SCIENTIFIC Publishing Limited

Digital Technologies Research and Applications

http://ojs.ukscip.com/index.php/dtra

Review

An Extensive Analysis of Techniques and Developments in the Progress of RISC-V Customization and Hardware/Software Co-Design

Jala Himabindhu and Shaik Karimullah *

Department of Electronics and Communication Engineering, Annamacharya University, Rajampet 516126, India

* Correspondence: munnu483@gmail.com

Received: 22 October 2025; **Revised:** 10 December 2025; **Accepted:** 13 January 2026; **Published:** 18 August 2026

Abstract: Co-designing hardware and software and customizing Reduced Instruction Set Computer-V (RISC-V) have become important strategies for maximizing computer systems' performance, adaptability, and resource efficiency. Architectural review, neural network-based partitioning, and reconfigurable array designs are some of the major developments in the field of co-design concepts explored in this research work. The major advances in parallel computing and automation of the design process include the Adaptive Dataflow Architecture for Processing and Training Optimization (ADAPTO) array, Configurable Tagged Memory Extension (COMET) technique, and Kahn Process Networks. Dynamic micro-decoders, unique instruction set, and vector extensions for Artificial Neural Network (ANN) optimization are a few of the important developments in this area. It comprehensively compares the performance improvement, resource usage, and design automation framework that overcomes the current constraints for the broadening of RISC-V applications to many computing domains. The paper also underlines some state-of-the-art techniques. Current innovations enable us to realize a number of new technologies, which utilize advanced techniques, such as hardware-software co-design; adaptive pipelines; and intelligent memory management, leading to greatly enhanced computation efficiency. A number of the new methods being studied for both AI and edge computing are: custom neural processors, low-power accelerators, automated high-level synthesis tools, etc., all providing alternate means to meet increasing requirements for AI and edge computing use. All of these factors contribute to improved performance, lower latencies, and optimized use of resources and therefore, place RISC-V processing architectures as being the preferred architecture for the Intelligent Computing Solutions of the future.

Keywords: Hardware/Software Co-Design; Design Automation; Instruction Set Architecture (ISA) Customization; Processor Optimization; Instruction Set Expansion

1. Introduction

Recent studies on RISC-V customization and hardware/software (HW/SW) co-design have attracted significant attention due to their potential impact on embedded systems and modern processor architecture design. As computing systems continue to evolve rapidly, there is an increasing demand for architectures that are not only efficient but also scalable, flexible, and adaptable to diverse workloads. Conventional approaches that treat hardware and software as independent entities often struggle to meet these requirements, as performance bottlenecks arise when software demands exceed hardware capabilities. To address these challenges, researchers have explored various Instruction Set Architecture (ISA) extension techniques and HW/SW co-design methodologies, enabling tighter in-

tegration between hardware and software. Such approaches help bridge the gap between computational demands and available system resources, resulting in enhanced performance, energy efficiency, and application-specific optimization.

The open-source RISC-V ISA is particularly well suited to such efforts due to its modular and extensible design. This flexibility enables designers to introduce custom instructions, optimize execution pipelines, and integrate application-specific accelerators, thereby enhancing performance for targeted computing workloads in a clear and efficient manner. Classes of computing tasks, including signal processing, security, machine learning, and edge computing, will see performance gains with custom RISC-V cores characterized by lower latency, higher throughput, and improved energy efficiency. The open-source nature of RISC-V is further enhanced by access to open-source toolchains, simulation tools, and verification frameworks for prototyping and experimentation that reduce time and expense to develop and prototype thus providing more opportunities compared to proprietary architectures.

The aforementioned performance enhancement can be completely realized by employing hardware/software co-design. Performing hardware/software co-design exploration at all abstraction levels including processing cores, memory hierarchies, specialized accelerators and associated software routines allows for tuning resource allocation; enhancing pipeline efficiency; optimizing system-wide effective performance. Early co-design consideration was based on instruction-level simulation, as well as analytical capacity models to evaluate architectural performance, bottlenecks, and optimization opportunities. As of recent, additional (software-based) automated partitioning has been added to design, to algorithms that can automatically distribute workloads among hardware cores or smart schedulers that, through run-time learning, determine what tasks best executed in hardware and/or software. The work also incorporated machine learning to guarantee the techniques work fairly, and that the algorithms are designed to avoid unutilized machines that have nothing to execute Several real-world use cases highlight the benefits of co-designing using RISC-V. Edge AI products leverage specialized RISC-V cores customized to accelerate convolutions for Convolutional Neural Network (CNN) inference, the primary bottleneck in providing real-time execution in small battery powered form factors with very little power. RISC-V cores that have been designed for high performance computing (HPC) for dedicated matrix operations, cryptography and data compression will enable higher throughput and lower latency over general purpose processors. Embedded systems demonstrate smaller form factor, less power and optimized for the application due to specialized peripheral integration and hardware acceleration. Overall, these use cases indicate that hardware and software co- design not only improve computing performance, but also easily enable application centric system optimization. While the future is promising, there remain various challenges to overcome. Designing an ideal RISC-V system requires some compromises to be made in respect to program quantity, performance, power, area and software flexibility. Further complexity comes from increasing instances of simultaneous multi-core architectures, heterogeneous systems and real-time applications to be jointly designed. As a result, optimization and empirical techniques can take you part way but generally find solutions that may not be globally efficient across the system. As a consequence, there is an increasing demand for end-to-end processes that take advantage of simulation, formal verification, automatic assessment and adaptive monitoring of workload.

In the short future, the horizons for RISC-V hardware/software co-design will be intertwined with AI, machine learning and design automation. For instance, AI influenced partitioning, adaptive ISA extension and HW configuration tuning would enable designers to achieve higher performance in shorter design time period while providing more efficient power and System-on-Chip (SoC) performance. As open-source collaboration and community-based toolchains evolve, this will deliver innovation along with an increased ease of use of methodologies for RISC-V co-design in industry and academia. In conclusion, RISC-V customization and Hardware/Software (HW/SW) co-design are critical forces in today's advances in computation, as they enable efficiency, flexibility and performance optimization for specific applications. Advanced co-design methodologies, open-source architecture and intelligent optimization methods will empower researchers in this space to create computing systems that are able to respond to the challenges of the ongoing complexity of workloads in addition to paving the way for future processing and packaged embedded systems development.

