# Toward Robust Neural Network Computation on Emerging Crossbar-based Hardware and Digital Systems

Yiyu Shi, Masanori Hashimoto

Jan. 22, 2024

# Toward Robust Neural Network Computation on Emerging Crossbar-based Hardware and Digital Systems

Yiyu Shi, Masanori Hashimoto

Jan. 22, 2024

# Organization

**Yiyu Shi**

- Efficient worst-case analysis for neural network inference using emerging device-based CiM,
- Enhancement of worst-case performance through noise-injection training,
- Co-design of software and neural architecture specifically for emerging device-based CiMs.

**Masanori Hashimoto**

- Identification of vulnerabilities in neural networks,
- Reliability analysis and enhancement of AI accelerators for edge computing,
- Reliability assessment of GPUs against soft errors.

# Toward Robust Neural Network Computation on *Emerging Crossbar-based* Hardware and Digital Systems

Yiyu Shi

Dept. of CSE, University of Notre Dame

yshi4@nd.edu

Jan. 22, 2024

UNIVERSITY OF NOTRE DAME

# Outline

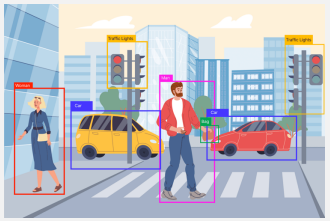
- Introduction: Crossbar-based Hardware and their Robustness Issues
- Remedy Methods: Cross-Layer Co-Design
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

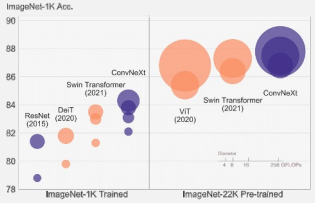
# Outline

- **Introduction: Crossbar-based Hardware and their Robustness Issues**
- Remedy Methods: Cross-Layer Co-Design
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# DNN on Edge: Memory Wall Issue

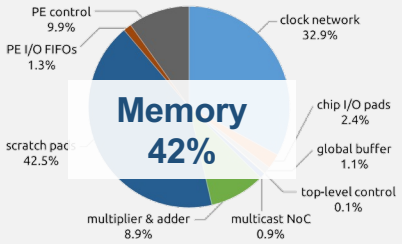
## ❑ Goal: Deploy DNN on Edge

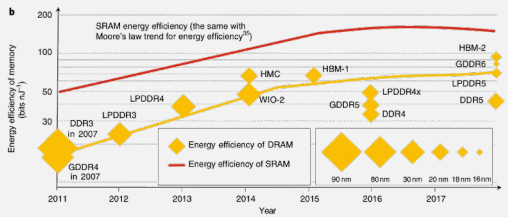


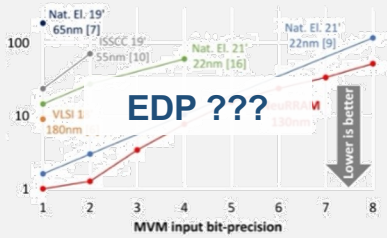



Handle Various Tasks  Gets SOTA Performances  Promising for Edge Apps.  Limited Power Budget

## ❑ Memory Wall issue for Efficient DNN Acceleration



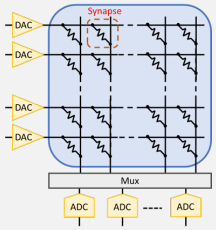




High Memory Access Cost [1]  Slow Memory Tech Improvement  Efficiency Bottleneck

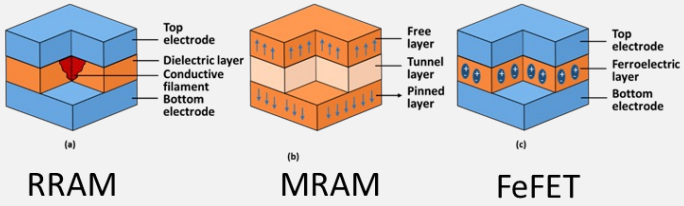
[1] Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." *IEEE journal of solid-state circuits* 52.1 (2016)

# Hardware Solution: Compute-in-Memory

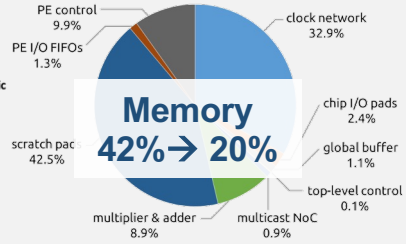
## ❑ CiM DNN Accelerator using Crossbar Arrays: Advantages



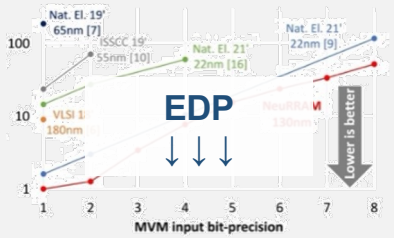

CiM Architecture [2]




Emerging NVM Devices



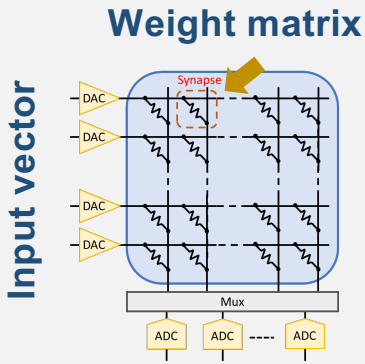

Lower Memory Cost

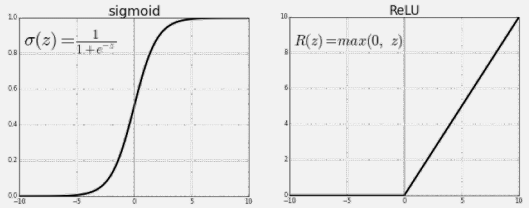



Higher Efficiency

## ❑ Crossbar Array: VMM Engine

- **Input:** Voltage
- **Weight:** Conductance
- **Output:** Current



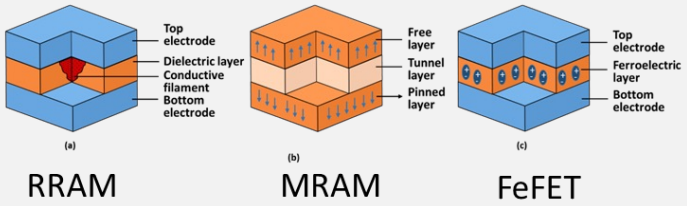

A/D Conversion Needed

[2] Zheyu Yan, X. Sharon Hu and Yiyu Shi, "On the Reliability of Computing-in-Memory Accelerators for Deep Neural Networks", chapter in *System Dependability and Analytics: Approaching System Dependability from Data, System and Analytics Perspectives*, Springer, 2023.

# Emerging Technology: Pros and Cons

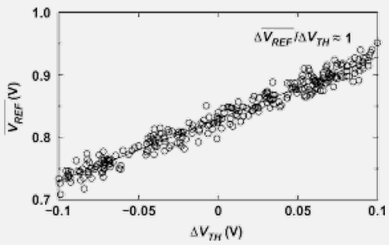
## ❑ Emerging NVM Devices Advantages

- **Non-volatile:** used as storage & memory
- **Compact:** more data on chip
- **Read:** Fast & Low energy

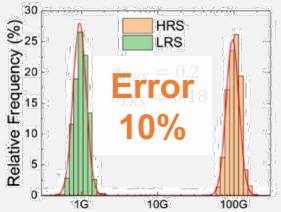



**Emerging NVM Devices**

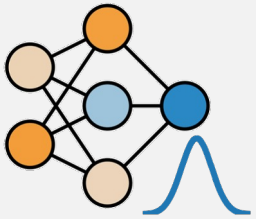
## ❑ Challenges from Device Variations



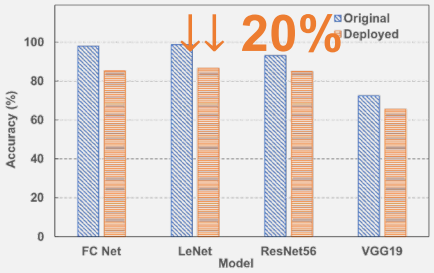

**D2D Variations**




**Incorrect Conductance**



**Incorrect Weight**

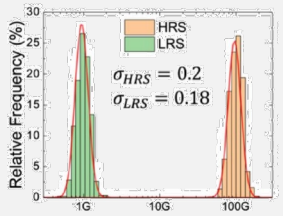



**Accuracy Drops [3]**

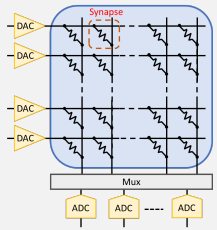
[3] Yan, Zheyu, et al. "Uncertainty modeling of emerging device based computing-in-memory neural accelerators with application to neural architecture search." *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2021.

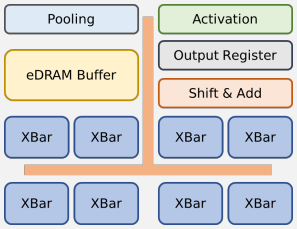
# Device Variations: Evaluations

## ❑ Existing Evaluation Workflows

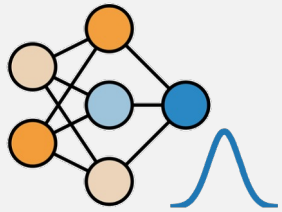



Device Modeling



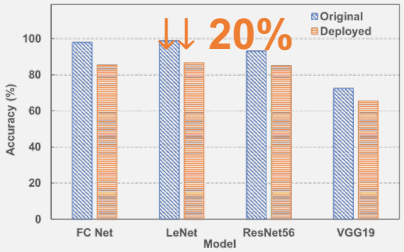




Circuit/Arch Abstraction

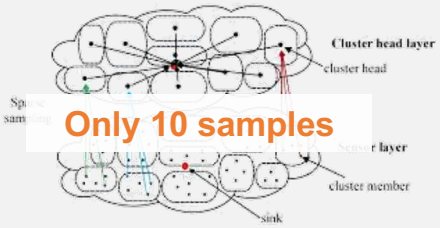


Monte Carlo Simulation

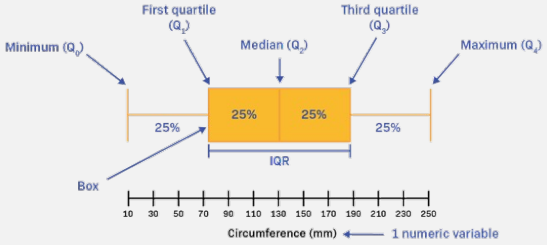
## ❑ Issues of Existing Methods




Focus on Average Case



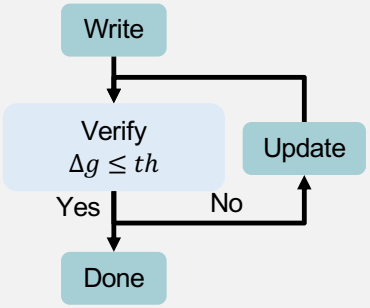

Very Few MC runs



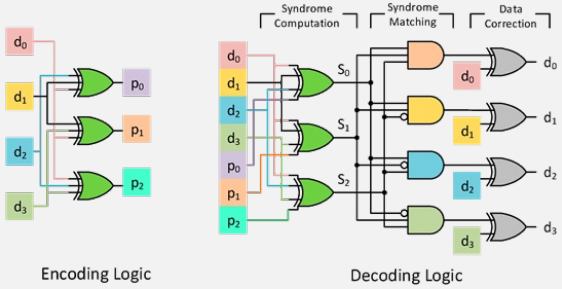

Lack Error Bound Guarantee

# Device Variations: Remedies

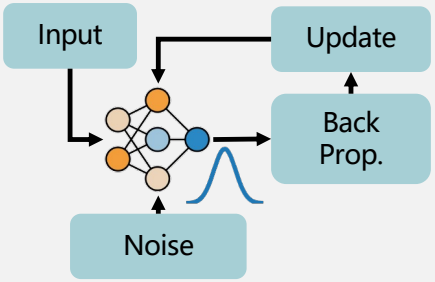
## ❑ Device Variation: Existing Solution




