International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(13s) 2026 pp. 229-257
https://doi.org/10.70917/ijcisim-2026-4051
Article

An Extensive Analysis of RISC-V Floating-Point Units: Architectures, Difficulties, and Prospects

Naveen Kumar V[1], Chandrashekar M Patil[2], Geethashree A[3]

[1,2,3] Department of Electronics and Communication Engineering , Vidyavardhaka College of Engineering, Mysuru, India
[1]Government Polytechnic Arakere, Srirangapatna
Corresponding Author: Naveen kumar.V

Abstract: Floating-point computation is a fundamental aspect of scientific, engineering and artificial intelligence workloads today. As an open-source, modular and extensible instruction set architecture (ISA), the RISC-V has been driving innovation in floating-point unit (FPU) design. From most simple of scalar FPUs for embedded systems to highly capable of vector FPUs for high performance computing, communications, and security applications, a wide variety of FPUs have been proposed. Despite significant advancements, however, there is still no systematic description of the design space, optimizations, and trend for RISC-V FPUs. This paper addresses this void and provides a detailed overview about RISC-V FPUs from different angles. First, the presentation discusses the floating-point formats and new numerical systems like FP8, bfloat16, Posit, and Block floating-point and their importance in terms of energy and accuracy. Second, the evolution of RISC-V FPUs from scalar to trans precision and vectorised pipelines, and special-purpose hardware for particular applications. Third, integration strategies such as tightly coupled, moderately coupled, loosely coupled, and vector-lane distributed FPUs are compared in terms of performance, modularity, and energy efficiency. Finally, the use of RISC-V FPUs in areas such as AI/ML, IoT, communications, graphics, and cryptography is examined. The paper concludes by identifying open challenges and emerging trends, such as heterogeneous FPUs, chiplet-based integration, and security-aware floating-point designs. Furthermore, RISC-V-enabled multi-FPGA platforms for event-driven Spiking Neural Network (SNN) simulation broaden the applicability of RISC-V floating-point and neuromorphic co-design toward ultra-low-power edge intelligence.
Keywords: floating-point Unit (FPU), RISC-V, Instruction Set Architecture (ISA), floating-point Formats

1. Introduction

A Floating-Point unit (FPU) is a specialized hardware module designed to speed up arithmetic operations on real numbers, as defined by the IEEE 754 standard. General-purpose processors can execute Floating-Point (FP) arithmetic using software libraries, but this comes at a very high-performance and energy cost, particularly when the task to be executed is numerically intensive . These operations include addition, subtraction, multiplication, division, and square root calculations for formats like single precision (FP32), double precision (FP64), and increasingly, low-precision formats such as FP16, bfloat16, which are popular in AI and edge computing. This is often resolved by implementing FPUs as hardware accelerators either tightly coupled to the processor pipeline or as separate coprocessors interfaced by a hardware-specific mechanism.

The Reduced Instruction Set Computer (RISC-V) Instruction Set Architecture (ISA) is an open, royalty-free and extensible ISA developed at the University of California, Berkeley. It is designed to support a variety of applications . RISC-V's design philosophy emphasizes modularity, simplicity, and extensibility. This is one of the reasons why this approach is different from proprietary ISAs, which are limited by backward compatibility and licensing concerns. The base integer ISAs (RV32I, RV64I, and RV128I) cover a core set of operations, to which standard extensions can be added to adapt the ISA for specific domains . Some of the common standard extensions include M (Integer), A (Atomic), F/D/Q (floating-point), and V (Vector) [4].

Custom extension mechanism is one of the main strengths of the RISC-V architecture. This feature enables designers to add domain-specific instructions, features, or accelerators without impairing the base architecture. These

customizations usually happen through standard or open interfaces, like TileLink or Rocket Custom Coprocessor (RoCC). These interfaces help integrate the base RISC-V ISA core with hardware modules like FPUs, Artificial Intelligence/Machine Learning(AI/ML) accelerators, and cryptographic engines . Overall, this flexibility has led to the adoption of RISC-V in academia and industry prototyping. The base RISC-V ISA is designed to be minimal, but the growing demand for energy-efficient and high-performance computation for media, scientific and machine learning applications has brought to light the need for efficient floating-point (FP) arithmetic. The main instruction set (ISA) of RISC-V supports optional extensions for floating-point operations (F, D, Q) and provides support for general-purpose floating-point execution in scalar pipelines, which may have suboptimal performance or energy consumption in embedded and low-power devices [4]. To overcome these restrictions, FPUs are used as dedicated units of hardware designed to do floating-point calculations. These FPUs can either be tightly coupled to the main pipeline or accessed through standard interfaces such as RoCC or TileLink, which helps minimize the energy consumption, latency, and throughput of the main pipeline for FP intensive tasks.

This modularity of the RISC-V ecosystem is especially well suited for including FPUs with the main processor. Designers can develop application-specific coprocessors (e.g., for digital signal processing, graphics, control systems or AI workloads) and still support compatibility with software toolchains thanks to the open ISA [8]. Moreover, several research and industry initiatives have shown that floating-point accelerators can be successfully integrated into RISC-V cores, and that these cores can be orders of magnitude more area- and power-efficient than their scalar counterparts [9]..As domain-specific workloads increasingly rely on mixed-precision and floating-point intensive operations, the demand for optimized FPUs in custom RISC-V cores continues to grow, especially in edge-AI, scientific computing, and real-time control applications .

The paper presents a comprehensive literature survey on the design, development, and deployment of FPUs within the RISC-V ecosystem. The main contributions of this paper are:

- The survey highlights the development of RISC-V FPUs from early scalar units to advanced domain-specific accelerators like Heterogeneous Chiplets, FAUST, Concurrent execution of mixed OPerations on Integer and Floating-Point Threads (COPIFT), Spatz, and Marian. It also outlines their uses in AI/ML, cryptography, graphics, Internet of Things (IoT), and High-Performance Computing(HPC).
- We present a comprehensive taxonomy of RISC-V FPU integration styles, highlighting their advantages and disadvantages.
- We survey different floating-point formats and emerging number systems, highlighting their role in domain-specific acceleration for AI, HPC, and IoT.
- We identify design challenges and highlight some possible future opportunities in the field of RISC-V FPU design.

The rest of the paper is organized as follows: An overview of the RISC-V ISA is given in Section 2. The various floating-point formats, their extensions, and common application domains are covered in Section 3. Section 4 then provides an overview of the evolution of the RISC-V FPU. Section 5 presents a discussion of the various integration strategies utilized to couple RISC-V with an FPU/coprocessor. Recent Research on RISC-V FPU is included in Section 6. Design trade-off difficulties using important optimization approaches are discussed in Section 7, while Open Research difficulties in RISC-V are reviewed in Section 8. Finally, section 9 concludes this paper

1.1 RISC-V Instruction Set Architecture (ISA)

The RISC-V ISA is an open-source, modular, and extensible ISA designed to provide scalability across diverse computing domains, ranging from embedded IoT devices to high-performance computing (HPC) systems. According to the latest version of the RISC-V specification released in 2019, RISC-V ISA can be divided into two categories: non-privileged and privileged, as shown in Figure 1. Unprivileged ISAs are user-level instruction sets that are independent of the operating system and suitable for user applications. They include base integer ISA (RV32I/RV64I/RV128I), floating-pointISA (F/D/Q) complying fully with IEEE-754 standards, compressed instructions, atomic instructions, Control and Status Registers (CSR) access instructions, and vector instructions.

The Privileged ISA defines system-level capabilities for implementing operating systems, hypervisors, and control software in embedded systems. It comprises Machine (M-mode), Supervisor (S-mode), and User (U-mode) levels, with optional Hypervisor (H-mode) extensions for facilitating virtual machine management and nested virtualization . It also handles exceptions, virtual memory translation, context switching, interrupt handling, and system protection through Control and Status Registers (CSRs) . Additionally, it includes memory protection features like physical memory protection and supervisor-level address translation for controlled access between software . The

privileged ISA has been implemented in the real world, for example, in Andes AX45MP SoCs [15], SiFive U74 SoCs [16], and Microchip PolarFire SoCs [17]. Therefore, the Privileged ISA provides a foundation to design scalable, secure and virtualizable processors for RISC-V platforms.

The RISC-V F extension ISA has a dedicated 32-bit FP register file, which supports addition, subtraction, multiplication, division, fused multiply-add (FMA), square root, comparisons, and type conversions instructions [18]. The RISC-V Vector Extension (RVV) is a scalable SIMD (Single Instruction Multiple Data) extension that greatly improves the FP performance's crucial for 5G/6G wireless communications, machine learning (ML), and high-performance DSP applications [19]. Additionally, RVV FP allows for user-defined extensions to speed up the innovation of new features in trans-precision computing and domain-specific accelerators in the system on chip (SoC) design, thereby providing rapid innovation within the system [20].

The open source development and active ecosystem is making RISC-V an attractive option for implementing highly-configurable FPUs, such as HardFloat, FPnew, and MiniFloat [21] which is driving adoption in both academia and industry.

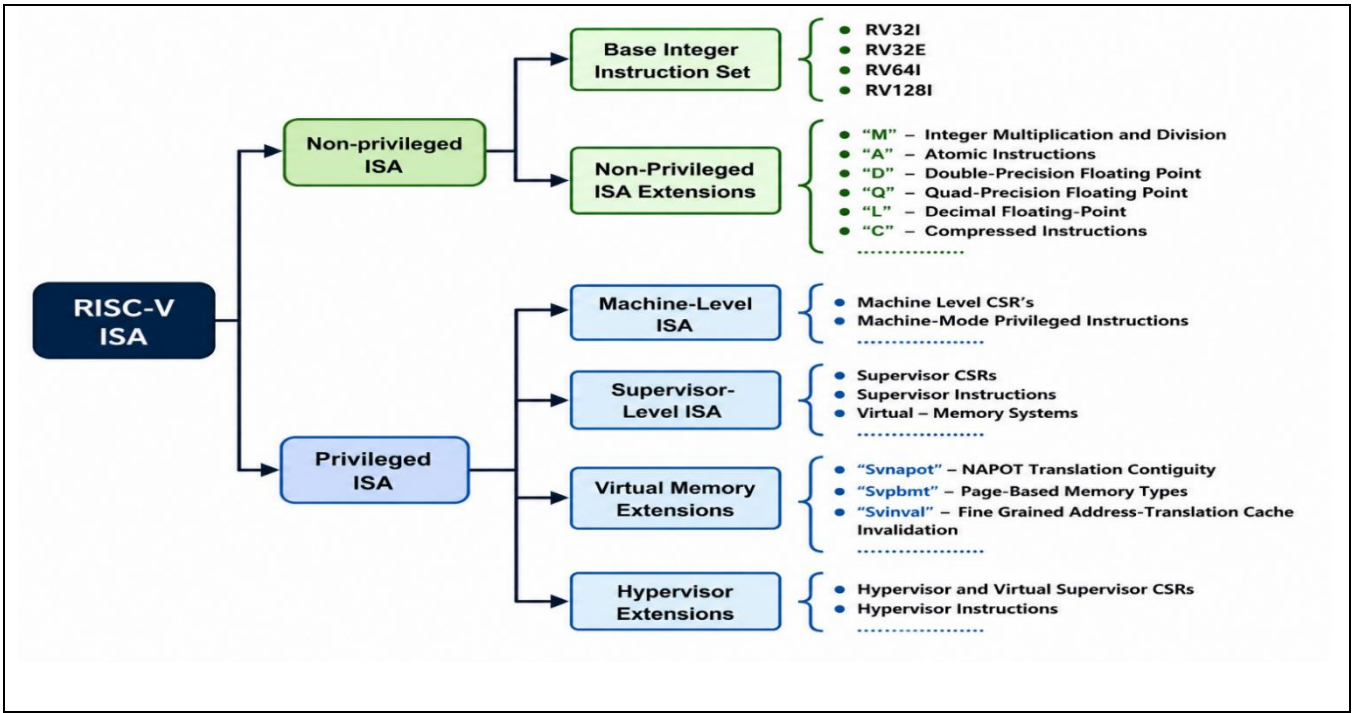


Figure 1: RISC-V ISA , Reproduced from [[22]

1.2 Floating-point formats

FP Formats play a crucial role in determining the dynamic range, precision, energy efficiency, and performance of FPUs in RISC-V SoC architectures. Figures 2 and 3 illustrate the classification of floating-point formats and their corresponding representations, respectively. RISC-V ISA natively supports the IEEE-754 standard formats (FP32, FP64, FP128). It also allows the implementation of reduced-precision formats such as FP16 (half precision), bfloat16, and TensorFloat-32, often used in domains such as ML, Wireless communication (5G/6G), and edge computing, where energy efficiency and computational throughput are important . FPnew and MiniFloat-NN support transprecision computing by Figure 2: Floating-Point Format

classification enabling multiformat execution in the same datapath by dynamically switching between FP64, FP32, FP16, and FP8 to optimize performance-per-watt.

In addition to all the above-mentioned formats, novel representations like Posit, Block Floating-Point (BFP), Logarithmic Number systems (LNS), and Dynamic Fixed Point (DFP) have been used in domain-specific applications that require higher accuracy and low power consumption than the traditional IEEE formats .These developments indicate the transition from fixed-precision FPUs to reconfigurable and multiformat architectures, which are suitable for applications like signal processing, AI acceleration, and HPC.