Figure 1, displayed below, presents a continuum of how computer systems can be designed from exclusively hardware to entirely software-driven designs. The position of ASICs on the far-left illustrates an extremely hardware-driven design. An Application-Specific Integrated Circuit, or ASIC, is a hardware device that has been customized for one function and cannot be used for any other function. A general purpose processor can perform a variety of

calculations across a full range of applications, while ASICs have been completely optimized for the application it performs. This level of optimization can result in higher performance and higher efficiency in terms of operating power when compared to any other hardware, even the most general hardware. When the architecture optimally performs one function, every logic gate, memory element, and path for data can be optimized for speed, latency, and power. For instance, when used for cryptocurrency mining, video encoding or processing, ASICs can operate and run several orders of magnitude faster than an equivalent general-purpose CPU, since ASICs have a purpose-built, highly focused architecture. Unfortunately, such extreme focus has its drawbacks, there is no flexibility. Once the devices are built, they cannot be modified or pivoted to a new use, or a completely new algorithm, or even a modified new version of the algorithm, actually, they have zero flexibility. Further, the design and implementation of ASICs can often be a challenging and expensive process because of their extreme dependence on box ultra- complex fabrication and multiple set up for iterations of production test lots. While there may be some challenges with ASICs, if the highest priority requirements are 'performance' or 'energy used' even with all the challenges, they will win every time.

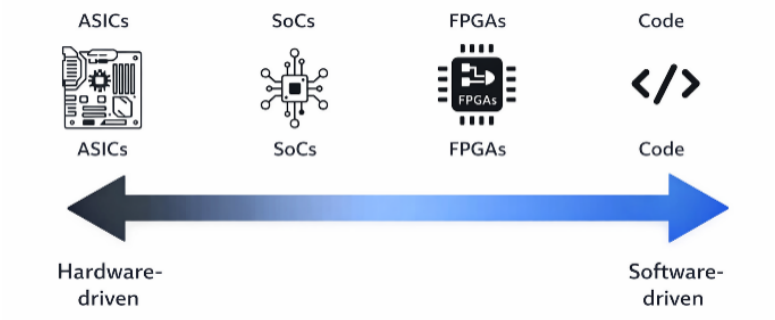

Figure 1. Hardware-Software Co-Design.

There are SoC implementation in the sphere of application specific ICs, that are somewhere in between performance and flexibility. An SoC implementation can accommodate significant processing in technology terms, a CPU and GPU cores, memory-management unit, and a communication interface, often to external peripheral, for e.g., HDMI, as one piece of silicon substrate. An SoC can, however, still achieve the required level of integration, to fulfill the ultimate requirements of a use, for e.g., hardware optimizations. An SoC can also provide further flexibility or trigger instances of hardware to achieve features when compared to piecing together a locked down ASIC. Many current smart phones and embedded devices are generally using one SoC, that features a variety of processing function—imagery processing, supportive functions, communication functions, security processing, and user interfacing, for e.g., SoCs reduce the time of communication among components and also consume less power than the system using discrete chips. Also, SoCs give designers a chance to choose between computational efficiency, flexibility, and therefore easy updates and addons (programmatically) as opposed to hardware. Even though SoC devices are more flexible than ASIC devices, they usually won't be as performant-efficient for a dedicated process than a customized ASIC. Designers need to carefully consider workload, power limits, and cost if they are to select an SoC for a particular application.

In comparison, an FPGA (Field-Programmable Gate Array) provides slightly greater flexibility than SoCs. FPGAs possess an additional layer of configurability, meaning it can potentially be configured after the hardware is produced to specify the logic and connections within the hardware. Because you can program the FPGA, engineers have practically unlimited ability to create or recreate complex digital logical operations without needing to produce sufficient silicon. FPGAs are advantageous when tight schedules for quality rapid prototyping are desirable, or exploratory algorithm development is favored or pending feedback, such as for machine learning accelerators, DSP, or packet processors. FPGAs may additionally be programmed in "on the fly," assuming the FPGA hardware meets its constraints, to execute multiple tasks, or executions workflows in parallel, improving performance for some computational workloads. FPGAs may also be programmed...to reach bi-directional power efficiency or performance goals via logic configuration and clocking schemes, albeit at an added cost. Understanding that FPGAs have been used for decades to enable performance when compared to ASICs, FPGAs commonly require more power and provide

inferior raw performance than ASICs, primarily because programmable routing and programmable logic blocks carry added overhead not present in a fixed-function design. Programming FPGAs requires an understanding of specific hardware description languages (e.g., VHDL or Verilog), resulting in a learning curve that is longer than what software engineers typically experience. Nonetheless, FPGAs occupy a niche in-between ASICs that require a defined hardware function to be incorporated, and SoC systems that are more flexible to program compared to ASICs. FPGAs afford users hardware performance without the ASIC like performance efficiency, coupled with FDA flexible process that require the developing, modifying, and/or implementing process for hardware where maybe the available SoCs do not easily or efficiently allow. Typically, the considerations of determining whether to design a system around an ASIC, SoC or FPGA comes down to specifications for the project, schedule, budget and expected lifetime of the product. That said, ASICs can still be leveraged in a flexible, programmable component that can be used more than one function across multiple product processing workloads. ASICs make sense when you are primarily developing a high volume, high specialty systems that is always going to be under fixed processing workloads, because there is usually not an alternate to compensate for the work efficiency of an ASIC. SoCs are great option when developing a System that requires some flexibility on the design, and working with one or multiple subsystem based implementations. FPGAs are almost synonymous with the hardware development lifecycle of systems with prototyping, iterative development or real-time updates of algorithms, and in all cases some form of building or coding an algorithm to implement an embedded architected or engineered algorithm to hardware will be of the same nature of algorithm computation workload implementation by the SoC or customer in product.

Hybrid SoCs or hybrid FPGAs are a newer design practice, and Engineers and researchers need to be mindful of the characteristics and trade-offs of the technology between both design methodologies subset so that they can design computing systems that are performance-oriented, energy-efficient, and flexible. The ability to select and combine the various technologies designers can create computing systems that satisfy current computational requirements and are position to advance future innovation and changing application requirements.

2. Methodological Landscape

The quick evolution of computer technology has prompted significant research efforts in customizing RISC-V and co-design methods. The significance of these methodologies continues to grow, as a successful modern computing systems must be high-performance, energy-efficient, value-oriented, and adaptable depending on the application.

