

Frieren: A Fault-Tolerant Reconfigurable Energy-Efficient Computing Architecture with Enhanced Reliability in Harsh Environments

Quan Cheng, Qiufeng Li, Weirong Dong, Mingtao Zhang, Ruilin Zhang, Mingqiang Huang, Hao Yu, Yiyu Shi, Hiromitsu Awano, Takashi Sato, Mehdi Saligane, Longyang Lin, Masanori Hashimoto*

Abstract—In harsh environments such as space, strong radiation effects often induce single-event effects that threaten the reliability of computing systems. Meanwhile, edge artificial intelligence (AI) processors deployed in these conditions must not only tolerate faults but also operate under stringent resource constraints, while still ensuring efficient task execution. Achieving high-performance and energy-efficient computation with adaptive reliability in such harsh conditions is therefore of great importance. This work presents Frieren, a fault-tolerant and reconfigurable computing architecture for reliable operation in harsh environments. A 22 nm system-on-chip (SoC) prototype is implemented to validate Frieren and evaluate its resilience to soft errors. Frieren operates in three primary modes: (1) a high-throughput computation engine mode, (2) a multi-core mode featuring adaptive dual-core lockstep (DCLS) for fault tolerance and programmable parallel computing, and (3) a JTAG-assisted scan-chain-based fault injection (FI) mode. The first two modes fully share processing elements and memory resources, ensuring zero data movement during mode transitions, while the third mode supports pre-deployment reliability evaluation by emulating transient faults. Both irradiation and hardware-level FI experiments are conducted to verify reliability, confirming the robustness of Frieren. Radiation tests of the SoC indicate that DCLS can correct up to about 83% of RISC-V errors, while customized parallel computing in multi-core mode achieves a $17.77\times$ latency reduction. Moreover, the SoC delivers up to 17.18 TOPS/W in computation engine mode and 1.92 TOPS/W in multi-core mode, demonstrating an energy-efficient and resilient platform for AI deployment under harsh conditions. In real workloads, the SoC achieves peak energy efficiencies of 14.72 TOPS/W on SuperYOLO and 12.33 TOPS/W on DROID-SLAM.

Index Terms—Fault tolerance, reconfigurable architecture, single-event effect, edge AI, energy-efficient computing.

I. BACKGROUND AND MOTIVATION

COMPUTE-INTENSIVE tasks are increasingly required in edge and safety-critical platforms to support mission-critical applications (e.g., scientific data processing, com-

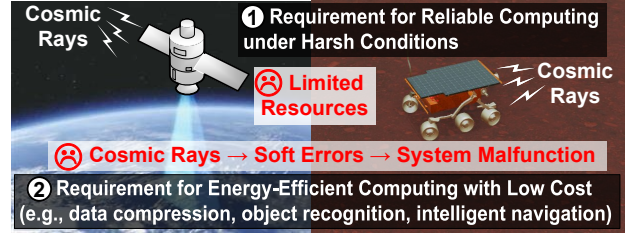


Fig. 1: Challenges of intelligent computing in harsh environments: limited resources and soft errors.

munication, and control) [1], [2]. Intelligent algorithms, including artificial intelligence (AI) for object recognition or autonomous navigation, further increase the demand for such computation [3], [4]. However, operating in harsh environments, such as low-Earth orbit and deep space, poses severe challenges to system reliability and efficiency [5]. In particular, strong radiation conditions introduce single-event effects (SEEs) [6], while limited energy budgets and constrained computing resources further restrict the feasibility of deploying conventional computing systems [7]–[9]. These unique constraints demand a new class of fault-tolerant and resource-efficient system-on-chip (SoC) architectures [10], [11].

As shown in Fig. 1, harsh environments (e.g., cosmic radiation in space or electromagnetic interference in industrial settings) can induce frequent transient faults that flip configuration bits, corrupt memory contents, or disrupt computation, thereby degrading system reliability and potentially causing mission failures [12]. Meanwhile, edge platforms are typically resource-constrained [13], [14], making conventional redundancy techniques such as triple modular redundancy (TMR) impractical due to their high hardware overhead. In safety-critical applications such as autonomous driving and miniaturized satellites, microprocessors are widely used for reliable system control [15]. To enhance reliability, dual-core lockstep (DCLS) with pipeline flush for error correction [16] or triple-core lockstep (TCLS) with direct error correction [17] are commonly adopted. However, these schemes are usually enabled continuously, consuming redundant resources even when high reliability is not strictly required. Moreover, not all logic exhibits the same sensitivity to SEEs, particularly in neural networks (NNs), which inherently show a degree of robustness [18], [19]. Therefore, dynamically configurable error detection and correction is essential to balance reliability, resource efficiency, and performance [20], [21].

Manuscript received 5 December 2025; revised 13 Mar 2026; accepted 24 April 2026.

Qiufeng Li, Mingqiang Huang, Hao Yu and Longyang Lin are with the School of Microelectronics, Southern University of Science and Technology, Shenzhen, China.

Yiyu Shi is with Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, USA.

Mehdi Saligane is with School of Engineering, Brown University, Rhode Island, USA.

Quan Cheng, Weirong Dong, Mingtao Zhang, Ruilin Zhang, Hiromitsu Awano, Takashi Sato, and Masanori Hashimoto are with Department of Communications and Computer Engineering, Kyoto University, Kyoto, Japan. (e-mail: hashimoto@i.kyoto-u.ac.jp)

Furthermore, adaptive task deployment introduces additional design challenges. Depending on mission phases, the system must flexibly support both high-throughput workloads, such as data compression and large-scale object recognition, and highly reliable control tasks including robotic navigation [22], [23], all within the same hardware resources. Under constrained resources, processing large data volumes often leads to high latency and inefficiency. Intelligent Early Discard (IED) [24] mitigates this issue by filtering irrelevant data at early stages to reduce unnecessary computation. Similarly, robotic workloads such as simultaneous localization and mapping (SLAM) [25] face challenges in harsh environments, including radiation-induced faults, limited computing capability, and strict power budgets. As a result, both workloads rely heavily on efficient resource utilization and fault-tolerant processing. To support such applications, the hardware must efficiently execute diverse computational kernels, including linear algebra operations, convolutional processing, and dataflow-oriented preprocessing pipelines [26], [27]. Meanwhile, dynamic allocation of on-chip resources is essential to maximize utilization and throughput [28], [29]. Since data movement across memory hierarchies often dominates energy consumption and latency, effective architectures should not only provide reconfigurable compute units but also minimize communication overhead through tightly coupled compute and memory [30], [31]. These challenges highlight the need for SoCs capable of dynamically balancing performance, energy efficiency, and reliability.

To address these challenges, we propose **Frieren**, a fault-tolerant and reconfigurable architecture with enhanced reliability targeting harsh environments, validated with a 22 nm SoC prototype. The main contributions are as follows:

- **Multi-modal operation for adaptive reliability:** Frieren integrates four operation modes (i.e., three primary modes and an additional mixed mode). The first is a computation engine mode designed to accelerate matrix and convolution operations. The second is a multi-core mode that supports programmable parallel computing and incorporates adaptive DCLS-based fault tolerance. The third is a mixed mode that allows the computation engine and multi-core execution to operate simultaneously. These modes are tightly integrated and fully share the processing elements (PEs) and memory resources. This heterogeneous design significantly reduces area overhead and minimizes data movement between modes, thereby improving both energy efficiency and execution latency. In addition, a JTAG-assisted fault-injection (FI) mode is implemented for system-level reliability analysis.
- **In-situ reliability analysis through fault injection:** The JTAG-assisted FI mode is implemented by integrating the conventional DFT flow with the JTAG protocol. The scan chains (SCs) are used to access on-chip flip-flops (FFs), while the system bus provides access to memories, enabling direct emulation of transient faults under harsh conditions. This capability supports pre-deployment reliability evaluation and facilitates adaptive configuration of error detection and correction mechanisms.

- **Experimental validation under real-world scenarios:**

To evaluate Frieren's effectiveness, we test representative workloads including SuperYOLO [32] for in-orbit computing and DROID-SLAM [33] for robot-assisted simultaneous localization and mapping. The 22 nm SoC prototype demonstrates superior energy efficiency across these tasks. Moreover, reliability is validated not only through controlled FI experiments, which enhance reliability tuning for different workloads, but also through real neutron and alpha radiation tests, confirming consistency between fault emulation and physical irradiation results.

Through this combination of adaptive modes, FI-based analysis, and irradiation tests, Frieren provides a unified platform capable of sustaining reliable and energy-efficient AI workloads in radiation-prone environments. This work demonstrates that by tightly co-designing fault tolerance, reconfigurability, and energy efficiency, edge AI SoCs can achieve both high throughput and robust reliability in harsh conditions.

This paper offers deeper insights and additional materials compared to the preliminary version presented in [34]. Specifically, we provide a more comprehensive explanation of the architecture's reliability features, including the adaptive DCLS mechanism, pipeline rollback support, and radiation-aware reconfiguration strategies. We further introduce a JTAG-assisted FI framework that enables single-event upset (SEU) emulation and detailed characterization. In addition, we extend the multi-modal execution capabilities by supporting a new FI mode and mixed-mode operation. The application scope is also expanded with full implementations of DROID-SLAM and SuperYOLO, enabling deeper analysis of different workloads. Finally, the experimental section is significantly enriched to provide complete reliability evaluation and detailed architectural breakdowns to validate the newly introduced components.

The remainder of this paper is organized as follows. Section II introduces the overview of Frieren, including its architecture, PE structure, and enhanced DCLS. Section III presents the multi-modal operation, such as computation engine mode, multi-core mode, and FI mode. Section IV discusses the performance evaluation of a 22 nm SoC prototype based on Frieren. Section V presents irradiation and FI experimental results evaluating the reliability of the prototype. Section VI compares the prototype with prior state-of-the-art (SOTA) designs. Finally, Section VII concludes the paper.

