Maestro: A 302 GFLOPS/W and 19.8GFLOPS RISC-V Vector-Tensor Architecture for Wearable Ultrasound Edge Computing

Maestro: A 302 GFLOPS/W and 19.8GFLOPS RISC-V Vector-Tensor Architecture for Wearable Ultrasound Edge Computing

Mattia Sinigaglia, Amirhossein Kiamarzi, Marco Bertuletti, *Student Member, IEEE*, Luigi Ghionda, Mattia Orlandi *Student Member, IEEE*, Riccardo Tedeschi, Aurora Di Giampietro, Yvan Tortorella, Luca Bertaccini, *Member, IEEE*, Simone Benatti *Member, IEEE*, Giuseppe Tagliavini, *Senior Member, IEEE*, Luca Benini, *Fellow, IEEE*, Francesco Conti, *Senior Member, IEEE*, and Davide Rossi, *Senior Member, IEEE*

***Abstract*—Most Wearable Ultrasound (WUS) devices lack the computational power to process signals at the edge, instead relying on remote offload, which introduces latency, high power consumption, and privacy concerns. We present Maestro, a RISC-V SoC with unified Vector-Tensor Unit (VTU) and memory-coupled Fast Fourier Transform (FFT) accelerators targeting edge processing for wearable ultrasound devices, fabricated using low-cost TSMC 65nm CMOS technology. The VTU achieves peak 302GFLOPS/W and 19.8GFLOPS at FP16, while the multi-precision 16/32-bit floating-point FFT accelerator delivers peak 60.6GFLOPS/W and 3.6GFLOPS at FP16. We evaluate Maestro on a US-based gesture recognition task, achieving 1.62GFLOPS in signal processing at 26.68GFLOPS/W, and 19.52GFLOPS in Convolutional Neural Network (CNN) workloads at 298.03GFLOPS/W. Compared to a state-of-the-art SoC with a similar mission profile, Maestro achieves a 5× speedup while consuming only 12mW, with an energy consumption of 2.5mJ in a wearable US channel preprocessing and ML-based postprocessing pipeline.**

***Index Terms*—Heterogeneous, RISC-V, Vector, Tensor, FFT, Low-Power, embedded, ultrasound SoC, WUS SoC, frequency-domain SoC**

I. Introduction

THE recent advancements in Ultrasound (US) [1] transceiver technology are enabling the development of Wearable Ultrasound (WUS) systems [2] for the next-generation Human-Machine Interfaces (HMIs), with applications such as movement tracking [3] and gesture recognition [2], [4], benefitting from an increase in the miniaturization and energy efficiency of US probes: correspondingly, a fundamental requirement for all wearable systems is that they must be unobtrusive, compact and cost effective. Next-generation WUS systems also need to reduce the physical footprint of batteries that support onboard processing and ensure low operating temperature without active cooling. This translates in very strict <50mW power constraints, which in turn require an extremely energy-efficient edge processing platform.

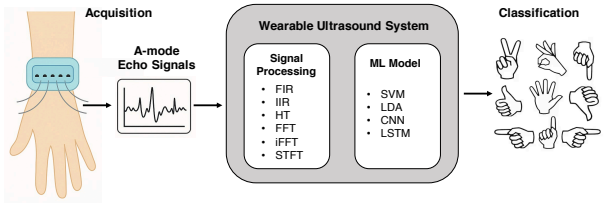


Fig. 1. Overview of a wearable gesture recognition system, highlighting the computational demands for ultrasound-based processing.

For wearable processing systems focusing on biopotential signals (e.g., Electroencephalography (EEG), Electromiography (EMG)), several energy-efficient SoCs have been proposed for [5]–[7]. For this class of signals, input bandwidth is in the low-KHz range, preprocessing and conditioning effort is relatively low.

On the other hand, US signals have significantly higher input bandwidth and require more elaborate signal conditioning to transform A-mode data—echoes reflecting muscle tissue dynamics—into features suitable for downstream tasks such as gesture classification [8]–[12]. This processing pipeline depicted in Fig.1 is partly handled in the analog front-end, which excites the transducers, captures the echoes, and amplifies the signal. The remaining steps take place in the digital domain, typically involving classical Digital Signal Processing (DSP) techniques such as Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filtering, Hilbert Transform (HT), and Fast Fourier Transform (FFT). The resulting data—often in the frequency domain—is then processed by a variety of Machine Learning (ML) models [10], ranging from traditional approaches [9] such as Support Vector Machines (SVM) and Linear Discriminant Analysis (LDA), to deep learning-based classifiers such as Convolutional Neural Network (CNN) or

Manuscript submitted March 6, 2025.

Mattia Sinigaglia, Amirhossein Kiamarzi, Luigi Ghionda, Mattia Orlandi, Riccardo Tedeschi, Aurora Di Giampietro, Yvan Tortorella, Giuseppe Tagliavini, Luca Benini, Francesco Conti, and Davide Rossi are with the Department of Electrical, Electronic and Information Engineering (DEI), University of Bologna, 40136 Bologna, Italy. Simone Benatti is with the Department of Science and Methods for Engineering (DISMI), University of Modena e Reggio Emilia, Italy. Marco Bertuletti, Luca Bertaccini and Luca Benini are with the Integrated Systems Laboratory (IIS), ETH Zürich, 8092 Zürich, Switzerland.

This work was supported in part through the ISOLDE (101112274) and TRISTAN (101095947) project that received funding from the HORIZON CHIPS-JU program and in part by the Spoke 1 on Future High-Performance-Computing (HPC) of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) that received funding from the Ministry of University and Research (MUR) for the Mission 4–Next Generation EU program.

Long Short-Term Memory (LSTM) [4]. This diverse pipeline calls for architectures that balance flexibility and efficiency, supporting both DSP and ML within wearable system constraints.

State-of-the-Art Ultra-Low Power (ULP) Microcontroller Units (MCUs) [5], [13], [14], commonly used in wearable scenarios, prioritize minimal power consumption with respect to performance and are unsuited to the complexity of WUS workloads. On the other hand, modern Artificial Intelligence (AI)-accelerated MCUs provide advanced AI capabilities but generally lack high-performance DSP features, particularly for frequency-domain processing [6], [15]–[17]. Whenever a given workload is not directly accelerated, these systems rely on software execution on simple, low-performance RISC processors, which severely hinders end-to-end performance in the case of WUS. Embedded vector processors [18] provide an attractive alternative to couple performance and efficiency on a wide variety of DSP tasks with the flexibility of a programmable architecture. However, for modern AI workloads, vector processors may need external tensor units to reach target performance, incurring notable area and power overhead.

In this work, we propose to go one step further in integration, with an open-source-ip-based WUS processing architecture that offers a unified vector/tensor unit to boost efficiency on AI and linear-algebra-centric workloads. By merging the tensor unit into the vector processor, we are able to share part of their microarchitectural resources, minimizing the overhead of the tensor unit while still retaining its performance.

Additionally, to enable fast and efficient frequency-domain processing in the WUS scenario, we couple our processor with a companion multi-precision floating-point FFT engine. As an embodiment of this architectural approach, we present *Maestro*, a unified Vector-Tensor WUS System-on-a-Chip (SoC) prototyped in 65nm CMOS technology. In detail, the key contributions of this work are the following:

- a multi-precision unified Vector Unit (VU), based on the Spatz architecture [19], supporting the RISC-V Vector (RVV) Instruction-Set Architecture (ISA) with FP16–FP64 data, and high-performance General Matrix-Matrix Multiplication (GEMM) operations in FP8–FP16 precision in a fully integrated Vector-Tensor Unit (VTU);
- a newly designed multi-precision FP16–FP32 radix-2 DIT FFT accelerator employing novel fused-arithmetic dual-output sum-of-dot-products modules for high-precision, low-energy operation;
- a heterogeneous compute cluster tightly integrating the unified vector-tensor unit and FFT engine, all connected through a shared memory subsystem with 128KiB of low-latency L1 memory;
- the first silicon prototype implementation of the Spatz-based vector unit, tightly integrated with a tensor unit and an FFT accelerator, realized in commercial 65nm CMOS technology as part of a complete and functional SoC;
- full end-to-end application mapping and evaluation on a real-world ultrasound-based gesture recognition workload, including comparative analysis against GAP9 [7], a commercial high-performance, energy-efficient SoC.

TABLE I
ARCHITECTURES FOR VECTOR, TENSOR, FFT ACCELERATION AND ULTRASOUND SYSTEMS

Architecture	Technology	Max Frequency	Vector Accelerator	Tensor Accelerator	FFT Accelerator	Power Envelope
Ara [20]	GF22 FDX	1 GHz	✓	-	-	763 mW
YUN [21]	TSMC 65	280 MHz	✓	-	-	330 mW
Speed [22]	TSMC 28	1.05 GHz	✓	✓	-	533 mW
Spatz [19]	GF 12LPP	1 GHz	✓	-	-	164 mW
Gemmini [23]	GF22 FDX	961 MHz	-	✓	-	-
Tsunami [24]	65nm CMOS	200 MHz	-	✓	-	419 mW
DARKSIDE [25]	TSMC 65	290 MHz	-	✓	-	213 mW
ECHOES [26]	TSMC 65	350 MHz	-	-	✓	134 mW
[27]	16nm FinFET	940 MHz	-	-	✓	22.6 mW
[28]	65nm CMOS	100 MHz	-	-	✓	59.11 mW
[29]	TSMC 28	270 MHz	-	-	✓	126 mW
STM32F7 [30] (product)	90nm CMOS	216 MHz	-	-	-	900 mW
TMS320F28379D [31] (product)	-	200 MHz	-	-	✓	576 mW
dsPIC33EP512MU814 [32] (product)	-	70 MHz	-	-	-	350 mW
GAP9 / VEGA [7]	GF22 FDX	370 MHz	-	✓	-	50 mW
This work	**TSMC 65**	**210 MHz**	✓	✓	✓	**212 mW**

Maestro delivers 1.62GFLOPS and 19.52GFLOPS in frequency domain and CNN tasks, respectively, outperforming GAP9 by 2.63× and 6.53×. It delivers its highest energy efficiency in CNN processing at 298GFLOPS/W, 2.19× higher than GAP9. Overall, our results show that Maestro's unified Vector/Tensor processor + FFT companion architecture is highly competitive with GAP9 on WUS applications, achieving 5× better end-to-end latency on our target WUS workload despite GAP9's much more aggressive technology node (22nm). Furthermore, when employed in a Maestro-based ultrasound system, which acquires and processes data at the edge, the entire system operates with an ultra-low power consumption of just 12mW, achieving an energy cost of only 2.5mJ and operating up to 94 hours.

The manuscript is organized as follows: Section II reviews related work on vector processors, AI and FFT accelerators, and WUS systems. Sections III–V describe the Maestro architecture and its accelerators. Section VI presents the silicon prototype and measurements, while Section VII evaluates an A-Mode WUS application. Section VIII provides a comparative analysis, and Section IX concludes with key findings and future research directions.

II. RELATED WORK

A. Vector Cores

Vector processing effectively combines programming flexibility with performance and energy efficiency (Table I), as a Single-Instruction-Multiple-Data (SIMD) instruction can process multiple data elements, reducing instruction fetch and dispatch overhead [33].