1.2 Emerging alternative FP formats for RISC-V architecture

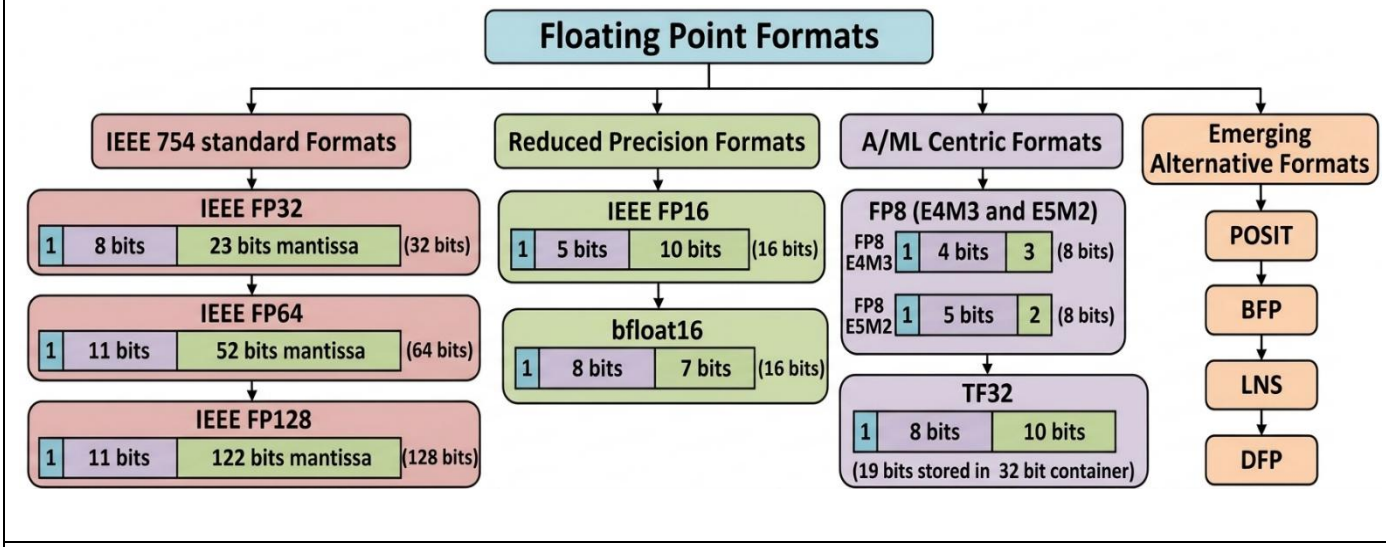


Figure 2:Floating-Point Formats and Representation of the IEEE754 Standard,reduced precision, and AI/ML-Centric FP Formats

Some of the recent studies have discussed alternative FP formats to improve the efficiency, accuracy, and energy optimization of FP operations. The Posit format proposed by Gustafson offered higher precision and dynamic range than IEEE-754 FP formats. It utilizes a tapered precision representation, which is accurate for both small and large numbers, and consumes less power.

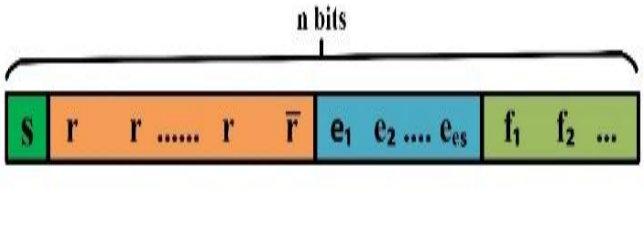


Figure 3: The Basic Structure of an (n, es) Posit Number, where n is the word size and es is the exponent field size [28].

Malathi et al implemented RISC—V Posit FPU and demonstrated up to 3 times better energy efficiency than conventional FPUs in AI/ML applications . As shown in Figure 3, an (n, es) posit number consists of four parts: a sign bit (s), the regime field (r), the exponent field (e), and the mantissa field (f). The es controls the trade—off between dynamic range and numerical precision. The regime field is encoded by a run—length method, leading to the boundary between exponent and mantissa being changeable, which is the principal difference between posit and float .

BFP arithmetic is gaining prominence in RISC—V accelerators for 5G/6G wireless baseband, Multiple-Input, Multiple-Output (MIMO) systems, and Fast Fourier Transform/ Discrete Fourier Transform (FFT/DFT) computations. BFP shares a common exponent for a block of data, significantly reducing storage and power costs while retaining sufficient precision for vectorised operations, making it

highly suitable for Digital Signal Processing (DSP)—heavy **S**ystem on a Chip (SoCs) . Figure 4 shows examples of the BFP number format representation. Flex point proposed a BFP format with a 16—bit mantissa (m=16) and a 5—bit shared exponent across an entire tensor. improving the efficiency of DNN training as a middle ground between fixed and floating-point.

LNS represents numbers as logarithms, converting multiplication and division into simple addition and subtraction, thus improving throughput in communication systems and neural network accelerators. Recent studies show LNS that LNS—based FPUs integrated into RISC—V architectures achieve up to 50% area savings while

maintaining competitive accuracy for trans precision applications. Collectively, these emerging formats present a promising direction for designing energy—efficient and domain—optimized FPUs in future RISC—V SoCs.

An FP number has one exponent for each variable, whereas DFP shares a single exponent with multiple variables as shown in Figure 5. It represents the number with greater accuracy, gives better performance for floating—point operations, and reduces arithmetic errors. Table 1 gives the details of different FP formats used, their ISA extensions, and their usage in different applications.

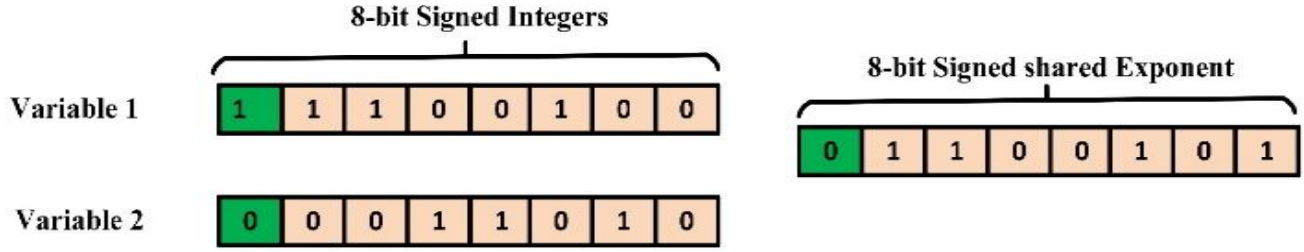


Figure 5: Example of 8-bit Dynamic Fixed Point (DFP8) [33]

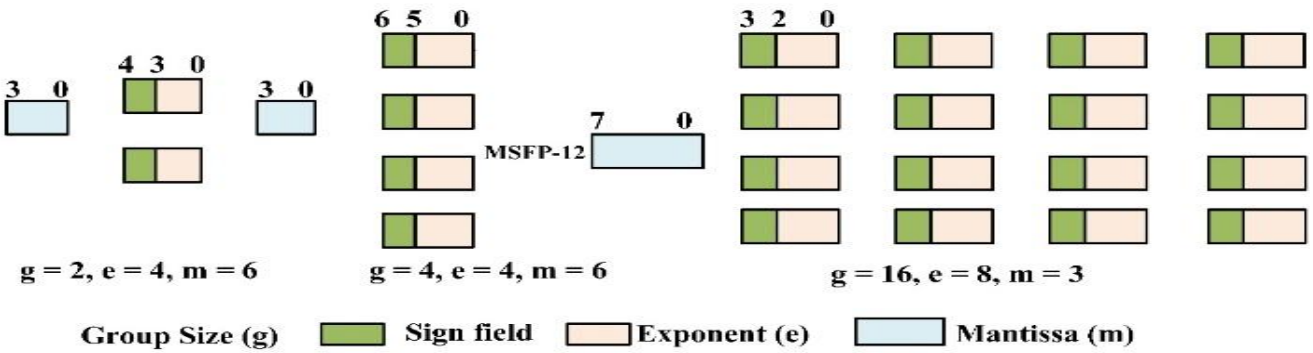


Figure 4: Examples of BFP FP number format [29]

Table 1: Floating-point number formats for RISC-V

Formats	References	RISC-V Support	Strengths	Weaknesses	Applications
FP32		F-extension	Supports IEEE-754 compliance	Higher energy cost for edge devices	General Purpose FP Operations, ML, GPU.
FP64		D- extension	Standard HPC format, High precision	Large area and power	HPC, scientific computing .
FP16		Custom F/ RVV	Energy efficient, widely used in AI/ML	IEEE compliance is optional, limited precision.	Edge computing, DSP, AI/ML.
Bfloat16		Minifloat-NN, CVFPU	Good dynamic range, optimized for AI training.	Lower precision for small values.	Google TPUs, AI accelerators.

TF32		Custom ML extension.	Optimized for tensor core, good throughput.	NVIDIA proprietary, limited portability.	AI accelerators, NVIDIA Tensor Core .
FP8 (E4M3/E5M2)		FPnew, Hopper GPU.	Extreme low-precision, high speed.	Not IEEE compliant, Accuracy loss.	AI inference, low-power ML accelerators, NVIDIA Hopper.
Posit		Experimental RISC-V FPU .	Tapered accuracy, high precision per bit.	The tool chain is immature and not standardized.	Edge ML, IoT, neuromorphic system .
BFP		Integrated in RISC-V ML and DSP accelerators.	Block-based exponent, efficient vector storage.	Lower per-value accuracy.	5G/6G baseband MIMO, FFT/DFT.
LNS		DSP extension.	Efficient mul/div as add/sub operations.	Limited adoption, exp/log overhead.	DSP, neural accelerators, wireless baseband
DFP		IoT class custom FPUs.	Efficient SIMD accumulations, reduced errors	Standardization and limited toolchain.	eBPF, AI/ML, SIMD

1.3 Evolution of Floating-Point Units for RISC-V Architectures

The evolution of floating-point units (FPUs) shows a move from latency-focused scalar pipelines to transprecision, vectorized, and domain-specific accelerators. With the invention of the RISC-V ISA with F and D extensions (2011), enabled single and double-precision computation. But with the advent of modern applications such as IoT, communication, AI/ML and HPC, there was a need to make the FPUs more efficient and flexible. FPnew (2020) solved this by adding support for transprecision (FP8–FP64), modular pipelines and SIMD execution, enabling FPnew to deliver energy-proportional computing in embedded and HPC domains [20].

FPnew, FAUST (2023), and COPIFT (2025) furthered the paradigm to vector and dual—issue execution, respectively.Extending the paradigm, FPnew, FAUST (2023) and COPIFT (2025) proposed vector and dual—issue execution, respectively. In FAUST, the RVC (REXE) floating point units are pipelined to meet RVV (REXE) compatibility requirements, and optimized for HPC workloads, and in COPIFT, the floating point units are pipelined to support dual—issue mixed integer—FP scheduling in in—order cores, thus improving throughput and energy efficiency. These are complemented by the Xvpfloat extension, which provides flexible experimental FP formats (FP8, bfloat16) in vector pipelines, directly targeting AI workloads with precision settings.

Coupled designs were further explored in clustered coprocessors. Spatz (2023) and Spatzformer (2024) developed compact vector coprocessors with RVV instructions and added a small area penalty for the control logic of Snitchs [55] [56] with high utilization.

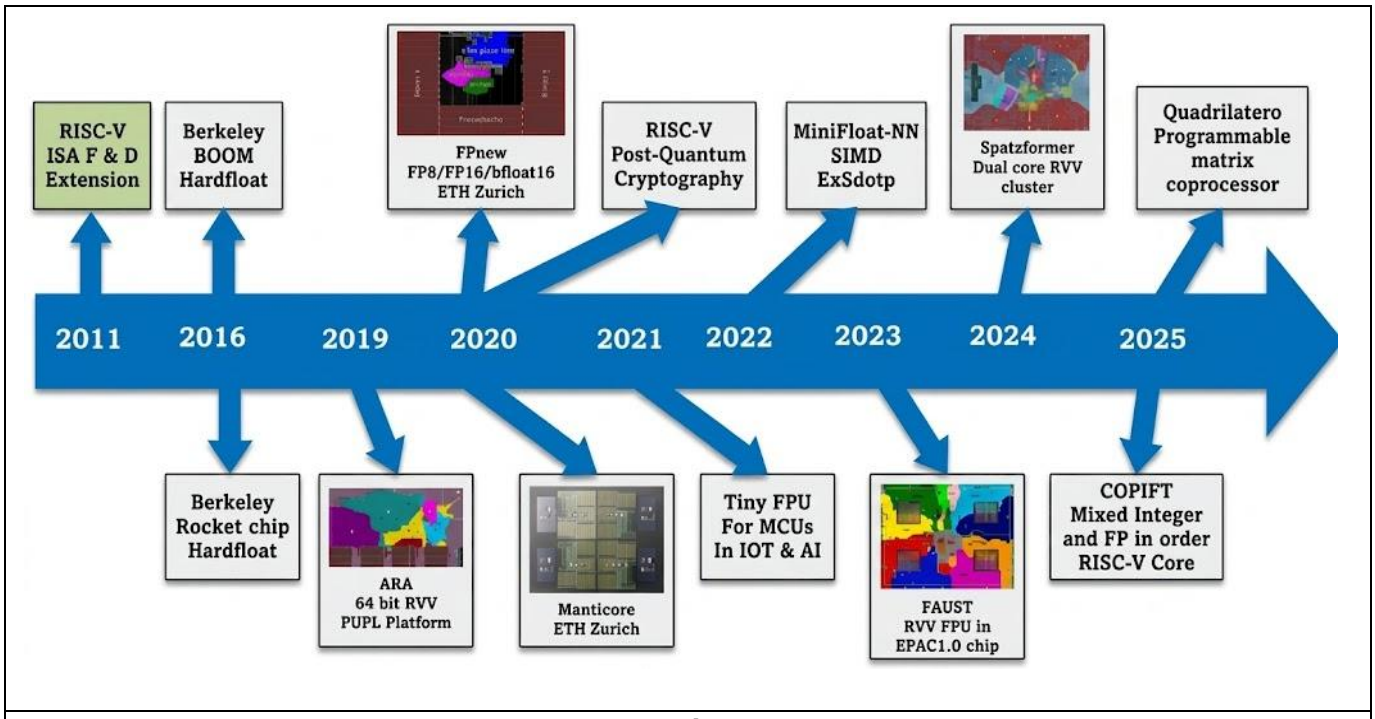


Figure 6: Typical RISC-V FPCs

In larger scales, Manticore (2021) showed a lightweight RV32 controller-based multicore fabric with the ability to keep more than 90% utilization on 4096 cores through the use of Stream Semantic Registers (SSR) and Floating-Point Repetition (FREP) buffers for streaming data along with wide FP pipelines [34].

In the meantime, new FPCs have been brought up by domain specialization. RISQ-V (2020) presented two tightly coupled FP arithmetic accelerators for post-quantum cryptography (PQC) [57] and Quadrilatero (2025) presented a loosely coupled programmable matrix coprocessor for low-power edge AI [58]. Tiny-FPU (2021) and FPUx (2024) provided IEEE-compliant FP32/FP64 support with area and energy savings of up to 100x and 1000x, respectively, for IoT MCUs [59] [60].

