tracing programs. For instruction windows of 32, 64, 128, and larger sizes, RISC-V outperformed X86 in ILP, though it was slightly behind ARM. This indicates that the RV32X instruction set enhances parallelism through optimized hardware pipeline design. In the instruction mix analysis, the proportion of arithmetic instructions in the RISC-V architecture was higher than in X86 but lower than in ARM, while the proportion of memory access instructions was higher than in X86 but lower than in ARM. These results reflect the efficiency of RISC-V's lightweight instruction set in graphics rendering tasks.

In memory-level feature analysis, the RV32X architecture exhibited shorter register dependency distances and memory reuse distances. The reduction in register dependency distance can be attributed to RISC-V's streamlined design and efficient register usage strategy, while the advantage in memory reuse distance indicates that RV32X maximizes data reuse through registers, reducing the need for frequent memory accesses. Compared to X86 and ARM, RV32X showed higher memory reuse efficiency.

By integrating multiple subsets of graphics processing, the RV32X extension instruction set significantly improved the performance and flexibility of ray tracing programs. The support from the LLVM backend enables the RV32X architecture to be compatible with a wide range of graphics rendering algorithms, covering applications from basic rasterization to complex ray tracing. Additionally, RV32X's vector register extension and custom register design optimized pipeline performance while reducing power consumption. These architectural advantages lay the foundation for future CPU-GPU fusion processor designs based on RISC-V.

The research conducted by Wang's team implemented the design and support of the RISC-V RV32X graphics extension instruction set through the LLVM toolchain, and experimentally validated the performance advantages of the RV32X architecture using ray tracing programs. The efficiency and flexibility of RV32X demonstrate the significant application potential of the RISC-V architecture in the field of graphics rendering. Future research directions include the design of a RISC-V-based GPU simulator to further optimize rendering pipelines, cache units, and scheduling strategies, with the aim of supporting more complex graphics processing tasks [3].

5. Modular design of the pluggable vector unit (PVU)

The architecture design focuses on modularity and scalability, minimizing reliance on the scalar core by adding vector instruction queues and synchronization mechanisms. The vector instruction queue facilitates the transmission of instructions and control information between the scalar core and the vector unit, while also supporting synchronization and backpressure control. The design incorporates speculative execution to reduce potential performance losses caused by vector instructions, and an accurate trap mechanism ensures the ability to precisely restore the execution state in case of exceptions.

The architecture was validated on the Xilinx Zynq UltraScale+ FPGA platform, with resource usage and performance evaluated under various vector length (VLEN) configurations. The tests were conducted using an extended CVA6 core, with the vector unit implemented in 10,000 lines of

DOI: 10.54254/2755-2721/121/2025.19489

SystemVerilog code, while the scalar core extension required fewer than 400 lines of code. The design also includes a prototype scratchpad memory, serving as a cache for vector memory accesses.

The experiment tested a vectorized AES encryption algorithm (XTS-AES) and compared the performance between vectorized and scalar implementations. The results reported the usage of look-up tables (LUTs), flip-flops (FFs), and the maximum frequency performance under different vector lengths (128, 256, and 512 bits).

The experimental results show that the resource usage of the vector unit increases linearly with the vector length. For a VLEN=256 configuration, the LUT usage was 94.8K, the FF usage was 31.2K, and the maximum operating frequency was 84 MHz. In the XTS-AES encryption algorithm, the vectorized implementation achieved encryption speeds 2.9 times faster than the scalar implementation at VLEN=256 and 4.1 times faster at VLEN=512. Additionally, the vectorized execution consumed slightly lower frequency compared to the scalar implementation, but the overall speedup was significant.

The design of the vector unit offers unique advantages over other academic or industrial implementations, particularly in supporting speculative execution and precise trap handling. For instance, the vector register file employs a multi-copy design, allowing fast rollback through a register renaming mechanism in case of exceptions or branch prediction failures. In contrast, other implementations typically rely on more complex rollback techniques, which result in additional hardware overhead.

