Semiconductor Optoelectronics ISSN: 1001-5868

COMPARATIVE STUDY OF RISC-V ISA FOR LOW-POWER APPLICATIONS

H S Kavitha[1], Manohar B S[2], Raghavendra L[3], G Parmeshappa[4]
[1,2,4]Assistant Professor, Department of Electronics & Communication Engineering, JSS Academy of Technical Education, Bengaluru and Visvesvaraya Technological University, Belagavi-590018, Karnataka, India.
[3]Associate Professor, Department of Electrical and Electronics Engineering, ATME College of Engineering, Mysuru and Visvesvaraya Technological University, Belagavi-590018, Karnataka, India
Corresponding Author; Manohar B S, Department of Electronics & Communication Engineering, JSS Academy of Technical Education, Bengaluru Karnataka, India,

Abstract: Free, open-source technology called the RISC-V Instruction Set Architecture (ISA) has attracted a lot of interest in the low-power gadget market. RISC (reduced instruction set computing) design is a good option for low-power devices. Recently, several companies have adopted the RISC-V ISA for use in a range of low-power applications, including Internet of Things (IoT) devices, wearable technology, and embedded systems. The RISC-V architecture's systems require little power, making them ideal for battery-operated devices where prolonged battery life is crucial. At the start of this paper, the RISC-V ISA will be introduced along with its key features. Thereafter, recent developments in the market for low-power applications will be looked at. The RISC-V ISA's unique features and advantages in each application will then be highlighted in in-depth reviews of the different low-power applications utilized. In summary, the ISA's open-source and modular architecture makes it an extremely desirable option for a low-power application market that will examine and optimize.

Index Terms: RISC-V, ISA, Micro-architecture

1. INTRODUCTION

The call for portable, electric-powered devices like mobile phones, medical apparatus, and Internet of Things (IoT) gadgets is growing in the current technological era. Improvements in CMOS technology over the past four decades have increased processor speed, but these improvements are hitting physical limits. Designers are therefore looking for alternative methods to increase power and efficiency within these constraints. Microarchitecture and the Instruction Set Architecture (ISA) are the essential components of CPU design [1].
Microarchitecture, also known as computer organization, outlines a processor core's physical and logical design. It encompasses the instruction set design, memory hierarchy, and execution unit. Microarchitecture significantly affects the performance and energy efficiency of a processor and is an essential factor in the overall design of a computer system. Various

manufacturers have developed diverse microarchitectures each offering a distinctive balance between performance, power consumption, and cost. The microarchitecture of a processor core greatly influences the system's capabilities and performance, determining the execution of instructions, storage and, retrieval of data, and the large number of commands that can be implemented each clock cycle. Advancements in microarchitecture have yielded the creation of multi-core processors, enhancing the overall performance and efficiency of computer systems. The selection of microarchitecture is a crucial aspect in the design of a computer system and can greatly influence the overall efficiency of the processor.

1.1 The microarchitecture of a processor core has certain disadvantages, including:

1) Complexity: The creation of microarchitectures can be laborious and time-consuming, necessitating a substantial outlay of money and knowledge.
2) Compatibility: It can be difficult to guarantee compatibility with current systems and software because various microarchitectures may have different instruction sets and memory hierarchies.
3) Performance trade-offs: The microarchitecture's design decisions may affect performance, power usage, and cost. It can be difficult to strike a balance between these trade-offs, and a design that favors one aspect may be detrimental to another.
4) Limited scalability: As microarchitecture designs become more complex, scaling them to accommodate the requirements of bigger and more powerful systems becomes more challenging.
5) Technology limitations: It may be difficult to meet desired performance and power consumption targets because the technology used to execute microarchitecture and design may be constrained by the state of the technology.

Overall, the design of a microarchitecture is a complex task that requires careful consideration of trade-offs and technology limitations to achieve the desired performance, power consumption, and cost goals.

