HachiFI: A Lightweight SoC Architecture-Independent Fault-Injection Framework for SEU Impact Evaluation

Quan Cheng[1,2], Wang Liao[3], Ruilin Zhang[1], Hao Yu[2], Longyang Lin[2], Masanori Hashimoto[1,*]

[1]Department of Communications and Computer Engineering, Kyoto University, Kyoto, Japan
[2]School of Microelectronics, Southern University of Science and Technology, Shenzhen, China
[3]School of Systems Engineering, Kochi University of Technology, Kochi, Japan
[*]{hashimoto@i.kyoto-u.ac.jp}

***Abstract*—Single-Event Upsets (SEUs), triggered by energetic particles, manifest as unexpected bit-flips in memory cells or registers, potentially causing significant anomalies in electronic devices. Driven by the needs of safety-critical applications, it is crucial to evaluate the reliability of these electronic devices before they are deployed. However, traditional reliability analysis techniques, such as irradiation experiments, are costly, while fault injection (FI) simulations often fail to provide full coverage and have limited effectiveness and accuracy. To address these issues, we introduce HachiFI, a lightweight, architecture-independent framework that automates fault injection with 100% coverage via memory and scan-chain accesses and simulates the behavior of SEUs based on specific cross-sections. HachiFI supports configurable fault injection patterns for both system-level and module-level reliability analysis. Using HachiFI, we demonstrate a low hardware overhead (<2%) and a high match (R^2=0.984) between FI and irradiation experiments, verified on a 22nm edge-AI chip.**

***Index Terms*—single-event upset, architecture-independent, fault injection, irradiation**

I. Introduction

In recent years, reliability-demanding electronic devices have grown significantly, particularly in safety-critical applications such as aerospace, automotive, medical devices, and industrial automation [1]. These devices are often exposed to harsh conditions, including radiation from energetic particles that can cause severe disruptions to electronic systems. One of the most common and challenging effects induced by radiation is Single-Event Upsets (SEUs) [2], [3]. SEUs occur when high-energy particles (e.g., protons, neutrons, or heavy ions) strike sensitive regions of semiconductor devices, causing unexpected bit-flips in memory cells, registers, or other critical components. These bit-flips, while transient, may lead to significant anomalies, including functional errors and system crashes, which in turn may have catastrophic consequences, such as navigation errors in spacecraft, malfunctioning medical devices, or failure of autonomous vehicle control systems [4]–[6]. Therefore, evaluating the reliability of electronic devices against SEUs before their deployment is crucial, particularly for safety-critical applications. To ensure the robustness of these systems, comprehensive reliability assessments must be performed during the design and testing phases.

Traditionally, reliability analysis of electronic devices has been conducted using irradiation experiments and fault injection (FI) experiments [7]–[11]. Irradiation experiments involve exposing the device under test (DUT) to a radiation source, such as particle accelerators or radioactive isotopes, to induce SEUs and observe the behavior of the system. Although irradiation experiments provide valuable insight into system's vulnerability, these experiments are typically costly, require specialized facilities, and their availability is highly limited. Furthermore, the turnaround time from the plan to the experiment takes several months, making instant or iterative testing infeasible. On the other hand, FI technique provides a more accessible and cost-effective alternative for SEU analysis by artificially injecting faults into the system to mimic SEUs. FI technique allows users to model and analyze the system's response to various fault scenarios, offering a flexible approach for reliability analysis. However, FI technique usually cannot achieve full coverage or accurate FI of all potential SEU scenarios due to vast state space and complex interactions within electronic systems. Besides, FI technique is often constrained by the assumptions and simplifications made in the fault models, which can lead to an incomplete or inaccurate representation of real-world SEU effects. Additionally, the effectiveness of FI simulations heavily depends on the quality and granularity of FI patterns, which can vary widely between different studies.

To address the limitations of traditional reliability analysis techniques, we introduce HachiFI, a Handy, ArCHitecture-Independent FI framework designed to enhance SEU and hardware security analysis, where the SEU analysis is focused in this paper. HachiFI aims to provide a comprehensive solution by automating the FI process, offering full coverage of potential SEU scenarios, and accurately simulating SEU behavior based on specific cross-sections. Unlike traditional FI tools, which are often constrained to specific architectures and require extensive manual configuration [12], HachiFI is designed to be easily adaptable to a wide range of hardware platforms. It features a unified FI and analysis interface through a customized OpenOCD [13] that accesses on-chip memories, processor register files, as well as scan-chains inside the processor and peripherals, achieving a full coverage of storage components on a chip. Furthermore, HachiFI allows for user-defined error patterns using TCL scripting, enabling automated FI across multiple abstraction levels—from system-level down to module-level analysis. This hierarchical approach lets users configure FI patterns tailored to their system's specific requirements, facilitating detailed reliability assessments that account for the unique characteristics of each component.