2.1. Role of RISC-V in Co-Design Methodologies

Co-design is fundamentally different from the traditional practice of creating hardware and software components within separate workflows, as co-design proposes the development of the hardware and software components together. The combination of the hardware and software components allows the system designer to accomplish system level performance optimization while sustaining flexibility and decreasing the development cost and time to market of new processors and embedded platforms. The open, modular, and customizable surfaces of the RISC-V ecosystem allow for the development of system solutions that can utilize off-the-shelf RISC-V hardware and enablers and incorporate custom hardware and software components. The characteristics of the RISC-V Instruction Set Architecture (ISA) will provide Dee a launching pad for co-design activities to have researcher and engineers optimize process for some developer application, artificial intelligence acceleration, edge computing, high-performance scientific computing.

2.2. Early Hardware/Software Partitioning Frameworks

Initial research in this focus area, was mostly exploration for architectural evaluation and simulation. Hubert and Hammami proposed several automatic methods for architectural evaluation, in those approaches, even though they were simulation at the instruction level, produced reasonable estimates of performance. It gave the designer a chance to observe the execution characteristics of that computing, predicting key problem areas, bottlenecks, and optimizations, without building a valued hardware prototype. At the same time, Nişancı et al. proposed COMET, a co-design framework, to divide workload tasks using neuro networks. COMET provided partitioning of workloads onto hardware/software to best meet timing and memory constraints and (sometimes) fit by autonomous imple-

mentation, as a partition delineates the balance of extent and performance requirements. The originality shown hinted at the balance of managing both Hardware/Software operating in pairs of tasks, instead of two streams of development.

In recent years, applied design tools and prototyping methods have advanced in RISC-V co-design. Similarly, Mohr et al. had a team of computer scientists create a desktop application that facilitates RISC-V processor design and debugging. The application is beneficial to researchers and educators by extending the approach to modern desktop computing. Likewise, Pieper et al. enhanced virtual prototyping frameworks by adding modular device behavior and scripting engines. These frameworks increase the ability to configure processor components on the fly, decrease the time to complete experimental studies and validate designs more quickly. Additionally, flexible software tools that facilitate simulations also allow for the consideration of an even broader spectrum of optimization strategies that apply not only to the RISC-V pipeline but also to the instruction set architectures and how these architectures can be exploited to accommodate specific application constraints on computing platforms. Despite these advancements, challenges remain in achieving a fully optimal design. According to Bastien Hubert and Omar Hammami, empirical optimization techniques typically yield designs that are locally optimal or optimal within a very limited domain. Adding to this, as modern systems are on a trend of increasing complexity multi-core processors, heterogeneous computing platforms, or any number of application-specific accelerators—co-design is increasingly relevant as an abstraction for balancing trade-offs and between performance, energy, hardware area, and software flexibility. Thus, when designing for multi-objectives, we must consider complex toolchains, automated assessment strategies, and intelligent workload partitioning, to maximize performance of the end design in multiple contexts.

2.3. Real-World Applications of RISC-V Co-Design

The co-design process using RISC-V has been used in exciting real-world use cases. For example, edge AI hardware can leverage specialized RISC-V CPU cores designed to execute CNN inference in small power envelopes, while high-performance computing systems could utilize RISC-V cores for matrix, cryptography, or data compression-type workloads that have low latency and high throughput. In embedded systems, co-design using RISC-V can provide ways to quickly design and build application-specific accelerators and peripherals that allow for a smaller, lower-power solution while being more application specific than alternative architectures. These use cases illustrate that the co-design paradigm is not merely designing for better, faster computing, but a way to enable, scalable application-driven optimizations and adapt to new calculation form factors as technology trends continue to change Hardware/software co-design using RISC-V is also expected to develop in the future, due to the growing excitement of integrating artificial intelligence, machine learning, and Electronic Design Automation (EDA) tools into the design process. Current research is exploring AI-driven partitioning strategies, automated instruction set generator strategies, and adaptive micro-architecture designs that can change based on the characteristics of the observed workload. Such change provides the possibility of a shorter design cycle, improved energy efficiency, and improved performance predictability. Further, with the level of open-source toolchain sufficiently matured, collaboration between academic researchers, industry engineers, and open-source ecosystems will likely development speed of innovation, and create a more accessible co-design experience.

In summary, custom RISC-V and hardware/software co-design represents a significant development in modern computing systems, allowing for a means of managing and balancing the performance, flexibility, and efficiencies of computing resources. The use of automated evaluation methods; modular software and hardware infrastructures; and intelligent approaches to workload-partitioning have begun to allow designers to develop processors and systems based on RISC-V that can be customized in responses to present and future demands of systems applications. The interplay of useable tools, theoretical models, and empirical approaches to optimization has established a robust and evolving sub-field of research and custom RISC-V co-design; designers will continuously work to simplify. The challenges of basic computing naturally imply the challenge of integrating hardware and software skills and ultimately, the ongoing need to optimize an existing system for creating new systems in RISC-V co-design. Challenges exist across an array of systems, including AI and EDGE computing, high-performance computing, and embedded systems.

3. Comparative Analysis

This device improved the cycle counts by over 60% and significantly sped up the progress of RISC-V processors while assuming background knowledge in the RISC-V cores and hardware description language. Zou et al. (2023) [1] improved the Blake3 cryptographic hash function on RISC-V architecture to achieve a 10× speedup versus a Xeon 8280 processor which demonstrated around a 10% performance boost compared to the same implementation for Blake3 but with AVX512. The reasoning was to indicate the aided improvement of performance of RISC-V architecture with the utilization of the RISC-V implementation, but to indicate that the use case was branching or continuously encrypting of operations versus the performance increase directly imposed with RISC-V operations. In 2022, Choudhury et al. [2] designed the pipelined RISC-V processor, and added the Tournament Branch Predictor (TBP) processor with a pipelined RISC-V processor, with all 5 stages. The TBP increased the execution times, or time of computation and performance of operations as a direct result of the TBP. The TBP prediction methodology utilized an approach that predicted the flushing of instructions in the RISC-V program countered the stalled pipeline, as they were waiting for all possible combinations of the different protocol instructions, using data collected from the, especially for bypassing branch flushes stalled or maximizing the usage of the instruction types. The work pointed to the combination of intelligent prediction strategies mechanisms as the best option for high performance, since branching can generate performance degradation at high rates of computation.

