How Accurately Can Soft Error Impact Be Estimated in Black-box/White-box Cases? – A Case Study with an Edge AI SoC –

Quan Cheng[1], Qiufeng Li[2], Longyang Lin[2], Wang Liao[3], Liuyao Dai[4], Hao Yu[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
[4]Department of Computer Science and Engineering, University of California, Merced, USA
[*]{hashimoto@i.kyoto-u.ac.jp}

ABSTRACT

Artificial intelligence (AI) edge devices often feature numerous storage units and sequential logic circuits, making them vulnerable to soft errors. For reliable and critical edge AI applications, assessing System-on-Chip (SoC) reliability in advance is essential. Here, there are two cases: a self-designed SoC (white-box), or a commercial off-the-shelf (COTS) chip (black-box). This study uses alpha particle irradiation results on our 22nm AI SoC as a golden reference to estimate soft error impacts, injecting faults across the entire chip in the white-box case and into the accessible memory and registers in the black-box case. The results demonstrate a high degree of consistency between the white-box case and golden reference, meaning that pre-silicon reliability assessment is feasible. As for the black-box case, the proportion of memory in the SoC remains unchanged and is still significantly larger than that of registers, and hence the simulation results between black-box and white-box are not substantially different.

KEYWORDS

AI SoC; Accelerator; reliability; pre-silicon; white-box / black-box

ACM Reference Format:
Quan Cheng, Qiufeng Li, Longyang Lin, Wang Liao, Liuyao Dai, Hao Yu, and Masanori Hashimoto. 2024. How Accurately Can Soft Error Impact Be Estimated in Black-box/White-box Cases? – A Case Study with an Edge AI SoC – . In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3656537

1 INTRODUCTION

The rise of Artificial Intelligence (AI), especially through convolutional neural networks (CNNs), has increased the demand for high-performance computing units capable of parallel multiply-and-accumulate operations. Deploying AI models on-chip necessitates significant on-chip logic circuits and storage, and hence traditional units using floating-point or single-precision integers are resource-intensive and not ideal for edge computing devices [1]. Multi-precision (MP) neural networks (NNs), using quantized data precision, offer a solution to reduce memory requirements for efficient edge AI applications [2], though memories and sequential cells still consume over 50% of resources in AI edge devices [3, 4].

Recent applications of Deep Neural Networks (DNNs) include safety-critical ones, such as autonomous driving, and then the reliability of DNN and its hardware platform is drawing attention. Soft error, which is primarily caused by cosmic rays in a terrestrial environment and caused by protons and heavier ions in a space environment [5], is the dominant error source during the intermediate device lifetime. Besides, it is known that memories and sequential cells are sensitive to radiation [5]. Therefore, AI accelerators, which inherently include a lot of storage, are threatened by soft errors.

When designing reliability-aware or reliability-demanding systems, it is crucial to model and estimate the impact of soft errors considering all the errors occurring in core components (e.g., processor core) and non-core components (e.g., AI accelerator). Especially, with the growing trend of integrating hardware AI accelerators into edge devices as peripherals [6, 7], there is a need for comprehensive reliability analysis of these core and uncore components [8] from a comprehensive view [9]. However, as most components in commercial off-the-shelf (COTS) platforms are either inaccessible or accessible only at limited timings [10, 11], they can only be treated from a black-box or semi-black-box view, making it challenging to accurately estimate the system disturbance due to soft errors. Moreover, a few studies construct designs from a white-box perspective and precisely estimate soft errors at the component level, but they do not model the entire large-scale AI System on Chip (SoC) and analyze its reliability at a system level [12]. Overall, analyzing the reliability of AI SoCs from a white-box perspective and evaluating the consistency between the white-box and black-box cases is very important. Additionally, the accuracy of the white-box estimation itself also needs to be validated.

