Reliability Exploration of System-on-Chip with Multi-bit-width Accelerator for Multi-precision Deep Neural Networks

Quan Cheng, Mingqiang Huang, Changhai Man, Ao shen, Liuyao Dai, Hao Yu, Masanori Hashimoto*

***Abstract*—Deep neural networks (DNNs) in safety-critical applications demand high reliability even when running on edge-computing devices. Recent works on System-on-Chip (SoC) design with state-of-the-art (SOTA) hardware artificial intelligence (AI) accelerators and corresponding multi-bit-width (MBW) convolutional neural network (CNN) generation strategies show that MBW CNNs can effectively explore the trade-off between network accuracy and hardware efficiency. However, reliability has not been considered in such trade-off analysis, even though highly quantized CNNs may elevate the impact of bit flips in the hardware. Also, the reliability of the microcontroller and its interface operating with the AI accelerator are not studied. This work evaluates the reliability of DNN computation in an SoC that includes a processor, SOTA AI accelerator, and NN models highly optimized for computation efficiency using a neural architecture search (NAS) method. Focusing on neutron-induced soft error, which is the primary source of bit-flip errors in a terrestrial environment, we perform fault injection and neutron beam experiments. For these experiments, we prototype the SoC on a flash-based FPGA platform, in which the configuration memory is robust to neutron irradiation. Then, we analyze the experimental data and identify vulnerable components in the system. Furthermore, we evaluate how the SoC running different NAS-optimized MBW LeNet5 networks impact the performance, radiation sensitivity, failure rate of MBW accelerator, and crash rate of the system on the FPGAs. Our results show that instruction and data tightly coupled memory (I/DTCM) are the most vulnerable parts and the control status registers (CSRs) in our accelerator are the second most vulnerable component. Moreover, MBW networks have higher susceptibility to critical errors than single-precision networks, low-precision data are more likely to affect the classification results, and the high bits are more sensitive to faults.**

***Index Terms*—Multi-bit-width, CNN, Accelerator, NAS, Reliability, FPGA, SoC.**

I. Introduction

EDGE-COMPUTING devices integrating artificial intelligence (AI) computation units are emerging as great alternatives to old-fashioned devices that are operated by human labor without automation or advanced technology in recent years. Nevertheless, deep neural networks (DNNs) containing hundreds of megabytes of weights [1] are usually difficult to be deployed into resource-constrained edge-computing devices straightforwardly. Meanwhile, reliability is a key foundation even for edging devices as safety-critical AI applications increase. The reliability of edge devices, as well as the trade-off between performance and AI accuracy, should be taken into full consideration.

In most AI applications, the multiply-and-accumulate (MAC) unit typically handles floating point (FP) or large bit-width single-precision integer (INT) computations. However, constructing computation arrays with these requirements consumes significant resources, which is unsuitable for edge-computing devices. Fortunately, taking advantage of the approximate and resultant quantization-compatible features of CNNs, NAS methods could be applied for data precision reduction with limited accuracy loss [2], [3], thereby reducing the memory footprint and the hardware resource consumption. Therefore, combing NAS-based methods with MBW MAC units is a promising direction for resource-constrained edge applications. Here, based on the approximate characteristics of CNNs, not all soft errors are catastrophic and greatly impact the output of neural networks (NNs) [4], [5]. However, highly precision-reduced NNs mean that each bit needs to carry more information. Therefore, it is particularly important to analyze the reliability of these multi-precision networks.

Moreover, in most low-power edging devices, SoCs, e.g., [6], [7], basically consist of a microcontroller and its peripherals, which is a prevalent trend in current products and research. For AI applications, the hardware AI accelerator can be integrated into the SoC as a peripheral [8], [9]. Depending on the used IPs for the microcontroller and peripherals, black box testing may be conducted for reliability analysis, and some errors are not observable or analyzable. In this work, to comprehensively analyze the reliability of the SoC system, we adopt an open-source microcontroller (e.g., RISC-V and MIPS) for white box testing. Besides, in addition to the reliability of MBW CNNs, it is also essential to analyze the weak points of the entire SoC with MBW accelerator and build high-efficiency error detection and mitigation techniques with low overhead. Therefore, how to tackle these challenges in edging devices deserves further exploration.

To answer the questions mentioned above from a white box side, we perform a case study using a SOTA SoC design that accepts NAS optimized LeNet5 with MNIST data set

Manuscript received 21 Apr 2023; revised 4 July 2023; accepted 29 July 2023. (Corresponding author: Masanori Hashimoto)

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

Quan Cheng and Masanori Hashimoto are with Department of Communications and Computer Engineering, Kyoto University, Kyoto, Japan. (e-mail: hashimoto@i.kyoto-u.ac.jp)

Changhai Man is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA.

Mingqiang Huang, Ao shen, and Hao Yu are with the School of Microelectronics, Southern University of Science and Technology, Shenzhen 518055, China.

Liuyao Dai is with the Department of Electrical Engineering and Computer Science, University of California, Merced, Merced, USA.

and implemented into Microsemi MPF300T Eval Kit, a flash-based FPGA. We analyze the reliability of our SoC by fault injection (FI) and neutron irradiation experiments, aiming to provide valuable insights and serve as crucial references for future reliability-aware designs. The configuration memory in the flash-based FPGA is robust to neutron irradiation compared with SRAM-based FPGA [10]. Therefore, this FPGA-based SoC implementation reproduces the susceptibility of any dedicated SoC chips as long as the flash-based FPGA chip has sufficient resources and the board has required peripherals. Namely, flip-flops and SRAMs in the processor and accelerator are susceptible, while the hardware functionality defined by configuration memories is not damaged by irradiation. This paper aims to analyze the vulnerable parts in MBW CNNs and the reliability of SoC with the MBW accelerator. The main contributions are summarized as follows:

- **Fault injection environment for SoC**: A FI framework is proposed to inject errors and analyze the error results automatically with high efficiency.
- **System-on-Chip with MBW accelerator**: Using an open-source lightweight RISC-V core as a foundation, we construct an SoC with an MBW accelerator to create a practical SOTA AI hardware platform. This platform serves as a white-box solution for FI and radiation experiments, enabling thorough analysis and optimization.
- **Fault injection experiment for MBW Accelerator**: We perform 102.414 million FIs into weights in MBW LeNet5 models with three different precision configurations and similar operations in CSRs in the MBW accelerator.
- **Neutron experiment for SoC**: We proceed with a 4-day 4-configuration neutron experiment. The conclusion is that the RISC-V core, with the highest cross section (1.2348e-6 cm^2 and 1.7680e-6 cm^2 in flash-based FPGA and SRAM-based FPGA respectively) for critical errors, is the most vulnerable part of the SoC, and about 72% events in MBW accelerator are tolerable.

The remainder of this article is organized as follows. Section II introduces backgrounds regarding NAS, the prevalent structure of MBW accelerator, FI framework, and soft error mitigation techniques. Section III discusses the NAS optimization and quantization strategy of MBW LeNet5 and introduces the implementation of the FI environment for our SoC. Section IV presents the FI experiment to evaluate the error immunity performance of MBW LeNet5 and analyze the reliability of the MBW accelerator using architectural vulnerability factor (AVF)-based metrics. Section V presents a neutron experiment and analyzes the cross section of the entire SoC. Section VI discusses our work and related works in soft error assessment and mitigation techniques. Finally, section VII concludes this article and mentions a few possibilities for future work.

II. Background

For safety-critical applications, soft errors need to be detected and alleviated in both hardware and software, and it is essential to reduce the probability of critical failures. To make a comprehensive exploration of the reliability of MBW applications in edging SoC devices, we build an SoC with an up-to-date MBW accelerator for relevant experiments. For this purpose, we survey the literature related to MBW NN training strategies, MBW accelerators, and edging SoC in this article.

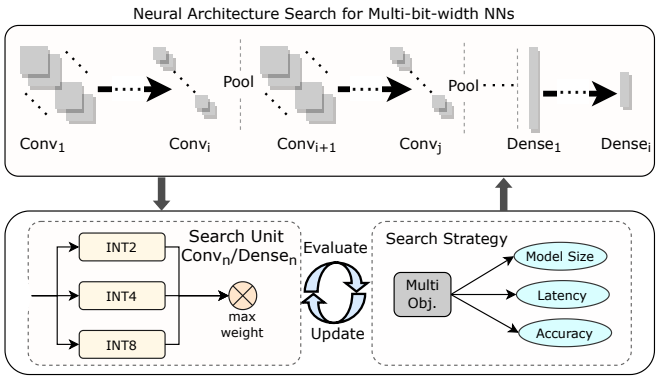


Fig. 1. A subset of NAS for training MBW CNNs.

A. MBW CNN Training and Reliability of Reduced Precision

NN is a mathematical model for emulating how the brain works. NN is typically made up of a large number of neurons. The neurons between every two adjacent layers are interconnected, forming the basic structure of NN. Nevertheless, unlike traditional NN, CNN includes convolutional layers as well as pooling layers, where these layers are connected via sparse connectivity. Although CNN is quite effective, it still consumes high computational power when running on high-performance devices. To achieve low latency, memory overhead, and power consumption, some researchers have proposed that quantizing weights and activations into low-precision data (e.g., INT8/4/2) is an effective approach in edging devices in, e.g., [11], [12].

In addition, the extremely low-bit quantization NNs (e.g., binary NN [13]) often sacrifice the accuracy in exchange for an improvement in energy efficiency [14]. However, as MBW CNNs maintain the balance between model accuracy and energy efficiency by assigning appropriate bit widths to individual layers [15], it is feasible to deploy MBW CNNs into edge devices. To realize high energy-efficiency design in resource-limited edge devices, B. Lu et al. propose performance predictors that can take energy consumption, inference latency, and accuracy into consideration [16]. Nevertheless, searching for the layer-wise bit-width of different layers in deeper NNs remains challenging to obtain satisfactory NNs automatically based on the above predictors. Fortunately, NAS can solve this problem by automatically searching efficient and low-latency MBW NNs with precision reduction as shown in Fig. 1, and it also exhibits better performance than uniformly aggressively-quantized networks [17]. Furthermore, NAS has exhibited unprecedented performance on large-size NNs, such as accuracy-oriented tasks, and size-oriented tasks [18], which means it is applicable to energy-efficient data processing in edge devices [19].