The RISC—V floating—point units (FPUs) have evolved from scalar FPUs that follow IEEE standards (RV F/D) to advanced SIMD FPUs with transprecision operations (FPnew), vector pipelines (FAUST, COPIFT, Xvpfloat, etc.), clustered FPUs (Spatz, Manticore, etc.), and specialized domain-specific FPUs (RISQ—V, Quadrilatero, etc.). This evolution shows a definite trend towards the development of domain-specific floating-point coprocessors that are configurable and energy-efficient, and come with lower latency FPUs. The evolution seeks to address the wide range of requirements of IoT, AI, high-performance computing (HPC) and cryptographic applications.

1.4 Integration Approaches

The integration of FPUs/ accelerators to the RISC-V core is broadly classified into five categories, considering trade-offs in performance, area, and modularity of design. In a tightly coupled approach, FPUs are directly embedded into the processor pipeline. The FPU will have its own FP register file and CSR support for IEEE-754 compliance, but it shares the decode and issue logic with the base ISA core. This approach minimizes instruction and data latency at the cost of power and area overhead. Due to its low communication cost, it is widely used in high-performance real-time systems, like FAUST , FPnew , COPIFT , and RISQ V .

In contrast, loosely coupled FPUs are coprocessors or accelerators, integrated with the RISC-V core via RoCC or CFU interfaces. The approach is more modular and more scalable than a tightly coupled approach, but adds additional communication delays. Manticore, Quadrilatero , and Snitch architecture uses a loosely coupled approach with a concurrent focus on configurability and effective offloading of FP workloads. Moderate coupling is the compromise between tight and loose coupling. It offloads FP operations to a nearby coprocessor and often shares memory or instruction streams. This approach reduces the binding between core and FPUs, retaining locality and efficient data reuse. Spatz and Spatzformer are moderately coupled FPUs that share both memory and instruction streams.

Recently, the rise of the RVV has resulted in distributed FPUs within vector lanes. This configuration supports high-throughput FP computations. Finally, the hybrid approach used in Manticore explores chiplet-based or heterogeneous integration, which combines scalar, vector, and specialized FPUs for domain-specific acceleration. Together, these works show a range of integration choices based on the needs of applications in embedded systems, HPC, AI/ML, and security. These integration strategies vary from simple embedded FPUs to extensive, parallel vector datapaths. They show how the RISC-V ecosystem can meet application demands. Table 2 summarizes all the integration approaches with their advantages and disadvantages.

Table 2: Integration approaches of FPUs with RISC-V cores.

Integration	**References**	**Advantages**	**Disadvantages**	**Application Domain**
Tightly coupled	FAUST , FPnew, COPIFT, RISQ-V	Seamless ISA compliance (F/D/Q), low latency, direct access to FP register.	Less modular, higher area, and power.	HPC, AI/ML, real-time systems.
Loosely coupled	FPUx , Quadrilatero, Manticore	Configurable, modular, easy to scale.	Higher communication latency.	Edge computing, IoT.
Moderately coupled	Spatz , Spatzformer	Balance between modularity and performance, with a lesser area than tightly coupled, FPUs can be scaled or reused.	Instruction scheduling complexity, latency is slightly more than tightly coupled and less than loosely coupled.	DSP SoCs, mixed workloads.
Vector-lane	Marian , Ara2	Higher throughput, FP operations are performed in parallel across lanes.	More complex design, verification overhead, and larger silicon footprint.	HPC, AI/ML.
Hybrid	Manticore ,	Combines scalar, vector, and accelerator FPUs; optimized for specific workloads like HPC, cryptography.	Power and interconnection challenges, system-level integration complexity.	HPC, cloud-scale.

1.5 Recent Research on RISC-V FPU

The RISC-V extensible ISA and its ecosystem initiated the development of dedicated FPUs custom-made to domain-specific applications. The domain-specific RISC-V FPUs are implemented by adding different features like transprecision formats, SIMD, vector pipelines, or low-latency execution units. These features help speed up calculations in AI and ML, cryptography, signal processing, scientific computing, and IoT applications. We review several FPU designs and their functions in each area and summarize how features like mixed-precision, vector extensions, and ISA accelerators meet the needs of each domain.

1.6 Artificial Intelligence and Machine Learning

AI workloads such as neural network training and inference have encouraged the design of specialized FP hardware that trades precision for speed and efficiency. AI/ML algorithms often require FP calculations for tasks like training models and performing complex analyses. The combination of FPU and vector extensions results in substantial performance improvement in vector calculations and matrix multiplication, which are critical to neural networks and other AI workloads.

RISC-V FPUs provide the necessary hardware support for these operations, improving accuracy and speed. In recent years, many AI/ML accelerators have adopted smaller FP formats (FP16, bfloat16, FP8, and TF32) because (Deep Neural Network) DNNs can often tolerate reduced precision. Among these, IEEE 754 FP16 and Google's bfloat16 are popularly used in many applications. Table 3 gives a summary of the comparative analysis of RISC-V FPUs in AI/ML workloads.

The comparative analysis shows a clear shift in how RISC-V FPUs are being modified for AI/ML workloads. Early research, like that of Tsai and Lin , demonstrated the efficiency of bfloat16 arithmetic in training neural networks. FPnew also illustrated the benefit of a flexible mixed-precision pipeline, improving upon 33% reductions in energy with no accuracy loss. Designs such as COPIFT and Manticore built on this concept by combining FPUs with cooperative threading or stream-based data delivery. This allowed for nearly perfect floating-point use and significant energy savings compared to Central Processing Units (CPUs) and Graphics Processing Units(GPUs).

Vector-centric processors such as Ara2 and SPEED offer a complementary approach. They use scalable RVV lanes and low-precision formats, down to 4-bit, to balance throughput and efficiency for ML inference. In parallel, transformer-specific computational accelerators - including the pruning-based bfloat16 library from Dequino et al. and the VEXP softmax accelerator from Wang et al. - suggest how FPUs can be co-designed with approximations from the model-level to minimize latency and energy consumption with preserved accuracy. Recent proposals such as MXDOTP show a trend towards using microscaling formats in the field such as FP8. These formats are supported by special hardware support for dot product accumulations which are used in transformer workloads.

Figure 7 shows the detailed architecture of tiny Transformers built on PLUP (GAP9). Figure 8 shows the architecture of the snitch cluster with Floating-Point Repetition (FPR) and Stream Semantic Registers (SSR) for managing data access with minimal software overhead. It also shows the order of the operations performed in VEXP for exponential approximation..

All of these advancements point to the transition of FP32 to reduced-precision formats such as FP16, BF16, and FP8. The combination of this change, and innovations in streaming, clustering and pruning, puts RISC-V FPUs more and more at the center of the investigation of how to accelerate AI and ML using a chip that can deliver power savings.

In addition to traditional deep neural networks, more recent neural networks like spiking neural networks (SNNs) are another promising avenue for ultra low power neuromorphic computing. With modularity and extensibility, RISC V enables SNN acceleration.. Liang et al. (2025) [93] proposed an event driven multi FPGA cluster for quantitative SNN simulations, achieving scalable and efficient neuromorphic processing. Although this work focuses on event scheduling and Field-Programmable Gate Array (FPGA) resource management, it reveals a valuable trend: RISC V floating point units can be closely integrated with SNN neuron and synaptic units to balance numerical precision, dynamic power consumption, and real time performance. Such co design of RISC V FPU and event driven neuromorphic architectures remains an important open direction for future research.

Table 3: Comparative analysis of RISC-V FPUs and their extensions in AI/ML workloads

Reference	FP formats	Target AI/ML task	Features	Performance
Tsai & Lin [65]	Bfloat16	Fully connected NN training .	Exploit activation sparsity, forward and backward propagation	102.43 GOPS @ 200 MHz, more energy efficient than GPU, enabling on-chip training with 6.44 W.
COPIFT [54]	FP32 and INT	Common AI kernels (activation, reduction).	Cooperative parallel INT & FP threads, Dual issue.	Approximately 1.47 times faster, 1.37 times energy efficient than scalar RV32G.
Manticore [34]	FP32, FP16	Data-parallel ML kernels .	Chiplet design with wide FPU, FREP & SSR streaming.	It is five times energy efficient than CPU & GPU, 90% FP utilization .
Ara2 [65]	FP32 (lanes)	Vector ML workload.	2-16 lanes, Open source RVV vector FPU.	95% utilization, 8X2-lane cluster, 3x performance, 1.5x energy efficiency vs single lane core.
Speed [66]	4 – 16-bit FP formats	DNN inference.	Custom RVV extensions for low-precision ML.	Throughput-287.41 GOPS, energy efficiency-1335.79 GOPS/W at 4-bit precision, area efficiency of 2.04x (16-bit) & 1.63x (8-bit) vs Ara.

Dequino [36]	Bfloat16	TinyML transformers.	Transformer pruning, optimized library.	220x speed, MobileBert (~94%) and TinyViT (~57%) reduced memory footprints. A TinyLLAMA SLM on a microcontroller achieved a throughput of 1,219 tokens/s, with an average power consumption of 57 mW.
Wang [67]	BF16	Softmax acceleration in the transformer .	Snitch cluster and FREP/SSR, exponential approximation.	~5.8x latency reduction, ~3.6x energy reduction, negligible accuracy loss.
MXDOTP [68]	FP8 class (MXFP8)	Transformer dot product.	ISA extension for Microsoft FP8 dot product.	High utilization for block-structured dot product; software modeling integrated into Spike.

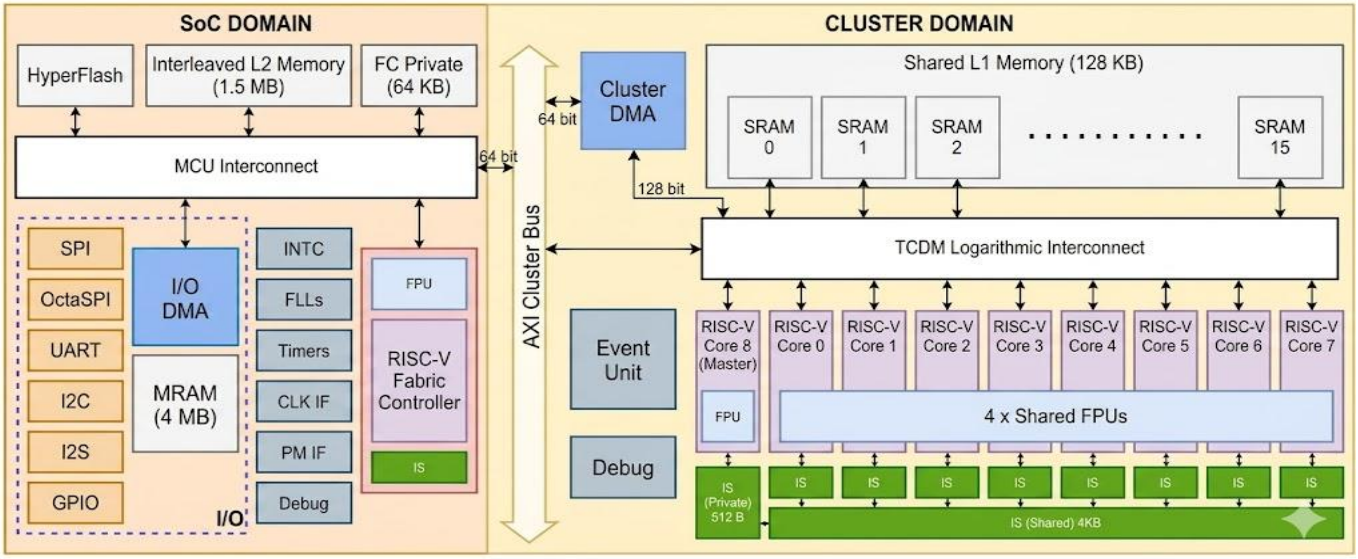


Figure 7: Architecture of PULP paradigm (GAP9) used in Tiny Transformers [36]

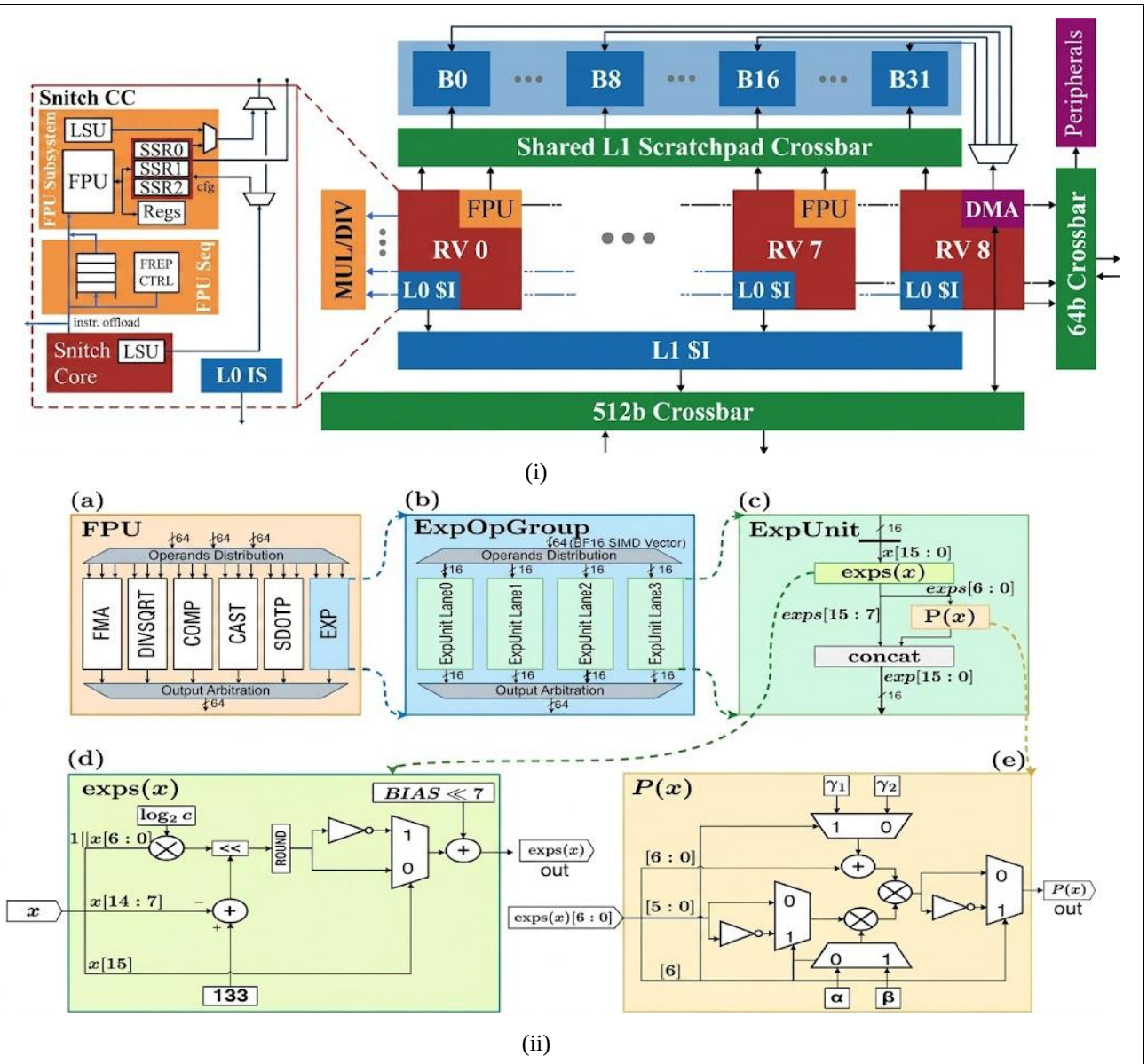


Figure 8 : (i) Architecture of snitch cluster with FREP and SSR used in VEXP, (ii) Block diagram of sequence of operations performed for exponential approximation in VEXP [67]

2. IOT and Embedded Systems

The use of lightweight RISC-V cores in processors tailored for IoT end-node devices is on the rise. As the range and complexity of these applications grow, there is an increasing demand for processors that can handle floating-point operations. This poses a significant challenge because most lightweight RISC-V cores are integer cores lacking a floating-point unit (FPU). This limitation makes it difficult to design processors optimized for applications that require floating-point operations concurrently with integer operations.