Both ARM and RISC-V instruction sets support advanced vector processing capabilities with dedicated extensions. ARM introduced the Scalable Vector Extension (SVE) [34] and its successor, SVE2, while the RISC-V community developed the RISC-V Vector (RVV) ISA [35], known for its open and extensible design. The RVV specification reached its frozen 1.0

version in 2021, marking a significant milestone in RISC-V's vector processing capabilities. Since then, both industry and academia have actively contributed to a wide range of vector processor implementations. These include High-Performance Computing (HPC) designed for large-scale processing and complex dataset analysis leveraging scalability, efficiency, and parallelism [20], [36], [37], Deep Neural Network (DNN) [22], [38], [39] and embedded applications [18], [19], [40].

Ara [20] is the first open-source RVV processor, operating at more than 1GHz in 22FDX Global Foundries technology. It integrates the Linux-capable CVA6 core [41] with a vector accelerator featuring 16 double-precision lanes, each accessing 16×64-bit elements of the Vector Register File (VRF). This design delivers 33GFLOPS and 41GFLOPS/W for double-precision matrix multiplication, making it a power-efficient HPC solution with a peak power consumption of 763mW. Yun [21] is the first silicon prototype based on the smallest Ara configuration, featuring 4×64-bit lanes and a Vector Length (VLEN) of 4096-bits. It operates up to 280MHz in 65nm TSMC technology and achieves 2.83GFLOPS and 10.8GFLOPS/W for double-precision matrix multiplication, with power consumption ranging from 57mW to 330mW.

Speed [22], aims to enable efficient quantized DNN inference operating at a frequency of 1.05GHz in TSMC 28nm technology. It supports processing precision ranging from 4-bit to 16-bit and integrates within each of the 4×16-bit lanes an integer-multi-precision (4-8b) tensor unit. Experimental results show that SPEED achieves a peak throughput of 7.37TOPS and an energy efficiency of 1.38TOPS/W for 4-bit operations consuming up to 533mW.

These architectures target large-scale HPC clusters, where their large VRF supports efficient long-vector compute-intensive edge and cloud systems processing across many cores. Although their per-core power consumption is relatively low (<1W), their efficiency relies on high utilization via massive parallelism—making them ideal for large, uniform workloads but inefficient for smaller, fragmented tasks. Additionally, their design targets systems with far more relaxed power constraints than those typical of WUS.

The Spatz architecture [19] operates at over 1GHz in 12LPP technology. It implements an open-source dual-core vector processor based on a cluster that shares 128KB of memory. Each vector core is equipped with a 512-bit VRF divided across four 64-bit lanes. This smaller VRF design enhances utilization, ensuring more efficient use of functional units, particularly on small workloads, making it well-suited for embedded processing domains. The Spatz cluster achieves a peak performance of 15.7GFLOPS and a peak energy efficiency of 95.7GFLOPS/W when executing a double-precision matrix multiplication workload, consuming up to 164mW.

The cluster proposed in Maestro, built on the Spatz cluster template, extends the vector DSP capabilities with frequency domain acceleration by integrating a memory-coupled FFT engine and a tightly coupled tensor unit accelerating ML and general matrix-matrix operations, offering increased performance and energy efficiency.

B. Tensor Cores

AI workloads are dominated by linear operations, such as matrix multiplication, making Tensor Core architectures the preferred solution for accelerating these computations. State-of-the-art implementations explore various architectural approaches to optimize performance, power efficiency, and area utilization. We focus the following survey on tensor engines designed for low-power edge applications.

Gemmini [23] is an open-source systolic array of 256 Computing Elements (CEs) of 8-bit Integer multiply-accumulate designed for inference of DNN that support runtime-programmable stationary weight and output data flows. It has been implemented in Intel 22FFL technology operating at over 961MHz achieving a peak energy efficiency of 73.3GOPS/W.

Tsunami [24], manufactured in 65nm CMOS technology, targets high energy efficiency in DNN training with 1024×16-bit floating-point CEs at 200MHz. Relying on pruning to eliminate unnecessary computations, Tsunami delivers 310GFLOPS and 1.71TFLOPS/W on 16-bit floating-point operations with a maximum power consumption of 419mW.

DARKSIDE [25] is a heterogeneous RISC-V compute cluster for TinyML at the extreme edge. It features 8 DSP-enhanced RISC-V cores, a Depth-Wise Convolution Engine, a DataMover, and a 16-bit floating-point tensor core for matrix operations consisting of 32 FP16 Fused Multiply-Accumulate (FMA) units. Fabricated in TSMC 65nm, it runs at 290MHz and delivers 18.2GFLOPS at 300GFLOPS/W, with a peak power consumption of 213mW.

Our work, Maestro, introduces a unified vector-tensor processing framework, integrating a tensor unit within the vector core to enable tightly coupled cooperation with the VRF. Unlike previous designs [22], [23], Maestro's tensor unit, an area-improved design of the open-source RedMulE architecture [42], supports mixed-precision 8/16-bit floating-point arithmetic (FP8/FP16) and offers up to 48 CEs, enabling efficient embedded processing across diverse workload types by coordinating its execution among other functional units. The inclusion of FP8 and FP16 support balances precision and efficiency for edge WUS applications scenarios with AI and DSP workloads.

C. FFT Accelerators

As frequency-domain workloads grow in complexity, software-based FFT solutions struggle to achieve real-time performance and energy efficiency, highlighting the need for specialized hardware accelerators. Wang et al. [27] introduced an FFT accelerator capable of handling 24-bit precision fixed-point data, while Sinigaglia et al. [26] supports 8/16/32-bit fixed-point precision on the TSMC 65nm implementation of a frequency domain Echoes SoC delivering up to 200GOPS/W and running up to 350MHz with a power envelope of 134mW.

Wang et al. [27] present a 16nm FinFET runtime-reconfigurable $2^n3^m5^k$ FFT fixed-point accelerator designed for multi-standard wireless communication systems such as LTE and Wi-Fi. The accelerator is built using a memory-based architecture optimized for high area efficiency and is integrated

with a RISC-V core to demonstrate a complete software-defined radio (SDR) system operating up to 940MHz, and consuming up to 22.6mW depending on the FFT workload. The use of fused floating-point operations tailored for FFT processors is explored in studies such as [43], [44].

Hitesh et al. [28] investigate the implementation of a high-precision frequency measurement system for MEMS gyroscopes, employing a low-precision (11-bit) floating-point approach for radix-2 FFT computation. The system enhances computational efficiency by optimizing memory access patterns through the matrix transposition technique (CTMT), which is applied alongside the Cooley-Tukey FFT algorithm, a standard approach for FFT computation. The design is capable of performing a 256-point FFT while maintaining scalability to variable-length FFTs. Implemented in 65nm CMOS technology, it operates at 100MHz, while delivering a power consumption of approximately 59.1mW at a 1.2V power supply voltage.

The work by Larry et al. [29] introduces a hardware accelerator specifically designed to enhance the performance of FFTW software library for scientific computing. Fabricated in TSMC 28nm technology, the accelerator adopts a fully unrolled radix-8 FFT architecture. It employs single-precision floating-point arithmetic. Operating at 270MHz with a 0.9V power supply, the accelerator has a power consumption of 126 mW, making it a highly efficient solution for embedded signal processing delivering 16.5GFLOPS/W @ 270MHz.

Most existing hardware FFT implementations either rely on fixed-point arithmetic [45], which is unsuitable for frequency-domain processing in WUS applications; or they do not target embedded applications. In this work, a memory-coupled multi-precision 16/32-bit floating-point radix-2 FFT accelerator design is proposed. It is integrated as a Cluster-coupled Hardware Processing Engine (HWPE), where both the vector core and the accelerator share memory directly through the same interconnect. Unlike traditional FFT accelerators, it does not require large internal buffers and instead relies entirely on the memory interconnect for data access. This approach improves the efficiency of the accelerators, reducing area usage while maintaining strong computing performance. The accelerator supports up to 512 FFT points for FP32 precision and 1024 FFT points for FP16 precision, where both the real and imaginary components are represented in FP32 or FP16, respectively.

D. Wearable Ultrasound Applications and Systems

State-of-the-art embedded WUS systems often use MCU primarily for echo acquisition and data transmission, offloading most processing to external computers.

In [9], a prosthetic hand control system employs four A-mode transducers and a STM32F7 [30] MCU to acquire echoes, perform basic filtering, and transmit data to a Personal Computer (PC) for further feature extraction and classification. Although the MCU features a 32-bit Cortex-M7 running at 216MHz with a DSP unit, it mainly serves for data acquisition, offloading signal processing—i.e., envelope extraction via HT, Principal Component Analysis (PCA) for dimensionality reduction, and ML-based classification—to an external computer. Similarly, the wearable system in [10] uses a TMS320F28379D [31] MCU (dual-core, 200MHz) to digitize analog US signals, apply a 1024-point FFT, and transmit results to a PC, where the LDA model is applied for classification.

Xia et al. [8] propose a portable hybrid surface Electrom-MyoGraphy (sEMG)/A-mode US system for Human-Machine Interface (HMI), integrating a composite sensor armband, signal acquisition modules, a DSP, and a communication module. The US acquisition module consists of two main components: the Signal Excitation Part (SEP), responsible for generating and firing ultrasound waves via a beamformer, and the Signal Conditioning Part (SCP), which amplifies, filters, and optimizes the received echoes. The processed analog signals are digitized and further refined by a dsPIC33EP512MU814 DSP [32], which applies filtering, amplification, and scaling to optimize signal quality for subsequent analysis. This system has also been used by Zeng et al. [4]: in this work, the authors propose an adaptive CNN based on A-mode US for gesture recognition tasks. The raw A-mode US data undergoes a series of preprocessing steps—Time Gain Compensation (TGC), filtering, envelope detection via HT, and log compression—before being fed into the CNN. Then, at test time, the feature extractor part is updated via backpropagation based on a pseudo-label. However, the test-time adaptation is still not performed locally but on an external PC.

Frey et al. proposed an sEMG-triggered Ultrasound System platform [46], which integrates sEMG and US for the long-term monitoring of muscle activity. This system integrates two state-of-the-art wearable devices: BioGAP [47] for biosignal acquisition and WULPUS [48] for ultrasound data acquisition. It operates at 7.8mW when only BioGAP performs single-channel sEMG measurement, with WULPUS in sleep mode, and consumes 29.8 mW when both sEMG and US systems are continuously active. BioGAP is powered by GAP9, a commercial version of VEGA [7] along with an nRF52811 [49] that provides data communication capability through a flexible Bluetooth Low Energy (BLE) communication. GAP9 is an ultra-low-power processor designed for hearable and battery-powered smart devices. It delivers 32.2 GMACs for ML tasks and 15.6 GOPs for DSP workloads, operating within a power envelope of up to 50mW. WULPUS is built around an MSP430 [50] MCU that manages communication with the US transducers and forwards the incoming data to an nRF52832 [51] MCU that streams them via BLE to an external PC that receives both sEMG and US data from BioGAP and WULPUS, respectively.

In contrast to these designs that depend on external compute or offer limited DSP support, Maestro integrates a vector unit, hardware FFT accelerator, and tensor engine in a compact, low-power cluster—enabling efficient, on-chip frequency-domain processing and ML inference for WUS applications without external data transfers.