Furthermore, to study the impact of data precision, Basso et al. compare the reliability of FP16, FP32, and FP64 in Matrix

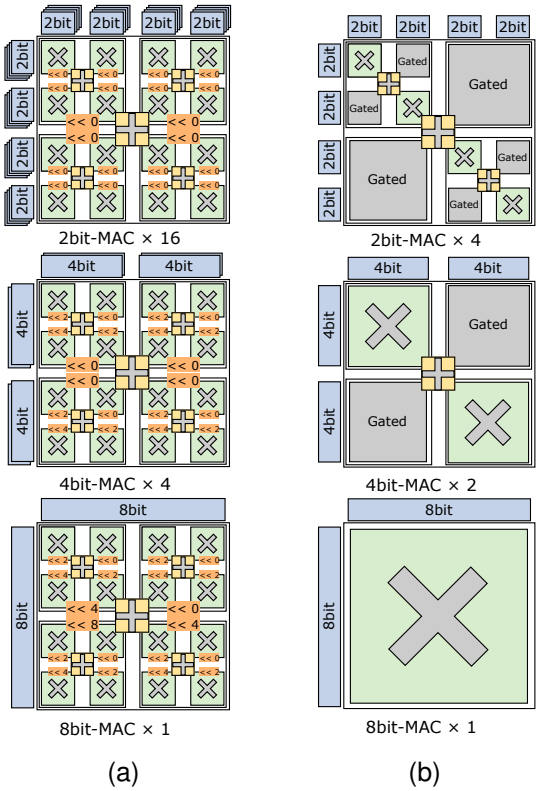


Fig. 2. Multi-bit-width multiply-and-accumulate (a) 2D D&C ST. (b) SWP ST.

Multiplication [20]. They observe that reducing precision improves not only the GPU performance and efficiency but also its reliability. In [21], Libano et al. deploy MNIST CNN on FPGAs using FP32, FP16, and INT8. The experiment shows that reducing the data precision in CNNs can significantly improve overall reliability. The reliability improvement comes from small memory usage. Libano et al. report that binary quantization of weights in convolutional layers leads to lower vulnerability factors, but increases error criticality of misclassification [22]. The MBW NNs optimized by NAS also encounter the increase of error criticality in our FI experiment.

B. MAC and Dataflow in Multi-bit-width Accelerator

To support MBW computations, a basic MBW MAC is necessary. A review of multi-precision MAC is presented in [23]. Basically, two typical MBW MACs are 2D Divide-and-Conquer Sum Together (2D D&C ST) and Subword-Parallel Sum Together (SWP ST), as shown in Fig. 2. 2D D&C ST is based on bottom-up strategy configured for one 8b×8b, four 4b×4b, and sixteen 2b×2b. Conversely, SWP ST is based on a top-down strategy configured for one 8b×8b, two 4b×4b, and four 2b×2b. Almost all MBW MACs are implemented based on these two strategies.

2D D&C ST unit achieves multi-precision MAC operations by adding low-precision multiplication units with configurable shifters. Fig. 2(a) shows 2D D&C ST design for 8-bit, 4-bit, and 2-bit modes, respectively. Each 2D D&C ST unit includes 16 basic 2-bit multiplications. According to the configuration of shifters in different precision modes (e.g., 2bit, 4bit, and 8bit), these 16 shifted results are divided into 4 groups, and the final result is obtained by summing them up. SWP ST unit achieves multiple-precision MAC operations by splitting one high-precision multiplier into multiple lower-precision multipliers. Fig. 2(b) shows the SWP ST design. Compared with 2D D&C ST method, the throughput of low-precision cases is lower. SWP ST method sacrifices the utilization of adders in low-precision computation, resulting in the wastage of hardware resources. In general, in comparison to SWP ST MAC, the control logic, including shifters, selectors, and compensating logic, of 2D D&C ST MAC is more complex, resulting in higher resource consumption. Thus, we adopt SWP ST-based MAC as our baseline.

Furthermore, to achieve parallel MBW computations with large volumes, some accelerators, including MBW processing element (PE) array [24], [25] and configurable MBW data flow [26], [27], are proposed. In addition, it is equivalently important to schedule the data flow in the accelerator. Systolic array (SA) architecture has been introduced to improve energy efficiency by utilizing data reuse and reducing memory footprint of external memory [28]. Moreover, the demand for high-volume parallel MAC operations in edge devices has become an accelerated trend in academia and industry. Thus, SA for accelerating AI computations deserves further exploration.

Moreover, according to the data state in MACs, systolic data flow can be classified as non-stationary type [29], [30] and stationary type [31], [32]. Compared with the broadcast array, SA has better timing and is more applicable for data reuse. On the other hand, it consumes more flip-flops (FFs) and control logic. In general, most designs are based on SA or similar architectures. Overall, a well-designed systolic-based accelerator should possess high-throughput capabilities and an efficient data flow to deploy NNs on edge devices.

C. Fault Injection Framework

Fault injection is a standard technique for evaluating the reliability of systems [33]. Any fault, whether transient or permanent, can lead to application failures, potentially causing disasters in our daily lives. A high-efficiency FI environment is critical to evaluate the system's reliability. FI techniques are categorized into two types: 1) hardware-based (injecting faults into hardware), and 2) software-based (simulating hardware faults at abstract levels). No matter what kind of FI environment is deployed, the essence is to compare the differences between the golden version and the dirty version to analyze the system's integrity. In addition, the experimental results of hardware-based FI platforms [34] are closer to real situations, while the deployment of software-based FI [35] platforms is cheaper and more convenient for reliability analysis.

While both methodologies are applicable to a wide range of applications, FI for accelerators can be directly realized on FPGAs or simulated by code on computers. As the AI accelerator is peripheral in SoC, the output of CNNs in the accelerator will not crash the entire SoC but produce some mismatched outputs. It is important to note that mismatched outputs can potentially lead to significant failures at the application level, while the SoC itself continues to operate. On

the other hand, unacceptable CSR values can cause dead loops or unresponsiveness in the SoC since CSR registers are used for the control of AI accelerator. As a result, in the experiment, we focus on analyzing (1) how the impact of faults leads to silent data corruption (SDC) and (2) how misconfigured CSRs cause detectable unrecoverable error (DUE) across our SoC. Moreover, given the large size of NNs, it is crucial to develop a highly efficient FI platform for conducting reliability analysis.

To study the reliability of CNNs, Xu et al. inject faults into a processor and effectively stimulate the probability of NN failure caused by soft errors while analyzing node data of visual CNNs using Simics simulation platform [36]. Similarly, Barbirotta et al. perform FI and analyze the reliability of RISC-V cores on the UVM platform, relying on the co-design of software and hardware platforms [37]. Paiva et al. propose a comprehensive FI platform for CubeSats software based on JTAG and UART interface [38], providing a good example of FI on the hardware side. In addition, Mamone et al. present a Python-based host computer software to replicate the effects of single event upset (SEU) [39]. Overall, when analyzing the reliability of an AI accelerator, a dedicated FI platform that co-designs host software and hardware platforms is essential to improve test efficiency and flexibility.

D. Soft Error Mitigation Techniques

Mitigating soft errors is accompanied by the reliability analysis of SoC. Related works in literature are implemented based on redundant code and replicated modules. Triple modular redundancy (TMR) is a very common and useful technique in soft error mitigation, and it is adopted in many studies. Wilson et al. [40] apply TMR techniques to a VexRISC-V processor via triplicating all FFs, LUTs, BRAMs, and DSPs on FPGAs, which achieve a 10× improvement in SEU-induced mean fluence to failure but with a cost of 4× resource consumption. Meanwhile, a popular mitigation technique at the instruction level, introduced by [41], involves implementing TMR to recover from soft errors occurring in the register file, and triplicating the verification points. Although these works can be regarded as good references, the high hardware consumption is often unacceptable if the design aims for industrial products. Additionally, register allocation technique (RAT), which is a compiler-based software technique that restricts the number of available CSRs when executing specific functions [42], is also used in many safety-critical designs. RAT reduces the effective bits exposed to radiation, contributing to reliability improvement. RAT does not involve code redundancy and is an architecture-independent approach. Abich et al. [43] attempt to improve soft error reliability via the adoption of two software-based mitigation techniques: Partial-TMR (P-TMR) and RAT.

Another related work includes a fault-tolerant RISC-V processor that incorporates TMR and Hamming code to protect vulnerable parts [44]. Similarly, Cho et al. [45] analyze the effects of soft errors on two RISC-V cores: Rocket and Berkeley Out-of-Order Machine (BOOM). Dos Santos et al. conduct an irradiation experiment to assess the soft error susceptibility of a multi-core RISC-V ASIC platform (GAP8) by executing four simple algorithms and MNIST CNN [46].

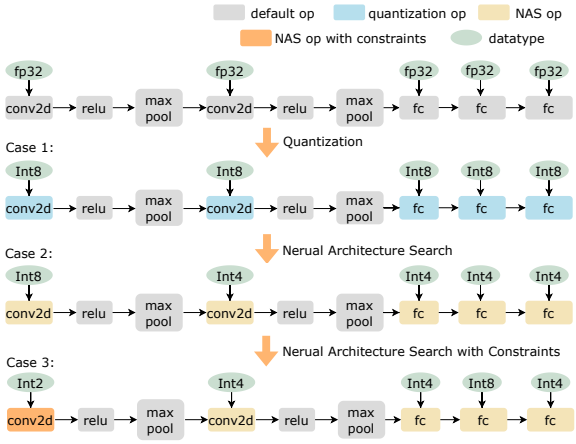


Fig. 3. NAS flow of Multi-bit-width LeNet5.

Currently, although a number of soft error mitigation works are proposed aiming for processor systems, there is no related research that explicitly considers peripheral circuits, such as AI accelerators and their interfaces. Especially, a high-efficiency low-power SoC running AI applications is needed in reliability-demanding applications. Therefore, to analyze and enhance the reliability of SoC, constructing a 100% white-box SoC with a SOTA accelerator is crucial, allowing for an exhaustive exploration of all the details.

