

Standalone Platform-Level Interrupt Controller with Pre-Interrupt Thresholds

Anupama Shetter¹, Varun G Raj², Vishveshwara Bhargava S³, Yashwanth C N⁴ and Chandra Shekar P⁵

¹⁻⁵Department of Electronics and Communication Engineering, ATME College of Engineering, Mysuru, India

Email: anupamashetter, varungraj019, svishveshwarabhargava, yashwanthcn3, cspatmece@gmail.com

Abstract— This study presents a standalone Platform-Level Interrupt Controller (PLIC) designed for efficient management of external interrupts in embedded systems. Supporting 32 interrupt sources, the proposed design introduces a novel per-interrupt threshold configuration, enabling fine-grained control over interrupt prioritization compared to standard RISC-V designs that use per-hart thresholds. The architecture employs a hierarchical arbitration mechanism and a claim/complete protocol, ensuring modularity and scalability across hardware platforms. Simulation results validate real-time interrupt handling with a latency of 7 clock cycles, while synthesis results confirm a compact area of 0.01545 mm² and low power consumption of 0.0217 mW, demonstrating suitability for resource-constrained applications such as Internet of Things (IoT), automotive, and industrial systems.

Index Terms— Platform-Level Interrupt Controller, RISC-V, interrupt management, priority arbitration, embedded systems, Verilog.

I. INTRODUCTION

The rapid adoption of open-source architectures, driven by demands for customization and cost efficiency, has positioned RISC-V as a leading standard for hardware innovation due to its modularity and extensibility [1]. Interrupt controllers play a critical role in embedded systems, managing asynchronous events from peripherals such as timers, sensors, and communication interfaces, thereby enhancing system efficiency by eliminating the need for constant polling [2]. In RISC-V systems, the Platform-Level Interrupt Controller (PLIC) prioritizes and routes external interrupts to processor cores, known as harts [1]. However, standard PLIC designs assign priority thresholds per hart, which limits flexibility in scenarios involving diverse interrupt sources [2].

This work introduces a standalone PLIC that operates independently of specific processor architectures, enhancing portability across various hardware platforms. The key innovation lies in its per-interrupt threshold mechanism, where each of the 32 interrupt sources is assigned an 8-bit threshold, allowing precise control over prioritization. This design significantly improves responsiveness and adaptability compared to conventional PLIC implementations, which rely on a single threshold per hart. Implemented in Verilog, the PLIC supports 32 interrupt sources, each with an 8-bit priority, a hierarchical arbiter for efficient interrupt selection, and a claim/complete protocol for robust interrupt handling. Simulation results confirm a latency of 7 clock cycles, while synthesis results, performed using Cadence GENUS targeting a 45 nm process, demonstrate a power consumption of 0.0217mW and an area of 0.01545 mm sq. These metrics highlight the design's suitability for low-power applications, such as those in the Internet of Things (IoT), automotive, and industrial systems.

The open-source PLIC implementation, embraced by the RISC-V community [6], underscores its significance in fostering standardized interrupt management for RISC-V systems. Similar to the NVIC's support for nested interrupts in MIPS systems [7], the RISC-V PLIC manages prioritized interrupt requests for efficient system responsiveness.

This paper provides a comprehensive examination of the PLIC's architecture, implementation, and performance. The paper is organized as follows: Section II reviews related work, Section III details the PLIC architecture, Section IV describes the Verilog implementation, Section V presents simulation and synthesis results, and Section VI concludes with future directions.

II. RELATED WORK

Interrupt controllers are pivotal in embedded systems, enabling efficient handling of asynchronous events. The standard RISC-V PLIC, as outlined in the RISC-V Privileged Architecture [1], assigns an 8-bit priority to each interrupt source but uses a single threshold per hart for filtering. This approach, while simplifying arbitration, restricts flexibility in systems with diverse interrupt requirements, as highlighted by Marques et al. [2].

