Low-Complexity Algorithmic Test Generation for Neuromorphic Chips

Hsu-Yu Huang[1], Chu-Yun Hsiao[2], Tsung-Te Liu[3] and James Chien-Mo Li[4]
National Taiwan University, Taipei, Taiwan
[1,2,3,4]{r10943182, r12k41017, ttliu, cmli}@ntu.edu.tw

Abstract

Neuromorphic chips are promising hardware implementations for artificial intelligence (AI) applications owing to their low power consumption. However, neuromorphic chips are difficult to test since they have many potential configurations but lack design for testability (DfT). We propose an algorithmic test generation method for neuromorphic chips without DfT, including fault activation and fault propagation. Fault activation differentiates a neuron's good output and faulty output. Fault propagation sensitizes fault effects to differentiate outputs of faulty chips and good chips. On an L-layer Spiking Neural Network (SNN) model, we achieve 100% fault coverage using $O(L)$ test configurations and test patterns under negligible or no weight variation. Our results show that test effectiveness is maintained even with 4-bit weight quantization. We incur no test escape and overkill even under 10% weight variation. Our total test length is over 73K times shorter than previous works.

1 Introduction

As machine learning achieves great performances on tasks such as computer vision and natural language processing, its implementation on edge devices is essential for real-time interaction and privacy requirements [13]. Due to the high computational complexity and data access of machine learning tasks, traditional Von Neumann architectures consume too much power, making them unsuitable for edge devices.

Neuromorphic chips reduce power consumption by using event-driven or in-memory computing schemes [9], so they are more favorable for power-limited edge devices than AI chips based on traditional Von Neumann architectures. However, neuromorphic chips are often implemented in asynchronous circuits [1], making scan *Design-for-Testability (DfT)* hard to apply. Moreover, neuromorphic chips can be configured for various applications, so the tests must be application-independent. Finally, as small device perturbations may degrade the chip performance [11], a high fault coverage is demanded to ensure the reliability of safety-critical missions.

In practice, weights of neuromorphic chips are not accurate. To reduce the computation and memory requirement, weights are often quantized into discrete states [12]. For emerging memory devices such as memristors, weights even suffer from stochastic variation, causing each programmed weight to shift from the accurate value [13]. Therefore, a practical neuromorphic chip test method has to consider both impacts of weight quantization and variation.

Most research in testing neuromorphic chips considered only *application-dependent tests* of AI chips, whose configurations are fixed to specific applications. The corresponding test patterns can be selected from the benchmark dataset [4][7] or from synthetic data [10]. These works only target faults that may significantly degrade the chip performance.

On the other hand, *application-independent tests* target overall fault coverage, so they are more suitable for configurable neuromorphic chips. Previous works [3][2] utilized machine learning and statistics to generate multiple test configurations and patterns for neuromorphic chips, achieving high fault coverage at the cost of high test complexity. Therefore, an effective and efficient application-independent test method for neuromorphic chips is still highly demanded.

We propose to generate *test configurations* and *test patterns* for neuromorphic chips without scan DfT. A test configuration is a set of weights programmed to the neuromorphic chip, and a test pattern is a set of inputs to the neuromorphic chip. The proposed test generation consists of *fault activation* and *fault propagation.* In fault activation, we differentiate a neuron's good output and faulty output. In fault propagation, we sensitize the fault effects to make the outputs of the faulty chip different from the good chip. Inspired by the concept of *C-testable* array-structured chip, which can be tested in constant time [6], we require only a constant number of test configurations and test patterns for each layer, under negligible or no variation.

On an L-layer *Spiking Neural Network (SNN)* model, we demonstrate three advantages. First, we prove to use only $O(L)$ test configurations and test patterns to achieve 100% fault coverage on adopted five fault models, under negligible or no weight variation. Second, experimental results show that our method is effective even with 4-bit weight quantization. Lastly, we incur no test escape and overkill even under 10% weight variation. In total, we achieve an over 73K shorter test length than previous works [3][2].

The paper is organized as follows. Section II introduces backgrounds. Section III details the proposed test generation. Section IV analyzes test complexity. Section V presents experimental results, and section VI concludes this paper.

2 Background

2.1 Spiking Neural Network

In this paper, we adopt *Spiking Neural Network* (SNN) as our test generation model since prominent neuromorphic chips, *e.g.* TrueNorth and Loihi, are implementations of SNN [1]. In our model, neurons in adjacent layers are fully connected by synapses, whose weights are configurable. A *spike* fired by a neuron is transmitted through synapses. In our figures and algorithms, we denote a spike as '1' and no spike as '0'. We adopt *Leaky Integrate-and-Fire* (LIF) [1] mechanism, where each neuron has *membrane potential* (MP). Let $n^{k,i}$ be the i^{th} neuron in layer k and $w^{k,i,j}$ be the weight between

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3655916

$n^{k,i}$ and $n^{k+1,j}$. Let $x^{k,i}$ be the spike from $n^{k,i}$, and $y^{k+1,j}$ be the sum of weighted spikes to $n^{k+1,j}$ (Eq. 1a). When $n^{k+1,j}$ receives a spike, its MP is increased by $y^{k+1,j}$; otherwise, its MP decays over time. Once the neuron's MP surpasses the threshold (θ), it fires a spike and resets its MP (Eq. 1b).

$$y^{k+1,j} = \sum w^{k,i,j} \cdot x^{k,i} \tag{1a}$$

$$x^{k,i} = \begin{cases} 1, & \text{if the MP of } n^{k,i} > \theta \\ 0, & \text{otherwise} \end{cases} \tag{1b}$$

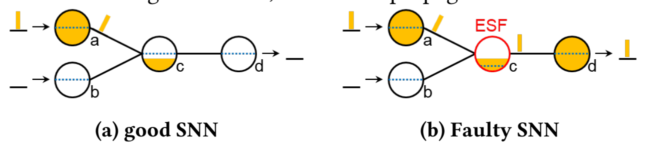
In this paper, the number of neuron layers is denoted as L. The first layer, indexed by 1, is the *input layer*. The last layer, indexed by L, is the *output layer*. Neurons in the input layer are *input neurons*, and those in the output layer are *output neurons*. Neurons apart from input neurons have LIF mechanism and with MP equal to 0 initially. An input neuron fires a spike once it receives a spike from the primary input.