III. Design and Implementation of Multi-precision CNN on FPGA

As discussed in section II-B, systolic data flow are widely used in commercial products such as Google TPU. In this work, we construct an SoC including a systolic-array-based MBW accelerator [27] due to its high data reuse and low-latency characteristics. To ensure the relevance of the aforementioned analysis, it is essential to design a low-power SoC specifically for reliability analysis. The following explains the implemented SoC and MBW NNs.

A. NAS Optimized Multi-bit-width LeNet5

To evaluate how MBW data impact the reliability of CNNs, we prepare three cases with different precision distributions. The NAS flow based on differential NAS approach [27] for training MBW LeNet5 is shown in Fig. 3.

First, we train LeNet5 on GPU and generate a full-precision model with FP32 weights. Second, to get a single-precision model of INT8 weights, we quantize all the FP32 weights into INT8 and re-train it to maintain accuracy as much as possible. In this case, we regard FP32 as INT8 ranging from -128 to 127 with certain constraints (INT8 model). Third, we apply this strategy on differential NAS to generate optimized MBW LeNet5 automatically. The obtained model consists of INT4 and INT8 (INT4/8 model). However, the precision distribution of the generated MBW LeNet5 does not contain INT2, indicating that INT2 data adversely affects the performance of LeNet5. Conversely, this work aims to analyze the reliability of comprehensive precisions, including INT2,

TABLE I
PRECISION DISTRIBUTION ON THREE DIFFERENT MBW MODELS.

Layer	Lenet5 Backbone		INT8 model		INT4/8 model		INT2/4/8 model	
	Activation (NCHW)	Weight (OIHW)	Precision (Bit)	Weight Size (Byte)	Precision (Bit)	Weight size (Byte)	Precision (Bit)	Weight size (Byte)
Conv1	1×1×32×32	6×1×5×5	8	150	8	150	2	37.5
Pooling1	1×6×28×28	-	8	-	8	-	2	-
Conv2	1×6×14×14	16×6×5×5	8	2400	4	1200	4	1200
Pooling2	1×16×10×10	-	8	-	4	-	4	-
FC1	1×16×5×5	120×16×5×5	8	48000	4	24000	4	24000
FC2	1×120×1×1	84×120×1×1	8	10080	4	5040	8	10080
FC3	1×84×1×1	10×84×1×1	8	840	4	420	4	420
Total	-	-	-	61470	-	30810	-	35737.5
Accuracy	-	-	98.48%		95.42%		90.25%	

TABLE II
RESOURCE UTILIZATION OF OUR SOC ON MICROSEMI MPF300T EVAL KIT.

Component		Fabric 4LUT	Fabric DFF	Interface 4LUT	Interface DFF	Math (18×18)	uSRAM	LSRAM
Accelerator (100 MHz)	**Conv Unit**	31339	21839	13032	13032	136	54	208
	Pool Unit	33250	26064	408	408	4	22	0
	DMA	375	75	336	336	0	28	0
	Interface	71	55	0	0	0	0	0
	Total	65035	48033	13776	13776	140	104	208
RISCV E203 (16 MHz)		18591	9414	2448	2448	0	12	64
PF_DDR4 (400 MHz)		18404	15219	1272	1272	0	43	21
AXI4 Interconnect		3458	3138	636	636	0	53	0
Others		61	36	0	0	0	0	0
Total		105549	75840	18132	18132	140	212	293
* ITCM Size: 64KB LSRAM; DTCM Size: 64KB LSRAM; Accelerator Buffer Size: 512KB LSRAM; CSR Size: 79*32Bits uSRAM								

INT4, and INT8. Therefore, we introduce special constraint items into the training method to generate a special case (INT2/4/8 model) for LeNet5.

These three models are shown in Table I. INT4/8 model achieves about 50% weight size reduction from INT8 model with 3% accuracy loss. The differential NAS-optimized method has a balanced trade-off between accuracy and weight size. On the other hand, the top-1 accuracy of the INT2/4/8 model has a larger loss, and the weight size is even larger compared with the INT4/8 model. Although this model will not be adopted in terms of performance, we use it as an aggressively-quantized MBW model for reliability analysis.

B. Multi-bit-width Accelerator Implementation

Our SoC platform incorporates the high-performance MBW vector systolic accelerator [27] for FI and neutron experiments. The MAC unit in this accelerator is based on a multi-precision Booth (mp-Booth) multiplier based on the SWP ST method. Also, a vector systolic dataflow is proposed to improve the throughput of the whole system. Considering the available resources in MPF300T Eval Kit, we build a 16 × 8 array, which means that the parallelism of inputs is 16, 32, and 64 for INT8, INT4, and INT2 respectively, and the number of output channel is 8. The resource utilization is shown in Table II. Note that a 64 × 32 PE array implemented on Xilinx ZCU102 [27] achieves 721.5 GOPS on the NAS-optimized VGG16 network on average, and the peek performance of the convolutional layer can reach 764 GOPS for INT8, 1515 GOPS for INT4, and 2574 GOPS for INT2, respectively. Thus, the adopted MBW accelerator is competitive at this moment.

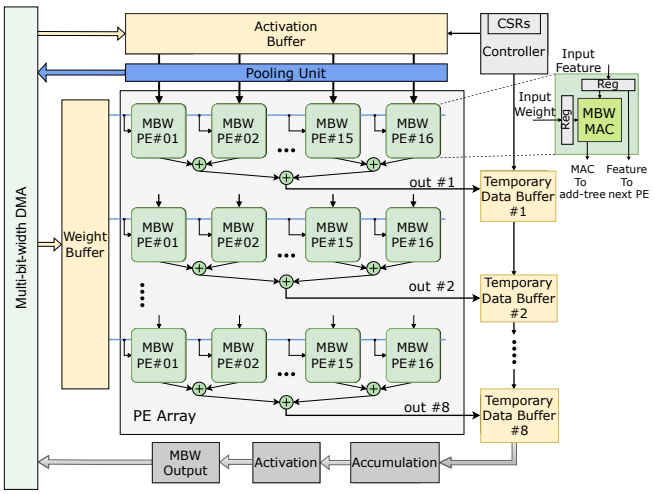


Fig. 4. Architecture of multi-bit-width accelerator.

Fig. 4 shows the architecture of the MBW accelerator. The MBW DMA is used to interact with external memory for data exchange of weights and activations. The operating flow of our MBW accelerator is as follows: 1) We store weights and activations in DRAM, 2) Once the controller completes initialization, the MBW DMA reads data from DRAM and loads it into corresponding buffers sequentially, 3) The data is then sent to PE array of the convolutional unit or pooling unit, controlled by CSR values, 4) For pooling unit, the outputs are transmitted back to DRAM through the MBW DMA for further computations. For the convolutional unit, the outputs are first sent to temporary buffers until the

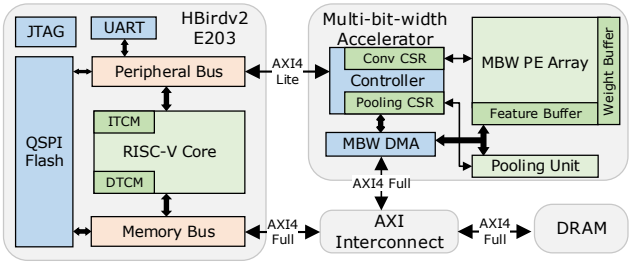


Fig. 5. Block diagram of SoC.

accumulation operations are completed. Then, the final results will be transmitted back to DRAM.

This design involves configuring various parameters of CNNs, such as the weight/activation addresses, model size, and the number of slices, in CSRs. Once these parameters have been set, one of the CSRs can be used to start the computation. The accelerator then conducts the computation according to these parameters, such as slicing the data into multiple rounds of calculation. Once the computation is completed, the results are sent back to the external memory. The accelerator then updates the CSR with a completion signal. This completion signal can be used for various purposes, such as triggering other processes or allowing the microcontroller to retrieve the outputs. Using CSRs to manage the computation process provides a convenient and efficient way to conduct AI model computations. Additionally, it facilitates seamless synchronization between the accelerator and microcontroller.

C. System on Chip Implementation

Fig. 5 shows the architecture of our SoC. This SoC mainly consists of three components: 1) MBW accelerator, 2) HBirdv2 e203 RISC-V processor [47], and 3) DDR4 DRAM. HBirdv2 e203 is an open-source lightweight 32-bit RISC-V microprocessor with 2-stage pipeline, supporting RV32IMAC or RV32EMAC instruction set architecture (ISA). E203 has a wealth of peripherals like UART, QSPI-flash, JTAG, etc.

To provide fast and efficient access to instructions and data, instruction tightly coupled memory (ITCM) and data tightly coupled memory (DTCM) are equipped in e203. Prior to the SoC design, we compare prevalent open-source microcontrollers including Pulpino, HBirdv2 e203, Tiny RISC-V, rocket, BOOM, etc. To make our SoC applicable for resource-constrained platforms, we choose HBirdv2 e203, a pertinent controller with low overhead and abundant peripherals.

We extend two groups of AXI ports on e203 to control the MBW accelerator and access the DRAM. Similarly, we also build two groups of AXI ports in our accelerator. These two ports in the MBW accelerator are interconnected to the peripheral and memory bus in e203. In other words, to integrate this accelerator into our SoC, we develop an AXI4-Lite bus to control our accelerator, and an AXI4-Full bus to exchange data with external memory (DRAM). E203 can directly configure the CSRs in the accelerator via the AXI-Lite bus. The MBW DMA can exchange data with DRAM automatically according to CSR configuration. In short, this accelerator connected to e203 is regarded as a peripheral with a DMA channel in our SoC. Our accelerator and DRAM are extended from the unused space of e203 for user customization. In addition, we allocate 32-bit registers for both convolutional and pooling operations. The detail of key CSRs is shown in Table III.

TABLE III
KEY CSRS OF MBW ACCELERATOR. TOP HALF FOR CONVOLUTION AND BOTTOM HALF FOR POOLING.