978-3-9826741-0-0/DATE25/© 2025 EDAA

To validate the applicability of HachiFI, an edge-AI processor fabricated with a 22nm technology is adopted as a case study. By using experimental results from alpha irradiation on our processor as a golden reference, this work examines how accurately and effectively HachiFI can estimate system reliability solely through FI, without the need for irradiation experiments. Specifically, faults are injected into the entire gate-level netlist in the simulation case, while in the hardware case, faults are injected into the actual chip under real conditions. The experimental results demonstrate a high correlation between HachiFI's FI results and the measured effects of irradiation, accurately replicating the impact of SEUs observed in real-world testing. The key contributions of this paper include:

- **HachiFI Framework for Fault Injection**: This paper introduces the HachiFI framework, a flexible and adaptable FI tool designed to perform FI at multiple levels of abstraction, from system-level to module-level, using TCL-based error patterns. HachiFI supports software-based (simulation) and hardware-based (emulation and fabricated chip) environments, providing comprehensive error patterns to evaluate the reliability of hardware platforms without the need for costly irradiation experiments. Its optimized integration with OpenOCD and scan-chains through custom commands further enhances its capabilities, making it suitable for a wide range of hardware platforms.
- **Validation with a 22nm Edge-AI Processor**: The effectiveness of the HachiFI framework is validated using a 22nm edge-AI processor, specifically designed for energy-efficient data processing in harsh environments. By combining FI analyses and irradiation measurements, this study provides a comprehensive evaluation of the processor's reliability and resilience against SEU effects. Experimental results demonstrate a high correlation between FI results performed by HachiFI and actual irradiation data, confirming HachiFI's accuracy in replicating SEU effects.

II. Related Work

Existing methods for analyzing and improving system resilience against SEUs can be broadly classified into two main categories: irradiation experiments and fault injection techniques. Each has its own advantages and limitations, shaping their applicability for modern electronic systems.

A. Irradiation Experiment

Irradiation experiments are considered gold standards for evaluating the impact of SEUs on electronic devices. The key advantage is their ability to directly observe the physical impact of energetic particles on real hardware, yielding highly accurate data on device vulnerabilities and failure modes.

However, irradiation experiments have significant drawbacks. The high cost of access to specialized irradiation facilities, along with the need for extensive safety protocols and complex experimental setups, makes these tests expensive and time-consuming [14]. J. Gava et al. conducted a detailed analysis of the Zynq-7000 device under neutron irradiation [10]. In the irradiation experiment (fluence: 10^{10} neutrons/cm^2), approximately 300 events were observed. Similarly, Cheng et al. analyzed their custom SoC implemented on a flash-based FPGA under neutron irradiation [11]. During a 4-day irradiation experiment (fluence: $2{\times}10^8$neutrons/cm^2), fewer than 300 events were recorded. Due to the high cost and the limited number of data points, the reliability analysis might be prone to bias, potentially leading to less accurate conclusions about the system's robustness. Additionally, irradiation experiments are often limited by the availability of specific particle sources and energy levels, which may not fully represent the diverse radiation environments encountered in real-world applications [15]. Besides, irradiation experiments are often scheduled several months in advance, causing significant delays. These constraints limit the practicality of irradiation experiments, especially during the early stages of design when frequent testing and adjustments are needed.

B. Fault Injection Technique

FI techniques provide a more accessible and flexible alternative to irradiation experiments by simulating SEU effects in a controlled simulation environment. FI techniques allow researchers to systematically explore the system's response to various fault scenarios, assess its fault tolerance mechanisms, and identify potential weaknesses. Despite their advantages, traditional FI simulation techniques have notable limitations. One major issue is the lack of full coverage in FI scenarios. TensorFI [12] is a high-level FI framework for TensorFlow applications, designed to inject both hardware and software faults. Its main limitation is the inability to directly inject faults into low-level hardware, restricting its use for hardware-specific testing. Similarly, Fiji-FIN [16] is a FI framework for evaluating deep learning models on IoT devices. Its primary drawback is that it can only inject faults into memory cells, not registers, which limits its coverage for testing hardware fault resilience.

Furthermore, current FI techniques face significant challenges in precisely targeting specific time points or particular system operating cycles for FI. A FI technique that injects errors into the FPGA's configuration memory limits its ability to perform fine-grained fault analysis, potentially oversimplifying the process by failing to capture dynamic or complex fault scenarios [17]. Also, Zhang et al. proposed an SEU FI framework based on UART port [18]. However, using UART to interact with the DUT during FI can interfere with the original execution of the DUT's program. Additionally, UART cannot support instruction-level FI, limiting its accuracy for precise analysis. Meanwhile, these existing tools remain tied to specific platforms, limiting their applicability across different systems.