IoT devices typically need data processing and wireless communication. However, these devices struggle with a lack of computing power, memory, and energy resources. Table 4 summarizes the comparison of RISC-V FPUs in IoT and embedded system areas. This comparative analysis of RISC-V FPUs for IoT applications shows a common theme: the tradeoff between floating-point capability and energy and area limits.

Designs like FPUx [61] use hybrid pipelines with simplified bus interfaces to meet the real-time needs of cloud and fog-enabled IoT systems while ensuring compatibility with microcontroller architectures. Figure 9 shows the architecture of FPUx, which employs a hybrid pipeline and state-machine design to maximize throughput while maintaining a blocking execution model to avoid data conflicts. FPnew, when implemented on the GAP9 SoC, proves the merit of transprecision support in enabling the adaptive use of FP32 through to 8-bit formats for DSP and mixed-precision analytics, thus saving energy as well as silicon cost.

Table 4: Comparative analysis of RISC-V FPUs in IoT and the embedded domain

Reference	FP formats	Target IoT task	Features	Performance
PUx [59]	FP32	Cloud/fog-enabled IoT MCUs.	Hybrid pipeline & state machine; easy lite interface between MCUs and FPUx.	High throughput with significant reduction in energy consumption.
FPnew (GAP9 integration)	FP8 – FP32	IoT SoC, DSP, edge ML.	Transprecision FPU; loosely/ moderately coupled to core.	precision reduces energy; flexible casting saves area; validated in 22 nm GAP9.
Tiny-FPU [60]	FP32,FP64	Cost-constrained IoT MCUs	Designed for compactness and efficiency, the Tiny-FPU employs optimization strategies that balance performance with reduced hardware complexity. It can be paired with lightweight microcontrollers and leverages software emulation to further minimize implementation cost.	When integrated with the Snitch core, Tiny-FPU achieves speedups of approximately 18.5× (double precision) and 15.5× (single precision) over software-emulated floating-point operations. (Adapted from [62].)
Florian [69]	FP32, FP64	Multicore IoT Processors.	External lightweight FPU + RISC-V integer core with low-power multicore architecture.	97.6 % energy efficiency improvement in quad-core prototype; sustains IoT FP kernels.
Marsellus [70]	FP + ultra low-bit accelerators	Heterogeneous AI-IoT SoC.	Cluster of 16 RISC-V DSP cores + MAC and LOAD operations; DNN Accelerator.	180 Gop/s or 3.32 Top/s/W on 2-bit precision arithmetic in software, and up to 637 Gop/s or 12.4 Top/s/W on hardware-accelerated DNN layers.

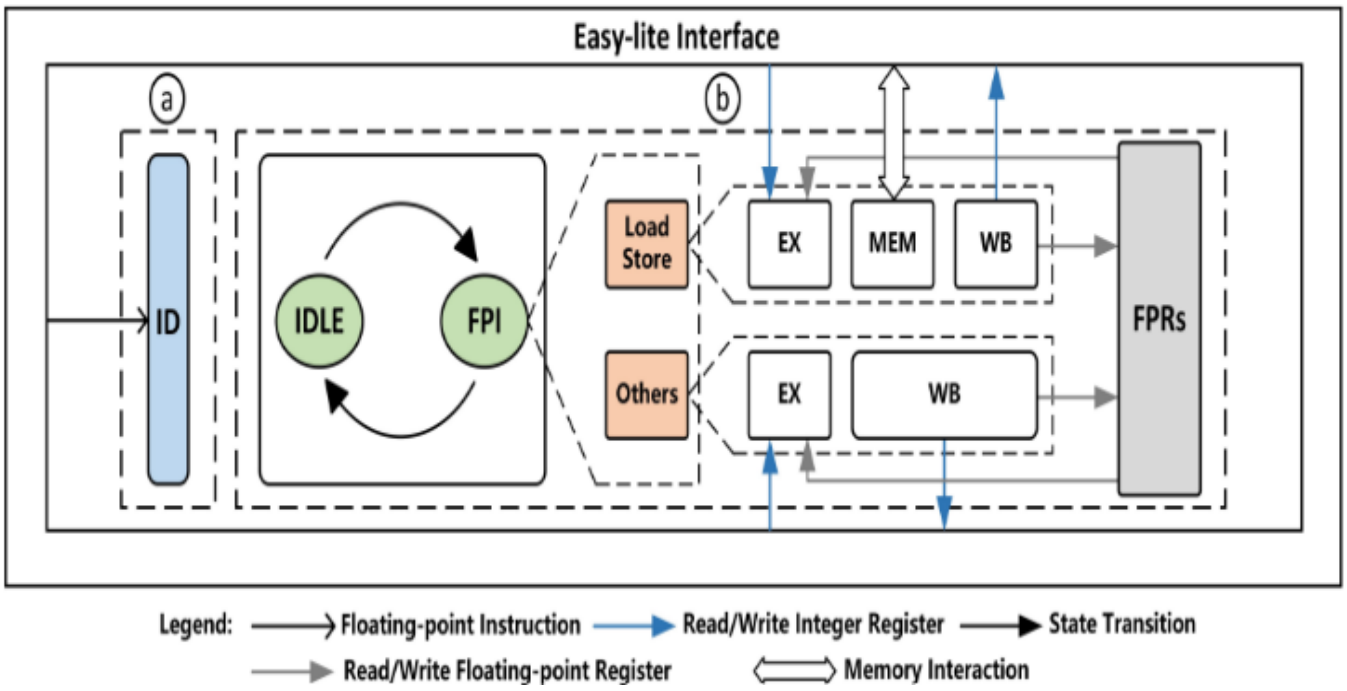


Figure 9: FPUx Architecture for IOT application [59]

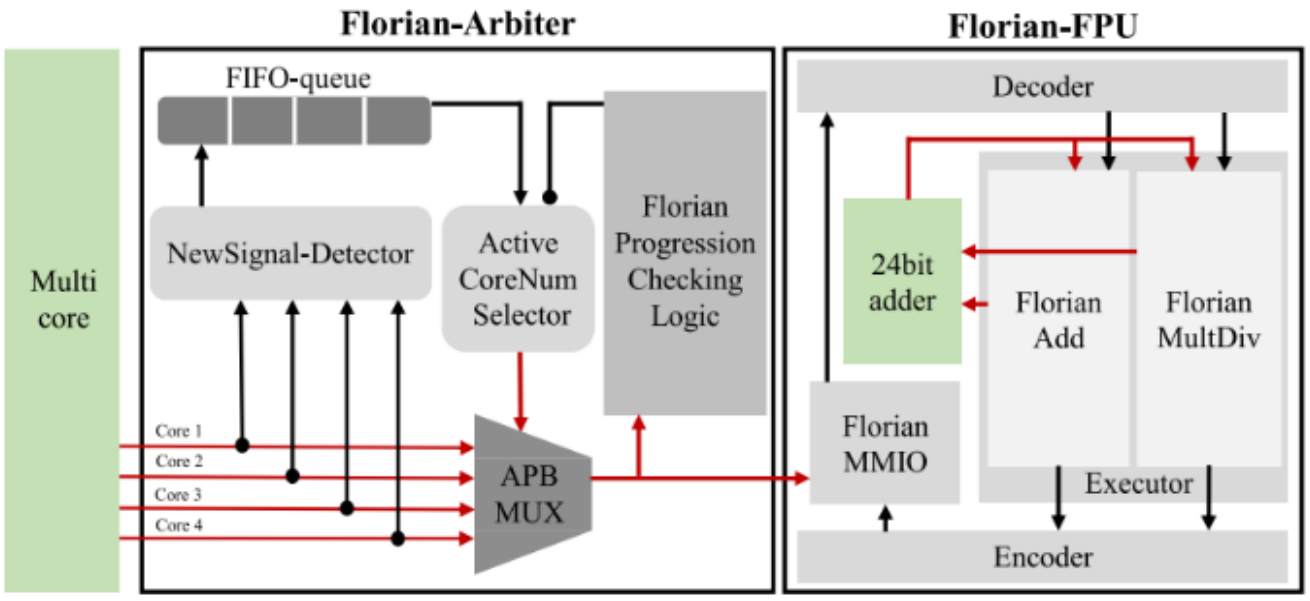


Figure 10: Florian: a lightweight FPU for a Multicore processor [69]

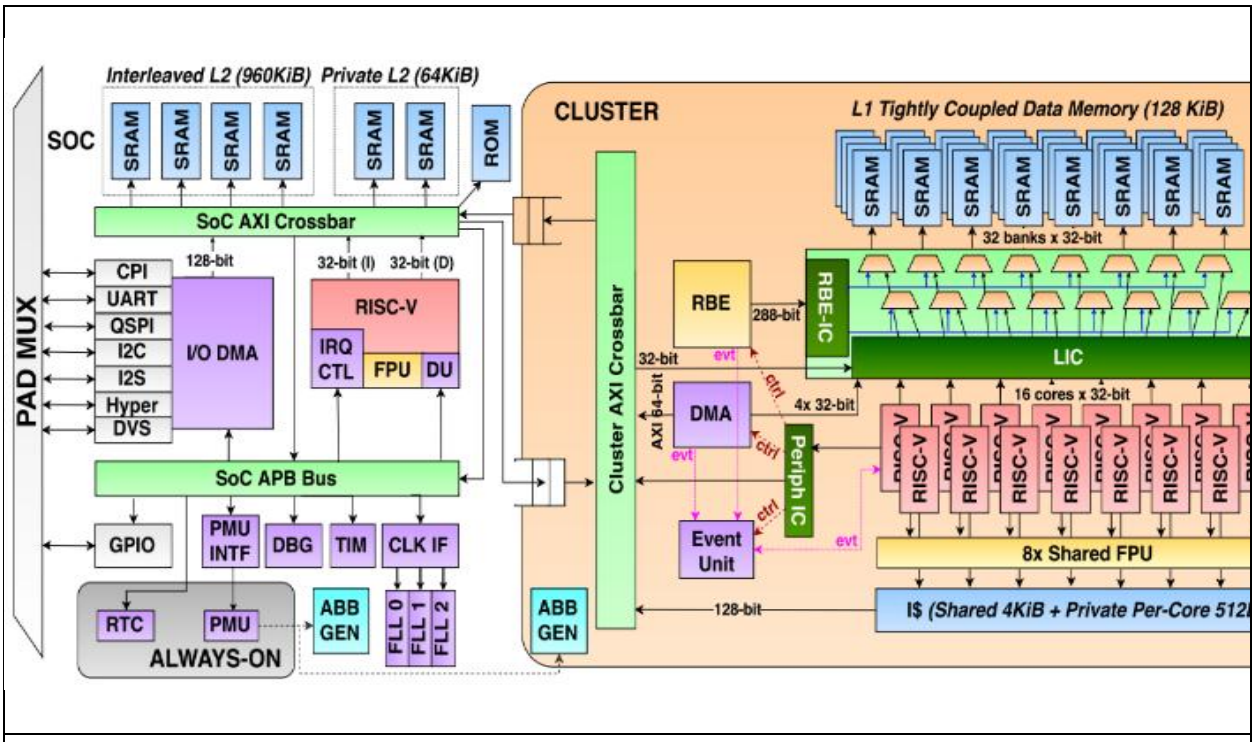


Figure 11: Detailed architecture of the MARSELLUS SOC [70]

Tiny-FPU shows a different approach, where reusing the datapath and using multi-cycle execution focus on saving space. This strategy cuts hardware costs by up to half and boosts speed to more than ten times that of software emulation. At the multicore level, Florian [69] introduces a shared lightweight FPU that significantly improves energy efficiency by removing unnecessary hardware from across cores. This shows that IoT workloads can manage resource sharing without a significant decline in performance.

Figure 10 shows the structure of the Florian architecture. The architecture comprises two main components: the Florian-FPU and the Florian-Arbiter. Finally, Marsellus demonstrates a system-level approach that involves floating-point support integrated with ultra-low-bit AI accelerators, enabling heterogeneous processing in tens of milliwatts. Figure 11 shows the architecture of MARSELLUS SOC designed for IoT applications. It has a microcontroller unit and AI accelerators for parallel and heterogeneous computing.

Collectively, these studies show that IoT FPUs are shifting towards lightweight, shared, or transprecision designs that maximize efficiency while providing enough floating-point capability to support analytics, signal processing, and edge intelligence in low-power settings.

