Hierarchical Fault Simulation of Deep Neural Networks on Multi-Core Systems

Masoomeh Karami[1], Mohammad-Hashem Haghbayan[1], Masoumeh Ebrahimi[1,2], Antonio Miele[3], Hannu Tenhunen[1,2], Juha Plosila[1]
[1]Department of Future Technologies, University of Turku, Turku, Finland
[2]Department of Electronics and Embedded Systems, Royal Institute of Technology (KTH), Kista, Sweden
[3]Dip. Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy
Email: {mkaram, mohhag, juplos}@utu.fi, {mebr, hannu}@kth.se, antonio.miele@polimi.it

***Abstract*—In this paper, a hierarchical fault simulation technique for neural networks is proposed, supporting both permanent and temporary faults. In the proposed technique, different levels of hierarchy are used, forming a mixed-level simulation environment. In such an environment, the pre-synthesis behavioral specification of the network and the post-synthesis gate-level model are co-simulated. To accelerate the fault simulation process, faults are injected in the gate-level specification of the selected neurons while the behavioral model in different levels of abstraction is used to simulate the remaining neurons. Further speedup is obtained through event-driven simulation and parallelization. Experimental results confirm the time efficiency of the proposed fault simulation technique.**

***Index Terms*—Fault Simulation, Neural Network, Reliability**

I. Introduction

Deep Neural networks (DNNs) are one of the most promising and widely used applications in different fields such as autonomous driving, aerospace, medical, and robotics [1]. Considering these applications, the neural network computations should satisfy *reliable operation* [1]. To ensure the system reliability, in both design-time and run-time, the occurrence and the impact of faults on the system should be thoroughly investigated in the underlying hardware. To do this, first, an appropriate fault model is defined and *injected* into the hardware, based on the modeling abstraction level, and after that, the behaviour of the hardware with regard to the injected fault is evaluated. This process is called *fault simulation*. Fault simulation of DNNs is a time-consuming process as they involve massive computation of a large number of neurons [2].

Various studies implemented the fault simulation of DNNs, however, they are usually limited to a specific fault model or application. TensorFI is a high level fault injection (FI) framework for applications that are implemented by TensorFlow [3]. TensorFlow describes computations by using a data-flow graph. To inject faults into this application, TensorFI duplicates the data-flow graph and creates a FI graph. The fault model of TensorFI is a high-level fault model. Although TensorFI offers timing benefits, the fault coverage is limited compared to the low-level stuck-at models. BinFI [4] presents a fault injector for soft errors to find the safety-critical bits in ML applications. Ares [5] presents a DNN-specific fault injection framework at the application level and considers memory faults.

In this paper, a novel approach is proposed to improve the fault simulation process for DNNs by taking advantage of their special hierarchical structure. The main idea is to take advantage of the fast simulation time of the higher abstraction level simulation in the cases where low abstraction level simulation is not demanded. The proposed fault simulation technique for DNNs is further accelerated through well-known techniques of parallel and event-driven fault simulation.

II. Hierarchical Fault Simulation of DNNs

The first step of the proposed method is determining different abstraction levels of DNN modeling. Figure 1 shows the levels defined for a simple feed-forward neural network. The top-level view of such a DNN consists of a number of computational *layers*. The lowest abstraction level in our modeling approach is the gate-level, that is compatible with our stuck-at fault model.

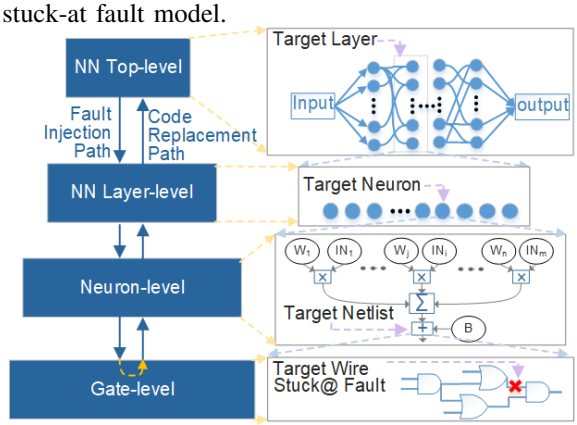

Figure 1: Simulation abstraction levels

Hierarchical fault model: In this model, the fault injection process is a *top-down* process and relies on *high-level effective faults* at each level. The lowest level fault is the stuck-at fault (see Figure 1). The high-level fault model for each neural network layer is defined in terms of cases where neurons' outgoing signals from the layer are faulty. Furthermore, the fault model for a neuron considers cases where the neuron's output changes according to the fault. Defining the high-level fault model removes those low-level faults whose effect is *masked*; they do not *propagate* to higher levels of hierarchy. This process is done during the fault collapsing phase that significantly reduces the number of faults.

Multi-thread fault simulation: To enhance parallelization, multi-thread simulation is used. Since layers are executed in a pipeline manner, the best parallelization performance could be obtained when all threads in the system are allocated to the same layer at the time.