Therefore, departing from previous studies with black-box testing (e.g., [10, 11]) that often leaves many units inaccessible, this work targets our custom (i.e., white-box) AI SoC fabricated in a

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3656537

22nm technology. Using the experimental results of alpha particle irradiation to our white-box SoC as a golden reference, this work investigates how accurately soft error impact can be estimated without irradiation experiments by white-box and black-box fault injection (FI) analyses. Specifically, the white-box case injects faults into the entire gate-level netlist, and the black-box case injects faults into the memories and registers visible to programmers and accessible via, for example, a debugger. The white-box FI is supposed to reproduce the malfunctions observed in the alpha irradiation experiment and provide their category-wise error rates. We demonstrate the correlation between the actual measurement and white-box simulation. As for the black-box case, we show how many and what kind of errors can be reproduced by injecting faults into the memory and registers visible to programmers. In other words, we reveal how many and what types of errors originate from the memories and registers that are not disclosed to SoC users. In addition to these, we share our findings regarding the vulnerable components in our SoC and provide insights helpful to improve system reliability. Highlights of this paper are:

- **Silicon Measurement-based Discussion**: The study focuses on a 22nm AI SoC prototype highly optimized for energy-efficient computation, combining pre-silicon and post-silicon measurements to evaluate AI SoC reliability and radiation effects comprehensively. The discussion is secured by solid silicon results of the latest AI SoC.
- **Intensive Black-box and White-box Fault Injection**: The differences between black-box and white-box systems are explicitly evaluated in the context of COTS designs. We confirm that white-box fault injection accurately replicates actual irradiation results, and the black-box case similarly maintains a comparable level of estimation for edge AI SoC in our tests. Besides, it is important to note that when the Error Correction Code (ECC) is applied to memory and register files accessible in the black-box case, the significance of internal flip-flops (FFs) increases. This is expected to widen the discrepancy between white- and black-box scenarios.

2 RELATED WORK

2.1 Soft Error Estimation from Black-box and White-box Views

In recent years, black-box and white-box soft error analyses have been applied in the field of hardware design, particularly in the design of AI SoCs. Lotfi et al. evaluate the resiliency of NN-based automotive object detection running on a GPU through neutron beam experiments [13]. However, their fault injection experiment is limited to NN weights and input images, meaning that only black-box analysis is performed. On the other hand, Cheng et al. conduct an in-depth analysis of their custom AI SoC implemented on a flash-FPGA with neutron irradiation and fault injection [3]. However, their fault injection setup does not directly correspond to white-box or black-box analyses.

Furthermore, in assessing the reliability of these approaches, the study by Balasch et al. on COTS platform components through black-box testing highlights the increased complexity and challenge of fault injection in low-cost devices [14]. This underscores the difficulty of the black-box method in real-world applications. Conversely, Fiji-FIN [12], a comprehensive fault injection framework, is proposed for testing AI models on IoT devices, particularly effective in pinpointing vulnerable spots in Quantized NNs (QNNs). While it assesses the vulnerability of QNN accelerators to various errors, its focus remains limited to the component level without considering overall SoC reliability.

2.2 Reliability of Hardware Designs Targeting AI Applications

Assessing reliability is essential in critical applications, such as autonomous vehicles and medical devices. Various studies have been carried out to evaluate their vulnerability to radiation-induced faults. Some studies reveal that not all soft errors critically impact NNs. For instance, research on Tensor Processing Units (TPUs) indicates that most errors marginally affect convolution output, thereby preserving the accuracy of CNN-based detection and classification in embedded applications, even under high error rates from atmospheric neutrons [10]. Similarly, an analysis of the ARM Cortex-M7 platform shows that MobileNet CNNs are more prone to soft errors in configurations with higher bit-width precision, with activation precision being especially susceptible [11]. Additionally, studies involving flash-based FPGAs demonstrate that while microcontroller memory is significantly vulnerable to neutron-induced faults, AI models exhibit notable fault tolerance [3]. It is also reported that lower data precision, such as FP16 in matrix multiplication, enhances reliability compared to higher precisions like FP32/64 [15]. Moreover, reducing precision in CNNs can lower memory usage and vulnerability, though it potentially increases the severity of errors [16]. Also, DNNs, crucial for object detection in autonomous vehicles, are susceptible to random hardware faults in GPUs [13]. Existing chip-level safety mechanisms, such as ECC and parity, effectively detect transient faults, but the authors suggest a need for more robust strategies to address permanent faults.