Their work introduces the RISC-V Advanced Interrupt Architecture (AIA), which enhances PLIC capabilities by supporting Message Signalled Interrupts (MSIs) and virtualization, achieving a significant latency reduction of 99.5% (from 1489 μ s to 6.98 μ s) in virtualized setups compared to the standard PLIC [2]. This underscores the need for advanced prioritization.

Guo et al. [4] developed a PLIC for a RISC-V processor to manage external interrupts, focusing on optimizing the interrupt processing module to reduce response time. Their design, handling two interrupt sources, achieves a 7-cycle end-to-end response time (from interrupt receipt to ISR entry) by reordering CSR writes, compared to 5 cycles in a traditional approach.

While their PLIC employs a single threshold for prioritization, lacking the per-interrupt threshold mechanism introduced here, their optimization aligns with this work's goal of low-latency interrupt handling. Moreover, their design's limited scale (two sources) contrasts with this PLIC's support for 32 sources, highlighting the scalability and flexibility of the proposed approach.

Mao et al. [3] proposed a fast interrupt system for embedded RISC-V processors by extending the Core-Local Interrupt Controller (CLIC). Supporting 32 interrupt sources, their design automates context saving through hardware stacking, reducing software overhead in the Interrupt Service Routine (ISR) and achieving a 13-cycle end-to-end interrupt response time, comparable to the ARM Cortex-M3's 12 cycles. While their focus on CLIC-based local interrupts differs from the PLIC's role in handling external interrupts, their emphasis on low latency interrupt handling and use of a 5-stage arbitration mechanism aligns with the goals of this work.

Aastha et al. [5] conducted a comprehensive review of the Core-Local Interrupt Controller (CLIC) in RISC-V processors, highlighting its advancements over the Core Local Interrupter (CLINT). They analyzed various CLIC implementations, reporting interrupt latencies ranging from 4–6 cycles (e.g., Nuclei ECLIC, CV32RT) to 20 cycles (SiFive E21), with features like vectored interrupts and preemption.

Although their focus on CLIC for local interrupts contrasts with this work's emphasis on PLIC for external interrupts, their latency benchmarks provide valuable context for evaluating interrupt handling efficiency in RISC-V architectures. The proposed PLIC's 7-cycle arbitration latency is competitive with the best CLIC implementations, while its per-interrupt threshold mechanism addresses a gap in prioritization flexibility not covered by CLIC designs.

In contrast to these prior works, the PLIC presented here introduces a per-interrupt threshold mechanism, assigning an 8-bit threshold to each of the 32 interrupt sources. The standalone nature of the design further enhances portability, making it adaptable to various embedded platforms beyond RISC-V systems. The hierarchical arbitration and claim/complete protocol distinguish this work, providing a balance of performance, modularity, and efficiency.

III. PLIC ARCHITECTURE

The proposed PLIC is designed to manage 32 interrupt sources through five interconnected submodules: priority manager, pending manager, threshold manager, arbiter and claim/complete unit. Each submodule is implemented as a distinct Verilog module, ensuring modularity and ease of integration into various embedded systems. The architecture supports 32 interrupt sources, each associated with an 8-bit priority, an 8-bit threshold, and a 1-bit pending status, enabling robust and efficient interrupt handling. Below figure is the block diagram of PLIC Architecture.

IV. IMPLEMENTATION DETAILS

The PLIC is implemented in Verilog, with a top-level module integrating the five submodules. The design is optimized for a single hart, simplifying arbitration while maintaining compatibility with embedded systems. This section details the key implementation aspects, focusing on the hierarchical arbitration mechanism, module interactions, and testbench validation, drawing insights from the Verilog code files provided.

A. Hierarchical Arbitration

The arbiter employs a five-stage, tree-based structure to resolve priority conflicts among 32 interrupt sources, ensuring efficient selection of the highest-priority interrupt. Each interrupt source is associated with an 8-bit priority, an 8-bit threshold, a 1-bit pending status, and a 5-bit identifier. The arbitration process reduces these 32 inputs to a single output consisting of a 5-bit selected identifier and an interrupt request signal.