The pluggable vector unit designed by the Maisto team offers several significant advantages. First, through a microprogrammed SIMD pipeline design, it simplifies the serialization and data allocation logic of vector operations, thereby reducing the pressure on the vector register file. Second, the custom locking protocol for the vector register file ensures data consistency while enhancing resource utilization efficiency. Additionally, the architecture is designed with versatility in mind, enabling easy integration of third-party arithmetic units and providing flexible interfaces for extending advanced features such as floating-point operations.

The design of the vector unit also emphasizes scalability and modular reuse. For instance, by introducing minimal extension logic within the scalar core, the integration of the vector unit remains highly compatible with other RISC-V cores. Compared to industry implementations such as SiFive's DLEN architecture, this design is more efficient in supporting cross-element operations, such as vector gather and scatter instructions. Furthermore, the single MMU design eliminates the need to configure independent TLBs for each vector channel, significantly simplifying the hardware architecture.

The Maisto team's design introduces a RISC-V vector unit that supports speculative execution and precise trap handling, achieving efficient data parallel processing through a modular approach. Experimental results show that the design strikes a good balance between performance, resource utilization, and scalability. Future work will focus on further optimizing the memory architecture, exploring the possibility of shared caches between scalar and vector units, and extending the register renaming logic to improve data dependency resolution. This research provides valuable insights for efficient RISC-V hardware implementations aimed at data-parallel applications [4].

6. Reconfigurable CNN accelerator in instruction-extended RISC-V cores

CNN-AS, based on the RISC-V RISCV32IMA instruction set extension, has been designed with a dedicated instruction set tailored for CNN inference. Unlike traditional FPGA implementations that rely on a single-architecture CNN accelerator, this architecture leverages the modular design of RISC-V, allowing for runtime reconfigurability of various CNN models via software API adjustments. The core hardware includes a specialized Computing Engine (CE), specific memory structures such as

DOI: 10.54254/2755-2721/121/2025.19489

weight caches and feature map caches, and a statistical module utilizing a dynamic fixed-point scheme.

To address the convolution operations, which account for over 90% of CNN inference, CNN-AS adopts a single CE architecture with a layer-by-layer acceleration mechanism. This architecture reduces external memory accesses and optimizes data flow design, significantly lowering IO bandwidth requirements while improving the accelerator's versatility across multiple network models. The computing engine features a general-purpose Multiply-Accumulate (MAC) array capable of supporting various convolution kernel sizes (1×1, 3×3, and 7×7) and optimizes loop unrolling, loop nesting, and buffer management.

A key innovation of CNN-AS is its dynamic fixed-point (DFP) number scheme. By assigning common exponents to specific data groups, the DFP scheme combines the dynamic range of floating-point numbers with the low-latency advantages of fixed-point numbers. In convolution operations, the DFP scheme not only reduces the shifting overhead typical in floating-point computations but also improves hardware performance and inference efficiency through a dynamic balance of quantization and truncation errors.

To comprehensively evaluate the performance of the CNN-AS architecture, the study integrates hardware-software co-simulation and physical hardware implementation to test several CNN models. Experiments were conducted on the Xilinx Alveo U200 FPGA platform, using deep learning models such as RESNET34, RESNET50, RESNET101, and VGG16. The testing process was divided into software generation, instruction verification, simulation execution, and performance measurement stages, providing detailed insights into the hardware resource utilization, power consumption, and throughput efficiency of the CNN-AS architecture.

The experiment begins with converting pre-trained CNN models into instruction sets suitable for hardware accelerators. To achieve this, the researchers employed the RISC-V GNU toolchain to translate the RESNET and VGG models from the Keras and TensorFlow frameworks into assembly-level instruction files. Using the LLVM-C2RTL toolkit, all CNN-AS designs were written in C++, and then compiled to generate RTL code in Verilog format. This step ensured the cycle accuracy and bit-accuracy of the hardware description while laying the groundwork for hardware simulation.

Subsequently, the experiment validated the custom instruction set designed for CNN-AS, which includes data load instructions, zero-overhead loop instructions, and computational instructions. The data load instructions primarily serve to load feature maps and weight data via the AXI4 interface, ensuring pipeline continuity. The zero-overhead loop instructions, implemented in hardware, enable loop unrolling, thereby improving the efficiency of convolution operations during CNN inference. The computational instructions cover convolution operations for different kernel sizes (such as 1×1, 3×3, and 7×7) as well as instructions for pooling layers, including max pooling and average pooling. These instruction sets, through hardware-software co-design, ensure CNN-AS’s adaptability across different CNN models.

