Spiking Neuron Hardware-Level Fault Modeling

Sarah A El-Sayed, Theofilos Spyrou, Antonios Pavlidis, Engin Afacan, Luis A Camuñas-Mesa, Bernabé Linares-Barranco, Haralampos-G. Stratigopoulos

Spiking Neuron Hardware-Level Fault Modeling

Sarah A. El-Sayed*, Theofilos Spyrou*, Antonios Pavlidis*, Engin Afacan*,
Luis A. Camuñas-Mesa†, Bernabé Linares-Barranco†, Haralampos-G. Stratigopoulos*
*Sorbonne Université, CNRS, LIP6, Paris, France
†Instituto de Microelectrónica de Sevilla (IMSE-CNM), CSIC y Universidad de Sevilla, Sevilla, Spain

***Abstract*—The deployment of Artificial Intelligence (AI) hardware accelerators in a variety of applications, including safety-critical ones, requires assessing their inherent reliability to hardware-level faults and developing cost-effective fault tolerance techniques. This entails performing large-scale fault simulation experiments. However, transistor-level fault simulation is prohibitive and fault simulation should be carried out at a higher abstraction level. In this work, we focus on spiking neural networks (SNNs), and we follow a bottom-up approach starting from transistor-level simulations for developing a neuron behavioral-level fault model that can be readily employed for performing behavioral-level fault simulation of deep SNNs.**

I. Introduction

Artificial Intelligence (AI) has seen several breakthroughs in recent years that paved the way for numerous applications in a wide range of fields. In particular, Deep Neural Networks (DNNs) have shown outstanding capability of solving high-dimensional cognitive tasks autonomously and successfully, in some cases surpassing human performance.

In parallel, there is intense activity in designing dedicated AI hardware accelerators optimized for the computation, memory storage, and networking requirements demanded by AI workloads. Beyond speeding up training and inference on complex DNN models, there is also an incentive to design AI hardware accelerators that can fit inside the resource-constrained Internet of Things (IoT) edge devices, which is to push the execution of AI algorithms from the cloud to the edge. This is motivated by concerns of availability, latency, network bandwidth, and privacy. AI hardware accelerators widely used today include Field-Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs), but efficiency gains of several orders of magnitude can be achieved via Application-Specific Integrated Circuit (ASIC) implementations.

The reliability aspects of AI hardware accelerator architectures need to be carefully addressed given that several targeted applications are safety-critical and mission-critical. In particular, AI hardware accelerators should meet the functional safety standard regulated by the application domain, i.e. ISO 26262 for automotive and IEC 61508 for industrial systems.

An important step towards reliable design is assessing the effect of hardware-level faults on the DNN performance. Hardware-level faults include process variations, manufacturing defects, aging phenomena, soft errors, etc. The assumption that AI hardware accelerators are inherently fault-tolerant since they are modeled after the architecture and operation principles of biological neural networks has been proven false. The over-provisioning, massive spatial redundancy and parallel computing help the network learn around relatively high fault densities; however, a fault occurring after training in the field of application can have detrimental effect on the inference and can seriously jeopardize the application.

The goal is to identify critical fault types and fault locations in the DNN architecture and, subsequently, take action to render the design fault-tolerant. Fault-tolerance strategies can be categorized into proactive strategies, i.e. assessing reliability risks and adjusting the design with the aim to prevent or mitigate those risks, and reactive strategies, i.e. dealing on-the-fly with failures occurring in the field of application by relying on on-chip self-test mechanisms to detect failures and on-chip error correction mechanisms to recover from failures with low-latency, preferably without interrupting the application.

Several fault injection experiments have been reported in the literature for various DNN models running on different types of AI hardware accelerators. Transistor-level fault simulations can be performed only at neuron-level [1] or for small-size networks [2]. In general, performing large-scale fault injection experiments necessitates the use of a fault model of higher abstraction in order to make simulation traceable. This also enables a reliability analysis at higher-level independent of the specific hardware implementation. To this end, fault injection experiments have been performed using behavioral-level faults at the synapse and neuron level [3], static and transient bit flips in data-paths and memories [4]–[6], and stuck-at faults at gate-level [7], at quantizing activations [8], or at the conductance of memristors in memristor crossbars [9].