These conclusions suggest that weight and activation memories and data-path computation are inherently robust in AI applications. In addition, fault injection to the weight and activation memories is easy. Meanwhile, assessing the effects of the remaining FFs, such as finite state machine (FSM), FIFO, counters, and pipeline registers, which are not usually disclosed to SoC users and are highly dependent on individual chips, is indispensable for reliability-demanding applications. It is crucial to reveal the opportunity and limitations of black-box fault injection.

3 CASE STUDY FRAMEWORK

In this work, we build a RISC-V-based SoC prototype that includes an MP accelerator with a flexible data flow featuring high data reuse and low-latency characteristics for both pre-silicon and post-silicon soft error evaluation. Additionally, a fault injection framework is developed to evaluate the soft error impact in pre-silicon phase.

3.1 AI System-on-Chip Framework

As shown in Fig. 1, our AI processor is architecturally built around four key components: 1) A RV32IMAC RISC-V processor with a 2-stage pipeline. It is equipped with 64KB Instruction Tightly-Coupled Memory (ITCM), 64KB Data TCM (DTCM), and an affluent set of peripherals such as UART, QSPI flash, and JTAG. 2) A MatrixConv unit

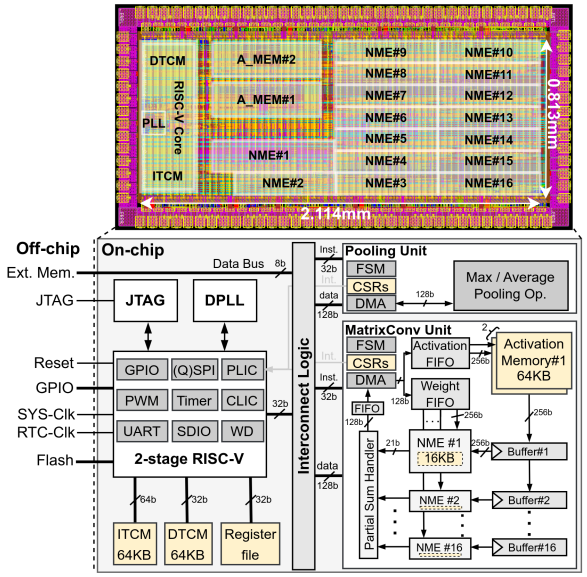

Figure 1: System-on-Chip Framework. The memories and registers in yellow are accessible in the black-box case.

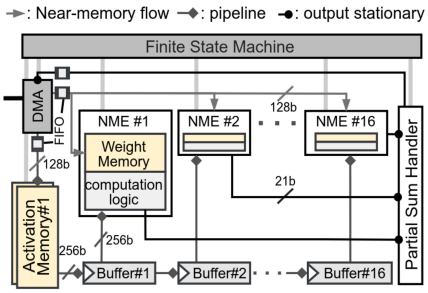

Figure 2: Data Flow in MatrixConv Unit.

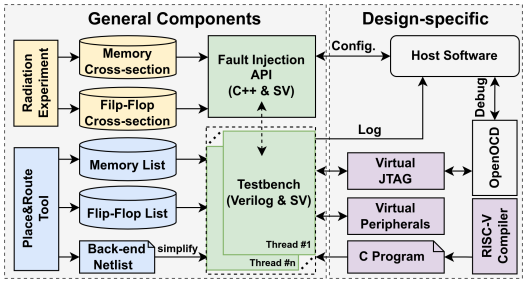

Figure 3: Simulation-based Fault Injection Framework.

designed for convolution and matrix computation requirements, supporting various layer and kernel sizes. It internally comprises a total of 16 Near-Memory Engines (NMEs), 256KB weight buffers, and 128KB activation buffers. 3) A Pooling unit, supporting both max pooling and average pooling. 4) A Digital Phase-Locked Loop (DPLL), providing clock outputs ranging from 100 MHz to 1.2 GHz. Besides, our chip has 89,045 FFs.