These limitations lead to substantial inaccuracies in reliability analysis, as the injected faults may not accurately reflect real-world conditions or device behavior. The discrepancies between FI results and real system behaviors under radiation can undermine the reliability of FI frameworks for fault analysis.

III. HachiFI Framework

The HachiFI framework offers a comprehensive solution for evaluating the reliability of electronic systems against SEUs. It integrates a small circuitry responsible for FI capabilities directly into Hardware Description Language (HDL) designs, enabling both hardware-based and software-based FI. HachiFI

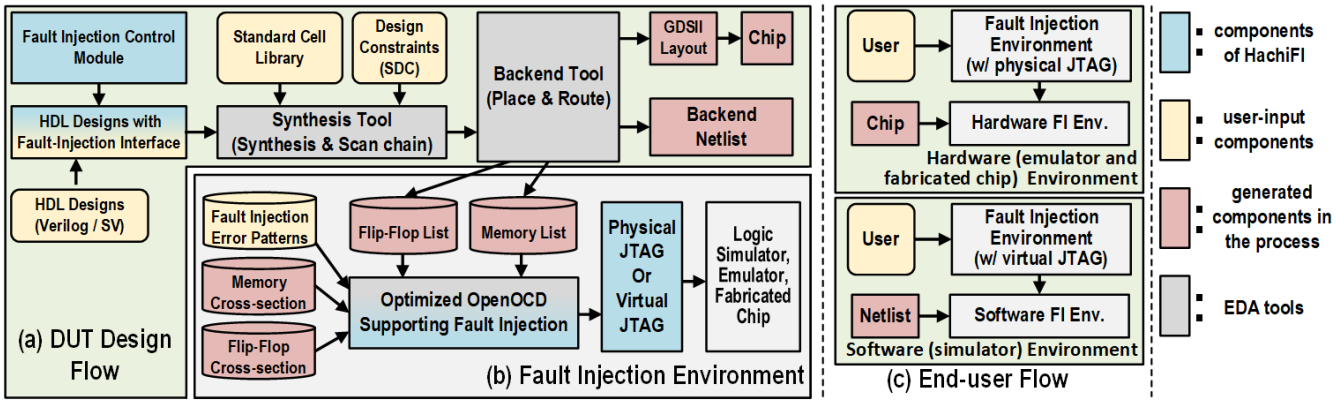


Fig. 1. HachiFI design flow. (a) DUT design flow. (b) Fault injection environment. (c) End-user flow.

is designed to seamlessly work with existing Application-Specific Integrated Circuit (ASIC) design flows, providing full coverage and configurability for reliability analysis.

The HachiFI design flow (Fig. 1) consists of the DUT design flow, FI environment, and end-user flow, each of which will be explained in the following subsections. The blue blocks represent HachiFI components, such as the FI control module and error patterns. The yellow blocks are user-provided inputs, like the standard cell library and design constraints (SDC), which guide synthesis and place-and-route. The pink blocks are generated or inherent components during the design process, including netlists, GDSII layouts, and memory/flip-flop (FF) lists and cross-section data. The gray blocks indicate EDA tools, such as synthesis, backend, and debug tools, which convert HDL designs into physical chips and perform debugging.

A. DUT Design Flow

A DUT design in the HachiFI framework starts with standard HDL designs (VerilogHDL or SystemVerilog) that are enhanced with a FI interface by incorporating FI control module, as shown in Fig. 1(a). Based on system bus interface and abstract commands in OpenOCD, the FI control module manages FI and interacts with designs to introduce controlled SEU simulations. This FI control module includes a logic that can efficiently schedule the scan shift and a read/write (R/W) module for reading and writing on-chip memory. The design is synthesized using standard EDA tools (e.g., Design Compiler) that can build scan-chains and incorporate FI logic, and then processed through backend tools for place-and-route, and generating the netlist for simulation and the GDSII file for chip fabrication.

The scan-control logic circulates the bit sequence in the scan-chain by connecting the scan-out to scan-in, and at the specified bit location, modifies the value and injects it to the scan-chain. During the integration process, the scan-chain control module can automatically adapt to the HDL design, while the memory reading and writing module requires the user to expand the memory R/W interface to interact with the module. Notably, memory access interfaces are commonly available in most processors to support debugger access.

