

Late Breaking Result: FPGA-Based Emulation and Fault Injection for CNN Inference Accelerators

Filip Masar, Vojtech Mrazek, Lukas Sekanina

Brno University of Technology, Faculty of Information Technology, Brno, Czechia

Email: xmasar18@stud.fit.vutbr.cz, mrazek@fit.vutbr.cz, sekanina@fit.vutbr.cz

Abstract—A new field programmable gate array (FPGA)-based emulation platform is proposed to accelerate fault tolerance analysis of inference accelerators of convolutional neural networks (CNN). For a given CNN model, hardware accelerator architecture, and FT analysis target, an FPGA-based CNN implementation is generated (with the help of the Tengine framework), and fault injection logic is added. In our first case study, we report how the classification accuracy drop depends on the faults injected into multipliers used in Multiply-and-Accumulate Units of NVDLA inference accelerator executing ResNet-18 CNN. The FT analysis emulated on Zynq UltraScale+ SoC is an order of magnitude faster than software emulation.

I. INTRODUCTION

While convolutional neural networks (CNNs) are inherently error-resilient [1], there is still a critical need for ensuring fault tolerance (FT) in some of their applications, e.g., in autonomous driving and automotive systems. This paper deals with FT analysis of hardware inference accelerators of CNNs.

A common approach to the FT analysis is based on injecting faults into a CNN software model or CNN circuit implementation (depending on the chosen level of abstraction), collecting responses for test stimuli, and computing final statistics [2]. The easiest but the least reliable FT analysis is performed at the CNN execution graph in software (e.g., by introducing “stuck-at-0” faults at the outputs of multiplication operations or disconnecting some model’s components). More advanced modeling approaches consider the real architecture of the hardware inference accelerator, where a CNN model is executed [3]. The accelerator is seen as an array of Processing Elements (PEs). The CNN model is executed “layer by layer” according to a precomputed execution plan (also called a mapping). For example, Fidelity generates a simulation model based on a real accelerator such as the NVDLA (NVIDIA Deep Learning Accelerator) and injects bit flip errors into the software model [3]. SAFFIRA employs a simulation model based on a homogeneous PE network and utilizes the Uniform Recurrent Equation system to capture the CNN behavior [4]. As this method is computationally expensive, only the most critical CNN layers are analyzed [4]. Simulator-based FT analysis often suffers from imprecise modeling of the underlying hardware (i.e., the FT analysis is not reliable) and long execution times [2], [4], [5].

To overcome these limitations, we propose to extend an existing hardware accelerator to (1) support fault injection (FI) according to user-preferred strategies and (2) perform a fast FT analysis using a hardware emulator implemented on the Zynq UltraScale+ SoC.

Available CNN accelerators developed for FPGAs and ASICs differ in their ability to incorporate additional error injection mechanisms. For instance, the AMD Deep Learning Processor Unit is built entirely from firm IP cores. These cores are configurable units that cannot be modified to emulate faults within specific compute units. Another example is the DnnWeaver project aiming at converting CNN models into FPGA-compatible formats for computational acceleration [6]. As this conversion is done through the CNN specification in Caffe, the resulting accelerator is highly specialized to the input CNN. It does not usually show the architecture typical for generic CNN accelerators. Eyeriss [7] – a typical example of a generic accelerator based on a PE array – has not been designed for FPGAs. For purposes of this project, we chose the open-source NVIDIA Deep Learning Accelerator (NVDLA) [8] utilizing a generic array of PEs. Although primarily intended for ASICs, with some minor modifications, it can be utilized on an FPGAs.

Our goal is to develop an FPGA-based emulation platform to accelerate fault tolerance analysis of inference accelerators of CNNs. In addition to accelerating the FT analysis, the platform could be used in the future to validate software FT analysis tools for CNNs and study the impact of introducing various FT mechanisms (such as error-correcting codes or redundant components) into inference accelerators of CNNs. In the case study, we consider NVDLA accelerator implementing CIFAR-10 image classifier based on ResNet-18 CNN. We analyze (1) how the classification accuracy drop depends on the faults injected into multipliers used in Multiply-and-Accumulate (MAC) units and (2) the speedup of the FT analysis conducted in FPGA contrasted to a software simulation. The proposed emulator is available as open-source software at <https://github.com/ehw-fit/zynq-nvdla-faultinjection>.