Two ISAs currently rule the processor market: x86, which is used in PCs and servers, and ARM, which is used in portable devices.[2]

1.2 Disadvantages of commercial ISA:

1) Commercial ISAs are proprietary, and they do not welcome freely available competitive implementations.
2) It is extremely difficult to execute the two most popular commercial ISAs (x86 and ARM) in hardware in a way that supports widely used software stacks and operating systems.
3) Business ISAs are the only widely used market segments.
4) Previous study infrastructures were created around commercial ISAs that are no longer in use or demand, such as SPARC and MIPS (Alpha). These lose out because the ISA's ongoing intellectual property problems and accompanying tools make it difficult for interested parties to continue supporting the ISA.
5) The prevalent business ISAs did not focus much on extensibility.

It is possible for anyone to use the RISC-V (Reduced Instruction Set Computing V) instruction set architecture (ISA) for any reason. It was created by UC Berkeley and is available for use in the study, education, and industry implementation of computer architecture. RISC-V is a reduced instruction set computing (RISC) architecture that is made to serve a variety of computing platforms, including both entry-level microcontrollers and cutting-edge supercomputers. For a wide range of uses, the RISC-V ISA offers an easy-to-use and flexible instruction set architecture.[3]

A processor core is the foundational component of a processor, consisting of instructions and logic used to process data and perform operations. It executes instructions and performs operations, such as fetching instructions from memory, reading and writing data, and executing arithmetic and logic operations. A processor core typically includes multiple pipeline stages, each performing a specific task.

Pipeline architecture refers to the design of the processor's instruction set and data paths, allowing instructions to be executed in a sequence of stages. This design enables instructions to be processed in parallel, enhancing the speed of instruction execution. The common stages of the pipeline are fetch, decode, execute, and WB.

1.3 Characteristics of RISC-V:

1) RISCV is an open ISA that is accessible for free to both industry and academic researchers.
2) Not just simulation or binary translation, but direct native hardware execution is also an option.
3) Any microarchitecture style or technique can be implemented effectively.
4) It is divided into a compact basic integer ISA that can be used by specialized accelerators.
5) It allows a variety of ISA variants and extensive user-level extensions.
6) It enables implementations like heterogeneous multiprocessors and highly parallel multicore.

1.4 Advantages of RISC V:

The RISC-V architecture was established in 2010, and its creators benefited from the experience gained from previous instruction set architectures, leading to the development of a streamlined base ISA known as RV32I. This base ISA, which consists of 32-bit wide registers for integer operations, remains unchanged but can be expanded as needed for specific applications. While a simple embedded processor might only require a limited number of extensions, a chip intended for use in servers can have a broader range of capabilities. These capabilities may evolve, but the core of the system will never become overly complex.

2. RELATED WORK

The RISC-V ISA has been implemented with the first dual-core processor, which features a 64-bit vector accelerator and was fabricated using a 45 nm SOI process. With a clock frequency of up to 1.3 GHz at 1.2 V and an area of 3 mm^2, sets a new standard for performance and power efficiency [4]. The shared LNU architecture in [5] integrates multiple LNUs into a multi-core system with four 32-bit open RISC cores. Tests show that this design can be up to 4.1 times more energy-efficient than other architectures, creating a promising solution for low-power applications. A 5 nm-fabricated system-on-chip (SoC) that is built around a tightly coupled multi-core ensemble. It is ideal for a variety of IoT applications thanks to its average power consumption of less than 20 mW at 0.8 V and secure near-sensor data analytics architecture[6]. A SEE-tolerant microprocessor architecture built on the RISC-V ISA is introduced to address reliability concerns. Two novel re-computation methods are suggested by this architecture to detect errors, and error-correcting codes are used to tolerate errors in registers and memories [7]. The impact of coefficient quantization is examined for fixed-width multipliers with linear correction functions. Two topologies are found to be the most efficient ones in the research, providing ideas for the creation of future fixed-width multipliers [8]. The idea behind this is to create a regular partial product matrix with fewer partial product rows and very little overhead to simplify partial product reduction and decrease the area, delay, and power of modified booth encoding (MBE) multipliers [9]. [10], a 2's complement arithmetic-based algorithm speed, m-bit by n-bit parallel array multiplication is described. The 2's complement multiplication issue is transformed into a parallel array addition problem using this algorithm. Ref. [11] suggests and examines high-radix dividers at various levels while taking into consideration different design factors. The research suggests that error-compensation and cell truncation techniques increase the usefulness of imprecise computation in high-radix dividers.