CSR	Offset Address	Description
STCONV	0x0000	Start conv operation
CBWIN	0x0008	Bit-width of input data
CFADDR	0x0050	Activation base address
CFSHP	0x0010	Shape of activation
CCONF	0x0024	Padding, Stride and Kernel
CWADDR	0x0060	Weight base address
COADDR	0x0088	Output data base address
CDONE	0x0094	If conv operation done
STPOOL	0x0200	Start pooling operation
PCONF	0x024C	Pad, Stride and Kernel
PIADDR	0x020c	Input data address
POADDR	0x0218	Output data address
PIOSHP	0x0228	Shape of activation
PBWIN	0x02ac	Bit-width of input data
PDONE	0x0204	If pool operation done

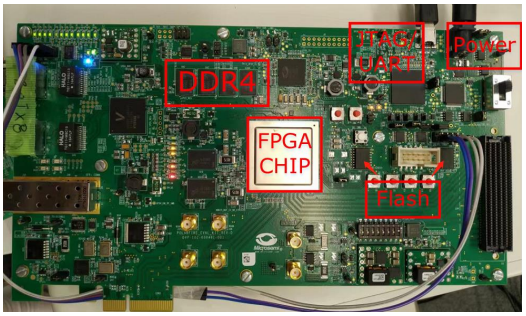


Fig. 6. SoC implementation on MPF300T Eval Kit.

In this SoC, the clock tree includes three domains. The accelerator, DDR4, and e203 run at the speed of 100MHz, 400MHz and 16MHz respectively. An AXI-interconnect IP is used to bridge these components for clock domain cross (CDC) communications. Table II shows the resource utilization. As e203 is a lightweight microprocessor, we only develop a bare-metal program on e203 to control the whole SoC.

Thus, we have successfully constructed an SoC with the MBW accelerator. The SoC is simulated, synthesized, and implemented using Libero on Microsemi MPF300T Eval Kit board as shown in Fig. 6. Then, we adopt three same FPGA boards with the same Libero project but different weights and configuration parameters for the following experiments.

D. Fault Injection Environment

For the reliability analysis of the MBW accelerator, it is vital to develop a FI environment to reproduce the effects of SEU, in particular single bit upset (SBU), focusing on the weights, activations, states of the controller, and configuration parameters of CNNs. In addition, logging the results and saving them via host software for analysis is also necessary.

Our FIE is a hardware-software co-design. The SoC side of FIE, which is developed by C language, works on the RISC-V microprocessor e203, and the host side, developed by C++

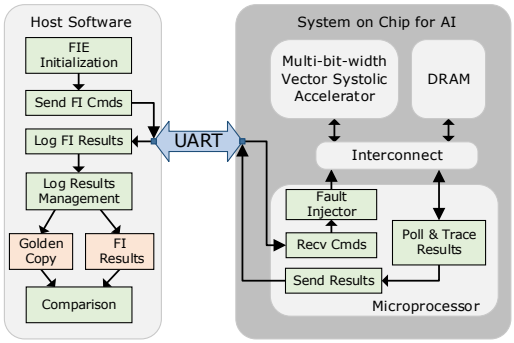


Fig. 7. Fault injection environment for MBW accelerator.

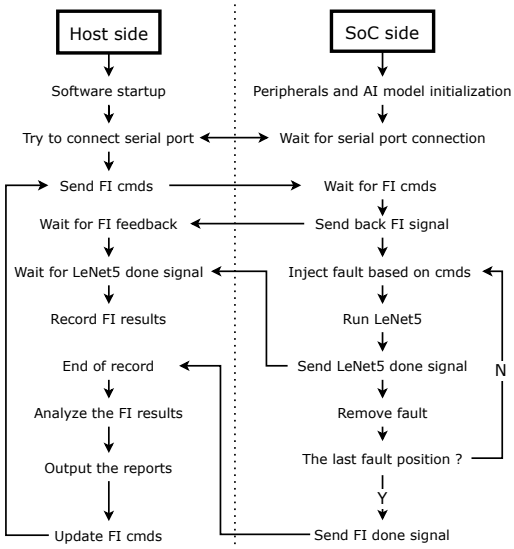


Fig. 8. Fault injection flow.

language, works on the Qt platform as host software. The SoC for AI is controlled by the Qt software that can configure FI campaigns and record the experimental results. Also, our FIE is able to inject SBUs in different targets (e.g., the location of weights, activations, and CSRs), and count errors in NNs automatically. As shown in Fig. 7, the FIE is based on the UART protocol between MPF300T and the computer. Firstly, we define relevant data frames as FI commands and send them to the SoC side via UART after initialization. Secondly, the SoC side receives and decodes the commands. Thirdly, the SoC side injects the fault into the target object via the system bus based on the decoded FI commands. Fourthly, the poll and trace module will get the injection results and send them to the host side via UART. Finally, the host side retrieves all the results and compares them with a golden copy. Besides, the FIE can perform FI campaigns automatically, and all the faults can be injected into desired bits of the data in designated memory locations at given times. Meanwhile, the user has the option to enable or disable this feature, allowing host software to automatically restart the SoC in case of unresponsiveness or when it enters a dead loop without sending completion signals within a specified timeframe.

Fig. 8 depicts the FI flow of the whole system. Retrieving the instructions from the non-volatile memory and loading them into ITCM in RISC-V to initialize the MBW accelerator and relevant peripherals (e.g., UART, GPIO) is the first setup for the SoC. Simultaneously, the host side initiates the software and builds a connection with our SoC. After building the connection, the host side can send the FI commands to the SoC side and wait for the feedback. Then, the SoC side will execute the FI commands (bit-wise operations) and run the AI benchmark after the completion of FI. During the running period, the host side will poll one CSR to check whether the testing of the benchmark is finished. Once the benchmark is finished, the host side will record all the experiment results and the SoC will remove the faults. Furthermore, our FIE supports long-term FI experiments. So, if the FI commands contain multiple error injections, the SoC will loop the process of FI and run the benchmark periodically until all FIs are completed.

IV. Fault Injection Experiment

A. Fault Injection Experimental Methodology

The fault injection experiment aims to assess the sensitive parts in MBW LeNet5. To achieve this, we conduct an exhaustive 3-month experiment for data acquisition. First, we train the MBW LeNet5 using PyTorch with three different precision distributions, as explained in section III-A. This allows us to create three models with varying weights. Specifically, we have 61470, 30810, and 35737.5 bytes of weights in these models as shown in Table I. Second, all the weights, images, and configuration parameters for the MBW accelerator and C program of RISC-V e203 are stored in external flash memory on FPGA boards before the interaction phase. Finally, we adopt our self-developed FIE to coordinate with the MPF300T hardware platform for SBU simulation.

At the beginning of the SBU simulation, all the weights and images will be transferred from external flash to DDR memory under the control of the RISC-V processor. Then, as described in section III-D, the processor will receive FI commands from the host via UART and decode them. It will then perform FIs on the corresponding weight data in DDR memory based on decoded commands. Configuring CSRs will activate the accelerator to perform model computations. Once the computation is completed, it will output the results to the host software and rectify the polluted weight data. The above process will be repeated until all weight errors have been injected. Namely, each bit in the weights is flipped exhaustively one by one. Also, we randomly select 100 data from the test dataset in the experiment, which means we can get 49.176, 24.648, and 28.59 million dirty samples for these cases, respectively. Besides, in this experiment, after linking the FPGA board to our FI host software and initializing the FI mode in FIE, the host software can coordinate with the MPF300T board and collect and analyze all the prediction outputs automatically.

The aforementioned experiment only focuses on FI to the data in NNs. Meanwhile, we need to estimate the effect of CSRs since CSR values can also be affected by radiation-induced upsets. Therefore, we inject faults to some key CSRs, as mentioned in section III-C. We execute the FIs similar

TABLE IV
FAULT INJECTION AT DIFFERENT BIT POSITIONS IN WEIGHTS. ERROR REPRESENTS MISCLASSIFICATION OF NNS. ONLY SDCS, I.E., NO DUES ARE OBSERVED.