II. THE PROPOSED EMULATION PLATFORM

Fig. 1 shows the architecture of the proposed emulation platform based on UltraScale+ XCZU7EV Zynq SoC. The accelerator has been created with the help of NVDLA Xilinx FPGA Mapping tool [9]. It comprises 8x8 signed 8-bit multipliers arranged into eight MAC units, each of them containing eight multipliers. The FPGA logic accommodates the array of MAC units (and other operations of the accelerator) whose outputs we equipped with a fault injection logic. A CNN (pre-trained in the Caffe tool) is converted to an execution plan for the accelerator using the Tengine framework [10]. The execution of the CNN and fault injection are controlled by software running on on-chip ARM cores. As this is the first

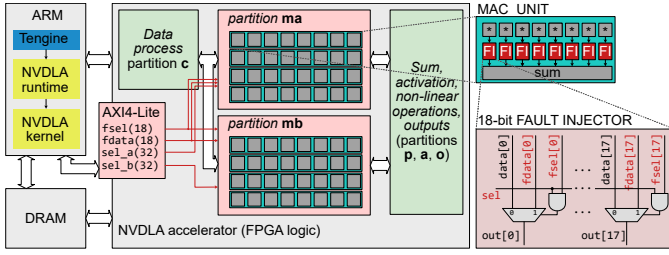


Fig. 1: The proposed emulation platform.

experiment, only a small ResNet-18 is considered, utilizing 8-bit precision and excluding features like SRAM cache, re-shaping, Winograd convolution, and weight compression. Our software modifications of the accelerator implementation [9] include an improved software stack for the Linux subsystem and an adapted Linux kernel to accommodate the proposed accelerator. The fault injection logic controlled via the AXI4 interface bus enables us to override the 18-bit output value of each multiplier with either zero or a constant value, emulating thus a stuck-at error or a pulse fault, respectively. Other fault models can easily be incorporated by modifying the source code.

III. EXPERIMENTAL RESULTS

Table I shows the performance and synthesis results for the 8-bit NVDLA accelerator with and without the fault injection support on Zynq, embedded ARM core, and a common PC when executing ResNet-18. The pre-trained ResNet-18 was taken from the Tengine Model ZOO and exhibits the 75.5% classification accuracy on CIFAR-10. The hardware implementation performs $4.9\times$ faster than a single-thread ARM implementation (without FI) and $2.5\times$ faster than a single-thread AMD Ryzen 7700 implementation. When a single global fault has to be injected, the number of LUTs (Look-Up Tables) implementing this fault injection to selected multiplier(s) increases by 18 compared to the NVDLA without the FI support. If the injected value can be dynamically selected, the number of LUTs increases by 0.71% and the number of flip-flops (FF) by 0.31%.

In the first experiment, we injected three types of constant errors (0, 1, and -1) into randomly selected multipliers. The results obtained from 210 FIs are illustrated using box plots in Fig. 2. As expected, a noticeable decrease in accuracy is observed when the number of affected multipliers increases, independently of the error value injected.

In the second experiment, we investigated whether specific positions of the multipliers or the entire MAC unit within the

TABLE I: Performance and synthesis results for the 8-bit NVDLA accelerator with and without the fault injection support on Zynq, embedded ARM core, and AMD Ryzen when executing ResNet-18 on CIFAR-10.