To enhance efficient interaction between the RISC-V core and the MatrixConv unit and Pooling unit, as well as access to external memory, on-chip data scheduling is facilitated through an AXI interconnect on the processor. Correspondingly, this interconnect includes two channels. One channel is introduced for configuring the MatrixConv unit and Pooling unit, and the other is used for data transmission to enable efficient access to external memory. The RISC-V core directly interfaces with an FSM in the accelerator via Control Status Registers (CSRs). Furthermore, the DMA units in the MatrixConv unit and Pooling unit are designed to automatically fetch data from external sources based on CSR values configured by the core. Regarding clock sources, our processor architecture primarily consists of two distinct clock domains. Specifically, the accelerator operates at a standard frequency of 600 MHz at 0.9V, while the RISC-V core runs at 100 MHz.

Our design, incorporating a high-performance MP accelerator, features high-performance arithmetic units and high data reuse. Based on the traditional Radix-8 Booth (R8B) algorithm and the booth-based MP multiplier design mentioned in [4], we construct an R8B-based MP multiplier with three configurable precisions (3b/6b/8b) as the basic computation unit. To enhance computational parallelism, we construct 32 MP multipliers within each NME for parallel computation. Within the constraints of the chip area budget, we architect 16 NMEs. This configuration allows for variable input weight data counts of 32, 32, and 64 for 8b, 6b, and 3b data types, respectively, with 16 designated output channels to form the data flow. As shown in Fig. 2, within each NME, the computation units are directly connected to the weight memory. This data flow accommodates three types of data streams: 1) pipeline data flow for activations, 2) near-memory data flow for weights, and 3) stationary data flow for outputs. Each NME can process 64 bytes per clock cycle. The basic arithmetic unit is the MP multiplier, supporting three different precisions (3b/6b/8b). Furthermore, to achieve high data reuse, each weight memory updates data every 1-to-16 cycles, with all NMEs alternating data updates. This means that each weight can be reused 1-to-16 times depending on the configuration. Moreover, since the pipeline stage is 16, activation data can also be reused 1-to-16 times. Besides, weight memories sum up to 256KB while activation memories total 128 KB. The memory size is determined based on the size of prevalent AI models and the number of on-chip PEs. Besides, it should be noted that not only memory sizes but also memory read/write patterns affect the soft error rate.

3.2 Developed Fault Injection Framework

The FI process is designed to evaluate the soft error rate of our design in both white-box and black-box scenarios. To assess the overall reliability of our SoC, we develop a fault injection framework that operates on Synopsys VCS, as illustrated in Fig. 3. It is important to note that while the black-box approach typically involves injecting faults into COTS platforms via debuggers, our simulation-based framework also simulates this aspect of black-box FI. This is achieved by limiting the FI targets, which will be discussed in the next subsection.

In the general parts on the left, the cross-section data for memory and FFs plays a crucial role. This data, representing the likelihood of these units being affected by a radiation particle strike, serves as a benchmark for FI simulations. In the radiation community, 'cross-section' refers to the effective area where a striking particle can cause an upset, a key parameter in evaluating semiconductor device reliability. This data, linked to the chip fabrication process, remains constant in specific radiation environments. We use this cross-section data to set the behavior for data bit-flips. The lists of memory and FFs include all available units for FI and are referenced during

Table 1: Accessible Ranges in White- and Black-Box Cases.

	White-Box		Black-Box	
	FF	**Memory**	**FF**	**Memory**
RISC-V	17176b	128KB	2620b	128KB
MatrixConv	45644b	384KB	793b	384KB
Pooling	26225b	0	607b	0
Total	89045b	512KB	4020b	512KB

bit-flip executions. The back-end netlist is utilized to construct the basic testbench.

Furthermore, to simulate bit-flip operations effectively, the Direct Programming Interface (DPI) allows integration with foreign languages, like C and C++, with the simulator. We develop the FI API for backdoor access to memories/FFs, using a hybrid programming approach with C++ and System Verilog. Additionally, in randomly injecting errors, the timing is determined by a random function, ensuring consistent cross-section values for memories/FFs.

In terms of design-specific parts on the right, since our system includes a processor, we also require a C-language compiler to build AI applications. Additionally, to facilitate the monitoring of the entire system, we develop a host software application based on C++. This application can directly configure and retrieve runtime data from the testbench and can monitor and debug the entire system. For example, after the SoC crashes, we can interrupt the SoC and read register and memory values using an open-source On-Chip Debugger, OpenOCD [17]. By and large, this simulation environment not only allows for injecting errors randomly throughout the entire system to assess its reliability but also enables pinpoint FI to analyze which components within the system are more vulnerable. Besides, as our simulation only focuses on function, all components are simplified with fast functions.