III. Maestro Architecture

The Maestro SoC, comprises two distinct clock domains, as depicted in Fig. 2: (i) the Host, featuring the main 32-bit core

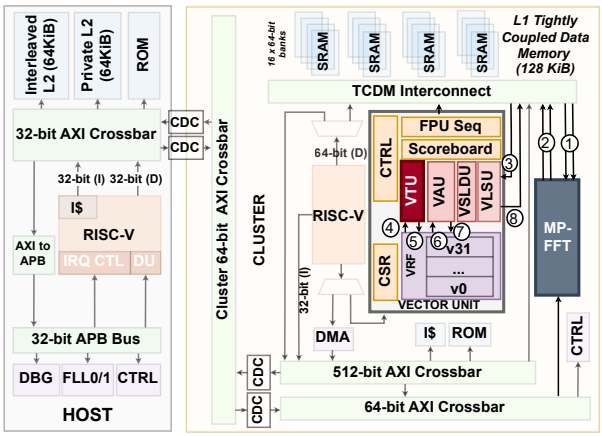


Fig. 2. Maestro architecture with Host, Cluster, and data movement representation. The Host domain includes a 32-bit RISC-V core, 64 KiB L2 memory, and two FLLs. The Cluster domain features a 32-bit RISC-V core, 128 KiB L1 memory, DMA, Vector Unit (VTU, VAU, VSLDU, VLSU, VRF), and MP-FFT accelerator, all sharing the L1 memory.

of the SoC, and (ii) the Cluster, equipped with an advanced VU extended with a Tensor Unit (TU) and a 16/32-bit multi-precision FFT processing engine.

A. Host Domain

The Host domain features a compact, 2-stage, 32-bit RISC-V RV32IMC core [52], optimized for low-cost and energy-efficient operation. This core is supported by a memory subsystem consisting of 64KiB of shared L2 Tightly-Coupled Data Memory (TCDM) organized into 4 interleaved SRAM banks, four additional 16KiB interleaved SRAM banks designated as private memory, and a boot ROM. All components are interconnected through a 32-bit Advanced eXtensible Interface (AXI)-4 crossbar, ensuring efficient access and integration. Additionally, an Advanced Peripheral Bus (APB) interconnect provides access to SoC control registers and a debug interface, enabling system management and debugging capabilities. To ensure robust clocking for both the Host and Cluster domains, the architecture incorporates two dedicated Frequency-Locked Loops (FLLs). In the context of this work, the Host domain serves as a testbench demonstrating the adaptability of the Cluster, which can be incorporated with more complet Host subsystems such as Echoes [26] to meet the needs of diverse applications or architectural requirements.

B. Cluster Domain

The Cluster domain hosts the core contribution of this work: a single-stage RISC-V scalar core [53] featuring a standard RV32IMAFD RISC-V ISA, coupled with the Zve64d-compliant VU [19] based on the RISC-V Vector ISA [1] with support for a reduced subset of RV32IMAFDVZFH_XDMA to target energy-efficient Embedded applications [2].

This architecture enhances the VU with a fully programmable TU based on the RedMulE[3] architecture [42], forming a unified and versatile area-optimized VTU for high-performance and energy efficiency vector-tensor operations (footnote includes detailed registers description). Moreover, the Cluster includes a floating-point multi-precision 16/32-bit FFT accelerator, integrated as HWPE, sharing the memory with the Cluster through a low-latency L1 TCDM interconnect optimizing data movement. As illustrated in Fig. 2, FFT results are written back to the shared L1 TCDM (1,2), which acts as a common resource across accelerators. The MP-FFT accesses the L1 memory directly—without relying on internal buffers—allowing FFT outputs to be immediately available to the VAU and VTU for further processing (3). This minimizes data movement and area overhead while ensuring tight integration among units. Within the compute cluster, the VAU and VTU exploit the shared VRF (4-7) to exchange intermediate results, avoiding memory accesses and improving operand locality. Final outputs from these two units can be stored back into the L1 TCDM (8). The VTU and FFT accelerators are controlled by the core through a memory-mapped interface, which offers greater flexibility and avoids the need for compiler-intrusive ISA extensions. This is acceptable in this case due to the coarse-grained nature of these two units, which execute processing tasks lasting several hundred cycles, making the control overhead negligible.

The memory subsystem of the Cluster is hierarchically organized to optimize data and instruction flow. Instructions and data are primarily loaded into the L2 memory, where the host core triggers Cluster execution by providing the entry point. The scalar core fetches instructions from L2 and includes a private 128B latch-based L0 instruction cache (I$), connected via a 512-bit wide AXI-4 crossbar. When execution begins, a 512-bit/cycle read and 512-bit/cycle write Direct Memory Access (DMA) controller [54] initializes the L1 memory with input data tiles from L2. The L1 memory comprises a full shared 128KiB TCDM organized into 16 interleaved SRAM banks to support high-throughput, low-contention, and parallel access among the units. A 512-bit wide AXI-4 crossbar connects the I$ and boot ROM to the scalar core, supporting high-throughput instruction access. A 64-bit AXI-4 interconnect provides access to control registers, the accelerators, and also serves as the main interface to the Host. Clock Domain Crossing (CDC) FIFO queues at the Cluster boundary ensure safe asynchronous communication across clock domains.

The scalar core pre-decodes and dispatches vector instructions via a generic accelerator interface, which also facilitates communication with the DMA, allowing parallel execution with the VU. The VU features 32×512-bit latch-based vector registers coupled with four Functional Units (FUs), featuring a VLEN of 512-bit and maintaining a throughput of 64-bit/cycle. The Vector Arithmetic Unit (VAU) consists of four mixed-precision Floating Point Units (FPUs) that support FP64, FP32, FP16, BF16, and FP8 (E4M3, E5M2), along with dot product operations featuring higher-precision accumulation, where 8-bit and 16-bit inputs are accumulated into 16-bit and 32-bit results (G16-32/G8-16). Additionally, it includes an Integer Processing Unit (IPU) that supports 8-bit, 16-bit, and 32-bit integer operations. The Vector Slide Unit (VSLDU)

[1] https://github.com/riscvarchive/riscv-v-spec/releases/tag/v1.0

[2] https://github.com/pulp-platform/spatz/blob/main/sw/riscvTests/CMakeLists.txt

[3] https://github.com/pulp-platform/redmule

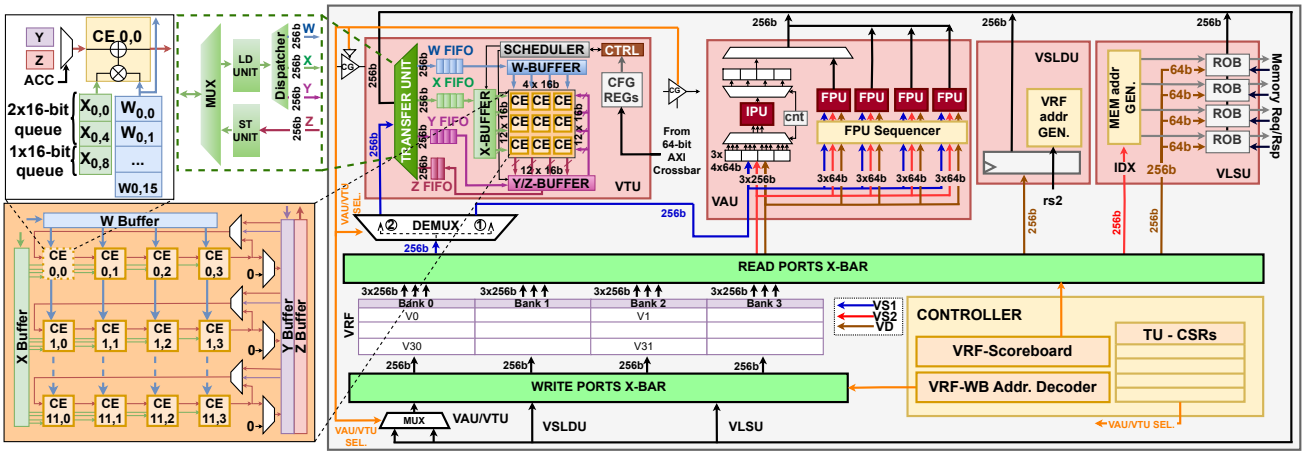


Fig. 3. Vector-Tensor Unit (VTU) architecture and Vector Register File (VRF) interconnect with Vector Functional Units (VFU). The VTU execution is configured by the Scalar core through the 64-bit AXI-4 crossbar. The VTU features the transfer unit and X, W, and Z/Y buffers filling the 4×12 Computing Elements (CEs) datapath. The Tensor Control Status Register (TCSR) manages Vector-Tensor capabilities.

handles vector permutation operations, including vector slide up/down and vector move instructions. The Vector Load Store Unit (VLSU) manages the memory-to-register and register-to-memory transfer, supporting indexed, non-unit-strided, and unit-strided memory accesses. Along these three FUs the VTU described in the following Section enhances tensor operations.

IV. Unified Vector-Tensor Unit

In this Section, we focus specifically on the combination of the VU and the VTU, which is depicted in Fig. 3. To realize the VTU, we augmented the Spatz's FUs with a tightly integrated TU [42] for GEMM and General Matrix-Matrix Operations (GEMM-Ops) execution consisting of two-dimensional array of 12×4 CEs, resulting in a peak performance rate of 48 FMA/cycle.

A. Vector-Tensor Unit Architecture

To feed the datapath, the VTU integrates a Transfer Unit (depicted in Fig. 3) following the HWPE design strategy [4]. It connects to 256-bit read/write crossbars, dispatching input streams to the X, Y, and W buffers and routing output Z streams back during store operations. The X-Buffer, which updates the 12-column inputs of the X matrix; the W-Buffer, composed of 4 shift registers, which efficiently distributes new 64-bit/cycle inputs of the W matrix to the CEs; and the Z-Buffer, which simultaneously stores and forwards computed outputs while pre-loading elements from the Y input matrix, functioning as both an output buffer and a pre-load buffer. The datapath efficiently exploits data reuse: temporal reuse occurs across the rows of the Computing Element (CE) array, where each element of X is multiplied with multiple W values; spatial reuse occurs between columns, where the same set of X elements is reused across different compute paths. Each X value remains active until all its contributions are complete, while W values are broadcast across CE columns to maximize throughput and resource utilization. We emphasize that the names X, W are not tied to the usage as input feature maps and weights: the roles of the two inputs can be swapped arbitrarily, therefore the VTU can operate both in a weight-stationary and in an input-stationary dataflow.

The VRF is equipped with 6×256-bit-wide read ports and 3×256-bit-wide write ports to support high-bandwidth vector operations efficiently. Of the six read ports, three are dedicated to the VAU, providing the 3×256-bits/cycle input bandwidth required for operations such as Vector Fused Multiply-Accumulate (VFMACC), which processes three vector inputs as defined by the vfmacc.vv instruction in the RISC-V Vector Extension (RVV) 1.0 specification [1]. The VAU also utilizes one write port to handle its 256-bits/cycle output. Two additional read ports are allocated to the VLSU for accessing vector data, while the remaining read port is assigned to the VSLDU, supporting sliding operation. The VRF, organized into four banks, features a multi-ported architecture with 3×256-bit wide read ports and 1×256-bit wide write port per bank.