Device	Threads / Frequency	Inference (ms)	#LUT	#FF
ARM Cortex-A53 (Zynq)	1 / 1.3 GHz	22.68	—	—
ARM Cortex-A53 (Zynq)	4 / 1.3 GHz	14.12	—	—
AMD Ryzen 7 7700 (int8)	1 / 3.8 GHz	11.57	—	—
AMD Ryzen 7 7700 (int8)	4 / 3.8 GHz	5.67	—	—
NVDLA	187.5 MHz	4.59	94438	104732
NVDLA + FI (constant error)	187.5 MHz	4.59	94456	104717
NVDLA + FI (variable error)	187.5 MHz	4.59	96081	106150

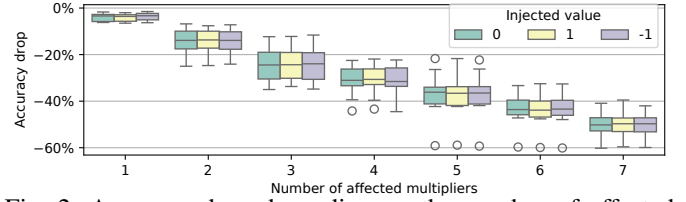


Fig. 2: Accuracy drop depending on the number of affected multipliers.

accelerator are more susceptible to faults. One multiplier was always consistently affected, resulting in a complete alteration of the output value from that multiplier. The heat map of the accuracy drop (Fig. 3) does not reveal a clear pattern, but it does indicate that some multipliers exhibit greater sensitivity to faults independently of the type of error. The most significant drop in accuracy occurred when the last multiplier of the MAC 1 unit is affected.

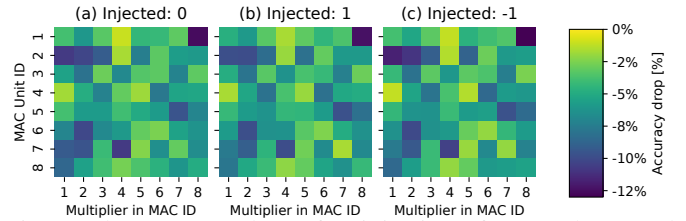


Fig. 3: The accuracy drop after injection of (a) 0, (b) 1, and (c) -1 to one multiplier of the MAC unit.

IV. CONCLUSIONS

The proposed FPGA-based open-source emulation platform supports real-time fault injection into CNN inference accelerators and its analysis. The area overhead associated with FI is negligible. Compared to recent software approaches, the proposed emulator can perform 217 inferences per second for the whole ResNet-18 while the software engine [4] completes 5.8 simulations per second for only two convolutional layers. Our future work will primarily be devoted to improving the emulator in flexibility, configurability, and scalability.

Acknowledgements: This work was supported by the Czech Science Foundation project 24-10990S.

REFERENCES

- [1] M. H. Ahmadilivani, M. Taheri *et al.*, “A systematic literature review on hardware reliability assessment methods for deep neural networks,” *ACM Comput. Surv.*, vol. 56, no. 6, pp. 141:1–141:39, 2024.
- [2] M. H. Ahmadilivani, M. Barbareschi *et al.*, “Special session: Approximation and fault resiliency of dnn accelerators,” in *VTS '23*, 2023.
- [3] Y. He, P. Balaprakash *et al.*, “Fidelity: Efficient resilience analysis framework for deep learning accelerators,” in *MICRO '20*, 2020.
- [4] M. Taheri, M. Daneshtalab *et al.*, “SAFFIRA: a framework for assessing the reliability of systolic-array-based dnn accelerators,” in *DDECS '24*, 2024.
- [5] S. Pappalardo, A. Ruospo *et al.*, “A Fault Injection Framework for AI Hardware Accelerators,” in *LATS '23*, 2023.
- [6] H. Sharma, J. Park *et al.*, “From high-level deep neural models to fpgas,” in *MICRO '16*, 2016.
- [7] Y.-H. Chen, T. Krishna *et al.*, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
- [8] “NVIDIA deep learning accelerator (NVDLA),” 2021. [Online]. Available: <http://nvdla.org>
- [9] L. Wang, (2021) NVDLA xilinx FPGA mapping. [Online]. Available: <https://leiblog.wang/NVDLA-Xilinx-FPGA-Mapping/>
- [10] OPEN AI LAB. Tengine. [Online]. Available: <https://github.com/OAID/Tengine>