II. OVERVIEW OF PROPOSED ARCHITECTURE

A. System Architecture

Fig. 2 illustrates the proposed architecture (Frieren) for energy-efficient and reliable computing in harsh environments, implemented as a 22 nm SoC prototype for validation. It features eight processing cores (PCs), each consisting of two 3-stage RISC-V cores augmented with DCLS checkers at the fetch, decode, and execute pipeline stages. These checkers can be selectively enabled or disabled to provide adaptive fault tolerance depending on application scenarios. When disabled, the RISC-V cores operate independently; when enabled, the DCLS checkers detect errors, trigger pipeline flush and memory integrity check for error correction. The cores are

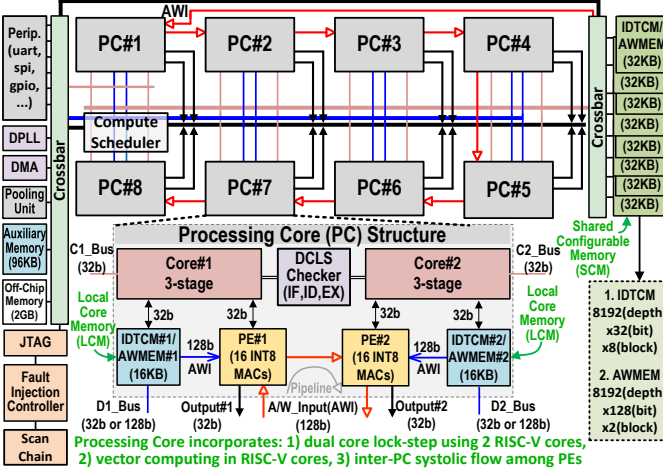


Fig. 2: Architecture of Frieren for energy-efficient and reliable computing, implemented as a 22 nm SoC prototype.

also optimized for programmable parallel computing, supporting inter-PC systolic dataflow across PEs. Data paths are interconnected through crossbar switches, enabling efficient communication between cores and memory blocks. These configurable memory blocks can be flexibly configured as Instruction/Data Tightly Coupled Memories (I/DTCMs) or activation/weight memories (A/WMEMs) to optimize data storage and retrieval. The system provides 512 KB of configurable memory, including 256 KB (32KB $\times$ 8) of Shared Configurable Memory (SCM) outside the PCs and 256 KB (16KB $\times$ 16) of Local Core Memory (LCM) inside each PC. Both SCM and LCM are configurable and can be flexibly allocated to different functions depending on the workload. An additional 96 KB auxiliary memory is provided for data buffering or other purposes. Memory accesses are supported through 32-bit interfaces for the RISC-V cores and JTAG, and 128-bit interfaces for the DMA and PEs to sustain high-throughput data movement. For reliability evaluation, a JTAG-assisted FI mechanism is integrated into the system. A dedicated FI controller, coupled with the on-chip JTAG module, enables precise bit-flip injection in FFs and memory cells, while interacting with external tools through OpenOCD [35].

B. Processing Element and Customized Instructions

Given the diverse computational patterns of modern NNs, flexible PE architectures are essential for efficiently supporting different dataflow and precision requirements [36], [37]. The PE structure is shown in Fig. 3. Each PE integrates 16 INT8 multiply-accumulate (MAC) units. Since most NNs involve large volumes of data, the final results cannot be computed in a single step. Instead, each PE accumulates intermediate results across multiple iterations. To exploit the high data-reuse characteristics of NNs, the PE supports configurable reuse across different dataflows. Specifically, each PE generates multiple intermediate results (e.g., from different output channels), which are temporarily stored in 16 dedicated buffers for accumulation. After accumulation is completed, the final results are forwarded to subsequent processing stages.

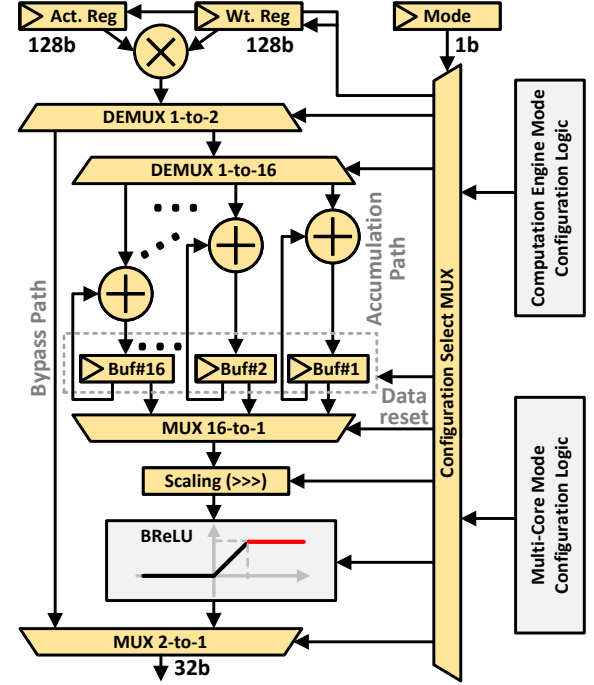


Fig. 3: Processing element structure.

M.BP: 16-way 8-bit MAC, result $\rightarrow$ RD (Bypass Path)

31	25	24	20	19	15	14	12	11	7	6	0
0000001	RS2	RS1	101	RD	1111011						

M.BUF: 16-way 8-bit MAC, result $\rightarrow$ Buf[idx] (Accum. Path)

31	25	24	20	19	15	14	12	11	7	6	0
0000001	RS2	RS1	110	IDX	1111011						

M.OUT: Read Buf[idx] and write result $\rightarrow$ RD (Accum. Path)

31	25	24	20	19	15	14	12	11	7	6	0
0000001	xxx	IDX	110	RD	1111011						

M.CLR: Clear selected Buf using one-hot mask (Accum. Path)

31	30	15	14	12	11	7	6	0
x	imm[15:0] (mask)	001	RD	1011011				

M.SCALE: Applying scaling factor to Buf data (Accum. Path)

31	15	14	12	11	7	6	0
imm[16:0] (>>>)				011	RD	1011011	

M.BRELU: Boundary config. for BReLU (Accum. Path)

31	15	14	12	11	7	6	0
imm[16:0] (upper threshold)		100		RD		1011011	

Fig. 4: Customized instructions for PE scheduling.

Furthermore, to enhance computing efficiency, several customized instructions (Fig. 4) are integrated into the RISC-V Instruction Set Architecture (ISA). These instructions extend the PE's functionality beyond standard MAC operations. First, M.BP performs a 16-way INT8 MAC and directly writes the result to RD via the bypass path, supporting low-latency vector operations. In contrast, M.BUF stores the intermediate MAC results into a specified buffer entry, enabling iterative accumulation across multiple cycles. The final accumulated value can be retrieved using M.OUT, which outputs the selected buffer content to RD. To manage buffer operations, M.CLR resets one or more buffer entries based on a one-hot mask, ensuring efficient re-initialization without software overhead. For data

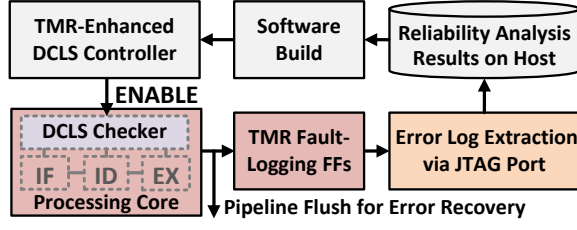


Fig. 5: Enhanced DCLS configuration flow under varying application requirements and environmental conditions.

TABLE I: Signals and DCLS coverage across pipeline stages.

Category	IF	ID	EX
Total signals	296	598	994
Signals covered by DCLS	289	592	683

normalization, M.SCALE applies a programmable scaling factor, making the PE flexible for quantization and range adjustment. Finally, M.BRELU allows boundary configuration for the boundary-aware activation function (BRELU [38]). BRELU improves robustness by capping activation values, where thresholds can be programmed to enforce non-linear activation behavior. These customized instructions collectively provide flexible dataflow control, efficient accumulation, and adaptive non-linear processing, significantly improving the performance and usability of the PE for NN workloads.

C. Enhanced Dual-Core Lockstep

In Fig. 5, Frieren employs an enhanced DCLS mechanism that dynamically adjusts its activation according to application demands and environmental harshness. Since enabling DCLS continuously can restrict system throughput, it is selectively activated only when reliability is critical. For instance, DCLS is enabled in safety-critical or potentially system-crashing routines to ensure fault tolerance, while non-critical components (e.g., UART logging or AI workloads with inherent robustness) operate without DCLS to maximize performance.

The TMR-enhanced DCLS controller dynamically enables or disables DCLS checkers in all pipeline stages (IF, ID, and EX). During program execution, the checkers continuously compare the outputs of two lockstep cores. When an error is detected, a pipeline flush is immediately triggered to prevent error propagation and restore correct execution. If the error originates from the IF stage, the corresponding instruction in ITCM is further checked to verify its integrity. The event information is captured in TMR fault-logging FFs and extracted through the JTAG interface. In addition, the SC and internal bus provide access to the FFs and memory blocks via JTAG, enabling deeper visibility into fault propagation paths and system state. These collected data logs are then analyzed on the host to evaluate reliability under various conditions, and the software build updates the configuration accordingly. These results are fed back to the controller, which refines DCLS enable policies for subsequent runs.

To better understand the protection scope, Table I summarizes the number of signals in different pipeline stages and the corresponding coverage of DCLS protection in PCs. Most

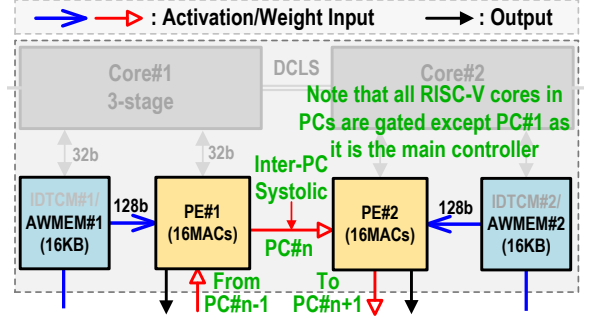


Fig. 6: Computation engine configuration within a single PC.

pipeline signals are included in the DCLS coverage. A small number of control signals, such as pipeline flush and reset signals, are excluded since they are directly related to the operation of the DCLS mechanism itself. Fault tolerance is achieved through DCLS between the two cores in each PC. When DCLS is enabled, the RISC-V core and the control path of the PE computation are covered by lockstep checking, while the PE data path is excluded from protection. This design choice mainly stems from the large bit-width of the PE datapath and the inherent robustness of NN workloads to minor data perturbations. As a result, 305 PE-related data signals are not included in the coverage, while the majority of pipeline control and data signals remain protected.

