Fault-Tolerance Mechanism Analysis on NVDLA-Based Design Using Open Neural Network Compiler and Quantization Calibrator

Shu-Ming Liu[1,2], Luba Tang[1], Ning-Chi Huang[2], Der-Yu Tsai[1], Ming-Xue Yang[2], Kai-Chiang Wu[2]
Skymizer Taiwan Inc.[1] National Chiao Tung University, Hsinchu, Taiwan[2]
{samuel, luba, a127a127}@skymizer.com;{nchuang, msyang.cs08g, kcw}@cs.nctu.edu.tw

Abstract—The NVIDIA Deep Learning Accelerator (NVDLA) provides free intellectual property licensing to IC chip vendors and researchers to build a chip that uses deep neural networks for inference applications. The Open Neural Network Compiler (ONNC) provides an extensible compiler, a quantization calibrator and optimization supports for running DNN models on NVDLA-based SoCs. Even with open-sourced NVDLA and ONNC, conducting the development of an AI chip still brings up many productivity issues in the mass production stage, such as SRAM MBIST (Memory Built-In Self Test) fail, scan-chain fail etc. When applying Fault-Tolerance Mechanism in error-tolerant applications such as image classification by using the AI CNN model, this paper presents a light-weight Fault-Tolerance Mechanism to effectively enhance the robustness of NVDLA-based edge AI chip when encountering internal SRAM stuck fault. Our non-accurate MAC calculation for the whole convolution computation leads to a very promising quality of results compared to the case when an exactly accurate convolution operation is used. The Fault-Tolerance Mechanism analysis and design described in this paper can also apply to the similar fixed-point deep learning accelerator design, and opens new opportunities for research as well as product development.

Index Terms—Deep learning accelerators, Compilers, NVDLA, ONNC, fault tolerant

I. Introduction

Accelerating AI at the edge is critical in enabling new applications in many industries. The NVIDIA Deep Learning Accelerator [1][2] (NVDLA) is a free and open architecture that provides a scalable, configurable and modular design to address the computational demands of neural network inference. Besides ITRI's NVDLA test chip shows in Fig. 1(a), more NVDLA based chips will soon be announced.

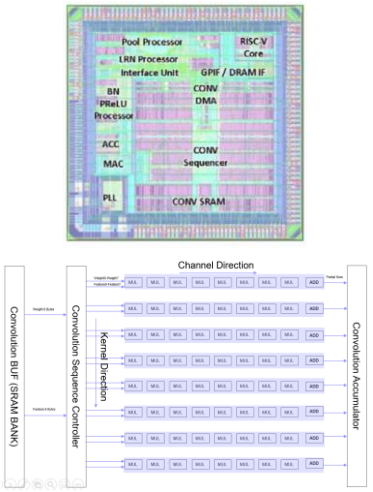

Fig. 1: (a) ITRI's NVDLA Test Chip Layout and (b) NVDLA Convolution MAC Cells Architecture

In addition to the open source hardware, NVDLA also has an open source software stack with a virtual platform, a pre-built Linux kernel, a user-mode driver (UMD), and a kernel-mode driver (KMD) for developers to explore the full-system design. However, NVIDIA released its compiler (NVDLA compiler) only supported limited AI models on Caffe framework. Lack of good support for various hardware configurations in the software stack has become an inevitable barrier for developers and researchers to improve and optimize an NVDLA- based design [3].

ONNC (Open Neural Network Compiler) is a compilation framework designed specifically for proprietary deep learning accelerators. As shows in Fig 2, its software architecture expedites porting ONNC to any DLA design that supports ONNX (Open Neural Network Exchange) operators. ONNC is the first open source compiler available for NVDLA-based hardware designs. Its NVDLA backend can compile a model into an executable NVDLA Loadable file. Integrating ONNC with the NVDLA software stack opens opportunities for developers and researchers to explore the NVDLA-based inference design at system level [3][4].