3. Communications

Wireless communication technologies have become an everyday part of our lives. The increase in the number of wireless devices and their diversity has further pushed the development and standardization of 5G wireless communication systems. The soon-to-arrive 6G technology is expected to focus on very low To meet these requirements, Application-Specific Instruction-Set Processors (ASIPs) are gaining attention due to their ability to balance flexibility, efficiency, and affordability in communication hardware. Among available architectures, RISC-V provides designers with a customizable and open platform that enables innovation in the development of communication-oriented ASIPs. Table 5 compares recent RISC-V FPU-based designs, highlighting FP formats used, architectural features, and reported performance results in 5G/6G baseband and positioning applications.

The communications field shows that RISC-V FPUs are becoming key to next-generation wireless baseband and signal-processing tasks. Early studies using RISC-V ISA extensions [71] [72] [73] focused on DSP kernels, error correction, and radio resource management. These works demonstrated that open ISAs can act as flexible building blocks for communication ASIPs. Recent designs focus on tightly coupled floating-point accelerators that more

directly target the throughput and latency requirements of 5G and 6G. Castañeda et al. [74] introduced PULPO, a programmable FP baseband accelerator made with 22FDX. It works with an SoC that hosts multiple RISC-V cores, achieving efficient massive MU-MIMO baseband processing while remaining programmable. Figure 12 shows the detailed implementation details of the PULPO baseband-processing accelerator and its host SoC.

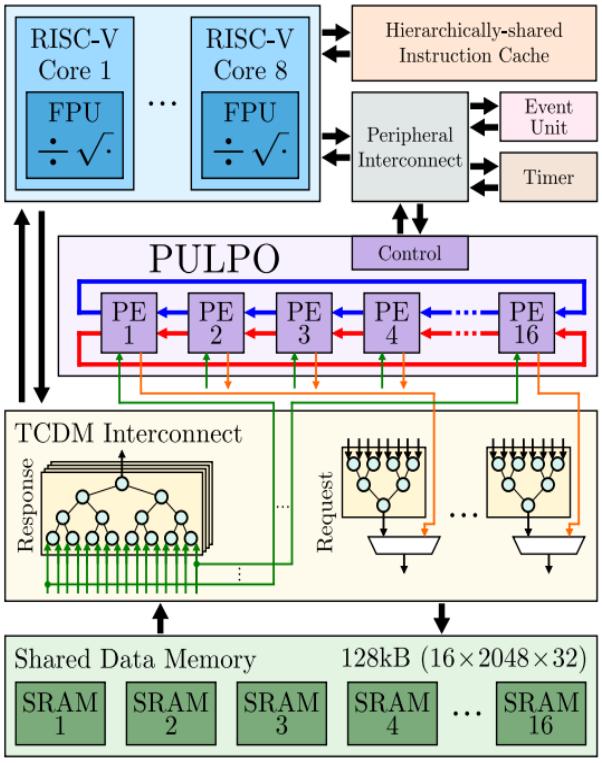
Attari et al. [75] developed this idea by proposing an FP ASIP with SIMD instructions and a dual-accelerator framework. This framework combines a systolic array for matrix-heavy MIMO precoding and detection with a CNN accelerator for positioning tasks. A dedicated systolic array accelerator handles matrix multiplications, which are key to MIMO detection and precoding. Meanwhile, a convolutional neural network (CNN) accelerator, along with its vector memory subsystem, supports effective execution of positioning tasks.

Together, these studies explain why floating-point arithmetic is becoming more popular than fixed-point in communication systems.

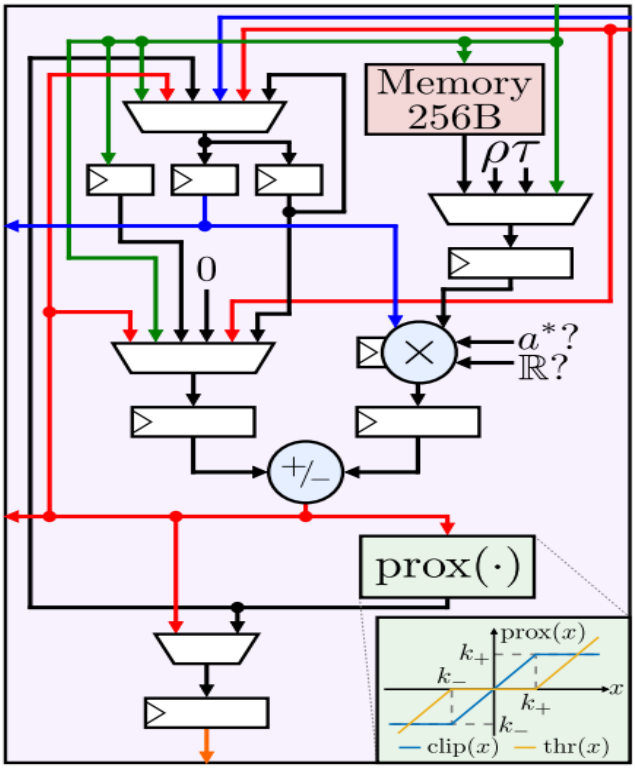
It simplifies algorithm design, reduces development complexity, and scales better for the diverse AI-driven workloads expected in 6G. This trend indicates that future RISC-V communication platforms will use FPUs not only for traditional signal-processing tasks but also as the foundation for heterogeneous baseband engines that include AI accelerators, ensuring both flexibility and high efficiency.

Table 5: RISC-V FPUs for Communication workloads

Reference	**FP formats**	**Communication task**	**Features**	**Performance**
PULPO [74]	FP32	Massive Mu-MIMO baseband processing.	Programmable baseband accelerator in 22FDX; tightly coupled to SoC with 8 PULP RISC-V cores.	High throughput baseband engine; integrated with multicore SoC, validated on silicon.
Attari et al. [75]	FP32 + SIMD	Communication + positioning task (MIMO, CNN)	FP ASIP with SIMD instructions; dual accelerators: systolic array for MIMO matrix operation & CNN accelerator with vector memory.	Meeting 5G/6G low-latency and adaptability needs; efficient precoding & positioning; reduced design complexity vs fixed-point.



a.PULPO-SoC system overview



b. PULPO Processing element(PE)

Figure 12: PULPO baseband-processing accelerator and its host SoC [74]

Graphics and Image Computing

Recent research shows that RISC-V FPUs are increasingly used in graphics, image processing, and GPU-like pipelines. RISC-V-based floating-point accelerators are designed to enhance graphics computing performance through efficient matrix operations and parallel processing techniques. Modern graphics workloads, including 3D rendering, texture mapping, pixel shading, and image transformations, heavily rely on floating-point intensive computations. Table 6 compares recent RISC-V FPU-based designs in the graphics and image computing domain, highlighting FP formats used, architectural features, and reported performance results.

Table 6: Comparative analysis of RISC-V FPUs in the graphics and image computing domain

Reference	FP formats	Target graphic task	Features	Performance
RV32X extensions [76]	FP32	Compact rendering operations like interpolation, perspective correction, and pixel math.	Custom Isa extension for graphics rendering.	FP central to pixel operations; shows feasibility of RISC-V graphics ISA.
Quadrilatero [58]	FP32	Graphics kernels like MAtMul, 3D rendering, image transformation.	Matric-centric ISA & systolic-array FPU; reduces register access latency and reuses data.	3.87x throughput, 77% better area efficiency, 15% less energy vs Spatz RVV.
Vortex GPGPU [77]	FP32	3D graphic acceleration.	Open source GPGPU with RVV-based SIMD pipeline.	Demonstrates the feasibility of GPU-like accelerators on RISC-V.
Ventus GPGPU [78]	INT8, FP16, FP32	Graphics & AI workloads.	Dual datapath (scalar & vector) with tensor cores; supports matrix precision.	High throughput, scalable, energy-efficient, flexible precision across tasks.

Custom FPOW instruction [79]	FP32	Image preprocessing (gamma correction).	Added FPOW instruction to RV32IMF; partial logic sharing with FPU.	Reduce hardware overhead; faster gamma correction; reduction in computation time – 58%, code size reduction- 77.26%.

Recent developments show how RISC-V FPUs are evolving to meet the needs of graphics and image computing, from ISA-level extensions to full GPGPU implementations. At the ISA level, RV32X introduced compact rendering instructions that leverage floating-point operations for interpolation, perspective correction, and pixel math, stressing the importance of FP in graphics primitives [80]. At the architecture level, Quadrilatero shows the advantages of a matrix-focused FPU design. It merges a programmable MatMul engine with a systolic-array FPU to address bandwidth issues in RVV. This approach achieves nearly 4× the throughput and offers significant gains in area and energy compared to Spatz.

In parallel, research into RISC-V GPGPUs has shown promise. Vortex validated a scalable open-source GPU-like pipeline. Meanwhile, Ventus advanced the state of the art with a dual datapath (scalar + vector) design improved by tensor cores that support FP32, FP16, and INT8 mixed-precision workloads for rendering and AI tasks. At the microarchitectural customization level, Liu et al. introduced a dedicated FPOW instruction to speed up gamma correction. This approach partially reuses FPU logic to enhance throughput without significant hardware costs.

Figure 13 shows the details of the microarchitecture of the Vortex GPGPU to support 3D graphics. Figure 14 shows the implementation of a custom FPOW instruction to speed up image processing. The integration of a dedicated FPOW, partially sharing logic resources with the existing FPU, reduces hardware overhead while enhancing computational throughput.

Collectively, these works suggest that RISC-V FPUs for graphics are being developed in three main areas: (i) ISA extensions for compact rendering, (ii) matrix or tensor-focused pipelines for high throughput, and (iii) lightweight custom instructions for specific image-processing kernels. This variety shows that RISC-V FPUs can scale from embedded shader accelerators to open-source GPGPUs for real-time graphics systems.

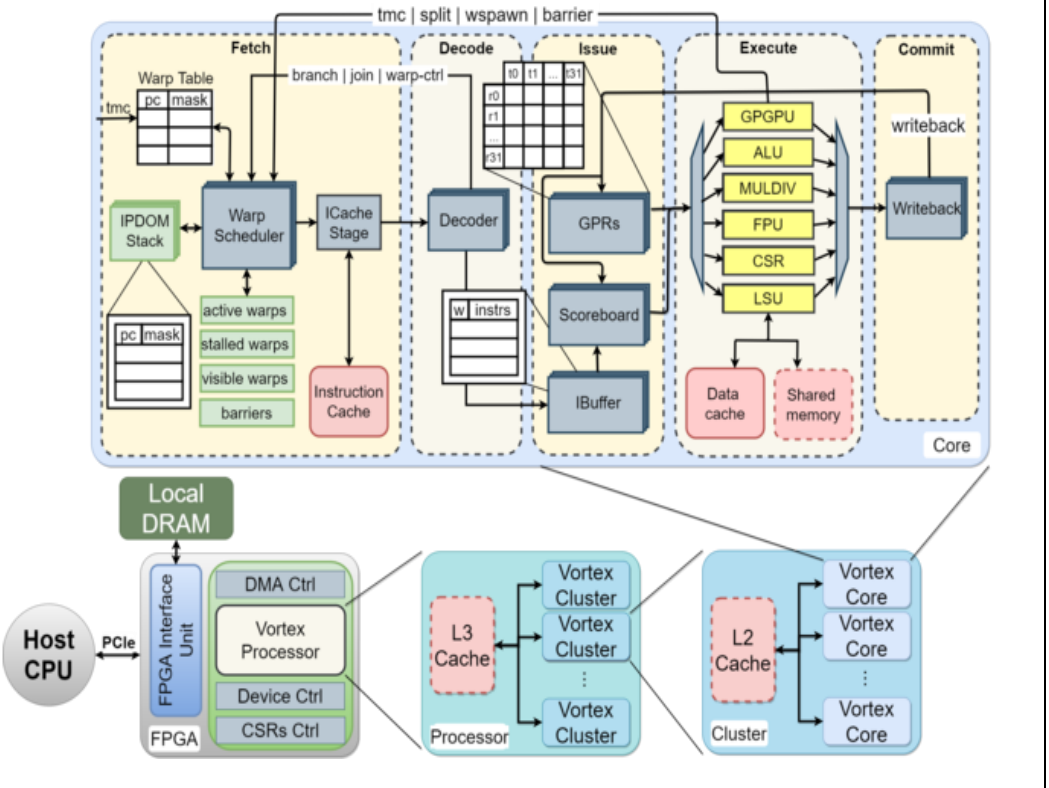


Figure 13: Microarchitecture of Vortex GPGPU [77]

Security

The number of networking devices worldwide is rapidly increasing due to the development of 5G and IoT, though many security concerns persist. Researchers have dedicated substantial effort to creating security solutions addressing the security risks associated with RISC-V. Table 7 gives the comparative analysis of recent papers on RISC-V FPUs' contribution in security domain.

Security research on RISC-V FPUs underscores both their potential as enablers of cryptographic acceleration and their risks as microarchitectural attack surfaces. LU Tao et al. [80] create a comprehensive framework for RISC-V security strategies, including countermeasures for side-channel leaks and control-flow integrity within a secure architectural design. Marian [62] builds on this by illustrating the secure integration of Floating Point Units (FPUs) and vector lanes with a cryptographic pipeline that is decoupled. By enforcing data-independent execution latency (Zvkt), Marian achieves order-of-magnitude performance gains on Advanced Encryption Standard (AES), Secure Hash Algorithm 2 (SHA-2), ShangMi 3 (SM3), and ShangMi 4 (SM4) while maintaining constant-time behavior, showing how floating-point/vector subsystems can actively enhance security.

By contrast, Gerlach et al [81] explain the misappropriation of the same microarchitectural features. Long latency floating point instructions that finish execution as single operations give rise to the CycleDrift attack. Privilege level access of counters, along with vendor specific caching operations, make possible attacks for recovering AES/RSA keys or circumventing KASLR (Kernel Address Space Layout Randomization). The SpecLFB defense [22] focuses on speculative cache side channels. It delays the execution of speculative load instructions that are most likely to cause information leaks. This countermeasure is effective to combat exploits of the Spectre class with low hardware costs. However, the authors recognize the challenge of excessive speculative leakage through FPUs, TLBs, and vector pipelines.

