Fault Tolerance in Neuromorphic Computing Systems

Yu Wang¹, Mengyun Liu², Krishnendu Chakrabarty², Lixue Xia¹

¹Tsinghua University, China
²Duke University, U.S.A.

Email: yu-wang@tsinghua.edu.cn
http://nicsefc.ee.tsinghua.edu.cn
Outline

• Background and Introduction
  – Convolutional Neural Network Accelerators
  – RRAM and RRAM-based Computing System
  – RRAM Fault Models

• Fault-tolerance for Device-level Faults
  – Stuck-At-Fault (SAF)
  – Limited Endurance
  – State Drifting Problem and Resistance Variation
  – Non-linear Resistance Distribution

• Fault-tolerance for Circuit-level Faults
  – Wire Resistance and IR-drop

• Fault-tolerance for System-level Faults
  – Unbalanced Writing
Outline

• Background and Introduction
  – Convolutional Neural Network Accelerators
  – RRAM and RRAM-based Computing System
  – RRAM Fault Models

• Fault-tolerance for Device-level Faults
  – Stuck-At-Fault (SAF)
  – Limited Endurance
  – State Drifting Problem and Resistance Variation
  – Non-linear Resistance Distribution

• Fault-tolerance for Circuit-level Faults
  – Wire Resistance and IR-drop

• Fault-tolerance for System-level Faults
  – Unbalanced Writing
Neural Network Accelerators

• The energy efficiency of existing Neural Network accelerators is limited at ~10TOPs/W (i.e. 0.1pJ/OP)
  – The gap between brain (500TOPs/W) and accelerators is more than 50x

• Large data movements cause high energy consumption in CNN
  – The data transfer in GPU consumes 2 orders of magnitude more energy than a floating-point operation [Han S, et al. ISCA’ 16]

RRAM-based Computing System (RCS)

- **RRAM and Processing-In-Memory** provide alternative solutions to realize better implementation of CNN.

\[ V_{oj} = r \cdot \sum_k (V_{ik} \cdot g_{kj}), \quad g_{kj} = \frac{1}{M_{kj}} \]

**O(n^2) \rightarrow O(n^0)**
## Representative RCS Design

<table>
<thead>
<tr>
<th>Device</th>
<th>RRAM Precision</th>
<th>Function</th>
<th>Performance</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISAAC</td>
<td>2 bits</td>
<td>CNN Inference</td>
<td><strong>14.8x</strong> Throughput and <strong>4.4x</strong> Energy Efficiency (Compared with DaDianNao)</td>
<td>A. Shafiee, et al. ISCA’16</td>
</tr>
<tr>
<td>PRIME</td>
<td>4 bits</td>
<td>CNN Inference</td>
<td><strong>2360x</strong> Speedup and <strong>895x</strong> Energy Efficiency (Compared with DianNao)</td>
<td>P. Chi, et al. ISCA’16</td>
</tr>
<tr>
<td>PipeLayer</td>
<td>5~6 bits</td>
<td>CNN Inference and Training</td>
<td><strong>42.45x</strong> Speedup and <strong>7.17x</strong> Energy Efficiency (Compared with GPU)</td>
<td>L. Song, et al. HPCA’17</td>
</tr>
<tr>
<td>TIME</td>
<td>4 bits</td>
<td>CNN/DRL Inference and Training</td>
<td>CNN: <strong>1.3x</strong> Speedup and <strong>19.6x</strong> Energy Efficiency (Compared with DaDianNao) DRL: <strong>126x</strong> Energy Efficiency (Compared with GPU)</td>
<td>M. Cheng, et al. TCAD’18</td>
</tr>
<tr>
<td>NTHU Chip</td>
<td>1 bit</td>
<td>Binary DNN/CNN</td>
<td>CNN: 14.8ns/MAC FCN: 15.6ns/MAC (LeNet-5 @ MNSIM)</td>
<td>W. Chen, et al. ISSCC’18</td>
</tr>
</tbody>
</table>
RCS Faults

• The general *RCS faults* mean *RRAM device faults* and other *non-ideal factors* which will cause computation deviation

<table>
<thead>
<tr>
<th>Device-Level</th>
<th>Circuit-Level</th>
<th>System-Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Stuck-At-Fault (SAF)</td>
<td>• Wire Resistance</td>
<td>• Unbalanced Writing</td>
</tr>
<tr>
<td>• Limited Endurance</td>
<td>• IR-drop</td>
<td></td>
</tr>
<tr>
<td>• State Drift Problem</td>
<td>• Sneak Path</td>
<td></td>
</tr>
<tr>
<td>• Resistance Variation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Non-linear Resistance Distribution</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• RCS faults make the CNN computing inaccurate and the system unreliable

<table>
<thead>
<tr>
<th>Yield</th>
<th>Ideal</th>
<th>95%</th>
<th>90%</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>97.8%</td>
<td>26.7~</td>
<td>15.5~</td>
<td>10.6~</td>
</tr>
<tr>
<td></td>
<td>60.4%</td>
<td>38.6%</td>
<td>28.0%</td>
<td></td>
</tr>
<tr>
<td>Reduction</td>
<td>-</td>
<td>&gt;37%</td>
<td>&gt;59%</td>
<td>&gt;69%</td>
</tr>
</tbody>
</table>
Fault Tolerance in RCS