3. LITERATURE REVIEW

Modern core makers place a high priority on creating instruction-level parallelism (ILP) in their designs as they move from straightforward to complex ones. These ILP designs enable parallel cache and memory operations, but they also increase the expense of accessing cache and memory through the processors they are based on. As a result, it is imperative to extract memory hierarchy parallelism (MHP) with the least amount of negative effect on ILP extract researchers.

The Load Slice Core architecture fills this need by adding a secondary in-order pipeline to the core's effective in-order, stall-on-use design to enable stalled instructions in the primary pipeline to be passed by memory access and address-generating instructions. [12] proposes a novel processor microarchitecture to extract memory hierarchy parallelism from unaltered binaries while reducing hardware overheads. This is accomplished by using a cutting-edge iterative algorithm that can be successfully applied in hardware to extract backward slices carrying address-generating instructions. The second in-order pipeline is then used to perform these backward slices, allowing them to go around instructions that are currently being held up by pending loads. The Load Slice Core architecture outperforms previous work by providing a lightweight, hardware-based method for completing previously pending loads without repeat execution.

The amount of pipeline stages is an important consideration in any processor design because it has a big effect on the throughput and operating frequency. Higher pipeline stages can boost frequency, but they can also generate data, manage risks, and reduce the number of instructions per cycle (IPC). The Icyflex1 processor, described in [13], blends DSP and microcontroller unit features with a 3-stage pipeline to produce a low-power, programmable processor with a re-configurable architecture and instruction set. The combination of these characteristics allows for effective DSP algorithm programming of the processor. By avoiding the need to introduce cycles for retrieving input data or saving output data, the architecture seeks to reduce latency. The Icyflex1 processor's shorter pipelines minimize latency caused by interrupts produced by external hardware, and it is intended to be implemented as a synthesizable VHDL software intellectual low-power core. [14] The Program Vectoring Unit, Data Move Unit, Data Processing Unit, Multicast, and Debug Unit make up the heart. It is important to note that the core's focus is on low-power rather than high-speed uses.

The Rocket processor, a six-stage, single-issue, in-order algorithm that can carry out 64-bit scalar RISC-V instructions. To lower parasitic capacitance within the device and boost performance, the 45nm SOI process was used to build the dual-core processor, which was built with a vector accelerator. Each Rocket core has a direct-mapped 8 KB instruction cache that is connected to a 2-way set-associative 16 KB instruction cache, and each vector accelerator has its own 8 KB instruction cache. The single-lane, decoupled vector processor that makes up the vector accelerator is tailored for an ASIC process. Chisel is a language for building hardware that enables designers to represent hardware as highly parametrized generators to help with design tuning under different performance, power, and process constraints. The processor's Code is written using a chisel. The Chisel-generated Verilog is mapped to a multi-Vt standard cell library and memory-compiler-generated SRAMs in a 45nm SOI technology using the Synopsys ASIC CAD tool pipeline, which includes the Design Compiler and IC Compiler. The findings of the chip are then edited and contrasted with those of IBM's Blue gene/Q processor.

The Andes core NX45, which is described in [15], is a 64-bit, 8-stage superscalar processor created by Andes Technology (Taiwan), which complies with the RISC-V specification and supports the "G" (integer base+IMAFD) standard instructions, "C" (16-bit compression instructions, and "N" for user- level interrupts) instruction sets. The performance efficiency for many applications is greatly improved by the two instructions it issues per cycle. Its "FD" extensions enable single-precision and double-precision floating-point instructions that go around with IEEE 754. Additionally, the processor integrates Mem Boost to significantly increase memory bandwidth and decrease memory latencies for apps that make frequent memory accesses. Additionally, NX45 has instruction and data caches, local memories, ECC error protection, and a cutting-edge low-power branch prediction system for effective branch processing. A vector and pre-emptive interrupt controller, an AXI 64-bit bus, sophisticated power management, and a JTAG debug and trace interface for software development assistance are also included.

