RISC-V SoC Design for IoT: A Survey of Open-Source Cores and Applications

Atharva Bagul
Sandip University, India
bagul.atharva.manoj@gmail.com

Anand Singh Rajawat
Sandip University, India

S.B. Goyal
City University, Malaysia

Abstract: While innovation can lead to groundbreaking products, it also typically entails higher costs due to the need for advanced design, production processes, and licensing fees for critical technologies. Therefore, RISC-V is crucial for cost-effective applications as a practical and robust instruction set architecture (ISA). RISC-V is a free-available and open-source instruction set architecture, allowing you to create CPU architectures that are not subject to any intellectual property rights. Due to the growing demand for IoT devices, it is more practical to create boards RISC-V specifically designed for IoT applications. Although significant progress has been made in creating open source CPU cores, more emphasis must still be placed on special abilities for IoT applications. Most open-source cores have inadequate documentation or are designed for specific purposes. This analysis focuses on multiple open-source CPUs and System-on-Chip (SoC) designs that leverage the RISC-V architecture, examining their relevance and effectiveness in IoT environments. The study further outlines a RISC-V SoC specifically designed for IoT applications and investigates the primary features of a suitable open-source IoT node, also identifying the need for ongoing R&D to support the development of specialized protocols.

Keywords: *IoT (Internet of Things), Power Efficiency*, *Low-Power Design, Instruction Set Architecture (ISA), RISC-V.*

I. INTRODUCTION

While the RISC-V architecture has enabled flexible and cost-effective CPU design, existing cores, such as PicoRV32 and RavenSoC, lack specific features essential for IoT applications, including low-latency power management, real-time processing, and enhanced cryptographic support. This work proposes a customized SoC design, addressing these deficiencies with a focus on IoT-specific needs, such as low-power operation, modular scalability, and robust security features.

The Internet of Things (IoT) has led to a substantial transformation in several aspects of human life, including work, living, and entertainment. The increasing number of IoT devices is rapidly growing in both consumer and industrial sectors and is estimated to reach a global cumulative count of around 75 billion units by 2025. The data produced by these Internet of Things (IoT) devices would be overwhelming and consequently necessitate a processor to effectively handle it. ARM, x86, and RISC-V are the current options for CPU development. However, x86 and ARM ISAs are not in the public domain and necessitate a substantial licensing fee while being extremely complex. This presents a challenge for small organizations and individuals engaged in the development of a custom microprocessor that is exclusively designed for IoT applications.[18]

Open-source hardware initiatives such as SHAKTI processors have demonstrated the potential for cost-effective and scalable solutions in processor design [13]. Considering the open standard nature of the RISC-V ISA, it provides an enticing alternative for designing custom IoT oriented chips without the hurdles of entry present in ARM and x86 ISA's. RISC-V is an instruction set architecture (ISA) based on reduced instruction set principles and was established as a research project at the University of California, Berkeley in 2010[3]. It is now maintained by SiFive inc. that handles all repositories linked with the RISC-V ISA and its development is sustained by its open-source nature[4]. Like x86, it supports both 32-bit and 64-bit and even 128-bit address variants but since its impracticality in the present CPU addressing needs, the 128-bit addressing version is not employed in the mainstream development of RISC-V CPUs. The RISC-V ISA includes a frozen basic instruction set known as RV32I (for 32-bit processors). Based on the requirements and design criteria, we can add particular instructions such as compressed (C), floating point (P), double precision floating point (D), vector calculations (V), and many others for various applications. Developers with minimal knowledge of computer architecture do not have to study about the RISC-V's internal architecture and emphasize on the true purpose of the project due to their extensions-based ISA [2][4][14].

While the RISC-V architecture has enabled flexible and cost-effective CPU design, existing cores, such as PicoRV32 and RavenSoC, lack specific features essential for IoT applications, including low-latency power management, real-time processing, and enhanced cryptographic support. This work proposes a customized SoC design, addressing these deficiencies with a focus on IoT-specific needs, such as low-power operation, modular scalability, and robust security features.

A customized CPU core that is ideal for deployment as an Internet of Things (IoT) node is provided by this study, which also examines the existing RISC-V implementations. The proposed core includes several key elements:

1) Scalability and Simplicity: A modular RISC-V core architecture that strikes a compromise between resource efficiency and performance, allowing for simple modification and flexibility to a range of Internet of Things applications. Additionally, dynamic clock gating is supported by the architecture, which aids in efficient power consumption management.
2) Versatile I/O: To interface with various external devices and sensors, the design has 64 general-purpose input/output (GPIO) pins with multi-state capability. To improve communication capabilities, a UART with DMA capability is also included.
3) The RV64GC instruction set, which consists of both compressed and general-purpose instructions, is implemented by this robust instruction set. The core is appropriate for a variety of applications due to its extensive instruction set, which facilitates the completion of a broad range of activities.
4) C Language Compatibility: The GNU RISC-V toolchain fully supports the compilation and running of C code. Because of this interoperability, developers can more easily create and launch apps on the IoT node by utilizing commonly used embedded systems development techniques.