III. OPERATING MODES IN FRIEREN

A. Computation Engine Mode

In computation engine mode, the SoC is configured as a unified AI accelerator (Fig. 6). The primary workload is executed by the PE array, while PC#1 mainly performs lightweight coordination tasks such as configuration. Therefore, only the core in PC#1 remains active to serve as the global controller for task scheduling, while the remaining PCs are clock-gated since they do not participate in computation in this mode. This design minimizes unnecessary power consumption and allows the available hardware resources to be dedicated to accelerator execution. Inside each PC, PEs equipped with 16 MACs operate on complementary data streams: one stream supplied from LCM (blue path) and the other from SCM (red path). Depending on the application scenarios, the two paths alternately carry activations or weights. Each PE performs local accumulation across multiple cycles, storing intermediate results until the final outputs are produced. The outputs are then passed along the black path for collection.

At the system level, Fig. 7 highlights how all PCs collaborate to maximize throughput with high data reuse. The blue paths correspond to data provided locally by LCMs, while the red paths correspond to data provided globally by SCMs, forming a complementary distribution fabric. These paths allow the same data to be reused by multiple PEs without redundant memory fetches, significantly reducing traffic and improving efficiency. A lightweight compute scheduler manages timing and task sequencing across the systolic-style network of PCs, ensuring outputs from different PEs are aligned and transferred

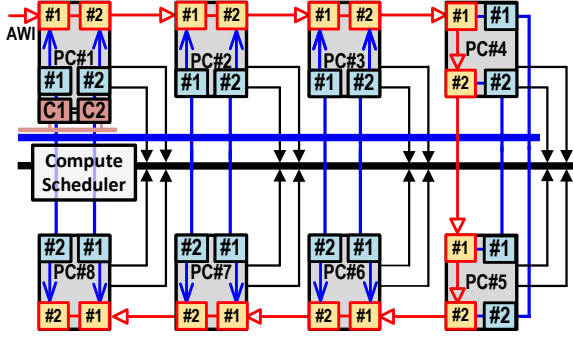


Fig. 7: Computation engine mode with high data reuse.

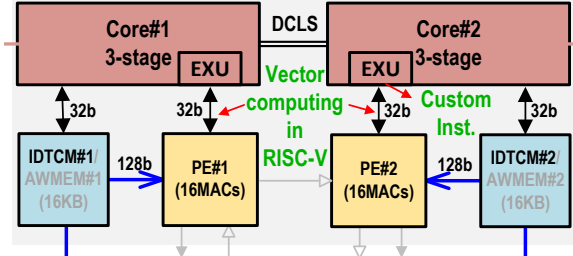


Fig. 8: Multi-core mode configuration.

in order. Once aligned, the final results are transmitted externally through DMA. By exploiting complementary intra-PC and inter-PC paths, computation engine mode reduces data movement and sustains high PE utilization, making it well suited for dense, matrix/convolution-dominated workloads.

B. Multi-core Mode

In contrast, Fig. 8 shows the multi-core configuration, where all PCs are fully activated. Each PE is directly coupled to its corresponding RISC-V core through a 32-bit interface. This tight coupling embeds the PE into the execution pipeline, allowing compute-intensive parallel operations seamlessly. The RISC-V cores are extended with customized instructions as discussed in Section II-B, enabling direct invocation of PE primitives without complex software orchestration and significantly accelerating non-matrix/convolution workloads, such as parallel reductions and scaling, element-wise operators, quantization/normalization, data layout transforms (packing/unpacking), and other pre-/post-processing kernels.

Each RISC-V core and PE is equipped with its own LCM (local I/DTCM), offering 16 KB of low-latency storage for instructions and data. Combined with a 128-bit interconnection between the PEs and on-chip LCM, this design minimizes off-chip memory access and provides sufficient bandwidth for parallel workloads. Moreover, when DCLS is enabled, the PE datapath remains outside the lockstep coverage. As a result, PCs can still execute the same instruction stream while processing different data through the PE datapath. This allows parallel data processing to continue, effectively mitigating the throughput degradation introduced by DCLS while maintaining system reliability.

Fig. 9 illustrates the system-level view of multi-core mode. All PCs collaborate with their PEs and LCMs, providing

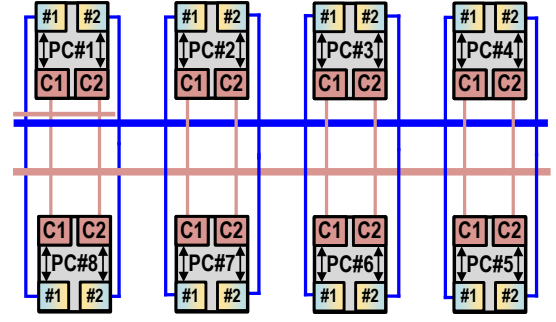


Fig. 9: Multi-core mode with customized parallel computing.

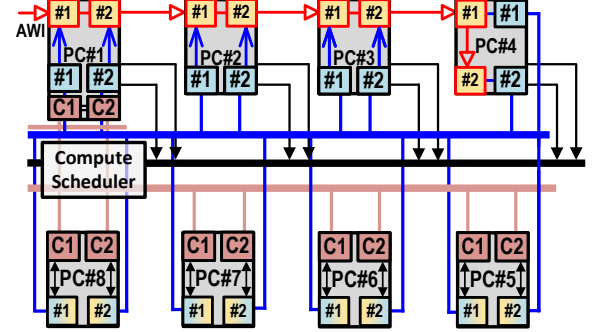


Fig. 10: Mixed mode with computation engine and multi-core.

distributed parallelism while maintaining flexible memory-to-PE data mapping. Customized instructions dispatched from each RISC-V core directly engage the co-located PEs, and the blue/red (LCM/SCM) interconnects allow complementary data streams to be efficiently shared across PCs. Importantly, Frieren reuses the PEs and memory blocks across both the computation engine mode and the multi-core mode, significantly reducing hardware overhead. As a result, extending the design from a standalone computation engine to the full multi-core mode with DCLS incurs only an 11.9% area overhead, which is mainly attributed to the additional RISC-V cores.

C. Mixed Mode

Frieren also supports a mixed mode of operation as shown in Fig. 10. In this mode, a subset of PCs from PC#1 to PC#n ($n < 8$) are grouped together to form a small computation engine, while the remaining PCs from PC#(n+1) to PC#8 operate in a multi-core mode. In this configuration, PC#1 serves as the entry point of the small engine, and the PCs from PC#1 to PC#n form a continuous pipeline where data flows sequentially through the PEs to complete convolution or matrix computations. This organization keeps the interconnect lightweight while enabling an efficient streaming-style execution within the computation engine. The remaining PCs operate independently in multi-core mode, providing additional task-level parallelism for other workloads. Although this mixed mode offers flexibility, it is not frequently used in practice. For most workloads, once the model is divided into multiple tiles or chunks, each partition can fully utilize the available hardware resources across all PCs. Only a few boundary tiles or residual

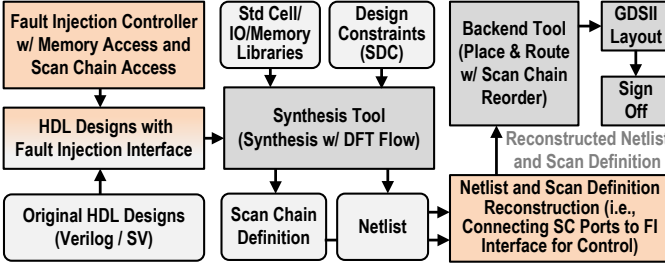


Fig. 11: DFT-based fault injection flow in Frieren.

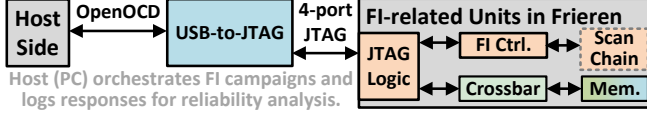


Fig. 12: Fault injection framework for reliability evaluation.

chunks that cannot saturate the chip resources benefit from the mixed mode. Thus, this configuration mainly serves as a fallback for edge cases to maintain full utilization and prevent idle resources during partial workloads. This mode is therefore mainly suitable for tiny workloads.

D. Fault-Injection Mode

Traditionally, software-based FI methods emulate temporal faults purely in software via backdoor access, resulting in extremely low efficiency due to the high computational cost of bit-level fault modeling [39], [40]. Such approaches can be orders of magnitude slower than hardware execution, making large-scale fault coverage impractical. In contrast, hardware-based FI methods inject faults directly into the design [41], enabling near real-time evaluation and seamless integration with existing DFT flows for scalable reliability testing.

Frieren adopts a hardware-oriented strategy by integrating a dedicated FI mode for systematic SEU evaluation. The FI mechanism is implemented through a JTAG-assisted FI controller and hierarchical SCs, enabling precise bit-level access to on-chip FFs and memory cells, as shown in Fig. 11. The SCs are inserted using a standard DFT flow during RTL synthesis and P&R to provide controllability and observability of sequential elements. After synthesis, the scan ports are connected to the on-chip FI controller, forming “internal scan chains” without requiring external scan pins. During FI, faults in FFs are injected via SC shift/update operations controlled by JTAG while the SC circulates continuously. When the circulating SC reaches a target FF, its scan-out value is inverted and fed back to the scan-in path to emulate a bit flip, whereas non-target FFs simply pass through the scan data. Memory FI is performed separately through JTAG-based bus accesses to modify memory contents. The FI controller and its JTAG logic are marked as dont-touch during synthesis to prevent their FFs from being replaced by scan FFs, avoiding potential deadlocks that could otherwise halt the FI process.

Following this DFT-based flow, a complete FI framework (Fig. 12) is established, enabling bit-level FI for reliability analysis. On the host side, OpenOCD is extended to orchestrate FIs. Through JTAG, the controller can halt execution,

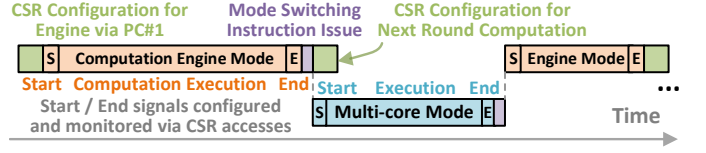


Fig. 13: Mode switching timeline across different modes.

shift scan configurations, inject bit flips, perform memory access, and resume normal operation. Specifically, all transitions and operations are fully observable and controllable, enabling reproducible FI campaigns. High-level commands such as scan shift, flip-at-index, memory access, and checkpoint/restore are provided to classify observed failures, including silent data corruption errors (SDC) and detected unrecoverable errors (DUE), enabling detailed reliability evaluation.

Moreover, to support runtime FI without introducing spurious glitches, Frieren incorporates a dedicated FI clock domain alongside the normal multi-core and compute-engine clocks, following the clock-manipulation scheme described in [42]. In the FI controller, a glitch-free clock-handover mechanism ensures safe transitions of all logic elements into the FI domain. The FI controller orchestrates the entire sequence, including halting the system, initiating cross-domain handshakes, switching to the FI clock, performing scan or memory operations, and finally restoring the original run clock.