As they can be used to train machine learning algorithms for data analytics and contain sensitive information, sensor data plays a major role in the modern world. [16] has proposed Fulmine, a System on Chip (SoC) that makes use of a closely coupled multicore cluster with specialized blocks for data processing and encryption operations, to handle the issues related to large data transfer requirements and the risk of data theft. Typically, the energy needed to transfer large amounts of sensor data to servers results in decreased sensor efficiency. Additionally, there is a danger of data theft due to the sensitive nature of this data. Fulmine provides a near-sensor smart data analytics solution to address these issues. Only the required and pertinent data is chosen, encrypted with specialized blocks, and transmitted to the server.

Fulmine is a 4-core 32-bit machine with a multi-precision convolution engine tailored for CNN calculations and an effective cryptographic engine for AES-128 and KECCAK-based encryption. [17]. The system is built on the PULP system, which consists of two SoCs—a cluster in the frequency domain and one in the voltage domain—that are parted by dual clock FIFOs and level shifters and communicate with one another via AXI4.

The Parallel Ultra Low Power (PULP) system, as proposed by [5], comprises a cluster of 32-bit Logarithmic Number System Units (LNU) to ensure efficiency. While fixed numbers and integers may be adequate for certain applications, a larger dynamic range requires the use of a floating point, which can be translated into simple integer operations by implementing the Logarithmic Number System (LNS). To maximize efficiency, the LNU is shared across multiple cores, as a standard LNU is larger than a Floating Point Unit (FPU).

The PULP system is fabricated using 65nm technology and consists of four chips, one of which is a single precision FPU, and the other three are LNU with different parameters.

The system features five datapath units, including a multi-unit for Fused Multiply-Add (FMA) and Fused Divide-Add (FDA) operations, an AddSub pre-processing unit, a Main Interpolator unit, a LogExp Interpolator unit for logarithmic operations, and a post-processing unit. These operations are frequently utilized in computer vision applications.

Each cluster of the PULP system has undergone an entire back-end design process utilizing the Cadence Innovus tool, with evaluation performed using the Synopsys Design Compiler.

The [14] lizard core processor is a commercial out-of-order processor that can be customized based on certain parameters. It fully supports the RISC-V RV64IM ISA, but it is not superscalar and can retire only one instruction per cycle. The core includes a pipelined multiplier and an iterative divider that supports all RV64M instructions. The Fetch unit also includes a parameterizable fully associative BTB to help amortize the branch resolution latency.

The Lizard core is written in PyMTL, which allows for adjusting the sizes of issue queues, the ROB, the number of physical registers, and the latencies of the multiplier and divider. Non-memory instructions are executed out of order, except for special instructions like CSRs. However, since the Lizard core does not have memory disambiguation, memory instructions can execute only partially out of order. To enable this, the Lizard core contains two issue queues, one for normal instructions that execute completely out of order, and another for load instructions and the address generation component of store instructions that execute out of order between store instructions.

The core also implements register renaming and snapshot- ting/restoring of the rename table, allowing it to speculate on branches to any parameterizable depth. It supports precise exceptions with an architectural rename table that rolls back to the state when an exception occurs.

The core has two parts: the front end and the back end. The front end is responsible for fetching and decoding instructions, while the back end executes instructions. Instructions enter the backend in order, where they are reserved a spot in the ROB and have their architectural registers renamed. The instructions are re-ordered in the commit stage using a reorder buffer. The pipeline’s length varies depending on the instruction type, but basic integer instructions using the ALU are approximately 10 cycles from fetch to retirement.

4. METHODOLOGY

The following passage delves into the details of micro-architectural optimizations, it is aimed at enhancing the efficiency of the processor core, with a specific emphasis on improving power and performance. Starting with the RISC- V ISA, the core of the discussion centers around the base integer instruction set and extensions that include integer multiplication and division. The micro-architecture discussed here serves as the foundation for an advanced core capable of supporting a wide range of extensions and features. Efficiency in the design of primary components is critical to the altogether enhancement of any system. Thus, in this presented micro-architecture, optimization efforts are focused on basic components such as the multiplier, divider, and DBP, to achieve optimal results. To allow the core to operate at a maximum frequency of about 200 MHz, the critical path of the core has also been optimized. To keep data, control, and structural hazards to a minimum, pipeline stages are arranged in a manner that limits their occurrence.