3.3 White- and Black-box Case Definition

Typically, COTS platforms do not grant full access to all on-chip components. Namely, only part of the components can be accessed. Therefore, it increases the difficulty of estimating the soft error impact, specifically from the system level. To better understand the difference from different standpoints, we define the following two cases for FI: 1) white-box case, all components can be accessed; 2) black-box case, only I/DTCM, CSRs, and General Purpose Registers (GPRs) in RISC-V core, and memories and CSRs in the accelerator can be accessed. In black-box cases, the internal logic cannot be accessed (e.g., FSM). The accessible memory and registers in the black-box case are colored yellow in Figs. 1 and 2. Besides, all memory components and FFs do not have any error detection and correction mechanisms. The amount of accessible components of white-box case and black-box case is detailed in Table 1.

4 EXPERIMENTS

In our study, the primary aim is to ascertain the reliability and accuracy of the estimated soft error impact for AI SoC in black-box/white-box cases. For this, we employ the SoC running an MP VGG16 network with 71.17% TOP-1 accuracy under ImageNet dataset for both alpha irradiation experiment and fault injection simulation. Alpha experimental results are regarded as a golden reference, and FI simulation results are analyzed on the basis of the definitions of white- and black-box cases mentioned in Section 3.3.

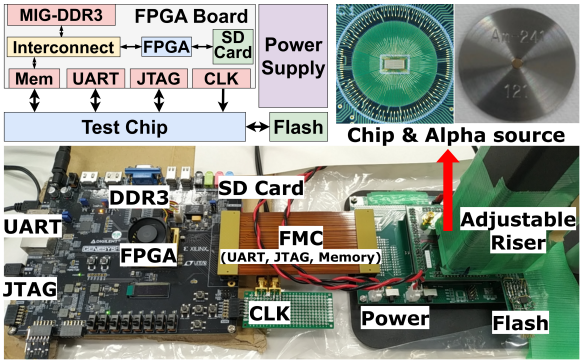

Figure 4: Experimental Setup for Alpha Irradiation.

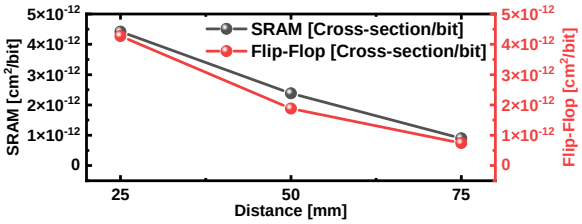

Figure 5: Measured Error Cross-Sections in SRAM and FF.

During the execution of NNs, 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 accelerator 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. Notably, we do not observe any cSDCs in the experiment due to the high inherent reliability of MP NNs and the limited amount of irradiated particles.

4.1 Post-Silicon Alpha Irradiation Experiment

4.1.1 Setup. To assess the robustness of our AI SoC and build a golden reference by measuring actual silicon operations, we design and fabricate a 22nm ASIC chip and conducted alpha irradiation experiments. This approach aims for a comprehensive analysis from a post-silicon perspective without omitting radiation physics and circuit operations, thereby forming a golden reference for comparing the white- and black-box cases.

Fig. 4 depicts the experimental setup employed for the irradiation experiment. We perform an irradiation experiment using an 8kBq Am-241 alpha source with a 2.4mm diameter, where the distance between the alpha source and the chip is varied to control the number and energy of alpha particles reaching transistors in the chip. We prepare a single board featuring our self-developed AI SoC. To begin with, we perform a simple test to estimate the error cross-sections of each type of memory and FFs. Fig. 5 shows the measured data, which is used as reference data in Section 4.2. Note that such cross-section data of basic storage elements could be provided from a foundry to industry.

Furthermore, we select 100 images from the dataset for verifying chip operation and classification accuracy. The testing of 100 images

Table 2: Event Statistics in Alpha Irradiation Experiment.