In the same year, Dow et al. [3] introduced SHORE, a hardware/software co-design framework to engineer safer memory systems on RISC-V hardware. SHORE provided an astounding 3.79× speedup over the conventional software memory safety techniques, in that the vital memory safety operations were able to be moved to the hardware. In other words, there were safer checks tests on memory safety, while the applications were running, adding to the level of robustness against memory issues. However, the method had some issues. The use of proprietary closed-source microprocessors made the method limited to sharing and updating, therefore, less easy to adapt for your own programmes. Moreover, integrating the framework with existing software workflows introduced additional challenges, particularly in defining source-level specifications that align with input requirements. As a result, although SHORE was developed to be efficient and dependable, practical deployment required careful consideration of hardware descriptions and compatibility with existing software environments.

In addition to providing a detailed evaluation of the pros and cons of the RISC-V hardware/software co-designs, they affirmed the importance of being flexible and ready for open-source repo from a practical applications perspective, and supporting performance deserves mention, as enhancements related to memory safety acceleration and branch prediction have been presented and demonstrated in the literature to support increases in processor performance. In support of this progression, Yu et al. (2023) [4] took a step further and presented an activation instruction evaluation in end-to-end workloads limited to the RISC-V architecture that read and wrote the data in a Number of Ways very efficiently resulted in a speedup of 4.89×, the energy consumption remaining low at 7.78% compared to the original approach. The proprietary principle remains the same in this study, but confidentiality agreements restrict the dissemination of the data publicly. In 2023, Raza et al. [5] who engineered a 16-bit RISC-V-based ALU utilizing the CADENCE and XILINX ISE tools. Their labor led to a 90% increase in processing speed, a size reduction of 33%, and an 80% increase in power efficiency, indicating the RISC-V's capability of quite economical computational systems. Karimullah and Vishnuvardhan (2023) [6] developed an Arithmetic Logic Unit (ALU) based on the RISC-V architecture. The study aims to enhance ALU performance by efficiently supporting arithmetic and logical operations while exploiting the flexibility and simplicity of the RISC-V framework. Hence, an excellent hardware optimization in RISC-V for the above-mentioned LU decomposition resulted in 338 times higher speed up by the efforts of Bhargavi et al. (2024) [7] with matrices of dimensions 256 × 256.

It was possible to achieve such extraordinary results because designing was done for a Hardware accelerator for computation within matrices of different sizes. This concept of agile chip design has been introduced, called the MinJie platform for high-velocity hardware development. On another note, agile methodology has been implemented and has produced two generations of the XiangShan RISC-V processors from Y. Xu et al. [8] (like other challenges, there are obstacles that you cannot avoid when adopting these agile methodologies in hardware development). Wang et al. (2023) [9] focused on x264 source code and LU optimization in their RISC-V hardware accelerator decomposition. Deng et al. (2022) [10] presented a RISC-V-based software-hardware cooperative graph computing accelerator that was developed to reduce power consumption and hardware resources. Their design

methodology provided a significant increase in efficiency over past approaches, including 75.6% less hardware resources than HitGraph and 92.3% less power consumed in the algorithm. The massive increase, which was aadmittedly a major breakthrough, made evident that hardware-software co-design could allow the development of more efficient specialized accelerators. However, at the same time, the study did also indicate drawbacks, as the irregularity of graph computing topologies and having data positioned unfavorably with respect to cache memory were significant factors contributing to poor overall performance.

In the same vein, You et al. (2022) [11] presented dataflow co-processors to perform most computations, at least with respect to convolutional neural networks (CNNs). The system exploited the inherent parallelism in dataflow based execution, which allowed for higher computational throughput and consequently faster and more efficient task execution in RISC-V platforms, which is also beneficial. Yet, the use case was constrained by power efficiency and parallel execution when ported to FPGA boards. The limitations stemmed to a large degree from the limitations of traditional Von Neumann and Harvard architectures which constrain memory bandwidth and data movement, which limits the use potential of parallelism in high-performance graph and neural network workloads.

Together, these studies illustrate both the promise and the remaining challenges in RISC-V hardware-software co-design for specialized accelerators, emphasizing the need for optimized memory hierarchies, improved data locality, and architecture- aware scheduling techniques to fully exploit parallelism while maintaining energy-efficient operation. Nişancı et al. [12] during 2022, discussed some cryptography performance using RISC-V processors and, at the same time, propose new instructions to be used to increase speed with cryptography while reducing memory. They achieved a reduction in memory from 1.2× to 5.8×, with RISC-V's cryptography efficiency enhanced between 1.5× and 8.6× with their designed logic. This work is on the optimization of RISC-V ISA utilization in MPSoC architectures, carried out by Lima et al. in 2020 [13]. They focused their work on the utilization of shared resources by coprocessors for minimizing area, power, and energy consumptions. Their design may have been plagued with performance problems due to a longer critical path, even though it had few implications concerning execution time. A multi-core RISC-V SoC architecture was proposed by F. Wu et al. [14], for edge computing applications, a system that had an intelligent decision module and optimized storage architecture; this greatly reduced the energy consumption and increased the core speed, as was shown by theoretical and experimental simulations. Lastly, technology-independent mathematical models were employed for RISC-V CV32A6 CPU optimization, relating to machine learning use, by authors Bastien Hubert and Omar Hammami (2023) [15], their model worked promisingly only for the case of the low degree that heuristic methods tend to only retain a local minimum solution without using heuristics over empirical ones excluded from optimization challenges. Dharsni et al. [16], in the year 2022, optimized the pipelined RISC-V processor for RV32I pipelined architecture hazard-free. They used techniques like branch prediction and data forwarding. This improved pipeline performance by 7.82%, reduced power consumption by 4%, and worked on the RISC-V processor to run it more efficiently. An et al. [17] developed an easy-to-operate interactive desktop tool application to design and debug for RISC-V CPUs. As the test and debugging environment for RISC-V ISAs is complete, this application will serve as an excellent teaching tool for processor design engineers. It uses Verilog HDL on FPGA and C. This processor is another useful tool regarding teaching and education and using 6,476 LUTs and 49 MHz. In their work, proposed in 2023, Anu Priya et al. [18] added a dynamic microdecoder unit inspired by some CISC CPU to a RISC-V processor. The aim of this module was to make the CPU accept dynamic instruction sequences.

This provides more flexibility at execution time but, on the other hand, also complicates the integration that may affect execution speed and energy efficiency, resulting in some trade-offs in performance.