To efficiently arbitrate accesses among competing functional units, a priority-based scheme is applied both for read and write operations on each bank. For read operations, port 0 is primarily dedicated to the VAU, granting it the highest priority for accessing the VS2 operand. If the VAU does not utilize the port in a given cycle, the VLSU is granted access to retrieve the VS2 operand. Similarly, port 1 follows a hierarchical priority scheme, with the VAU having precedence for accessing the VS1 operand. When the VAU does not require the port, the VSLDU gains access to fetch its VS2 operand. Finally, port 2 prioritizes the VAU for reading the VD operand. In the absence of VAU activity, the VLSU is permitted to use the port for VD operand handling.

The write priority scheme for the single write port of each VRF bank ensures efficient operation by giving the highest priority to the VAU for writing arithmetic results. If unused by the VAU, the VLSU gains access for memory operations, followed by the VSLDU, which writes only when neither the VAU nor VLSU requires the port.

A Tensor Control Status Register (TCSR) manages the

[4]https://hwpe-doc.rtfd.io/

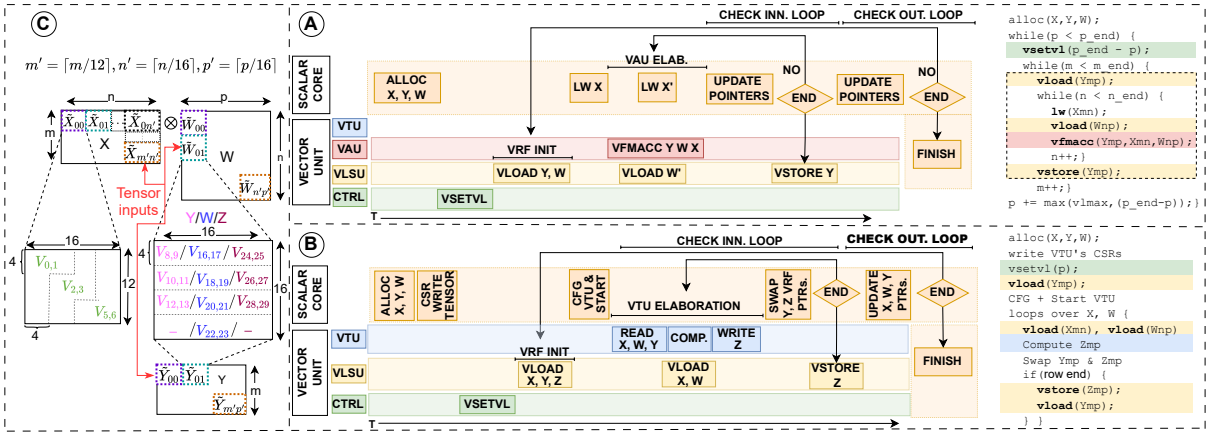


Fig. 4. Vector Arithmetic Unit (VAU) and Vector-Tensor Unit (VTU) temporal execution flow of a general matrix-matrix multiplication accompanied with pseudo code snippets (A-B). Vector Register File (VRF) data tile organization when executing on VTU (C).

clock-gating mechanism to disable the unused FUs, optimizing power efficiency while enabling tensor capabilities by multiplexing the VAU's VS1 read port to the TU. Additionally, the TU and the VAU share the same write port to the VRF.

B. Buffer Reduction

A key aspect of this integration is the VTU's buffer size reduction, which is made possible by leveraging the VRF as a shared resource between vector and tensor operations. In contrast to standalone TUs that use dedicated buffers and require extra memory transfers, our VRF-centric VTU reuses operands already in the VRF—reducing internal buffers, preserving data locality, and lowering L1 bandwidth and power.

Using a single 256-bit read port from the VRF, the design supplies data for all 48 CEs over three clock cycles. Each CE uses its X value for 16 clock cycles, eliminating the need for repeated reads. However, due to the staggered timing of computations across columns—immediate for the first column and delayed by 4, 8, and 12 clock cycles for the subsequent columns—an additional register is introduced for each CE to hold the new X value until it becomes valid for the respective column. This dual-register approach keeps operations synchronized and ensures they run correctly within the required timing. The result is a more efficient X-buffer design that maintains performance with significantly reduced memory overhead.

In the original TU, each CE has a queue of 4×16-bit elements, along with two additional queue slots to synchronize data usage across the processing array, leading to a total buffer size of 576 bytes, calculated as $(2+4)queue * 48CEs * 16bits = 576B$. In the integrated VTU design as shown in Fig. 3, both queue sizes have been halved per each CE, requiring only 288 Bytes: $(1+2)queue * 48CEs * 16bits = 288B$.

This optimization results in a 26.5% reduction of the whole buffer capacity with consequent 19.8% buffer area saving. Given that buffers account for 25.6% of the total RedMule area [42], this optimization reduces the TU area by 6%.

C. VAU and VTU Execution Flow

Fig. 4 depicts the execution flow of a GEMM, comparing how the two computational units—the VAU (A) and the VTU (B)—are engaged to perform the operation. When executing on the VAU Fig. 4a, the scalar core allocates input and output matrices, then fetches and pre-decodes the vector instructions, delegating their execution to the Vector Functional Units (VFUs). At the inner computation loop setup, the VU dynamically sets the vector length and loads Y and W data into the VRF. For smaller kernels, where tensor unit utilization is limited, the vector unit can efficiently execute computations using software-configurable input- or weight-stationary dataflows to maximize data reuse. For larger kernels, we adopt an outer product algorithm to fully exploit parallelism: a scalar from the input tensor is multiplied by a vector from the weight tensor, while the Snitch core and the VLSU pre-load the next scalar and vector inputs to sustain throughput. At the end of each iteration, the partial results are stored back in the memory, and the scalar core reassesses the computation's status, updating the pointers as needed. This execution flow relies on tight cooperation between the scalar core and the VU, as the scalar core handles the continuous pre-decoding, fetching, and dispatching of vector instructions. This behavior is further clarified by the accompanying pseudo-code snippet, which illustrates the execution flow.

When performing elaboration on the VTU Fig. 4b, the scalar core handles the initialization phase, which involves allocating the matrices and enabling tensor capabilities by configuring the TCSR, as described in Subsection IV-A. Following this, the scalar core delegates the task of loading X, W, and Y data from the L1 memory into the VRF, leveraging a Length Multiplier (LMUL)=8 configuration. This approach combines eight vector registers into a larger logical register, optimizing data locality and access efficiency. Fig. 4c shows how data is organized within the VRF during computation. The initial chunks of the input matrices—X and Y (both sized 12×16) and W (16×16)—are loaded into the VRF. The inspection view highlights how each matrix is allocated in the VRF: matrix X is loaded using a strided vector load, while W, Y, and Z

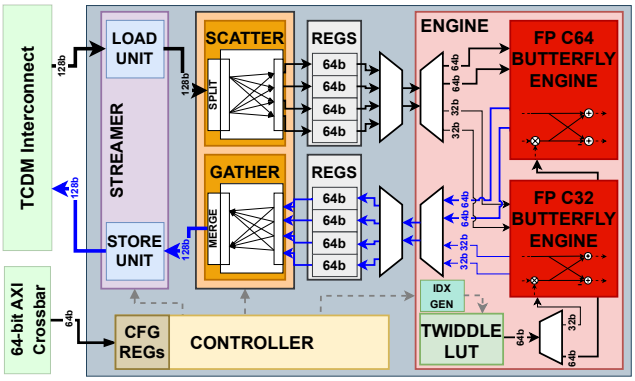


Fig. 5. Multi-Precision FFT Accelerator featuring the Streamer, Controller, Gather and Scatter units, along with two Butterfly Engines: one C32 and one C64.

are loaded with standard vector load instructions. When the VRF initialization is complete, the scalar core configures and triggers the VTU execution, while the VLSU simultaneously loads new X and W values. Upon completion, the VTU streams the computed Z back to the VRF. The scalar core then applies double-buffering, swapping the Y and Z pointers used by the VTU while relaunching the inner loop computation. Once this process concludes, that is, when the current chunked row of matrix X has been fully processed, the VLSU stores the final Z output, and the outer loop check is performed by updating pointers and reinitializing the VRF for the next iteration.

V. Mixed precision FFT accelerator

The MP-FFT[5] accelerator shown in Fig. 5 implements a configurable mixed-precision floating-point FP16 and FP32 radix-2 Cooley-Tukey FFT [55] following the template of the fixed-precision accelerator in [56]. In this Section, we descibe in detail the architecture of the MP-FFT, as well as the novel floating-point butterfly implementation that we propose and the integration in the Maestro system.

A. MP-FFT Architecture

The accelerator targets IEEE-754 single- and half-precision floating-point arithmetic [57] and supports up to 1024-point and 512-point FFT computations for complex data with 16-bit and 32-bit real and imaginary parts, respectively, defined as C32 and C64 formats. Its design implementation consists of several submodules organized in a pipelined architecture described as follows.

The Streamer serves as a specialized DMA unit, efficiently managing data transfers between L1 memory and the accelerator. It manages operand fetch and result storage according to a memory access scheme tailored to the algorithm and accelerator requirements. The MP-FFT accelerator interfaces with memory through four 64-bit ports, partitioned into two input and two output channels. The Controller is responsible for managing the status of various accelerator modules, coordinating their operations, handling the register file used for programming, and controlling the peripheral interconnect. The Gather-Scatter Unit reorganizes the samples. The Scatter module inserts new data from the Streamer into the butterfly registers, while the Gather module extracts processed data and returns it to the Streamer to store it in memory. The Butterfly Registers, comprising a pair of four 64-bit registers, act as temporary storage positioned in front of the Butterfly Unit, which is the main computation engine of the MP-FFT. For coefficient management, the Twiddle Factor Look-Up Tables (LUTs) store precomputed twiddle factors corresponding to the maximum FFT points. The architecture integrates a C64 LUT with 65 elements and two C32 LUTs, each containing 129 elements, ensuring efficient multi-precision support. To fetch the twiddle factor, a dedicated index generator driven by the controller computes the appropriate address for the butterfly operation being processed.

At the core of the accelerator, the Floating-Point Butterfly Unit performs the radix-2 FFT computation. It consists of two Butterfly Engines, one FP16 for C32 precision and one FP32 for C64 precision. Each Butterfly Engine incorporates two fused-arithmetic Dual-Output Sum of Dot Product (SDOTP) modules, which are discussed in detail in Sec. V-B. Since the accelerator reuses larger engines to compute lower-precision butterflies, the accelerator can compute either one C64 or two C32 butterflies at each cycle. In case an FFT C32 is performed, the inputs are rearranged to words with the right data size and are scattered to both butterfly engines reusing the C64 Butterfly for C32 computation. At the output, the results of the butterflies are concatenated to come back to two 32-bit complex words.

B. Floating-Point Butterfly Engine Implementation

Each MP-FFT's Butterfly Engine implements the radix-2 Decimation In Time (DIT) butterfly operation. By exploiting its symmetry, the butterfly can be rewritten so that the real and imaginary parts of both the left and right-sided can be expressed in the form $E \pm (A \cdot B \pm C \cdot D)$. This computation can be conveniently implemented as a high-accuracy fused-arithmetic floating-point operation.