Despite its comprehensive functionality, the FI logic remains lightweight. The FI controller, including the clock-switch and scan-control logic, incurs only about 0.2% area overhead, as verified in the 22 nm implementation. This enables practical and cost-effective in-situ FI for pre-deployment SEU evaluation without compromising the deployed SoC resources.

E. Mode Switching and Memory Management

The mode switching process is illustrated in Fig. 13. The mode switching instruction requires 3 system cycles to complete. Afterward, the start signal for the execution phase is configured through control and status register (CSR) accesses via the internal bus. Without bus contention, this configuration requires 5 cycles. These cycles are sufficient for the mode transition to stabilize, completing the mode switch. The CSR configuration for the computation engine mode involves programming about 40 registers, which requires about 85 cycles to complete. However, this configuration can be performed while the system operates in multi-core mode, allowing the configuration latency to overlap with normal program execution. As a result, the system does not experience idle cycles due to this configuration overhead. The only resource impact is that PC#1 is used to perform the configuration, meaning that only PC#1 incurs about 85-cycle overhead. It is worth noting that if the system continuously operates in computation engine mode, the CSR configuration overhead cannot be hidden, since the configuration of the next round can only begin after the previous engine-mode computation has finished.

In Fig. 14, the memory hierarchy of Frieren consists of SCM and LCM, both of which can be flexibly configured. Partition boundaries are tunable per application, allowing adaptive allocation between instruction, data, activation, and

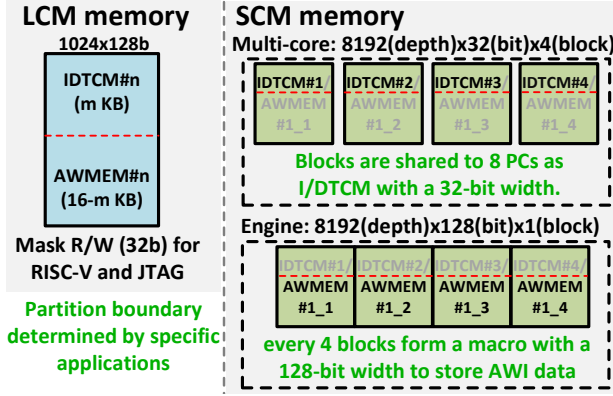


Fig. 14: Mode-specific on-chip memory management.

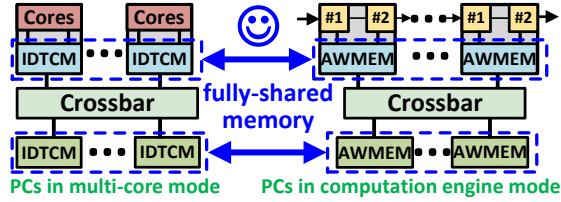


Fig. 15: Memory fully shared in different modes.

weight storage. In computation engine mode, four SCM blocks are aggregated into a single 128-bit macro, and with eight blocks in total, up to two macros can be constructed. This supports ping-pong access, where one macro performs reads while the other performs writes, sustaining high throughput. The LCM acts as A/WMEM, providing localized storage of activations and weights. In multi-core mode, both SCM and LCM can be flexibly configured as I/DTCM. This allows each RISC-V core to efficiently access low-latency local memory. In mixed mode, PCs #1–#n are grouped into the computation engine, which requires at least one 128-bit SCM macro. Two macros are required when ping-pong buffering is desired for high-throughput computing. The remaining PCs operate in multi-core mode, where their LCM functions as I/DTCM. For engine-side PCs, the LCM is reconfigured as A/WMEM to provide high-throughput feed of activations and weights. This design ensures efficient memory utilization across modes while avoiding unnecessary data movement.

Besides, as illustrated in Fig. 15, the memory blocks are fully shared between the multi-core and computation engine modes, enabling direct data access. This unified memory design allows both modes to operate on the same data space seamlessly, achieving zero data movement.

IV. PERFORMANCE EVALUATION

A. Fabricated SoC Prototype

Fig. 16 shows the die micrograph of the SoC prototype, fabricated in a 22 nm CMOS process with a total area of 2.35 mm². The SoC integrates 8 PCs, each containing two RISC-V cores, two PEs, and two reconfigurable LCM blocks for flexible use as instruction, data, or activation/weight storage. In addition, the SCM blocks and auxiliary memory are included to improve data reuse and reduce off-chip traffic. All PCs are interconnected through a crossbar for efficient data

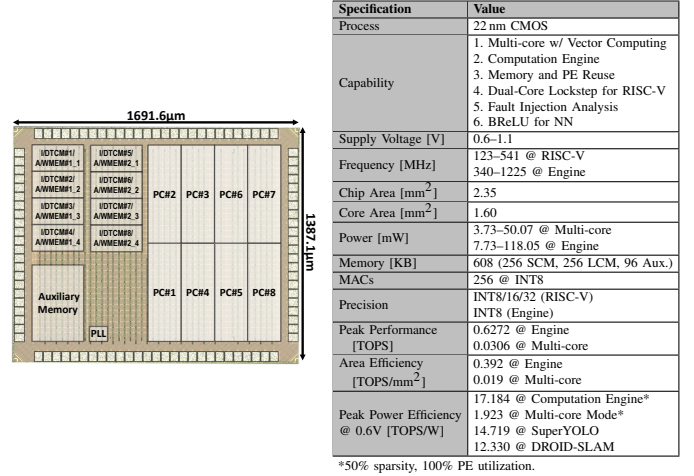


Fig. 16: Die micrograph and metrics.

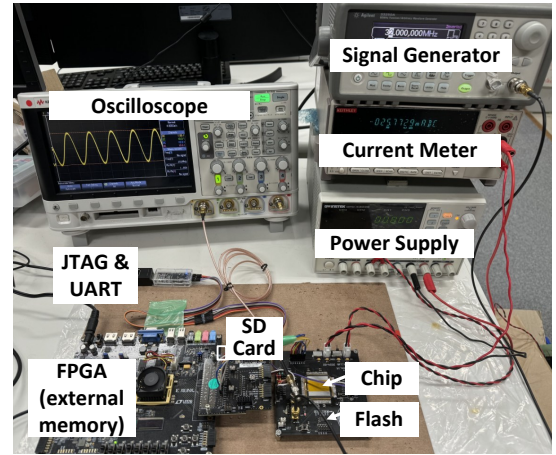


Fig. 17: Performance test platform.

sharing across modes. The SoC also integrates a PLL as a clock source and a JTAG-based controller for FI experiments. With this architecture, the system supports three primary modes (computation engine, multi-core, and fault injection) while reusing the same PEs and memory resources.

B. Performance Measurement Setup

The performance test platform is shown in Fig. 17. The UART module provides log outputs for real-time monitoring, while the JTAG interface is used for programming, debugging, and FI. The QSPI flash stores the compiled C programs for the RISC-V cores. The chip PCB is connected to an FPGA board, which emulates the external memory. The chip is powered by a DC supply providing 1.5–1.8 V for I/O and 0.6–1.1 V for the core. During operation, the RISC-V cores first boot from the QSPI flash to perform model decomposition, initialization, and memory allocation. The FPGA then loads model data from the SD card into the emulated external memory. Finally, PC#1 configures CSRs and orchestrates mode switching and task execution among computation engine, multi-core, and mixed modes for efficient system operation.

1) *Remote Sensing Image Processing using SuperYOLO for IED*: Remote sensing image processing requires flexibility

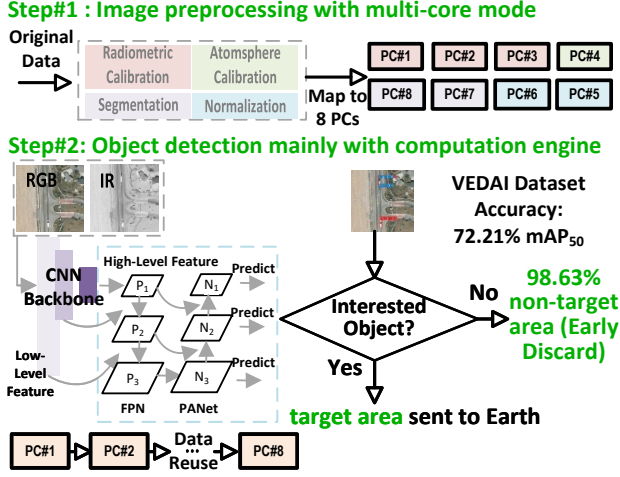


Fig. 18: Remote sensing image processing flow.

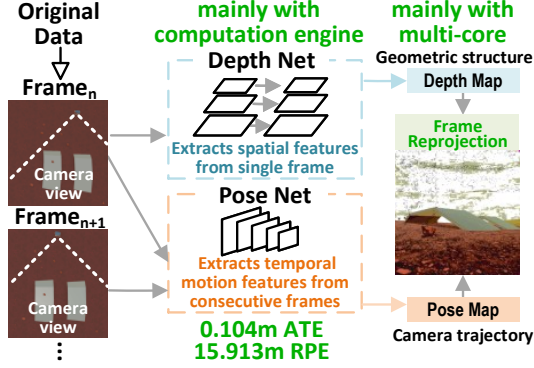


Fig. 19: Robot-assisted SLAM flow integrating depth estimation and motion tracking.

to handle diverse needs and ensure data quality through key pre-processing steps. As shown in Fig. 18, these include radiometric calibration to correct sensor imperfections, atmospheric correction to offset degradations [43], [44], resizing to a $3 \times 1024 \times 1024$ standard, and normalization for analysis. Multi-core processing with customized parallel computing is used for efficient parallel handling of tasks. Object detection is then performed in computation engine mode, streamlining data processing for remote sensing models like SuperYOLO [32]. SuperYOLO integrates RGB and IR inputs, with IR imagery providing critical data in adverse conditions to boost detection accuracy. The system detects target objects in the imagery and transmits the target area to Earth if detected. In experiments using the VEDAI dataset, our design's IED strategy reduces unnecessary (non-target area) data transmission by 98.63%.

