Hardware Trojans in eNVM Neuromorphic Devices

Lingxi Wu, Rahul Sreekumar (joint 1st author), Rasool Sharifi, Mircea Stan, Kevin Skadron, Ashish Venkat

University of Virginia
EE, CS Dept
April 2023
Emerging non-volatile memory (eNVM) is a memory technology that stores bits/values in the form of conductance.

eNVM-based accelerators that mimic biological neuron computations (neuromorphic) in the analog domain are gaining considerable traction for DNN acceleration.

But their security implications remain largely unexplored.

Designed and manufactured in a decentralized way.

Motivate the supply-chain attack: stealthy injection of hardware Trojans.
Security Threat: Bad Actors in the Supply Chain

- Setting up an end-to-end IC factory is expensive and time-consuming ($20 billion in 2020)
- IC supply chain is distributed
- Frequent algorithm update and tuning
- IC supply chain is susceptible to hardware Trojan insertion
  - Tainted 3rd party IP blocks or CAD tools
  - Rogue engineers insert Trojans to RTL
  - Malicious foundry tamper with the mask layout …
Neuromorphic Chips Deployment & Use Case

Development stage:
- SW algorithm/model selection
- HW teams implement model structure (VGG, ResNet, etc.)
- Manufacturing (fab)

Post-deployment stage:
- Transfer to clients
- Individual users, cloud provider, ML service providers (ML-as-a-service e.g., BigML)
- Train the model (configure weights)
- Prediction using the model

Fig. Neuromorphic devices product life cycles & parties involved
Neuromorphic Chips Deployment & Use Case w/ Vulnerability (Threat Model)

- Vulnerable to Trojan insertion at the design and fab stage
- Colluding malicious entities: embed + activate Trojan
- Or simply publish Trojan code

Either way, even if the synaptic weights are programmed into the device by a trusted entity, neuromorphic chips would still remain vulnerable to a Trojan placed in the supply chain.

Fig. Trojan-infected neuromorphic devices product life cycles & parties involved
Vulnerabilities in the neuromorphic systems?

- Identify **exploitable vulnerabilities** in the eNVM-based neuromorphic devices
- **Opportunities** for the supply chain attackers

Need to understand neuromorphic architecture
Exploitable Vulnerability – Analog Current

**Fig (d):** one eNVM cell that holds weight in the form of conductance/resistance

**Fig (a):** mapping of one MLP layer to eNVM cell array → incoming weights \( W_{0,1}, W_{1,1}, W_{2,1} \) are coded as conductance \( G_{0,1}, G_{1,1}, G_{2,1} \)

Weighted sum produces an analog current & **Strength** of the current depends on the weights → larger weights = higher conductance level = larger current
Integrate-and-fire ADC generate spike train → larger current = more spikes = transient power switching activity fluctuate

Popular design due to energy efficiency
ADCs consumes 80% power and are time-shared

Key insight: larger weights result in more intensive transient power switching activity → power side-channel
Model extraction: stealing synaptic weights

We devise an attack scheme that leaks model parameters, i.e., neural network synaptic weights from a neuromorphic system through a power side-channel.

Why steal the weights?

1. Synaptic weights are the core IP
2. Stealing weights is increasingly more economical
   a. Needs a large set of high quality labeled data
   b. Needs a proprietary training algorithm
   c. Slow
Challenges

Synaptic arrays compute weighted sum in parallel

- Each ADC receives current produced by multiple eNVM cells
- Multiple ADCs work concurrently

Needs to attribute the power signal to a particular eNVM cell (weight)

Unknown hyper parameters:

- Number of layers
- Size of each layer

Colluding adversaries can insert a hardware Trojan in the supply chain

HW Trojan selectively suppresses the ADC

Current related to input $\rightarrow$ input image with malicious content can trigger the Trojan and select target row of eNVM cells to activate

Malicious entities along the supply chain has the knowledge of the model structure

Well-known NN models are documented (VGG, ResNet, etc.)
Attack Procedure & Results