In this work, we focus on spiking neural networks (SNNs), and in particular on the acceleration of fault simulation using a behavioral-level fault model at the neuron level. Such a behavioral-level fault model was proposed in [3]; however, it was designed intuitively and does not originate from low-level, i.e. transistor-level, fault simulations. Herein, we follow a bottom-up approach for deriving a behavioral-level fault model. We consider a transistor-level design of an Integrate-and-Fire (I&F) spiking neuron and we perform Monte Carlo simulation and structural defect simulation to analyze the effect of process variations and defects, respectively, on the neuron operation. We collect all types of possible faulty behaviors and categorize them to constitute a behavioral-level fault model. Finally, we show how the faulty behaviors can be easily reproduced at behavioral-level.

II. The Spiking neuron

A. Behavioral Model

The I&F model is nowadays one of the most dominant and widely used for describing spiking neurons [10]. It offers

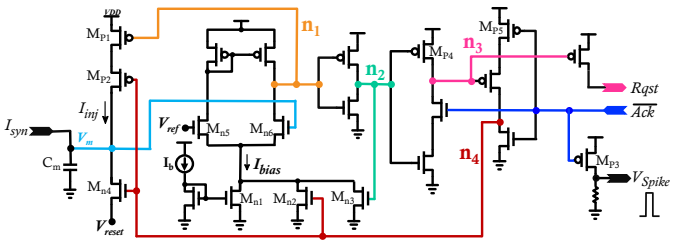


Fig. 1: Neuron circuit.

enough complexity to capture the characteristics of biological neural processing, while being simple enough for analysis and intuitive understanding of its dynamics.

The I&F model explains the dynamics of a neuron through its membrane potential, V_m:

$$C_m \cdot \frac{dV_m}{dt} = I_{syn} + I_{inj}, \tag{1}$$

where C_m is the membrane capacitance, I_{syn} is the post-synaptic current fed to the neuron, and I_{inj} is the current injected into the neuron either externally or through a positive feedback path.

The simplicity of the I&F model comes from the separation of the sub-threshold integration dynamics from the spike generation mechanism. Since a spike is a momentary surge in voltage whose form holds no information, the shape of the spike is not formally stated in the model. Instead, the spike generation behavior is characterized by a firing time t^f and a threshold criterion, i.e., the neuron produces a spike at time t^f when V_m reaches the threshold value, V_{ref}:

$$t^f : V_m(t^f) = V_{ref}. \tag{2}$$

As soon as the neuron fires, the membrane potential is reset to a value $V_{reset} < V_{ref}$:

$$\lim_{t \to t^f; t > t^f} V_m(t) = V_{reset}. \tag{3}$$

For $t > t^f$, the neuron dynamics again follow Eq. (1) until the next time V_m reaches V_{ref}.

B. Transistor-Level Design

Fig. 1 shows the transistor-level design of the I&F neuron used in this work. It is designed in the ams $0.35\mu m$ technology, and was originally part of a neuromorphic cortical-layer processing chip for spike-based processing systems [11].

The neuron takes the input current spikes I_{syn} coming from the synapses, integrates them on capacitor C_m, and fires a spike at the output V_{spike} when the capacitor voltage V_m reaches a certain threshold V_{ref}. The circuit has an extra set of input/output nodes, namely the $\overline{Ack}$ and *Rqst* nodes, which are used by the Address Event Representation (AER) communication protocol.