Table 7: Comparative analysis of RISC-V FPUs in security domain

Reference	**Security Focus**	**Features**	**Implications and results**
Lu Tao [80]	Survey on RISC-V security issues.	Summarized defenses: side channel resistance, control flow integrity, and reducing attack surface.	Provides a taxonomy of RISC-V security methods, establishes baseline challenges.
Marian [62]	Secure vector cryptography .	Decoupled 3-stage crypto unit; constant-time execution (Zvkt); integrated with FPU & vector datapath.	12 – 118x cycle reduction vs OpenSSL C refs; crypto unit adds ~6.4% area in 22nm Application-Specific Integrated Circuit (ASIC); blueprint for secure vector-FPU integration.
Gerlach [81]	Microarchitectural side-channel attacks.	New primitives: cache & time, flush & fault, cycle drift; exploits FP multi-cycle latency.	Demonstrated AES/RSA key recovery, KASLR break; highlights FPU/vector pipelines as attack surface.
SpecLFB [22]	Defense against speculative cache side channels.	Security check in line-fill buffer; ROB unsafe mask; delays MUSLs only.	Defends Spectre v1/v2/v4/v5; ~0.6% hardware overhead, ~2–3% runtime overhead; shows FP/vector speculation still a leakage source.

These studies show that FPUs in RISC-V systems serve a dual role. They are strong accelerators for secure cryptography when constant-time execution is enforced. Yet they also pose a risk for timing and speculative attacks if they are not protected. This duality highlights that future designs for FPU and vector units must balance performance with strong execution models to minimize their security risk.

4. Discussion and Insights

In fields like AI/ML, IoT, communications, graphics, and security, RISC-V FPUs provide a good example of how application characteristics affect architectural choices. AI accelerators tend to provide very high-throughput and

mixed precision support, which helps with inference using FP16, bfloat16, and FP8. Meanwhile, IoT is mainly focused on ultra-low-power and very small hardware, which is often achieved through the use of resource sharing. Communications require FPUs that are programmable with very low latency and can support complex baseband and MU-MIMO processing.

FPUs also resemble GPUs because of a focus on graphics workloads and include special instructions for rendering and image processing. In the context of security, FPUs render two contradictory functions. They can complete cryptographic operations very quickly using constant-time vector crypto units, as was illustrated in Marian. On the other hand, they can introduce side-channel attacks through variable latency and speculative execution, as was illustrated by Gerlach et al. and SpecLFB. These research works show that FPUs are becoming more domain-specific to a greater degree, where the quantities of performance, energy, and scaling allow a specific cross-domain trust in the RISC-V flexible ecosystem. Table 8 gives an overview of the major RISC-V FPU architectures and highlights the analyzed trade-offs and intended application domains.

Simulation and Evaluation Methodologies for RISC-V FPUs

Because performance and energy metrics are based on assessment environments, modern techniques for evaluating RISC-V floating-point computability need a simulation methodology. The most common methods for designing RISC-V floating point units currently include RTL, cycle-accurate architecture simulators, FPGA synthesis, ASIC synthesis, and various techniques within the firm and high-level analytical models. RTL and cycle-accurate simulators provide timing and dependency information, but require a lot of simulation time. Resource and clock frequency constraints considered real systems, but cannot be implemented on an ASIC. FPGA systems offer realistic hardware validating and allow estimation of workloads with real time consumption. High-level analytical models provide a faster design-space exploration, but may not consider the effects at the microarchitectural level.

For ease of use, FPnew and COPIFT are designed with flexible and variable lengths of floating point formats, where the design includes power-gating and dynamic scaling of bounds, and may introduce time-related dependencies that are difficult to capture using the design models. Better benchmarking of these designs requires simulation, operational load and power modeling with FPGA or ASIC physical representation, and cycle-accurate modeling.

Table 8: Comparative analysis of Major RISC-V FPU Architecture

FPU	Integration	Precision	Key features	Tech node/ Evaluation Platform	Trade-off	Target Domain
FPnew (2020)	Tightly coupled	Mixed (FP8-FP64)	Modular, transprecision, SIMD support.	22 nm, FPGA & ASIC.	Energy-efficient, flexible, limited vector scalability.	Embedded and HPC.
FAUST (2023)		FP16-FP64	Pipelined lanes, RVV-compliant vector FPU.	22 nm ASIC.	High throughput, large area & power, high.	HPC/vector DSP.
COPIFT (2025)		FP16-FP64	Cooperative scheduling, dual-issue integer and FP.	RTL Simulation.	Energy efficient, scheduling complexity.	AI/ML kernels.
Spatz (2023)	Moderately coupled	FP16-FP64	Snitch-based cluster coprocessor.	ASIC prototypes.	Balance between performance and modularity.	Vector compute.
Tiny FPU (2021)	Loosely coupled	FP32/FP64	Iterative datapath, aggressive reuse.	FPGA& synthesis.	Very small area with high latency.	IoT, MCUs.
FPUx (2024)		FP16- FP64	Hybrid pipeline, easy light interface.	22 nm ASIC.	Energy efficient, not for HPC.	IoT and Edge devices.
Quadrilatero (2025)		FP16 – FP32 with MatMul	Programmable systolic MatMul FPU.	22 nm ASIC prototype.	High throughput for graphics .	Graphics and edge AI.

Marian (2024)	Vector lane	Multi-format and crypto operations	Zvk crypto extensions, constant-time execution.	22 nm ASIC.	Security hardened with added logic area around 6.4%.	Cryptography and security.
Manticore (2021)	Hybrid	FP32/FP64	4096-core chiplet fabric, SSR, and FREP buffers.	22 nm, ASIC prototype.	Extreme scalability with interconnect and verification overhead.	Massive parallel HPC.

Table 9: Comparison of Optimization Techniques for RISC-V FPUs

Optimization Technique	Methods used	Advantages	Goal	Limitations	Reference
Deep pipelining	Adding pipeline stages for FP operations like add/mul/div/sqrt.	Very low latency, high frequency.	Maximize throughput.	Higher area and power, increased complexity	[19] [27]
Adaptive precision	Use reduced or variable precision depending on workload.	Lower energy and area with sufficient accuracy for ML.	Energy and area efficiency.	Not IEEE-754 compliant, accuracy loss.	[82] [66] [83] [70]
Clock gating	Disable unused datapath sections dynamically.	Significant energy savings.	Reduce energy.	Adds control overhead, may increase latency.	[84] [20]
Resource sharing	Share multipliers/dividers between FP and integer units.	Small silicon footprint, lower cost.	Area reduction.	Higher operation latency, stalls.	[85] [55] [86] [59]
Constant-time execution	Force all FP/crypto ops to use uniform latency.	Prevents timing side-channel leakage.	Security.	Higher average latency, area overhead.	[62] [87]
Vectorization	Multi-lane FPUs with wide vector registers.	Massive parallel FP performance.	High throughput for AI/HPC.	Large area/power, complex verification.	[88] [89] [90]
Power/clock gating of lanes	Dynamically disable unused lanes.	Scales the power to the workload.	Energy reduction in vector FPUs.	Requires fine-grained control logic.	[91]
Software fallback	Emulate unsupported FP ops in software.	Reduces hardware complexity.	Area reduction.	Very high latency, unsuitable for HPC.	[92]

Challenges In Design Trade-Offs

The design of FPUs for RISC-V systems comes with several challenges that need careful consideration based on the application domain. Table 9 summarizes the major optimization techniques explored for RISC-V FPUs, highlighting their methods, advantages, limitations, and application contexts. Deep pipelining improves throughput, whereas adaptive precision and clock gating target energy efficiency, either by dynamically reducing the precision of computations or by disabling unused datapath sections. The reduction in area is based on a reuse of functional units like multipliers between integer and floating-point operations. Constant-time execution ensures consistent latency to reduce timing side-channel leakage. This comes at the cost of increased latency and extra hardware. Vectorization efficiently divides floating-point processing across wide vector lanes for high-performance workloads, offering massive speedups in AI and HPC. This comes at the cost of significant silicon area and complex verification. further energy reduction can be accomplished through clock or power-gating each lane, dynamically scaling active lanes to the workload. Lastly, in the lightweight embedded cores, software fallback replaces hardware FP with software emulation, which reduces area but imposes very high latency. Altogether, these approaches show the varied design

options of RISC-V FPUs, where decisions depend on trade-offs between performance, area, energy, accuracy, and security.

Open Research Challenges in RISC-V FPUs Design

Within the framework of RISC-V floating-point unit (FPU) design, several open-ended research challenges at varying levels of abstraction, including benchmarking, toolchains, architectural considerations, and verification, remain unsolved. The main challenges associated with design and deployment of RISC-V FPUs are listed in Table 10. These include a lack of benchmarking, toolchain, and verification support, trade-offs in energy, and design safety. One significant challenge is the lack of developed standardized benchmarks and compiler backends for the new floating-point formats, including the recently developed bfloat16 and FP8 formats. The impact of this challenge is felt in the design of RISC-V FPUs for use in AI/ML, HPC, IoT, and cryptography.Verification is also a significant challenge, due to the IEEE-754 corner cases and architectural complexities of multi-lane mixed-precision FPUs, which expose new difficulties for verification suites. Energy and performance trade-offs are still not resolved. Deeply pipelined FPUs provide high throughput but use considerable area and power. In contrast, lightweight FPUs save energy but can increase latency.

Table 10: Open Research Challenges in RISC-V FPU Design

Areas	**Challenges**	**Open problem**	**Future Directions**
Standardization and benchmarking (2 – 3 years)	FPUs are evaluated using heterogeneous benchmarks, tools, and methodologies. Makes it difficult to compare results across works.	RISC-V ecosystem lacks a standardized benchmarking suite for FPUs that spans AI/ML, HPC, DSP, and cryptography.	Developing domain-aware, open-source benchmark suites.
Tool chain and compiler support (2 – 3 years)	Formats like posit, Bfloat16, FP8, and BFP are experimentally supported in hardware, but compilers, libraries, and debuggers rarely provide full integration.	Lack of mature compiler backends (LLVM/GCC) limits usability and slows real-world deployment.	Building compiler-aware transprecision frameworks and extending the RISC-V GNU toolchain to natively support new FP formats will accelerate adoption.
Verification and compliance (2 – 3 years)	IEEE-754 corner cases (NaNs, denormals, rounding modes) are hard to implement and verify, especially in reduced-area IoT FPUs.	Existing compliance suits are insufficient for emerging formats, which is a verification bottleneck	Creating formal verification frameworks and compliance suits for transprecision FPUs will improve trust and reliability.
Energy vs. performance trade-offs (3 – 5 years)	Deeply pipelined HPC FPUs deliver high throughput but at a large energy cost, while iterative FPUs for IoT save energy but compromise latency.	There is no unified design methodology to systematically balance area, energy, and accuracy across application domains.	Adaptive FPUs with runtime precision scaling and workload-aware gating could dynamically tune themselves for different workloads.
Security and side-channel resistance (3 – 5 years)	FPUs and vector pipelines can leak sensitive data through timing, cache, or speculative execution channels.	Marian addresses this partially by using time constant FP pipelines. AI/ML accelerators are vulnerable.	Developing security-hardened FPUs with fixed-latency execution. Register file partitioning and speculative-load defenses will be critical for secure AI and HPC.
Chiplet and heterogeneous integration. (5 – 10 years)	Interconnect latency, communication overhead, cache coherence complexity, bandwidth limitations, synchronization, and verification become major obstacles in distributed architectures.	The integration of heterogeneous FPUs within chiplet architectures can lead to increased power consumption while presenting challenges related to synchronization, data movement, and system-level verification. Additionally,	Research on NoC-aware FPU partitioning, scalable chiplet interconnect protocols, communication-efficient architectures, adaptive cache coherence mechanisms, and intelligent data-placement techniques can improve data

		loosely coupled or clustered FPU configurations may encounter communication bottlenecks and coherence traffic, which can hinder scalability and diminish overall performance efficiency.	delivery efficiency and enable practical deployment of heterogeneous FPU tiles.
Domain-specific Acceleration (5 – 10 years)	Most FPUs are either general-purpose or AI-centric, but domains like graphics, neuromorphic, and post-quantum cryptography need specialized support.	Domain-optimized FPU extensions limit efficiency for workloads like rendering, softmax, or lattice-based cryptography.	Extending RISC-V with application-driven FPU ISAs like graphics shaders, crypto pipelines, and transformer acceleration will broaden its industrial adoption. Support neuromorphic computing and SNN acceleration through co-design of RISC-V FPUs with event-driven processing architectures and scalable multi-FPGA deployment platforms.
Scalable verification for vector and tensor FPUs (5 – 10 years)	Vector-lane and tensor FPUs scale to wide datapaths, but verification and exception handling are extremely complex.	Ensuring correctness across multi-lane execution, exception propagation, and mixed-precision interactions is still a bottleneck.	Developing scalable verification methodologies, like AI-assisted, will be essential for future wide-vector and tensor FPUS.

Architectural Analysis Survey Report: Critical Bottlenecks in Next-Generation AI Hardware

As deep learning models expand from specialized convolutional networks to trillion-parameter autoregressive Large Language Models (LLMs), the primary constraint on AI system throughput has fundamentally shifted from compute saturation to data movement. While processing elements have achieved massive advancements in raw peak FLOPS, real-world hardware efficiency remains severely throttled by interconnected system bottlenecks.

This report provides an architectural analysis addressing the four critical pillars currently missing from basic literature surveys:

1.Floating-point data storage optimization within the memory hierarchy

2.Network-on-Chip (NoC) routing and bandwidth boundaries

3. Heterogeneous chiplet integration complexities

4. Multi-GPU scale-out communication bottlenecks

By leveraging and extending foundational frameworks established in recent literature [94], [95], [96], [97], this analysis maps out the exact hardware mechanics, architectural friction points, and industry mitigation pathways for each domain.

1.Floating-point data storage optimization within the memory hierarchy

The classical von Neumann architecture is based on the separation of the processor and memory, which creates an extreme operating danger: the "Memory Wall. The biggest challenge for AI accelerators is the high energy and latency (EL) consumption associated with moving a floating-point representation from one level of the storage hierarchy to another [94].

Architectural Mechanics & Friction Points

In deep learning workloads, transferring a data byte from off-chip memory (such as LPDDR or high capacity Flash memory) consumes many times more energy than performing a local math operation. The IEEE 754 standard for single precision floating point (FP32) formats offer wide dynamic range but have impractically large memories.