2) Robot-Assisted SLAM with Real-Time Processing:

Robotic navigation using SLAM requires both high-throughput processing and reliable execution to sustain real-time performance in dynamic environments. As shown in Fig. 19, the system leverages multi-core mode with customized parallel computing to handle the pre-processing of incoming frames, including feature extraction, image normalization, and motion estimation. The computation engine mode is then employed for heavy matrix operations in the depth net and pose net of DROID-SLAM [33]. The depth net extracts spatial features

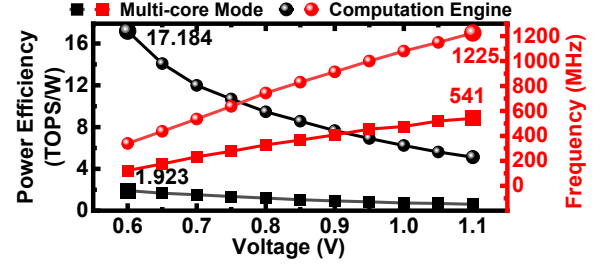


Fig. 20: Voltage scaling of the SoC prototype.

TABLE II: Latency comparison in multi-core mode.

Normalized Latency	w/ PE in Multi-core	w/o PE in Multi-core
	1.00	17.77

TABLE III: Area breakdown of the SoC prototype.

Component	Core Logic	Memory	FI Controller	Digital PLL
Percentage	*37.64%	61.92%	0.19%	0.25%

*Core logic area increases by 1.74% by replacing FFs with scan FFs.

from single frames to construct a depth map, while the pose net processes consecutive frames to capture temporal motion features for pose estimation. The results are fused into a global depth map and pose trajectory to guide navigation. This hybrid execution framework ensures efficient data reuse and real-time SLAM performance. Under this setup, the DROID-SLAM model achieves an absolute trajectory error (ATE) of 0.104 m and a relative pose error (RPE) of 15.913 m.

C. Performance Results

The voltage scaling results are shown in Fig. 20. The SoC operates across 0.6–1.1 V, with the RISC-V cores running at 123–541 MHz and the computation engine achieving 340–1225 MHz. The peak power efficiency reaches 17.184 TOPS/W in computation engine mode and 1.923 TOPS/W in multi-core mode at 0.6 V under 50% data sparsity, demonstrating the design's adaptability across both energy-constrained and high-performance scenarios. Furthermore, as shown in Table II, under peak performance in multi-core mode, the SoC achieves a $17.77\times$ reduction in latency compared to an identical system without PEs. This improvement arises because, without the proposed PEs, the RISC-V cores rely solely on native multipliers, where repetitive instruction execution and sequential operations result in much lower computational efficiency than the parallel PE-based architecture.

The area breakdown is presented in Table III. The memory system dominates the area cost with 61.92%, while the RISC-V cores and logic contribute 37.64%. The FI controller and digital PLL occupy only 0.19% and 0.25%, respectively. Notably, the core logic area increases by 1.74% due to replacing FFs with scan FFs in the DFT-based FI flow. Such scan insertion is a mandatory overhead in VLSI designs, as scan FFs are essential for production testing to ensure controllability and observability of sequential elements before chip shipment.

As summarized in Table IV, SuperYOLO achieves a peak energy efficiency of 14.72 TOPS/W at 0.6 V, with an average efficiency of 8.48 TOPS/W across engine and multi-core modes, while DROID-SLAM reaches 12.33 TOPS/W peak

TABLE IV: Performance evaluation of different benchmarks.

Benchmark	SuperYOLO	DROID-SLAM
Dataset	VEDAI	EuRoC MAV
Accuracy	72.21% mAP	0.104m ATE 15.913m RPE
Weight Sparsity	16.11%	24.63%
Activation Sparsity	69.31%	39.84%
Latency (Engine/Multi-core/Total) @ 1.1V [ms]	109.47/20.48/129.95	74.20/60.82/135.02
Average Power Efficiency (Engine/Multi-core/Total) @ 0.6V [TOPS/W]	9.82/1.33/8.48	8.73/1.28/5.37
Peak Power Efficiency in Models @ 0.6V [TOPS/W]	14.719	12.330
Memory Access Reduction ¹	27.8%	14.3%

* Assuming external memory bandwidth of 128 bits/cycle.
¹ Compared to an identical design w/ separate engine and multi-core.

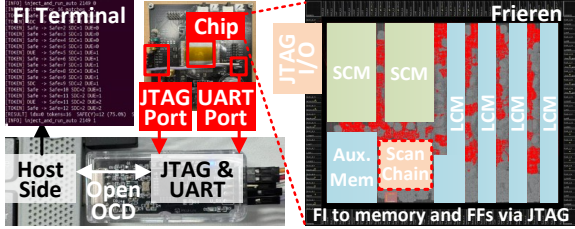


Fig. 21: Fault injection experimental setup.

and 5.37 TOPS/W on average. The difference mainly stems from workload characteristics. SuperYOLO is dominated by convolution operations with high weight reuse and regular data flow, leading to efficient utilization of on-chip buffers and PEs. In contrast, DROID-SLAM contains a larger portion of non-matrix operations (e.g., geometric estimation and feature association), which result in lower data reuse and more irregular memory accesses, reducing overall efficiency. In terms of latency, DROID-SLAM shows a longer multi-core latency (60.82 ms) than SuperYOLO (20.48 ms), reflecting the sequential dependencies and synchronization overhead in the SLAM pipeline. However, its computation-engine latency is lower (74.20 ms versus 109.47 ms) due to the smaller overall computational workload. Besides, both workloads benefit from the shared-memory architecture, which reduces redundant data transfers between multi-core and engine domains. Compared with an identical design using separate memory regions, memory accesses are reduced by 27.8% for SuperYOLO and 14.3% for DROID-SLAM. These results demonstrate that Frieren’s hybrid execution model effectively adapts to diverse workloads while improving throughput and energy efficiency.

V. RELIABILITY EVALUATION

A. Fault Injection Setup

For the FI setup, we implement a hardware-based test environment in which faults are directly injected into the 22 nm SoC chip through a physical JTAG interface (Fig. 17). The overall FI experimental setup is shown in Fig. 21. On the host side, OpenOCD is used to communicate with the chip via JTAG and UART ports, enabling configuration and control of the on-chip FI controller. During operation, the FI controller manipulates the SC to inject faults into FFs and accesses the on-chip memory bus to emulate SEU-induced errors in

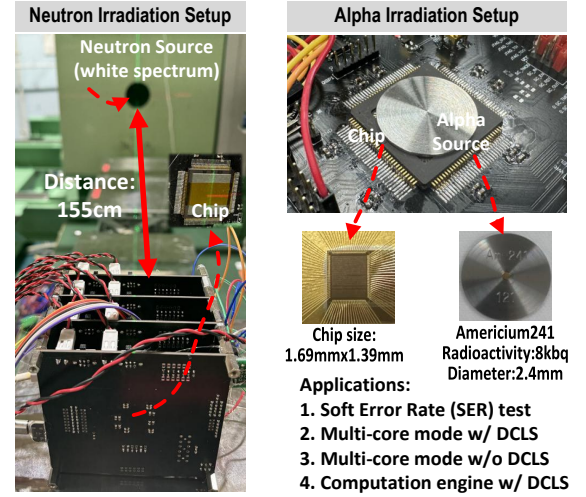


Fig. 22: Irradiation test of the SoC chip.

memory blocks. The FI efficiency for FFs is determined by the total SC length multiplied by the scan clock cycles, along with the JTAG configuration latency. For memory FI, the efficiency depends on the bus access cycles and the same JTAG configuration latency. This design enables fine-grained and reproducible FI across both sequential and memory elements at the hardware level. On the chip layout, the red regions indicate the distributed SC network supporting FF-level FI, while memory-level FI is realized via direct bus access.

B. Radiation Test Setup

Irradiation experiments (Fig. 22) are conducted using both neutron and alpha particle sources to evaluate the reliability of the SoC chip under radiation-induced soft errors. On the left, the neutron irradiation setup is shown, where the chip was placed 155 cm away from a white-spectrum neutron beam source at Research Center for Nuclear Physics (RCNP), Osaka University. This high-energy beam has an effect similar to that of protons in space [45]. Multiple chips are mounted on test boards to enable parallel testing. On the right, the alpha irradiation setup is presented, in which an Americium-241 (²⁴¹Am) alpha source (8 kBq, 2.4 mm diameter) is positioned directly above the chip (die size: 1.69 mm × 1.39 mm).

These complementary irradiation setups enable controlled radiation exposure for measuring the soft error rate (SER) and validating the effectiveness of DCLS in both multi-core and computation engine modes. Additionally, memory blocks and FFs are characterized separately as reliability references. The memory exhibited cross-sections of $4.07268\text{e-}15 \text{ cm}^2/\text{bit}$ under neutron exposure and $1.70871\text{e-}11 \text{ cm}^2/\text{bit}$ under alpha particle exposure, while the FFs showed comparable cross-sections of $3.95078\text{e-}15 \text{ cm}^2/\text{bit}$ and $1.6538\text{e-}11 \text{ cm}^2/\text{bit}$, respectively.

C. Reliability Analysis of Neural Network Workloads via FI

To evaluate the reliability of NN workloads under SEUs, FI experiments are performed by randomly flipping bits in the memory blocks associated with the linear layers of the models. In Fig. 23, both SuperYOLO and DROID-SLAM are tested

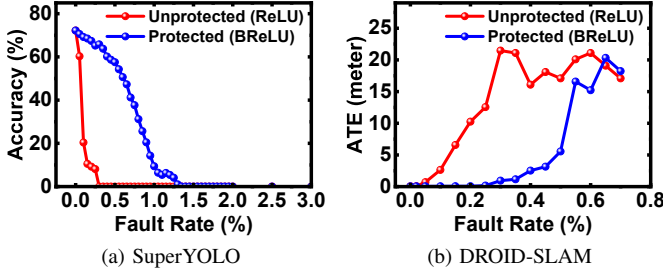


Fig. 23: Fault resilience under FIs in linear layers.

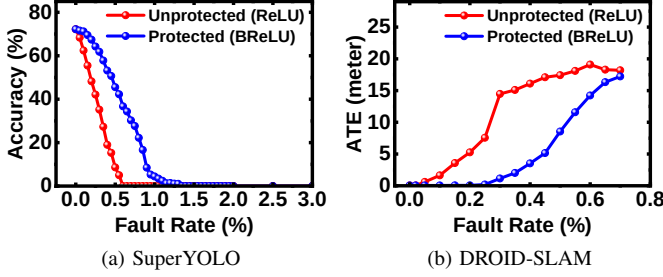


Fig. 24: Fault resilience under FIs in non-linear layers.

under increasing fault rates to observe variations in model behavior. The boundary-aware activation function (BReLU [38]) effectively suppresses fault-induced deviations by constraining activations within a bounded range. For SuperYOLO, BReLU maintains stable inference even when fault rates rise up to 3%, while for DROID-SLAM, the ATE remains significantly lower compared to the unprotected ReLU baseline.