This study's main goals are to analyze and assess current open-source RISC-V implementations, with an emphasis on their suitability for Internet of Things applications in terms of power consumption, form factor, and peripheral support; propose a modular RISC-V System-on-Chip (SoC) design that incorporates key IoT-centric features like enhanced UART with DMA support, GPIO connectivity, and dynamic clock gating, while balancing performance and resource efficiency; and address the difficulties of developing cost-effective IoT hardware by utilizing RISC-V ISA's open-standard status, which allows customization without the licensing restrictions found in proprietary architectures like ARM and x86.

A thorough summary of the open-source RISC-V-based CPUs that are currently available is given in Section 2. Section 3 examines their prospective uses in the Internet of Things industry as well as their individual strengths and weaknesses. A unique SoC design based on the RISC-V architecture that is especially suited for Internet of Things applications is proposed in Section 4. Section 5 examines the social implications of an openly available RISC-V core for IoT. Section 6 concludes the discussion and suggests possible directions for further study and advancement in this field.

II. RELATED WORK

A. PicoRV32

The PicoRV32 is a small RISC-V CPU core developed for utilization in ASICs and FPGAs. Its tiny size and high performance make it well-suited for different embedded applications[7]. Three bus configurations are available in the core for a range of applications. To guarantee compatibility with various peripherals, the picorv32_wb configuration makes use of a Wishbone master interface. An AXI4-lite interface, which is frequently utilized in system-on-chip (SoC) designs, is incorporated into the picorv32_axi configuration. Finally, a straightforward bus multiplexer interface is used in the basic setup, enabling direct system integration.

The basic PicoRV32 CPU includes specific technical features. It integrates 32KB of on-chip random access memory (RAM) for data storage and processing. The processor supports the RV32IMC instruction set, offering essential functionality for embedded systems. Its design provides flexibility by enabling users to adjust CPU features according to application needs. For memory interfacing, it includes options such as the multiplexer (mux) and AXI4 standards, ensuring compatibility with various systems.

For Internet of Things applications, the PicoRV32 is a low-power, small, and open-source RISC-V CPU core. By using less chip space and power, it is well suited for the resource-constrained nature of IoT devices, which require between 2,000 and 4,000 LUTs (Lookup Tables) in an FPGA system. With an average power consumption of roughly 100 µW/MHz in a 65nm CMOS process, its design places a strong emphasis on energy conservation. This is crucial for battery-powered IoT devices to guarantee extended operation without frequent battery replacement or recharging.

PicoRV32's versatility has made it a popular choice for incorporating custom CPUs into a variety of IoT applications.[16] Its open source openness and versatility have even led to its use as the foundation for a large number of RISC-V-based system-on-chip (SoC) designs. The official PicoRV32 repository provides a complete SoC reference design that includes critical components such as a GPIO controller, SPI flash controller, SRAM, and UART. This reference design also allows you to add customized blocks, such as extra communication protocol controllers or memory expansions, to fulfill unique application needs. PicoRV32 has been widely employed throughout the years in several IoT and embedded system designs, exhibiting its versatility and efficiency in handling the specific difficulties and opportunities given by these applications.

B. MicroRV32

MicroRV32, commonly referred to as RV32, is an open-source RISC-V platform primarily designed for educational and research purposes.[8] It integrates a 32-bit RISC-V core with various peripherals through a generic bus system. The platform is implemented using the latest version of SpinalHDL and is capable of running the FreeRTOS operating system. The top-level design incorporates the RV32I core, which interacts with peripherals such as memory, GPIO, and UART via a straightforward bus interface[5]. The core communicates with other peripherals through an interface defined by address, command, and data signals. The design employs a valid-ready handshake bus protocol where the bus master (CPU core) signals the bus slaves (peripherals) of available payload by asserting a valid signal. At the top level, peripherals are memory-mapped, with transaction packets routed to the appropriate peripheral based on the memory address. Peripherals respond to transaction requests after a single clock cycle, and if a peripheral fails to respond within this timeframe, the CPU will stall

The architecture's performance for various C++ programs, including Fibonacci and greatest common divisor (GCD) calculations, is evaluated using an RTL simulator. Furthermore, the complete System-on-Chip (SoC) has been tested on the latest Lattice Semiconductor HX8K Development Board, achieving a maximum clock frequency (fmax) of 28.61 MHz with 55% device utilization. The extent to which Block RAM cells are utilized, reaching 78% on the FPGA, varies depending on the program employed for memory configuration.