D. Markov and A. N. Romanov [19], in the year 2022, proposed the Zbb instruction set in the RISC-V architecture, which was designed for efficient bit manipulation. Implemented in Verilog, tested with Model Sim, it provides a 29.9% speed increase of a program and a 37.5% reduction in memory, but with the increased expense of logic elements and registers, thus limiting scalability for resource-constrained architectures. Another significant contribution was from Nair et al. in the year 2022 [20]. They did some work on Convolution Image filtering using a RISC-V-based Architecture. In this work, Shen and Liao [21], in the year 2024 explore ways to boost the performance of RISC-V processors using Macro-Op Fusion (MOF). MOF is a technique that merges multiple simple instructions into a single, more complex operation, allowing the processor to execute tasks more efficiently. The method also has a positive effect on energy efficiency by minimizing the number of instruction cycles as well as pipeline delays. The study features the use of MOF on actual RISC-V architectures, and also identifies methodologies to automatize

the detection of instruction fusion cases. As a result, MOF offers a possible way to improve the CPU performance without any software modifications. Mohr et al. [22] usher in the year 2024 to secure and make it more dependable as well as pursuing RISC-V open-source processors through hardware–software leakage contracts. Sensitive information is described in these contracts as potential leaks that can occur between hardware and software components. Researchers are trying to achieve by unintentional information flow detection and protection, a higher level of security in processor designs through the reduction of vulnerabilities to RISC-V Vector Extensions (RVV) power [23].

The investigation pertains to the effectiveness of the RISC-V Vector Extensions (RVV) in accelerating Artificial Neural Networks (ANN) algorithms. The authors demonstrate that the RVV may help increase parallel processing of neural networks, improving throughput and lowering power. The RISC-V architecture has the ability to execute vectorized instructions which makes it suitable for high-performance AI workloads because it is capable of processing several data array elements in parallel and giving the appearance of a single instruction per cycle. In this study, multiple neural networks performance improvements are evaluated, and RVV is demonstrated to be a promising way to accelerate AI processing on RISC-V chips. The outcomes may facilitate the development of energy efficient and AI-enabled RISC-V systems. Pieper and colleagues [24] demonstrate the use of an open-source RISC-V virtual prototype for modeling and simulating embedded systems.

As a result, developers can test and validate their hardware/software designs prior to implementation which would save time and cost implications. Moreover, performance optimization of embedded RISC-V systems is effectively done through the framework by simulating different scenarios of the sub-operations. The project's emphasis is on low power workloads. It serves as a flexible research platform with the aim to assist system designers of RISC-V architecture in evaluating performance, discovering bottlenecks to performance, and efficiency improvements. The project provides the RVCoreP, a RISC-V soft processor optimized for low latency and high throughput instructions through RISC-V architectures in a soft processor implementation [25]. The architecture, RVCoreP design is based on performing at least 5-stage instruction-level pipelining fairly efficiently thus enabling a small peripheral footprint for RISC-V performance. The RVCoreP architecture therefore formats for an efficient mechanism for executing RISC-V instructions specifically in a low power embedded application space while managing resource usage in architecture performance.

Its simple architecture for soft processors, this research really helps the practical deployment of RISC-V cores in environments where resources are limited, all the while providing very high-speed processing capabilities. This study aims at cutting down RISC-V code size using a combination of hardware/software approaches [26]. The authors investigate several techniques such as compiler optimizations, instruction set modifications, and hardware support to reduce the memory footprint while still having the same performance level. Such techniques have great importance for embedded systems and other places with limited resources where memory is a constraint. The execution of programs in the RISC-V processor becomes more efficient through the combination of hardware and software-level optimizations resulting in the reduction of storage requirements without the slowing down of processing. This research provides methods that are applicable to the generation of lightweight RISC-V cores for embedded applications, IoT devices, and low-power systems; hence, giving a good combination of efficiency and performance. Gamino del Río et al. [27] present a RISC-V processor architecture with capabilities for transparent tracing. Transparent tracing comes with the ability of developers to observe processor execution, fix software issues and perform system behaviour analysis without being a hindrance to normal operations. The authors incorporated hardware mechanisms to monitor instruction execution, memory access patterns, and pipeline events during operation. These features are so detailed that they can be used for performance tuning and error detection. This approach allows RISC-V systems to be verified and debugged very easily and quickly, especially for complex embedded applications. The proposed processor has the capability of non-intrusive monitoring, which not only guarantees the reliability of software applications but also provides a very minimal impact on the performance in terms of speed and resource utilization. Wu et al. [28] (2020) examine the concept of communal coprocessors within a RISC-V–based MPSoC to leverage Instruction Set architecture features for improving overall system-level efficiency. This is done by augmenting an existing core with shared coprocessors for task-specific use, which allows a system to achieve higher hardware resource efficiency with fewer hardware devices. This ultimately leads to a reduction in die area, less energy used, and greater throughput for multi-core RISC-V systems. The authors also illustrate that, even for quite complex SoC designs, a better hardware/software design methodology is feasible

with proper decomposition of tasks and using just the right number of instruction extensions to enable system-level efficiency. This approach will be desirable for embedded and high-performance computing systems that often have multiple processing units and need to share the same resource-enhancing coprocessors. Saiprathyusha and Chandrasekhar (2019) [29] present a RISC-V processor that implements user-defined instruction set hardware acceleration. Custom instructions can be created by the designer, and will in effect allow the processor to accelerate application-specific tasks, improving the usefulness for a broader range of tasks. The instruction hardware acceleration function can reduce execution times for compute critical operations and computation typically represented by heavy workloads. The proposed architecture also provides a design framework that can result in a system design that yields an acceptable level of design under manufacturing conditions and performance level. The benefits of this practice tend to be based upon energy improvement at an embedded system, IoT device, or wherever else the routing improvement alone can lead to a multiplicative benefit to energy reduction as compared to the optimal processing. Ko Stoffelen (2019) [30] performed a study in regards to RISC-V architectures for load with the least amount of energy, and memory. This study explored the possibilities for extending or modifying the open RISC-V instruction set ability to perform encryption, decryption, and hashing that would reduce the energy load, and ideally, without additional cost. This study may also utilize instructions specifically designed to provide efficiencies to cryptographic methods along with continuing with the combination of hardware/software co-design method intended for energy reduction. This study will focus on areas in regards to high sensitivity in energy which could be applied to an embedded system, IoT device or for applications in secure communications. RISC-V has other potential benefits which may include the performance aspects of real time energy performance improvement in relation to hardening applications, in addition to being energy efficient. The authors demonstrate that RISC-V processors can be a viable option for cryptographic applications with minimal architectural modification, suggesting strong performance and energy efficiency for security-centric computation. This paper offers a design automation methodology to quickly generate high quality optimized RISC-V processor cores from a functional instruction set specification. The author proposes automating the process of generating the processor cores from an abstract ISA description without losing the ability of the designers to quickly generate the optimized cores in advance with performance and correctness qualities known. This design methodology offers a good chance for a reduction in design time, as well as reducing the chances of errors and encouraging confidence in reliability, an important aspect to modern SoC design. During this process, it was found that the created processors allow different settings which makes them versatile enough to be used in different areas ranging from low-power chips to supercomputers. Thus, the present work plays a crucial role in RISC-V efficient personalization by making core generation and verification easier and resulting in a considerably quicker development of cycles without sacrificing good quality processor implementations. Liu et al. (2019) [31] presented a methodology for the rapid generation of high-quality RISC-V processors directly from functional instruction set specifications. Their work focuses on automating the processor design process by translating high-level ISA descriptions into efficient hardware implementations, thereby reducing manual design effort and development time. The proposed approach enables fast exploration of architectural design choices while maintaining competitive performance, area efficiency, and correctness. By supporting customization and extensibility, this methodology is particularly valuable for application-specific processor design, making it suitable for embedded systems, accelerators, and research-oriented RISC-V development.