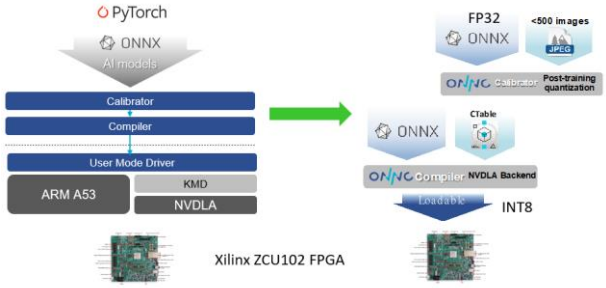

Fig. 2: ONNC NVDLA Compiler and Calibrator

In this paper, we present an ONNC-based software development platform for configurable NVDLA designs. It facilitates the software/hardware co-design process and early-stage software development for NVDLA-based designs. We will discuss the quantization result after calibrating from FP32 pre-trained ONNX model to INT8 deployed models in Section II, describes the accuracy drop trend when injecting stuck faults in Section III, and describe our proposed recovery mechanisms in Section IV, and conclude in Section V.

II. Post-Training Quantization Results

First of all, we selected seven state-of-the-art CNN pre-trained models from PyTorch torchvision package, and convert these models to ONNX format models. We use ONNC to compile these models and generate the calibration CTable. Then, we follow the calibration procedure to generate INT8 NVDLA Loadables. Finally, we use ImageNet ILSVRC2012 test set to check the accuracy on Xilinx ZCU102 FPGA. From the results shown in Fig. 3, we observed the quantization slightly drop on the TOP5 accuracy within 2% for all these selected models. In this paper, we base on this Quantization Accuracy result to check the accuracy drop caused by weight or activation input encountered bit stuck fault, and we also set the accuracy result as the target for evaluating the fault recovery mechanisms.

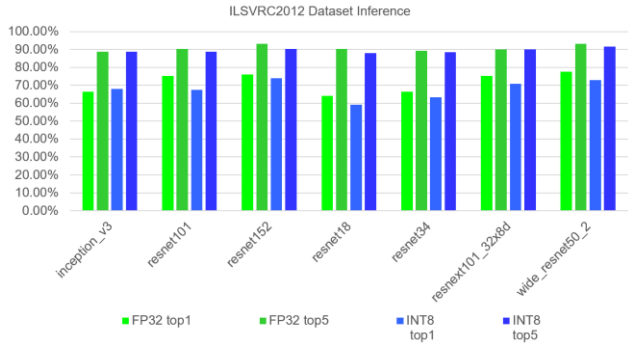

Fig. 3: Comparison of the Accuracy between FP32 and INT8

III. Stuck Fault Injection

In this paper, we adopt NV_SMALL 64MAC configuration version as our CNN model inference platform. First of all, we want to check the fault occurs from CBUF SRAM which impact the convolution computation, so we tie each bit separately, to check the accuracy drop trend when fault occur from low bit to high bit.

To implement this experiment, we injected stuck-at-fault in the NVDLA open source CMAC Verilog code, then synthesized the whole NVDLA RTL to implement the MAC array (shown in Fig. 1(b)) and generated its gate-level netlist. Finally, we wrote out the FPGA bitstream and make as Xilinx ZCU102 SD Card image to bootup the NVDLA Linux platform for inference test.

From the result shown in Fig4, the higher bit stuck fault will cause the accuracy drop significantly than fault occurs on low bit. Fault occur above bit[3] shows almost no accuracy.

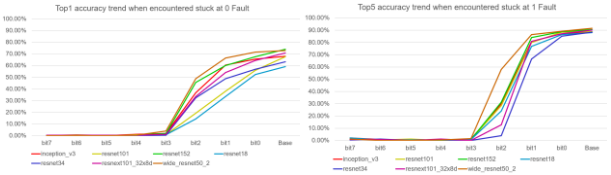

Fig. 4: Top1/Top5 Accuracy Drop Trend When Stuck Fault Occurs on Weight Input

IV. Proposed Fault Recovery Mechanisms

For data-driven applications such as AI image classification model inference, the perfect answer is not always possible to find because of imperfect input data. And it is not necessary to rely on the exactly accurate convolution computation to get the final inference result. Actually some error might be acceptable to lead the final accuracy result [5]. Thus, we propose three light-weight fault recovery mechanisms which doesn't use the same weight or activation input value, then we check the results under applying such recovery mechanisms.