As MicroRV-like CPUs are deployed in real-world scenarios, they are expected to find diverse applications. MicroRV's minimalist architecture makes it particularly effective for demonstration purposes and educational use, with considerable potential for instructional applications. The architecture's simplicity allows for easy integration into low-power IoT applications, but its limited peripheral support remains a drawback.[17]

C. RavenSoC

The RavenSoC is a flexible System-on-Chip (SoC) built around the PicoRV RISC-V core, invented by Clifford Wolf. While the PicoRV core provides a foundational platform for testing and experimentation, the RavenSoC extends upon its capabilities to offer a more comprehensive and practical solution for numerous applications.

RavenSoC extends the capabilities of the PicoRV32 CPU core with a comprehensive range of I/O options, including GPIO and SPI memory interfaces[11]. This core has been successfully integrated into many FPGA platforms, exhibiting its compatibility and adaptability. The RavenSoC is distinguished by its open-source nature, enabling a transparent and adaptable platform for developers.

In terms of memory resources, the RavenSoC features a sizable on-chip 32x1024 SRAM, sufficient for handling a wide range of simple applications. Despite its relatively modest clock rate of 100 MHz, the core's efficient ISA and low overhead allow it to achieve remarkable performance.

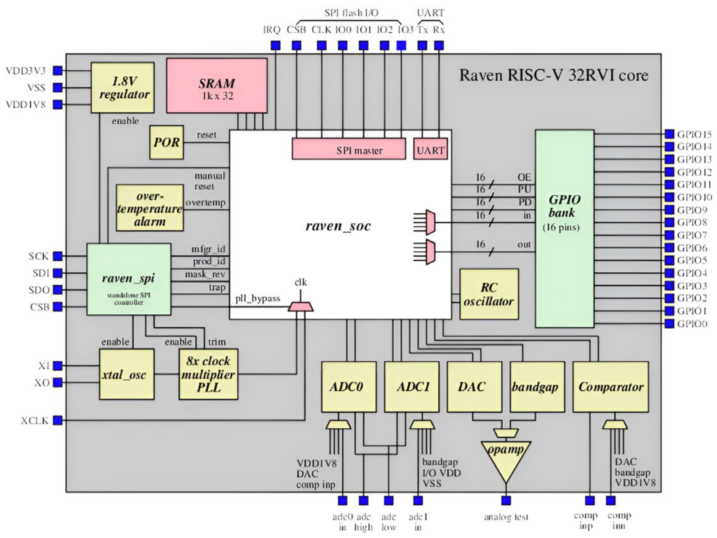

Fig.1. RavenSoC Block Diagram

The RavenSoC has a rich range of I/O capabilities, as seen in Figure 1. A significant feature is its GPIO bank, including 16 pins that can be set for input, output, or interrupt operations. This enables seamless connection with numerous Arduino and Raspberry Pi modules, extending the SoC's versatility.

For program memory, the RavenSoC has a separate SPI interface, allowing users to load programs into the core. Additionally, it features SPI memory, which can be utilized for interim storage or temporary program instructions. To ease analog-to-digital and digital-to-analog conversions, the SoC features two ADCs and one DAC.

Figure 1. shows The RavenSoC incorporates a UART controller, providing direct connection via a typical UART interface with other modules. Moreover, it also incorporates an over-temperature alert, a valuable safety feature typically ignored in open-source versions[20]. This method helps minimize overheating, boosting the SoC's reliability and lifetime.

D. UltraEmbedded RISC-V (biRISC-V)

This 32-bit RISC-V CPU core, available under Apache License, provides open-source communities with a superscalar (dual-issue) in-order 6 or 7 stage pipeline. It supports RISC-V's integer (I), multiplication and division (M), and CSR instructions (Z) extensions (RV32IMZicsr).

UltraEmbedded's biRISC-V core implements a six-stage pipeline for optimized performance in resource-constrained environments[15]. The pipeline begins with Instruction Fetch, which reads the next instruction from memory using the program counter (PC). This is followed by Instruction Decode, which decodes the fetched instruction to determine the operation to be performed and the operands required. The Register Fetch stage then reads the necessary operands from the register file. In the Execute stage, the instruction is executed using the ALU or other functional units. The Memory Access stage reads or writes memory for instructions that need memory access. The outcome of the instruction is then written back to the register file by the Writeback stage. The core's inadequate peripheral support restricts further IoT usage, even though it is compatible with Linux [19]. The core is intended to efficiently boot a basic open source operating system.