The main blocks are a comparator and a set of inverters that control the signal flow. During the charging time of the capacitor the circuit is inactive, transistors M_{p1} and M_{n4} are off, and transistor M_{p2} is on. Since the comparator is constantly following V_m and comparing it to V_{ref}, its bias current is kept low through transistor M_{n1} to minimize power consumption. As V_m increases towards V_{ref}, node n_1 starts changing state and switches on two transistors: (i) transistor M_{p1}, which slowly introduces a positive feedback current that accelerates the charging of the capacitor, and (ii) transistor M_{n3} through node n_2, which offers a brief surge in the comparator bias current. Combined, these actions speed up the transition time of the comparator output.

Once the transition is complete, i.e. node n_1 is low and node n_2 is high, node n_3 goes low and an output request signal is sent to the AER communication block by pulling up line *Rqst*. After a few nanoseconds, the AER block acknowledges back the request and the $\overline{Ack}$ input pulls node n_4 up and turns on transistor M_{p3} which produces the output spike of the neuron. Node n_4 has three main effects on the neuron circuit: (i) it turns transistor M_{n2} on to keep the comparator bias current high during the back transitioning, (ii) it turns off transistor M_{p2} which cuts off the positive feedback path to the capacitor, and (iii) it turns transistor M_{n4} on to reset V_m to V_{reset} so the capacitor is able to charge again.

III. Fault Simulation

A. Fault Simulation Framework

To build a taxonomy of neuron faulty behaviors we perform: (a) Monte Carlo simulation with 1000 runs using the technology Process Design Kit (PDK), considering both global and local process variations; (b) structural defect simulation in an automated workflow using the mixed-signal defect simulator Tessent®DefectSim by Mentor®, A Siemens Business [12].

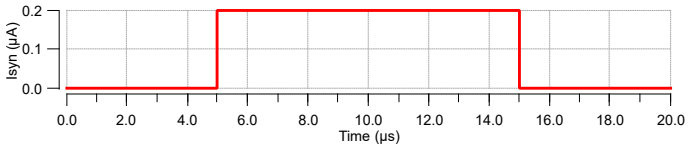
We consider a standard defect model for the transistors that includes stuck-on and stuck-off behaviors. Stuck-on is modeled with a short-circuit across the drain and short terminals implemented with a default small resistance of 10Ω. Stuck-off is modeled with an open-circuit in the gate terminal. Since the simulator cannot handle ideal opens and since a very high series-resistance would have no effect, a gate open is implemented with a weak pull-up or pull-down gate voltage. In particular the gate-to-source voltage is controlled by the drain-to-source voltage with a gain coefficient set to a default value of 0.5 [13]. Finally, for passive elements, i.e., resistors and capacitors, the defect model includes large variations of $\pm 50\%$. For our neuron, the defect model size is $N_{defects} = 46$.

B. Spiking Neuron Faulty Behaviors

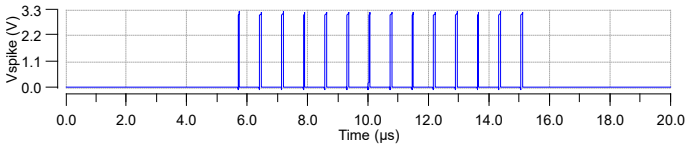
To stimulate the neuron, an input current pulse of $10\mu s$ width was used, shown in Fig. 2a. In a fault-free scenario, the neuron should start spiking at regular intervals after the input stimulus begins and stop spiking once the input stimulus is over, as shown in Fig. 2b.

Simulation experiments revealed different types of faulty behaviours ranging from catastrophic, i.e. the neuron is clearly non-functional, to parametric, i.e. the neuron still produces an output spike train but it shows variations in timing parameters with respect to the nominal response. Catastrophic faulty behaviors were observed in 31 defect simulations, while parametric faulty behaviors were observed across the 1000 Monte Carlo runs and in the rest 15 defect simulations.