B. Fault Injection Environment

The FI environment in Fig. 1(b) utilizes memory and FF cross-section data, user-defined error patterns, and memory and FF lists to simulate SEUs in memory and FFs, where the user-defined error patterns will be exemplified in Section III-D. The cross-section data, representing the probability of a radiation particle causing an upset, is crucial for assessing semiconductor reliability. This data, tied to the chip fabrication process, is constant in specific radiation environments and is used to model bit-flips during FI, where it could be provided by foundries, literature, or irradiation experiments. The memory and FF lists, detailing all units available for FI, are referenced during these simulations. Our optimized OpenOCD leverages the cross-section data, along with the memory and FF lists, to enable precise FI via the JTAG interface. In the extended OpenOCD setup for scan-chain operations, new commands that we implement for HachiFI (e.g., scRst for scan-chain reset, scRWb for BYTE-level R/W of scan-chain, scRWw for WORD-level R/W, and scRWd for DWORD-level R/W) allow fine control over the scan-chain. Note that additional new commands can be easily added to OpenOCD, if necessary. These commands facilitate accurate FI and debugging by directly manipulating the scan-chain by enabling the scan-enable signal and switching the system clock to scan-chain clock.

A key advantage of HachiFI is its precise control over system execution through JTAG. Commands such as halt and breakpoint can stop the system at a specific instruction for FI, which is especially useful for identifying vulnerabilities in buses and interfaces between processors and peripherals. Additionally, Hachi's scan-chain control and a timer built into OpenOCD allow comprehensive FI coverage, even when the processor is idle or polling.

C. End-User Flow for Simulation, Emulation and Chip

The HachiFI framework supports end-users in both software-based and hardware-based environments, as shown in Fig. 1(c). In hardware settings, users can employ physical JTAG connections to interact with the fabricated chip, injecting faults and analyzing its performance in a real-world hardware environment. Alternatively, virtual JTAG connections enable FI at the simulation level, allowing users to test the netlist before hardware is available. This simulation process interacts with external tools or environments, enabling the design to be tested and refined across multiple runs before reaching the physical chip stage. Besides, the netlist can also be imported onto

an FPGA for FPGA-based prototyping and FI testing from a physical view. This dual approach ensures flexibility and broad applicability of the framework across various stages of the design and validation process.

D. Error Pattern Definition and Accurate Execution

HachiFI significantly enhances FI processes by introducing a customizable, architecture-independent platform for more precise FI. Leveraging TCL scripts within OpenOCD, HachiFI allows users to define several error patterns:

1) Spatially and Temporally Random (STR) Pattern: STR FI allows for global FI simulation that closely reproduces real-world irradiation experiments. This pattern, supporting the access of all FFs and memories, enables the reproduction of chip behavior under actual radiation conditions.

2) Biased Random (BR) Pattern: However, when dealing with DUTs that integrate protection mechanisms like Error-Correcting Code (ECC) or Triple Modular Redundancy (TMR), STR FI may not yield accurate or efficient reliability insights, as these mechanisms significantly lower the error rate. Continuing to use STR FI on such protected systems may fail to expose vulnerabilities effectively. To overcome this, BR FI pattern is introduced. This pattern enables HachiFI to target meaningful faults by injecting Multi-Bit Upsets (MBUs) into protected regions considering spatial proximity, bypassing irrelevant Single-Bit Upsets (SBUs). This targeted approach ensures that reliability analysis focuses on critical fault scenarios, thereby improving the efficiency and relevance of FI testing.

3) Cross-Domain (CD) Pattern: HachiFI also facilitates the analysis of faults propagating within different modules across various functional domains of the chip, including memory, peripherals, and I/Os. To realize this, a CD FI error pattern is introduced for detailed tracking of how faults in one domain or module can impact the system, highlighting potential vulnerabilities and improving fault tolerance in large-scale designs.

4) Timing-based Attack (TA) Pattern: This pattern specifically targets the security integrity of subsystems or modules by manipulating clock signals and injecting faults to simulate timing attacks. By adjusting the clock or causing intentional delays, this pattern recreates the conditions under which timing-based security exploits occur, such as leaking sensitive information through clock glitches or disrupting secure processes.

In summary, the HachiFI framework introduces four key FI patterns to enhance the accuracy and relevance of FI testing across a wide range of scenarios. The FI flow using optimized OpenOCD is illustrated in Flow 1. These customizable patterns significantly improve FI precision, targeting both reliability and security concerns in complex systems.