During the simulation phase, the researchers designed a comprehensive LLVM-based hardware simulation environment to perform functional verification of the RTL code. Using simulation tools, they were able to analyze the instruction flow step by step, while also identifying delays, data conflicts, and resource bottlenecks in the hardware. The simulation results demonstrated that the CNN-AS instruction set was fully aligned with the hardware design, providing reliable data support for subsequent hardware testing [5].

In the hardware implementation, the experiment deployed the RTL code of CNN-AS onto the Xilinx Alveo U200 FPGA board to evaluate its performance during actual operation. The experiment focused on several key metrics: throughput efficiency (TE), which is the ratio of actual throughput to theoretical maximum throughput; GOPS/W (Giga Operations Per Second per Watt), used to measure energy efficiency; and hardware resource utilization, including the occupancy of LUTs, DSPs, and

DOI: 10.54254/2755-2721/121/2025.19489

on-chip memory. Specifically, the experiment measured the feature map cache hit rate and weight loading efficiency for different network architectures (such as RESNET and VGG), and compared the impact of the dynamic fixed-point number scheme on inference accuracy and power consumption.

To further optimize the inference process, the researchers implemented a dynamic Ping-Pong buffering mechanism in the experiment. This mechanism allows the next set of weight data to be loaded into the on-chip cache via the AXI4 interface while the computing engine is performing the current convolution operation, thereby reducing pipeline idle time caused by weight loading. Additionally, the experiment adjusted the buffer allocation strategy and data flow structure based on the weight sizes and feature map dimensions of different convolution layers, maximizing the utilization of on-chip resources.

The experimental results provide a comprehensive analysis of CNN-AS's performance in real-world applications by measuring inference time, power consumption, and accuracy for RESNET34, RESNET50, RESNET101, and VGG16 models. The researchers also compared CNN-AS with other FPGA-based CNN accelerators, including those using multi-compute engine (Multi-CE) architectures and non-SoC designs, in order to fully assess CNN-AS's strengths and weaknesses. This experimental analysis offers significant guidance for optimizing and improving hardware accelerators.

CNN-AS achieved a throughput efficiency (TE) of 90% on the RESNET34 model, comparable to Multi-CE non-SoC FPGA architectures. In the VGG16 model, TE slightly decreased due to the large weight data size and relatively low computational load of the fully connected layers. Compared to other CNN accelerators, CNN-AS excelled in the GOPS/W (operations per watt) metric, particularly showing significant advantages in RESNET models.

The dynamic fixed-point (DFP) scheme notably reduced hardware resource overhead during inference. Results showed that CNN-AS, using DFP, minimized power consumption and improved throughput efficiency while maintaining a small loss in accuracy. For RESNET50, CNN-AS achieved a Top-1 accuracy of 70%, close to the 71% achieved by the TensorFlow floating-point implementation, but with only 47% of the power consumption.

Hardware resource analysis indicated that CNN-AS utilized LUT and DSP resources efficiently, while maintaining power consumption at 3.26 watts, far below that of many Multi-CE architectures. Its on-chip memory design effectively cached feature maps and weight data, reducing external DDR access frequency. Moreover, the Ping-Pong buffering mechanism further optimized weight loading and computation pipelines for convolutional layers.

The main advantages of CNN-AS lie in several areas. First, the dedicated instruction set extends the capabilities of the RISC-V core, accelerating high-frequency operations in CNN inference, and improving instruction efficiency through hardware loop control units. Second, the single-CE architecture's layer-by-layer execution strategy allows the accelerator to adapt to different CNN models without large-scale hardware modifications. Third, the dynamic fixed-point scheme optimizes quantization and truncation errors, significantly reducing power consumption while ensuring inference accuracy. Finally, the software-hardware co-optimization design reduces data movement overhead, significantly enhancing system energy efficiency.