III. Proposed RISC-V SoC for IoT Applications

Our proposed System-on-Chip (SoC) design leverages the RISC-V Instruction Set Architecture (ISA) to build a solution optimized for the unique demands of Internet of Things (IoT) environments. This design focuses on integrating sensors efficiently, managing power well, and being able to grow. It meets important IoT requirements like using little power, working with a wide range of peripherals, and keeping data safe. The following sections provide an extensive review of its architecture, power efficiency mechanisms, and functional specifications.

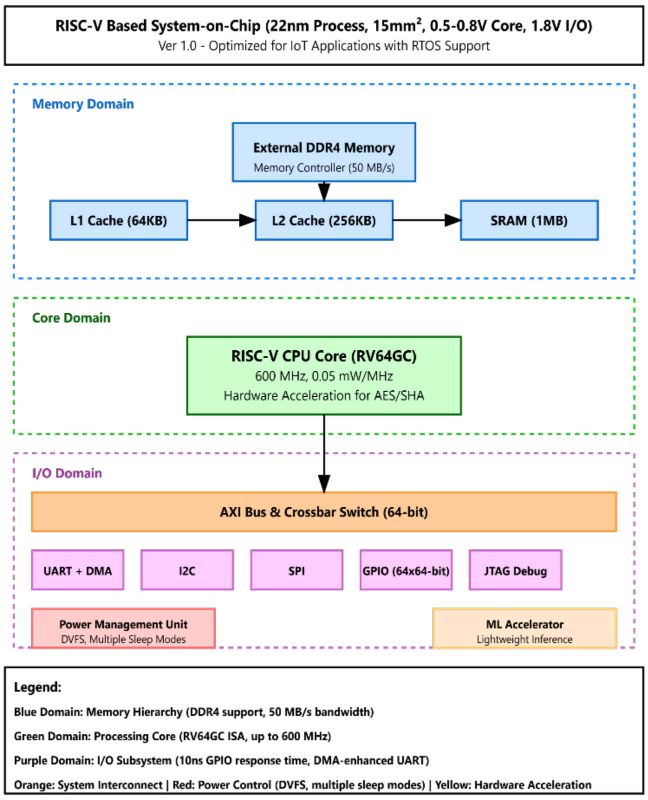
RISC-V Based System-on-Chip (22nm Process, 15mm², 0.5-0.8V Core, 1.8V I/O)
Ver 1.0 - Optimized for IoT Applications with RTOS Support
Memory Domain
External DDR4 Memory
Memory Controller (50 MB/s)
L1 Cache (64KB)
L2 Cache (256KB)
SRAM (1MB)
Core Domain
RISC-V CPU Core (RV64GC)
600 MHz, 0.05 mW/MHz
Hardware Acceleration for AES/SHA
I/O Domain
AXI Bus & Crossbar Switch (64-bit)
UART + DMA
I2C
SPI
GPIO (64x64-bit)
JTAG Debug
Power Management Unit
DVFS, Multiple Sleep Modes
ML Accelerator
Lightweight Inference
Legend:
Blue Domain: Memory Hierarchy (DDR4 support, 50 MB/s bandwidth)
Green Domain: Processing Core (RV64GC ISA, up to 600 MHz)
Purple Domain: I/O Subsystem (10ns GPIO response time, DMA-enhanced UART)
Orange: System Interconnect | Red: Power Control (DVFS, multiple sleep modes) | Yellow: Hardware Acceleration

Fig.2. *Block diagram of the proposed 22nm RISC-V SoC architecture, having hierarchical memory organization, hardware-accelerated security, and IoT-optimized interfaces. The design emphasizes power efficiency through domain partitioning and specialized management units, addressing IoT applications with RTOS support.*

A. SoC Architecture and Communication Interfaces

The SoC's architecture supports seamless communication with sensors and external devices through interfaces like UART, SPI, I2C, and GPIO, allowing compatibility with a broad range of IoT protocols and peripherals[16]. Each component links to the central CPU via an Advanced eXtensible Interface (AXI) bus and a crossbar switch, ensuring high data throughput and scalability. This configurable setup simplifies the addition of peripherals such as I3C, a better alternative to traditional I2C in terms of data rate and power efficiency.

The following steps are involved in the compiling and loading process:

- Compilation with gcc generates an assembly file (.s format).
- The CMake-based build system transfers this file to the assembler, creating a .out file.
- The linker produces a .elf file, further

processed by objcopy to generate the final .hex file.

- The program memory loads the .hex file, where the preprocessor fetches and executes instructions, enabling the SoC to run optimized C code for IoT applications.