Program **Activation Code:**
- Assigning unique activation code for each trojan

**Attack Phase 1:**
**Trojan Embedding**

- **Neuromorphic Chip Production**
- Untrusted parties
  - Hardware IP vendors
  - Design teams
  - Foundry
- **Chip Delivery**

**Malicious party embeds Trojan**
- Distribute Trojan code (trigger)

**Offline Characterization:**
- Build a reference FFT trace library
- One weight → one trace

**Trojan Embedding Phase:**
- Malicious party embeds Trojan
- Distribute Trojan code (trigger)
Attack Procedure & Results

Online Weight Stealing:
- Trigger Trojan
- Collect power trace
- Signal analysis (FFT)
- Deduce weights by searching FFT pattern against a library

Key insight: Different spiking outputs $\rightarrow$ unique FFT signature

Trojan Activation Phase:
- Malicious party triggers Trojan
- Create a power-side channel
Attack Results

- Recover more than **90%** of the weights
- Attack improves with ADC resolution → **> 30% ↑ recovery** for 2-bit↑ in resolution
- Recovered Accuracy is comparable (~ ±2.65% Δ) even for low precision ADC's.

Trojan Stealth analysis highlights:

- **Noise** contribution from Trojan << overall noise floor (~150μV²/Hz)
- % **Area overhead** is a scalable knob. (0.28 - .87% total overhead)
- **False triggering** of trojan << 1 in 10⁴ input sequences
Backup Slides: Trigger + Payload Design

**Fig.** Trojan trigger module and payload circuit

Trigger ckt. determines payload operating condition

**Trigger (HIGH):** payload active (neuron suppressed)

**Trigger (LOW):** payload inactive (neuron active)

Trigger state value $\Rightarrow$ unique pixel combination

Custom PUN/PDN circuit preferred over standard cells due to area constraints

**Key insight:** 
`# of input pixel combination` for unique Trigger code correlates to probability of false triggering
Backup Slides: Trace Denoising + benefits of FFT

Differential signal denoising:
- lowers noise floor
- improves detection of frequency signature

Dominant signature components: Clock routing/ DC power costs

(spike rate in trace A > B) Key insight: Intensity/Spike rate of neuron has direct impact on amplitude and frequency component within FFT signature
Backup Slides: Trojan Stealth Performance

![Figure 1: Spectral comparison depicting noise contribution by embedded Trojan](image1)

![Figure 2: Area overhead as a function of trojan bits (pixel combination) for trigger](image2)

**Figure** Spectral comparison depicting noise contribution by embedded Trojan

**Figure** Area overhead as a function of trojan bits (pixel combination) for trigger

**Trojan Stealth analysis highlights:**

- **Noise** contribution from Trojan $\ll$ overall noise floor ($\sim 150 \mu V^2/Hz$)
- **% Area overhead** is a scalable knob. (0.28 - 0.87% total overhead)
- **False triggering** of trojan $\ll 1$ in $10^4$ input sequences

![Figure 3: P_leak as a function of trojan bits](image3)
Backup Slides: MNCS Architecture + Attack strategy algorithm

**Algorithm 1: High-level phase two attack procedure.**

```plaintext
input: Trojan activation images - trojan_imgs
Row activation images for input layer - act_imgs
output: Recovered weights - weights
// Weights are recovered layer-by-layer
for each synaptic core do
  // Iteratively suppress all ADCs except one
  for each ADC of synaptic_core do
    act_trojan(trojan_imgs[synaptic_core_idx][ADC_idx]);
    // Activate synaptic core row-by-row
    for each row in synaptic_core do
      if first synaptic_core then
        // Row activation using images
        act_rows(act_imgs[ADC_idx]);
      else
        // Row activation leveraging Trojan
        act_rows(synaptic_core_idx, ADC_idx);
      end
      pwr_trace = get_power_trace();
      // Multiple columns per ADC
      for each SL connected to ADC do
        fft = FFT(pwr_trace[SL idx]);
        conductance = search_ref_lib(fft);
        weight = cond_to_weight(conductance);
        weights.add(weight);
      end
      // Reset Trojan trigger
      deact_trojan(trojan_imgs[synaptic_core_idx][ADC_idx]);
    end
  end
end
```