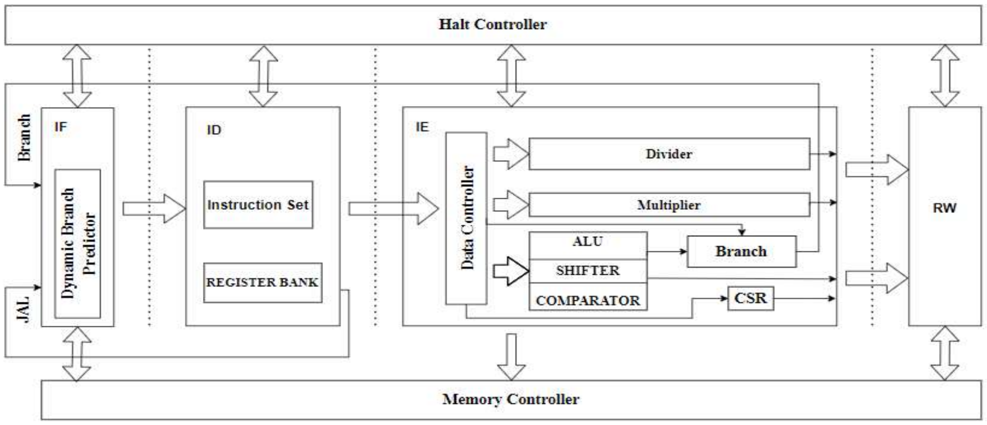

Fig. 1. Steps in the pipeline of the existing microarchitecture.

The pipelining technique is typically used to conserve time and energy so that hardware is fully utilised, and as a result, we get the fastest processing speed possible. Pipelining typically reduces power by reducing the critical route latency by adding a latch between combination blocks, and for additional power reduction, asynchronous blocks are coupled to synchronous blocks using handshake signals to provide high speed and good power reduction. To save time and reduce minute dangers, more multi-cycled clocks are employed from outside to complete the current step as quickly as feasible. The number of pipeline stages is a key consideration in the design of any processor. Incorporating more pipeline stages can enable a higher operating frequency, leading to greater overall throughput. Nonetheless, this approach also heightens the potential for data and control hazards and diminishes the IPC (Instructions per Cycle). To attain top performance, processors are fine-tuned by employing techniques such as branch predictions, prefetch buffers, and speculation. Nonetheless, such optimizations elevate power consumption and may not be well-suited for low-power applications [1].

As an illustration, the ARM Cortex-M4 micro-architecture employs a three-stage pipeline and features a register bank with an individual write-back interface. The lack of a fourth pipeline step and a second write-back port, however, can hinder load operations. Although a three-stage pipeline can have high IPC, its frequency is limited by the fact that numerous operations can happen in a single stage.

To achieve a balance between IPC and frequency, the proposed micro-architecture has been designed with four pipeline stages. The core's pipeline stages are Instruction Fetch (IF), Instruction Decode (ID), Instruction Execute (IE), and Register Write (RW), as illustrated in Figure 1. The majority of instructions can be executed within these four stages, although some may need to be modified to achieve optimal performance. For example, the unrestricted jump instruction JAL is completed following the initial phase since it does not require any register read/write operation.[18] On the other hand, load instructions require an additional stage (Memory Read) before reaching the RW stage due to the inherent memory access delays involved in their execution. In the time of the Instruction Fetch cycle, similar to the Thumb-2 instruction set of ARM, the RISC-V standard offers a definition for compressed instructions, which are just 16-b long and replicate full instructions with a decreased immediate size and RF address. These 16-b instructions are simple to handle as long as the 32-b word boundaries are in alignment with them. Every time two compressed instructions are fetched, a compressed instruction decoder recognises them, decompresses them into standard 32-b instructions, and delays the fetch unit for one cycle. In the Instruction Decode Stage, the instructions are delivered to the instruction decoder from program memory along with the expected PC and the appropriate PC from the fetch step. Instructions are categorised using the lower 7-bit (Op-code) of the instruction. [19] Decoding of subgroup information from a 3-bit field (function field). Regardless of the type of instruction, the position of the sources' and destinations' register addresses, which each have a 5-bit field, is constant. For additional processing, the instantaneous data is sign-extended to 32 bits if an immediate type instruction has been executed.[20] During the IE stage, the micro-architecture first checks whether register values are available to read. If so, the execution proceeds; otherwise, the values are obtained from the IE/RW stage, based on the preceding instructions. Once obtained, then, these register numbers are directed to various Tol Units (FU) within the IE stage for execution. Five FUs make up the IE stage: ALU, Comparator, Shifter, Multiplier, and Divider. The clock cycles and operations of these FUs are as follows:

1) ALU: ALU deals with addition, subtraction, logical and, logical or, logical xor (1 clock)
2) Comparator: Comparator deals with signed or unsigned comparison (1 clock)
3) Shifter: Shifter deals with logical shift and arithmetic shift (1 clock)
4) Multiplier: Multiplier deals with signed and unsigned multiplication (2 clocks)
5) Divider: Divider deals with signed and unsigned division (8 clocks).

Additionally, the IE phase is responsible for producing signals that enable memory read/write operations and branch signals. The ALU unit provides the memory read/write address, while the second source register (rs2) furnishes the memory read/write data[21]. The meteorite enables the signal to be generated in the instruction ID stage, and these three signals are merged to regulate the memory read/write operations. As for the branch signals, they consist of two elements: branch enables and branch address. The ALU output is utilized to assign the branch address, while the comparator output determines whether the branch enables signal is triggered, depending on the instruction type. When needed, the data memory is accessed at the Read Write stage, which is also known as the Memory Access stage. In this stage, data is either written or kept in data memory. In loading and storing instructions, it is used.

5. ANALYSIS OF RISC-V IMPLEMENTATION

5.1 BOOM

BOOM is one of the processors that support the RISC-V ISA and can be utilized to substitute the Rocket core. With a branch mispredict penalty of 12 cycles, it has a complex 10-stage pipeline design. The front end consists of a customizable branch prediction unit with a high accuracy rate, a banked L1I cache, a TLB, a decode stage, and other components. The execute pipeline is composed of eight different functional units that can be allocated by a distributed scheduler. Custom ISA extensions can be implemented via the RoCC interface.

The BROOM variant of BOOM features a built-in 1 MB L2 cache. The intended operating range for the design is the highest frequency of 1 GHz and a voltage of 0.9 V, with a performance specification of 3.77 CoreMark/MHz. SonicBOOM is the current release of BOOM, which is the RISC-V core with the highest publicly available speed.

5.2 CVA6

CVA6 is a 64-bit processor designed for application-class tasks that implement the RV64GC standard. A branch prediction device with BTB, BHT, and RAS is part of its 6-stage pipeline. The ALU, specialized multiplier/divider, optional FPU, CSR buffer, branch unit, and load/store unit are the six functional units to which instructions are sent. Critical elements like the register file and caches can be configured to maximize either area or timing.

CVA6 has been incorporated into both Chipyard and the OpenPiton3 projects. It has been recorded six times in two different technologies: GlobalFoundries 22 nm FD-SOI technology and UMC 65 nm process. However, the intermediate Tile Link translation required is the primary reason for performance loss when utilizing Chipyard.

5.3 SHAKTI-C CLASS

The IIT Madras-led SHAKTI Processor Program aims to create power processors, System-on-Chips (SoCs), and peripheral IPs for an open-source ecosystem. The SHAKTI C-Class processor, which is specifically designed for the IoT, industrial, and automotive segments, is an appropriate processor with a 5-stage pipeline. It supports both the RV32I and RV64I ISA and can be configured with selective activation of extensions.[22] The CPU supports both AXI4 and Tile Link interconnects and has a GShare two-level branch predictor. The SHAKTI C-Class processor has its framework for SoC designs, software creation, and verification even though it is not integrated into the Chipyard framework. The SCL 180 nm and Intel 22 nm FinFET technologies were used in the production of the CPU. 1.68 DMIPS/MHz is used to show its performance.