2.2 Fault Models

To generalize our method without hardware dependence, we adopt behavior fault models from [10], including neuron and synapse faults. Neuron faults include *Neuron-Always-Spike Fault (NASF)*, *Easy-to-Spike Fault (ESF)* and *Hard-to-Spike Fault (HSF)*. NASF makes a neuron always spike. ESF and HSF shift a neuron's threshold from θ to $\hat{\theta}$. For ESF, $\hat{\theta} < \theta$, making a neuron easier to spike. For HSF, $\hat{\theta} > \theta$, making a neuron harder to spike. Synapse faults include *stuck-weight fault (SWF)* and *synapse-always-spike fault (SASF)*. SWF makes a synapse's weight stuck at $\hat{\omega}$, and SASF makes a synapse always spike.

3 Proposed Test Generation

Our test generation consists of *fault activation* and *fault propagation*. Fault activation makes a neuron produce different good outputs and faulty outputs. Fault propagation sensitizes the *fault effects* such that the faulty SNN's outputs differ from the good SNN. Fig. 1 is an example of test generation for an SNN with neurons a, b, c and d. The orange sticks are spikes and the height of the orange filling is MP. Fig. 1a is the good SNN, and Fig. 1b is the faulty SNN with ESF on c. To test the ESF, a is *stimulated*, while b is *inhibited* . A stimulated neuron has MP higher than its threshold and fires a spike. An inhibited neuron has MP lower than its threshold and does not fire a spike. The weight from a to c is set low enough that MP of c only surpasses the threshold with ESF, making c stimulated in the faulty SNN but inhibited in the good SNN. Hence, the fault is activated. The weight to d is set high enough that d is stimulated if c has ESF but inhibited if c is good. Hence, the fault is propagated.



(a) good SNN **(b) Faulty SNN**

Figure 1: Example of test generation

3.1 Overview

For NASF and SASF, all faults can be tested together without interference, so they are tested with a single test configuration. For ESF, HSF and SWF, faults in different layers cannot be tested at the same time, so they are tested layer by layer with $O(L)$ test configurations.

Test generation for ESF, HSF and SWF is illustrated in Fig. 2, with layer indices at the bottom. In layer $\ell-1$, some neurons are designated as *pre-target neurons* (in orange) and others are pre-ancillary neurons (in red). In layer ℓ, some neurons are designated as *target neurons* (in green) and others are *ancillary neurons* (in blue). Synapses between pre-target neurons and target neurons are *target synapses* (highlighted). For ESF or HSF, we assume one of the target neurons is faulty. For SWF, we assume one of the target synapses is faulty. The other neurons and synapses are assumed to be fault-free.

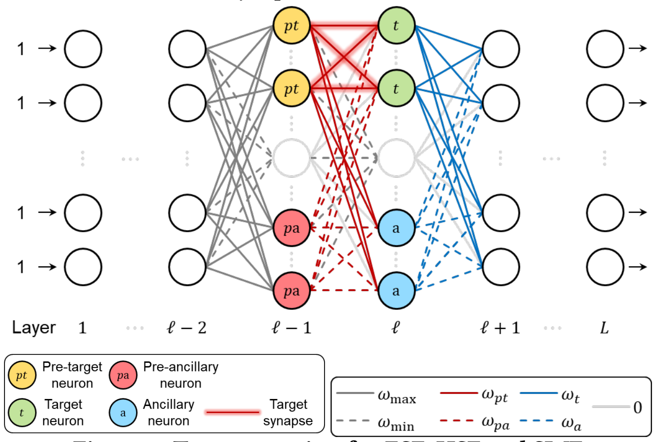


Figure 2: Test generation for ESF, HSF and SWF

All weights between layer 1 and $\ell-2$ and between $\ell+1$ and L, not illustrated for simplicity, are set to the maximum value ($\omega_{\max} \gg \theta$). Hence, all neurons in layer $\ell-2$ are stimulated if an input neuron fires a spike. Setting weights between layers to $\omega_{\max}$ is presented in Algorithm 1, where $\mathcal{N}^k$ is the set of all neurons in layer k. Between layer $\ell-2$ and $\ell-1$, the weights to target or ancillary neurons are set to $\omega_{\max}$, and the others are set to the minimum value ($\omega_{\min} = -\omega_{\max}$). Hence, pre-target and pre-ancillary neurons are stimulated, but the others in layer $\ell-1$ are inhibited. We perform fault activation via pre-target and pre-ancillary neurons (Section 3.2). Similarly, all neurons in layer L are stimulated if a neuron in layer $\ell+1$ fires a spike. We achieve fault propagation via target and ancillary neurons (Section 3.3) to propagate fault effects to layer L.

Algorithm 1 Setting weights between layer *start* and *end* to $\omega_{\max}$

```
1: procedure MAXIMIZE_WEIGHTS($start, end$)
2:     for $k = start$ to $end - 1$ do
3:         for all $n^{k,i}, n^{k+1,j}$ do
4:             $w^{k,i,j} \leftarrow \omega_{\max}$
```

In test generation, we consider two issues on weight inaccuracy. First, we consider weight quantization. We compose a test configuration with at most six levels of quantized weights. The levels of weights required are independent of the model architecture. Second, we consider weight variation. As a neuron's output is determined by whether its MP surpasses its threshold, we make each $y^{k,j}$ differ from θ significantly to tolerate weight variation as much as possible.