Write-Verify (W-V)



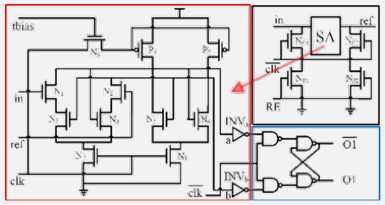

Error Correction/Denoising



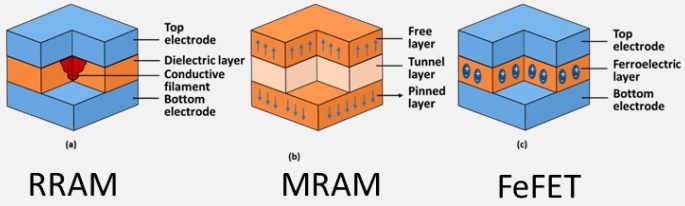

Noise-Aware Training

## ❑ Drawbacks of These Solutions


Human Labor



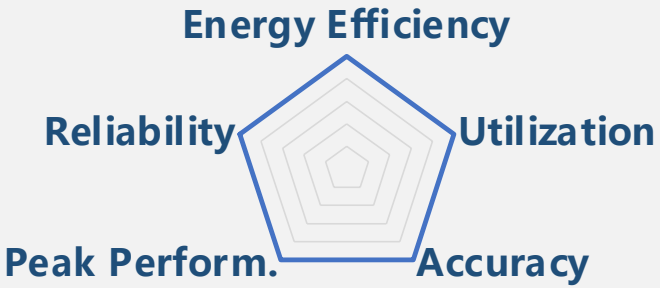
Peripheral Circuit Overhead



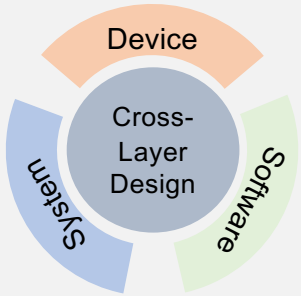

Device Type Dependent

# Our Approach: Cross-Layer Co-Design

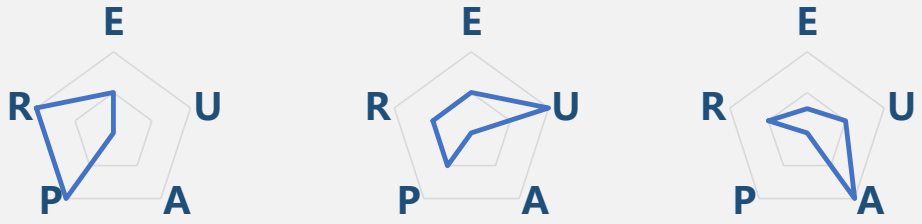
## ❑ Metrics for AI Acceleration




## ❑ AI Acceleration Design Levels

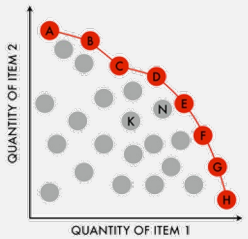



## ❑ Contributions of Different Layers

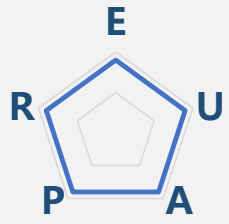



Device/Circuit | System | Software

## ❑ Advantages for Co-Design




Multi-object Opt.



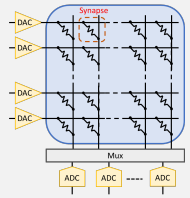

Joint Optimal

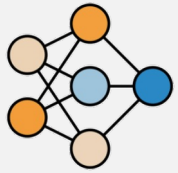
# Outline

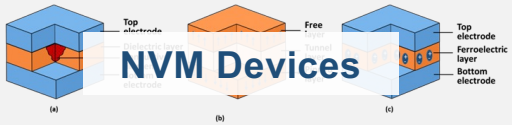
- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# Remedy Methods: Overview

## Background: CiM for DNN



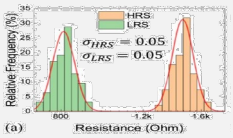





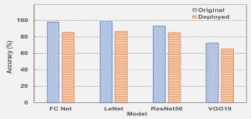

RRAM MRAM FeFET

## Key Problem:

**Device Variation & Acc. Drop**




Data Difference




Accuracy Drop

## Solution: Cross-Layer Co-Design

| | |
|---|---|
| Device | Detect Sensitive Devices [DAC 22] |
| Circuit/ Arch | Worst-Case Analysis [ICCAD 22] |
| | AutoML [ASPDAC 22] |
| Software | Train Robust DNN Models [ICCAD 23] |

**Co-Design**

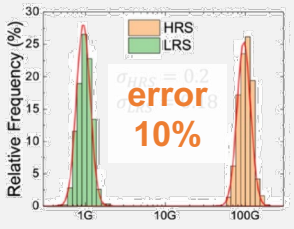
- Object Detection [Trans on Computers]
- Img Segmentation [Nature Electronics]
- Img Classification [SoCC 23]

# Outline

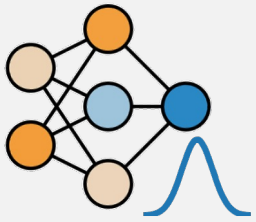
- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - **Device: Device Programming Techniques**
  - Circuit/Arch: Worst-case Analysis
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# Selective Write-Verify (1)

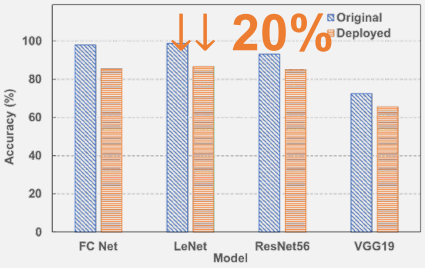
## ❑ Device Variation: Existing Solution



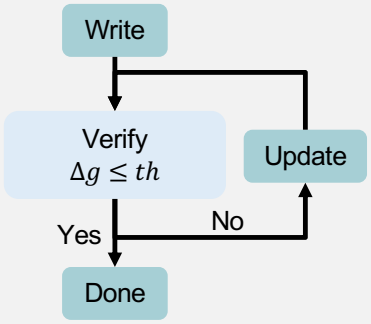

Device Variation



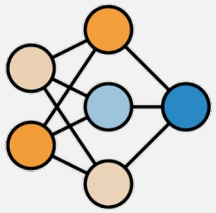
Incorrect Weights




Accuracy Drops




Write-Verify (W-V)



Accuracy Recovers

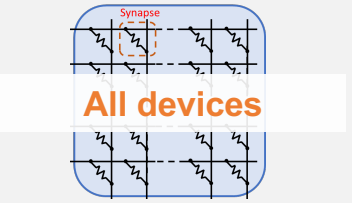
## ❑ W-V is Time Consuming


1 Day


Human Labor

## ❑ Reasons for W-V to be Slow


No Parallelism

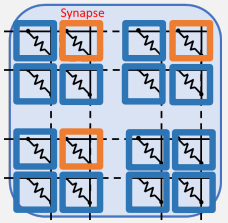



Program All Devices

# Selective Write-Verify (2)

## ❑ Overview

- Write-verify **a portion** of the devices
- Write the other device **once**
- **Accelerate** the deployment process




**Use Write-Verify**

**Write only once**

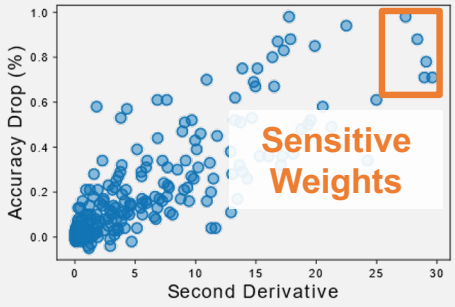
**W-V a portion of the devices**

## ❑ Solution: Detect Sensitive Device

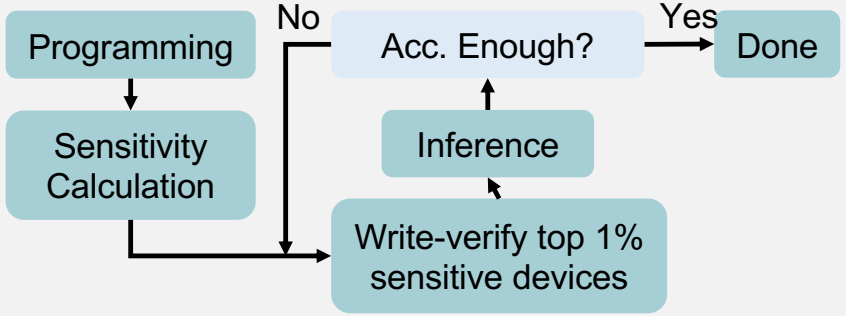
$$E[\Delta f(\mathbf{w})] \approx \frac{1}{2}\sum_{i=1}^{n}\mathcal{H}_{ii}E[(\Delta w_i)^2] + \frac{1}{2}\sum_{i\neq j}^{n}\mathcal{H}_{ij}E[\Delta w_i]\times E[\Delta w_j] = \frac{1}{2}\sum_{i=1}^{n}\mathcal{H}_{ii}E[(\Delta w_i)^2] = \frac{1}{2}\sum_{i=1}^{n}\frac{\partial^2 f}{\partial \bar{w}_i^2}E[(\Delta w_i)^2]$$

Second Derivative

**Statistical Analysis**




**Detect Sensitive Weights**



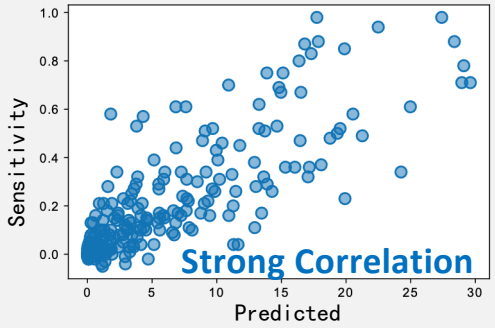

**Only W-V Sensitive Devices**

[4] **Z. Yan**, X. S. Hu, and Y. Shi, "SWIM: Selective write-verify for computing-in-memory neural accelerators," 2022 59th ACM/IEEE Design Automation Conference (**DAC**)

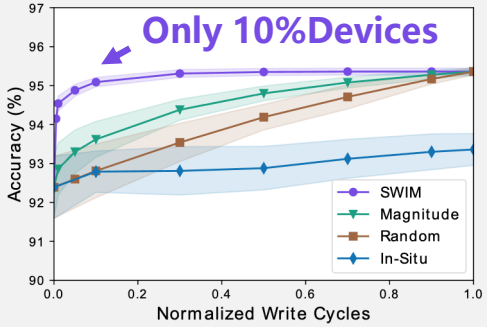
# Selective Write-Verify (3)

## ❑ Results

- Published in EDA top Conference **DAC** [3]
- Cited by **Nature Paper** from Ju Li, **MIT** [4]




**90% Sensitivity Measurement Acc**




**10x Deployment Speedup**

**nature**

**Thousands of conductance levels in memristors integrated on CMOS**

Memristive-switching devices are known for their relatively large dynam … e number of discret … n developed to accu … s with fewer than 200 conductance levels have been reported so far[22,32]. There are

**Can achieve accurate programming**

[3] Yan, Zheyu, Xiaobo Sharon Hu, and Yiyu Shi. "SWIM: Selective write-verify for computing-in-memory neural accelerators." DAC 2022 (CCF-A).
[4] Rao, Mingyi, et al. "Thousands of conductance levels in memristors integrated on CMOS." ***Nature*** 615.7954 (2023): 823-829.