Rapid RISC, a quick-fix customization mechanism for RISC-V chips, is presented by Donofrio and Leidel [32]. The methodology focuses on fast and efficient production of personalized RISC-V cores for definite usages. Rapid RISC accomplishes this by automating the configuration and code generation process, hence cutting down development time and effort while allowing performance and area optimization. It also incorporates hardware/software co-design, which gives designers the possibility to successfully tailor the processor architecture for the target workloads. The whole framework is very advantageous for embedded systems, AI accelerators, and special computing tasks, where application-specific RISC-V cores can offer improvements in terms of speed, energy costs, and a very flexible development platform. They have established a methodical structure for the examination of openness RISC-V processors regarding security vulnerabilities through their research without compromising the safety of hardware and software communications. This approach is particularly valuable for securing embedded and critical computing systems against potential attacks.

A number of studies have recently focused on better understanding how IoT intelligence has improved due to advances in IoT processors and how the RISC-V architecture has improved performance. One study presented a

framework for moving away from traditional IoT technologies and into the world of smart IoT by using semantic, cognitive and perceptual computing; this process will allow for more intelligent data processing, enabling better decision making [33]. In addition, a second study proposed a way to use artificial neural networks in order to recognize Spectre attacks, thereby providing a higher level of security within RISC-V processor-based systems [34]. Finally, the researchers described a five-stage, pipelined RISC-V core (RV32I ISA) and tested it to validate the increase in speed (efficiency) in embedded processing applications [35]. Overall, this body of work reflects the importance of intelligent computation, security and efficient processor design in the rapidly evolving IoT world today. By applying loop unrolling optimization techniques to a Network-on-Chip (NoC)-based multiprocessor architecture for parallel execution, they reported a significant improvement in operational performance. Although the suggested method shows how to effectively enable high-speed parallel computation, specific quantitative performance metrics were not given. Different methodologies which attain the Speed-up metric is as shown in **Table 1**.

Table 1. Existing Speed-Up Methods.

Speed-Up		
S.No	**Methodology**	**Increased by X Times**
1.	Cryptographic Hash Optimization [1]	10
2.	LU Decomposition Optimization [8]	338
3.	Cryptographic Performance [14]	8.6
4.	x264 Optimization [10]	9.8

There are many speed-up techniques that have already been put out to improve the computational efficiency in a variety of fields. Starting with a tremendous increase, the optimization of LU decomposition has improved the performance by 338 times. The optimization of cryptographic hash has obtained a speedup of 10 times. Also, the optimization of x264 has produced a 9.8-fold speedup and cryptographic performance improvements have produced an 8.6-fold speedup. These methods are an indication of key developments in the computational process optimization techniques and graphical representation as shown in **Figure 2**.

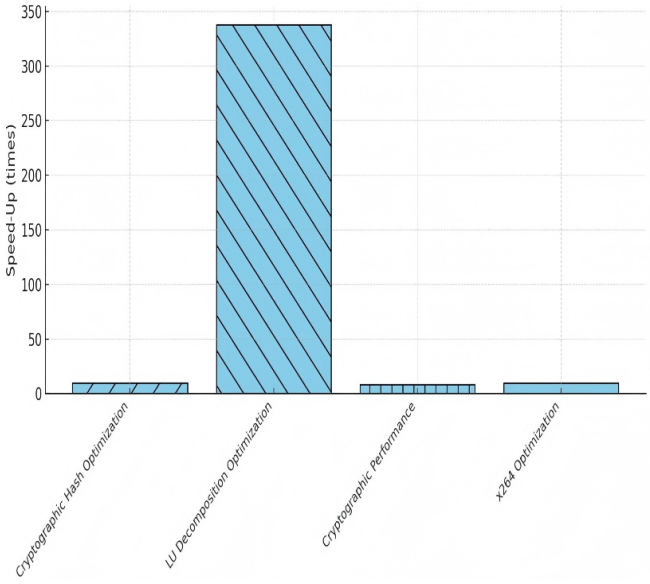

Figure 2. Graphical Representation of Speed-Up Methods.

Table 2 summarizes the power efficiency achieved by different techniques. The power efficiency achieved in the RISC-V ALU Optimization was 80%, while the Graph Computing Accelerator realized a massive 92.3% efficiency. These highlight key developments toward higher levels of optimization for both power consumption in computation and graphical representation, as illustrated in **Figure 3**.

Table 2. Existing Power Efficient Methods.