Event-driven fault simulation: The event-driven simulation has been used traditionally for different levels of hierarchy. In DNN fault simulation, the event can be generated through a change in the input signals or by injecting a fault into the system. In our proposed fault simulation technique, a hierarchical event-driven process is proposed where the definition of the *events* in each level of hierarchy is not the same. The main idea here is that, during the simulation in each level of hierarchy, only those parts of the modules that encounter an event in its input will be simulated. Similar to the fault model definition, the event-driven simulation is a top-down process, where an event in the highest possible level of abstraction modeling is considered in fault simulation. The hierarchical event-driven simulation *only* can be applied to the same abstraction level, where concurrent execution would be possible.

Stuck-at faults embedded in gate-level: As mentioned earlier, stuck-at fault should be modeled at the gate-level. Due to the large number of multipliers and adders in DNNs, we consider to reproduce these operators with embedded stuck-at faults operators. The reproduced operator is then called *fault-injectable* operator. There are three main steps to prepare a gate-level implementation for use at a higher level. The **first step** is to describe the operators in the HDL code. **The next step** is to provide a netlist of the operators. For this purpose, the HDL code is synthesized by the synthesis tool (i.e., ISE Xilinx). **Finally**, the netlist of operator converts into the fault-injectable operator at a high-level. Through this process, the stuck-at fault model is embedded on each wire in the netlist.

Overall flow: In the proposed approach, the fault simulation is performed in the highest possible abstraction level. Whenever a fault is injected, the faulty units are replaced with their equivalent lower abstraction level of the target module. Algorithm 1 shows the proposed hierarchical fault simulation. Each level in the algorithm is represented through an integer number starting from the lowest gate-level that takes number zero. The *fault-injection* function injects the faults recursively from the top abstraction level down to gate-level. The *fault-injection* function goes through the abstraction levels of DNN to find the target wire for injecting a fault.

Algorithm 1 The fault-injection function

Inputs: module; level; faults; input;
Body:

```
1:  Initialization phase ();
2:  FS ← select-faults (module, faults); //Faults are grouped into different
    fault sets (FS).
3:  for each event on (FS, modules, level) do
4:      if is_faulty (FS, module, level) then
5:          if level = 0 then
6:              events ← simulate-stuck-at (module, FS, inputs);
7:              return (events);
8:          else
9:              inject-fault (module, level-1, FS);
10:     else
11:         events ← simulate (events);
12:         return (events);
```

III. Experimental Setup and Result

To evaluate the proposed hierarchical fault simulation technique, we adopt the well-known LeNet-5 neural network model, coded in C++ programming language, simulated on the tiny-DNN simulation environment, and based on Ubuntu OS. The fault simulation input is selected from the MNIST benchmark. Figure 2 shows the simulation time of the high-level model versus the gate-level model for a multi-layer perceptron (MLP) neural network. Simulating at a behavioural level significantly improves the simulation time with regard to the number of neurons compared to gate-level simulation.

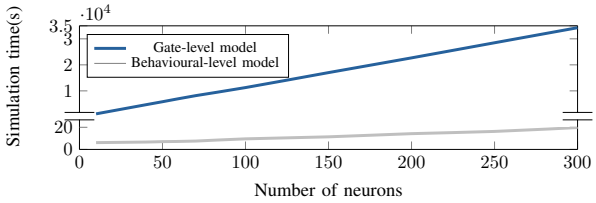

Figure 2: Simulation time of behavioural-level model vs. gate-level model

For gate-level implementation of the neural network, 32-bit fixed-point number representation is used while for modeling the high-level implementation, floating-point variable is used. Figure 3 shows how replacing a behavioral-level implementation of a single neuron with its equivalent gate-level implementation affects the total simulation time. For this purpose, we replace the code for a single neuron in each layer and measure the total simulation time. The total simulation time is around 27 seconds before making any changes, and, for example, it could be increased by around 7 seconds when inserting a gate-level implementation of a single neuron in the first convolutional layer. This result indicates the importance of mixed-level simulation in reducing the simulation time.

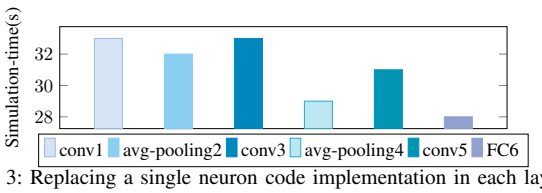

Figure 3: Replacing a single neuron code implementation in each layer.

IV. Conclusion

In this paper, a novel hierarchical fault simulation for deep neural networks (DNNs) is presented. In this approach, the DNN is modeled and can be simulated in different abstraction levels, and for each level of abstraction, a fault model is proposed. Experimental results show that the proposed technique significantly reduces the fault simulation time compared to traditional fault simulation techniques.

References

[1] S. Mittal, "A survey on modeling and improving reliability of dnn algorithms and accelerators," in *Journal of Systems Architecture*, 2020.

[2] A. Chatterjee and L. R. Varshney, "Energy-reliability limits in nanoscale neural networks," in *CISS*, 2017, pp. 1–6.

[3] Z. Chen, N. Narayanan, B. Fang, G. Li, K. Pattabiraman, and N. DeBardeleben, "Tensorfi: A flexible fault injection framework for tensorflow applications," in *arXiv*, 2020.

[4] Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, "Binfi: An efficient fault injector for safety-critical machine learning systems," in *High Performance Computing, Networking, Storage and Analysis*, 2019.

[5] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G. Wei, "Ares: A framework for quantifying the resilience of deep neural networks," in *DAC*, 2018, pp. 1–6.