Exploiting this observation, we designed each Butterfly Engine with two fused-arithmetic Dual-Output Sum of Dot Products (DO-SDOTP) units. DO-SDOTP handles subnormal numbers consistently with other IEEE-754 operations, and consists of a five-operand module: four inputs and an accumulator input/output.

We designed the DO-SDOTP module as a parametric design that can be configured to support multiple precision formats ranging from FP8 to FP64; we employed the FP16 and FP32 versions in Maestro. It includes a special input signal, MOD, which inverts the sign of the third operand. As a result, the implemented hardware module is illustrated in Fig. 6.

Compared to a conventional Radix-2 FP datapath, the fused DO-SDOTP datapath avoids precision loss in low-bitwidth FP formats by reducing rounding errors. Each unit instance operates within the largest exponent and mantissa widths defined by the format parameterization, allowing lower-precision computations to be mapped onto the same datapath by assigning

[5]https://github.com/pulp-platform/fft-hwpe/tree/fp-fft-hwpe

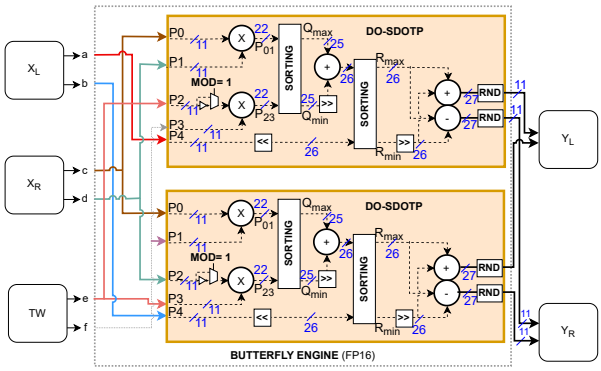


Fig. 6. Microarchitecture and dataflow of Floating-Point Radix-2 FP16 Butterfly Engine.

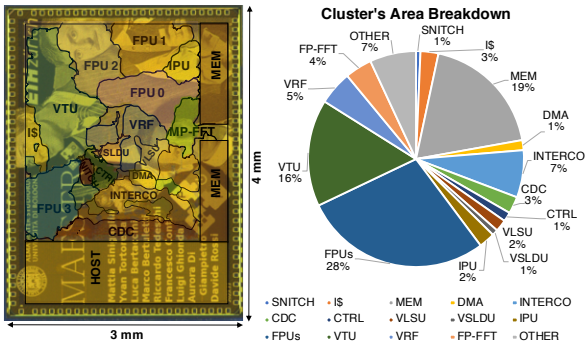


Fig. 7. Chip micrograph and Cluster's area breakdown.

narrower exponent and mantissa fields to specific bit positions. Fig. 6 shows the dataflow computation starting with mantissa multiplication, producing values at double precision. These are sorted, zero-padded, and aligned by right-shifting based on the exponent difference. A second sorting stage incorporates a fifth operand, ensuring precision consistency. Two adders then perform addition and subtraction, with the sign bit inverted in the exponent datapath. Finally, results undergo normalization and rounding, restoring the original precision.

C. FFT Accelerator Integration

The FFT accelerator is integrated into Maestro's cluster as an HWPE engine, tightly coupled with the 128KiB L1 memory through a low-latency interconnect. The programmability of the accelerator is ensured by connecting the HWPE's control port to the 64-bit AXI-4 crossbar. When an FFT computation needs to be accelerated, the Scalar core in the system configures the accelerator through its memory-mapped control registers (detailed in footnote 5) and triggers the execution of a job. This register file holds the key parameters required to execute an FFT job, including the number of FFT points, the address of the input samples, and their data precision.

The accelerator begins by loading a set of left-wing samples from consecutive memory addresses through the Streamer. In the subsequent cycle, it recovers the corresponding right-wing samples for the same butterflies. The Scatter Unit then reorganizes these samples into two of the four 64-bit registers positioned in front of the Butterfly Unit. Once both the left and right wings of the butterflies are stored in the registers, the Butterfly Unit processes them. After computation, the processed partial results are sampled back into the registers. As the computation progresses, the Scatter Unit continues to insert new inputs into the registers, overwriting samples that the Butterfly Unit has already processed. Meanwhile, the Gather Unit works in tandem with the Scatter Unit, extracting outputs from the butterfly engines and passing them to the Streamer, which sends the processed samples back to memory. Assuming no pipeline stalls, the latency between issuing a load request for a pair of input samples and the corresponding store request for the processed outputs is eight clock cycles. To eliminate the need for an internal buffer for operand reshuffling, [56] introduces a scheduling strategy that systematically avoids memory bank conflicts between concurrent loads and stores. To remain compliant with this strategy while accommodating a deeper pipeline than in the original configuration, we modified the access scheme by reversing the order of two consecutive store operations. This adjustment ensures conflict-free access to the shared memory banks during normal operation.

The accelerator outputs data in bit-reversed order, requiring special care during the final FFT stage to avoid banking conflicts when writing reordered samples, as only two bit-reversed samples can be written per cycle. In such cases, the pipeline is temporarily stalled to prevent overwriting meaningful data in the accelerator registers. This limitation allows the full output bandwidth to be utilized only in C64 FFTs, while in C32 FFTs, only half of the bandwidth can be exploited.

To reduce power, a two-level clock-gating scheme is used. At the cluster level, a software-controlled gating cell disables the accelerator when idle. At the accelerator level, only the control unit remains active until execution starts, after which functional units are automatically gated upon job completion.

VI. Physical Implementation and Measurements

Fig. 7 shows the chip micrograph and the area breakdown of the Cluster in the Maestro SoC, highlighting the main building blocks described in Section III. The SoC is implemented in TSMC 65nm CMOS technology and occupies 12mm^2 (3mm $\times$ 4mm) of die area, of which the Cluster domain occupies 7.28mm^2 (2.8mm $\times$ 2.6mm). It has been synthesized with Synopsys Design Compiler 2022.03, while Place & Route has been performed with Cadence Innovus 21.17, and chip finishing with Calibre 2022.3. Power measurements were conducted using the Advantest SoC hp9300, a high-precision automatic test equipment platform. For each kernel, we swept power and frequency to identify the maximum operating point, then ran the kernel in an infinite loop—including initialization and computation phases—to induce dynamic switching activity in the targeted unit (VTU, VAU, or MP-FFT) and enable accurate current measurement.

Fig. 9 illustrates the maximum operating frequency and power consumption across a voltage range of 0.85V to 1.2V for representative computational workloads operating on low data precision FP16-FP32. These workloads include key tasks for embedded machine learning and frequency domain applications, executed on the VAU, VTU, and the MP-FFT

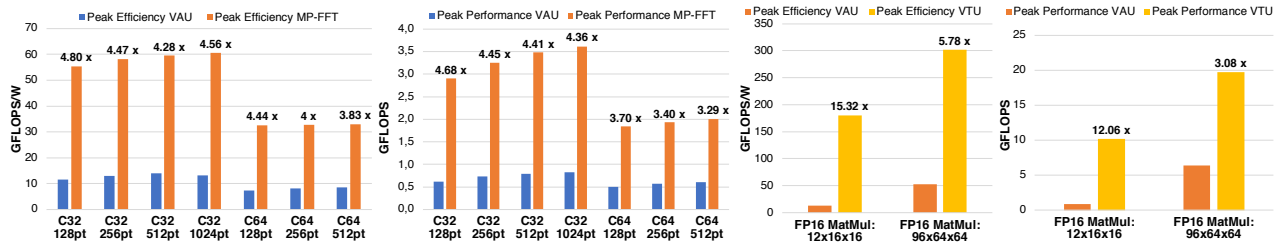


Fig. 8. Peak Performance and Energy Efficiency for FFT and MatMul Kernels over VAU, VTU, and MP-FFT accelerator.

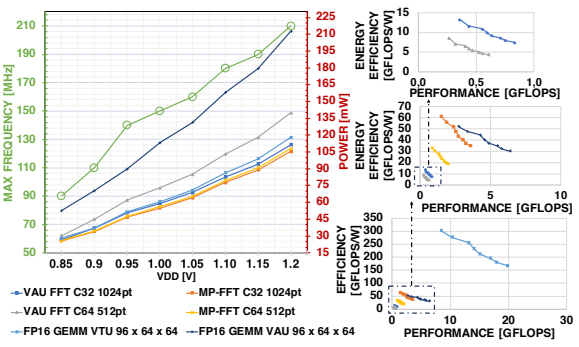


Fig. 9. Power - Frequency and Energy- Performance sweeps for FFT and MatMul Kernels.

accelerator. The Cluster operates within a frequency spectrum from 90MHz at 0.85V to 210MHz at 1.2V, with range power consumption varying from 25mW to 212mW.

Fig.9 also summarizes performance and energy efficiency across operating points, while Fig.8 details peak values for each kernel configuration across varying tensor and FFT workloads. For MatMul, it evaluates a smaller 12×16 matrix multiplication, while for FFT, it examines different data precisions across various point sizes.

The VTU achieves 98% utilization of its internal FMAs, delivering near-to-ideal performance of 47 actual FMA/cycle on a 96×64 FP16 MatMul (out of 48 ideal FMA/cycle). This represents a 3× and 5.78× improvement over the Vector FPUs, achieving, respectively, a peak performance of 19.8GFLOPS and an energy efficiency of 301.7GFLOPS/W. Even with 50% utilization on a 12×16 FP16 MatMul, the VTU enhances performance by up to 15× and energy efficiency by up to 12× for MatMul kernels with the same precision compared to VAU. The reported energy efficiency results are normalized, enabling a fair comparison across the units. Furthermore, despite the larger number of FPUs in the VTU compared to the VAU, the VTU accounts for only 57% of the VAU's total FPU area. This highlights the compactness of the VTU's design, which—combined with its specialization and efficient internal resource reuse described in Section IV-A—achieves high performance and energy efficiency.

Similarly, the FFT processor outperforms the VU in both performance and efficiency across various FFT workloads by up to 4.80× and 4.68×, respectively on C32 128-point. Notably, the C32 FFT 1024-point configuration achieves a peak performance of 3.6GFLOPS and a peak efficiency of 60.6GFLOPS/W. This advantage stems from hardware specialization: the accelerator incorporates dedicated butterfly engines, LUT-based twiddle-factor storage, and a dedicated streamer that feeds data directly into the datapath. By eliminating control overhead, instruction handling, and register-file accesses, the MP-FFT sustains high computational utilization (up to 95.5%) while reducing both execution time and power consumption up to 4.4× and 23%, respectively.

VII. End-to-end Ultrasound Application

In this section, we present the hardware mapping of an example of WUS application pipeline to demonstrate the efficiency of the proposed SoC. The application is inspired by the work by Zeng et al. [4] and consists of a preprocessing chain followed by a CNN for gesture classification and is designed to process and recognize hand gestures using 8 ultrasound transducers, each producing 512 echo samples. While we do not include image-based results, our evaluation focuses on application-level performance for A-mode processing, which suits wearables due to its low memory and power demands. Although B-mode is more resource-intensive, Maestro can still support it by offloading image generation to a host while handling local preprocessing. More generally, the proposed SoC is designed to sustain diverse scenarios and address a wide range of application needs, including fully FFT-intensive pipelines (e.g., TinyProbe [58]), pipelines combining FFT with lightweight non-DNN classifiers (e.g., SonarWatch [59]), and hybrid pipelines that combine FFT preprocessing with DNN-based inference, as demonstrated in our WUS use case. Due to power limits, WUS's perceptual focus, and lack of meaningful sparsity, we target dense, well-structured computation over transformer-based models for better performance and efficiency.