Power	
Methodology	**Power Efficient (%)**
RISC-V ALU Optimization [6]	80
Graph Computing Accelerator [11]	92.3

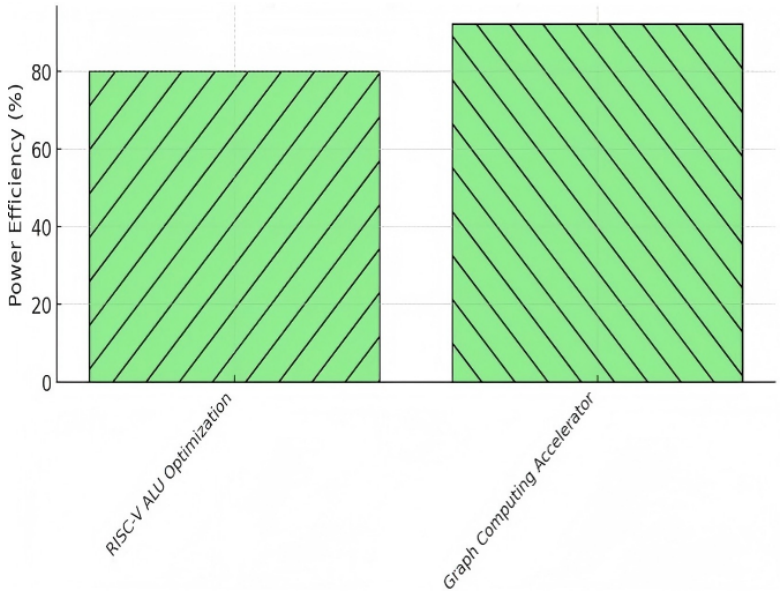

Figure 3. Graphical Representation of Power Efficiency Methods.

4. Conclusion and Future Work

The reviewed literature demonstrates significant advancements in RISC-V ISA customization, computer-aided design tools, and application-specific optimizations. Studies such as those by Juliette Pottier et al. on dynamic micro-decoding, SHORE's memory safety acceleration, and DELTA-V's automated ASIP design illustrate how modern frameworks enable rapid, efficient processor tailoring for diverse applications. These works emphasize improved performance, optimized instruction execution, and streamlined hardware/software co-design processes, which are crucial for both academic research and industrial deployments. Despite these improvements, several challenges persist, particularly in enhancing system robustness, achieving seamless architecture integration, and realizing effective hardware/software mapping. While open-source RISC-V frameworks, advanced compiler support, and AI-driven toolchains have made processor customization increasingly feasible, designing systems that balance flexibility, security, and performance remains complex. Future research must focus on interoperability across heterogeneous systems, security-enhanced architectures, and stronger adherence to co-design principles, ensuring that hardware and software components are developed cohesively. By addressing these challenges, designers can create next-generation computing platforms that are not only high-performing but also reliable, scalable, and adaptable to emerging workloads. It is anticipated that as RISC-V customization continues to advance, developers will be greatly empowered with cutting-edge tools and design techniques that can meet the increasing demands of contemporary embedded systems, artificial intelligence, and high-performance computing applications. Future studies will probably concentrate on intelligent and automated co-design frameworks that incorporate adaptive instruction set extensions, machine learning-driven optimization, and reconfigurable microarchitectures suited to workloads specific to particular applications. By facilitating smooth collaboration between academia and industry, shortening development cycles, and enhancing design predictability, the growing maturity of open-source RISC-V ecosystems will further spur innovation. Furthermore, it is expected that the combination of RISC-V with cutting-edge technologies like edge intelligence, heterogeneous computing platforms, and domain-specific accelerators will enable new levels of scalability, performance, and energy efficiency.

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable. This study did not involve human participants, animals, or any experimental procedures requiring ethical approval.

Informed Consent Statement

Not applicable. The study did not involve human subjects.

Data Availability Statement

No new datasets were generated or analyzed during the current study. Data sharing is not applicable to this article.

Acknowledgments

The authors would like to express their sincere gratitude to their respective institutions for providing the necessary facilities and support to carry out this research. The authors also acknowledge the valuable contributions of researchers whose published works formed the basis of this study.

Conflicts of Interest

The authors declare no conflict of interest.

AI Use Statement

The authors use artificial intelligence (AI)-assisted tools solely for language enhancement, grammar checking, and manuscript editing. All scientific content, analysis, interpretations, and conclusions presented in this manuscript were developed, verified, and approved by the authors. The authors take full responsibility for the accuracy, integrity, and originality of the work.

References

1. Zou, X.; Peng, Y.; Li, T.; et al. Accelerating Blake3 in RISC-V. In Proceedings of the 2023 2nd International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT), Xiamen, China, 4–7 August 2023. [CrossRef]
2. Choudhury, A.; Siddamal, S.V.; Mallidue, J. An optimized RISC-V processor with five stage pipelining using Tournament Branch Predictor for efficient performance. In Proceedings of the 2022 International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), Shivamogga, India, 14–15 October 2022. [CrossRef]
3. Dow, H.-K.; Li, T.; Miles, W.; et al. SHORE: Hardware/Software Method for Memory Safety Acceleration on RISC-V. In Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 5–9 December 2021. [CrossRef]
4. Yu, H.; Yuan, G.; Kong, D.; et al. An Optimized Implementation of Activation Instruction Based on RISC-V. *Electronics* **2023**, *12*, 1986. [CrossRef]
5. Raza, M.A.; Shahzad, I.; Anwar, H.; et al. An Optimum Design and Implementation of a 16-bit ALU on CADENCE Using RISC-V Architecture. In Proceedings of the 2023 IEEE International Conference on Emerging Trends in Engineering, Sciences and Technology (ICES&T), Bahawalpur, Pakistan, 9–11 January 2023. [CrossRef]
6. Karimullah, S.; Vishnuvardhan, D. Pin density technique for congestion estimation and reduction of optimized design during placement and routing. *Appl. Nanosci.* **2023**, *13*, 1819–1828. [CrossRef]
7. Bhargavi, M.B.; Harshith, G.S.S.; Parameswaran, S.; et al. Optimizing LU Decomposition with RISC-V Based Hardware Acceleration. In Proceedings of the 2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Knoxville, TN, USA, 1–3 July 2024. [CrossRef]