5.4 ROCKET

Rocket is a scalar processor created at UC Berkeley with an appropriate design that employs a 5-stage pipeline. It uses the Chisel Hardware Description Language and handles the RISC-V Ind RV64G and RV32G variants. Users can modify Rocket by electing to activate ISA extensions, and the load-store architecture supports both blocking and non-blocking L1D cache configurations. In addition, Rocket offers the Rocket Chip Coprocessor (RoCC) interface, which enables the integration of specialized coprocessors or accelerators[23]. The open-source SoC design generator Rocket Chip Generator can be used to build an integrated System-on-

Chip (SoC) with Rocket cores, caches, and interconnects. The Rocket Chip Generator is integrated into the free and open-source Chipyard framework, which includes sample setups with pre-set cache- and predictor parameters. A setup may also include arbitrary accelerators and peripherals. Since 2012, there have been numerous records out, with SiFive U54 being one of many customizable IP cores offered by SiFive.

6. Summary of Comparisons

The Rocket processor excels in various valuation standards excluding the ASIC processing performance. It provides superior FPGA performance, little FPGA power use, a small ASIC impression, and high efficiency in electricity. Moreover, its flexible layout options make it suitable for academic and commercial projects. In this study, BOOM is the sole Out-of-Order core evaluated, and it can substitute the Rocket core. BOOM exhibits the best ASIC performance among the analyzed cores but has the disadvantage of high resource usage of the FPGA, ASIC area footprint is big, and energy efficiency is poor.

TABLE I: Characteristics of different RISC-V implementations.

	SHAKTI	**CVA6**	**BOOM**	**ROCKET**
Bits	32/64	64	64	32/64
Stages	5	6	10	5
Funct. Units	3	6	8	4
Interfacing	AXI4/TL	AXI4	Tile Link	Tile Link
HDL	BSV	SV	Chisel	Chisel
License	BSD	Solder Pad	BSD	BSD
Framework	shakti-soc	Open Piton	Chipyard	Chipyard

When used with an FPGA, SHAKTI is the option that uses the least electricity and energy. Nevertheless, its L1 cache aspect ratio negatively impacts ASIC design, limiting the maximum frequency, and increasing memory areas, and power consumption. Nevertheless, if this concern is resolved, SHAKTI can achieve superior performance, area, and power efficiency.

7. Conclusion

This study compares various ideas in core architecture, examining their differences and identifying which is superior for performance. It provides a comparative analysis of different implementations of RISC-V, describing the instruction set architecture in the low-power device space. The study also reviews a micro-architecture based on the RISC-V ISA, which achieves high performance with low power requirements, surpassing many existing commercial and open-source recorders of performance, power consumption, and area requirements. In summary, the RISC-V ISA offers an efficient solution for low-power systems, making it an excellent option for battery-operated devices that require long battery life.