A. Wearable Ultrasound Preprocessing

The preprocessing in Fig. 10 utilizes DSP algorithms to perform amplification, filtering, and envelope detection on A-mode US signal acquisition. Incoming US signals are first amplified using TGC and filtered with Gaussian Filtering (GF) to eliminate noise and enhance relevant information. By leveraging vector operations, the TGC computation processes 512 data samples simultaneously, significantly improving throughput. Implementing the Gaussian filter involves applying a Fourier transform of the US signal using the FFT accelerator and then multiplying transformed data by a Gaussian kernel through the VU, leveraging parallel processing. Finally, the

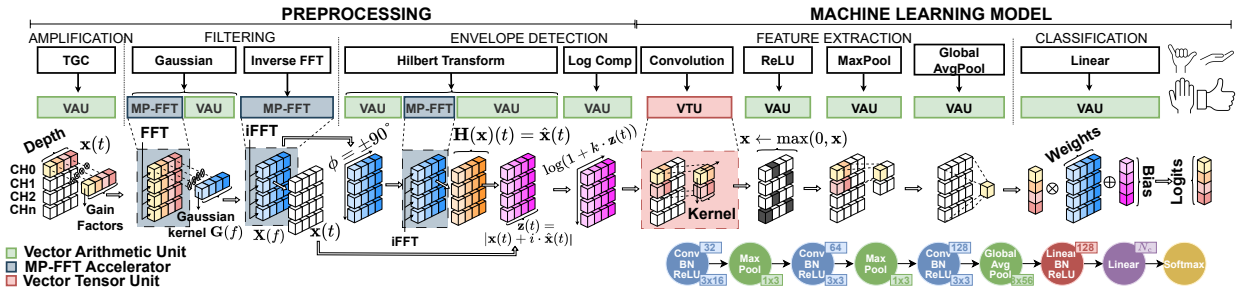


Fig. 10. Wearable Ultrasound A-Mode gesture recognition pipeline on Maestro SoC.

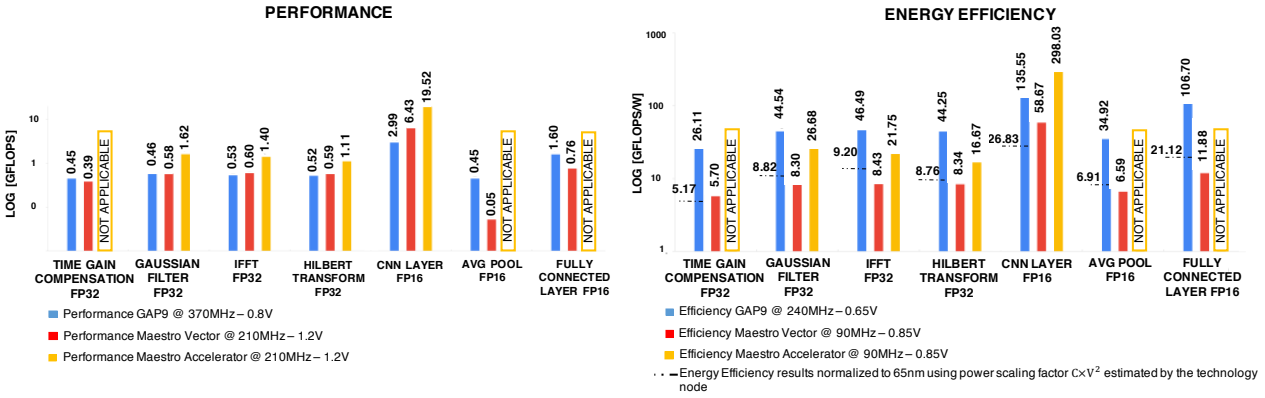


Fig. 11. Wearable Ultrasound A-Mode performance and energy efficiency comparison, showing Maestro executing on the VAU (red), VAU + MP-FFT or VTU (yellow), and GAP9 (blue).

filtered signals are reconstructed via an Inverse Fast Fourier Transform (iFFT), efficiently executed on the FFT accelerator. Envelope detection leverages the HT, with computations distributed between the VU and the FFT accelerator. The process begins with the VU applying a phase shift to the filtered output. Next, the FFT accelerator performs iFFT to reconstruct the signal. Finally, the VU calculates the magnitude to extract the envelope, completing the detection phase. Lastly, the VU performs logarithmic compression on the envelope derived from the HT process, reducing the dynamic range and enhancing the visibility of key signal features.

B. Wearable Ultrasound Machine Learning Model

The CNN model derived from [4] is illustrated in Fig. 10. It comprises three 2D convolutional layers, each followed by Batch Normalization (BN), ReLU activation, and Max Pooling (MP), progressively compressing data along the echo dimension while increasing the number of feature maps to 128. In particular, convolutional and BN layers are fused together and executed on the VTU, whereas MP are mapped to the VU. Then, global average pooling is applied, resulting in a 128-dimensional vector, which is passed through a Fully Connected (FC) layer—also followed by BN and ReLU. Likewise, the FC and BN layers are fused together and executed on the VU. Finally, a linear layer and softmax output the gesture class probabilities.

To deploy the neural networks on Maestro, the Deploy framework [60], [61] can be used, which translates ONNX models into optimized C code for heterogeneous SoCs.

C. Wearable Ultrasound Mapping and Performance Results

The capabilities of Maestro have been evaluated by running the WUS application on the VAU alone and, where applicable, leveraging the combined execution of the Vector-Tensor-FFT engines. The evaluation focuses on latency, performance, and energy efficiency, comparing results against GAP9, a commercial version of the VEGA SoC [7], developed by GreenWaves Technologies and fabricated in GlobalFoundries 22nm FDX technology, with operating ranges from $0.65\,\mathrm{V}$ @ $240\,\mathrm{MHz}$ to $0.8\,\mathrm{V}$ @ $370\,\mathrm{MHz}$.

The preprocessing phase is evaluated using FP32 precision on both Maestro and GAP9, analyzing each individual preprocessing step for a single US channel. As shown in Fig. 11, Maestro achieves 1.62GFLOPS for GF compared to 0.46GFLOPS on GAP9, 1.40GFLOPS for the iFFT versus 0.53GFLOPS, and 1.10GFLOPS for the HT versus 0.52GFLOPS. These results translate to an average performance improvement of approximately $2.6\times$ across the three preprocessing tasks compared to general-purpose execution on GAP9. To demonstrate the effectiveness of the MP-FFT accelerator, we evaluated a TinyProbe-inspired [58] FFT-intensive workload involving the execution of a 512-point STFT 40 times per second; the use of MP-FFT yields a $3.13\times$ speed-up over the vector-only implementation and a $4.71\times$ improvement over GAP9.

The ML model evaluation is performed using FP16 precision on the entire gesture recognition pipeline, where the eight US channels are preprocessed before feature extraction. The

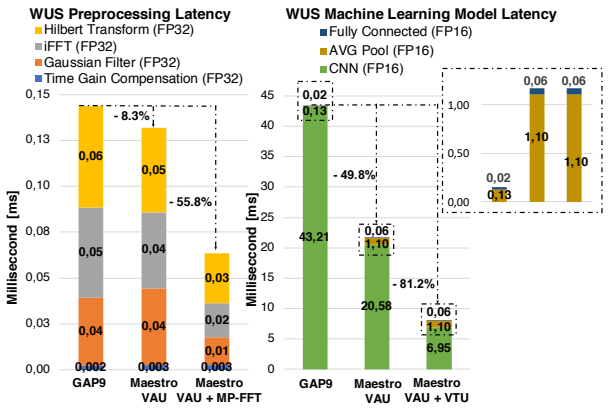


Fig. 12. Wearable Ultrasound A-Mode execution latency on GAP9, Maestro VAU only and Maestro VAU + MP-FFT or VTU.

VTU demonstrates higher performance in CNN-based feature extraction, achieving 19.8GFLOPS and an energy efficiency of 298.03GFLOPS/W. In comparison, GAP9 achieves only 2.99GFLOPS and 135.55GFLOPS/W on the same task. Despite Maestro being implemented in a less advanced TSMC 65nm CMOS process (versus GAP9's 22nm FDX), it delivers more than 2× the energy efficiency in CNN workloads.

Fig. 12 presents the execution latency of the WUS application. While Maestro runs at 1.76× slower frequency than GAP9, its VU and the MP-FFT accelerator enable the preprocessing phase (over a single channel) to be completed in 0.06ms, 2.3× faster than GAP9, which takes 0.14ms, demonstrating the efficiency of the architecture in handling frequency-domain and signal processing tasks. The majority of the ML execution time is consumed by CNN computations, which take 20.58ms on the VAU, 6.95ms on the VTU, and 43.21ms on GAP9. In contrast, average pooling and fully connected layers account for less than 1% of the total execution time considering GAP9 execution. The primary reason for the suboptimal performance of the average pooling layer on the VU is its reliance on reduction operations, which are not well-suited to leverage the VU's capabilities. Additionally, the vector-matrix multiplication used in fully connected layers reduces the parallelism of the VU, limiting its ability to achieve maximum throughput. However, since these two layers contribute only 1% to the total computation time, the overall performance impact remains minimal. The throughput of the VTU (19.8GFLOPS) compensates for the performance limitations of the average pooling and fully connected layers. While the VAU achieves a 2× speedup compared to GAP9, the VTU accelerates CNN-based feature extraction by up to 6.2×, reducing the ML execution time by 81%. Overall, the entire WUS application is executed 5× faster on Maestro. In applications like LSTM inference—where the workload maps efficiently onto the vector unit but does not benefit from the matrix-centric VTU—the tensor unit is clock-gated to save power. In this configuration, the LSTM kernel with 128 input and hidden units (batch size 1) achieves 90% utilization on the vector core—complementing convolutional feature extraction with support for temporal modeling.

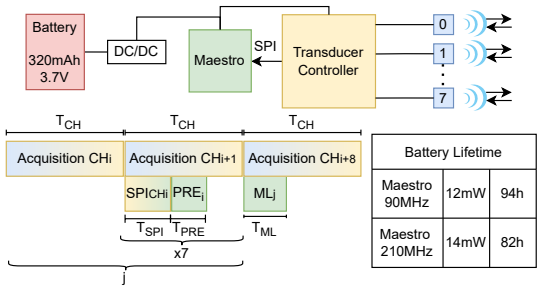


Fig. 13. System-level analytical evaluation of a future Maestro-based WUS processing platform.

D. Power/lifetime of ideal Maestro-based WUS processor

We conceptualized a future Maestro-based WUS platform, inspired by Frey et al. [48], to analytically evaluate the feasibility and energy efficiency of on-device US processing for wearable applications. The study in Fig. 13 considers a 320mAh, 3.7V LiPo battery powering a system composed of eight US transducers, a transducer controller for managing data acquisition, and the Maestro SoC, which receives input samples via SPI. The platform is configured to acquire 512 echo samples per frame at a frame rate of 39Hz.