A. Fault Recovery by Applying Adjacent Bit within The Same Input Byte

Since in the application field, in order to fully utilize the quantization bit, our calibrator utilize each bit and assimilate as much information as possible. When the bit is stuck, we'd like to recover its variation by referencing the adjacent normal bit. And the result shows in Fig 5. We found the high-bit stuck faults are significantly recovered by adopting this bit reference policy.

Fig. 5: Top1/Top5 Accuracy Drop Recovery Trend for Weight Input Stuck Fault

B. Fault Recovery by Barrel-Shifter

However, by observing Fig 5, we found inception_v3 shows not recover well comparing with other models. And we know the bit [0] fault did not impact the whole accuracy drop too much. Thus, we can use the bit[0] space to store the higher bit information by using barrel-shifter to avoid the fault bit location. As shown in Fig 6, we assume the bit[4] location is fault, then our proposal is to let the data below bit[4] to be re-arranged and stored by software and let hardware to fetch and compute by using barrel-shifter. In this experiment, we provide a simple method to duplicate bit[1] to originally bit[0] location. We expect the accuracy drop in high bit will reduce to the condition as the fault occur at bit[0].

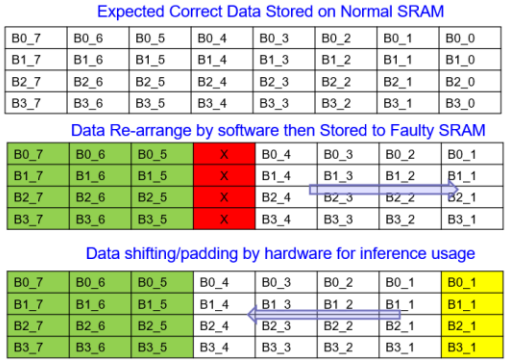

Fig. 6: Fault Recovery by barrel-shifter

C. Fault Recovery by Referencing the Corresponding Bit of Adjacent Channel's MAC Input

This proposal is used for dealing with the image related application, by obtaining the feature map on channel direction which is still image-like information. For certain AI models, we can take the adjacent channel's MAC input to replace the faulty bit if fault occurs on feature/activation data. The result from adopting this mechanism is shown on Fig 7, we can observe the high-bit fault can be significantly recovered.

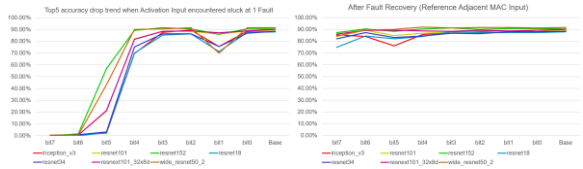

Fig. 7: Accuracy Drop Comparison between Activation Stuck Fault and Recovery Result

V. Conclusion and Future Work

This paper presents three light-weight mechanisms to deal with the input bit stuck fault before entering MAC's computation. We can simply apply the adjacent bit mechanism to effectively recover the accuracy significantly. And on the single fault condition, we can

even apply the barrel shift to let the fault impact just on the lowest bit.

Base on the experiment result, we can easily apply the mechanism on more than one bit fault condition. We provide a guideline to implement the SRAM Built-In Redundancy Analyzer to reconfigure the SRAM backup bit-cell. At the same time, we will evaluate the possibility of adding these mechanisms into SRAM compiler to make the generated SRAM macro more reliable used in CNN AI applications.

REFERENCES

[1] (2017) NVDLA deep learning accelerator. [Online]. Available: http://nvdla.org/

[2] S. Luo, *"Customization of a Deep Learning Accelerator,"* Hsinchu, Taiwan : International Symposium on VLSI Design, Automation and Test (VLSI-DAT), 2019, pp. 1-2

[3] W. F. Lin, D. Y. Tsai, L. Tang, C. T. Hsieh, C. Y. Chou, P. H. Chang, and L. Hsu, *"ONNC: A compilation framework connecting ONNX to proprietary deep learning accelerators,"* Hsinchu, Taiwan : IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS 2019). IEEE, 2019.

[4] (April 2020) ONNC open source project. release v1.3. [Online]. Available: https://github.com/ONNC/onnc

[5] N. Huang, S. Chen and K. Wu, *"Sensor-Based Approximate Adder Design for Accelerating Error-Tolerant and Deep-Learning Applications,"* Florence, Italy: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2019, pp. 692-697