Compared to other FPGA-based CNN accelerators, CNN-AS stands out in terms of throughput efficiency, power consumption, and resource utilization. Future research will focus on optimizing on-chip cache structures to further improve TE and GOPS/W metrics, and exploring the applicability of the DFP scheme in additional deep-learning models. This study provides an important technical reference and implementation example for designing RISC-V-based CNN hardware accelerators [6]. By exploring RISC-V's vector extensions, custom instruction sets, and their application in deep learning inference, the paper reveals the vast potential of RISC-V in the field of image processing. The related studies demonstrate, both theoretically and experimentally, the flexibility brought by RISC-V’s openness and modularity, while achieving efficient and low-power image processing

DOI: 10.54254/2755-2721/121/2025.19489

solutions through hardware optimization and algorithmic improvements. These achievements not only enhance RISC-V's influence in academic research but also offer important insights for future industrial applications [7][8]. Future research could focus on the following areas: firstly, optimizing hardware design and software interfaces to accommodate more complex models and algorithms, particularly for 3D graphics processing and larger-scale convolutional neural networks. Secondly, enhancing on-chip storage and cache structures to meet the demands of higher data throughput. Thirdly, exploring the potential of RISC-V architecture in heterogeneous computing, such as by integrating CPU, GPU, and NPU modules to develop efficient multi-core collaborative image processing solutions. Finally, the programmability of RISC-V offers promising opportunities for the fusion of artificial intelligence with traditional algorithms, paving the way for the development of more comprehensive toolchain support to lower the barriers for developers [9][10].

7. Conclusion

This paper provides a comprehensive review of the research progress in image processing and hardware accelerator design based on the RISC-V architecture.Despite achieving significant results, several challenges remain that warrant further investigation. In summary, research on RISC-V-based image processing hardware acceleration provides crucial support for the future of embedded computing, edge devices, and low-power applications. The findings presented in this paper not only enhance the academic understanding of RISC-V's potential but also offer reliable technical guidance for practical engineering design. With the introduction of more innovative technologies, the application prospects of RISC-V in the field of image processing will continue to expand, playing a pivotal role in the future of intelligent devices and efficient computing.

Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

References

[1] *Wu, Z.-M., Lin, Y.-C., & Liu, C.-W. (2024). AI-ISP Accelerator with RISC-VISA Extension for Image Signal Processing. 2024 International VLSI Symposium on Technology, Systems and Applications (VLSI TSA),*

[2] *Wang, P., & Yu, Z.-B. (2023). LLVM RISC-V RV32X graphics extension support and characteristics analysis of graphics programs. Ieee Access, 11, 67285-67297.*

[3] *Maisto, V., & Cilardo, A. (2022). A pluggable vector unit for RISC-V vector extension. 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*

[4] *Wang, H., Li, D., & Isshiki, T. (2023). Reconfigurable CNN accelerator embedded in instruction extended RISC-V core. 2023 6th International Conference on Electronics Technology (ICET),*

[5] *Wang, X., Feng, C., Kang, X., Wang, Q., Huang, Y., & Ye, T. T. (2024). RV-SCNN: A RISC-V Processor With Customized Instruction Set for SNN and CNN Inference Acceleration on Edge Platforms. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

[6] *Gholizadehazari, E., Ayhan, T., & Ors, B. (2021). An FPGA implementation of a RISC-V based SoC system for image processing applications. 2021 29th signal processing and communications applications conference (SIU)*

[7] *Li, D.-Z., Gong, H.-R., & Chang, Y.-C. (2018). Implementing RISCV system-on-chip for acceleration of convolution operation and activation function based on FPGA. 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*

[8] *Perotti, M., Cavalcante, M., Wistoff, N., Andri, R., Cavigelli, L., & Benini, L. (2022). A "new ara" for vector computing: An open source highly efficient risc-v v 1.0 vector processor design. 2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*

[9] *Tine, B., Saxena, V., Srivatsan, S., Simpson, J. R., Alzammar, F., Cooper, L. P., Jijina, S., Rajagoplan, S., Kumar, T. A., & Young, J. (2022). Accelerating Graphic Rendering on Programmable RISC-V GPUs. 2022 IEEE Hot Chips 34 Symposium (HCS)*

[10] *Yılmaz, Y., Tozlu, Y. S., & Örs, B. (2021). Design and Implementation of a 32-bit RISC-V Core. 2021 13th International Conference on Electrical and Electronics Engineering (ELECO)*