8. Xu, Y.; Yu, Z.; Tang, D.; et al. Towards Developing High Performance RISC-V Processors Using Agile Methodology. *IEEE Micro* **2023**, *43*, 98–106. [CrossRef]
9. Wang, J.; Wang, L.; Wang, P. Optimisation of x264 encoder acceleration based on RISC-V vector instructions. In Proceedings of the 2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS), Chengdu, China, 7–9 July 2023. [CrossRef]
10. Deng, J.; Ye, Z.; Zhou, K.; et al. Design of Software-Hardware Collaborative Graph Computing Accelerator Based on RISC-V. In Proceedings of the 2022 7th International Conference on Integrated Circuits and Microsystems (ICICM), Xi'an, China, 28–31 October 2022. [CrossRef]
11. You, C.-X.; Wang, Q.-T.; Zhong, H.; et al. RISC-V processor-based automatic access floating-point computing accelerated dataflow co-processor. In Proceedings of the 2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST), Guangzhou, China, 9–11 December 2022. [CrossRef]
12. Nişancı, G.; Flikkema, P.G.; Yalcin, T. Symmetric Cryptography on RISC-V: Performance Evaluation of Standardized Algorithms. *Cryptography* **2022**, *6*, 41. [CrossRef]
13. Lima, P.; Vieira, C.; Reis, J.; et al. Optimizing RISC-V ISA Usage by Sharing Coprocessors on MPSoC. In Proceedings of the 2020 IEEE Latin-American Test Symposium (LATS), Maceio, Brazil, 30 March–2 April 2020. [CrossRef]
14. Wu, J.; Tian, J.; Yang, G.; et al. Design of RISC-V heterogeneous multi-core SOC architecture for edge computing for power applications. In 2023 IEEE 6th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 24–26 February 2023. [CrossRef]
15. Hubert, B.; Hammami, O. Multi-objective Optimisation of RISC-V CV32A6 for ML application. In Proceedings of the 2023 IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Singapore, 18–21 December 2023. [CrossRef]
16. Dharsni, I.T.; Pande, K.S.; Panda, M.K. Optimized Hazard Free Pipelined Architecture Block for RV32I RISC-V Processor. In Proceedings of the 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 20–22 October 2022. [CrossRef]
17. An, H.; Kang, S.; Kanda, G.; et al. RISC-V Hardware Synthesizable Processor Design Test and Verification Using User-Friendly Desktop Application. *Webology* **2022**, *19*, 4597–4620. [CrossRef]
18. Anu Priya, D.; Vishnuvardhan, D.; Mamatha, G.; et al. VIP Development of SPI Controller for Open-Power Processor-Based Fabless SOC. In Proceedings of the 2nd International Conference on Cognitive and Intelligent Computing, Hyderabad, India, 27–28 December 2022. [CrossRef]
19. Markov, D.; Romanov, A.N. Implementation of the RISC-V Architecture with the Extended Zbb Instruction Set. In Proceedings of the 2022 International Ural Conference on Electrical Power Engineering (UralCon), Magnitogorsk, Russian Federation, 23–25 September 2022. [CrossRef]
20. Nair, M.K.A.; Gautam, V.V.; Revinipati, A.; et al. Implementation and Analysis of Convolution Image Filtering with RISC-V Based Architecture. In *Communications in Computer and Information Science*; Springer: Cham, Switzerland, 2022. [CrossRef]
21. Shen, J.-Y.; Liao, S.-W. Evaluating and Enhancing Performance through Macro-Op Fusion Optimization with RISC-V. In Proceedings of the 53rd International Conference on Parallel Processing, Gotland, Sweden, 12–15 August 2024. [CrossRef]
22. Mohr, G.; Guarnieri, M.; Reineke, J. Synthesizing Hardware-Software Leakage Contracts for RISC-V Open-Source Processors. *arXiv preprint* **2024**, *arXiv:2401.09383*. [CrossRef]
23. Rumyantsev, K.; Yakovlev, P.; Gorshkov, A.; et al. RISC-V RVV efficiency for ANN algorithms. *arXiv preprint* **2024**, *arXiv:2407.13326*. [CrossRef]
24. Pieper, P.; Herdt, V.; Drechsler, R. Advanced Embedded System Modeling and Simulation in an Open-Source RISC-V Virtual Prototype. *J. Low Power Electron. Appl.* **2022**, *12*, 52. [CrossRef]
25. Miyazaki, H.; Kanamori, T.; Islam, A.; et al. RVCoreP: An optimized RISC-V soft processor of five-stage pipelining. *IEICE Trans. Inf. Syst.* **2020**, *103*, 2494–2503. [CrossRef]
26. Perotti, M.; Schiavone, P.D.; Tagliavini, G.; et al. HW/SW approaches for RISC-V code size reduction. In Proceedings of the 4th Workshop on Computer Architecture Research with RISC-V (CARRV 2020), Online, 29 May 2020. [CrossRef]
27. Gamino del Río, I.; Martínez Hellín, A.; Polo, Ó.R.; et al. A RISC-V Processor Design for Transparent Tracing. *Electronics* **2020**, *9*, 1873. [CrossRef]
28. Wu, N.; Jiang, T.; Zhang, L.; et al. A Reconfigurable Convolutional Neural Network-Accelerated Coprocessor Based on RISC-V Instruction Set. *Electronics* **2020**, *9*, 1005. [CrossRef]
29. Saiprathyusha, P.; Chandrasekhar, C. Implementation of RISC-V Processor. *ITM Web Conf.* **2025**, *74*, 02006.

[CrossRef]
30. Stoffelen, K. Efficient Cryptography on the RISC-V Architecture. In *Progress in Cryptology – LATINCRYPT 2019*; Springer: Cham, Switzerland, 2019. [CrossRef]
31. Liu, G.; Primmer, J.; Zhang, Z. Rapid Generation of High-Quality RISC-V Processors from Functional Instruction Set Specifications. In Proceedings of the 56th Annual Design Automation Conference 2019, Las Vegas, NV, USA, 2–6 June 2019. [CrossRef]
32. Donofrio, D.; Leidel, J.D. Rapid RISC: Fast customization of RISC-V processors. In Proceedings of SPIE Defense + Security 2022, Orlando, FL, USA, 2022. [CrossRef]
33. Sheth, A. Internet of Things to Smart IoT Through Semantic, Cognitive, and Perceptual Computing. *IEEE Intell. Syst.* **2016**, *31*, 108–112. [CrossRef]
34. Le, A.-T.; Hoang, T.-T.; Dao, B.-A.; et al. Spectre attack detection with Neutral Network on RISC-V processor. In Proceedings of the 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 27 May–1 June 2022; pp. 2467–2471. [CrossRef]
35. Ahmed, S.; Harun-Ur-Rashid, A.B.M. Design, Implementation and Verification of Five Stage Pipeline RISC-V Core (RV32I ISA). In Proceedings of the 2025 International Conference on Electrical, Computer and Communication Engineering (ECCE), Chittagong, Bangladesh, 13–15 February 2025; pp. 1–6. [CrossRef]