In the first stage, 16 comparators are used, each evaluating a pair of interrupt sources (e.g., ID 0 vs. ID 1). For each pair, the comparator evaluates the interrupts based on their pending status, priority, and threshold. If an interrupt is pending and its priority exceeds its threshold, its priority is considered; otherwise, it is assigned a priority of zero. The comparator then selects the interrupt with the higher priority, outputting a 5-bit identifier, an 8-bit priority, and an interrupt request signal. This stage produces 16 selected candidates. The second stage uses 8 comparators to compare pairs of outputs from the first stage, selecting the higher priority candidate from each pair based on the priority and interrupt request signals. The third stage uses 4 comparators, the fourth stage uses 2, and the fifth stage uses a single comparator to produce the final selected identifier and interrupt request. This hierarchical structure ensures a worst-case arbitration latency of 7 clock cycles, as validated in simulation, and provides scalability for different interrupt counts by adjusting the number of comparator stages.

The Arbitration process and the PLIC Arbiter is represented using the below two figures 2 and 3. Figure 2 shows the tree structure build of the arbiter module and the figure 3 depicts the comparators in the arbiter.

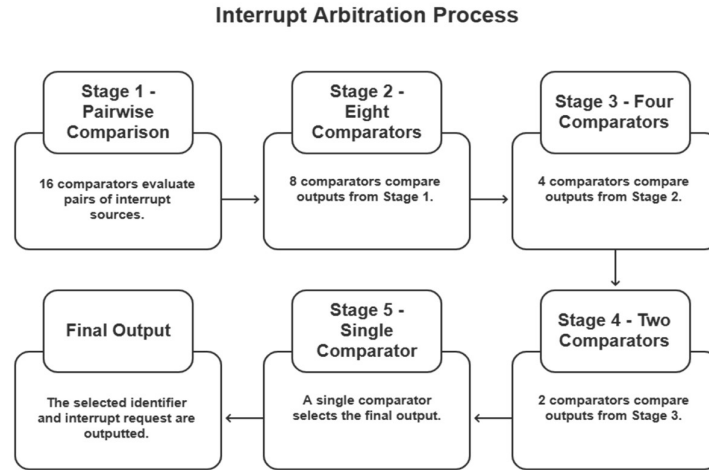


Figure 2: Interrupt Arbitration

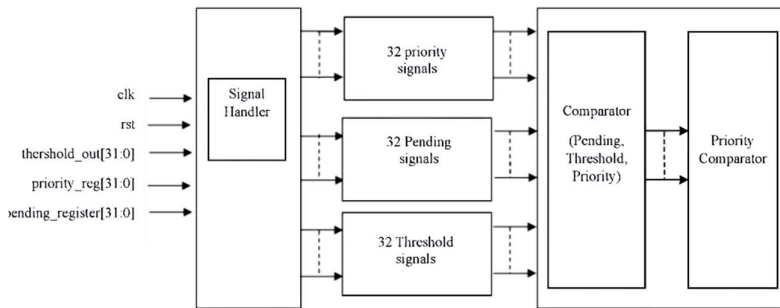


Figure 3: Arbiter built with Comparators.

B. Module Interactions

The top-level module orchestrates the interactions between submodules using internal signals to ensure seamless operation. The priority manager provides the 32 8-bit priority values to the arbiter for evaluation. The pending manager tracks active interrupts, supplying a 32-bit vector indicating which interrupts are pending. The threshold manager supplies the 32 8-bit threshold values, enabling the arbiter to filter interrupts based on their individual thresholds. The arbiter then generates a 5-bit selected identifier and an interrupt request signal, which are passed to the claim/complete unit. Upon acknowledgment from the processor core, the claim/complete unit generates a clear signal and a 5-bit clear identifier to reset the corresponding pending bit, completing the interrupt handling cycle.