Layer	Bit Position	INT8 Model			INT4/8 model			INT2/4/8 model		
		Samples	Errors	Error Ratio	Samples	Errors	Error Ratio	Samples	Errors	Error Ratio
Conv1 (1)	**0**	15K	0	0%	15K	0	0%	15K	120	0.8000%
	1	15K	0	0%	15K	7	0.0467%	15K	217	1.4467%
	2	15K	1	0.0067%	15K	13	0.0867%	-	-	-
	3	15K	0	0%	15K	57	0.3800%	-	-	-
	4	15K	1	0.0067%	15K	94	0.6267%	-	-	-
	5	15K	0	0%	15K	127	0.8467%	-	-	-
	6	15K	0	0%	15K	150	1.0000%	-	-	-
	7	15K	5	0.0333%	15K	143	0.9533%	-	-	-
LAVS(esb) / Dur^l(ns) / RT(ns)		**8.2220e-5 / 1174570 / 299562.5**			**6.9417e-3 / 1174570 / 299562.5**			**3.9583e-3 / 1174570 / 301937.5**		
Conv2 (2)	**0**	240K	0	0%	240K	408	0.1700%	240K	90	0.0375%
	1	240K	0	0%	240K	610	0.2542%	240K	206	0.0858%
	2	240K	1	0.0004%	240K	893	0.3721%	240K	302	0.1258%
	3	240K	11	0.0046%	240K	1415	0.5896%	240K	486	0.2025%
	4	240K	35	0.0146%	-	-	-	-	-	-
	5	240K	75	0.0313%	-	-	-	-	-	-
	6	240K	95	0.0396%	-	-	-	-	-	-
	7	240K	78	0.0325%	-	-	-	-	-	-
LAVS(esb) / Dur^l(ns) / RT(ns)		**5.8437e-4 / 198090 / 155250.0**			**6.5885e-3 / 198090 / 155375.0**			**2.1473e-3 / 198090 / 156562.5**		
FC1 (3)	**0**	4800K	0	0%	4800K	347	0.0072%	4800K	1394	0.0290%
	1	4800K	0	0%	4800K	766	0.0160%	4800K	2320	0.0483%
	2	4800K	0	0%	4800K	1778	0.0370%	4800K	4784	0.0997%
	3	4800K	0	0%	4800K	2688	0.0560%	4800K	9349	0.1948%
	4	4800K	0	0%	-	-	-	-	-	-
	5	4800K	0	0%	-	-	-	-	-	-
	6	4800K	1	0.0000%	-	-	-	-	-	-
	7	4800K	46	0.0010%	-	-	-	-	-	-
LAVS(esb) / Dur^l(ns) / RT(ns)		**4.2300e-8 / 90 /163937.5**			**5.0211e-6 / 90 / 164562.5**			**1.6062e-5 / 90 /165625.0**		
FC2 (4)	**0**	1008K	0	0%	1008K	64	0.0063%	1008K	7	0.0007%
	1	1008K	0	0%	1008K	200	0.0198%	1008K	38	0.0038%
	2	1008K	0	0%	1008K	436	0.0433%	1008K	67	0.0066%
	3	1008K	0	0%	1008K	699	0.0693%	1008K	81	0.0080%
	4	1008K	0	0%	-	-	-	1008K	93	0.0092%
	5	1008K	0	0%	-	-	-	1008K	102	0.0101%
	6	1008K	0	0%	-	-	-	1008K	178	0.0177%
	7	1008K	0	0%	-	-	-	1008K	378	0.0375%
LAVS(esb) / Dur^l(ns) / RT(ns)		**0 / 90 / 114812.5**			**1.2082e-6 / 90 / 107062.5**			**8.1527e-7 / 90 /115437.5**		
FC3 (5)	**0**	84K	0	0%	84K	22	0.0262%	84K	28	0.0333%
	1	84K	0	0%	84K	40	0.0476%	84K	42	0.0500%
	2	84K	0	0%	84K	74	0.0881%	84K	74	0.0881%
	3	84K	0	0%	84K	140	0.1667%	84K	161	0.1917%
	4	84K	0	0%	-	-	-	-	-	-
	5	84K	0	0%	-	-	-	-	-	-
	6	84K	2	0.0024%	-	-	-	-	-	-
	7	84K	56	0.0667%	-	-	-	-	-	-
LAVS(esb) / Dur^l(ns) / RT(ns)		**3.4800e-8 / 90 / 101500.0**			**1.656e-7 / 90 / 100062.5**			**1.8300e-7 / 90 / 101437.5**		
ONAVS(esb) / Total RT(ns)		**6.6667e-4 / 835062.5**			**1.3537e-2 / 826625.0**			**6.1227e-3 / 841000.0**		

* "Dur^l" refers to the effective duration in longest case, not the overall duration, of weight of each layer staying in the buffer.
* Supposing the operating period of the accelerator is P, the effective duration of each weight in LeNet5 can be expressed as:
Dur = $P\times(1 + T_{out\_current})$ + $P\times[(\mathrm{H}_a - \mathrm{H}_\mathrm{w} + 1)\times(\mathrm{W}_a - \mathrm{W}_\mathrm{w} + 1) - 1]\times ceil(\mathrm{I}_w \div T_{in})\times \mathrm{H}_w \times \mathrm{W}_w \times T_{out\_current}$
Where T_{in} = 16 at INT8, 32 at INT4, and 64 at INT2; T_{out} = 8; $T_{out\_current}$ = $(\mathrm{O}_{remaining} \geq T_{out})$? T_{out} : $\mathrm{O}_{remaining}$
* "RT", directly collected from the FPGA in practice, refers to the execution time of each layer per inference.

to the aforementioned experiment. Namely, we flip every bit in correlated CSRs before starting the computation in the accelerator and test them using the same 100 image data.

B. Fault Injection Experimental Results

Table IV lists FI experimental results to NN data. "Errors" represents the misclassification of NNs due to SDCs, where no DUEs are observed. We define Layer Architectural Vulnerability Scoring (LAVS) and Overall Network AVS (ONAVS) to analyze the reliability of weights throughout the entire model. LAVS and ONAVS are the metrics that consider the duration of each weight in the buffer, the averages of AVF across the layer, and the number of network parameters. LAVS and ONAVS are expressed as:

$$\mathrm{LAVS}(l) = [\mathrm{Error}(l)/\mathrm{Sample}(l)] * \mathrm{Dur}(l) * \mathrm{Weight}(l),$$

$$\mathrm{ONAVS} = \sum_{l=1}^{L} \mathrm{LAVS}(l),$$

where l is the layer number, Error(l) is the total error number in l-th layer, Sample(l) is the number of FI samples of l-th

layer, and Weight(l) is the bit number of weights in l-th layer. Then, Error(l)/Sample(l) represents the average AVF of l-th layer. One special treatment in LAVS and ONAVS is that we multiply the averages of AVF by the weight duration, Dur, of corresponding layers, respectively, for fair comparison since a longer duration time means a higher probability of having an upset in the accelerator. Here, as our accelerator's data flow involves data reuse, we provide the duration equation in Table IV for reference. Also, LAVS and ONAVS include Weight(l) since the larger number of weight bits experience more bit flips. With LAVS and ONAVS, the network error rate per inference is expressed as ONAVS $\times$ (SEU rate [/bit/s]). For a particular design, the SEU rate is identical, and hence LAVS and ONAVS represent the NN vulnerability.

The main conclusions are: 1) MBW networks have higher susceptibility to critial errors than single-precision networks. The ONAVS parameters, error-to-second-to-bit (esb), of single INT8 LeNet5, INT4/8 MBW LeNet5 and INT2/4/8 MBW LeNet5 are 6.6667e-4 esb, 1.3537e-2 esb and 6.1227e-3 esb, respectively. 2) The high bits, especially the sign bit, are more sensitive to faults. In this experiment, around 77.1% SDCs come from the high two bits.

These conclusions have not been explicitly addressed in one paper, but have been mentioned in different contexts across multiple papers. In the literature, Libano et al. prove that low-precision layers heighten error criticality of misclassification [22], aligning with our experiment. Additionally, some studies demonstrate that reducing precision can enhance reliability [20], [21]. However, merely reducing data precision does not guarantee an improvement in reliability. In our experiment, we observe that the LeNet5 model, being relatively small and having limited data reuse, exhibits similar execution times with different precisions. Meanwhile, the experimental results show that MBW LeNet5 has a higher misclassification rate, several hundred times greater than INT8 LeNet5. For larger networks, low-precision NNs can achieve significantly lower latency, several times faster than high-precision models (e.g., INT8 and FP32 VGG16). However, if the ONAVS of low-precision NNs is much higher than high-precision NNs, the low-precision models may be less reliable. However, if the ONAVS of low-precision NN is similar to high-precision NNs, low-precision NNs are more reliable due to shorter data duration in memory. In general, the reliability of low-precision NNs is determined by multiple factors, such as hardware architecture, dataset complexity, training strategy, and so on, rather than simply by data precision reduction. Meanwhile, those factors result in the difference of Dur in LAVS and ONAVS. Thus, ONAVS can guide the network reliability exploration.

The FI experimental results of CSRs in the accelerator are shown in Table V. "Acceptable range" of CSRs indicates that the SoC can still perform the model calculation and output the results, while "unacceptable range" indicates that the SoC cannot complete the model calculation, namely DUE errors. Also, the parameter "Error Ratio" means the misclassification ratio in overall errors. Unlike FI of weights, about 50% of CSR tests can inflict misclassification errors. Meanwhile, about 40% of the tests can lead to unacceptable errors. The main conclusions are: 1) the bit-flip of CSRs is far more sensitive than that of weights in NNs, 2) data errors in the acceptable range can lead to a high probability of misclassification, 3) error values in the unacceptable range cause the accelerator to enter into a deadlock or hang the AXI bus.

TABLE V
CRITICAL ERRORS REGARDING CSRS IN MBW ACCELERATOR.

CSR	Error Ratio / Acceptable (%)	Unacceptable (%)
STCONV	0 / 0	100.00
CBWIN	91.7557 / 100.000	0
CFADDR	75.5439 / 100.000	0
CFSHP	76.8271 / 83.0952	16.9048
CCONF	28.6260 / 35.0000	65.0000
CWADDR	69.8139 / 100.000	0
COADDR	77.5191 / 100.000	0
CDONE	0 / 0	100.00
STPOOL	0 / 0	100.00
PCONF	0 / 0	100.00
PIADDR	66.0663 / 100.000	0
POADDR	79.0672 / 100.000	0
PIOSHP	59.3511 / 100.000	0
PBWIN	88.5496 / 100.000	0
PDONE	0 / 0	100.00

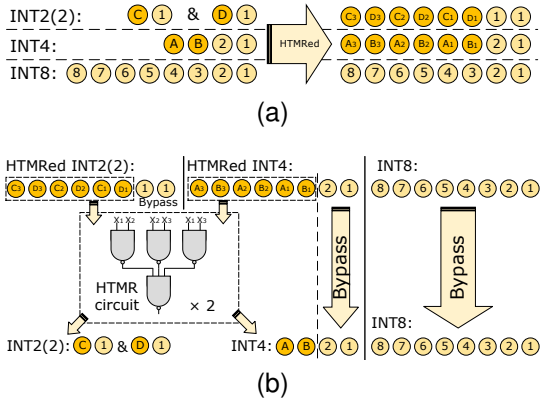


Fig. 9. Hybrid triple modular redundancy (a) Encoder. (b) Decoder.

C. Fault Tolerant Techniques

The above results of our FI experiment indicate that improving the reliability of the accelerator in safety-critical scenarios is necessary. As mentioned in section IV-B, the accelerator is highly sensitive to CSR values. Therefore, we directly introduce the TMR method to CSRs to remedy this situation.