Distance (mm)	Execution Mode	Fluence (n/cm^2)	mSDC	mDUE	cDUE
25	ITCM	7.8516e+09	168	387	244
50	ITCM	7.5936e+09	84	166	106
75	ITCM	6.6845e+09	55	116	69
25	FlashXIP	6.3114e+09	142	208	2
50	FlashXIP	7.7358e+09	91	134	1
75	FlashXIP	7.9577e+09	43	49	2

is regarded as one event to make event statistics intuitive. We employ the serial port and JTAG port to track all errors. The serial port is used for monitoring mSDC, cSDC, and mDUE. JTAG is employed to monitor cDUE. When the system crashes, JTAG dumps all core register data through the OpenOCD interface. All logs are recorded in the software backend.

4.1.2 Results. During the initial round of the experiment, we observe that the system has a relatively high likelihood of system crash, with the majority of errors originating from the ITCM. Consequently, we introduce an additional operating mode called Flash Execute-In-Place (FlashXIP), where code can be executed directly on the external flash memory. This mode contrasts with the ITCM mode, where the program is first copied to the on-chip ITCM SRAM, and then the code is executed on the ITCM. Note that FlashXIP operation can be regarded as the chip implementation in which ECC is applied to ITCM, and correct instructions are always given[1].

Table 2 presents the statistical breakdown of events, accompanied by fluence data for all 6 cases, which span a range from 6.3114e+9 n/cm^2 to 7.9577e+9 n/cm^2. Fluence is defined as the number of irradiated particles per cm^2. The results for our chip under ITCM mode and FlashXIP mode suggest that the ITCM is markedly more susceptible to error accumulation than the accelerator. This is evidenced by a substantial 98.8% reduction in cDUEs from ITCM mode to FlashXIP mode. In the RISC-V core, the integrity of instruction data within the ITCM is pivotal for application stability. Accumulation of errors within these regions could precipitate a system crash. This result suggests ECC adoption for reduced system crash rate.

Our alpha experiment highlights that the memory in the RISC-V core is more vulnerable compared to the AI accelerator. Moreover, the majority of mDUEs in the accelerator are due to bit flips in the I/DTCM detected by JTAG. In contrast, bit-flips in the memories of the AI accelerator have a minimal impact, as they seldom result in misclassifications. Ultimately, the implementation of error mitigation strategies will depend on the specific requirements of the application, the criticality of the system, and the desired level of reliability needed to safeguard against potential faults.

4.2 Fault Injection Simulation

Table 3 lists FI experimental results of our AI SoC. Even while operating in FlashXIP mode, the system still exhibits a high probability of generating mDUEs. Basically, our observations show that almost all mDUEs originate from bit-flips in DTCM. This is because the configuration information it contains is repeatedly invoked, making it insensitive to the timing of FI. This behavior is explained by the

[1]Rigidly speaking, even ECC and bit interleaving are applied to SRAMs, the output can be wrong at a very low probability due to multiple bit upsets.

Table 3: Event Statistics in Fault Injection Simulation.

Simulated Distance (mm)	Execution Mode	Fluence (n/cm^2)	mSDC	mDUE	cDUE
25	ITCM	8.8137e+08	19/20	45/47	33/33
50	ITCM	1.7636e+09	17/20	43/46	30/30
75	ITCM	2.8392e+09	22/25	51/52	27/27
25	FlashXIP	2.5352e+09	46/51	71/72	2/2
50	FlashXIP	2.9648e+09	28/32	44/46	2/2
75	FlashXIP	4.2891e+09	18/20	21/22	1/1

/ is black-box/white-box

running RISC-V program. The execution code for the NN model, including the functions to divide the model into multiple chunks, and any code used for data processing, NN forward, is stored in ITCM. Meanwhile, the foundational information of the NN model, including network weight size, activation function parameters, intermediate data, etc., is stored in DTCM.

Fig. 6 shows the origins of error components in ITCM execution mode according to the recorded log files. cDUEs are mainly caused by single-bit upsets (SBUs) in ITCM, mDUEs are mainly caused by ITCM and DTCM, and mSDCs are mainly caused by memory components in the AI accelerator. Also, the bit-flips of FFs only account for a tiny part of these error events.