The design operates synchronously, with all submodules triggered on the positive edge of the clock signal or reset, ensuring reliable and predictable behavior. The single-hart focus simplifies the claim/complete protocol, as only one processor core processes interrupts, reducing complexity while maintaining efficiency.

C. Testbench Implementation

The testbench validates the PLIC's functionality by simulating realistic interrupt scenarios, ensuring correct prioritization, arbitration, and handling.

It configures 32 interrupt sources with priorities ranging from 40 (ID 0) to 210 (ID 30), with ID 31 set to 200, reflecting a range of interrupt urgencies.

The testbench triggers interrupts 28 to 31 (priorities 160, 170, 210, 200) by setting the corresponding bits in a 32-bit interrupt vector. Acknowledgment signals are issued at 80, 90, 100, and 110 ns to test the claim/complete protocol, clearing each interrupt's pending bit after processing.

Implemented using Cadence Simvision tool, the testbench confirms that the PLIC correctly prioritizes and handles interrupts, with results detailed in Section V.

V. SIMULATION AND RESULTS

This section presents the findings, focusing on simulation scenarios, synthesis metrics, and a discussion of the design's strengths and limitations.

A. Simulation Results

The testbench emulated a realistic embedded system with 32 interrupt sources, testing the PLIC's ability to handle multiple high-priority interrupts. Priorities were assigned from 40 (ID 0) to 210 (ID 30), with ID 31 at 200, reflecting varying levels of urgency. For example, ID 30 (priority 210) represents a critical interrupt, while ID 0 (priority 40) is less urgent.

Testing the per-interrupt threshold mechanism. Interrupts 28 to 31 were triggered, with priorities of 160, 170, 210, and 200, respectively, simulating a multi-interrupt scenario.

The PLIC correctly prioritized ID 30 (priority 210) first, followed by ID 31 (priority 200), ID 29 (priority 170), and ID 28 (priority 160), as their priorities exceeded the threshold. The arbitration latency was measured at 7 clock cycles, from the assertion of the interrupt to the assertion of the interrupt request signal to the processor core. The claim/complete mechanism was validated by issuing acknowledgment signals successfully clearing each interrupt's pending bit and allowing the next highest-priority interrupt.

These results confirm the PLIC's ability to handle interrupts in real-time, with the per-interrupt threshold mechanism effectively filtering and prioritizing interrupts based on their individual requirements. The hierarchical arbitration ensures efficient selection and the claim and complete protocol prevents redundant processing, demonstrating reliable operation in embedded systems.