Flow 1 FI Using Optimized OpenOCD with Pattern Support

```
Source all memory and FF information
Source memory and FF cross-section information
Record start
Repeat
  Select fault injection pattern:
    STR Pattern:
      Generate random spatial and temporal locations
      Halt system
      Inject errors at generated locations
      Resume system
    BR Pattern:
      Identify protected modules with ECC or TMR
      Generate multi-bit upset patterns
      Halt system
      Inject MBUs into protected regions
      Resume system
    CD Pattern:
      Select fault source in one domain/module (mem., I/O, etc.)
      Track fault propagation to other domains
      Halt system
      Inject faults and record propagation
      Resume system
    TA Pattern:
      Manipulate clock periods or introduce delays
      Halt system
      Inject timing faults into subsystems or modules
      Resume system
Until (system crashes or test is complete)
Record end
```

IV. An Edge-AI Platform for HachiFI Validation

A. Edge-AI Chip Framework

As shown in Fig. 2, our DUT of a 22nm edge-AI processor is composed of four main components: 1) an RV32IM RISC-V processor with a 3-stage pipeline, 64KB Instruction Memory, 32KB Data Memory, and peripherals such as UART, SPI, and JTAG; 2) a computation engine designed for convolution and matrix operations, featuring 16 Computation Cores (CCs),

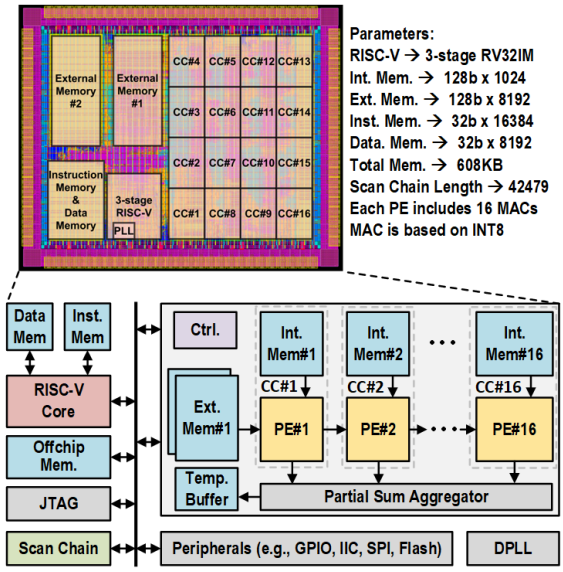


Fig. 2. Edge-AI chip framework.

256KB internal memories, and 256KB external memories; 3) a single scan-chain for supporting R/W operations for all FFs; and 4) a Digital Phase-Locked Loop (DPLL) generating clock outputs from 100 MHz to 1.2 GHz. Besides, our chip has 42,478 FFs. Specifically, the engine operates at a standard frequency of 600 MHz at 0.9V, while the RISC-V core and the scan-chain run at 100 MHz. In addition, protection mechanisms (e.g., ECC, TMR) are not applied to any SRAMs.

Our design integrates a high-performance computation engine with efficient arithmetic units and enhanced data reuse, using an INT8 multiplier as the basic computation unit. To

TABLE I
OVERHEAD ANALYSIS OF HACHIFI INTEGRATION ON THE EDGE-AI CHIP.

Component	Area(w/o HachiFI)	Area(w/ HachiFI)	Overhead
Core Logic	0.5642 mm^2	0.5910 mm^2	1.7398%
Memory	0.9723 mm^2	0.9723 mm^2	-
FI Controller	N/A	0.0030 mm^2	0.1948%
Digital PLL	0.0039 mm^2	0.0039 mm^2	-
Total	**1.5404 mm^2**	**1.5702 mm^2**	**1.9346%**
Decap cells, tap cells, endcap cells, and filler cells are not included.			

enhance computational parallelism, each CC features 16 multipliers, enabling multiply-accumulate (MAC) operations on 32 INT8 inputs to generate 1 output per data point. In each CC, the computation units are directly connected to the internal memory. This design accommodates three data flows: 1) pipeline data flow for external memory, 2) near-memory data flow for internal memory, and 3) stationary data flow for outputs. Each CC can process 32 bytes per clock cycle. Moreover, to achieve high data reuse, each internal memory updates data every 1-to-16 cycles, with all CCs alternating data updates. This means that each data can be reused 1-to-16 times depending on the configuration. Besides, it is notable that not only memory sizes but also R/W patterns affect the soft error rate.

B. Hardware Overhead Analysis of HachiFI Integration

To assess the impact of integrating the HachiFI framework on hardware resources, we conduct a detailed area overhead analysis. The evaluation focuses on the additional logic and resources required for the FI control module, scan-chain modifications, and error detection logic introduced by HachiFI. The analysis is performed on the 22nm edge-AI chip as mentioned in Section IV-A, comparing the original design without HachiFI to the modified design with FI capabilities.