- According to the type of RCS faults, we have proposed corresponding fault tolerance methods to rescue the computation accuracy and reliability of RCS.
Outline

- Background and Introduction
  - Convolutional Neural Network Accelerators
  - RRAM and RRAM-based Computing System
  - RRAM Fault Models

- Fault-tolerance for Device-level Faults
  - Stuck-At-Fault (SAF)
  - Limited Endurance
  - State Drifting Problem and Resistance Variation
  - Non-linear Resistance Distribution

- Fault-tolerance for Circuit-level Faults
  - Wire Resistance and IR-drop

- Fault-tolerance for System-level Faults
  - Unbalanced Writing
Overview of RRAM Device Faults

- Hard / Soft: Resistance Unchangeable / Changeable
- Static / Dynamic: Generated during Fabrication / Read-and-Write

- Endurance Limitation
  - $10^6 \sim 10^8$ for Multi-level Device

- Fabrication Defect
  - >30% for RRAM Array

- Dynamic
  - Write Variation
  - Write Disturbance
  - Read Disturbance
  - Fabrication Variation
  - Non-linear Distribution

- Static
  - Hard
  - Soft

- Dynamic
  - Up to 30% Resistance Deviation
  - Resistance Drifting
  - Appear in Multi-level Cell
Overview of RRAM Device Faults

- **Fault-tolerance methods** for RRAM device faults

**Hard**
- Our Solution: DAC 17-1
- TCAD 19-1

**Fabrication Defect**
- Our Solution 1: ASPDAC 17
- JETCAS 18

**Static**

**Endurance Limitation**
- Our Solution 2: ITC 18

**Dynamic**
- Write Variation
- Write Disturbance
- Read Disturbance

**Soft**
- Our Solution: DATE 14
- Our Solution: DATE 18

**Fabrication Variation**

**Non-linear Distribution**
Device Fault 1: SAF

- Stuck-At-Faults (SAFs): the resistance states cannot be changed
- SAFs cause significant accuracy loss of neuromorphic computing
  - The recognition accuracy of the MNIST drops to 17.75% @ 20% SAFs

<table>
<thead>
<tr>
<th>Yield</th>
<th>Ideal</th>
<th>95%</th>
<th>90%</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>97.8%</td>
<td>26.7~</td>
<td>15.5~</td>
<td>10.6~</td>
</tr>
<tr>
<td></td>
<td>60.4%</td>
<td>38.6%</td>
<td>28.0%</td>
<td></td>
</tr>
<tr>
<td>Reduction</td>
<td>-</td>
<td>&gt;37%</td>
<td>&gt;59%</td>
<td>&gt;69%</td>
</tr>
</tbody>
</table>

Stuck-at-0 (black) [C. Chen, IEEE Trans. Computers 2015]
Fault Tolerance Method for SAF

- SAF tolerant framework contains two parts:
  - **Fault detection**: identify the SAF position
  - **Fault tolerance**: restore the accuracy
SAF Detection: Identify SAF Position

- **Voltage comparison method** [DAC 2017 & TCAD 2019]
  - Speed up detection by more than **14X** compared with sneak-path technique

- **X-ABFT** [ITC 2018]
  - Identify the faulty column: add two checksum columns
  - Identify the faulty row: apply multiple test input vectors
SAF Tolerance: Restore the Accuracy

- Computation-oriented *redundancy scheme* [ASPDAC 2017 & JETCAS 2018]
  - Mapping algorithm with inner fault-tolerant ability when using redundant crossbars and independent redundant columns

- Improve the accuracy from **25% to 96%** w/ 10% SAF @ MNIST
Device Fault 2: Limited Endurance

- The write endurance of RRAM is limited
  - The typical endurance of a multi-level RRAM cell ranges from $10^6$ to $10^8$
  - Writing RRAM over the endurance may lead to SAF

- SAF caused by limited endurance will hurt training performance
Fault Tolerance Method for Limited Endurance

- **Fault-tolerant training and remapping** [DAC 2017-1 & TCAD 2019-1]
  - Use a threshold-training method to reduce the write times
  - Use a re-mapping scheme: map pruned network value to Stuck-At-0 cells

- Improve the accuracy from **37%** to **83%** @ Cifar-10
Device Fault 3: State Drifting Problem

- **State drifting**: Read operations also change the RRAM resistance slowly.
- State drifting causes *a decline of RCS’s performance*
- ICE: **Inline Calibration** for Memristor Crossbar-based Computing Engine  [DATE 2014]
  - Periodically interrupt-and-benchmark (I&B) RCS
  - Minimize the negative impact of the I&B operation on system performance
- Achieves a calibration efficiency of **91.18%** on average, improving **21.77%** compared to the one with a constant calibration period.
Device Fault 4: Resistance Variation

- **Resistance variation**: the actual change of RRAM resistance is different from the target \( \Delta R_{\text{real}} \sim N(\Delta R_{\text{accurate}}, 0.09R) \)