Moreover, the FI experiment indicates that higher bits in data are more sensitive than lower bits. To improve the reliability in low precision, we can introduce suitable mitigation techniques based on the data format stored in the buffer. In the original accelerator, each INT8 data, each two-packed INT4 data, and each four-packed INT2 data occupy 1-byte space. Exploiting this packing feature, we propose a hardware-based hybrid TMR (HTMR) method compatible with INT2 and INT4 computations to improve the reliability of the accelerator. The basic idea of HTMR is bit-selective TMR for INT2 and INT4 as shown in Fig. 9. Bits of C and D in two INT2s and bits of A and B in one INT4 are triplicated. Meanwhile, HTMR mechanism sacrifices the throughput of INT4 and INT2 data. Namely, for each INT4 data or every two INT2 data, there

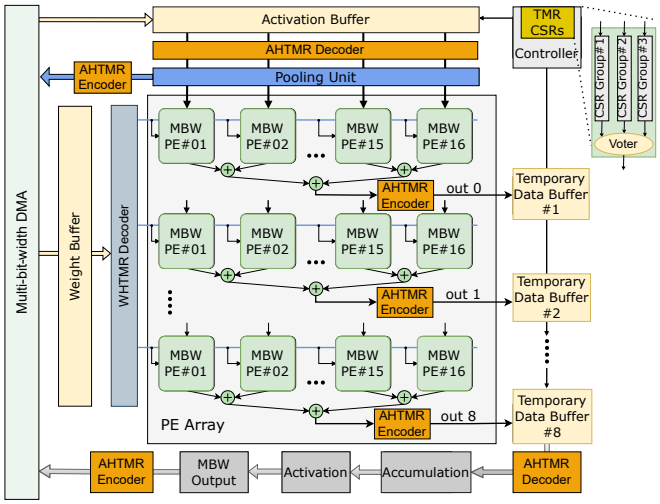


Fig. 10. Architecture of multi-bit-width fault-tolerant accelerator.

TABLE VI
RESOURCE OVERHEAD OF TMR_CSR AND HTMR IN ACCELERATOR.

	Conv Unit		Pool Unit	
	TMR_CSR	**HTMR**	**TMR_CSR**	**HTMR**
4LUT	1668	6406	658	757
DFF	1398	4311	724	785

are 4 redundant bits per byte. In this modified fault-tolerant design, each INT8 data, each INT4 data with HTMR, or each two packed-INT2 data with HTMR occupy 1-byte space.

Fig. 10 shows the architecture of our MBW fault-tolerant accelerator. For CSRs, we simply adopt the TMR method to triplicate every CSR and vote on three redundant inputs to determine final CSR values. As for data flow, we integrate HTMR encoder/decoder modules into the accelerator without significantly changing the data flow in our accelerator. HTMR can effectively fix most data errors in three data paths: 1) from weight/activation buffers to PE units, 2) from PE units to temporary buffers, and 3) from output buffers to DRAM. After integrating the HTMR units, the SDC events caused by the SEU of weights in high 1 bit in INT2 and the SEU of weights in high 2 bits in INT4 are diminished.

We pre-encode weights and activations into HTMRed format as shown in Fig. 9 and store these data in DRAM. All encoded data will be sent to the HTMR decoder and distributed to the PE array after being decoded. Regarding the pooling unit, the outputs will be directly transmitted into Activation HTMR (AHTMR) encoder. As for the convolutional unit, since the final outputs cannot be calculated immediately, we incorporate another group of AHTMR decoder/encoder adjacent to temporary buffers for accumulation operations to enhance the reliability of INT2 and INT4 cases. The mitigation technique can effectively reduce 72.72% weight-induced errors in the INT4/8 model, and 74.93% in the INT2/4/8 model. The resource overhead required for applying TMR CSR and HTMR techniques is shown in Table VI, where the resource overhead is less than 9% of the entire SoC. Also, the throughput of INT2 and INT4 will decrease by half due to the transmission of TMR bits.

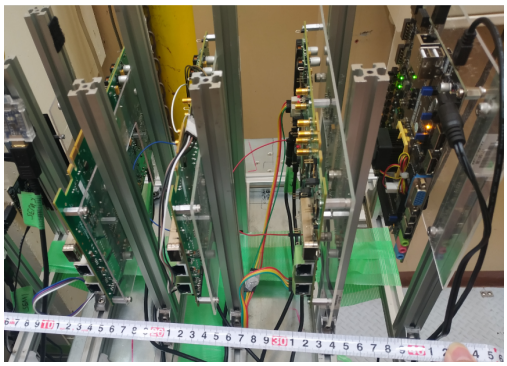


Fig. 11. Experimental setup for neutron irradiation at CYRIC, Tohoku University.

TABLE VII
FOUR RELIABILITY CONFIGURATIONS FOR NEUTRON IRRADIATION.

	Reset	TMR_I/DTCM	TMR_CSR
Config. 1	No	No	No
Config. 2	Yes	No	No
Config. 3	Yes	Yes	No
Config. 4	Yes	Yes	Yes

V. NEUTRON BEAM EXPERIMENT

A. Neutron Beam Experimental Methodology

As shown in the previous section, we analyze the reliability of our accelerator using FI experiments. We have found that the AVF of CSRs is significantly higher than that of weights, and the FI of CSRs has a high probability to cause DUE in the SoC. In addition, we need to evaluate the robustness of the entire SOC in the radiation environment to make a comprehensive comparison. For this purpose, this work adopts a neutron irradiation experiment to evaluate the reliability of our SoC including microprocessors, accelerator CSR, and weights. The neutron irradiation experiment is an effective way to evaluate the reliability of this SoC without any exceptions and overlooking and can help identify potential vulnerabilities. Thus, we can analyze the reliability of these components in our SoC through this experiment.

We perform a 4-day quasi-monoenergetic neutron irradiation experiment at Cyclotron and Radioisotope Center (CYRIC) at Tohoku University [48]. A 70-MeV proton source produces the neutron beam, and the neutron beam has a flux peak at the energy near 70 MeV. To cover INT8, INT4/8, and INT2/4/8 models mentioned in section III-A, we prepare three MPF300T boards that mimic ASIC SoCs thanks to radiation-tolerant CRAM, namely FFs and BRAMs are sensitive to radiation similar to ASIC SoCs. Moreover, to compare ASIC SoC and SRAM FPGA-based SoC, we prepare one extra Xilinx Genesys2 FPGA board running the INT4/8 model. Meanwhile, due to the fact that SRAM-based device is sensitive to radiation, it may accidentally change the configuration information in CRAM. To mitigate this problem, we adopt SEM (Soft Error Mitigation) to scan CRAM in real-time and

TABLE VIII
EVENT STATISTICS OF THREE MODELS IN FOUR RELIABILITY CONFIGURATIONS.

Model	Board	Configuration 1					Configuration 2					Configuration 3					Configuration 4				
		Fluence (n/cm^2)	TCE	TAE	CCE	CAE	Fluence (n/cm^2)	TCE	TAE	CCE	CAE	Fluence (n/cm^2)	TCE	TAE	CCE	CAE	Fluence (n/cm^2)	TCE	TAE	CCE	CAE
INT8	**MPF300T**	1.563e+7	6	13	19	6	9.098e+6	4	1	2	1	1.407e+7	0	2	0	0	9.682e+6	1	0	0	0
INT4/8	**MPF300T**	1.511e+7	6	7	16	4	8.793e+6	2	0	2	4	1.360e+7	0	2	0	1	9.357e+6	0	0	0	0
INT2/4/8	**MPF300T**	1.461e+7	3	10	21	5	8.503e+6	5	4	2	1	1.315e+7	0	4	0	1	9.049e+6	0	2	0	1
INT4/8	**Genesys2**	1.414e+7	4	15	25	8	8.227e+6	10	11	3	16	1.272e+7	1	1	1	8	8.755e+6	1	2	0	4
Total		-	**19**	**45**	**81**	**23**	-	**21**	**16**	**9**	**22**	-	**1**	**9**	**1**	**10**	-	**2**	**4**	**0**	**5**

recover configuration errors caused by radiation. Fig. 11 shows the board setup for the irradiation experiment.

To identify and analyze the vulnerable components of our SoC and rank them according to severity, we build four different reliability configurations using the control variable method for comparison as shown in Table VII. Config. 1 does not integrate any mitigation techniques and accumulates the errors without resetting operations until the system crashes. In Config. 1, we suspect the ITCM and DTCM SRAMs are the most sensitive parts of the SoC. Since we do not refresh the contents of these two SRAMs frequently, especially the ITCM SRAM, they will accumulate errors continuously and crash the SoC frequently, which increases the difficulty of data collection. To remedy this situation, we build Config. 2 which could reset the SoC (i.e., RISC-V, Accelerator, DDR and Bus interconnect) after each round of data collection. On the basis of the experimental data in Config. 2, we discover that ITCM and DTCM SRAMs still cause some critical errors and errors in the CSRs in the accelerator can lead to dead-loop or classification errors of NNs. To better estimate the vulnerability of SRAM in the e203 and the CSRs in the accelerator, Config. 3 replaces the normal SRAM with TMR SRAM to improve the robustness of the RISC-V core, where TMR is selected instead of ECC for just implementation simplicity. In comparison to Config. 2, we could identify the reliability of RISC-V. In addition, Config. 4 replaces the normal CSRs with TMR CSRs in our accelerator so that we can evaluate the reliability of the entire system and analyze the error contribution of CSRs when compared with Config. 3. Besides, in the experiment, we only find 2 samples of misclassification caused by weight/activation bit-flip in LeNet5. Then, we do not build an AI accelerator with the HTMR mechanism in this radiation experiment since enough events are difficult to obtain for comparison.

When running the CNNs, not all errors can be regarded as critical errors. Even though some outputs in middle layers are not expected, the classification result is still correct. Here, we identify four error classes: 1) Tolerable Core Event (TCE): RISC-V core experiences some misbehavior, but it does not affect the function of the NN application; 2) Tolerable Accelerator Event (TAE): classification result of a given image is correct, but the middle outputs are not as expected; 3) Critical Core Event (CCE): the RISC-V core runs away or crashes; and 4) Critical Accelerator Event (CAE): the accelerator has no correct response or correct classification result. Furthermore, we use the serial port to monitor all errors in the accelerator (i.e., TAE and CAE), as well as tolerable errors in the RISC-V core (i.e., TCE). When the RISC-V core encounters intolerable errors and is not able to respond to the commands from host-side software, we use the debug function in FPGA software to monitor the RISC-V privilege level registers to detect the error. This approach allows us to accurately identify and analyze errors during the experiment and comprehensively understand the SoC's fault tolerance and reliability.