The results, summarized in Table I, indicate that the integration of HachiFI incurs minimal area overhead. Specifically, the additional components required by HachiFI occupied less than 2% of the total chip area. This overhead primarily results from the area difference between normal FFs and scan-FFs. When the original design includes scan-FFs, which is common in industrial designs for shipping tests, the overhead for HachiFI is negligible. Also, the minimal area overhead observed during the integration of HachiFI is largely due to the simplicity of the scan-control logic. In synthesis, building the scan-chain only requires replacing normal FFs with scan-FFs. As FF cells are very small and relatively few in number, the size impact is minimal. Additionally, scan-FF cells only incur a 28% increase in area compared to normal FF cells, making the replacement cost-efficient. Moreover, as RISC-V natively supports R/W operations to on-chip memory, the integration of the FI system with memory access pathway requires minimal effort.

Additionally, the OpenOCD interface can be extended to interact with the R/W bus, ensuring that the memory-related overhead remains negligible as well. This minor increase in area ensures that the core performance and functionality of the chip are preserved while enabling comprehensive FI capabilities. The low overhead is a key advantage of HachiFI, making it suitable for integration into existing designs without significant impact on cost or power efficiency.

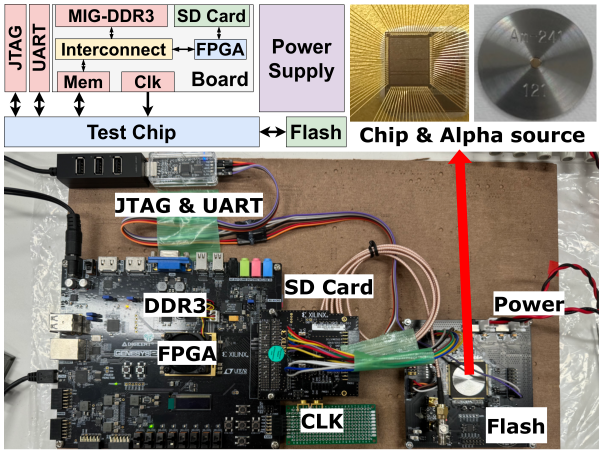


Fig. 3. Experimental setup for alpha irradiation and hardware fault injection.

V. EXPERIMENT AND EVALUATION

A. Experimental Setup

To validate the effectiveness of the HachiFI framework, we employ a three-pronged approach consisting of software-based FI via simulation tool, hardware-based FI via chip, and alpha irradiation experiments. The processor runs the ViT-Tiny network under ImageNet dataset for both alpha irradiation experiment and FI with STR error pattern. The number of cycles to execute one inference task is 13.94 million.

Firstly, for the software-based FI, we utilize a simulation environment based on Synopsys VCS that replicates SEU-like bit-flip errors in registers and memory blocks within a virtualized model of the 22nm edge-AI chip. The simulation environment allows us to configure various FI patterns, providing comprehensive coverage of possible fault scenarios without the need for physical hardware.

Secondly, for hardware-based FI, we implement a hardware-based environment where faults are directly injected into the 22nm edge-AI chip using physical JTAG interface. This setup enables real-time testing of the chip's response to injected errors, replicating SEU conditions at a hardware level. The FI setup in simulation and hardware is driven by the HachiFI framework's FI control module, allowing precise error pattern control and cross-section analysis of FFs and memory cells.

Lastly, we conduct alpha irradiation experiments to analyze the impact of energetic particles on the chip as shown in Fig. 3. We perform an irradiation experiment using an 8kBq Am-241 alpha source with a 2.4mm diameter. The chip is exposed to the alpha source, where the HachiFI framework's error detection and response are assessed under real-world SEU conditions. The cross-section data (memory: 1.7087e-11, FF: 1.6538e-11) are measured and used as reference data in software and hardware environments. Note that such cross-section data of basic storage elements could be provided from a foundry to industry. Based on the cross-section data and the volume of on-chip memories and FFs, the FI throughput for memory is 12.674 errors per second, while for FFs, it is 0.124 errors per second. The FI throughput for FF is determined by the scan-chain length, and can be improved by increasing the number of parallel scan-chains, similar to production tests before the

chip shipment. In this DUT, the necessary cycles per fault is 44,697 for FFs and 14 for memory. It should be noted that the number of FFs (42,479) is 117x smaller than the number of bits in SRAMs (4,980,736), indicating the lower FI throughput of FFs could be tolerable. The irradiation setup provides a benchmark for comparing the results from both software-based and hardware-based FI, enabling a holistic evaluation of HachiFI's capability to model SEU effects accurately. The combination of these three experimental setups provides a robust validation of the HachiFI's reliability and effectiveness in FI scenarios.