3.2 Fault Activation

3.2.1 NASF and SASF

To activate NASF and SASF, all primary inputs are set to 0, and all weights are set to $\omega_{\max}$. No neuron in the good SNN is stimulated. If a neuron has NASF, it is directly stimulated. If a synapse has SASF, it increases a neuron's MP by its weight, $\omega_{\max}$, and stimulates the neuron. Hence, every NASF and SASF is activated with a single test configuration.

3.2.2 ESF, HSF, and SWF

Activation of ESF, HSF and SWF is presented in Algorithm 2. The set of pre-target neurons and that of pre-ancillary neurons are $\mathcal{N}_{pt}^{\ell-1}$ and $\mathcal{N}_{pa}^{\ell-1}$. The set of target neurons and that of ancillary neurons are $\mathcal{N}_{t}^{\ell}$ and $\mathcal{N}_{a}^{\ell}$. Lines 2-9 stimulate all pre-target and pre-ancillary neurons but inhibit the others in layer $\ell-1$. If layer $\ell-1$ is the input layer (line 2), the primary inputs to pre-target and pre-ancillary neurons are set to 1, and the others are set to 0 (lines 3-4). Otherwise (line 5), all primary inputs are set to 1 (line 6). Lines 7-9 set weights between layer 1 and $\ell-1$ as stated in section 3.1. Lines 10-13 set weights between layer $\ell-1$ and ℓ. Among the weights to target or ancillary neurons (line 11), those from pre-target/pre-ancillary neurons are set to ω_{pt}/ω_{pa}, and the others are set to 0. Weights to the other neurons in layer ℓ are set to $\omega_{\min}$ (line 13).

Algorithm 2 Fault Activation for ESF, HSF and SWF

```
1:  procedure FAULT_ACT($\mathcal{N}_{pt}^{\ell-1}, \mathcal{N}_{pa}^{\ell-1}, \mathcal{N}_{t}^{\ell}, \mathcal{N}_{a}^{\ell}, \omega_{pt}, \omega_{pa}$)
2:      if $\ell = 2$ then
3:          $I \leftarrow [s^1, \cdots, s^j, \cdots s^{|\mathcal{N}^1|}]$, where
4:          $s^j \leftarrow \begin{cases} 1, & \text{if } n^{2,j} \in \mathcal{N}_t^\ell \cup \mathcal{N}_a^\ell \\ 0, & \text{otherwise} \end{cases}$
5:      else
6:          $I \leftarrow [1, \cdots, 1]_{|\mathcal{N}^1|}$
7:          MAXIMIZE_WEIGHTS($1, \ell-2$)
8:          for all $n^{\ell-2,i}, n^{\ell-1,j}$ do
9:              $w^{\ell-2,i,j} \leftarrow \begin{cases} \omega_{\max}, & \text{if } n^{\ell-1,j} \in \mathcal{N}_{pt}^{\ell-1} \cup \mathcal{N}_{pa}^{\ell-1} \\ \omega_{\min}, & \text{otherwise} \end{cases}$
10:     for all $n^{\ell-1,i}, n^{\ell,j}$ do
11:         if $n^{\ell,j} \in \mathcal{N}_t^\ell \cup \mathcal{N}_a^\ell$ then
12:             $w^{\ell-1,i,j} \leftarrow \begin{cases} \omega_{pt}, & \text{if } n^{\ell-1,i} \in \mathcal{N}_{pt}^{\ell-1} \\ \omega_{pa}, & \text{if } n^{\ell-1,i} \in \mathcal{N}_{pa}^{\ell-1} \\ 0, & \text{otherwise} \end{cases}$
13:         else $w^{\ell-1,i,j} \leftarrow \omega_{\min}$
```

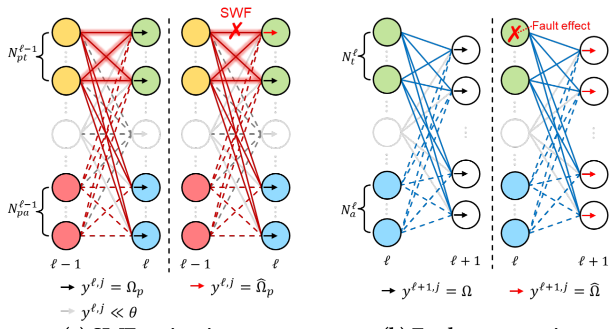
To test ESF or HSF, we designate the first neuron in layer $\ell-1$ as a pre-target neuron and none as a pre-ancillary neuron. To differentiate a target neuron's good output and faulty output, ω_{pt} has to be between θ and $\hat{\theta}$. We set ω_{pt} to $\frac{\theta+\hat{\theta}}{2}$. Hence, MP of each target and ancillary neuron is increased by $\frac{\theta+\hat{\theta}}{2}$, while MP of the other neurons in layer ℓ is increased by a value $\ll \theta$. For ESF, a target neuron is stimulated if it is faulty; otherwise, it is inhibited. For HSF, a target neuron is inhibited if it is faulty; otherwise, it is stimulated. Ancillary neurons are stimulated for HSF. The others in layer ℓ are always inhibited.