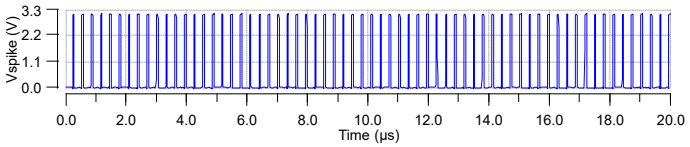
The catastrophic faulty behaviors observed are listed next, along with an example of a root-cause defect. Table I provides



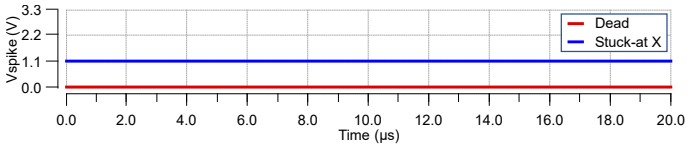
(a) Post-synaptic input current stimulus.



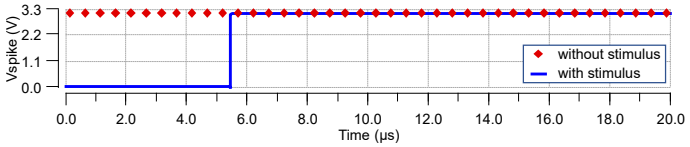
(b) Nominal neuron output.



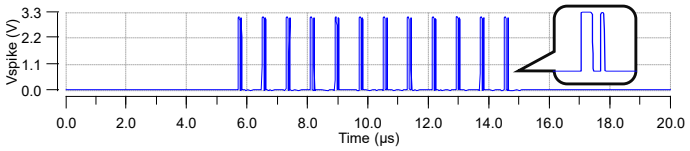
(c) Saturated output caused by a stuck-on M_{p1} transistor.



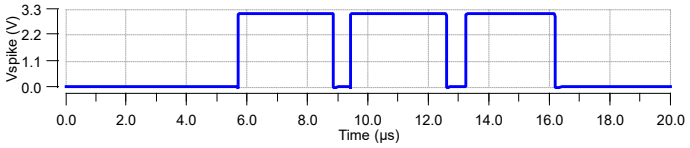
(d) Dead (or stuck-at-0) output caused by a stuck-on M_{n4} transistor and stuck-at-X output caused by a stuck-off M_{p3} transistor.



(e) Stuck-at-1 output caused by a stuck-on M_{n6} transistor (without requiring input stimulus) and by a stuck-off M_{n5} transistor (triggered with input stimulus).



(f) Output with ghost spikes caused by a stuck-off M_{p4} transistor.



(g) Long-duration spikes caused by a stuck-off M_{p5} transistor.

Fig. 2: Examples of catastrophic faulty behaviors.

a summary of catastrophic faulty behaviors and shows the number of defects that produce them.

1) Saturated output: A state where the neuron is constantly firing regardless of the presence of an input stimulus. Fig. 2c shows a saturated output caused by a stuck-on transistor M_{p1}. This defect triggers a constant high feedback current to the capacitor, so the capacitor is always charging even without a current from the synapse.

2) Dead output: A state where the neuron output is stuck-at-0 when it should be spiking. The red curve in Fig. 2d shows a dead output caused by a stuck-on transistor M_{n4}. The capacitor cannot charge since it is constantly held at its reset value and, thereby, the neuron is incapable of spiking.

TABLE I: Catastrophic faulty behaviors resulting from defect simulation.

Catastrophic faulty behavior	Number of defect simulations producing it ($N_{defects} = 46$)	Example neuron response
Saturated	5	Fig. 2c
Dead	12	Fig. 2d
Stuck-at-X	1	Fig. 2d
Stuck-at-1	10	Fig. 2e
Ghost-spike firing	1	Fig. 2f
Long-duration spike firing	2	Fig. 2g

3) Stuck-at-X output: A state where the neuron output gets stuck at an arbitrary DC value between the supply voltage V_{dd} and ground. The blue curve in Fig. 2d shows such a faulty behavior caused by a stuck-off transistor M_{p3}. This defect isolates the neuron output from the $\overline{Ack}$ signal and, in the case of an ideal stuck-off, it turns the output node into a floating node which can settle to any DC value. Given our modeling of stuck-off transistor, the neuron node ends up settling at 1.1V.