**TABLE I: MNCS Architectural Parameters**

<table>
<thead>
<tr>
<th>Synaptic Core One (400 rows x 100 columns)</th>
<th>Synaptic Core Two (100 rows x 10 columns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Component</td>
</tr>
<tr>
<td>Power (W)</td>
<td>Power (W)</td>
</tr>
<tr>
<td>Latency (s)</td>
<td>Latency (s)</td>
</tr>
<tr>
<td>Area (m²)</td>
<td>Area (m²)</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Synaptic array</td>
<td>Synaptic array</td>
</tr>
<tr>
<td>6.38e-4</td>
<td>9.59e-5</td>
</tr>
<tr>
<td>7.14e-9</td>
<td>7.11e-9</td>
</tr>
<tr>
<td>8.08e-9</td>
<td>2.76e-9</td>
</tr>
<tr>
<td>WL decoder</td>
<td>WL decoder</td>
</tr>
<tr>
<td>1.11e-4</td>
<td>8.91e-6</td>
</tr>
<tr>
<td>6.21e-10</td>
<td>1.58e-10</td>
</tr>
<tr>
<td>1.12e-9</td>
<td>2.75e-10</td>
</tr>
<tr>
<td>SL switch matrix</td>
<td>SL switch matrix</td>
</tr>
<tr>
<td>5.07e-6</td>
<td>5.07e-7</td>
</tr>
<tr>
<td>5.56e-9</td>
<td>1.60e-9</td>
</tr>
<tr>
<td>2.99e-10</td>
<td>4.47e-11</td>
</tr>
<tr>
<td>BL switch matrix</td>
<td>BL switch matrix</td>
</tr>
<tr>
<td>2.01e-5</td>
<td>5.07e-6</td>
</tr>
<tr>
<td>2.59e-10</td>
<td>2.53e-10</td>
</tr>
<tr>
<td>1.49e-10</td>
<td>3.49e-10</td>
</tr>
<tr>
<td>Mux and Mux decoder</td>
<td>Mux and Mux decoder</td>
</tr>
<tr>
<td>5.64e-7</td>
<td>5.64e-7</td>
</tr>
<tr>
<td>4.84e-11</td>
<td>4.45e-11</td>
</tr>
<tr>
<td>2.20e-10</td>
<td>1.01e-10</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC</td>
</tr>
<tr>
<td>72.51e-6</td>
<td>14.27e-6</td>
</tr>
<tr>
<td>1.3e-9</td>
<td>1.3e-9</td>
</tr>
<tr>
<td>1.62e-9</td>
<td>0.23e-9</td>
</tr>
<tr>
<td>Others</td>
<td>Others</td>
</tr>
<tr>
<td>4.62e-6</td>
<td>6.61e-7</td>
</tr>
<tr>
<td>5.29e-10</td>
<td>5.29e-10</td>
</tr>
<tr>
<td>2.75e-10</td>
<td>3.48e-11</td>
</tr>
</tbody>
</table>

**TABLE II: Memristor Device Characteristics**

<table>
<thead>
<tr>
<th>Device Type</th>
<th>EpiRAM (Ag:SiGe)</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Conductance States</td>
<td>64</td>
</tr>
<tr>
<td>Nonlinearity</td>
<td>0.5 − 0.5</td>
</tr>
<tr>
<td>Rxx (kΩ)</td>
<td>81 kΩ</td>
</tr>
<tr>
<td>ON/OFF ratio</td>
<td>50:2</td>
</tr>
<tr>
<td>Weight increase pulse</td>
<td>5V/75μs</td>
</tr>
<tr>
<td>Weight decrease pulse</td>
<td>-3V/75μs</td>
</tr>
<tr>
<td>Cycle-to-cycle variation</td>
<td>2%</td>
</tr>
</tbody>
</table>