To test SWF, the settings are listed in Table 1. Row 1 indicates whether $\hat{\omega}$ is greater than θ. Row 2 indicates whether the setting considers weight variation. Rows 3 and 4 list the number of pre-target and pre-ancillary neurons. ν is the maximum number of stimulated neurons in a layer that have the outputs of each neuron under weight variation equal to those under no variation. The calculation of ν is in section 4.1. Rows 5 and 6 list ω_{pt} and ω_{pa}. The sum of weighted spikes to each neuron in layer ℓ is calculated in Eq. 2 and illustrated in Fig. 3a. MP of a target neuron is increased by $\hat{\Omega}_p$ if one of its connected target synapses is stuck at $\hat{\omega}$; otherwise, it is increased by Ω_p. MP of each ancillary neuron is increased by Ω_p. MP of the others in layer ℓ is increased by a value $\ll \theta$. Rows 7 and 8 of Table 1 list calculated Ω_p and $\hat{\Omega}_p$ of all cases. The greater a neuron's MP differs from its threshold, the more weight variation is tolerated without changing the neuron's outputs. Therefore, we make both Ω_p and $\hat{\Omega}_p$ significantly different from θ considering weight variation.

$$y^{\ell,j} = \begin{cases} \ll \theta, & \text{if } n^{\ell,j} \notin \mathcal{N}_t^\ell \cup \mathcal{N}_a^\ell \\ \hat{\Omega}_p = \Omega_p - \omega_{pt} + \hat{\omega}, & \text{if } w^{\ell-1,i,j} \text{ stuck at } \hat{\omega} \\ \Omega_p, & \text{otherwise} \end{cases} \tag{2}$$
$$\Omega_p = |\mathcal{N}_{pt}^{\ell-1}| \cdot \omega_{pt} + |\mathcal{N}_{pa}^{\ell-1}| \cdot \omega_{pa}$$

Table 1: Settings of fault activation for SWF

	$\hat{\omega} > \theta$		$\hat{\omega} \le \theta$	
Consider variation?	No	Yes	No	Yes
$\lvert\mathcal{N}_{pt}^{\ell-1}\rvert$	$\lvert\mathcal{N}^{\ell-1}\rvert$	$\min\{\lceil \frac{\lvert\mathcal{N}^{\ell-1}\rvert}{4} \rceil, \lceil \frac{\nu}{4} \rceil\}$	$\lceil \frac{\lvert\mathcal{N}^{\ell-1}\rvert}{2} \rceil$	$\min\{\lceil \frac{\lvert\mathcal{N}^{\ell-1}\rvert}{4} \rceil, \lceil \frac{\nu}{4} \rceil\}$
$\lvert\mathcal{N}_{pa}^{\ell-1}\rvert$	0	$2\lvert\mathcal{N}_{pt}^{\ell-1}\rvert - 1$	$\lvert\mathcal{N}_{pt}^{\ell-1}\rvert - 1$	$2\lvert\mathcal{N}_{pt}^{\ell-1}\rvert - 1$
ω_{pt}	0	$-\omega_{\max}$	$\omega_{\max}$	$\omega_{\max}$
ω_{pa}	0	$\frac{\omega_{\max}}{2}$	$-\omega_{\max}$	$\frac{-\omega_{\max}}{2}$
Ω_p	0	$\frac{-\omega_{\max}}{2}$	$\omega_{\max}$	$\frac{\omega_{\max}}{2}$
$\hat{\Omega}_p$	$\hat{\omega}$	$\frac{\omega_{\max}}{2} + \hat{\omega}$	$\hat{\omega}$	$\frac{-\omega_{\max}}{2} + \hat{\omega}$



(a) SWF activation **(b) Fault propagation**

Figure 3: The sum of weighted spikes

Hence, when $\hat{\omega} > \theta$, a target neuron is stimulated if one of its connected target synapses is stuck at $\hat{\omega}$; otherwise, it is inhibited. When $\hat{\omega} \le \theta$, a target neuron is inhibited if one of its connected target synapses is stuck at $\hat{\omega}$; otherwise, it is stimulated. Ancillary neurons are stimulated when $\hat{\omega} \le \theta$. The other neurons in layer ℓ are always inhibited.

3.3 Fault Propagation

In fault activation, we differentiate the good and faulty outputs of neurons in layer ℓ. In fault propagation, we sensitize the differences, *i.e. fault effects*, to the output layer. Propagating NASF and SASF doesn't demand further efforts. Since all weights are set to $\omega_{\max}$, any spike in the faulty SNN is propagated to the output layer. Conversely, ESF, HSF and SWF are further classified into two categories. In one category, *e.g.* ESF and SWF ($\hat{\omega} > \theta$), a target neuron is stimulated only in the faulty SNN. In the other, *e.g.* HSF and SWF ($\hat{\omega} \le \theta$), a target neuron is stimulated only in the good SNN. Faults in the same category are propagated in the same way.

Fault propagation for ESF, HSF and SWF is presented in Algorithm 3. Lines 2-3 set weights between layer ℓ and $\ell+1$. Those from target/ancillary neurons are set to ω_t/ω_a, and the others are set to 0. Line 4 sensitizes the fault effect to the output layer by setting all weights after layer $\ell+1$ to $\omega_{\max}$.