Bit-flips in FFs cause only mDUEs in our fault injection experiment while, in theory, they can cause cDUEs as well. We observe that bit-flips result in mDUEs only when they occur at specific spatial and temporal points during operation. FFs typically refresh their values at each clock edge, often nullifying the impact of bit-flips. This tendency is especially notable in the FFs within data paths. Furthermore, since FFs constitute a relatively small proportion of the SoC, and a bit-flip in an FF must occur at a precise moment to trigger a cDUE, the probability of this happening is quite low. Therefore, we do not observe any cDUEs caused by bit-flips in FFs in our experiment.

Additionally, our statistical analysis reveals that most mSDCs are caused by bit-flips within the memories of the accelerator. This is partly because the accelerator memory volume is 384KB, making up 75% of the total system memory capacity and significantly exceeding the FFs in volume. Furthermore, another FI result obtained using GPUs shows that the cSDC probability for individual weight memory upsets is only 0.00492%; therefore, bit-flips within the accelerator memory predominantly lead to mSDCs. This result is consistent with the observation in [3] that the error propagation probability to misclassification is very low due to the inherent redundancy of NN computation. After a rough estimation, the cross-section of cSDCs should be less than 6.8e-10cm^2 at 2.5mm distance, two orders of magnitude less than other error types.

4.3 Discussion on White-box/Black-box Soft Error Estimation

Fig. 7 shows the event cross-section, which is the number of observed errors divided by the total alpha fluence. Both white- and black-box fault injection simulations exhibit a strong correlation with those from the post-silicon irradiation experiments as depicted in Fig. 8. It can be observed that the results of fault injection in the white- and black-box cases are relatively reliable.

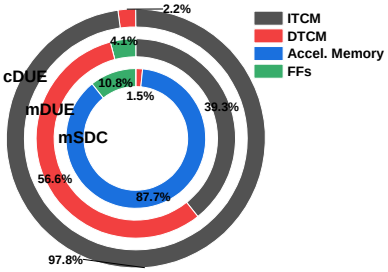

Figure 6: Proportions of Error Source Components.

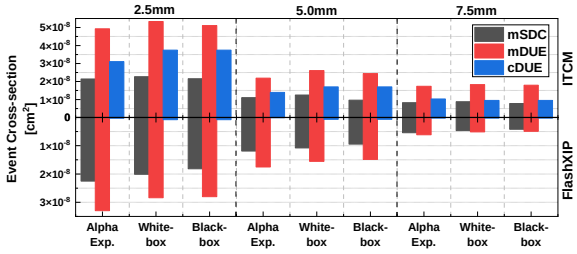

Figure 7: Cross-Section Comparison between Measured, White-Box, and Black-Box Estimates.

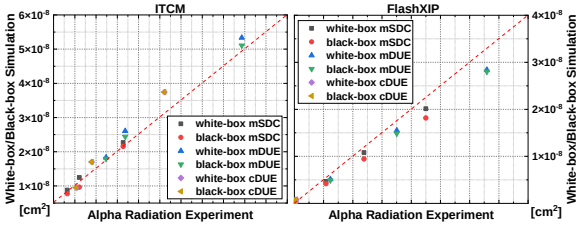

Figure 8: Post-silicon Cross-section vs. Pre-silicon Cross-section.

Both simulation and experimental findings indicate that the I/DTCM are the weakest components in the entire SoC. Furthermore, due to the fact that FFs account for only about 2% of the entire SoC, and a significant part of FFs is used for caching AI model data, coupled with the inherent high reliability of AI models, there is not much difference in the results of experiments in both white-box and black-box cases. Thus, for COTS platforms targeting AI applications, if FI into memory at any given time point is supported, the assessment of soft error impact remains quite reliable even in black-box cases. Besides, assuming the FI of memory components in the accelerator cannot be accessed, the mSDCs reproduced in the black-box FI could decrease significantly. Also, if the FI of ITCM/DTCM cannot be realized, the cDUEs cannot be observed.

Meanwhile, if the memory in COTS platforms contains error detection or correction mechanisms (e.g., ECC, Parity), then the aforementioned analysis no longer applies. Additionally, we construct a special case where only errors are injected into FFs with a simulation distance of 2.5mm, and cDUEs are not observed in our limited simulation time. The rough estimate of the cDUE cross-section caused by FFs should be less than 2.17e-11 cm^2.