B. Neutron Beam Experimental Results

Table VIII shows the event statistical results. The fluence data of four boards, ranging from 8.227e+6 n/cm^2 to 1.563e+7 n/cm^2, are also listed. Furthermore, the cross section (cm^2), which is the number of observed errors divided by the entire irradiated neutron influence, is shown in Fig. 12. In the experiment, we select a set of data from the MNIST dataset that contains 100 image data as those mentioned in section IV-A. We regard the testing of 100 images as one event to make event statistics intuitive. In these four configurations, Config. 1 corresponds to a conventional design and operation without any countermeasures to soft errors, and the results reflect the reliability of the original SoC. In Config. 2, we reset our application after each sampling so that the RISC-V core itself would not accumulate errors continuously. Moreover, in Genesys2, even when we reset the programmable logic circuit, the CRAM may accumulate errors since we do not reload the bitstream to reconstruct the circuit. On the other hand, even though we have adapted SEM interface IP into Genesys2, the SEM IP itself has limited efficiency in scanning and rectifying the error CRAMs of SRAM-based boards. In fact, in Config .2/.3/.4, even if we reset the system, the error with the highest proportion in SRAM-based FPGAs is CAE, whereas this phenomenon does not occur in flash-based FPGAs.

The results of MFP300T in Configs. 1 and 2 indicate that the RISC-V core is more sensitive to the accumulated errors than the accelerator, which comes from the observed large cross section reduction of 81.59% in CCE from Config. 1 to Config 2 in MFP300T. In the RISC-V core, the data in I/DTCM determine the running of the application. When the data in I/DTCM experiences SEUs and the errors are accumulated, the SoC could crash easily. Meanwhile, after deploying the TMR I/DTCM in the RISC-V core (Config. 3), we find that the CCEs almost decrease to zero and the other events also have a significant decrease. Namely, the errors in the accelerator become dominant. Furthermore, after deploying TMR CSRs into our design (Config. 4), the errors generated at the accelerator decreased. The overall cross section is reduced by 78.05% compared with Config. 1.

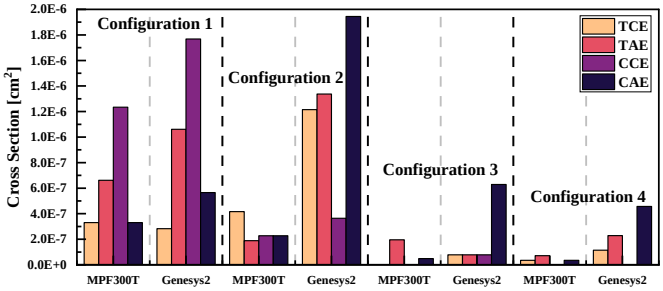


Fig. 12. Neutron cross section for four reliability configurations.

TABLE IX
RELATED WORKS IN SOFT ERROR ASSESSMENT OF NN APPLICATIONS.

Work	Model / Dataset	Datatype	Approach	Platform	Mitigation
[21]	CNN / MNIST	FP32, FP16, INT8	Neutron irradiation	Zynq-7000 & UltraScale+	-
[43]	MobileNet / ImageNet	Mixed INT2/4/8	Simulation	Cortex-M7	P-TMR & RAT
[49]	2-L NN / Iris Flower & 7-L CNN / MNIST	-	Emulation & Neutron irradiation	Zynq-7000	P-TMR & F-TMR
[50]	ANN / MNIST	-	Neutron irradiation	NeuroShield	P-TMR
[51]	7-L CNN / CIFAR-10	Fixed point	Neutron irradiation	Cortex-M7 & Cortex-M4	RAT
Ours	LeNet5 / MNIST	Mixed INT2/4/8	Emulation & Neutron irradiation	MPF300T	HTMR & F-TMR

P-TMR: Partial TMR; F-TMR: Full TMR; L: layer

The observations indicate that the RISC-V core is more vulnerable than the accelerator, and implementing mitigation techniques (e.g., Error Correcting Codes (ECC), TMR) is necessary for instruction and data memory to strengthen the SoC. In addition, applying mitigation techniques to these memories could reveal the vulnerability of the interface between the processor and accelerator. Besides, the impact of precision differences in NNs is limited. Meanwhile, after deploying the above-mentioned countermeasures, the accelerator errors are dominant. In this case, implementing HTMR can effectively improve the reliability of weights. Thus, it is up to the specific application requirements to decide whether to deploy the mitigation techniques listed above. Depending on the criticality of the system and the level of reliability required, a combination of these techniques may be necessary to ensure the SoC's overall reliability to potential faults.

VI. COMPARISON WITH RELATED WORK

Table IX lists the related works in soft error assessment of NN applications. Even though these designs are realized on different hardware platforms and have different architectures, there are two main ways to improve the reliability of the system: 1) using redundant mechanisms to repeat those crucial bits, susceptible to errors or vulnerabilities caused by FIs or radiation in AI applications, in vulnerable components and 2) reducing the usage of crucial bits in vulnerable components exposed to radiation. When analyzing our SoC architecture from a white-box perspective, it is important to select or design appropriate methods that improve system reliability while minimizing overhead. Therefore, as the data flow in our accelerator is shareable at different precisions, we propose the HTMR method to improve both the INT2 and INT4 cases. Moreover, as I/DTCM in the microcontroller and CSRs in the accelerator are the most vulnerable parts verified by the irradiation experiment, we can adopt TMR or ECC methods to these components to further improve the reliability of our SoC. In summary, our efforts contribute to the development of SoC design with accelerator architectures from a white-box perspective, prioritizing reliability in safety-critical applications. Our research on the reliability analysis of SoC with MBW accelerator provides valuable insights for designing reliable hardware for AI applications.

VII. CONCLUSION AND FUTURE WORK

This paper assesses the reliability of a RISC-V-based SoC with the MBW accelerator. The FI results show that CSRs are the most vulnerable component in the accelerator, and the layers with INT2/4 data are more vulnerable than those of INT8 data since every bit of low-precision should carry more information compared with that of high-precision. The results also demonstrate that MBW networks are more fragile than single-precision networks. Moreover, the high bits in low-precision are more prone to the occurrence of misclassification errors. From the hardware-based perspective, we propose an HTMR method to improve the reliability of the MBW accelerator. Even though the HTMR circuit can effectively eliminate the 72.72% and 74.93% errors generated by the INT4/8 model and INT2/4/8 model, it will sacrifice half of the bandwidth and SRAM use of INT4 and INT2 cases. Therefore, this trade-off still needs to be taken into consideration.

In the neutron radiation experiment, we find that the RISC-V core, especially the I/DTCM, is the most sensitive component in the SoC. Moreover, the SBU of CSRs in our accelerator can also cause critical DUE errors (e.g., dead loop, unresponsiveness). Also, the effect of weights and activations has the lowest cross section in the entire SoC. Hence, if an extremely safety-critical AI application is not necessary for edging devices, we could ignore the influence of weights and activations in MBW CNNs since the weights are loaded from external DRAM regularly and the activations are updated frequently.

Future works will seek to improve the reliability of our SoC with the MBW accelerator. We need to look into the errors caused by the data path in the RISC-V core in detail during instruction decoding and execution. We have three areas to focus on. First, we will analyze and optimize the control flow and data in the RISC-V core to attain fault tolerance with low overhead. Second, we will optimize the data flow of our accelerator by combining systolic and broadcast frameworks to reduce FFs overhead and improve reliability. Lastly, we will assess NAS-optimized MBW networks and improve their reliability with a reliability-aware NAS algorithm.

REFERENCES

[1] P. Gysel, J. Pimentel, M. Motamedi and S. Ghiasi, "Ristretto: A Framework for Empirical Study of Resource-Efficient Inference in Convolutional Neural Networks," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 11, pp. 5784-5789, Nov. 2018.

[2] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," 2020.

[3] T. Veniat and L. Denoyer, "Learning time/memory-efficient deep architectures with budgeted super networks," 2018.

[4] M. Klachko, M. R. Mahmoodi and D. Strukov, "Improving Noise Tolerance of Mixed-Signal Neural Networks," 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-8.

[5] T. Liu, W. Wen, L. Jiang, Y. Wang, C. Yang and G. Quan, "A Fault-Tolerant Neural Network Architecture," 2019 56th ACM/IEEE Design Automation Conference (DAC), 2019, pp. 1-6.

[6] A. Pullini, D. Rossi, I. Loi, G. Tagliavini and L. Benini, "Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing," in IEEE Journal of Solid-State Circuits, vol. 54, no. 7, pp. 1970-1981, July 2019.

[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," 2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), 2018, pp. 1-3.

[8] E. Flamand et al., "GAP-8: A RISC-V SoC for AI at the Edge of the IoT," 2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2018, pp. 1-4.

[9] Y. Ju and J. Gu, "A Systolic Neural CPU Processor Combining Deep Learning and General-Purpose Computing With Enhanced Data Locality and End-to-End Performance," in IEEE Journal of Solid-State Circuits, vol. 58, no. 1, pp. 216-226, Jan. 2023.

[10] M. Wirthlin, "High-Reliability FPGA-Based Systems: Space, High-Energy Physics, and Beyond," in Proceedings of the IEEE, vol. 103, no. 3, pp. 379-389, March 2015.

[11] T. Yu, B. Wu, K. Chen, C. Yan and W. Liu, "Energy-efficient Oriented Approximate Quantization Scheme for Fine-Grained Sparse Neural Network Acceleration," 2022 IEEE 40th International Conference on Computer Design (ICCD), Olympic Valley, CA, USA, 2022, pp. 762-769.

[12] W. Cheng, I. -C. Lin and Y. -Y. Shih, "An Efficient Implementation of Convolutional Neural Network With CLIP-Q Quantization on FPGA," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 10, pp. 4093-4102, Oct. 2022.

[13] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating Binarized Convolutional Neural Networks with Software-Programmable FPGAs," in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '17). Association for Computing Machinery, New York, NY, USA, 15–24.