# Outline

- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - **Device: Device Programming Techniques: Technical Details**
  - Circuit/Arch: Worst-case Analysis
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# Weight Sensitivity Evaluation

- Target: statistically evaluate the influence of device variations
- Method: Taylor series of the DNN loss function
- Annotation:
  - $f$: loss function
  - $\boldsymbol{w} = \tilde{\boldsymbol{w}} + \Delta\mathbf{w}$: weight
  - $H(\tilde{\boldsymbol{w}})$: Hessian matrix
  - $E$: expectance (average)
- **Conclusion:** write-verify weights with high 2nd derivatives

$$f(\boldsymbol{w}) = f(\tilde{\boldsymbol{w}}) + \frac{\partial f}{\partial \tilde{\boldsymbol{w}}}\Delta\mathbf{w} + \frac{1}{2}\Delta\boldsymbol{w}^T H(\tilde{\boldsymbol{w}})\Delta\mathbf{w} + \mathrm{o}(\Delta\boldsymbol{w}^3)$$

$$\Delta f(\boldsymbol{w}) \approx \frac{1}{2}\Delta\boldsymbol{w}^T H(\tilde{\boldsymbol{w}})\Delta\mathbf{w}$$

$$= \frac{1}{2}\sum_{i=1}^{n} H_{ii}(\Delta w_i)^2 + \frac{1}{2}\sum_{i\neq j}^{n} H_{ij}\Delta w_i \Delta w_j$$

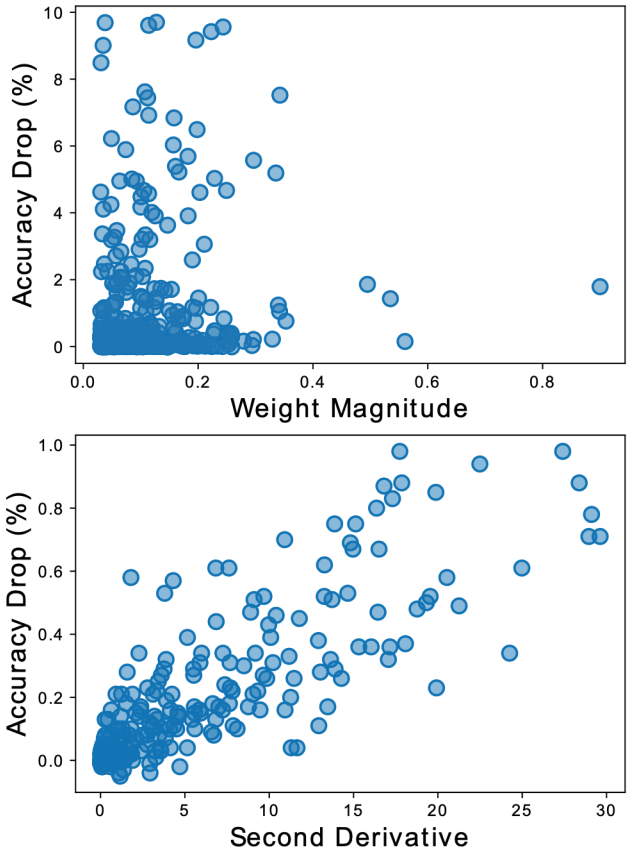
$$\Delta w_i \sim N(0, \sigma)$$

$$E[\Delta f(\boldsymbol{w})] \approx \frac{1}{2}\sum_{i=1}^{n} H_{ii}\, E[(\Delta w_i)^2] = \frac{\sigma^2}{2}\sum_{i=1}^{n} H_{ii}$$

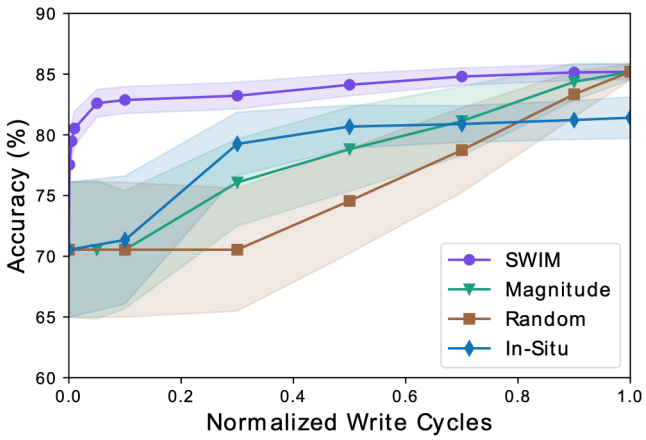
**Weight's sensitivity to device variations can be represented by its second derivative**

# Effectiveness of Using Second Derivative

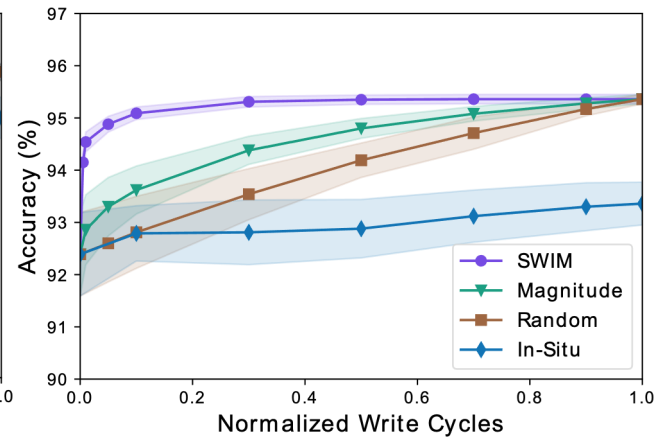
- Annotations
  - Y axis for both figures: accuracy drop when changing a weight (MNIST)
  - X axis for figure up: Weight magnitude
  - X axis for figure down: Weight second derivative
- Conclusions
  - Accuracy drop and weight magnitude are poorly co-related
  - Accuracy drop and second derivatives are strongly co-related
  - Second derivative is a good metric for sensitivity estimation



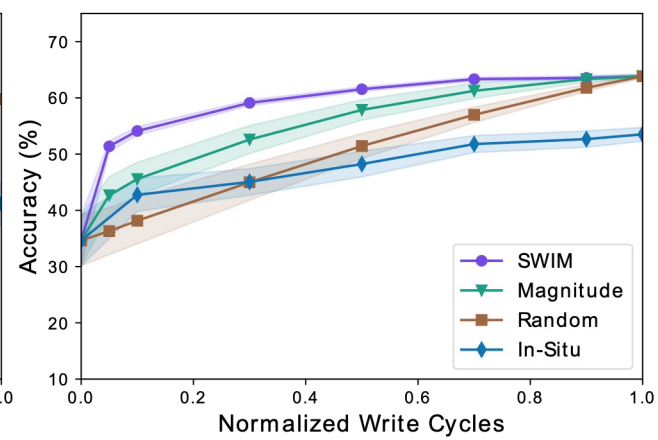
# SWIM Vs Baselines on Different Datasets




CIFAR-10 - ConvNet




CIFAR-10 – ResNet-18




Tiny ImageNet – ResNet-18

- Baselines: use weight magnitude or random as weight selection + on device training
- Solid line: average performance, Shadow: ranges for standard deviation
- SWIM much better than all baselines
- Achieves low enough (less than 2%) accuracy drop by writing-verifying less than 10% of the weights

# Summary

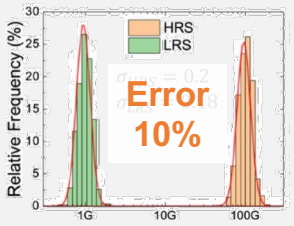
- Proposed a framework that requires writing-verifying only a small portion of weights
- The framework can maintain DNN accuracy
- In the meantime, programming time drastically reduced
- Specifically, the proposed framework achieves up to 10x speedup

# Outline

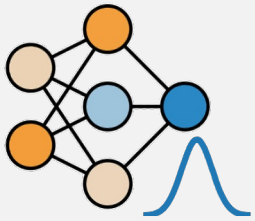
- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - Device: Device Programming Techniques: Technical Details
  - **Circuit/Arch: Worst-case Analysis**
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# Worst-Case Analysis (1)

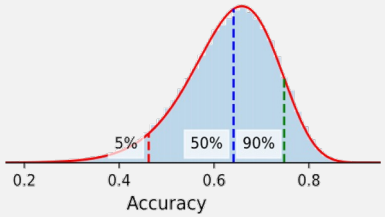
## ❑ Background: Reliability of nvCiM DNN Accelerators




Device Variations

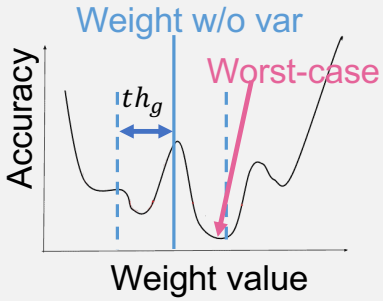


Random Weights




Random Acc.