4) Stuck-at-1 output: A state where the neuron output gets stuck at V_{dd}. The dotted red curve in Fig. 2e shows this faulty behavior for a stuck-on transistor M_{n6}. In this case, the output gets stuck-at-1 at start-up without requiring an input stimulus. This defect forces node n_1 to be permanently low and, thereby, node n_2 to be permanently high. At start-up, $\overline{Ack}$ is high, thus n_3 enables the *Rqst* signal and $\overline{Ack}$ goes low. When $\overline{Ack}$ goes low, node n_3 is floating but retains its low value, thus $\overline{Ack}$ is permanently set low and the output gets stuck-at-1. The blue curve in Fig. 2e shows another example of a stuck-at-1 behavior caused by a stuck-off transistor M_{n5}, but this time the faulty behavior is triggered only when an input stimulus comes along. This defect cuts off V_{ref} from the comparator input. According to our modeling of stuck-off transistor, the gate voltage of M_{n5}, $V_{G,M_{n5}}$, follows the sum of its drain and source voltages. In the beginning, the capacitor keeps charging and at some point V_m exceeds $V_{G,M_{n5}}$ and the neuron output goes high. At the time of spiking, $V_{G,M_{n5}}$ is lower than V_{reset}, thus node n_1 gets permanently stuck at a low value and the output gets stuck-at-1, as explained above for the stuck-on transistor M_{n6}.

5) Ghost-spike firing: A state where the neuron generates extra spike(s) that is(are) not a result of the membrane potential exceeding the reference voltage. We refer to these spikes as "ghost" spikes. Fig. 2f shows such a faulty behavior caused by a stuck-off transistor M_{p4}. When the neuron spikes, the path from node n_3 to ground gets cut-off. Node n_3 is floating since the defect isolates node n_3 from node n_2. Because of our defect model, node n_3 will eventually be weakly pulled up to V_{dd}. Simulations show that it first gets weakly pulled up to V_{dd} stopping spiking and then again it is weakly pulled down to ground producing a second ghost spike before it is finally stabilized bringing the neuron to its resting state.

6) Long-duration spike firing: A state where the neuron produces spikes of longer duration. Fig. 2g shows such a faulty behavior caused by a stuck-off transistor M_{p5}. When signal $\overline{Ack}$ goes low and the neuron spikes, node n_4 does not go

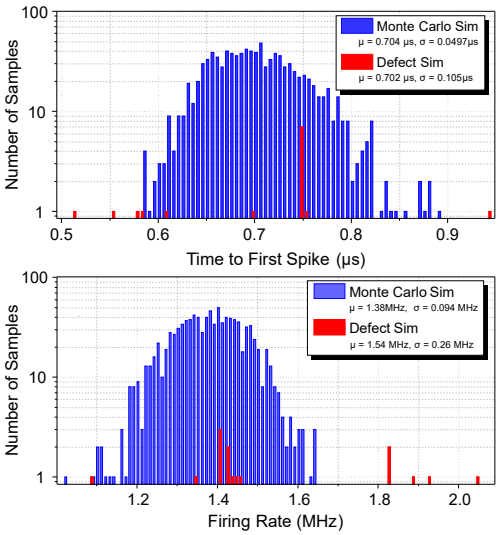


Fig. 3: Histograms of timing variations.

immediately high to instantaneously reset the membrane potential and restart the integration. Instead, node n_4 is initially weakly pulled up to V_{dd} and gradually increases. As a result, the capacitor starts resetting but at a slow rate, thus extending the duration of the output spike.

As for the parametric faulty behaviors, we consider two types of timing parameters, namely the time-to-first-spike and the firing rate. It should be noticed that such timing variations may not be problematic at network-level, i.e. they may be accommodated during training.