- Resistance variation makes write operation *inaccurate*

- **Variability-free Tuning** Scheme  [DAC 2017-2 & TCAD 2019-2]
  - Use ideal value and variance of RRAM model for tuning w/ 3\(\sigma\) principle

- The energy efficiency is improved 2.28x on average, 2.29x at most compared with existing tuning scheme
**Device Fault 5: Nonlinear Resistance Distribution**

- The actual fabricated multi-level devices show the distribution of resistance level is *not linear*
  - Nonlinear resistance distribution causes *high computing RMSE*

- Computation *accuracy recovery* framework [DATE 2018]
  - Applying non-linear voltage, retraining, and designing new sensing structure

- Devices with non-linear conductance levels can achieve **the same accuracy** as the ideal linear devices, reduce *99% RMSE*

\[
g_k = \begin{cases} 
  Ck + \delta_k & \text{for } k \geq 0 \\
  Ck^\alpha & \text{for } k < 0 \\
  C\alpha^k & \text{for } k > 0 
\end{cases}
\]

Non-linear resistance distribution

The overall accuracy recovery framework
Outline

• Background and Introduction
  – Convolutional Neural Network Accelerators
  – RRAM and RRAM-based Computing System
  – RRAM Fault Models

• Fault-tolerance for Device-level Faults
  – Stuck-At-Fault (SAF)
  – Limited Endurance
  – State Drifting Problem and Resistance Variation
  – Non-linear Resistance Distribution

• Fault-tolerance for Circuit-level Faults
  – Wire Resistance and IR-drop

• Fault-tolerance for System-level Faults
  – Unbalanced Writing
• **Wire resistances** cause *IR drop problem*, resulting in accuracy loss in large crossbars

• Choices of *load resistor* and *RRAM resistance range* also influence the computation accuracy
Fault Tolerance Method for Circuits Fault

- **Technological exploration** of RRAM crossbar to overcome the circuit-level faults and non-ideal factors [ASPDAC 2015 & JCST 2016]
- **Results of Technological exploration:**
  - Achieve 10.98% improvement of recognition accuracy on the MNIST dataset and 26.4% energy savings compared with previous work
  - More than 84.4% power saving can be achieved at the cost of little accuracy reduction
Outline

• Background and Introduction
  – Convolutional Neural Network Accelerators
  – RRAM and RRAM-based Computing System
  – RRAM Fault Models

• Fault-tolerance for Device-level Faults
  – Stuck-At-Fault (SAF)
  – Limited Endurance
  – State Drifting Problem and Resistance Variation
  – Non-linear Resistance Distribution

• Fault-tolerance for Circuit-level Faults
  – Wire Resistance and IR-drop

• Fault-tolerance for System-level Faults
  – Unbalanced Writing
RRAM Endurance Problem in Training

- The lifetime of existing training RCS is **short**
  - Reason I: RRAM endurance is **limited** (e.g., $10^6 \sim 10^8$), but the number of iteration and RRAM writing in training is **large**
  - Reason II: The weight update is **unbalanced**

<table>
<thead>
<tr>
<th>model</th>
<th>iteration</th>
<th>dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet-5</td>
<td>10,000</td>
<td>MNIST</td>
</tr>
<tr>
<td>ResNet-20</td>
<td>64,000</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>VGG-11</td>
<td>78,200</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>500,000</td>
<td>ImageNet</td>
</tr>
</tbody>
</table>

Expected Lifetime (if endurance $\sim 5 \times 10^6$)

$(5 \times 10^6)/(5 \times 10^5) \approx 10$ times
Fault Tolerance Method for Training RCS

- Long Live TIME: *improve the training lifetime* [DAC 2018]
  - SGS: Structured Gradient Sparsification

Two benefits: Structured Weight Updating
Less Sorting Operations

Convolution kernel: $k^2C \times N$
Complexity: $k^2C \times N + k^2C$
$O(n\log k) \rightarrow O(n)$
Fault Tolerance Method for Training RCS

- Long Live TIME: improve the training lifetime [DAC 2018]
  - SGS: Structured Gradient Sparsification
  - ARS: Aging-aware Row Swapping

Counter Memory Requirement:
\[ k^2 C \times B, B = \log_2 \text{MaxIter} \]

Two hyper-parameters:
ARS Interval: \( SI \)
Number of swapped rows: \( R \)
• Long Live TIME: *improve the training lifetime* [DAC 2018]
  – SGS: Structured Gradient Sparsification
  – ARS: Aging-aware Row Swapping
  – *SGS-ARS Training Framework*
Fault Tolerance Method for Training RCS

- **Results of Long Live TIME** [DAC 2018]
  - Achieve 356x longer lifetime in the training task of ResNet-50 on ImageNet
Conclusions

• An RRAM-based computing system (RCS) provides a promising solution for neuromorphic computing

• RCS is vulnerable to faults
  – RCS contains 3-level faults: device, circuit, and system
  – Testing and fault-tolerant designs are important for RCS
  – Promising solutions have recently developed for testing and fault tolerance in RCS

• Next steps: Better understanding of the physics of defects and the impact of faults on circuit operation
Our Related Work


Our Related Work


Thanks for your attention