During the execution of applications, not all errors lead to critical consequences. We categorize errors into four classes, based on the severity of their impact: 1) minor Silent Data Corruption (mSDC), where the classification outcome for an image remains correct despite unexpected intermediary outputs; 2) critical SDC (cSDC), where there is an incorrect classification result; 3) minor Detectable Unrecoverable Error (mDUE), the computation engine fails to respond or malfunctions but it can be restarted without power cycling; and 4) critical DUE (cDUE), which involves the RISC-V core either running out of control or crashing and requires the system rebooting.

B. Experimental Results

In the results of the experiments, we observe some differences in the efficiency and accuracy of FI methods between the software FI, hardware FI, and alpha irradiation experiments, highlighting the strengths of the HachiFI framework. Firstly, the software-based FI demonstrates significantly lower efficiency due to the computational burden of simulating SEU-induced faults entirely in software, performing about five orders of magnitude slower than the hardware FI. This emphasizes the limitations of traditional software-based FI methods, particularly when dealing with complex, large-scale fault scenarios. In contrast, the hardware FI environment, which directly injects faults into the fabricated 22nm edge-AI chip, demonstrates efficiency levels closely matching those observed in the alpha irradiation experiments. This close alignment in performance is due to the hardware FI's efficiency in replicating real-world SEU conditions, without the heavy time consumption typically seen in software-based simulations.

1) Efficiency Analysis of FI: The efficiency of FI methods varies significantly between software-based FI and hardware-based FI, highlighting the strengths and limitations of each approach. Software-based FI, conducted entirely in software, is inherently 400,000x slower (running on EPYC 7502P) due to the high computational burden of mimicking SEU-induced faults at the bit level. This process requires extensive computational resources and time, especially when dealing with complex systems, resulting in prolonged simulation times and limiting its practicality for large-scale fault coverage.

In contrast, hardware-based FI for chip offers a dramatic improvement in efficiency. By directly injecting faults into the hardware, FI operates in real time, effectively replicating the SEU effects without the computational delays typical of software simulations. Our experiments demonstrate that the hardware FI approach achieves performance levels comparable to physical alpha irradiation tests, with only a 0.03% additional time overhead for system FI, providing near-instantaneous feedback on fault impact and system's response. This high efficiency makes hardware FI a highly suitable method for validating the reliability of safety-critical systems, bridging the gap between software simulations and costly irradiation experiments while maintaining accuracy and speed.

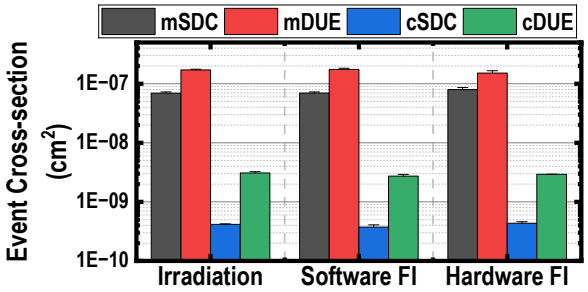


Fig. 4. Cross-section comparison among software FI, hardware FI, and irradiation estimates. The standard deviations for the mSDC, mDUE, cSDC, and cDUE are 5.881e-9, 1.24796e-8, 3.13663e-11, and 1.75107e-10, respectively.

2) Discussion on SEU Estimation: Across software-based FI, hardware-based FI, and physical alpha irradiation, the results of FI are remarkably consistent. The STR error pattern and the system behaviors show a high degree of correlation (R^2=0.984), thereby validating the HachiFI's accuracy and reliability in modeling SEU effects. These consistent results across diverse validation methods affirm the effectiveness of HachiFI as a versatile FI framework capable of bridging the gap between software simulation and hardware testing. The comparable accuracy between the hardware FI and the physical irradiation experiments further demonstrates that HachiFI can serve as a reliable tool for SEU analysis, significantly reducing the need for costly and time-consuming irradiation tests while maintaining high fidelity in fault modeling.

VI. Conclusion

SEUs present major reliability challenges, especially in safety-critical systems. Traditional methods like costly irradiation experiments and limited FI simulations often fall short in covering all SEU scenarios. To address these issues, we developed HachiFI, a lightweight, architecture-independent framework that automates FI and simulates SEU behavior with full coverage with the aid of a small FI control logic embedded in DUT. Our validation on a 22nm edge-AI chip shows a high correlation (R^2=0.984) between FI and irradiation experiments and $<$2% hardware overhead for HachiFI integration, confirming HachiFI's accuracy and low overhead. By bridging simulation and real-world testing, HachiFI enhances reliability assessments in radiation-prone environments and aids in the development of fault-tolerant systems. Our future work is to apply HachiFI to radiation-hardened designs and hardware security analyses.