5 CONCLUSION

This paper provides a comprehensive evaluation of the reliability of a 22nm AI SoC from both pre-silicon and post-silicon perspectives. The post-silicon alpha experimental results are built as a golden reference. To analyze the difference between pre-silicon white-box and black-box views, we define two different scenarios. Additionally, all these results reveal that the ITCM/DTCM, which store the static program data, NN model information, and cache data, are the most vulnerable component in the AI SoC. Also, we discover that when using FlashXIP as the processor execution mode, the critical errors in the system are due to the non-responsiveness of AI tasks, mainly caused by the upsets in DTCM. Besides, the FI experiment on the NNs shows that the MP NN is considerably more reliable than other system components, and hence, bit-flips in accelerator memories concerning the NN data have almost no impact on the system. On the other hand, the white-box and black-box simulation results are both in line with each other. The comparison indicates that this conclusion applies only to COTS platforms without error detection or correction mechanisms, as it shows that the soft error impact on black-box AI COTS platforms can be well estimated as long as the fault injection can operate on any memory at any given time.

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 JP19H05664, by JST CREST, Japan, under Grant JPMJCR19K5, and the Grant-in-Aid for Early-Career Scientists from JSPS under Grant JP21K17721.

REFERENCES

[1] S. Liu et al., "Enabling Resource-Efficient AIoT System With Cross-Level Optimization: A Survey," in IEEE Communications Surveys & Tutorials.
[2] C. Gong, Z. Jiang, D. Wang, Y. Lin, Q. Liu, and D. Z. Pan, "Mixed Precision Neural Architecture Search for Energy Efficient Deep Learning," Proc. ICCAD, 2019.
[3] Q. Cheng et al., "Reliability Exploration of System-on-Chip With Multi-Bit-Width Accelerator for Multi-Precision Deep Neural Networks," IEEE Trans. CAS-I, Oct. 2023.
[4] M. Huang et al., "A High-Performance Multi-Bit-Width Booth Vector Systolic Accelerator for NAS Optimized Deep Learning Neural Networks," IEEE Trans. CAS-I, Sept. 2022.
[5] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," IEEE Trans. nuclear Science, 2003.
[6] 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," Dig. ISSCC, 2023.
[7] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, "Quentin: an Ultra-Low-Power PULPissimo SoC in 22nm FDX," Proc. S3S, 2018.
[8] H. Cho, E. Cheng, T. Shepherd, C. -Y. Cher and S. Mitra, "System-Level Effects of Soft Errors in Uncore Components," IEEE Trans. CAD, Sept. 2017.
[9] E. Cheng et al., "(Invited) Cross-Layer Resilience: Challenges, Insights, and the Road Ahead," Proc. DAC, 2019.
[10] R. L. Rech and P. Rech, "Reliability of Google's Tensor Processing Units for Embedded Applications," Proc. DATE, 2022.
[11] G. Abich, J. Gava, R. Garibotti, R. Reis and L. Ost, "Applying Lightweight Soft Error Mitigation Techniques to Embedded Mixed Precision Deep Neural Networks," IEEE Trans. CAS-I, Nov. 2021.
[12] N. Khoshavi, C. Broyles, Y. Bi and A. Roohi, "Fiji-FIN: A Fault Injection Framework on Quantized Neural Network Inference Accelerator," Proc. ICMLA, 2020.
[13] A. Lotfi et al., "Resiliency of automotive object detection networks on GPU architectures," Proc. ITC, 2019.
[14] J. Balasch, B. Gierlichs and I. Verbauwhede, "An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs," Workshop on Fault Diagnosis and Tolerance in Cryptography, 2011.
[15] P. M. Basso, F. F. d. Santos and P. Rech, "Impact of Tensor Cores and Mixed Precision on the Reliability of Matrix Multiplication in GPUs," IEEE Trans. Nuclear Science, July 2020.
[16] F. Libano et al., "Understanding the Impact of Quantization, Accuracy, and Radiation on the Reliability of Convolutional Neural Networks on FPGAs," IEEE Trans. Nuclear Science, July 2020.
[17] D. Rath, "Open On-Chip Debugger," Diploma, Department of Computer Science, University of Applied Sciences Augsburg, 2005