B. Power Efficiency and Performance Optimization

Our proposed SoC is designed to meet the performance, energy efficiency, and flexibility required in IoT applications. To ensure optimal energy efficiency and functionality, the core frequency is configurable up to 600 MHz[19], balancing power consumption and processing speed for typical IoT tasks. This frequency is paired with an approximate power consumption of 0.05 mW/MHz in active mode at a core voltage range of 0.5V - 0.8V, achievable through advanced power management techniques[17]. In sleep and deep-sleep modes, the SoC achieves 1 µW and 0.05 µW, respectively, leveraging leakage suppression and optimized power gating, which is essential for extended battery life in IoT applications where devices may remain idle for extended periods[10].

The memory bandwidth is maintained at 50 MB/s, which is suitable for handling data generated from typical IoT sensors without introducing excessive power demands. To accommodate rapid communication with connected devices, the GPIO response time is set at 10 ns, ensuring low-latency control in time-sensitive IoT applications.

C. System Features

To support a variety of IoT use cases, our SoC incorporates operating system support primarily for RTOS options like Zephyr and FreeRTOS, widely used in embedded applications for their real-time capabilities. Limited Linux support is also achievable for applications requiring more advanced OS features.

Security is an essential concern in IoT[16], and combining secure boot procedures and hardware-based cryptographic accelerators in our proposed SoC highlights data integrity and significantly improves the security of IoT deployments.

The architecture provides hardware acceleration for cryptographic functions, such as AES and SHA, which enables safe data handling without severely reducing overall speed or power consumption. An optional hardware accelerator is also available for lightweight machine learning inference, allowing easy data processing and decision-making at the edge, thus minimizing the requirement for constant cloud access.

Time-sensitive networking (TSN) functionality is built for deterministic communication, ensuring that the SoC can handle applications where exact timing is important, such as industrial IoT and real-time monitoring systems.

In addition, This design uses dynamic voltage and frequency scaling (DVFS), which assures optimal power efficiency[10]. DVFS optimizes power usage based on the present workload, which is extremely helpful in IoT scenarios where power limits are a main concern. This functionality ensures that the SoC can dynamically alter its energy usage, conserving power during periods of reduced activity.

The power consumption of the SoC, P, is determined by dynamic voltage scaling:

$$P = C * V^2 * f \qquad 1)$$

Where,
C is the capacitance,
V is the operating voltage,
f is the operating frequency.

For dynamic power management, the SoC applies alternative voltage-frequency pairs based on workload needs, using Dynamic Voltage and Frequency Scaling (DVFS). In active mode, the power consumption is roughly 0.05 mW/MHz at a core voltage between 0.5V and 0.8V.

D. Physical Implementation

The 22nm process node is selected for its balance between cost-effectiveness and energy efficiency, making it optimal for IoT devices with limited resources, where balancing performance with production cost is important[6]. The core voltage range of 0.5V - 0.8V aligns with this process node's capabilities, optimizing power use without sacrificing functionality. Standard package options like BGA and QFN are utilized to ensure compatibility with various IoT device form factors, from compact consumer electronics to industrial systems.[19]

With a die size estimate of approximately 15 mm² and an estimated gate count of 100k-200k gates, the SoC achieves an optimal balance between complexity and resource constraints, well-suited for IoT applications where die area and gate count must be minimized to conserve power and reduce costs. The adjusted design also maintains I/O voltage at 1.8V, ensuring compatibility with standard peripheral devices commonly used in IoT systems.

SoC Specifications:

- Supports 64-bit RV64GC instruction set with general-purpose and compressed extensions.
- Multi-level memory hierarchy: 64KB L1 cache, 256KB L2 cache, 1MB on-chip

SRAM, and external DDR4 support
- 64' 64-bit GPIO Connection pins
- Enhanced UART with DMA support
- AXI bus for high-performance components and APB for low-power peripherals
- Dynamic clock gating and multiple power domains with DVFS support
- JTAG debug interface and embedded trace macrocell
- Capable of executing C code with support for Zephyr RTOS or FreeRTOS

Reconfigurable components, such as eFPGA integration demonstrated in [17], enable the proposed SoC to be highly adaptable to various IoT applications, providing flexibility while minimizing design complexity

IV. Comparative Analysis

As described in the above sections, each implementation is unique and optimized for distinct objectives. When analyzed for the aim of IoT applications, below is the comparison of features of each implementation.