We also conduct a separate FI experiment targeting the non-linear layers (Fig. 24). Although faults occurring in non-linear layers tend to cause larger output deviations, these layers constitute only a very small fraction of the overall model computation, accounting for merely 0.23% and 0.45% of the total computation in SuperYOLO and DROID-SLAM, respectively. As a result, their overall impact on system-level reliability is negligible. These results demonstrate that BReLU provides strong resilience against soft errors in data memories, ensuring reliable execution across different workloads.

D. Reliability Analysis of the SoC Prototype

To facilitate a systematic reliability analysis, we classify observed faults into four categories. A minor SDC event (mSDC) occurs when the final results remain correct despite unexpected intermediate outputs. A critical SDC event (cSDC) refers to cases where the final results are incorrect due to fault propagation. A minor DUE event (mDUE) is defined as a scenario where the computation engine fails to respond or temporarily malfunctions. Finally, a critical DUE event (cDUE) corresponds to severe failures where the RISC-V cores crash or become uncontrollable.

To better assess the reliability of each component, we split the SuperYOLO and DROID-SLAM into two parts: convolution/matrix, pooling, and activation functions run in the computation engine mode, while the remaining computations run in the multi-core mode. Detailed statistics for irradiation

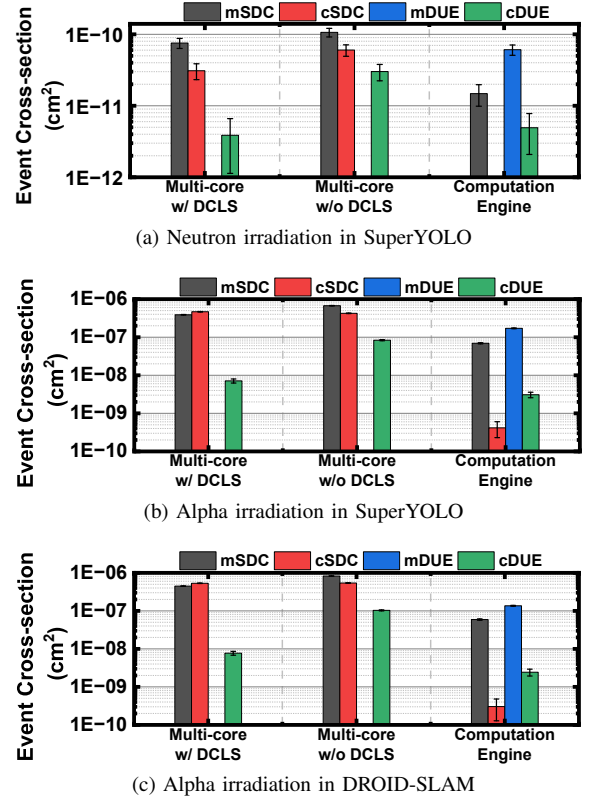


Fig. 25: Event cross-section comparison across different irradiation experiments and workloads.

experiments and application scenarios are summarized in Table V. It can be observed that these workloads exhibit consistent trends across neutron and alpha irradiation experiments, validating the repeatability of the observed fault behaviors.

However, due to the limited number of neutron-irradiated samples obtained in the current experiments, the statistical variance of the neutron results is relatively high. The neutron irradiation campaign is conducted over a 3-day experimental period. Achieving a sample size comparable to that of the alpha irradiation experiments would require approximately one year of continuous neutron testing, which is beyond the scope of our current experimental capability. Therefore, the subsequent quantitative analysis mainly relies on the alpha irradiation data, which provides more statistically meaningful results and more comprehensive fault coverage for evaluating the system's radiation-induced reliability characteristics.

To visualize the data more clearly, we convert the data into an event cross-section comparison chart (Fig. 25). Error bars represent $\pm 1\sigma$ standard deviations assuming Poisson statistics for event counts, and are larger for neutron data points due to the smaller number of observed events. Nevertheless, although the number of observed neutron-induced events is relatively limited, the overall trend and distribution of the events are consistent with those observed in the alpha irradiation experiments, indicating similar reliability behavior and therefore providing meaningful qualitative reference. Moreover, with DCLS protection, about 83% of errors in RISC-V cores are detected and corrected, reducing the cDUE cross-section by

TABLE V: Event statistics across different irradiation experiments and application scenarios.

Application	SuperYOLO (Neutron Experiment)					SuperYOLO (Alpha Experiment)					DROID-SLAM (Alpha Experiment)				
	Fluence (n/cm ²)	mSDC	cSDC	mDUE	cDUE	Fluence (n/cm ²)	mSDC	cSDC	mDUE	cDUE	Fluence (n/cm ²)	mSDC	cSDC	mDUE	cDUE
Multi-core w/ DCLS	5.168e+11	39	16	NA	2	9.273e+9	3596	4320	NA	66	9.731e+9	4360	5177	NA	75
Multi-core w/o DCLS	4.978e+11	53	30	NA	15	9.825e+9	6606	4168	NA	820	9.462e+9	7817	5114	NA	975
Computation Engine	6.080e+11	9	0	37	3	1.200e+10	831	5	2054	37	9.879e+9	585	3	1344	24
Total	-	101	46	37	20	-	11033	8493	2054	923	-	12762	10294	1344	1074

TABLE VI: Error distribution in RISC-V cores.

Application	Fetch	Decode	Execute
SuperYOLO	68.4%	4.7%	26.9%
DROID-SLAM	71.2%	5.1%	23.7%

TABLE VII: Projected Error distribution with ECC in ITCM.

Application	Fetch	Decode	Execute
SuperYOLO	36.9%	9.4%	53.7%
DROID-SLAM	38.4%	10.9%	50.7%

nearly an order of magnitude, significantly enhancing system reliability. Besides, as discussed in Section II-C, some errors remain undetected because certain data paths (e.g., the PE computing path) are not included in the DCLS coverage.

In addition, we analyze the distribution of errors in irradiation experiments based on DCLS checker data. In Table VI, the majority of errors originate from the Fetch stage, accounting for 68.4% in SuperYOLO and 71.2% in DROID-SLAM. This dominance indicates that instruction fetching and control-path operations are the most susceptible to transient faults, primarily due to frequent memory accesses and instruction buffering activities. In contrast, the Decode and Execute stages exhibit lower error rates of approximately 5% and 25%, respectively. On the other hand, unexpected bit flips in ITCM, which are the primary cause of Fetch-stage failures, can be mitigated through lightweight protection mechanisms such as periodic instruction-memory scrubbing, guided by the memory and event cross-section data obtained from irradiation experiments.

To evaluate the potential benefit of memory-level protection, we conduct an analysis assuming that ECC protection (Single Error Correction, Double Error Detection) is integrated into ITCM. Under this assumption, single-bit errors in ITCM can be corrected before propagating to the Fetch stage. After the analysis, the overall number of errors can be reduced by approximately 50% with ECC. Table VII presents the projected error distribution. Although the ratio of Fetch-stage errors is significantly reduced, a considerable portion of failures still originates from the fetch control logic and pipeline registers, which cannot be addressed by memory-level ECC alone. This observation further justifies the need for the proposed DCLS mechanism, which provides broader protection across the pipeline rather than only protecting ITCM. Furthermore, replacing all ITCM blocks with ECC-protected memories would introduce approximately 20% additional area overhead. Therefore, the use of ECC protection requires careful consideration of the associated hardware cost.

E. Comparison Between FI and Irradiation Experiments

To obtain a more realistic assessment of overall system reliability, the FI and alpha irradiation experiments are conducted

TABLE VIII: Event statistics of irradiation and FI experiments.

Test Case	Fluence (n/cm ²)	mSDC	cSDC	mDUE	cDUE
SuperYOLO (Alpha Experiment)	9.945e+9	1272	323	1897	97
SuperYOLO (FI Experiment)	1.000e+10	1555	295	2110	120
DROID-SLAM (Alpha Experiment)	9.653e+9	2507	1362	987	267
DROID-SLAM (FI Experiment)	1.000e+10	2300	1550	1140	310

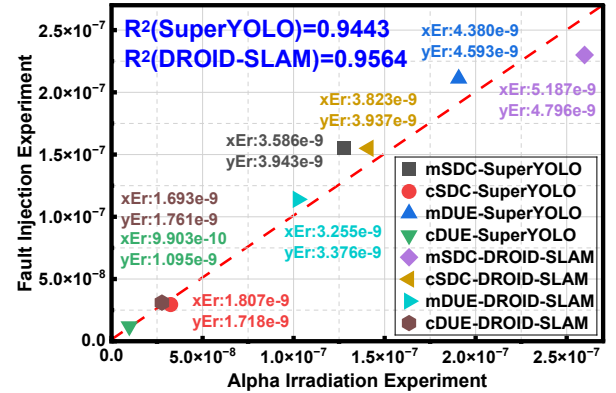


Fig. 26: FI cross-section vs. Alpha irradiation cross-section.

on complete application executions rather than separately evaluating the multi-core and computation engine modes as in the previous tests. This evaluation provides a holistic view of the system's fault response and reliability behavior under representative workloads, as summarized in Table VIII.

The results are illustrated as an event cross-section comparison in Fig. 26. The X-axis represents the cross-section of each event in the alpha radiation experiment, the Y-axis represents the event cross-section in the FI experiment, and xEr and yEr denote the $\pm 1\sigma$ Poisson-standard-deviation uncertainties of the X and Y data points, respectively. The closer the symbol is to the red dotted line, the higher the correlation. The correlation coefficients (R^2) reach 0.9443 for SuperYOLO and 0.9564 for DROID-SLAM, indicating that the FI framework accurately replicates the fault behavior observed under real irradiation. This high degree of agreement validates the fidelity of the FI methodology in modeling SEU effects at the system level. Moreover, the close correspondence between FI and irradiation results demonstrates that the FI framework provides a reliable and cost-effective alternative to physical radiation testing.

VI. COMPARISON WITH STATE-OF-THE-ART

Table IX summarizes the performance of the SoC in comparison with recent SOTA designs. Fabricated in 22 nm CMOS, the fabricated SoC integrates 16 RV32IM cores and a computation engine. Operating from 0.6 V to 1.1 V, the SoC demonstrates a wide dynamic range in both performance and

TABLE IX: Performance comparison with state-of-the-art designs.