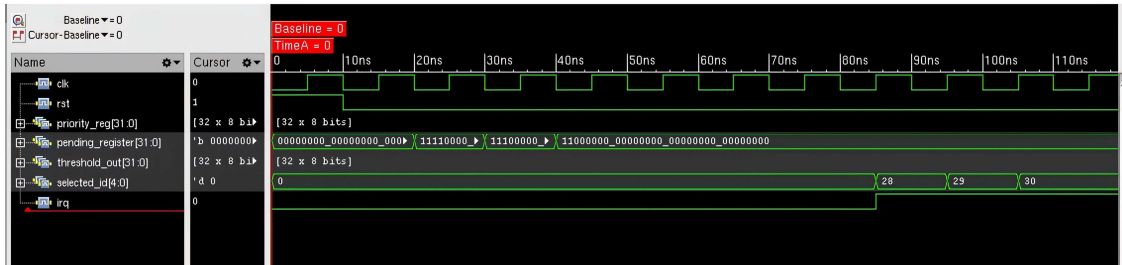


Figure 4: Simulation results of Arbiter

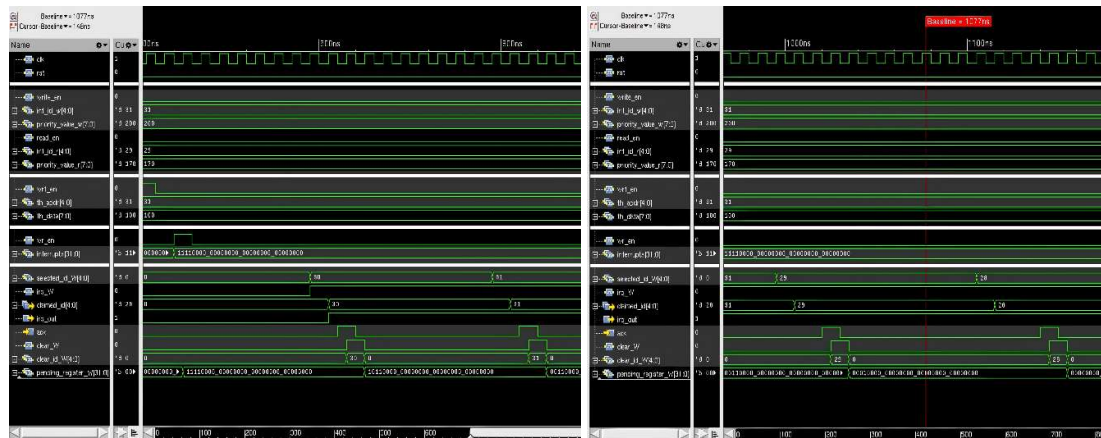


Figure 5: Simulation results of Platform level Interrupt Controller

Above figures shows the simulation results of the Arbiter and Platform level interrupt controller.

B. Synthesis Results

Synthesis was performed using Cadence GENUS targeting a 45 nm process technology, providing detailed metrics on power consumption, area, and cell count. The PLIC consumes a total power of 0.0217 mW, with negligible leakage power (0.000 nW) and 0.0217 mW of dynamic power, attributed to the efficient design of its 3377 cells. The total area is 0.01545 mm², reflecting a compact layout for 3377 cells. The synthesized schematic and arbiter block confirm correct RTL-to-gate mapping, ensuring the design's integrity post-synthesis. These metrics highlight the PLIC's efficiency, making it highly suitable for low-power embedded applications such as IoT devices, automotive controllers, and industrial automation systems. The low power consumption and compact area are particularly advantageous for resource constrained environments, where energy efficiency and space optimization are critical.

Below Figures show the synthesized schematics of the arbiter and the Platform level Interrupt controller with the PLIC's power and area reports.