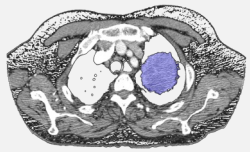
Safety Critical Apps




Worst-Case Analysis

## ❑ Safety Critical Apps. & Worst-Case

Diagnosis Support



Cancer Miss Rate

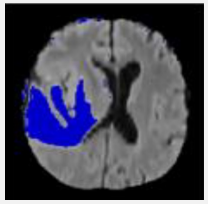
Autonomous Driving

Accident Rate

[5] **Z. Yan**, X. S. Hu, and Y. Shi, "Computing in memory neural network accelerators for safety-critical systems: Can small device variations be disastrous?" 2022 International Conference on Computer-Aided Design (**ICCAD**)

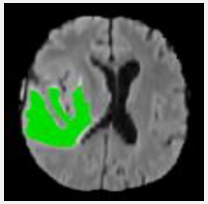
# Worst-Case Analysis (2)

## ❑ What if Ignoring Worst-Case Analysis

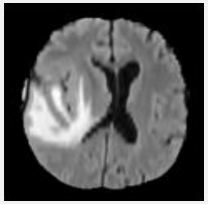


Target:
Find Cancer Cells

Average Only

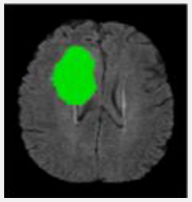


99% Find All Cells

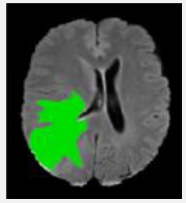


1% Find NO Cell

Ideally

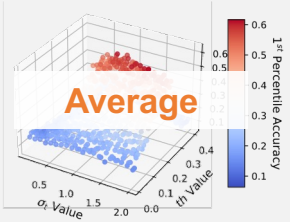


100% Find Some Cells

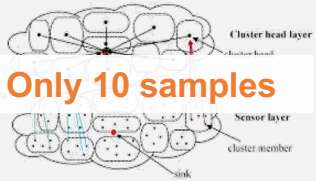


0% Find NO Cell

## ❑ No WC Analysis Now

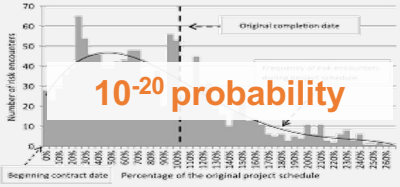



Only Average Acc.

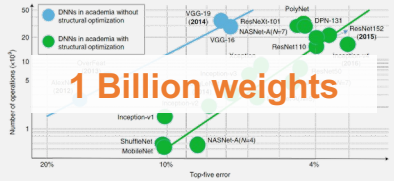



Few Samples

## ❑ Existing Methods Would Not Work



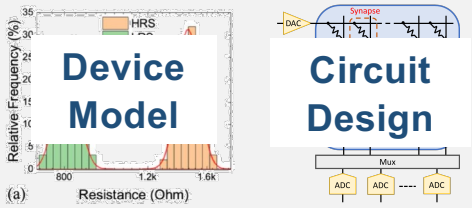

Monte Carlo Methods
x Low Probability




Exhaustive Search
x High Dimensions

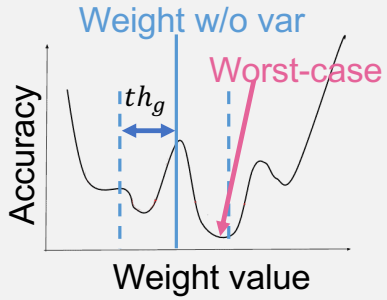
# Worst-Case Analysis (3)

## ❑ Solution: Define it as Constrained Optimizations




$$w - th_g \le \hat{w} \le w + th_g$$

**Build a Noise Model**




**Def. Constrained Opt. Problem**

$$\underset{\Delta \boldsymbol{W}}{\text{minimize}} \sum_{\boldsymbol{x} \in D} p(\boldsymbol{x}, \{f, \boldsymbol{W}\})$$

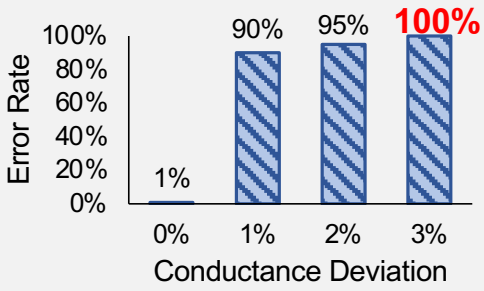
$$s.t. \quad L(\Delta \boldsymbol{W}) \le th_g$$

$$p(\boldsymbol{x}, \{f, \boldsymbol{W}\}) = \max\{O_t - \max_{i \ne t}(O_i), 0\}$$

**Relax to Differentiable Objective**

## ❑ Findings: Very Low Reliability!

- **3% conductance deviation → Miss-classify all inputs**
- Existing protection methods are not effective
- Published in ICCAD 22 [5]




**Very Low Reliability**

[5] **Z. Yan**, X. S. Hu, and Y. Shi, "Computing in memory neural network accelerators for safety-critical systems: Can small device variations be disastrous?" 2022 International Conference on Computer-Aided Design (**ICCAD**)

# Outline

- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - Device: Device Programming Techniques
  - **Circuit/Arch: Worst-case Analysis: Technical Details**
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# Formulating Worst-Case using Optimization

$$f(\boldsymbol{W} + \Delta\boldsymbol{W}, \boldsymbol{x})$$

$$(\boldsymbol{x}, t) \in D$$

$$\underset{\Delta\boldsymbol{W}}{\text{minimize}}\ |\{f(\boldsymbol{W} + \Delta\boldsymbol{W}, \boldsymbol{x}) == t | (x, t) \in D\}|$$

$$s.t. \quad L(\Delta\boldsymbol{W}) \leq th_g$$

- Neural architecture $f$, weight $\boldsymbol{W}$ and weight perturbation $\Delta\boldsymbol{W}$
- Input $\boldsymbol{x}$, label $t$ and dataset $D$
- Minimize the size of the set of correctly classified inputs → minimize accuracy
- Subject to the constraint that perturbation distance smaller than $th_g$

# Solving Optimization using Relaxation

- Goal: $\underset{\Delta \boldsymbol{W}}{\text{minimize}} \, |\{f(\boldsymbol{W} + \Delta \boldsymbol{W}, \boldsymbol{x}) == t | (x, t) \in D\}|$
- The goal is discrete and difficult to optimize, relaxation needed
- Relax to a continuous function for each input:
  - $\underset{\Delta \boldsymbol{W}}{\text{minimize}} \sum_{\boldsymbol{x} \in D} p(\boldsymbol{x}, \{f, \boldsymbol{W} + \Delta \boldsymbol{W}\})$
  - $p(\boldsymbol{x}, \{f, \boldsymbol{W} + \Delta \boldsymbol{W}\}) > 0$, if and only if, $f(\boldsymbol{W} + \Delta \boldsymbol{W}, \boldsymbol{x}) == t$
  - Function that satisfies the requirement

$$p(\boldsymbol{x}, \{f, \boldsymbol{W} + \Delta \boldsymbol{W}\}) = \max\{O_t - \max_{i \neq t}(O_i), 0\}$$

- Constraint $L(\Delta \boldsymbol{W}) \leq th_g$: Lagrange multiplier c
- $\underset{\Delta \boldsymbol{W}}{\text{minimize}} (c \cdot \sum_{\boldsymbol{x} \in D} p(\boldsymbol{x}, \{f, \boldsymbol{W} + \Delta \boldsymbol{W}\}) + (L(\Delta \boldsymbol{W}) - th_g))$
- Gradient descent can be used to solve this problem

[5] Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." *2017 IEEE symposium on security and privacy (sp)*. IEEE, 2017.

# Major Results for Worst-Case DNN Performance

- Baselines: Monte-Carlo Simulation (MC) & Projected Gradient Descent (PGD)
- Proposed method discovers models with lower accuracy (tighter lower bound)
- MC method failed to find models with low enough accuracy
- The proposed method finds wors-case performance efficiently

| Dataset | Model | Ori. Acc. | Worst-case Accuracy (%) | | | Time (Minutes) | | |
|---|---|---|---|---|---|---|---|---|
| | | | MC | PGD | Proposed | MC | PGD | Proposed |
| MNIST | LeNet | 99.12 | 97.34 | 13.44 | **7.35** | 900 | 3.3 | 5.5 |
| CIFAR-10 | ConvNet | 85.31 | 60.12 | 10.00 | **4.27** | 2700 | 4.2 | 6.0 |
| CIFAR-10 | ResNet-18 | 95.14 | 88.77 | 10.00 | **0.00** | 5400 | 13.3 | 20.0 |
| Tiny ImageNet | ResNet-18 | 65.23 | 25.33 | 0.50 | **0.00** | 14400 | 40.0 | 60.0 |
| ImageNet | ResNet-18 | 69.75 | 43.98 | 0.10 | **0.00** | 231000 | 1980 | 2880 |
| ImageNet | VGG-16 | 71.59 | 66.43 | 0.10 | **0.06** | 313800 | 2530 | 3820 |

# Summary

- Proposed an efficient framework to examine worst-case performance of DNNs
- Showed that the accuracy of a well-trained DNN can drop drastically to almost zero with very subtle perturbations
- Existing methods are either too costly (for stronger write-verify) or ineffective (for training-based methods)
- Further research is needed to find a solution to this issue

# Outline

- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - **Software: HW-Aware Training**
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

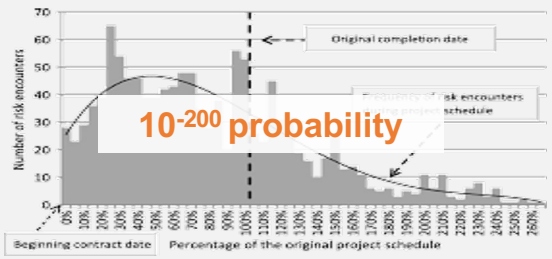
# Outline

- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - **Software: HW-Aware Training**
    - **Overview**
    - Detailed Solution
    - Experimental Results
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# Realistic Worst-Case (1)

- **New Challenge: Very Low Reliability!**
  - **3% conductance deviation → Miss-classify all inputs**
  - Existing protection methods not effective
  - **End of the world?**

- **Issues for Absolute Worst-Case**



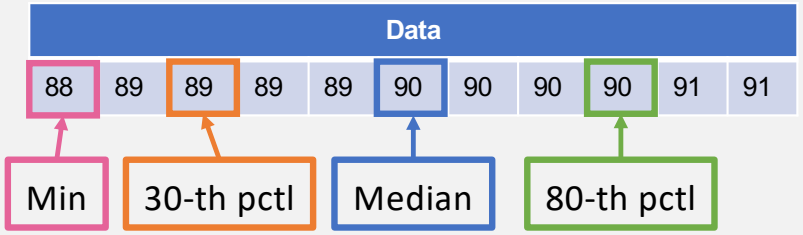

Low Probability

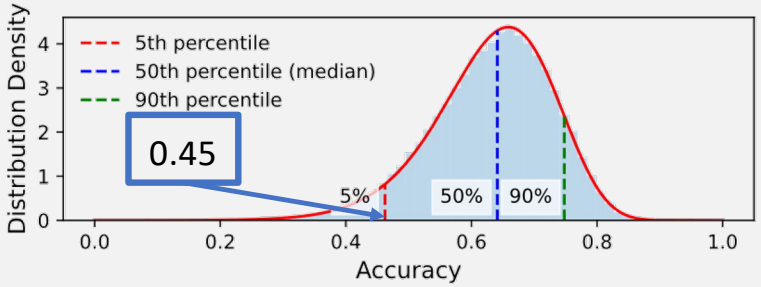

Partially Verified


Too Costly

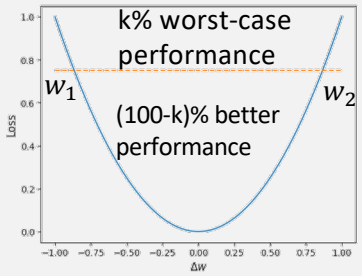
# Realistic Worst-Case (2)