Algorithm 3 Fault propagation for ESF, HSF and SWF

```
1: procedure FAULT_PROP($\ell$, $\mathcal{N}_t^\ell$, $\mathcal{N}_a^\ell$, $\omega_t$, $\omega_a$)
2:     for all $n^{\ell,i}, n^{\ell+1,j}$ do
3:         $w^{\ell,i,j} \leftarrow \begin{cases} \omega_t, \text{ if } n^{\ell,i} \in \mathcal{N}_t^\ell \\ \omega_a, \text{ if } n^{\ell,i} \in \mathcal{N}_a^\ell \\ 0, \text{ otherwise} \end{cases}$
4:     if $\ell + 1 < L$ then MAXIMIZE_WEIGHTS($\ell + 1, L$)
```

Table 2 lists the settings to propagate ESF, HSF and SWF. Row 1 indicates the fault models. Row 2 indicates whether the setting aims to consider weight variation. Rows 3 and 4 list the numbers of target and ancillary neurons. Note that all neurons in layer ℓ are designated as target neurons to test ESF and SWF ($\hat{\omega} > \theta$). Rows 5 and 6 list ω_t and ω_a. The sum of weighted spikes to each neuron in layer $\ell + 1$ is calculated in Eq. 3 and illustrated in Fig. 3b. MP of each neuron in layer $\ell + 1$ is increased by Ω if no target neuron has fault effect; otherwise, it is increased by $\hat{\Omega}$. Rows 7 and 8 of Table 2 list calculated Ω and $\hat{\Omega}$ of all cases. Considering weight variation, we make both Ω and $\hat{\Omega}$ differ from θ significantly for ESF and SWF ($\hat{\omega} \leq \theta$). For ESF and SWF ($\hat{\omega} > \theta$), only $\hat{\Omega}$ needs to differ significantly from θ. As no neuron in layer ℓ is stimulated without a fault, Ω equals 0 regardless of variation.

$$y^{\ell+1,j} = \begin{cases} \Omega, & \text{if fault-free} \\ \hat{\Omega} = \Omega + \omega_t, & \text{if } \exists \text{ ESF/SWF}(\hat{\omega} > \theta) \\ \hat{\Omega} = \Omega - \omega_t, & \text{if } \exists \text{ HSF/SWF}(\hat{\omega} \leq \theta) \end{cases} \tag{3}$$

$$\Omega = \begin{cases} 0, & \text{if ESF/SWF}(\hat{\omega} > \theta) \\ |\mathcal{N}_t^\ell| \cdot \omega_t + |\mathcal{N}_a^\ell| \cdot \omega_a, & \text{if HSF/SWF}(\hat{\omega} \leq \theta) \end{cases}$$

Table 2: Settings of fault propagation

	ESF and SWF ($\hat{\omega} > \theta$)		HSF and SWF ($\hat{\omega} \leq \theta$)	
Consider variation?	No	Yes	No	Yes
$\lvert\mathcal{N}_t^\ell\rvert$	$\lvert\mathcal{N}^\ell\rvert$	$\min\{\lvert\mathcal{N}^\ell\rvert, v\}$	$\lceil \frac{\lvert\mathcal{N}^\ell\rvert}{2} \rceil$	$\min\{\lceil \frac{\lvert\mathcal{N}^\ell\rvert}{4} \rceil, \lceil \frac{v}{4} \rceil\}$
$\lvert\mathcal{N}_a^\ell\rvert$	0		$\lvert\mathcal{N}_t^\ell\rvert - 1$	$2\lvert\mathcal{N}_t^\ell\rvert - 1$
ω_t	$\omega_{\max}$		$\omega_{\max}$	$\omega_{\max}$
ω_a	0		$-\omega_{\max}$	$\frac{-\omega_{\max}}{2}$
Ω	0		$\omega_{\max}$	$\frac{\omega_{\max}}{2}$
$\hat{\Omega}$	$\omega_{\max}$		0	$\frac{-\omega_{\max}}{2}$

Hence, if a target neuron is stimulated by the fault, all neurons in layer $\ell + 1$ are also stimulated. If a target neuron is inhibited by the fault, all neurons in layer $\ell + 1$ are also inhibited. Fault effects are then propagated from layer $\ell + 1$ to the output layer.

3.4 Fault Detection

After fault activation and propagation, the faults can be detected. A fault is detected if the outputs of CUT are different from the good chip. Formally, let the output of a chip be the number of spikes fired from each of its SNN output neurons. For example, if its SNN has 3 output neurons and only the second fires a spike, its output is $[0, 1, 0]$. Given a test pattern x, let $G(x)$ be the good chip output and $B(x)$ be the CUT output. The CUT fails the test if $B(x)$ differs from $G(x)$.

4 Test Complexity

In this section, we formally analyze the test complexity for each fault model. While considering weight variation, we suppose weight errors due to variation are *i.i.d.* random variables of a zero-mean normal distribution with standard deviation σ. It is reasonable to assume $\sigma \ll \omega_{\max}$. We further assume $\sigma \ll |\theta - \hat{\theta}|$ to let fault effects of ESF/HSF be significant enough to alter a neuron's output.

4.1 Calculation of v

v stimulated neurons in a layer cause errors of v weights to accumulate in the MP of each neuron in the next layer. The sum of v weight errors follows a zero-mean normal distribution with standard deviation $\sqrt{v}\sigma$. To make each neuron's outputs unchanged under weight variation, the sum of errors should not alter whether each neuron's MP is higher than θ. To find the value of v feasible for each fault model, the largest possible value of v is derived from Eq. 4, corresponding to the sufficient condition for every SWF to be tested. c is large enough to satisfy the confidence level. For example, let $c = 3$ to satisfy the confidence level of 99.7%. Hence, the upper bound of v is $\propto (\frac{\omega_{\max}}{2\sigma})^2$.

$$c\sqrt{v}\sigma < \min\{|\Omega_p - \theta|, |\hat{\Omega}_p - \theta|, |\Omega - \theta|, |\hat{\Omega} - \theta|\} \approx \frac{\omega_{\max}}{2} \tag{4}$$

4.2 Analysis of Test Complexity

Table 3 lists the exact number of test configurations and test patterns for each fault model, which are proved in the remainder of this section. Column 1 indicating the magnitude of weight variation. While considering weight variation, we regard weight variation as *negligible* if $v > |\mathcal{N}^\ell|$ for all ℓ.