Acknowledgment

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, by JST CREST, Japan, under Grant JPMJCR19K5, and the Grant-in-Aid for Early-Career Scientists from JSPS under Grant JP21K17721.

REFERENCES

[1] I. Hill, P. Chanawala, R. Singh, S. A. Sheikholeslam and A. Ivanov, "CMOS Reliability From Past to Future: A Survey of Requirements, Trends, and Prediction Methods," in IEEE Transactions on Device and Materials Reliability, vol. 22, no. 1, pp. 1-18, March 2022.

[2] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," in IEEE Transactions on Nuclear Science, vol. 50, no. 3, pp. 583-602, June 2003.

[3] F. Wang and V. D. Agrawal, "Single Event Upset: An Embedded Tutorial," 21st International Conference on VLSI Design (VLSID 2008), Hyderabad, India, 2008, pp. 429-434.

[4] T. Tanaka, W. Liao, M. Hashimoto and Y. Mitsuyama, "Impact of Neutron-Induced SEU in FPGA CRAM on Image-Based Lane Tracking for Autonomous Driving: From Bit Upset to SEFI and Erroneous Behavior," in IEEE Transactions on Nuclear Science, vol. 69, no. 1, pp. 35-42, Jan. 2022.

[5] K. M. Girgis, T. Hada and S. Matsukiyo, "Estimation of Single Event Upset (SEU) rates inside the SAA during the geomagnetic storm event of 15 May 2005," 2021 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), Cleveland, OH, USA, 2021, pp. 27-30.

[6] L. M. Luza et al., "Emulating the Effects of Radiation-Induced Soft-Errors for the Reliability Assessment of Neural Networks," in IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 4, pp. 1867-1882, 1 Oct.-Dec. 2022.

[7] G. Abich, J. Gava, R. Garibotti, R. Reis and L. Ost, "Applying Lightweight Soft Error Mitigation Techniques to Embedded Mixed Precision Deep Neural Networks," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 11, pp. 4772-4782, Nov. 2021.

[8] F. Libano et al., "Selective Hardening for Neural Networks in FPGAs," in IEEE Transactions on Nuclear Science, vol. 66, no. 1, pp. 216-222, Jan. 2019.

[9] S. Blower, P. Rech, C. Cazzaniga, M. Kastriotou and C. D. Frost, "Evaluating and Mitigating Neutrons Effects on COTS EdgeAI Accelerators," in IEEE Transactions on Nuclear Science, vol. 68, no. 8, pp. 1719-1726, Aug. 2021.

[10] J. Gava et al., "A Lightweight Mitigation Technique for Resource-Constrained Devices Executing DNN Inference Models Under Neutron Radiation," in IEEE Transactions on Nuclear Science, vol. 70, no. 8, pp. 1625-1633, Aug. 2023.

[11] Q. Cheng et al., "Reliability Exploration of System-on-Chip With Multi-Bit-Width Accelerator for Multi-Precision Deep Neural Networks," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 70, no. 10, pp. 3978-3991, Oct. 2023.

[12] Z. Chen, N. Narayanan, B. Fang, G. Li, K. Pattabiraman and N. DeBardeleben, "TensorFI: A Flexible Fault Injection Framework for TensorFlow Applications," 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), Coimbra, Portugal, 2020, pp. 426-435.

[13] D. Rath, "Open On-Chip Debugger," Diploma, Department of Computer Science, University of Applied Sciences Augsburg, 2005

[14] G. S. Was, "Challenges to the use of ion irradiation for emulating reactor irradiation," Journal of Materials Research, vol. 30, no. 9, pp. 1158–1182, 2015.

[15] Ripoll, J-F., et al. "Particle dynamics in the Earth's radiation belts: Review of current research and open questions," Journal of Geophysical Research: Space Physics 125.5 (2020): e2019JA026735.

[16] N. Khoshavi, C. Broyles, Y. Bi and A. Roohi, "Fiji-FIN: A Fault Injection Framework on Quantized Neural Network Inference Accelerator," 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 2020, pp. 1139-1144.

[17] F. Libano, B. Wilson, M. Wirthlin, P. Rech and J. Brunhaver, "Understanding the Impact of Quantization, Accuracy, and Radiation on the Reliability of Convolutional Neural Networks on FPGAs," in IEEE Transactions on Nuclear Science, vol. 67, no. 7, pp. 1478-1484, July 2020.

[18] Zhang, F., et al. "An SEU fault injection platform for radiation-harden design debugging in the FPGA," Journal of Instrumentation 17.08 (2022): P08007.