By pipelining the acquisition phase with the preprocessing and ML model, described in Section VII-A and Section VII-B, respectively, this system fully supports energy-efficient WUS edge processing. The platform achieves up to 94 hours of battery life at Maestro's highest energy efficiency point, consuming 12mW and 2.5mJ, while at peak performance conditions, it operates for 82 hours, consuming 14mW and 2.9mJ.

VIII. SoA Comparison

Table II compares the Cluster of Maestro against a selection of state-of-the-art SoC with vector, tensor, and FFT processors. Compared to YUN [21], Maestro meets the requirement of low power consumption (21mW - 212mW) for a wearable device, while YUN relies on a less efficient architecture tuned for HPC, resulting in lower efficiency (-28% compared to FP32 execution in Maestro's vector unit and 2.57× compared to FP16). The FFT accelerators in [26], [27] are limited to fixed-point precision, making them less suitable for WUS applications with wide data ranges. Maestro's FFT accelerator supports floating-point data, with only 50% lower efficiency than ECHOES [26] in C32int and 6.3× lower than [27], which targets a more aggressive technology node (16nm). Maestro's tensor unit achieves 301.7GFLOPS/W, 19.8GFLOPS@FP16 outperforming YUN [21] in terms of peak performance and [7], [21] in energy efficiency on GEMM kernels. Compared to Darkside [25], Maestro provides similar energy efficiency and slightly better performance with significantly improved flexibility due to the tight Vector-Tensor integration and FFT processor tuned for frequency domain processing. Maestro achieves a notable 2.33× improvement in energy efficiency over VEGA [7] for FP16 tensor operations. While VEGA provides balanced performance across FP16 and FP32, Maestro's architecture enhances FP16 performance by up to 6× on the tensor unit and 2× on the vector unit, with a 0.9×

TABLE II
COMPARISON WITH SOA SOCS.

	[21] *YUN*	[26] *ECHOES*	[27] *LTE/Wi-Fi FFT SoC*	[25] *DARKSIDE*	[7] *VEGA*	*MAESTRO (this work)*
Technology	65-nm TSMC	65-nm TSMC	16-nm FinFET	65-nm TSMC	22-nm CMOS FDSOI	65-nm TSMC
Die size [mm^2]	6	4	1.28	12	12	12
Application	Vector	IoT+FFT	IoT+FFT	IoT+NSAA+ DNN+QNNs	IoT+NSAA+DNN	IoT+Radar+FFT
Cores	RV64GCV0.9	1xRV32FXpulp+FFT	Rocket-Chip	8xRISCY-NN RVC32IMFXpulpNN2	10xRI5CY RVC32IMF-Xpulp	1xRV32IMC + 1xRVV 1.0 ZVE64d
Supply Voltage [V]	0.85–1.2	0.9–1.2	0.57–0.9	0.75–1.2	0.5–0.8	0.85–1.2
Frequency [MHz]	100–280	150–350	40–320	40–290	32e-3–450	90–210
Power range [mW]	57–330	8–133.5	0.46–22	12–213	**1.7e-3–49.4**	21–218
FP precision	FP64, FP32	FP32	FP16, FP32, FP64	FP16, FP32	bfloat16, FP16, FP32	**FP64, FP32, FP16, BF16 FP8 (4,3), FP8 (5,2)**
FPUs	4 × FP64	1 × FP32	1 × FP64	4 × FP32 + 32 × FP16	bfloat16, 4 × FP32	**4 × FP64 + 48 × FP16**
FFT Accelerator Peak Perf	-	10.16 GOPS[A,4] 5.8 GOPS[B,3] 3.2 GOPS[D,1]	8.60 GOPS[C,2]	-	-	**3.6 GFLOPS[B,3] 2 GFLOPS[D,1]**
FFT Accelerator Peak Efficiency	-	200 GOPS/W[A,4] 89.5 GOPS/W[B,3] 41.6 GOPS/W[D,1]	380.55 GOPS/W[C,2]	-	-	**60.6 GFLOPS/W[B,3] 33 GFLOPS/W[D,1]**
Tensor Peak Perf [GFLOPS]	-	-	-	18.2[b]	-	**19.8[b]**
Tensor Peak Eff [GFLOPS/W]	-	-	-	300[b]	-	**301.7[b]**
FP Peak Perf [GFLOPS]	2.83[d] **5.70[c]**	0.16[c]	320 MFLOPS*[,b]	1.02[b] 1.02[c]	3.3[b] 2[c]	12.6[a] **6.6[b]** 3.28[c]
FP Peak Eff [GFLOPS/W]	10.8[d] 23.4[c]	9.68[c]	-	6.2[b] 5[c]	**129[b] 79[c]**	**114.10[a]** 60.23[b] 29.85[c]

FP Precision: a) FP8 - b) FP16 - c) FP32 - d) FP64 **FFT format:** A) C16 - B) C32 - C) C48 - D) C64
FFT points: 1) 512 - 2) 972 - 3) 1024 - 4) 2048 **Notes:** (*) Estimated from the paper

factor in FP32. Furthermore, in an end-to-end WUS application, Maestro's VAU achieves a 2× speedup. In contrast, its VTU accelerates CNN-based feature extraction by up to 6.2× compared to VEGA, resulting in an overall 5× faster WUS execution on Maestro.

IX. CONCLUSION

We presented Maestro, a specialized open-source SoC enabling on-device WUS processing without reliance on remote PCs. Integrates frequency domain and tensor acceleration within a compact RV32-based vector architecture, reducing latency, power, and privacy concerns for wearable use. Fabricated in low-cost TSMC 65nm CMOS, it features Vector-Tensor accelerators and a multi-precision FFT engine, enabling efficient execution of diverse embedded workloads within a 212mW envelope. The WUS application evaluation demonstrates that this WUS-optimized architecture significantly accelerates processing while maintaining ultra-low power consumption of 12mW, with an energy consumption of 2.5mJ. Maestro balances general-purpose vector acceleration (60.2 GFLOPS/W, 6.6 GFLOPS @ FP16) with the efficiency of the Tensor Unit (301.7 GFLOPS/W, 19.8 GFLOPS @ FP16) and the FFT accelerator (60.6 GFLOPS/W, 3.6 GFLOPS @ C32).

REFERENCES

[1] J. S. Letchumanan, S. Gandhi *et al.*, "A mechanically flexible 32-by-32-element pitch-matched ultrasound front-end transceiver with two-stage beamforming for 3d imaging," in *2024 IEEE Custom Integrated Circuits Conference (CICC)*, 2024, pp. 1–2.

[2] H. Hu, C. Hu *et al.*, "Wearable ultrasound devices: An emerging era for biomedicine and clinical translation," *Ultrasonics*, vol. 142, p. 107401, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0041624X24001641

[3] X. Yang, Y. Zhou *et al.*, "Wearable ultrasound-based decoding of simultaneous wrist/hand kinematics," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 9, pp. 8667–8675, 2021.

[4] J. Zeng, Y. Sheng *et al.*, "Adaptive learning against muscle fatigue for a-mode ultrasound-based gesture recognition," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–10, 2023.

[5] M. Eggimann, S. Mach *et al.*, "A risc-v based open hardware platform for always-on wearable smart sensing," in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)*, 2019, pp. 169–174.

[6] J. Liu, Z. Xie *et al.*, "Biowap: A reconfigurable biomedical ai processor with adaptive processing for co-optimized accuracy and energy efficiency," in *2024 IEEE Custom Integrated Circuits Conference (CICC)*, 2024, pp. 1–8.

[7] D. Rossi, F. Conti *et al.*, "4.4 a 1.3tops/w @ 32gops fully integrated 10-core soc for iot end-nodes with 1.7μw cognitive wake-up from mram-based state-retentive sleep mode," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 60–62.

[8] W. Xia, Y. Zhou *et al.*, "Toward portable hybrid surface electromyography/a-mode ultrasound sensing for human–machine interface," *IEEE Sensors Journal*, vol. 19, no. 13, pp. 5219–5228, 2019.

[9] Z. Yin, H. Chen *et al.*, "A wearable ultrasound interface for prosthetic hand control," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 11, pp. 5384–5393, 2022.

[10] A. Bashatah, B. Mukherjee *et al.*, "Wearable ultrasound system using low-voltage time delay spectrometry for dynamic tissue imaging," *IEEE Transactions on Biomedical Engineering*, vol. 71, no. 11, pp. 3232–3243, 2024.

[11] J. Mendez, R. Murray *et al.*, "A-mode ultrasound-based prediction of transfemoral amputee prosthesis walking kinematics via an artificial neural network," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 31, pp. 1511–1520, 2023.

[12] K. Mohit, R. Gupta *et al.*, "A survey on the machine learning techniques for automated diagnosis from ultrasound images," *Current Medical Imaging*, vol. 20, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1573405624000055

[13] M. Gautschi, P. D. Schiavone *et al.*, "Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, 2017.

[14] S. Microelectronics. (2019) Stm32l0 series. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32l0-series.html

[15] Syntiant. (2024) Syntiant. [Online]. Available: https://www.syntiant.com/ndp250

[16] S. Microelectronics. (2024) Stm32n6 series. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32n6-series.html

[17] A. Semiconductor. (2024) Balletto. [Online]. Available: https://alifsemi.com/products/balletto/

[18] D. P. Dabbelt, C. Schmidt *et al.*, "Vector processors for energy-efficient embedded systems," *Proceedings of the Third ACM International Workshop on Many-core Embedded Systems*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:16738646

[19] M. Perotti, S. Riedel *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*, pp. 1–1, 2025.

[20] M. Cavalcante, F. Schuiki *et al.*, "Ara: A 1-ghz+ scalable and energy-efficient risc-v vector processor with multiprecision floating-point support in 22-nm fd-soi," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 530–543, 2020.

[21] M. Perotti, M. Cavalcante *et al.*, "Yun: An open-source, 64-bit risc-v-based vector processor with multi-precision integer and floating-point support in 65-nm cmos," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 10, pp. 3732–3736, 2023.

[22] C. Wang, C. Fang *et al.*, "Speed: A scalable risc-v vector processor enabling efficient multiprecision dnn inference," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 1, pp. 207–220, 2025.

[23] H. Genc, S. Kim *et al.*, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 769–774.

[24] S. Kim, J. Lee *et al.*, "Tsunami: Triple sparsity-aware ultra energy-efficient neural network training accelerator with multi-modal iterative pruning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 4, pp. 1494–1506, 2022.

[25] A. Garofalo, Y. Tortorella *et al.*, "Darkside: A heterogeneous risc-v compute cluster for extreme-edge on-chip dnn inference and training," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 2, pp. 231–243, 2022.

[26] M. Sinigaglia, L. Bertaccini *et al.*, "Echoes: a 200 gops/w frequency domain soc with fft processor and i2s dsp for flexible data acquisition from microphone arrays," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[27] A. Wang, B. Richards *et al.*, "A 0.37mm2 lte/wi-fi compatible, memory-based, runtime-reconfigurable 2n3m5k fft accelerator integrated with a risc-v core in 16nm finfet," in *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2017, pp. 305–308.

[28] H. K. Sahu, E. Sarkar *et al.*, "Implementation of fft using low-precision floating point for rapid high precision mems readouts," in *2023 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2023, pp. 173–177.

[29] L. Tang, S. Chen *et al.*, "A high throughput hardware accelerator for fftw codelets: A first look," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, 2022, pp. 1–7.