Table 3: No. of test configurations and test patterns

Variation	NASF/ SASF	ESF	HSF	SWF	
				$\hat{\omega} > \theta$	$\hat{\omega} \leq \theta$
No	1	$L - 1$	$2(L - 1)$	$L - 1$	$2 \times 2(L - 1)$
Negligible			$4(L - 1)$	$4(L - 1)$	$4 \times 4(L - 1)$

Lemma 1. *We need only one test configuration and one test pattern for each NASF/SASF, regardless of weight variation.*

Proof. Test generation for each NASF and SASF is presented in Algorithm 4. Since each NASF and SASF can be tested in this way, only one test configuration and test pattern are required. Since $\sigma \ll \omega_{\max}$, the test configuration is still effective regardless of weight variation. □

Algorithm 4 Test generation for NASF and SASF

```
1: $I \leftarrow [0, \cdots, 0]_{|\mathcal{N}^1|}$, save $I$ as a test pattern
2: MAXIMIZE_WEIGHTS(1, $L$)
3: Save $\bigcup w^{k,i,j}$ as a test configuration
```

Lemma 2. *We need $O(L)$ test configurations and test patterns for each ESF/HSF under negligible or no weight variation.*

Proof. Test generation for each ESF or HSF is presented in Algorithm 5. Line 2 iterates ℓ from 2 to L since a fault may occur in all neurons except input neurons. Lines 3-4 designate each neuron in layer ℓ as a target neuron at least once. Lines 5-8 generate a test configuration and a test pattern for the faults on target neurons, executed $\lceil |\mathcal{N}^\ell| / |\mathcal{N}_t^\ell| \rceil$ times to cover every ESF or HSF in layer ℓ. Hence, the total number of test configurations and test patterns for

ESF or HSF is $\lceil |\mathcal{N}^\ell|/|\mathcal{N}_t^\ell| \rceil \cdot (L-1)$. The exact numbers are listed in Table 3 columns 3-4, all of which are $O(L)$. □

Remark 1. If weight variation is not negligible, the number of test configurations and test patterns for each ESF/HSF becomes $O(\sum_{\ell=2}^{L} \lceil |\mathcal{N}^\ell|/v \rceil) = O((\max\{\mathcal{N}^\ell\})(\frac{2\sigma}{\omega_{\max}})^2(L-1))$.

Algorithm 5 Test generation for ESF and HSF

```
1: Select $\mathcal{N}_t^\ell$ and $\mathcal{N}_a^\ell$ from Table 2
2: for $\ell = 2$ to $L$ do
3:     while $\exists n_i^\ell$ not once covered by $\mathcal{N}_t^\ell$ do
4:         Update $\mathcal{N}_t^\ell$ and $\mathcal{N}_a^\ell$
5:         FAULT_ACT($\{n^{\ell-1,1}\}, \emptyset, \mathcal{N}_t^\ell, \mathcal{N}_a^\ell, \frac{\theta+\hat{\theta}}{2}, 0$)
6:         if $\ell < L$ then FAULT_PROP($\ell, \mathcal{N}_t^\ell, \mathcal{N}_a^\ell, \omega_t, \omega_a$)
7:         Save $I$ as a test pattern
8:         Save $\bigcup w^{k,i,j}$ as a test configuration
```

Lemma 3. *We need $O(L)$ test configurations and test patterns for each SWF under negligible or no weight variation.*

Proof. Test generation for each SWF is presented in Algorithm 6. Line 2 iterates ℓ from 2 to L. Lines 3-5 designate each synapse as a target synapse at least once. Lines 6-9 generate a test configuration and a test pattern for the faults on target synapses, executed $\lceil |\mathcal{N}^{\ell-1}|/|\mathcal{N}_{pt}^{\ell-1}| \rceil \cdot \lceil |\mathcal{N}^\ell|/|\mathcal{N}_t^\ell| \rceil$ times to cover every SWF between layer $\ell-1$ and ℓ. Hence, the total number of test configurations and test patterns for SWF is $\lceil |\mathcal{N}^{\ell-1}|/|\mathcal{N}_{pt}^{\ell-1}| \rceil \cdot \lceil |\mathcal{N}^\ell|/|\mathcal{N}_t^\ell| \rceil \cdot (L-1)$. The exact numbers are listed in Table 3 columns 5-6, all of which are $O(L)$. □

Remark 2. If weight variation is not negligible, the number of test configurations and test patterns for each SWF becomes $O(\sum_{\ell=2}^{L} \lceil |\mathcal{N}^{\ell-1}|/v \rceil \cdot \lceil |\mathcal{N}^\ell|/v \rceil) = O((\max\{\mathcal{N}^\ell\})^2(\frac{2\sigma}{\omega_{\max}})^4(L-1))$.

Algorithm 6 Test generation for SWF

```
1: Select $\omega_{pt}, \omega_{pa}, \omega_t, \omega_a, |\mathcal{N}_{pt}^{\ell-1}|, |\mathcal{N}_{pa}^{\ell-1}|, |\mathcal{N}_t^\ell|, |\mathcal{N}_a^\ell|$ from Table 1, 2
2: for $\ell = 2$ to $L$ do
3:     while $\exists n_i^{\ell-1}$ not once covered by $\mathcal{N}_{pt}^{\ell-1}$ do
4:         while $\exists n_j^\ell$ not once covered by $\mathcal{N}_t^\ell$ do
5:             Update $\mathcal{N}_{pt}^{\ell-1}, \mathcal{N}_{pa}^{\ell-1}, \mathcal{N}_t^\ell, \mathcal{N}_a^\ell$
6:             FAULT_ACT($\mathcal{N}_{pt}^{\ell-1}, \mathcal{N}_{pa}^{\ell-1}, \mathcal{N}_t^\ell, \mathcal{N}_a^\ell, \omega_{pa}, \omega_{pt}$)
7:             if $\ell < L$ then FAULT_PROP($\ell, \mathcal{N}_t^\ell, \mathcal{N}_a^\ell, \omega_t, \omega_a$)
8:             Save $I$ as a test pattern
9:             Save $\bigcup w^{k,i,j}$ as a test configuration
```