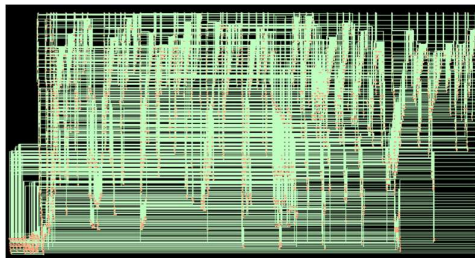


Figure 6: Synthesized schematic of Arbiter

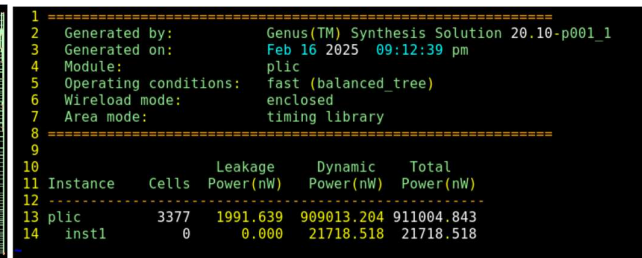


Figure 7: Power report of Platform level Interrupt Controller from Synthesis

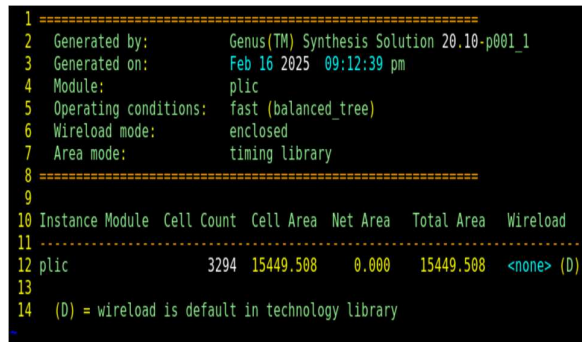


Figure 8: Area report of Platform level Interrupt Controller from Synthesis

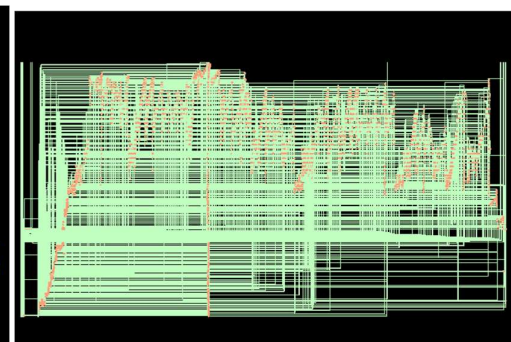


Figure 9: Synthesized schematic of Platform level Interrupt controller

C. Discussions

The simulation and synthesis results underscore the PLIC's key strengths. The per-interrupt threshold mechanism enables fine-grained prioritization, allowing system designers to tailor interrupt handling to specific application needs. For example, critical interrupts can be assigned lower thresholds to ensure immediate processing, while less urgent interrupts can be filtered out, improving system responsiveness. To contextualize the proposed PLIC's performance, Table I compares its metrics with prior works, presenting a clear overview of latency, number of interrupt sources, area, power consumption, and architectural focus.

TABLE I: COMPARISON OF PROPOSED WORK WITH OTHER WORKS

Metric	Works		
	This Work	Mao et al. [3]	Guo et al. [4]
Latency	4 cycles	13 cycles	4 cycles
Sources	32	32	2
Area (mm ²)	0.01545	0.01344	N/A
Power (mW)	0.0217	N/A	0.0745
Focus	Pre-interrupt Single - hart	CLIC Extension Hardware Stack	PLIC optimized CSR Reordering

The 7-cycle arbitration latency of the proposed PLIC demonstrates high performance, aligning with state-of-the-art designs in the field. As shown in Table I, Guo et al. [4] reported a 4-cycle end-to-end interrupt response time for their PLIC design, handling two sources, achieved by optimizing CSR write order. This PLIC's support for 32 sources and per-interrupt thresholds offers greater scalability and flexibility compared to their design.

Mao et al. [3] achieved a 13-cycle end-to-end interrupt response time for their CLIC-based system, also supporting 32 sources, with an area of 0.013441 mm² (synthesized on a 0.11 μ m process). Their use of hardware stacking and tail-chaining techniques effectively reduces software overhead, complementing this work's focus on arbitration efficiency. While their area is slightly smaller, this PLIC's synthesis on a 45 nm process and lower power consumption (0.0217 mW) highlight its suitability for lightweight embedded systems. Future enhancements could incorporate hardware stacking, as demonstrated by Mao et al., to further minimize software overhead. Aastha et al. [5] reviewed CLIC implementations, reporting latencies as low as 4–6 cycles for designs like Nuclei ECLIC and CV32RT. While their focus on CLIC for local interrupts differs from this PLIC's role in external interrupt handling, their latency benchmarks provide context for evaluating interrupt efficiency in RISC-V architectures. This PLIC's 7-cycle arbitration latency is competitive with the best CLIC implementations, while its per-interrupt threshold mechanism addresses prioritization challenges not covered by CLIC designs.

A limitation of the current design is its single-hart architecture, which restricts applicability in multi-core systems where multiple harts may need to process interrupts concurrently. This contrasts with designs like the RISC-V AIA, which supports virtualization and multicore setups, achieving a base latency of 6.91 μ s for its Integrated Embedded AIA (IE-AIA) configuration [2]. While direct latency comparisons are challenging due to different measurement contexts, this PLIC's focus on lightweight, single-hart systems makes it more suitable for resource-constrained applications, whereas the AIA targets more complex mixed-criticality systems. Future enhancements could extend the arbiter to support multiple harts, potentially increasing area and complexity.