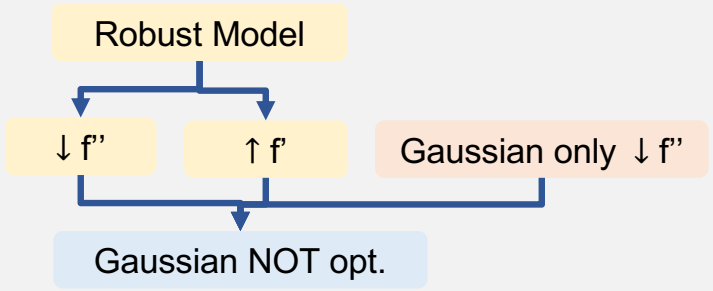
## ❑ New Metric: K-th Percentile Performance (KPP)

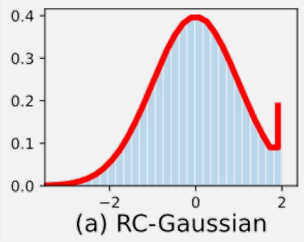



## ❑ Improving KPP




**Statistical Model for KPP** | **Gaussian Noise Injection is not Optimal** | **Design New Noise**

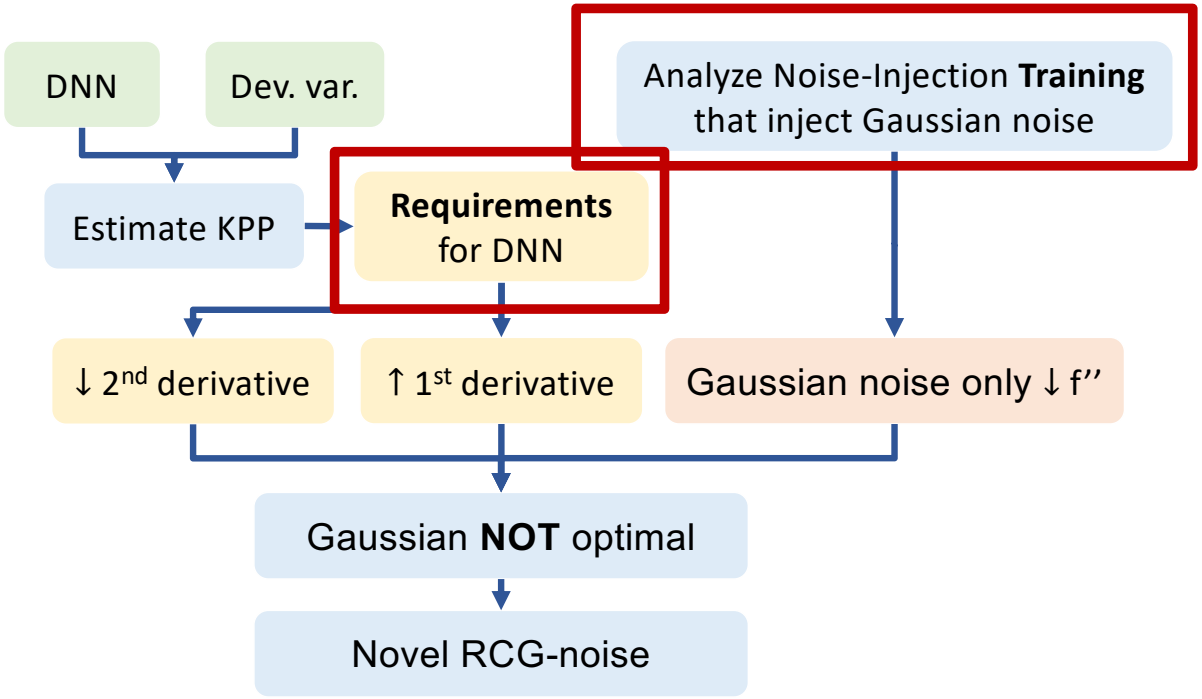
[6] **Z. Yan**, Y. Qin, X. S. Hu, and Y. Shi, "Improving realistic worst-case performance of nvcim dnn accelerators through training with right-censored gaussian noise," 2023 International Conference on Computer-Aided Design

# Outline

- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - **Software: HW-Aware Training**
    - Overview
    - **Detailed Solution**
    - Experimental Results
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# Detailed Solution Overview

- Goal: improve realistic worst-case accuracy (KPP) of DNN under device variations



# Link KPP with Model Properties

- Use loss to represent performance, find its Taylor series (the smaller the better)

$$f(w_0 + \Delta w) = f(w_0) + \Delta w f'(w_0) + \frac{(\Delta w)^2}{2} f''(w_0) + o\left((\Delta w)^3\right) \quad (1)$$

Dev. Var.

- ***KPP*** estimation
  - Given: a DNN model, device variation distribution, and probability k%
  - Find: KPP
  - Key: write KPP in the form of an **equation** with **model properties** and **k**

# KPP Estimation: Details

$$f(w_0 + \Delta w) = f(w_0) + \Delta w f'(w_0) + \frac{(\Delta w)^2}{2} f''(w_0) + o((\Delta w)^3) \quad (1)$$

$\Delta w$ Distribution:
$\Delta w \sim N(0, \sigma_d)$

Let *KPP* be an unknown variable

Find the range of $\Delta w$ that loss $\geq KPP$

Eq. (1): Quadratic function

$\Delta w \leq w_1$ or $\Delta w \geq w_2$
Roots of Quadratic function

Find probability k% of $\Delta w$ in this range

$(100 - k)\% = cdf_G(w_2) - cdf_G(w_1)$

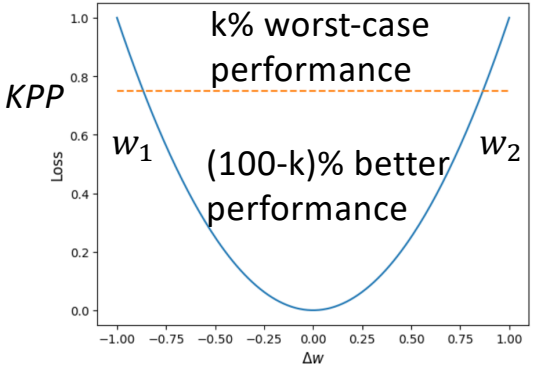
Represent KPP with k%

Closed-form impossible

Approximations

1st order approximation: next slide

3rd order approximation: more than 4 A4 pages




$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2} \left[ x + \frac{x^3}{3} + \frac{x^5}{3 \cdot 5} + \cdots + \frac{x^{2n+1}}{(2n+1)!!} + \cdots \right]$$

# Desired Model Properties

- Use loss to represent performance, find its Taylor series (the smaller the better)

$$f(w_0 + \Delta w) = f(w_0) + \Delta w f'(w_0) + \frac{(\Delta w)^2}{2} f''(w_0) + o\left((\Delta w)^3\right)$$

Dev. Var.

- ***KPP* estimation**

Percentile

$$-KPP \approx -\frac{f'(w_0)^2}{2f''(w_0)} + f(w_0) + \frac{f''(w_0)\pi k^2 E[(\Delta w)^2]}{4} \qquad (3)$$

- **Requirements:** $f''(w_0)\downarrow$, $f(w_0)\downarrow$, and $|f'(w_0)|\uparrow$

# Noise Injection Training Process Analysis

- Noise injection training weight update

$$w_{t+1} = w_t - \alpha f'(w_t + w_n)$$

Updated w | Original w | LR | Loss fn | Gradient | Noise

- Taylor Series:

$$w_{t+1} = w_t - \alpha \left( f'(w_t) + w_n f''(w_t) + \frac{w_n^2}{2} f'''(w_t) + o\left((w_n)^3\right) \right)$$

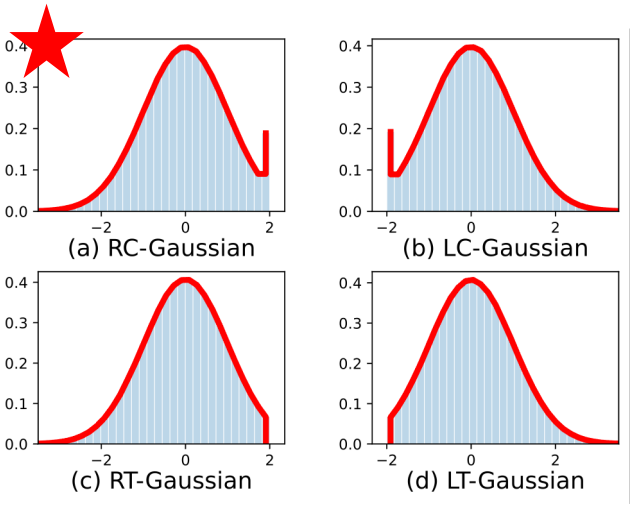
- Averaged effect: $w_{t+1} = w_t - \alpha E[f'(w_t + w_n)]$

$$w_{t+1} = w_t - \alpha \left( f'(w_t) + \mathrm{E}[w_n] f''(w_t) + \frac{E[(w_n)^2]}{2} f'''(w_t) \right) \qquad (1)$$

In paper, $w_n$ is denoted as $\Delta w$. We change this terminology to avoid ambiguity in the slides.

# Findings: How Noise-Injection Training Improves KPP

- How noise-injection training fulfills the requirements
  - Requirements: $f(w_0)\downarrow$, $|f'(w_0)|\uparrow$, and $f''(w_0)\downarrow$
  - $w_{t+1} = w_t - \alpha\left(f'(w_t) + E[w_n]f''(w_t) + \frac{E[(w_n)^2]}{2}f'''(w_t)\right)$
- Desired noise properties $\boldsymbol{E[w_n] \neq 0}$, $\boldsymbol{E[(w_n)^2] > 0}$
- Gaussian **does not** hold this property!
- Propose four candidates
- Training with RIght-Censored Gaussian NoisE (**TRICE**)



# Outline

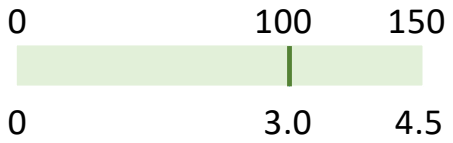
- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - **Software: HW-Aware Training**
    - Overview
    - Detailed Solution
    - **Experimental Results**
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# Experimental Setups

- Baselines:
  - Training **w/o** noise
  - CorrectNet [7]
  - Injecting **Gaussian** noise in training
- Evaluation method:
  - Metric: **KPP**, k = 1 (p = 1%).
  - Monte Carlo runs: 10,000

- **Device mapping model**

Device conductance (g): 0 – 150 $\mu$S*




Weight value (w): 0 – 4.5**

- **Device variation model**
  - Conductance follows Gaussian dist.
  - $g = N\left(\frac{g_t}{\max(g)}, \sigma_d\right) \times \max(g)$

* Absolute 0 $\mu$S is impossible so here it means a very high resistance
** Negative weights are mapped to another array

[7] Eldebiky, Amro, et al. "CorrectNet: Robustness Enhancement of Analog In-Memory Computing for Neural Networks by Error Suppression and Compensation." *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023.