Fig. 3 shows the histograms of the timing parameters observed across the 1000 Monte Carlo runs and the 15 defect simulations. As evident from the histograms, the timing parameters are sensitive to process variations. We observe also that certain defects cause significant timing variations, while others have a barely noticeable effect. Two defects that clearly have a significant effect on timing parameters are the $\pm 50\%$ variations in C_m. Other defect examples are stuck-off transistors M_{n2} and M_{n3}. As explained in Section II-B, these transistors form a dynamic biasing circuit to control the comparator transition rate, thus a stuck-off in these transistors ends up significantly altering the timing parameters of the neuron. An example of a defect that has negligible effect on timing parameters is a stuck-off transistor M_{p2}. It comes into play as V_m approaches V_{ref}, increasing the resistance of the positive feedback path and slowing down the acceleration of the capacitor charging. However, this does not seem to affect the spike train timing. In addition, when the neuron spikes, this defect withholds n_4 from immediately cutting-off the positive feedback, although this happens quickly anyways since node n_1 transitions back right away.

IV. Behavioral-Level Fault Model

The behavioral-level fault model for I&F neurons includes the faulty behaviors observed in Section III-B, i.e., the catastrophic faulty behaviors listed in Table I and the variations in the two timing parameters in Fig. 3.

These faulty behaviors can be easily reproduced at behavioral-level. Timing variations can be emulated by directly changing model parameters described in Section II-A, i.e., C_m, V_{ref} or V_{reset}. The dead, stuck-at-X, and stuck-at-1 outputs can be simulated by simply forcing the neuron output to take the respective value. A saturated output is obtained by forcing a high constant input current applied from the start. Long-duration spikes can be produced by delaying the resetting mechanism. Finally, the ghost-spike firing can be recreated by superimposing ghost spikes to the nominal spike train.

V. Conclusions

We performed Monte Carlo analysis and defect simulation for an I&F neuron, then we observed and categorized faulty behaviors to form a behavioral-level fault model. This behavioral-level fault model can be readily used to enable accelerated fault-simulation of deep SNNs, in order to assess reliability properties and develop cost-effective fault-tolerance strategies.

Acknowledgments

This work has been partially funded by the Penta HADES project. T. Spyrou has a fellowship from the Sorbonne Center for Artificial Intelligence (SCAI). L. A. Camuñas-Mesa was funded by the VI PPIT through the Universidad de Sevilla.

References

[1] S. A. El-Sayed et al., "Self-testing analog spiking neuron circuit," in *Proc. International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*, 2019.

[2] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in *Proc. International Symposium on Computer Architecture*, 2012, pp. 356–367.

[3] E. Vatajelu et al., "Special session: Reliability of hardware-implemented spiking neural networks (SNN)," in *Proc. IEEE VLSI Test Symposium*, 2019.

[4] G. Li et al., "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017.

[5] B. Reagen et al., "Ares: A framework for quantifying the resilience of deep neural networks," in *Proc. ACM/ESDA/IEEE Design Automation Conference*, 2018.

[6] M. A. Neggaz et al., "Are CNNs reliable enough for critical applications? an exploratory study," *IEEE Design & Test*, vol. 37, no. 2, pp. 76–83, 2020.

[7] J. J. Zhang et al., "Fault-tolerant systolic array based accelerators for deep neural network execution," *IEEE Design & Test*, vol. 36, no. 5, pp. 44–53, 2019.

[8] G. Gambardella et al., "Efficient error-tolerant quantized neural network accelerators," in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2019.

[9] L. Xia et al., "Stuck-at fault tolerance in RRAM computing systems," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 102–115, 2018.

[10] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological Cybernetics*, vol. 95, no. 1, pp. 1–19, 2006.

[11] R. Serrano-Gotarredona et al., "A neuromorphic cortical-layer microchip for spike-based event processing vision systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 12, pp. 2548–2566, 2006.

[12] S. Sunter et al., "Using mixed-signal defect simulation to close the loop between design and test," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2313–2322, 2016.

[13] B. Esen et al., "Effective DC fault models and testing approach for open defects in analog circuits," in *Proc. IEEE International Test Conference*, 2016, Paper 3.2.