Table 11:AI Hardware Memory Hierarchy & Energy Cost Analysis

Memory Tier Location	Physical Implementation	Relative Performance Cost	Format Optimization & Traffic Characteristics
Registers /Local SRAM	Integrated directly onto the compute die (L1/L2 caches, register files).	**Lowest Cost** (Minimum energy consumption & sub-nanosecond latency)	Ideal for tight, high-frequency execution loops. Works best when utilizing **Narrow Precision Formats (FP8, FP4)** to maximize on-chip data density and minimize switching power.
Stacked High-Bandwidth Memory (HBM)	Stitched alongside the logic die using 2.5D/3D silicon interposers (e.g., HBM3e).	**Moderate Cost** (High bandwidth, but introduces packaging transport overhead)	Serves as the primary storage repository for active layer weights and activations during large-scale matrix operations.
Off-Package DRAM / Flash	Standard system memory modules (DIMMs) or NVMe storage drives separated by physical PCBs.	**Highest Cost** (Severe data-movement penalty; up to \$100\times\$ energy drop vs. local)	Subject to severe **Data Transport Energy Penalties**. Massively throttles the compute pipeline if model weights must be continuously swapped in from off-chip storage.

During training, gradients, weights and activations are constantly being shuffled in and out of register files, SRAM caches and off-package memory, leading to instant bandwidth saturation when using FP32. Fixed lower precision formats like FP16 or BF16 will increase storage density by 2x, but may lead to underflow / overflow during weight updates and cause gradient descent optimization loops to become unstable.

Mitigation Frameworks & Mathematical Optimizations

To avoid the memory space and power consumption overhead of using wide-precision floating-point execution, modern architectures use Hybrid Precision Floating-Point (HPFP) schemes and narrow-precision execution [95, 97]. The next generation of accelerators employ automatic precision selection algorithms rather than being tied to a fixed number of numbers. Table 11 Shows the analysis of AI Hardware Memory Hierarchy & Energy Cost

By profiling neural layers dynamically, these frameworks modulate the exponent and mantissa configurations layer-by-layer:

•Layers displaying highly volatile gradient shifts are allocated wider bit-allocations.

•Less sensitive layers are quantized down to narrow-precision footprints (FP8, FP6, or even FP4) [95].

As analyzed by Junaid et al. [95], adopting an algorithmically driven hybrid precision configuration can decrease memory access cycles by up to 37.5% and lower execution energy consumption by 49.4% while preserving model top-1 accuracy (under <0.8% variance).

2. Network-on-Chip (NoC) Bandwidth Limits

To overcome the physical limitation of the reticle size, the current AI accelerators have thousands of homogeneous matrix-multiplication processing elements (including tensor processing cores or systolic arrays) integrated on a single silicon substrate [94]. These dense execution units need to be connected using a Network-on-Chip (NoC), and it quickly becomes the dominant player of on-die performance.

Architectural Mechanics & Friction Points

Deterministic unicast packet routing protocols for point-to-point CPU transfers are used in standard computer architectural NoCs. But in neural processing, collective data operations, such as broadcasting weight tensors to an array of independent vector processing units (VPUs) or aggregation of activation maps across adjacent layers [97] plays a crucial role.

A typical NoC packet-switched network suffers from severe structural restrictions under GEMM operation:

- **Routing Congestion:** Unicast architectures push redundant data flows over common routing nodes, which leads to packet collisions and execution stalls.

- **Timing Closure and Wire Delays:** At advanced sub micron fabrication geometries (e.g., 3nm and below), global physical wire delays exceed logic gate delays, and wide and flat routing topologies will no longer be able to support high clock frequencies with significant latency penalties.

Mitigation Frameworks

Modern AI architectures embed hardware-level multicast protocols and in-network reduction in the hardware [94, 97] so as to avoid the NoC routing overheads. Instead of the NoC working as a passively transmitting communication pipe, the intelligent routing nodes are placed in the middle of the data path to intercept the data packets in flight.

During synchronization routines, the NoC routers do the arithmetic processing (e.g., partial summation reduction), so that the final consolidated output data only uses bandwidth on the global interconnect grid. This arrangement is an appreciable reduction in the total number of wires used and eliminates the routing delay on the critical execution path.

3. Heterogeneous Chiplet Architectures

The size limitations of manufacturing a single, large AI processing chip has resulted in the industry transitioning to modular and disaggregated solutions: Heterogeneous Chiplets connected with cutting-edge packaging technology [94, 96].

Architectural Mechanics & Friction Points

Instead of creating an all-in-one chip, it breaks the components into different functional units, known as "chiplets. The cutting-edge nodes with high cost (e.g., 3nm) are used to make compute-intensive parts (e.g., tensor calculation engines), and mature and high-yield nodes (e.g., 5nm or 7nm) are used to make communication-centric parts (e.g., I/O drivers, analog blocks, and PCIe controllers). These chiplets are stitched back together on a silicon interposer or organic substrate [94].

This approach poses distinct architectural challenges:

- **Non Uniform memory Access(NUMA):** A Die-to-Die (D2D) interface, such as UCIe or ultra-short-reach links, has a higher latency and lower energy efficiency than moving data across a local monolithic cache [96].
- **Workload Partitioning:** The mapping software for workload partitioning can have a significant impact on system throughput, if this software is not smart enough to preserve a model's layer hierarchy on the different physical chiplets, then cross-chiplet interface serialization overheads may kill system throughput by over one third, as compute engines wait for cross-die data transfers.

Mitigation Frameworks

The solution of chiplet bottlenecks rests in the hands of a deep hardware-software co-design [94, 96]. Advanced layout mapping algorithms are used by designers to keep high frequency data-loops, such as those for recurrent attention token generation, on a single compute chiplet. At the same time, packaging tech is working on 3D stacking (Through-Silicon Vias or TSVs) that lets logic be stacked vertically right on top of memory, and replaces the horizontal, high-latency D2D route with a short vertical interconnect path.

4. Multi-GPU Communication & Scale out Bottlenecks

Scalability issues, such as the memory constraint, occur as soon as the big models of today are running on a single device when scaling deep-learning frameworks to accommodate them. Workloads need to be parallelized in a massive cluster across thousands of computing nodes using either tensor, pipeline or Mixture-of-Experts (MoE) parallelization approaches [96].

Architectural Mechanics & Friction Points

In a distributed cluster training, individual GPUs have to always update each other's' weight parameters. This synchronization makes mathematical communication collectives such as AllReduce, AlltoAll and ReduceScatter repetitive. These sync packets will determine the aggregate throughput of an AI data center.

Mitigation Frameworks

The industry tackles scale-out issues from two main industry structural directions:

Unified Rack-Scale Clusters: Clusters such as NVLink domains scale out with several dozen independent GPUs connected via a single copper backplane. This removes the barrier of individual server nodes and offers peer-to-peer memory semantic speeds directly across the rack footprint.

Data Processing Units (DPUs) & Asynchronous Overlapping: Network-adjacent processing cores enable clusters to offload network packet tracking to independent network adapters. This topology allows for processing engines to do calculations for later layers asynchronously while the networking infrastructure handles the calculation of the parameters in the background.

As multi-GPU workloads scale outwards, they hit a steep Interconnect Performance Cliff where bandwidth scales downwards across system layers.

The following Table 12 presents the major communication and memory bandwidth levels in modern AI accelerator systems:

Communication Layer	Technology	Typical Bandwidth
On-Package Memory Access	High-Bandwidth Memory (HBM3e)	**4.0 – 4.8 TB/s**
Intra-Node GPU-to-GPU Communication	NVLink / UALink	**1.0 – 1.8 TB/s**
System Bus Routing	PCIe Gen 6	**128 GB/s**
Inter-Node Cluster Networking	InfiniBand / RoCE	**50 – 100 GB/s**

5.Summary Architectural Synthesis

The next taxonomy describes how the major architectural constrictions relate to the overall system impacts and related frameworks for hardware/software mitigation: Table 13 Shows Architectural Solutions for AI Compute Bottlenecks

Table 13: Architectural Solutions for AI Compute Bottlenecks

Bottleneck Domain	Core System Consequence	Dominant Architectural Solution	Literature Reference Alignment
Floating-Point Storage	Memory capacity exhaustion; high data-shuffling energy costs.	**Hybrid Precision Selection (HPFP)**, dynamic numerical bit-width quantization (FP8/FP4).	*[95], [97]*
NoC Bandwidth	Internal routing gridlocks during collective tensor broadcasting.	**In-Network Reduction**, hardware multicast support, collective-capable routing grids.	*[94], [97]*
Heterogeneous Chiplets	Interface serialization latencies; NUMA memory access penalties.	**3D Packaging Integration**, communication-aware dataflow mapping.	*[94], [96]*
Multi-GPU Scale-Out	Compute stalling during cluster parameter synchronization routines.	**Rack-scale scale-up fabric domains**, asynchronous computation/communication overlapping via DPUs.	*[96]*

5. Conclusion

Floating-point units (FPUs) have been key to the success of RISC-V so much so that they have gone beyond being just optional accelerators and are now a must-have in the ecosystem. The survey presents the status of the RISC-V FPU research and development, including scalars, transprecision, vectors and specialized accelerators. The paper discusses the various innovations in this field, by exploring the new number formats, integration approaches, optimization strategies, and application dependent deployments.FPU design in RISC-V is no longer one-size-fits-all; it reflects the different needs of areas like AI/ML, HPC, IoT, cryptography, and graphics. Techniques such as deep pipelining, adaptive precision, constant-time execution, and resource sharing show how designers balance area, performance, energy, and security.

The future direction of the field is towards CPUs with heterogeneous, secure and energy efficient FPUs tightly coupled with vector and tensor pipelines and chiplet based architectures. The paper highlights that there are several open challenges including limited support for non-IEEE formats by compiler and toolchain, the need for standardized benchmarks, bottlenecks in verification, integration of chiplets in heterogeneous chiplet fabrics, and the side channel vulnerabilities. Solving these issues will be a critical step to make RISC-V FPUs mainstream.

In conclusion, RISC-V FPUs are no longer peripheral design elements but are becoming key drivers of performance, efficiency, and trust in modern computing systems. The survey offers a clear overview of the current state-of-the-art references and a roadmap for future research. It provides guidance for academics, practitioners, and industry stakeholders who are working to develop the next generation of floating-point architectures. In the meantime, a promising path toward extremely effective edge-neuromorphic systems is provided by the integration of RISC-V FPUs with event-driven spiking neural network designs on multi-FPGA platforms.

Declarations

Funding: This research received no external funding.

AI Tools Declaration

The authors state that they used AI-based writing assistance tools (such as ChatGPT/OpenAI) to assist with language editing and proofreading to improve clarity and readability. However, the authors did not use these tools for any other portion of the technical content (e.g., developing data analysis or experimental designs).

References

1. IEEE Standard for Floating-Point Arithmetic, in IEEE Std 754-2019 (Revision of IEEE 754-2008) , vol., no., pp.1-84, 22 July **2019**, doi: 10.1109/IEEESTD.2019.8766229.
2. M. Flynn and S. Oberman, Advanced Computer Arithmetic Design, *Wiley-IEEE Press*, **2001.**
3. Sabbagh Molahosseini, Amir, et al. Low-precision floating-point formats: From general-purpose to application-specific. *Approximate Computing* **2012:** 77-98.
4. Waterman, Andrew, and Krste Asanovic. The risc-v instruction set manual volume i: Unprivileged isa. *Document Version 20191213* **2019:** 1-4.
5. Asanović, Krste, and David A. Patterson. Instruction sets should be free: The case for RISC-V. *EECS Department, University of California, Berkeley*, Tech. Rep. UCB/EECS-2014-146 **2014.**
6. Asanovic, Krste, et al. The rocket chip generator. *EECS Department, University of California, Berkeley*, Tech. Rep. UCB/EECS-2016-17 4 **2016:** 6-2.
7. Asanovic, Krste, David A. Patterson, and Christopher Celio. The Berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized RISC-V processor. No. *UCBEECS2015167*. **2015**.
8. Rossi, Davide, et al. PULP: A parallel ultra low power platform for next generation IoT applications. *2015 IEEE Hot Chips 27 Symposium (HCS). IEEE Computer Society*, **2015.**
9. Liu, Louis. RISC-V customized instructions for AI-related operations. *Diss. ResearchSpace@ Auckland*, **2024**.
10. El Zarif, N., Hemmat, M. A., Dupuis, T., David, J. P., & Savaria, Y. Polara-Keras2c: Supporting Vectorized AI Models on RISC-V Edge Devices. *IEEE Access.* **2024.**
11. Chelliah, Amir Masoud Rahmani, et al. 3.2 Edge AI. *Model Optimization Methods for Efficient and Edge AI* **2025**.
12. A. Waterman and K. Asanović, The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.12, RISC-V International, May 2021. [Online]. Available: https://riscv.org/specifications/privileged
13. https://five-embeddev.com/riscv-priv-isa-manual/Priv-v1.12/hypervisor.html
14. Sevaldson, Ruben. Memory protection for embedded RISC-V systems. *MS thesis. NTNU*, **2022.**
15. AX45MP Multicore 64-bit RISC-V Superscalar Application Processor, *Andes Technology Corporation*,**2024**
16. SiFive U74 Core Complex Manual 21G1.01.00. https://starfivetech.com/uploads/u74_core_complex_manual_21G1.pdf
17. Microchip, PolarFire® SoC FPGA Architecture, *Tech. Brief*, **2022.**