Table I.Comparison of RISC-V SoC Specifications

Implementation	PicoRV	MicroRV	RavenSoC	Ultra Embedded RISC V(biRISC-V)
Memory	32KB	32KB	32KB	64KB
Bus Interface	MUX, AXI4-Lite, AXI4, Wishbone	MUX	AXI	AXI
Instruction Set	RV32IMC	RV32I	RV32IMC	RV32IMZicsr
Pipelining	None	None	None	6-7 Stage
Peripheral	SPI, GPIO, UART	GPIO, UART	SPI, GPIO, ADC, DAC, Over-temperature	None
FPGA Implementation	Xilinx Kintex-7, Xilinx Artix-7, Virtex Series	Lattice HX8K	ASIC	Xilinx Artix7
Operating Frequency	714 MHz	28.61 MHz	100 MHz	>50 MHz

Table.II.SoC Comparative Review- Achievements vs. Drawbacks

Implementations	Achievements	Drawbacks
PicoRV32	Capable of executing C programs, supports several bus configurations including MUX and AXI4	Limited feature set
MicroRV	Comprehensive toolchain, Efficient pipeline design, Bare Metal Applications	Limited peripheral support, Absence of SPI flash memory and inadequate documentation
RavenSoC	Completely functional ASIC with temperature alarm, DAC, and ADC	Not Enough documentation
UltraEmbedded RISC V(biRISC-V)	Flexibility and Customization, Can run Linux, Has cache and TCM options	Limited peripheral support, Memory management limitations

Table III: Comparative Analysis of SoC Architectures for IoT Applications

Feature	PicoRV32	MicroRV32	RavenSoC	Proposed SoC
Instruction Set	RV32IMC	RV32I	RV32IMC	RV64GC
Power Management	Limited	Limited	Limited	DVFS, Multi-level Sleep
Security Features	Basic	Basic	Basic	Secure boot, AES/SHA Accel
Max Clock Frequency	714 MHz	28.61 MHz	100 MHz	600 MHz
Target IoT Features	General Purpose	Educational	Basic I/O	Sensor Interface, ML Accel

Table 3 presents a comparative analysis of existing SoC architectures alongside the proposed SoC. The comparison highlights key features such as instruction set, power management capabilities, security features, maximum clock frequency, and specific IoT-targeted features. The proposed SoC addresses several of these gaps[12][19], particularly in terms of energy efficiency and modular scalability.

For a hardware design to be IoT compatible, it must have few of these features:

- Low power consumption
- Small form factor
- Environmental robustness
- Support for High-Level Languages (C/C++)
- Interface with standard sensors
- Cost-effectiveness
- Support for simple network interfaces

There are many ARM implementations for such IoT applications, however as previously noted, these implementations require expensive licensing costs to adapt and develop. However, increased support for the RISC-V architecture, aided by the completion of GNU/Linux RISC-V libraries, has enabled the building of entire RISC-V systems for specific IoT requirements[17].

PicoRV (Section 2.1) proposes a basic architecture for a general-purpose low-power CPU, however it lacks sensor interface capabilities. Furthermore, its support for only RV32IMC extensions should be improved for certain tasks like cryptography, which is sometimes required for IoT applications[11]. Furthermore, many IoT devices are increasingly utilized to pre-process data from sensors. Thus, adding support for DSP extensions can be beneficial[2]. An ideal IoT-centric hardware must strike a compromise between implementation size and feature set. Power consumption is also a key concern. These parameters were investigated as part of the implementations in Section 2.

Though most implementations are fairly robust for general use, several lack IoT-specific capabilities. Furthermore, the design technique was chosen with a general-purpose CPU's requirements in mind. This means more complexity where cutbacks may be made to reduce power consumption.

V. Societal Implication

The Internet of Things (IoT) continues to be a transformative technological revolution, with its impact becoming increasingly apparent across various sectors. However, the widespread implementation of this technology still faces several challenges[1]. These issues include, but are not limited to:

- Interoperability issues
- Costs of implementation and operation
- Infrastructure constraints and scalability
- Ethics and regulations
- Environmental sustainability
- Issues with accessibility and the digital divide

Many of these issues can be addressed through the development of open-source, modular hardware solutions. The use of open Instruction Set Architectures (ISAs) like RISC-V, which doesn't involve licensing fees, helps to greatly reduce development and implementation costs. IoT deployments in underserved areas can gain from low-cost, energy-efficient hardware solutions[1][18]. And the flexibility of RISC-V ISA allows for region-specific changes and feature set optimizations.

For instance, IoT nodes in rural locations might prioritize energy efficiency and durability, whereas urban installations could focus on high-bandwidth connectivity and powerful data processing capabilities. The modular approach to SoC architecture, which includes controlled activation of selected modules, offers an adaptable platform that is responsive to broad use cases and environments.