Process	Versa [10] 28 nm	ISSCC23 [11] 22 nm FDX	ISSCC21 [16] 12 nm	Trikarenos [17] 28 nm	Frieren-based SoC 22 nm
Architecture	36× Cortex-M4F	1× RV32IMCFXpulp + 16× RV32IMCFXpulpnn + 1× RBE Accelerator	1× Cortex-R52 for control + 8× Cortex-A76 for application + 1× 3D GPU for application + 22 cores for image recognition	3× Ibex RISC-V	(1) 16× RV32IM cores w/ Vector Computing or (2) 2× RV32IM Cores + 1× Computation Engine
Supply [V]	0.55–1.0	0.5–0.8	0.8 (core), 0.85 (CPU, GPU)	0.9	0.6–1.1
Frequency	12–510 MHz	420 MHz	1.8 GHz Cortex-A76 1 GHz Safety Processor	250–270 MHz	123–541 MHz (RISC-V) 340–1225 MHz (Engine)
Area [mm ²]	12	18.7	230.46	2	2.35
Power [mW]	7.9–811.2	12.8–123	4400	2.4–19.7	3.73–50.07 (Multi-core) 7.73–118.05 (Engine)
Memory	1.27 MB	1152 KB	~41.6 MB	256 KB	608 KB
Precision	FP32	INT2/4/8/16/32 (RISC-V) INT2–8 (RBE)	INT8	-	INT8/16/32 (RISC-V) INT8 (PE)
Peak Performance [TOPS]	0.0119	0.637 @ RBE2×2	60.4	666 DMIPS	0.6272 (Engine) 0.0306 (Multi-core)
Peak Power Efficiency [TOPS/W]	0.0364	1.66 @ INT in RISC-V 12.4 @ RBE2×2 6.38@RBE Mixed ResNet20 5.83@RBE4×4 ResNet18	13.8 6.1 @ VGG16	0.0056	17.184 (Engine) 1.923 (Multi-core) 14.719 @ SuperYOLO 12.330 @ DROID-SLAM

TABLE X: Comparison with prior reliability-oriented architectures and techniques.

Category	Prior Art	Frieren
Architecture	Conventional architectures w/ fixed functional units (e.g., [16], [17], [20]) limited functionality	Reconfigurable architecture w/ full resource reuse across multiple modes high flexibility
Radiation-hardening technique	D/TCLS (e.g., [16], [17]) Hybrid redundancy (e.g., [20]) Timing error sensing (e.g., [1], [12]) high area/power overhead	Adaptive DCLS Boundary-aware ReLU adaptive configuration targeting different tasks
Reliability analysis technique	RTL/netlist FI (e.g., [39], [40]) Test-dedicated chip for FI (e.g., [41]) Cross-layer FI (e.g., [2], [15]) low efficiency & limited access & high overhead	JTAG-assisted integrated FI via SC(s) and bus access high efficiency & full on-chip access
Feature	Heavy data movement Few reusable components Limited parallel programmability (e.g., [16], [17], [20]) limited resource utilization	Zero data movement across different modes Full resource reuse Programmable parallel computing via custom inst. energy-efficient

power efficiency, with measured power consumption ranging from 3.73 mW to 118.05 mW across operating modes. In multi-core mode, the SoC achieves a 17.77× latency reduction in vector computing compared with a baseline RISC-V implementation. In computation engine mode, the peak power efficiency reaches 17.18 TOPS/W at 0.6 V and 340 MHz. During real-world workloads such as SuperYOLO and DROID-SLAM, the system delivers 14.719 and 12.330 TOPS/W, respectively. Compared to prior SoCs at similar process nodes, the SoC achieves 2.3–2.5× higher energy efficiency, primarily attributed to its optimized inter- and intra-PC dataflow design, which enables up to 16× data reuse across processing clusters while minimizing off-chip memory access.

Beyond performance comparison, Frieren introduces a different architectural paradigm compared with prior reliability-oriented designs, as summarized in Table X. While conventional solutions typically rely on fixed functional units and circuit-level hardening techniques, Frieren adopts a reconfigurable architecture that enables full resource reuse across multiple operating modes, significantly improving architectural flexibility and resource utilization while avoiding the high area and power overhead commonly associated with traditional redundancy-based techniques. In addition, Frieren

integrates adaptive DCLS and a JTAG-assisted FI mechanism that are not present in prior works. The adaptive DCLS dynamically enables fault-tolerant operation without significantly compromising energy efficiency. Meanwhile, the built-in JTAG-assisted FI approach facilitates pre-deployment reliability analysis by supporting on-chip FI and SEU analysis. By minimizing unnecessary data movement and enabling programmable parallel computing, Frieren achieves energy-efficient and highly flexible computing while providing robust reliability support for AI workloads in harsh environments.

VII. CONCLUSION

This work presents Frieren, a fault-tolerant and reconfigurable architecture for adaptive reliability in harsh environments, validated with a 22 nm SoC prototype. Frieren integrates three primary modes: a computation engine mode for high-throughput AI inference, a vector multi-core mode with adaptive DCLS functionality for reliable parallel execution, and a JTAG-based FI mode for pre-deployment reliability analysis. Unlike conventional designs, these modes fully share PEs and memory, minimizing data movement and ensuring efficient mode transitions. A dynamic resource allocation framework further adapts hardware utilization to mission requirements, achieving optimal performance under resource-constrained conditions. Experimental results of the 22 nm SoC demonstrate that DCLS can correct up to about 83% of RISC-V errors under radiation faults, while vector computing in multi-core mode achieves a 17.77× reduction in computing latency. In terms of energy efficiency, the SoC delivers up to 17.18 TOPS/W in computation engine mode and 1.92 TOPS/W in multi-core mode. In real workloads, it achieves peak efficiencies of 14.72 TOPS/W on SuperYOLO and 12.33 TOPS/W on DROID-SLAM. Moreover, the FI framework and neutron/alpha experiments validate the robustness of the design, enabling enhanced reliability for both in-orbit computing and fault-tolerant robotics applications.

ACKNOWLEDGMENTS

This work was supported by the Grant-in-Aid for Scientific Research (S) from Japan Society for the Promotion of Science (JSPS) under Grant 24H00073 and JP19H05664.