# Results on MNIST Dataset

| Dev. var. ($\sigma_d$) | Training Method w/o noise | CorrectNet | Gauss. | TRICE |
|---|---|---|---|---|
| 0.00 | 99.01 | 97.99 | 98.86 | 98.94 |
| 0.10 | 70.72 | 90.66 | 95.59 | **95.99** |
| 0.20 | 19.81 | 39.54 | 66.04 | **77.82** |
| 0.30 | 08.58 | 14.26 | 23.09 | **38.51** |
| 0.40 | 06.05 | 09.23 | 10.38 | **17.94** |

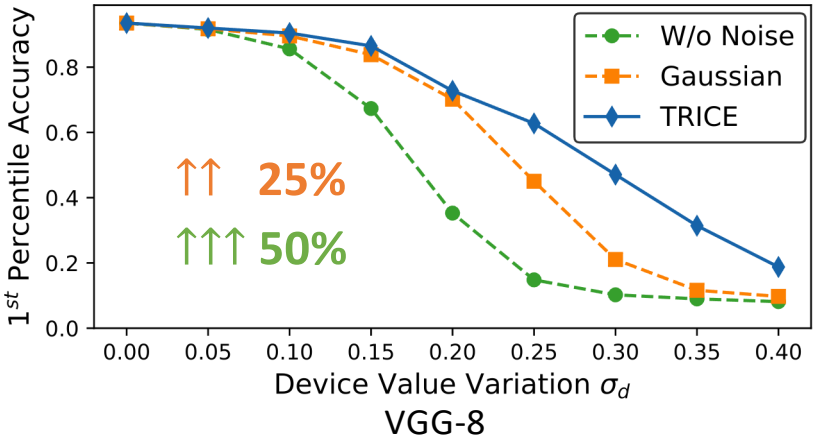
↑ 58%, 38%

↑ 14%

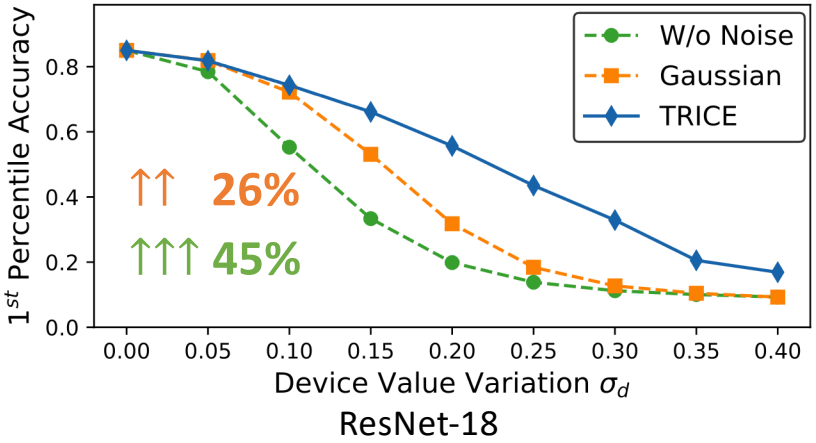
- **Model: LeNet,** 4-bit quantization
- Metric: K-th Percentile Performance (KPP) → 1-st percentile accuracy
- Columns: comparing three baselines with the proposed method TRICE
- Rows: over different device variation magnitude ($\sigma_d$)
- Following experiments: CorrectNet [4] ×

[7] Eldebiky, Amro, et al. "CorrectNet: Robustness Enhancement of Analog In-Memory Computing for Neural Networks by Error Suppression and Compensation." *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023.

# Results on CIFAR-10 Dataset



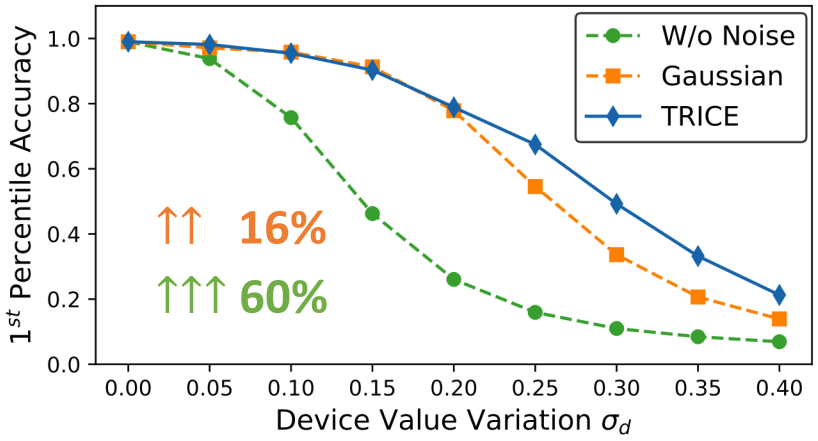
VGG-8



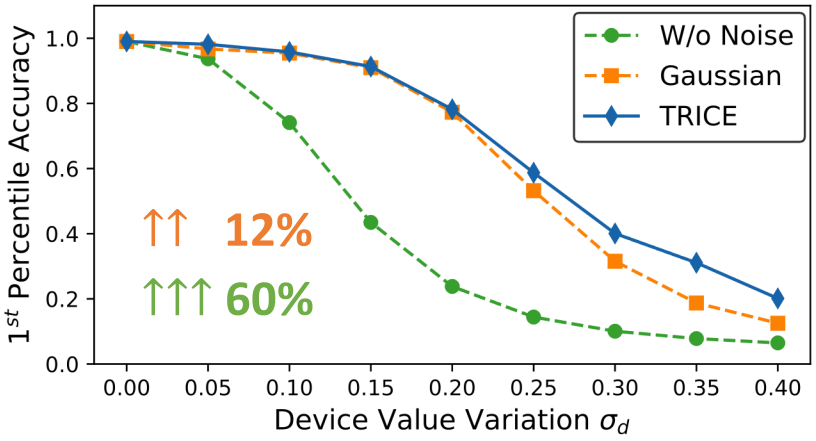
ResNet-18

- (a) VGG-8 model and (b) ResNet-18 model, 6-bit quantization
- X-axis: device variation magnitude ($\sigma_d$)
- Y-axis: KPP: 1-st percentile accuracy

# Results when using Different devices

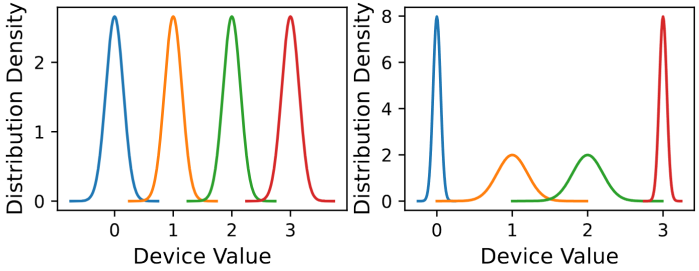



FeFET-1




FeFET-2

- Previous two experiments: **RRAM** devices vs. this experiment: **FeFET** devices
- **Model: LeNet,** 4-bit quantization
- Dataset: MNIST




RRAM FeFET

# Summary

- Advocate the use of a realistic worst-case performance metric (KPP)
- Propose a novel noise-injection training method to improve KPP
- Show that injecting right-censored Gaussian noise can effectively improve KPP
- The proposed framework improves KPP by up to 25%
- Published in ICCAD 23
- Received **Best Paper Award**

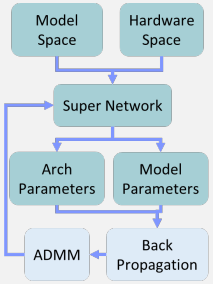
# Outline

- Introduction: Crossbar-based Hardware and their Robustness Issues
- **Remedy Methods: Cross-Layer Co-Design**
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - Software: HW-Aware Training
  - **Co-Design: HW-SW Co-Design Algorithm**
- Outlook & Conclusions

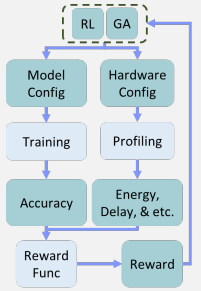
# HW-SW Co-Design Algorithm (1)

## ❑ Existing Methods

- Given: a task and a design space
- Find: the optimal HW-SW design pair

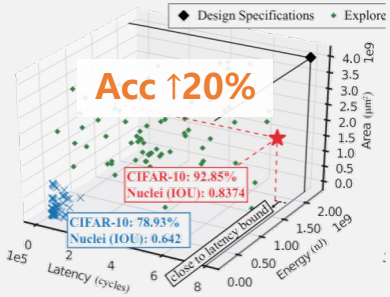



Differentiable Methods




Child Network-Based

## ❑ Using Existing Methods




Object Detection

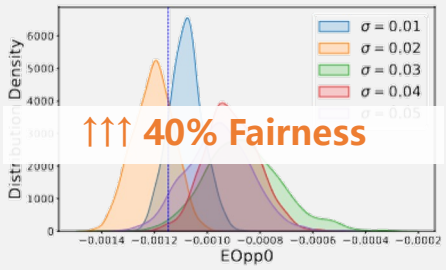
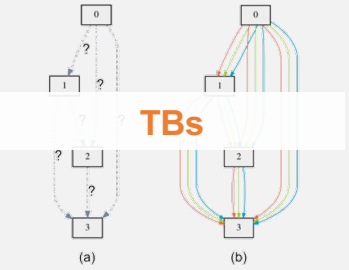



Image Segmentation

## ❑ Issues for Existing Methods


Search Time




Memory Cost

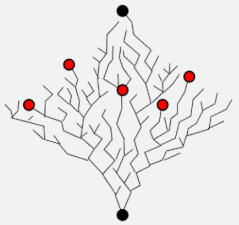
[8] W. Jiang, Q. Lou, Z. Yan, et al., "Device-circuit-architecture co-exploration for computing-in-memory neural accelerators," IEEE Transactions on Computers, 2020
[9] Y. Guo, Z. Yan, X. Yu, et al., "Hardware design and the fairness of a neural network", Nature Electronics (under review)

# HW-SW Co-Design Algorithm (2)

## ❑ Why Existing Methods are Not Efficient

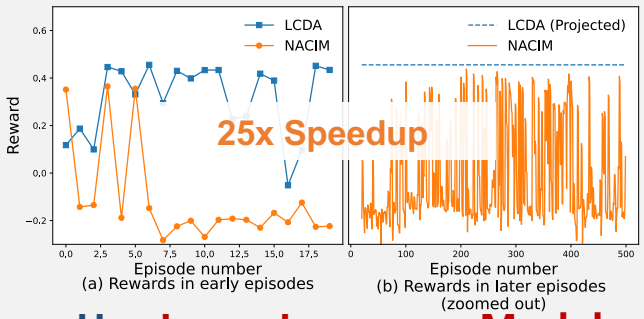
- Cold start: Random initialization
- Search space explosion

Cold Start

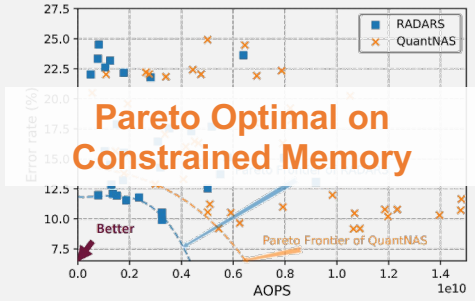


Search Space Explosion

## ❑ Dealing with These Issues




Use Large Language Models

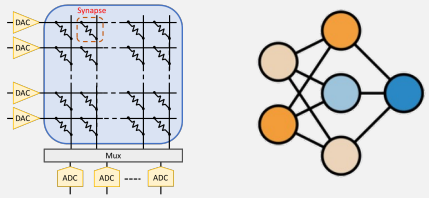



Combine RL with Differentiable Methods

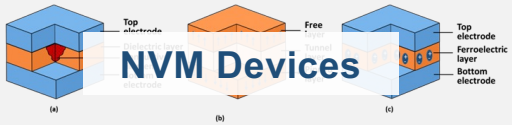
[10] Z. Yan, Y. Qin, X. S. Hu, and Y. Shi, "On the viability of using llms for sw/hw co-design: An example in designing cim DNN accelerators," SoCC 2023
[11] Z. Yan, W. Jiang, X. S. Hu, and Y. Shi, "Radars: Memory efficient reinforcement learning aided differentiable neural architecture search," ASP-DAC 2022