18. J. Hauser, HardFloat: IEEE-754 Compliant FPU Generator, *UC Berkeley*, **2019.**
19. Kovač, Mate, et al. FAUST: Design and implementation of a pipelined RISC-V vector floating-point unit.*Microprocessors and microsystems* **2023** *97* 104762.
20. S. Mach, F. Schuiki, F. Zaruba and L. Benini, FPnew: An Open-Source Multiformat Floating-Point Unit Architecture for Energy-Proportional Transprecision Computing, in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **April 2021** vol. 29, no. 4, pp. 774-787, doi: 10.1109/TVLSI.2020.3044752.
21. Bertaccini, Luca, et al. MiniFloats on RISC-V cores: ISA extensions with mixed-precision short dot products.*IEEE Transactions on Emerging Topics in Computin*g **2024** 12.4 1040-1055.
22. Cheng, Xiaoyu, et al. Speclfb: Eliminating cache side channels in speculative executions.*Presented at USENIX Security* **2024:** 1.
23. Lee, J., et al. Faster inference of LLMs using FP8 on the Intel Gaudi. *arXiv preprint arXiv* ***2025*:**2503.09975.
24. Chen, Ruiqi, et al. FPGA-based Approximate Multiplier for FP8. *2025 IEEE 33rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE,* **2025**.
25. Hunhold, L., & Gustafson, J. Spectral methods via FFTs in emerging machine number formats: OFP8, Bfloat16, Posit, and Takum arithmetics. *arXiv preprint arXiv*:**2025** 2504.21197.
26. Malathi, D., et al. Design of Risc-V Processing Unit Using Posit Number System. *2023 4th International Conference on Signal Processing and Communication (ICSPC). IEEE,* **2023.**
27. Rossi, F., et al. PPU: Design and implementation of a pipelined full posit processing unit. *arXiv preprint arXiv* **2023**:2308.03425
28. Lu, Jinming, et al. Evaluations on deep neural networks training using posit number system. *IEEE Transactions on Computers 70.2* **2020**: 174-187.
29. Zhang, Sai Qian, Bradley McDanel, and H. T. Kung. Fast: Dnn training under variable precision block floating-point with stochastic rounding. *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE,* **2022.**
30. Köster, Urs, et al. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. *Advances in neural information processing systems 30* **2017.**
31. Alsuhli, Ghada, et al. A Survey and Comparative Analysis of Number Systems for Deep Neural Networks. *Proceedings of the IEEE* **2025.**
32. G. Alsuhli, V. Sakellariou, H. Saleh, M. Al-Qutayri, B. Mohammad and T. Stouraitis, A Survey and Comparative Analysis of Number Systems for Deep Neural Networks, *in Proceedings of the IEEE* **Feb 2025**, vol. 113, no. 2, pp. 172-207, doi: 10.1109/JPROC.2025.3578756
33. Sakai, Yasufumi, and Yutaka Tamiya. S-DFP: shifted dynamic fixed point for quantized deep neural network training. *Neural Computing and Applications 37.2* **2025:** 535-542.
34. Zaruba, Florian, Fabian Schuiki, and Luca Benini. Manticore: A 4096-core RISC-V chiplet architecture for ultraefficient floating-point computing. *IEEE Micro 41.2* **2020***:* 36-42.
35. Rizwan, Ruhma, et al. Accelerating AI on RISC-V Optimizing BFloat16 for Improved Efficiency. **2024.**
36. Dequino, Alberto, et al. Optimizing BFloat16 Deployment of Tiny Transformers on Ultra-Low Power Extreme Edge SoCs. *Journal of Low Power Electronics and Applications 15.1* **2025**: 8.
37. Gong, Jing. Optimized Floating-Point Arithmetic for Efficient Operations in DNN Accelerators. *Diss. UNSW Sydney*, **2025**.
38. Valero-Lara, Pedro, et al. Mixed-precision S/DGEMM using the TF32 and TF64 frameworks on low-precision AI tensor cores. *Proceedings of the SC'23 Workshops of the International Conference on High-performance Computing, Network, Storage, and Analysis.* **2023.**
39. Hanindhito, Bagus, and Lizy K. John. Accelerating ml workloads using gpu tensor cores: The good, the bad, and the ugly. *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering*. **2024.**
40. Arslan, V., et al. Accelerating 3D Stencil Computations with Tensor Cores and TF32. *Empowering the Energy Shift: The Role of HPC in Sustainable Innovation. Vol. 2025. No. 1. European Association of Geoscientists & Engineers*, **2025.**
41. Donsez, Didier. Frugal AI for IoT and New Space.*JRAF 2024 Journées de Recherche en Apprentissage Frugal*. **2024**.
42. Chen, Cheng. Low-bitwidth Floating-Point Quantization for Diffusion Models. MS thesis. University of Toronto (Canada), **2024**
43. Fujii, K., Nakamura, T., & Yokota, R.. Balancing speed and stability: The trade-offs of FP8 vs. BF16 training in LLMs. *arXiv preprint arXiv* **2024** :2411.08719
44. Mallasén, D., et al. Increasing the energy-efficiency of wearables using low-precision Posit arithmetic with PHEE. *arXiv preprint arXiv* **2025**:2501.18253.
45. Murillo Montero, Raúl. Posit arithmetic and its applications. **2025.**
46. Khan, Daud, Latif Jan, and Mohammad Haseeb Zafar. Optimized FFT Designs for High-Performance LTE and 5G Networks.*Arabian Journal for Science and Engineering* **2025:** 1-20.
47. Tarkka, Antti. Architectural Exploration of Block Floating-Point Support for Matrix-Vector Processor. **2025.**
48. Xiong, Botao, et al. LUT-DSP usage trade-off for re-configurable convolution acceleration core based on small logarithmic floating-point epresentation. *International Journal of Circuit Theory and Applications 52*.4 **2024**: 1864-1871.

49. Xiong, Botao, et al. Design of Low-Cost and High-Accurate 8-bit Logarithmic Floating-Point Arithmetic Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **2025**.
50. Ellaithy, Dina Mohamed. Approximate Square and Square Root Calculation without Multiplication or Division for DSP Applications. *Journal of Advanced Research in Applied Sciences and Engineering Technology 48.2* **2024**: 121-135.
51. Osaki, Atsuya, et al. Dynamic Fixed-point Values in eBPF: a Case for Fully In-kernel Anomaly Detection.*Proceedings of the Asian Internet Engineering* Conference 2024. **2024**.
52. Fasolino, Andrea, et al. Dynamically adaptive accumulator for in-sensor and hardware accelerators. *2024 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE*, **2024**.
53. Raut, Gopal, et al. A simd dynamic fixed point processing engine for dnn accelerators. *2024 25th International Symposium on Quality Electronic Design (ISQED). IEEE*, **2024**.
54. L. Colagrande and L. Benini, Dual-Issue Execution of Mixed Integer and Floating-Point Workloads on Energy-Efficient In-Order RISC-V Cores,*2025 62nd ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA*, **2025**, pp. 1-7, doi: 10.1109/DAC63849.2025.11132520
55. Perotti, Matteo, et al. Spatz: Clustering compact RISC-V-based vector units to maximize computing efficiency. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2025.**
56. Perotti, Matteo, et al. Spatzformer: An Efficient Reconfigurable Dual-Core RISC-V V Cluster for Mixed Scalar-Vector Workloads. *2024 IEEE 35th International Conference on Application-specific Systems, Architectures and Processors (ASAP). IEEE*, **2024.**
57. Fritzmann, Tim, Georg Sigl, and Johanna Sepúlveda. RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography.*IACR Transactions on Cryptographic Hardware and Embedded Systems* **2020**: 239-280.
58. Cammarata, Danilo, et al. Quadrilatero: A RISC-V programmable matrix coprocessor for low-power edge applications.*Proceedings of the 22nd ACM International Conference on Computing Frontiers: Workshops and Special Sessions*. **2025**.
59. Lin, Xian, et al. FPUx: High-Performance Floating-Point Support for Cost-Constrained RISC-V Cores. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 32.10* **2024**: 1945-1949.
60. Bertaccini, Luca, et al. "=Tiny-FPU: Low-cost floating-point support for small RISC-V MCU cores. *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, **2021.**
61. Zaruba, Florian, et al. Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads. *IEEE Transactions on Computers* 70.11 **2020:** 1845-1860.
62. Szymkowiak, Thomas, Endrit Isufi, and Markku-Juhani Saarinen. Marian: An Open Source RISC-V Processor with Zvk Vector Cryptography Extensions. *Cryptology ePrint Archive* **2024**.
63. Perotti, Matteo, et al. Ara2: Exploring single-and multi-core vector processing with an efficient RVV 1.0 compliant open-source processor.*IEEE Transactions on Computers* 73.7 **2024**: 1822-1836.
64. Atef, Ahmed Kamaleldin. A Modular Platform for Adaptive Heterogeneous Many-Core Architectures. **2023.**
65. Tsai, Tsung-Han, and Ding-Bang Lin. An on-chip fully connected neural network training hardware accelerator based on brain float point and sparsity awareness. *IEEE Open Journal of Circuits and Systems* 4 **2023:** 85-98.
66. Wang, Chuanning, et al. A scalable risc-v vector processor enabling efficient multi-precision dnn inference.*2024 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, **2024.**
67. R. Wang et al., VEXP: A Low-Cost RISC-V ISA Extension for Accelerated Softmax Computation in Transformers,*2025 IEEE 32nd Symposium on Computer Arithmetic (ARITH), El Paso, TX, USA*, **2025**, pp. 37-44, doi: 10.1109/ARITH64983.2025.00016.
68. G. Islamoglu et al., MXDOTP: A RISC-V ISA Extension for Enabling Microscaling (MX) Floating-Point Dot Products, in *2025 IEEE 36th International Conference on Application-specific Systems, Architectures and Processors (ASAP), Vancouver, BC, Canada*, **2025**, pp. 81-84, doi: 10.1109/ASAP65064.2025.00021
69. Park, Jina, et al. Designing low-power RISC-V multicore processors with a shared lightweight floating-point unit for IoT endnodes. *IEEE Transactions on Circuits and Systems I: Regular Papers* **2024**.
70. Conti, Francesco, et al. 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* 59.1 **2023**: 128-142.
71. Amor, Hela Belhadj, Carolynn Bernier, and Zdeněk Přikryl. A RISC-V ISA extension for ultra-low power IoT wireless signal processing. *IEEE Transactions on Computers* 71.4 **2021:** 766-778.
72. Tourres, Mael, et al. Extended RISC-V hardware architecture for future digital communication systems.*2021 IEEE 4th 5G World Forum (5GWF)*. IEEE, **2021**.
73. Andri, Renzo, Tomas Henriksson, and Luca Benini. Extending the RISC-V ISA for efficient RNN-based 5G radio resource management. *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, **2020**.
74. Castañeda, Oscar, Luca Benini, and Christoph Studer. A 283 pj/b 240 mb/s floating-point baseband accelerator for massive mu-mimo in 22fdx. *ESSCIRC 2022-IEEE 48th European Solid State Circuits Conference (ESSCIRC)*. IEEE, **2022**.
75. Attari, M., Edfors, O., & Liu, L. Accelerator-assisted floating-point ASIP for communication and positioning in massive MIMO systems. *arXiv preprint arXiv* **2025.:**2502.09785.
76. Zhou, Yuzhi, Xi Jin, and Tian Xiang. RISC-V graphics rendering instruction set extensions for embedded AI chips implementation.*Proceedings of the 2020 2nd International Conference on Big Data Engineering and Technology*. **2020.**

77. Tine, Blaise, et al. Vortex: Extending the risc-v isa for gpgpu and 3d-graphics. *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. **2021**.
78. Li, Jingzhou, et al. RISC-V-Based GPGPU With Vector Capabilities for High-Performance Computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **2025.**
79. Liu, Juan, et al. Custom Instruction Design for Image Preprocessing in RISC-V. *Proceedings of the 2024 7th International Conference on Artificial Intelligence and Pattern Recognition*. **2024**.
80. Lu, T. . A survey on RISC-V security: Hardware and architecture. *arXiv preprint arXiv* **2021**:2107.04175.
81. Gerlach, Lukas, et al. A security risc: microarchitectural attacks on hardware risc-v cpus. *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, **2023**.
82. Li, Q., et al. Enable lightweight and precision-scalable Posit/IEEE-754 arithmetic in RISC-V cores for transprecision computing. *arXiv preprint arXiv* **2025.:**2505.19096.
83. Guthmuller, Eric, et al. Xvpfloat: RISC-V ISA extension for variable extended precision floating-point computation.*IEEE Transactions on Computers* 73.7 **2024**: 1683-1697.
84. Vo, Minh Huan.Hybrid Data Driven Clock Gating and Data Gating Technique for Better Saving Power in ALU RISC-V. *International Journal of Electrical and Electronics Research* 12.1 **2024**: 238-246
85. Islam, Md Ashraful, and Kenji Kise. An Efficient Resource Shared RISC-V Multicore Architecture. *IEICE TRANSACTIONS on Information and Systems* 105.9 **2022**: 1506-1515.
86. Shen, Diyou, et al. TCDM Burst Access: Breaking the Bandwidth Barrier in Shared-L1 RVV Clusters Beyond 1000 FPUs. *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, **2025**.
87. Saarinen, Markku-Juhani O. Brief Comments on Rijndael-256 and the Standard RISC-V Cryptography Extensions.*Cryptology ePrint Archive* **2025**.
88. Purayil, Navaneeth Kunhi, et al. AraXL: A physically scalable, ultra-wide RISC-V vector processor design for fast and efficient computation on long vectors. *2025 Design, Automation & Test in Europe Conference (DATE). IEEE,* **2025.**
89. Xu, Siyi, et al. Zoozve: A Strip-Mining-Free RISC-V Vector Extension with Arbitrary Register Grouping Compilation Support (WIP). *Proceedings of the 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*. **2025**.
90. V. Titopoulos, G. Alexakis, C. Nicopoulos and G. Dimitrakopoulos, Efficient Implementation of RISC-V Vector Permutation Instructions, in *2025 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Kalamata, Greece*, **2025**, pp. 1-6, doi: 10.1109/ISVLSI65124.2025.11130346.
91. Minervini, Francesco, et al. Vitruvius+: An area-efficient RISC-V decoupled vector coprocessor for high-performance computing applications. ACM Transactions on Architecture and Code Optimization 20.2 **2023**: 1-25.
92. A. Marcelli, 8—bit Vector SoftFloat Extension for the RISC—V Spike Simulator. *Authorea Preprints* **2025.**
93. Liang Z, Zhang X, Vousden M, et al. Quantitative simulations of Spiking Neural Networks on an event-driven FPGA cluster[J]. Integration, **2025**: 102590.
94. Ahsan, S. M. M., Hoque, T., Hasan, M. S., Chowdhury, M., & Dhungel, A. (2025). Hardware Accelerators for Artificial Intelligence. AI-Enabled Electronic Circuit and System Design, 497–535. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-71436-8_14
95. Junaid, M., Aliev, H., Park, S., Kim, H., Yoo, H., & Sim, S. (2024). Hybrid Precision Floating-Point (HPFP) Selection to Optimize Hardware-Constrained Accelerator for CNN Training. Sensors, 24(7), 2145. MDPI AG. https://doi.org/10.3390/s24072145
96. Kachris, C. (2024). A Survey on Hardware Accelerators for Large Language Models. Appl. Sci., 15(2), 586. https://doi.org/10.3390/app15020586
97. Mohaidat, T., & Khalil, K. (2024). A Survey on Neural Network Hardware Accelerators. IEEE Transactions on Artificial Intelligence, 5(8), 3801–3822. Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/tai.2024.3377147.