Rural and underserved locations in both developing and industrialized nations present distinct potential and challenges for IoT technology adoption. The price and accessibility of IoT devices in these locations are significant considerations in their effective implementation. To enable widespread acceptance and maximize social benefits, the design and development of IoT hardware must prioritize cost-effectiveness without compromising on important functionalities.

The deployment of affordable yet highly capable IoT devices has transformative potential for expanding internet access and connectivity in rural areas. IoT technology's increased connectivity has the potential to bring significant improvements in a variety of industries. In agriculture, IoT allows precision farming practices that maximize resource efficiency and increase crop yields, resulting in higher farmer income and food security. IoT-powered learning tools and distance education platforms can help narrow the educational gap between urban and rural places. IoT enhances environmental monitoring by allowing sensors to spot natural disasters, climate change effects, and pollutants early on, enabling better resource management and preparedness. Additionally, energy management can be revolutionized through smart infrastructure and IoT-based systems that improve energy efficiency and integrate renewable sources more effectively. Lastly, IoT promotes economic development by supporting a connected ecosystem of industries, encouraging value-added processing, marketing, and distribution, and eventually boosting the general economic growth of rural communities.

The introduction of IoT in rural and underdeveloped areas presents serious ethical and sociological challenges. Ensuring data privacy and security is critical, especially in places where cybersecurity precautions are not well understood. Targeted educational programs to boost digital literacy are required for effective IoT solution adoption, allowing local populations to harness these technologies proficiently. In addition, the design of IoT implementations must adhere to cultural norms and traditions, demonstrating a commitment to cultural sensitivity. Addressing sustainability is critical, as the carbon footprint of IoT device manufacturing, deployment, and disposal must be carefully managed[16], [18], [19].

VI. Conclusion and Future Scope

IoT is a fast expanding domain, determined by active developments and discoveries. As the technology grows, the needs for IoT hardware continue to rise, demanding constant upgrades and modifications to open-source designs. A complete solution that satisfies the different needs of a typical IoT node can greatly contribute to the success and widespread acceptance of IoT applications. In order to ensure that the hardware is both strong and useful, the suggested improvements in this area seek to achieve a careful balance between functionality and complexity.

Building a thorough verification environment is an important subject for further development. By using Universal Verification Methodology (UVM) tools, recursive revisions can add rigorous testing methodologies for the RISC-V core and system-on-chip (SoC). This methodological approach would provide complete validation of correctness, performance indicators, and system resilience under varied operational scenarios.

Pipeline optimization gives an extra option for architectural enhancement. Although our RV64GC core displays great performance capabilities, the adoption of more advanced pipeline mechanisms and better branch prediction algorithms should develop significant processing enhancements. This optimization would involve careful analysis of the trade-offs between increasing hardware complexity and performance advantages.

Power management optimization appears as a crucial area for future research and development. Building upon our current dynamic clock gating and multiple power domain architecture, future iterations may use more granular power gating methodologies and implement adaptive voltage scaling techniques. These enhancements would be particularly beneficial for battery-powered IoT devices, where energy consumption represents an important operational parameter.

Security enhancement is an important issue for future deployments. Given the growing importance of cybersecurity in IoT applications, future versions may have strong security features including hardware root of trust methods, secure boot protocols, and hardware-level isolation for secure enclaves. These security measures would greatly strengthen the system's resilience to possible cyber-attacks.

Integrating these recommended enhancements into future implementations will help develop more sophisticated, secure, and efficient IoT hardware solutions. This adaptive trend will allow for the continuing advancement of this innovative technologies, enabling fresh use cases and applications across a wide range of industrial sectors. As the IoT landscape grows, our improved RISC-V-based SoC design provides a solid platform for tackling present issues and future opportunities in this dynamic industry.

References

[1] B. B. Sinha and R. Dhanalakshmi, "Recent advancements and challenges of Internet of Things in smart agriculture: A survey," Future Generation Computer Systems, vol. 126, pp. 169-184, 2022

[2] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, et al., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 10, pp. 2700–2713, Feb. 2017, doi: 10.1109/TVLSI.2017.2654506

[3] D. A. Patterson and J. L. Hennessy, “Computer Organization and Design RISC-V Edition: The Hardware Software Interface.” Amsterdam: Morgan Kaufmann, 2021.

[4] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The RISC-V instruction set manual, volume I: Base user-level ISA," EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62, vol. 116, pp. 1–32, 2011.

[5] Ahmadi-Pour, S., Herdt, V., & Drechsler, R. (2021). “MircoRV32: an open source RISC-V cross-level platform for education and research.” In Proceedings of the Workshop on Design Automation for CPS and IoT.