Theorem. *By Lemma 1 to 3, we need $O(L)$ test configurations and test patterns for each NASF, SASF, ESF, HSF and SWF, under negligible or no weight variation.*

5 Experimental Results

5.1 Settings

We perform all experiments on NVIDIA GeForce RTX 4090 GPU and AMD Ryzen 9 3900X CPU. We use *snntorch* [5] for SNN simulation and *Brevitas* [8] for weight quantization. We compare the results with previous works [3][2] on two SNN models listed in Table 4. One model has 4 layers, and the other has 5 layers. The second row shows the architecture, with the number of neurons in each layer separated by dash. We set parameters as follows: $\theta = 0.5$, $\hat{\theta} = 0.1 \times \theta$ for ESF, $\hat{\theta} = 1.9 \times \theta$ for HSF, $\omega_{\max} = 20 \times \theta$ and $\hat{\omega} = 2 \times \theta$.

Table 4: Model Architecture

Model	4-layer model	5-layer model
Architecture	576-256-32-10	576-256-64-32-10

5.2 Test Generation Results

Table 5 and 6 present test generation results of neuron and synapse faults under no weight variation. Fault models are listed in the first row, with the corresponding number of faults in the second row. Neuron faults occur in all neurons except input neurons. Synapse faults occur in all synapses.

The third and fourth rows are the number of test configurations and test patterns. The fifth row shows the test repetition, which is the times each test pattern is applied. We need to apply each pattern only once owing to our deterministic algorithms. The sixth row shows the test length, calculated by multiplying the number of test patterns by the test repetition. Our method demonstrates an extremely short test length for each fault model despite the enormous number of synapse faults. In total, our test length is 73,826 times shorter than [3] and 157,344 times shorter than [2].

The seventh/eighth rows are the fault coverage without/with 8-bit weight quantization. Fault coverage is the percentage of detected faults. The ninth/tenth rows are the overkill without/with 8-bit weight quantization. Overkill is the percentage of 300 simulated good chips failing the test. Our method is the only one achieving 100% fault coverage on all fault models without overkill, regardless of weight quantization. Note that because we need only six levels of quantized weights, we maintain the same fault coverage and overkill even with 4-bit weight quantization.

5.3 Test Effectiveness under Weight Variation

To simulate weight variation, we modify each weight of the CUT by adding a random variable of a zero-mean normal distribution with standard deviation σ. We simulate an equal number of faulty chips to the total number of faults, injecting each chip with a distinct fault. Test escape is the percentage of the simulated faulty chips passing the test.

Fig. 4 shows test escape and overkill of our method, considering weight variation, along with those of [3][2]. We assume $v > |\mathcal{N}^\ell|$ for all ℓ to observe the range of negligible weight variation for each SNN model. The x-axes are the σ with unit θ. The results of our method, [3] and [2] correspond to the circles, squares and triangles, respectively. We incur 0% test escape and 0% overkill when σ is $\leq 10\%\ \theta$ for both SNN models, indicating that up to 10% θ of weight variation is negligible for our method. Hence, our method is still more effective than previous works under weight variation.

6 Conclusion

We propose an algorithmic test generation method for neuromorphic chips. In fault activation, we differentiate a neuron's good output and faulty output. In fault propagation, we sensitize the fault effects such that the outputs of the faulty chip are different from the good chip. A fault is detected if the outputs of CUT differ from the good chip. On an L-layer SNN, we generate $O(L)$ test configurations and test patterns to achieve 100% fault coverage, under negligible or no weight variation. Experimental results show that we maintain the test effectiveness even with 4-bit weight quantization. With the $O(L)$ test complexity, the proposed method incurs no test escape and overkill even when the standard deviation of weight variation is equal to 10% θ.

Table 5: Test Generation Results of Neuron Faults

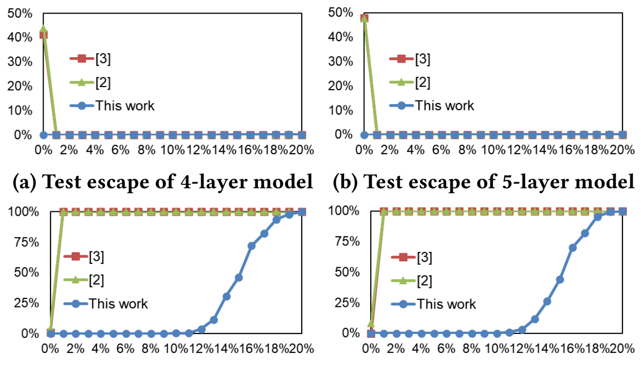
	4-layer model									5-layer model								
	[3]			[2]			Proposed			[3]			[2]			Proposed		
Fault Model	NASF	ESF	HSF	NASF	ESF	HSF	NASF	ESF	HSF	NASF	ESF	HSF	NASF	ESF	HSF	NASF	ESF	HSF
No. of faults	298									362								
No. of test config.	2	10	11	3	14	46	**1**	**3**	**6**	2	2	3	8	24	56	**1**	**4**	**8**
No. of test patterns	23	103	138	13	26	177	**1**	**3**	**6**	37	46	59	58	76	201	**1**	**4**	**8**
Test repetition	659	673	369	1,000	1,000	1000	**1**	**1**	**1**	733	886	488	1,000	1,000	1,000	**1**	**1**	**1**
Test length	15,157	69,319	50,922	13,000	26,000	177,000	**1**	**3**	**6**	27,121	40,756	28,792	58,000	76,000	201,000	**1**	**4**	**8**
Fault coverage[1](%)	99.66	100.0	99.66	100.0	100.0	100.0	**100.0**	**100.0**	**100.0**	96.96	70.27	61.88	98.07	99.45	99.67	**100.0**	**100.0**	**100.0**
Fault coverage[2](%)	99.66	100.0	99.66	84.23	92.62	99.66	**100.0**	**100.0**	**100.0**	99.72	72.16	61.05	98.62	99.17	99.17	**100.0**	**100.0**	**100.0**
Overkill[1](%)	0.00	0.00	0.00	0.00	0.33	1.67	**0.00**	**0.00**	**0.00**	0.00	0.00	0.00	0.00	0.67	1.67	**0.00**	**0.00**	**0.00**
Overkill[2](%)	0.00	0.00	0.00	0.00	1.00	2.00	**0.00**	**0.00**	**0.00**	0.00	0.00	0.00	0.33	1.00	0.67	**0.00**	**0.00**	**0.00**