REFERENCES

- [1] H. -M. Pham, S. Pillement and S. J. Piestrak, "Low-overhead fault-tolerance technique for a dynamically reconfigurable softcore processor," in *IEEE Transactions on Computers*, June 2013.
- [2] C. Bolchini, L. Cassano, A. Miele and A. Toschi, "Fast and Accurate Error Simulation for CNNs Against Soft Errors," in *IEEE Transactions on Computers*, April 2023.
- [3] P. Rech, "Artificial Neural Networks for Space and Safety-Critical Applications: Reliability Issues and Potential Solutions," in *IEEE Transactions on Nuclear Science*, vol. 71, no. 4, pp. 377-404, April 2024.
- [4] J. Wang et al., "Enhancing Neural Network Reliability: Insights From Hardware/Software Collaboration With Neuron Vulnerability Quantization," in *IEEE Transactions on Computers*, Aug. 2024.
- [5] G. Furano et al., "Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities," in *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, pp. 44-56, 1 Dec. 2020.
- [6] R. Harboe-Sorensen et al., "Observation and analysis of Single Event Effects on-board the SOHO satellite," *RADECS 2001. 2001 6th European Conference on Radiation and Its Effects on Components and Systems (Cat. No.01TH8605)*, Grenoble, France, 2001, pp. 37-43.
- [7] V. Sze, Y. -H. Chen, T. -J. Yang and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," in *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295-2329, Dec. 2017.
- [8] J. Huang et al., "A High-Intensity Solution of Hardware Accelerator for Sparse and Redundant Computations in Semantic Segmentation Models," in *IEEE Transactions on Computers*, Sept. 2025.
- [9] A. Symons et al., "Stream: Design Space Exploration of Layer-Fused DNNs on Heterogeneous Dataflow Accelerators," in *IEEE Transactions on Computers*, Jan. 2025.
- [10] S. Kim et al., "Versa: A 36-Core Systolic Multiprocessor With Dynamically Reconfigurable Interconnect and Memory," in *IEEE Journal of Solid-State Circuits*, vol. 57, no. 4, pp. 986-998, April 2022.
- [11] F. Conti et al., "22.1 A 12.4TOPS/W @ 136GOPS AI-IoT System-on-Chip with 16 RISC-V, 2-to-8b Precision-Scalable DNN Acceleration and 30%-Boost Adaptive Body Biasing," *2023 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, 2023, pp. 21-23.
- [12] A. Jain, A. M. Veggetti, D. Crippa, A. Benfante, S. Gerardin and M. Bagatin, "Radiation Tolerant Multi-Bit Flip-Flop System With Embedded Timing Pre-Error Sensing," in *IEEE Journal of Solid-State Circuits*, vol. 57, no. 9, pp. 2878-2890, Sept. 2022.
- [13] K. R. Scott and S. P. Khatri, "A Flash-based Reconfigurable QCNN Inference Accelerator for Edge Applications," in *IEEE Transactions on Computers*.
- [14] X. Peng, R. Liu and S. Yu, "Optimizing Weight Mapping and Data Flow for Convolutional Neural Networks on RRAM Based Processing-In-Memory Architecture," *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, 2019, pp. 1-5.
- [15] A. Vallero et al., "SyRA: Early System Reliability Analysis for Cross-Layer Soft Errors Resilience in Memory Arrays of Microprocessor Systems," in *IEEE Transactions on Computers*, May 2019.
- [16] K. Matsubara et al., "4.2 A 12nm Autonomous-Driving Processor with 60.4TOPS, 13.8TOPS/W CNN Executed by Task-Separated ASIL D Control," *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2021, pp. 56-58.
- [17] M. Rogenmoser and L. Benini, "Trikarens: A Fault-Tolerant RISC-V-based Microcontroller for CubeSats in 28nm," *2023 30th IEEE International Conference on Electronics, Circuits and Systems, Istanbul, Turkiye*.
- [18] F. Libano et al., "Selective Hardening for Neural Networks in FPGAs," in *IEEE Transactions on Nuclear Science*.
- [19] W. Li et al., "Soft Error Mitigation for Deep Convolution Neural Network on FPGA Accelerators," *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Genova, Italy.
- [20] L. Ghionda et al., "HMR-NEureka: Hybrid Modular Redundancy DNN Acceleration in Heterogeneous RISC-V SoCs," *2025 IEEE Computer Society Annual Symposium on VLSI*, Kalamata, Greece, 2025.
- [21] T. Singh et al., "Intelligent FaultEdge: AI-Driven Fault-Tolerant Edge Framework for Smart Grid Monitoring in IoT," *2024 IEEE 12th International Conference on Intelligent Systems*, Varna, Bulgaria, 2024.
- [22] Y. Tang et al., "Perception and Navigation in Autonomous Systems in the Era of Learning: A Survey," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 9604-9624, Dec. 2023.
- [23] D. Liu et al., "A reliability-aware and energy-aware task scheduling algorithm for heterogeneous multi-core systems," *2024 36th Chinese Control and Decision Conference*, Xi'an, China, 2024, pp. 3212-3217.
- [24] B. Denby and B. Lucia, "Orbital Edge Computing: Machine Inference in Space," in *IEEE Computer Architecture Letters*, vol. 18, no. 1, pp. 59-62, 1 Jan.-June 2019.
- [25] Y. Wang, Y. Tian, J. Chen, K. Xu and X. Ding, "A Survey of Visual SLAM in Dynamic Environment: The Evolution From Geometric to Semantic Approaches," in *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1-21, 2024, Art no. 2523221.
- [26] K. Z. Ibrahim and L. Olikar, "Preprocessing Pipeline Optimization for Scientific Deep Learning Workloads," *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Lyon, France, 2022.
- [27] H.-I. Liu et al., "Lightweight deep learning for resource-constrained environments: A survey," *ACM Computing Surveys*, vol. 56, no. 10, pp. 1-42, Oct. 2024.
- [28] B. Seyoum et al., "Spatio-Temporal Optimization of Deep Neural Networks for Reconfigurable FPGA SoCs," in *IEEE Transactions on Computers*, Nov. 2021.
- [29] S. Zeng et al., "Serving Multi-DNN Workloads on FPGAs: A Coordinated Architecture, Scheduling, and Mapping Perspective," in *IEEE Transactions on Computers*, May 2023.
- [30] M. Huang et al., "A High Performance Multi-Bit-Width Booth Vector Systolic Accelerator for NAS Optimized Deep Learning Neural Networks," in *IEEE Transactions on Circuits and Systems I: Regular Papers*.
- [31] M. Huang et al., "EdgeLLM: A Highly Efficient CPU-FPGA Heterogeneous Edge Accelerator for Large Language Models," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 7, pp. 3352-3365, July 2025.
- [32] J. Zhang et al., "SuperYOLO: Super Resolution Assisted Object Detection in Multimodal Remote Sensing Imagery," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1-15, 2023.
- [33] Z. Teed and J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [34] Q. Cheng et al., "A 22nm Resource-Frugal Hyper-Heterogeneous Multi-Modal System-on-Chip Towards In-Orbit Computing," *2025 IEEE Custom Integrated Circuits Conference*, Boston, MA, USA, 2025, pp. 1-3.
- [35] D. Rath, "Open On-Chip Debugger," Diploma, Department of Computer Science, University of Applied Sciences Augsburg, 2005.
- [36] V. Camus et al., "Review and Benchmarking of Precision-Scalable Multiply-Accumulate Unit Architectures for Embedded Neural-Network Processing," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 697-711, Dec. 2019.
- [37] X. Xie et al., "Accelerating LLM Inference via Low-Bit Fine-Grained Quantization Algorithm and Bit-Level Accelerator Co-Design," in *IEEE Transactions on Computers*, Feb. 2026.
- [38] J. Zhan et al., "Improving Fault Tolerance for Reliable DNN Using Boundary-Aware Activation," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.
- [39] D. -A. Yang et al., "ACE-Pro: Reduction of Functional Errors with ACE Propagation Graph," *2021 IEEE International Test Conference (ITC)*, Anaheim, CA, USA, 2021, pp. 10-19.
- [40] Q. Cheng et al., "How accurately can soft error impact be estimated in black-box/white-box cases? -a case study with an edge AI SoC-," *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024.
- [41] Q. Cheng et al., "Genshin: A Generalized Framework with Software-Hardware Co-design and Pruned Fault Injection for Reliability Analysis," *2025 IEEE International Test Conference (ITC)*, San Diego, CA, USA, 2025, pp. 27-36.
- [42] H. Wang et al., "A configurable fault-tolerant glitch-free clock switching circuit," *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Columbus, OH, USA, 2013, pp. 537-540.
- [43] Z. Mao et al., "Radiometric Calibration Scheme for COCTS/HY-1C Based on Image Simulation From the Standard Remote-Sensing Reflectance," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1-9, 2022, Art no. 5402209.
- [44] M. Nazeer et al., "Evaluation of atmospheric correction methods for low to high resolutions satellite remote sensing data," *Atmospheric Research* 249 (2021): 105308.
- [45] J. Baggio et al., "Analysis of proton/neutron SEU sensitivity of commercial SRAMs-application to the terrestrial environment test method," in *IEEE Transactions on Nuclear Science*.



Quan Cheng (Member, IEEE) received the Ph.D. degree in 2024 and is currently a researcher with the Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University, Japan. His research interests include digital IC design, FPGA, embedded systems, reliability analysis, etc.



Qiufeng Li received his Bachelor degree from Xidian University and Master degree from Southern University of Science and Technology (SUSTech). His research interests include VLSI design and the application of large language models for electronic design automation (LLM4EDA).



Weirong Dong received the M.S. degree in Electrical and Computer Engineering from the University of California, San Diego, CA, USA, in 2024. He is currently pursuing the Ph.D. degree with the Graduate School of Informatics, Kyoto University, Kyoto, Japan. His research interests include in-memory computing, reliability of emerging hardware and efficient AI.



Mingtao Zhang received the B.S. degree from Sichuan University, China, and the M.S. degree from Waseda University, Japan. He is currently pursuing the Ph.D. degree with the Graduate School of Informatics, Kyoto University, Japan. His research interests include circuit reliability evaluation, soft error injection and detection in digital circuits, and approximate computing.



Ruilin Zhang (Member, IEEE) received the Ph.D. degrees from Waseda University, Japan, in 2017 and 2022, respectively. From 2023 to 2025, she served as a Program-Specified Assistant Professor with the Graduate School of Informatics, Kyoto University. Since June 2025, she has been an Associate Researcher with the School of Integrated Circuits, Southeast University. Her research interests include true random number generators, stochastic number generators, and hardware security.



Mingqiang Huang received the Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China in 2018. Then he was a research fellow in Nanyang Technological University (Singapore), focusing on energy efficient microelectronics and logic circuits. Since November 2019, he has been with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, as a research associate professor. His current research interests include in-memory computing, AI accelerator.



Yiyu Shi (Senior Member, IEEE) received the B.S. degree in Electronic Engineering from Tsinghua University in 2005, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of California, Los Angeles, in 2007 and 2009, respectively. He is a Professor in the Departments of Computer Science and Engineering at the University of Notre Dame, USA. His research interests include 3D integrated circuits, hardware security, and renewable energy applications. Prof. Shi has received multiple awards, including the NSF CAREER Award, Face-

book Research Award, and the IEEE Computer Society Mid-Career Research Achievement Award, among others.



conferences, including CICC, ASSCC, DAC, and ICCAD.



CAD for VLSI design and hardware accelerators for machine learning.



nanometer-scale LSI design, fabrication-aware design methodologies, and performance optimization under process variations.



the Google Cloud Research Innovators Award (2023). He currently chairs the IEEE SSCS Technical Committee on Open-Source Ecosystem (TC-OSE).



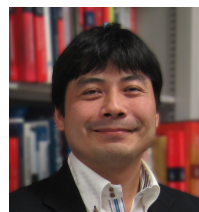
Hao Yu (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of California, Los Angeles, in 2007. He is currently with the Southern University of Science and Technology as founding Dean of Microelectronic School. His research interests include energy-efficient 3DIC and embodied AI chip. His honors include Best Paper Award of ACM-TODATE and IEEE-BIOCAS, and many other Best Paper Award nominations. Dr. Yu is a Senior Member of IEEE and served on technical program committees of major IEEE/ACM

Hiromitsu Awano (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in informatics from Kyoto University, Japan, in 2010, 2012, and 2016, respectively. He was with Hitachi, Ltd. in 2016, the VLSI Design and Education Center at the University of Tokyo from 2017 to 2018, and Osaka University from 2019 to 2020. From 2020 to 2026, he was an Associate Professor with the Graduate School of Informatics, Kyoto University. In April 2026, he joined Nagoya University, where he is currently a Professor. His research interests include

Takashi Sato (Senior Member, IEEE) received the B.E. and M.E. degrees from Waseda University, Tokyo, Japan, and the Ph.D. degree from Kyoto University, Japan. From 1998 to 1999, he was a Visiting Industrial Fellow at the University of California, Berkeley. He worked with Hitachi, Ltd. from 1991 to 2003, Renesas Technology Corporation from 2003 to 2006, and Tokyo Institute of Technology thereafter. In 2009, he joined the Graduate School of Informatics, Kyoto University, where he is currently a Professor. His research interests include CAD for

Mehdi Saligane received the Ph.D. degree in EE from the University of Aix-Marseille (IM2NP), in 2016. He is currently an Assistant Professor with the School of Engineering at Brown University and a Visiting Faculty member at Google Research. Prior to this, he was a Research Faculty member at the University of Michigan and a Research Engineer at STMicroelectronics, France. His research interests include energy-efficient IC design, biosensors, and AI-driven chip design. Dr. Saligane is the recipient of the Google Research Faculty Award (2021) and

Longyang Lin (M'16–SM'23) received the Ph.D. degree from the National University of Singapore, Singapore, in 2018, where he was a Research Fellow with the Department of Electrical and Computer Engineering from 2018 to 2021. He is currently an Assistant Professor with the School of Microelectronics, Southern University of Science and Technology, Shenzhen, China. His research interests include ultra-low-power circuit design, end-to-end compute-in-memory systems, cryogenic digital circuit design, and hardware security.



Masanori Hashimoto (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in communications and computer engineering from Kyoto University, Japan, in 1997, 1999, and 2001, respectively. He is a Professor with the Graduate School of Informatics, Kyoto University. His research interests include design for reliability, timing and power integrity, reconfigurable computing, and soft error. He has been a member of the technical program committees of major international conferences, including DAC, ICCAD, ITC, the Symposium on VLSI

Circuits, and DATE. He has served as Editor-in-Chief of Microelectronics Reliability (Elsevier) and as an Associate Editor for IEEE Transactions on VLSI Systems, and IEEE Transactions on Circuits and Systems I.