[14] I. Chakraborty, D. Roy, I. Garg, A. Ankit, and K. Roy, "Constructing energy-efficient mixed-precision neural networks through principal component analysis for edge intelligence," Nature Machine Intelligence, vol. 2, no. 1, pp. 1–13, 2020.

[15] R. Gong et al., "Differentiable Soft Quantization: Bridging Full-Precision and Low-Bit Neural Networks," arXiv:1908.05033, 2019.

[16] B. Lu, J. Yang and S. Ren, "Poster: Scaling Up Deep Neural Network optimization for Edge Inference," 2020 IEEE/ACM Symposium on Edge Computing (SEC), San Jose, CA, USA, 2020, pp. 170-172.

[17] H. Alemdar, V. Leroy, A. Prost-Boucle and F. Pétrot, "Ternary neural networks for resource-efficient AI applications," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 2017, pp. 2547-2554.

[18] C. Gong, Z. Jiang, D. Wang, Y. Lin, Q. Liu and D. Z. Pan, "Mixed Precision Neural Architecture Search for Energy Efficient Deep Learning," 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 2019, pp. 1-7.

[19] H. Cai et al., "Proxylessnas: Direct neural architecture search on target task and hardware," arXiv:1812.00332, 2019.

[20] 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," in IEEE Transactions on Nuclear Science, vol. 67, no. 7, pp. 1560-1565, July 2020.

[21] F. Libano, P. Rech, B. Neuman, J. Leavitt, M. Wirthlin and J. Brunhaver, "How Reduced Data Precision and Degree of Parallelism Impact the Reliability of Convolutional Neural Networks on FPGAs," in IEEE Transactions on Nuclear Science, vol. 68, no. 5, pp. 865-872, May 2021.

[22] 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.

[23] V. Camus, L. Mei, C. Enz and M. Verhelst, "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.

[24] C. Wu, M. Wang, X. Chu, K. Wang, and L. He, "Low precision floating point arithmetic for high performance fpga-based cnn acceleration," in Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, ser. FPGA '20, 2020.

[25] Y. Zhao et al., "Automatic Generation of Multi-Precision Multi-Arithmetic CNN Accelerators for FPGAs," 2019 International Conference on Field-Programmable Technology (ICFPT), 2019, pp. 45-53.

[26] B. Asgari, R. Hadidi and H. Kim, "MEISSA: Multiplying Matrices Efficiently in a Scalable Systolic Architecture," 2020 IEEE 38th International Conference on Computer Design (ICCD), 2020, pp. 130-137.

[27] 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, 2022.

[28] H. T. Kung and C. E. Leiserson, "Systolic arrays for VLSI sparse matrix proc," in Proc. Sparse Matrix Conf., 1978, pp. 256–282.

[29] H. T. Kung, "Why systolic architectures?" Computer, vol. 15, no. 1, pp. 37–46, Jan. 1982.

[30] H. Lim, V. Piuri, and E. E. Swartzlander, "A serial-parallel architecture for two-dimensional discrete cosine and inverse discrete cosine transforms," IEEE Trans. Comput., vol. 49, no. 12, pp. 1297–1309, Dec. 2000.

[31] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Architecture (ISCA), Jun. 2017, pp. 1–12.

[32] R. B. Urquhart and D. Wood, "Systolic matrix and vector multiplication methods for signal processing," IEE Proc. F, Commun., Radar Signal Process., vol. 131, no. 6, pp. 623–631, Oct. 1984.

[33] M. Kooli et al., "A survey on simulation-based fault injection tools for complex systems," in 2014 9th IEEE International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS), May 2014, pp. 1–6.

[34] M. Ebrahimi et al., "A fast, flexible, and easy-to-develop fpga-based fault injection technique," Microelectronics Reliability, vol. 54, no. 5, pp. 1000–1008, 2014.

[35] J. Carreira et al., "Xception: A technique for the experimental evaluation of dependability in modern computers," IEEE Transactions on Software Engineering, vol. 24, no. 2, pp. 125–136, February 1998

[36] X. Xu, Y. Fu, T. Liu and H. You, "Fault Tolerance Research of Visual Convolutional Neural Networks Based on Soft Errors," 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2020, pp. 566-570.

[37] A. Aponte-Moreno, F. Restrepo-Calle and C. Pedraza, "Reliability Evaluation of RISC-V and ARM Microprocessors Through a New Fault Injection Tool," 2021 IEEE 22nd Latin American Test Symposium (LATS), 2021, pp. 1-6.

[38] D. Paiva, J. M. Duarte, R. Lima, M. Carvalho, F. Mattiello-Francisco and H. Madeira, "Fault injection platform for affordable verification and validation of CubeSats software," 2021 10th Latin-American Symposium on Dependable Computing (LADC), 2021, pp. 1-11.

[39] D. Mamone, A. Bosio, A. Savino, S. Hamdioui and M. Rebaudengo, "On the Analysis of Real-time Operating System Reliability in Embedded Systems," 2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2020, pp. 1-6.

[40] A. E. Wilson, S. Larsen, C. Wilson, C. Thurlow and M. Wirthlin, "Neutron Radiation Testing of a TMR VexRiscv Soft Processor on SRAM-Based FPGAs," in IEEE Transactions on Nuclear Science, vol. 68, no. 5, pp. 1054-1060, May 2021.

[41] G. A. Reis, J. Chang, and D. I. August, "Automatic instruction-level software-only recovery," IEEE Micro, vol. 27, no. 1, pp. 36–47, Jan. 2007.

[42] J. Gava, R. Reis, and L. Ost, "RAT: A lightweight system-level soft error mitigation technique," in Proc. IFIP/IEEE 28th Int. Conf. Very Large Scale Integr. (VLSI-SOC), Oct. 2020, pp. 165–170.

[43] 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.

[44] D. A. Santos, L. M. Luza, C. A. Zeferino, L. Dilillo and D. R. Melo, "A Low-Cost Fault-Tolerant RISC-V Processor for Space Systems," 2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS), 2020, pp. 1-5.
[45] H. Cho, "Impact of Microarchitectural Differences of RISC-V Processor Cores on Soft Error Effects," in IEEE Access, vol. 6, pp. 41302-41313, 2018.
[46] F. F. Dos Santos, A. Kritikakou and O. Sentieys, "Experimental evaluation of neutron-induced errors on a multicore RISC-V platform," 2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS), 2022, pp. 1-7.
[47] Yuan, Jun, et al. "Audio Denoising Coprocessor Based on RISC-V Custom Instruction Set Extension." Acoustics. Vol. 4. No. 3. MDPI, 2022.
[48] Y. Sakemi, M. Itoh, T. Wakui et al., "High intensity fast neutron beam facility at CYRIC,", IAEA, 2014.
[49] 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.
[50] 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.
[51] 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.

Quan Cheng received dual bachelor degrees in Electronic Information Engineering from Wuhan University of Technology and Architecture from Huazhong University of Science and Technology, Wuhan, China, in 2017, and the M.Eng degree in Electronics and Communication Engineering from Wuhan University of Technology in 2019. He is currently the Ph.D. candidate with the Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University, Japan. His research interests include digital IC design, FPGA, embedded system, reliability analysis, etc.

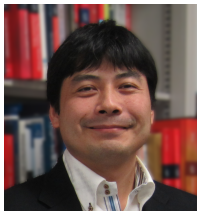
Mingqiang Huang received his B.Eng. and Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China in 2013 and 2018, respectively. Then he was a research fellow in Nanyang Technological University (Singapore), focusing on energy efficient microelectronics and logic circuits. Since 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 memristor in-memory computing, AI accelerator.

Changhai Man is currently the Ph.D candidate with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA. His research interests include automated machine learning, digital system design, and neural network acceleration.

Ao Shen is currently an undergraduate student in microelectronics science and engineering with the School of Microelectronics, the Southern University of Science and Technology, ShenZhen, China. His research interests include neural network compiler, digital system design.

Liuyao Dai is a Ph.D. student at the University of California, Merced. He previously obtained his M.S. degree in Electrical Engineering from the Southern University of Science and Technology and his B.S. degree from Huazhong University of Science and Technology in 2022 and 2018, respectively. His research interests lie in the co-design of computer software and hardware, with a particular focus on GPU-based systems.

Hao Yu (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, USA, in 2007. He is currently with the Southern University of Science and Technology, Shenzhen, China. His research interests include energy-efficient data links, sensors, and analysis. He has about 282 peer-reviewed IEEE/ACM publications; nine books; one Best Paper Award of ACM transaction; three keynote talks; three Best Paper Award nominations at the Design Automation Conference in 2006, the International Conference on Computer-Aided Design in 2006, and the International Conference on VLSI Design Automation in Asia and South Pacific region in 2012; one Inventor Award from Semiconductor Research Corporation; and 20 granted patents. He is a Senior Member of ACM. He is a Technical Program Committee Member of the IEEE Custom Integrated Circuits Conference, the IEEE Asian Solid-State Circuits Conference, the ACM-DAC, the ACM Design, Automation and Test in Europe Conference and Exhibition, and many IEEE/ACM international journals and conferences. He is an Associate Editor of *Scientific Reports* (Nature), IEEE TRANSACTION ON BIOMEDICAL CIRCUITS AND SYSTEMS, *ACM Transactions on Embedded Computing Systems*, and *Microelectronics* (Elsevier).



Masanori Hashimoto (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in communications and computer engineering from Kyoto University, Kyoto, Japan, in 1997, 1999, and 2001, respectively. He is currently a Professor with the Graduate School of Informatics, Kyoto University. His research interests include design for reliability, timing and power integrity analysis, reconfigurable computing, soft error characterization, and low-power circuit design. He was the TPC chair for Asia and South Pacific Design Automation Conference (ASP-DAC) 2022 and International Midwest Symposium on Circuits and Systems (MWSCAS) 2022. He was/is on the Technical Program Committees of international conferences, including Design Automation Conference (DAC), International Conference on Computer-Aided Design (ICCAD), International Test Conference (ITC), Symposium on VLSI Circuits, and Design, Automation and Test in Europe Conference (DATE). He serves/served as the Editor-in-Chief for *Microelectronics Reliability* (Elsevier) and an Associate Editor for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, and *ACM Transactions on Design Automation of Electronic Systems* (TODAES).