[30] S. Microelectronics. (2016) Stm32f7 series. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f7-series.html

[31] T. Instruments. (2025) Tms320f28379d. [Online]. Available: https://www.ti.com/product/TMS320F28379D?keyMatch=TMS320F28379D&tisearch=universal_search&usecase=GPN

[32] M. Technology. (2022) dspic33ep512mu814. [Online]. Available: https://www.microchip.com/en-us/product/dspic33ep512mu814

[33] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Fifth Edition: A Quantitative Approach*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[34] N. Stephens, S. Biles *et al.*, "The arm scalable vector extension," *IEEE Micro*, vol. 37, no. 2, pp. 26–39, 2017.

[35] R.-V. C. 2022. (2022) Risc-v "v" vector extension, version 1.0. [Online]. Available: https://github.com/riscvarchive/riscv-v-spec

[36] M. Sato, "The supercomputer "fugaku"," in *2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2022, pp. 1–1.

[37] F. Minervini, O. Palomar *et al.*, "Vitruvius+: An area-efficient risc-v decoupled vector coprocessor for high performance computing applications," *ACM Trans. Archit. Code Optim.*, vol. 20, no. 2, Mar. 2023. [Online]. Available: https://doi.org/10.1145/3575861

[38] A. Gonzalez, J. Zhao *et al.*, "A 16mm2 106.1 gops/w heterogeneous risc-v multi-core multi-accelerator soc in low-power 22nm finfet," in *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*, 2021, pp. 259–262.

[39] C. Schmidt, J. Wright *et al.*, "4.3 an eight-core 1.44ghz risc-v vector machine in 16nm finfet," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 58–60.

[40] T. Zhao and Z. Ye, "Zerovex: A scalable and high-performance risc-v vector processor core for embedded systems," in *2024 IEEE 35th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2024, pp. 32–33.

[41] F. Zaruba and L. Benini, "The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fdsoi technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2629–2640, 2019.

[42] Y. Tortorella, L. Bertaccini *et al.*, "Redmule: A mixed-precision matrix–matrix operation engine for flexible and energy-efficient on-chip linear algebra and tinyml training acceleration," *Future Generation Computer Systems*, vol. 149, pp. 122–135, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23002546

[43] E. E. Swartzlander and H. H. Saleh, "Fft implementation with fused floating-point operations," *IEEE Transactions on Computers*, vol. 61, no. 2, pp. 284–288, 2012.

[44] A. Kaivani and S. Ko, "Floating-point butterfly architecture based on binary signed-digit representation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 1208–1211, 2016.

[45] A. Pedram, J. D. McCalpin *et al.*, "A highly efficient multicore floating-point fft architecture based on hybrid linear algebra/fft cores," *Journal of Signal Processing Systems*, vol. 77, pp. 169 – 190, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:9264564

[46] S. Frey, V. J. Kartsch *et al.*, "A wearable ultra-low-power semg-triggered ultrasound system for long-term muscle activity monitoring," *2023 IEEE International Ultrasonics Symposium (IUS)*, pp. 1–4, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261706192

[47] S. Frey, M. Guermandi *et al.*, "Biogap: a 10-core fp-capable ultra-low power iot processor, with medical-grade afe and ble connectivity for wearable biosignal processing," in *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2023, pp. 1–7.

[48] S. Frey, S. Vostrikov *et al.*, "Wulpus: a wearable ultra low-power ultrasound probe for multi-day monitoring of carotid artery and muscle activity," in *2022 IEEE International Ultrasonics Symposium (IUS)*, 2022, pp. 1–4.

[49] N. Semiconductor. (2019) nrf52811. [Online]. Available: https://www.nordicsemi.com/Products/nRF52811

[50] T. Instruments. (2016) Msp430. [Online]. Available: https://www.ti.com/microcontrollers-mcus-processors/msp430-microcontrollers/overview.html

[51] N. Semiconductor. (2016) nrf52832. [Online]. Available: https://www.nordicsemi.com/Products/nRF52832

[52] P. Davide Schiavone, F. Conti *et al.*, "Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.

[53] F. Zaruba, F. Schuiki *et al.*, "Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads," *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1845–1860, 2021.

[54] T. Benz, M. Rogenmoser *et al.*, "A high-performance, energy-efficient modular dma engine architecture," *IEEE Trans. Comput.*, vol. 73, no. 1, p. 263–277, Nov. 2023. [Online]. Available: https://doi.org/10.1109/TC.2023.3329930

[55] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965. [Online]. Available: https://api.semanticscholar.org/CorpusID:121744946

[56] L. Bertaccini, L. Benini *et al.*, "To buffer, or not to buffer? a case study on fft accelerators for ultra-low-power multicore clusters," in *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2021, pp. 1–8.

[57] "Ieee standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[58] S. Vostrikov, J. Tille *et al.*, "Tinyprobe: A wearable 32-channel multimodal wireless ultrasound probe," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 72, no. 1, pp. 64–76, 2025.

[59] Y. Shi, C. Yu *et al.*, "Sonarwatch: Field sensing technique for smartwatches based on ultrasound and motion," 2024.

[60] M. Scherer, L. Macan *et al.*, "Deeploy: Enabling energy-efficient deployment of small language models on heterogeneous microcontrollers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, pp. 4009–4020, 2024.

[61] P. Wiese, G. İslamoğlu *et al.*, "Toward attention-based tinyml: A heterogeneous accelerated architecture and automated deployment flow," *IEEE Design & Test*, pp. 1–1, 2025.

X. Biography

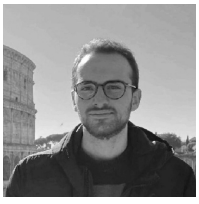
Mattia Sinigaglia received his Master's Degree in Computer Engineering from the University of Bologna in July 2021. He is currently pursuing a Ph.D. in Digital Systems Design at the Department of Electrical and Information Engineering (DEI) of the University of Bologna. His research interest is semiconductor engineering for ultra-low-power devices, mainly targeting hardware-software co-design of RISC-V-based heterogeneous System-on-Chip.

Amirhossein Kiamarzi received his Master's degree in Computer Engineering from the University of Shiraz in 2020. He is currently pursuing a Ph.D. in Digital Systems Design at the Department of Electrical and Information Engineering (DEI), University of Bologna. His research focuses on modeling RISC-V-based hardware cores and accelerators.

Marco Bertuletti received the B.Sc. and M.Sc. degree in Electrical Engineering in Politecnico di Milano, Milano, Italy. He is currently pursuing his Ph.D. at ETH, Zurich, Switzerland, in the Integrated Systems Laboratory (IIS). His main interests are in the design of multi and many-core clusters of RISC-V processors for next-generation telecommunications.

Luigi Ghionda received his Master's Degree in Electronic Engineering in March 2024 from the University of Bologna. He is currently a research fellow in the group of Professor Luca Benini at the Department of Electrical and Information Engineering (DEI) of the University of Bologna. His research interests include the design of PULP (Parallel Ultra-Low Power)-based hardware accelerators and the design of RISC-V-based computer architectures for space SoC.

Mattia Orlandi received his M.Sc. degree in Artificial Intelligence from the University of Bologna, Italy, in 2022. He is currently working toward his Ph.D. in Data Science and Computation under the supervision of Prof. S. Benatti at the Energy-Efficient Embedded Systems Laboratory (EEES Lab), DEI Department, University of Bologna. His research activities involve bio-signal processing with machine learning on low-power computing platforms. He is investigating how to decode EMG signals into spike trains to develop advanced human-machine interfaces.

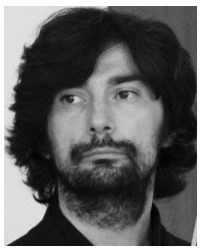


Riccardo Tedeschi received his Master's Degree in Electronic Engineering from the University of Bologna in 2023. He is currently a Ph.D. candidate in Digital Systems Design at the Department of Electrical and Information Engineering (DEI) at the University of Bologna. His research focuses on RISC-V architectures for embedded platforms, with an emphasis on performance and reliability.

Aurora Di Giampietro received her Master's Degree in Electronic Engineering in February 2024 from the University of Bologna. She is currently working as a Research and Development Engineer at Onsemi in Switzerland. Her research interests focus on designing and developing low-power ASICs, with a particular emphasis on energy-efficient solutions for advanced electronic systems.

Yvan Tortorella received the Ph.D. degree in Electronic Engineering from the University of Bologna, Italy, in April 2025. He is currently a Researcher with Fondazione Chips-IT, Italy. His research interests include scalable many-core digital architectures for high-performance and reliable computing.

Luca Bertaccini received the M.Sc. degree in Electronic Engineering from the University of Bologna in 2020 and the Ph.D. degree in Electrical Engineering from ETH Zürich in 2025. He is currently a postdoctoral researcher at ETH Zürich in the Integrated Systems Laboratory (IIS). His research interests include heterogeneous systems-on-chip, energy-efficient hardware accelerators, computer arithmetic, and mixed-precision computing.



Simone Benatti received the Ph.D. degree in electrical engineering and computer science from the University of Bologna, in 2016. Currently, he serves as an Associate Professor at the University of Modena e Reggio Emilia. His research interests focus on energy efficient embedded wearable systems, signal processing, sensor fusion, and actuation systems. He has collaborated with several international research institutes and companies. Previously, he worked for 8 years as an Electronic Designer and R&D Engineer of electromedical devices. He has authored or coauthored more than 120 papers in international peer-reviewed conferences and journals.

Giuseppe Tagliavini is a Tenure-Track Assistant Professor at the University of Bologna. His research interests are focused on programming models, orchestration tools, and compiler optimizations for AI-enabled resource-constrained computing platforms, with a strong emphasis on parallel architectures and accelerators in the context of ultra-low-power IoT end nodes. He is a member of the IEEE and ACM societies.

Luca Benini holds the chair of digital Circuits and systems at ETHZ and is Full Professor at the Università di Bologna. He received a PhD from Stanford University. Dr. Benini's research interests are in energy-efficient parallel computing systems, smart sensing micro-systems and machine learning hardware. He has published more than 1000 peer-reviewed papers and five books. He is a Fellow of the ACM and a member of the Academia Europaea.

Francesco Conti (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the University of Bologna, Italy, in 2016. He is currently a Tenure-Track Assistant Professor with the DEI Department, University of Bologna. From 2016 to 2020, he held a research grant with the University of Bologna and a position as Post-Doctoral Researcher with ETH Zürich. His research is centered on hardware acceleration in ultra-low power and highly energy efficient platforms, with a particular focus on System-on-Chips for Artificial Intelligence applications. His research work has resulted in more than 100 publications in international conferences and journals and was awarded several times, including the 2020 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS Darlington Best Paper Award.

Davide Rossi received the Ph.D. degree from the University of Bologna, Bologna, Italy, in 2012. He has been a Post-Doctoral Researcher with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi," University of Bologna, since 2015, where he is currently an Associate Professor. His research interests focus on energy-efficient digital architectures. In this field, he has published more than 100 papers in international peer-reviewed conferences and journals. He is recipient of Donald O. Pederson Best Paper Award 2018, 2020 IEEE TCAS Darlington Best Paper Award, 2020 IEEE TVLSI Prize Paper Award.