8. References

1. Satyajit Bora and Roy Paily, "A High-Performance Core Micro-Architecture Based on RISC-V ISA for Low Power Applications" IEEE Xplore, 1549-7747, 2020.
2. Alexander Dorflinger, Mark Albers, Benedikt Kleinbeck, Benedikt Kleinbeck, Harald Michalik, Raphael Klink, Christopher Blochwitz, Anouar Nechi, Mladen Berekovic, "A Comparative Survey of Open-Source Application-Class RISC-V Processor Implementations," CF '21: Proceedings of the 18th ACM International Conference on Computing Frontiers May, 2021, 1892.
3. A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The RISC- V instruction set manual, volume I: User-level ISA, version 2.0," and EECS Dept., Univ. California at Berkeley, Berkeley, CA, USA, Rep. UCB/EECS-2014-54, May 2014. [Online].
4. Y. Lee et al., "A 45 nm 1.3 GHz 16.7, double-precision GFLOPS/W RISC-V processor with vector accelerators," in Proc. 40th Eur. Solid-State Circuits Conf. (ESSCIRC), Sep. 2014, pp. 199–202. [Online].
5. M. Gautschi, M. Schaffner, F. K. Gurkaynak, and L. Benini, "An ex-tended shared logarithmic unit for nonlinear function kernel acceleration in a 65-nm CMOS multicore cluster," IEEE J. Solid-State Circuits, vol. 52, no. 1, pp. 98–112, Jan. 2017. [Online].
6. F. Conti, "An IoT endpoint system-on-chip for secure and energy-efficient near-sensor analytics," IEEE Trans. Circuits Syst. I, Reg.Papers, vol. 64, no. 9, pp. 2481–2494, Sep. 2017. [Online].
7. S. Gupta, N. Gala, G. S. Madhusudan, and V. Kamakoti, "SHAKTI-F: A fault-tolerant microprocessor architecture," in Proc. IEEE 24th Asian Test Symp. (ATS), Nov. 2015, pp. 163–168. [Online].
8. N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Design of fixed-width multipliers with linear compensation function," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 5, pp. 947–960, May 2011
9. S. Kuang, J. Wang, and C. Guo, "Modified Booth multipliers with a regular partial product array," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 56, no. 5, pp. 404–408, May 2009.
10. C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," IEEE Trans. Comput., vol. C-22, no. 12, pp. 1045–1047, Dec. 1973
11. L. Chen, J. Han, W. Liu, P. Montuschi, and F. Lombardi, "Design, evaluation, and application of approximate high-radix dividers," IEEE Trans.Multi-Scale Comput. Syst., vol. 4, no. 3, pp. 299–312, Jul. 2018.
12. AndesCore Processors. http://www.andestech.com/en/products-solutions/andescore-processors/. Accessed: 2022-12-29.
13. C. Arm et al., "Low-power 32-bit dual-MAC 120 μW/MHz 1.0 V ICYFLEX1 DSP/MCU core," IEEE J. Solid-State Circuits, vol. 44, no. 7, pp. 2055–2064, Jul. 2009.
14. Trevor E. Carlson, Wim Heirman, Osman Allam, Stefanos Kaxiras, Lieven Eeckhout,

Uppsala University, Sweden, Intel, ExaScience Lab, Ghent University, Belgium” The Load Slice Core Mi- microarchitecture” ISCA ’15: Proceedings of the 42nd Annual International Symposium on Computer Architecture June 2015 Pages 272–284https://doi.org/10.1145/2749469.2750407.

15. The ARM Cortex-M3 Processor, the Industry-Leading 32-bit Processor for Highly Deterministic Real-Time Applications. [Online]. Available: https://developer.arm.com/products/processors/cortex-m/cortex-m3
16. L. Dadda, “On serial-input multipliers for two’s complement numbers,” IEEE Trans. Comput., vol. 38, no. 9, pp. 1341–1345, Sep. 1989.
17. S. Chaudhry, P. Caprioli, S. Yip, and M. Tremblay, “High-performance throughput computing,” Micro, IEEE, vol. 25, no. 3, pp. 32–45, 2005.
18. T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, “An evaluation of high-level mechanistic core models,” ACM Transactions on Architecture and Code Optimization (TACO), vol. 11, no. 3, pp. 28:1–28:25, Aug. 2014.
19. S. Palacharla, N. P. Jouppi, and J. E. Smith, “Complexity effective superscalar processors,” in Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA), Jun. 1997, pp. 206–218.
20. R. Serrano et al., "A Low-Power Low-Area SoC based in RISC-V Processor for IoT Applications," 2021 18th International SoC Design Conference (ISOCC), Jeju Island, Korea, Republic of, 2021, pp. 375-376, doi: 10.1109/ISOCC53507.2021.9613880.
21. F. Taheri, S. Bayat-Sarmadi and S. Hadayeghparast, "RISC-HD: Lightweight RISC-V Processor for Efficient Hyperdimensional Computing Inference," in IEEE Internet of Things Journal, vol. 9, no. 23, pp. 24030-24037, 1 Dec.1, 2022, doi: 10.1109/JIOT.2022.3191717.
22. S. Shukla and K. C. Ray, "A Low-Overhead Reconfigurable RISC-V Quad-Core Processor Architecture for Fault-Tolerant Applications," in IEEE Access, vol. 10, pp. 44136-44146, 2022, doi: 10.1109/ACCESS.2022.3169495.
23. Hela Belhadj Amor, Carolynn Bernier , and Zdenek Prikryl, “A RISC-V ISA Extension for Ultra-Low Power IoT Wireless Signal Processing” March 2021 IEEE Transactions on Computers PP(99):1-1 DOI:10.1109/TC.2021.3063027