[6] N. Morawski, "Comparison and analysis of RISC-V processor implementations for resource-constrained FPGAs," 2024. [Online]. Available: https://mrwski.eu/projects/data/riscv_ice40.pdf.

[7] M. Jahnke, L. Bublitz, and U. Kulau, "Performance Evaluation of PicoRV32 RISC-V Softcore for Resource-Constrained Devices," in 2023 IEEE Nordic Circuits and Systems Conference (NorCAS), 2023, pp. 1–6.

[8] S. Ahmadi-Pour, V. Herdt, and R. Drechsler, “The MicroRV32 framework: An accessible and configurable open source RISC-V cross-level platform for education and research,” Journal of Systems Architecture, vol. 133, p. 102757, Oct. 2022, doi: 10.1016/j.sysarc.2022.102757.
[9] GuangTang, "Research and design of low-power, high-performance processor based on RISC-V instruction set architecture," J. Phys. Conf. Ser., vol. 2221, no. 1, p. 012008, May 2022, doi: 10.1088/1742-6596/2221/1/012008
[10] H. B. Amor, C. Bernier, and Z. Přikryl, "A RISC-V ISA Extension for Ultra-Low Power IoT Wireless Signal Processing," IEEE Trans. Comput., vol. 71, no. 4, pp. 766–778, 2022, doi: 10.1109/TC.2021.3063027.
[11]Efabless, “GitHub - efabless/raven-picorv32: Silicon-validated SoC implementation of the PicoSoc/PicoRV32,” GitHub. https://github.com/efabless/raven-picorv32
[12] A. Dörflinger, J. Schneider, A. K. Tesfamariam, S. Haas, and S. Ortega-Cisneros, "A comparative survey of open-source application-class RISC-V processor implementations," in Proc. 18th ACM Int. Conf. Computing Frontiers, 2021, doi: 10.1145/3457388.3458657
[13] N. Gala, A. Menon, R. Bodduna, G. S. Madhusudan, and V. Kamakoti, "SHAKTI Processors: An Open-Source Hardware Initiative," in Proceedings of the 2016 29th International Conference on VLSI Design and 15th International Conference on Embedded Systems (VLSID), Kolkata, India, 2016, pp. 7–8, doi: 10.1109/VLSID.2016.130
[14] M. A. Sadeeq, S. R. Zeebaree, R. Qashi, S. H. Ahmed, and K. Jacksi, "Internet of Things security: a survey," in Proc. 2018 Int. Conf. Advanced Science and Engineering (ICOASE), 2018, pp. 162–166, doi: 10.1109/ICOASE.2018.8548785
[15]Ultraembedded, “GitHub - ultraembedded/riscv: RISC-V CPU Core (RV32IM),” GitHub. https://github.com/ultraembedded/riscv
[16] F. Conti, G. Paulin, A. Garofalo, D. Rossi, A. Di Mauro, G. Rutishauser, G. Ottavi, M. Eggiman, H. Okuhara, and L. Benini, "Marsellus: A Heterogeneous RISC-V AI-IoT End-Node SoC With 2–8 b DNN Acceleration and 30% Boost Adaptive Body Biasing." IEEE Journal of Solid-State Circuits, vol. 59, no. 1, pp. 128–142, Jan. 2024, doi: 10.1109/JSSC.2023.3318301.
[17] P. D. Schiavone, D. Rossi, A. Di Mauro, F. K. Gürkaynak, T. Saxe, M. Wang, et al., "Arnold: An eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End Nodes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 29, no. 4, pp. 677–690, Apr. 2021, doi: 10.1109/TVLSI.2021.3058162
[18] K.-D. Nguyen, T.-K. Dang, B. Kieu-Do-Nguyen, C.-K. Pham, and T.-T. Hoang, "RISC-V-Based System-on-Chips for IoT Applications," in Proc. 2024 IEEE Hot Chips 36 Symposium (HCS), 2024, doi: 10.1109/HCS61935.2024.10665181
[19] L. Valente, Y. Tortorella, M. Sinigaglia, G. Tagliavini, A. Capotondi, L. Benini, et al., "HULK-V: A Heterogeneous Ultra-low-power Linux capable RISC-V SoC," in Proc. 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2023, pp. 1–6. doi: 10.23919/DATE56975.2023.10137252
[20] N. D. Nguyen, D. H. Bui, and X. T. Tran, "Tiny Neuron Network System based on RISC-V Processor: A Decentralized Approach for IoT Applications," in Proc. 2022 Int. Conf. Advanced Technologies for Communications (ATC), 2022, pp. 98–103, doi: 10.1109/ATC55345.2022.9942990