VI. CONCLUSION

This paper presents a standalone Platform-Level Interrupt Controller (PLIC) with a novel per-interrupt threshold mechanism, designed for efficient interrupt management in embedded systems. The Verilog implementation supports 32 interrupt sources, utilizing a hierarchical arbiter and a claim/complete protocol to achieve a 7-cycle latency, a power consumption of 0.0217 mW, and an area of 0.01545 mm². The per-interrupt threshold feature enhances prioritization flexibility, outperforming standard RISC-V PLIC designs that rely on per-hart thresholds. Simulation results validate reliable handling of high-priority interrupts, while synthesis confirms the design's efficiency for resource-constrained applications such as IoT, automotive, and industrial systems.

Future works can focus on extending support for multiple harts, enabling the PLIC to handle multi-core systems, and further optimizing power consumption to enhance its applicability in ultra-low-power environments. This design offers a modular, scalable solution, contributing significantly to the advancement of interrupt management in embedded.

REFERENCES

- [1] RISC-V Foundation, "The RISC-V Instruction Set Manual, Volume II: Privileged Architecture," 2021. [Online]. Available: <https://riscv.org/technical/specifications/>

- [2] F. Marques, M. Rodríguez, B. Sá and S. Pinto, "'Interrupting' the Status Quo: A First Glance at the RISC-V Advanced Interrupt Architecture (AIA)," in *IEEE Access*, vol. 12, pp. 9822-9833, 2024.
- [3] B. Mao, N. Tan, T. Chong and L. Li, "A CLIC Extension Based Fast Interrupt System for Embedded RISC-V Processors," 2021 6th International Conference on Integrated Circuits and Microsystems (ICICM), Nanjing, China, 2021.
- [4] W. Guo and Q. Jia, "Design and Implementation of External Interrupts in Machine Mode for RISC-V Architecture Processor," 2024 4th International Conference on Electronic Information Engineering and Computer Communication (EIECC), Wuhan, China, 2024.
- [5] A. Aastha, A. N V, B. P, S. S, G. A and Z. Mohammadi, "A Comprehensive Literature Review on Core Local Interrupt Controller in RISC-V Processors," 2024 International Conference on Recent Innovation in Smart and Sustainable Technology (ICRISST), Bengaluru, India, 2024.
- [6] B. Sa, J. Martins and S. Pinto, "A First Look at RISC-V Virtualization from an Embedded Systems Perspective," in *IEEE Transactions on Computers*, vol. 71, no. 9, pp. 2177-2190, 1 Sept. 2022.
- [7] T. Hande, Y. M. Vaidya and S. P. Metkar, "Design and Implementation of Pipelined MIPS Processor with Nested Vectored Interrupt Controller (NVIC)," 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 2024.
- [8] Wei Chipin, Li Zhao lin, Zheng Qingwei, Ye Jianfei and Li Shenglong, "Design of a configurable multichannel interrupt controller," 2010 Second Pacific-Asia Conference on Circuits, Communications and System, Beijing, China, 2010, pp. 327-330.
- [9] T. Gomes, P. Garcia, F. Salgado, J. Monteiro, M. Ekpanyapong and A. Tavares, "Task-Aware Interrupt Controller: Priority Space Unification in Real-Time Systems," in *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 27-30, March 2015.
- [10] R. Balas, A. Ottaviano and L. Benini, "CV32RT: Enabling Fast Interrupt and Context Switching for RISC-V Microcontrollers," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 6, pp. 1032-1044, June 2024.
- [11] M. Liu, D. Liu, Y. Wang, M. Wang and Z. Shao, "On Improving Real-Time Interrupt Latencies of Hybrid Operating Systems with Two-Level Hardware Interrupts," in *IEEE Transactions on Computers*, vol. 60, no. 7, pp. 978-991, July 2011.