[1]without / [2]with 8-bit weight quantization

Table 6: Test Generation Results of Synapse Faults

	4-layer model						5-layer model					
	[3]		[2]		Proposed		[3]		[2]		Proposed	
Fault Model	SASF	SWF	SASF	SWF	SASF	SWF	SASF	SWF	SASF	SWF	SASF	SWF
No. of faults	155,968						166,208					
No. of test config.	1	9	46	9	**1**	**3**	1	8	14	26	**1**	**4**
No. of test patterns	181	1,006	1,296	659	**1**	**3**	77	2,026	1,116	1,413	**1**	**4**
Test repetition	1776	707	1,000	1,000	**1**	**1**	1087	494	1,000	1,000	**1**	**1**
Test length	321,456	711,242	1,296,000	659,000	**1**	**3**	83,699	1,000,884	1,116,000	1,413,000	**1**	**4**
Fault coverage[1](%)	16.98	100.0	12.11	99.97	**100.0**	**100.0**	3.99	99.97	4.53	99.97	**100.0**	**100.0**
Fault coverage[2](%)	17.01	100.0	2.51	100.0	**100.0**	**100.0**	4.18	100.0	1.25	100.0	**100.0**	**100.0**
Overkill [1](%)	0.00	1.00	4.33	1.00	**0.00**	**0.00**	0.00	0.33	8.00	0.67	**0.00**	**0.00**
Overkill [2](%)	0.00	0.33	4.33	0.00	**0.00**	**0.00**	0.00	0.00	15.3	0.67	**0.00**	**0.00**

[1]without / [2]with 8-bit weight quantization



(a) Test escape of 4-layer model **(b) Test escape of 5-layer model**

(c) Overkill of 4-layer model **(d) Overkill of 5-layer model**

Figure 4: Test escape and overkill under weight variation

Acknowledgments

This research is supported by the National Science and Technology Council of Taiwan (NSTC113-2119-M-002-024).

References

[1] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. 2019. Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 15, 2 (2019), 1–35.

[2] Xin-Ping Chen and James Chien-Mo Li. 2023. *Test Compression for Neuromorphic Chips*. Master's thesis. National Taiwan University. https://doi.org/10.6342/NTU202303828

[3] I-Wei Chiu, Xin-Ping Chen, Jennifer Shueh-Inn Hu, and James Chien-Mo Li. 2022. Automatic Test Configuration and Pattern Generation (ATCPG) for Neuromorphic Chips. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–7.

[4] Sarah A El-Sayed, Theofilos Spyrou, Luis A Camuñas-Mesa, and Haralampos-G Stratigopoulos. 2022. Compact functional testing for neuromorphic computing circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[5] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. 2023. Training spiking neural networks using lessons from deep learning. *Proc. IEEE* (2023).

[6] Arthur D Friedman. 1973. Easily testable iterative systems. *IEEE Trans. Comput.* 100, 12 (1973), 1061–1064.

[7] Anteneh Gebregiorgis and Mehdi B Tahoori. 2019. Testing of neuromorphic circuits: Structural vs functional. In *2019 IEEE International Test Conference (ITC)*. IEEE, 1–10.

[8] Alessandro Pappalardo. 2023. Xilinx/brevitas. https://doi.org/10.5281/zenodo.3333552

[9] Fei Su, Chunsheng Liu, and Haralampos-G Stratigopoulos. 2023. Testability and dependability of AI hardware: Survey, trends, challenges, and perspectives. *IEEE Design & Test* 40, 2 (2023), 8–58.

[10] Hsiao-Yin Tseng, I-Wei Chiu, Mu-Ting Wu, and James Chien-Mo Li. 2021. Machine learning-based test pattern generation for neuromorphic chips. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–7.

[11] Zheyu Yan, Xiaobo Sharon Hu, and Yiyu Shi. 2022. Computing-in-memory neural network accelerators for safety-critical systems: Can small device variations be disastrous?. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[12] Ussama Zahid, Giulio Gambardella, Nicholas J Fraser, Michaela Blott, and Kees Vissers. 2020. FAT: Training neural networks for reliable inference under hardware faults. In *2020 IEEE International Test Conference (ITC)*. IEEE, 1–10.

[13] Wenqiang Zhang, Bin Gao, Jianshi Tang, Peng Yao, Shimeng Yu, Meng-Fan Chang, Hoi-Jun Yoo, He Qian, and Huaqiang Wu. 2020. Neuro-inspired computing chips. *Nature electronics* 3, 7 (2020), 371–382.