# Our Solution: Summary

## Background: CiM for DNN



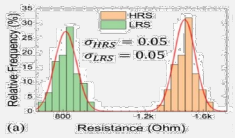


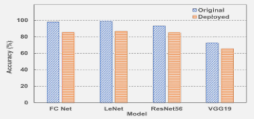

RRAM MRAM FeFET

## Problem:

**Device Variation & Acc. Drop**




Data Difference




Accuracy Drop

## Solution: Cross-Layer Co-Design

| | |
|---|---|
| Device | Detect Sensitive Devices<br>10x Protection |
| Circuit/ Arch | Worst-Case Analysis<br>100% Error |
| | AutoML<br>10x Speedup |
| Software | Train Robust DNN models<br>↑↑↑ 33% Acc |

**Co-Design**

- Object Detection ↓↓↓ 10x EDP
- Img Segmentation ↑↑↑ 40% Fairness
- Img Classification 25x Speedup

# Outline

- Introduction: Crossbar-based Hardware and their Robustness Issues
- Remedy Methods: Cross-Layer Co-Design
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- **Outlook & Conclusions**

# Outlooks

## ❑ Incremental Future Works

- **Using LLM to improve HW-SW Co-Design for robustness**
- **Accommodating SWIM to more types of devices**

## ❑ Future Directions

- **Hardware backdoors for CiM platforms**
- **Physical verifications for CiM techniques**
- **Mix-precision designs for robust DNN models**


**Hardware Backdoor**

# Conclusions

- Introduction: Crossbar-based Hardware and their Robustness Issues
- Remedy Methods: Cross-Layer Co-Design
  - Device: Device Programming Techniques
  - Circuit/Arch: Worst-case Analysis
  - Software: HW-Aware Training
  - Co-Design: HW-SW Co-Design Algorithm
- Outlook & Conclusions

# References


1. **Z. Yan**, X. S. Hu, and Y. Shi, "Swim: Selective write-verify for computing-in-memory neural accelerators," 2022 59th ACM/IEEE Design Automation Conference (**DAC**)
2. **Z. Yan**, X. S. Hu, and Y. Shi, "Computing in memory neural network accelerators for safety-critical systems: Can small device variations be disastrous?" 2022 International Conference on Computer-Aided Design (**ICCAD**)
3. **Z. Yan**, Y. Qin, X. S. Hu, and Y. Shi, "Improving realistic worst-case performance of nvcim dnn accelerators through training with right-censored gaussian noise," 2023 International Conference on Computer-Aided Design (**ICCAD) (Best Paper Award**)
4. **Z. Yan**, W. Jiang, X. S. Hu, and Y. Shi, "Radars: Memory efficient reinforcement learning aided differentiable neural architecture search," in 2022 27th Asia and South Pacific Design Automation Conference (**ASP-DAC**)
5. **Z. Yan**, D.-C. Juan, X. S. Hu, and Y. Shi, "Uncertainty modeling of emerging device based computing-in-memory neural accelerators with application to neural architecture search," in 2021 26th Asia and South Pacific Design Automation Conference (**ASP-DAC**)
6. **Z. Yan**, Y. Shi, W. Liao, M. Hashimoto, X. Zhou, and C. Zhuo, "When single event upset meets deep neural networks: Observations, explorations, and remedies," in 2020 25th Asia and South Pacific Design Automation Conference (**ASP-DAC**)
7. **Z. Yan**, X. S. Hu, and Y. Shi, "On the reliability of computing-in-memory accelerators for deep neural networks," in System Dependability and Analytics: Approaching System Dependability from Data, System and Analytics Perspectives
8. **Z. Yan**, Y. Qin, X. S. Hu, and Y. Shi, "On the viability of using llms for sw/hw co-design: An example in designing cim DNN accelerators," in Proceedings of the 36th IEEE International System-on-chip Conference
9. L. Yang, **Z. Yan**, M. Li, et al., "Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks," in 2020 57th ACM/IEEE Design Automation Conference (DAC), IEEE, 2020
10. W. Jiang, Q. Lou, **Z. Yan**, et al., "Device-circuit-architecture co-exploration for computing-in-memory neural accelerators," IEEE Transactions on Computers, 2020

# Thank You & Questions

# Toward Robust Neural Network Computation on Emerging Crossbar-based Hardware and Digital Systems

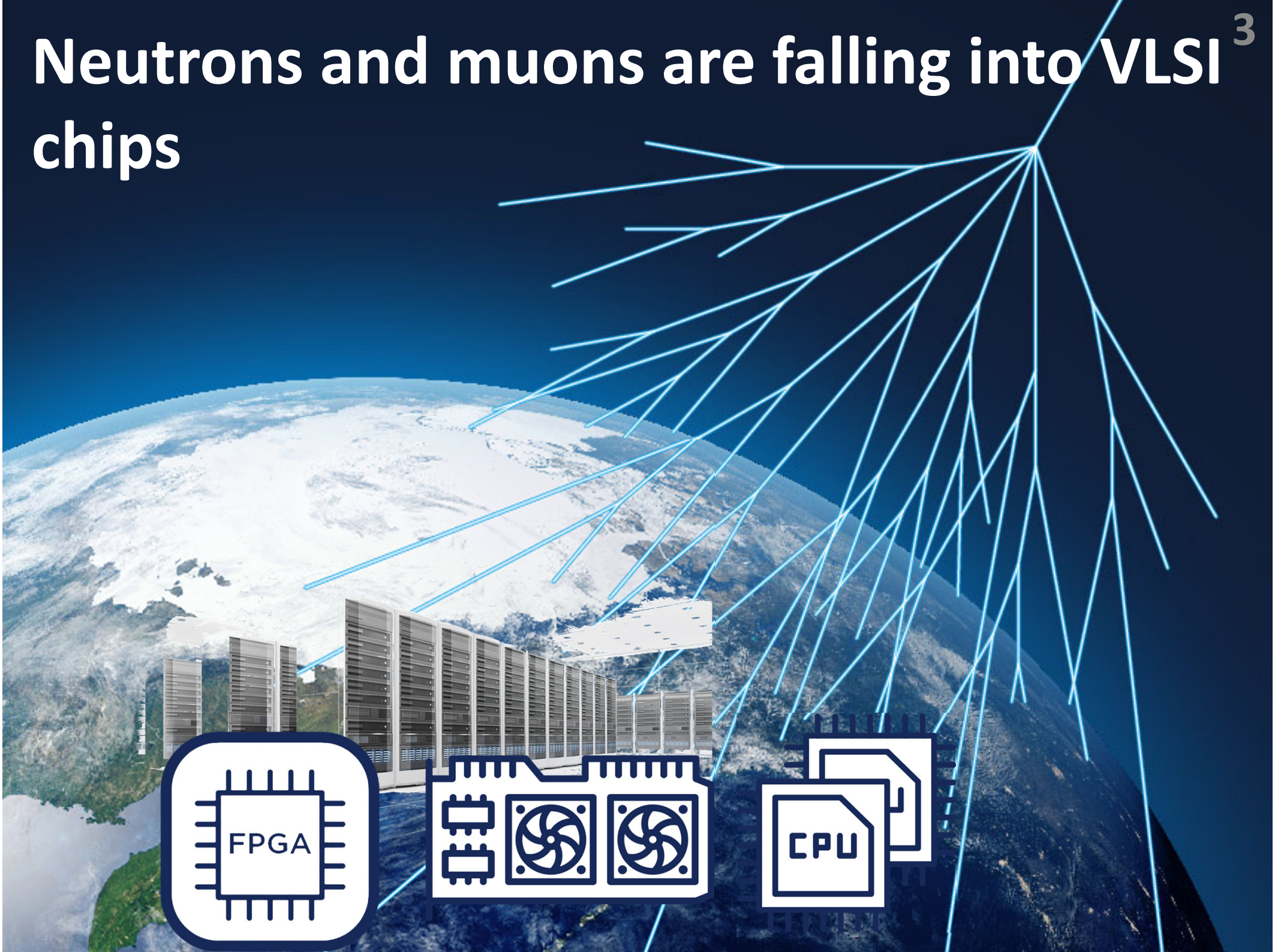
**Masanori Hashimoto**
Dept. Informatics, Kyoto University
hashimoto@i.kyoto-u.ac.jp
Jan. 22, 2024

# Abstract

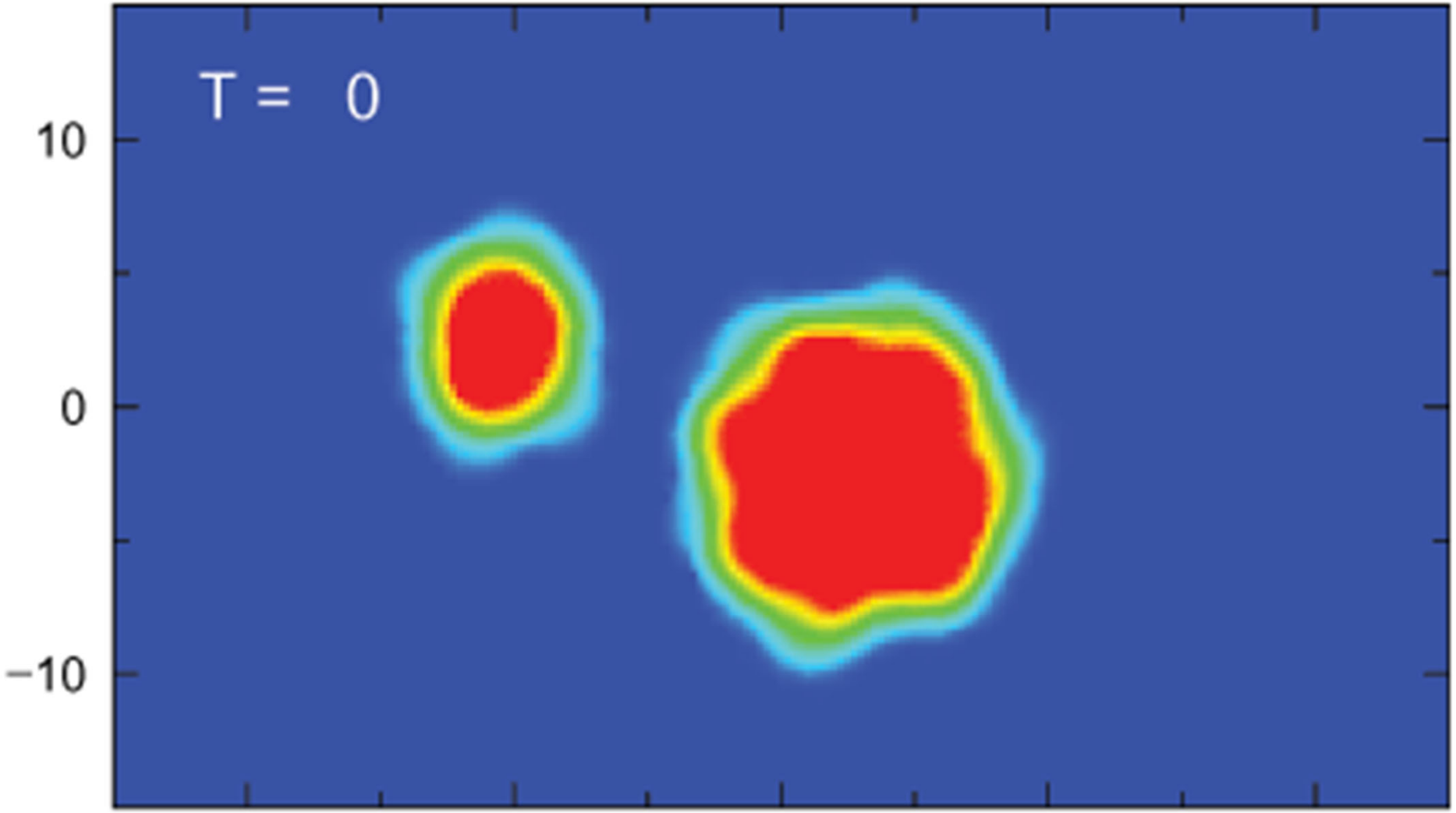
Deep Neural Networks (DNNs) are currently operated on GPUs in both cloud servers and edge-computing devices, with recent applications extending to safety-critical areas like autonomous driving. Accordingly, the reliability of DNNs and their hardware platforms is garnering increased attention. This talk will focus on soft errors, predominantly caused by cosmic rays, a major error source during an intermediate device's lifetime. While DNNs are inherently robust against bit flips, these errors can still lead to severe miscalculations due to weight and activation perturbations, bit flips in AI accelerators, and errors in their interfaces with microcontrollers, etc. The latter part of this tutorial will discuss:

- Identification of vulnerabilities in neural networks,
- Reliability analysis and enhancement of AI accelerators for edge computing,
- Reliability assessment of GPUs against soft errors.

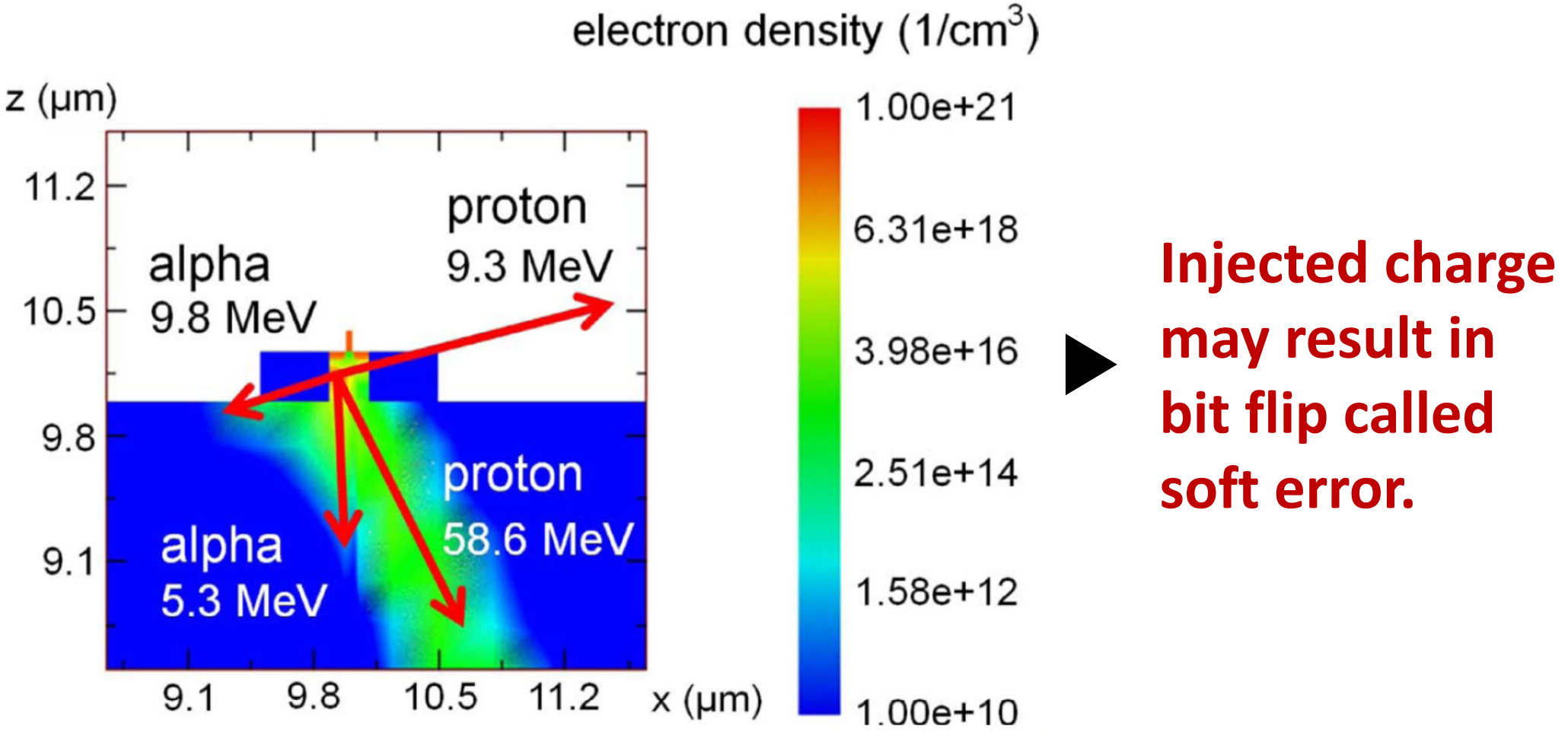
# Neutrons and muons are falling into VLSI chips



# Example of nuclear reaction



# Example of reaction in VLSI chip




**Injected charge may result in bit flip called soft error.**

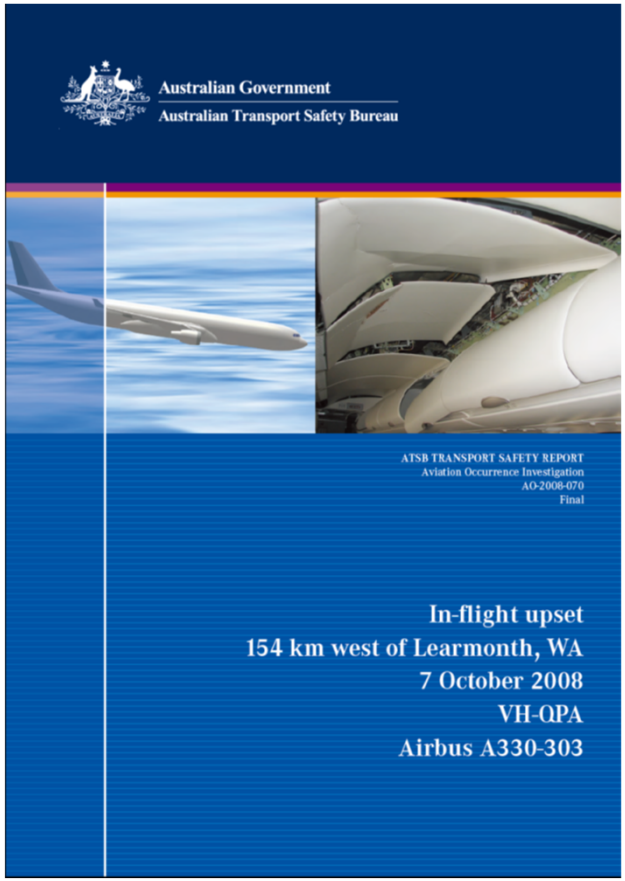
Fig. 3. Cross section view of the initial charge distribution by the nuclear reaction, $n + {}^{28}Si \rightarrow 3n + 2p + 2\alpha + {}^{16}O$, at the incident energy of 233 MeV. The track of ${}^{16}O$ ion with its kinetic energy of 4.23 MeV is not depicted because the direction of motion is nearly parallel to the y-axis [1].

[1] S. Abe, et. al, "Multi-scale Monte Carlo simulation of soft errors using PHITS-HyENEXSS code system," *IEEE Trans. Nuclear Science*, 2012

# Incident in aircraft (Oct. 2008)

A steep dive due to fly-by-wire system failure

- 1/3 customers and 3/4 crews injured




https://www.atsb.gov.au/media/3532398/ao2008070.pdf

**Table 26: Evaluation of potential triggers**

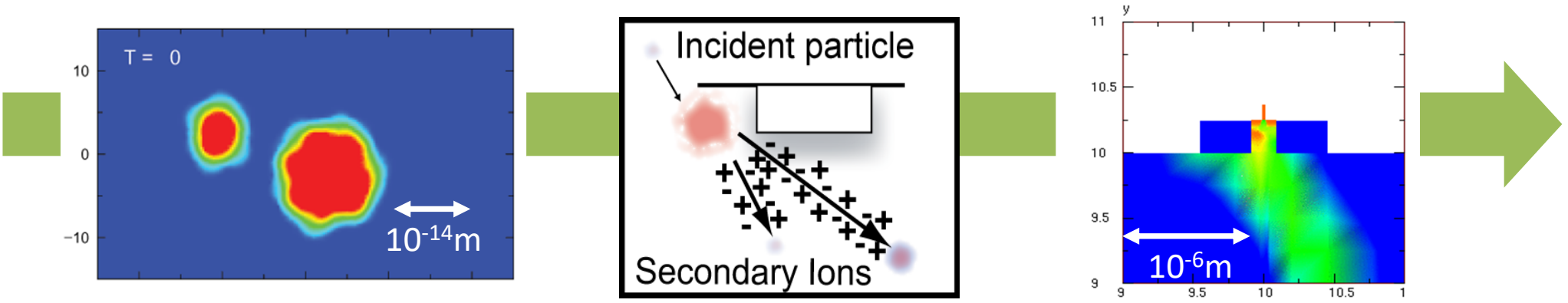
| Trigger | Key points | Assessment |
|---|---|---|
| Software corruption | ADIRU software was verified as intact after the occurrences.<br>Unit 4167's software was reloaded and verified between the two occurrences involving this unit. | Very unlikely |
| Software 'bug' | Would not be expected to occur twice on one unit without many other occurrences on other units.<br>Functional testing of software found no problems.<br>No unique circumstances identified with the occurrence flights that could trigger a rare bug. | Very unlikely |
| Hardware fault | Extensive unit and module testing found no problems.<br>Visual examination of the units did not identify any physical damage or other abnormalities.<br>Not consistent with a 'soft fault'. | Very unlikely |
| Physical environment | Unit testing beyond relevant standards found no problems.<br>Visual examination of the units did not identify any physical damage or other abnormalities that could result in a relevant equipment fault when exposed to normal or abnormal environmental conditions.<br>The physical environment was normal during the three flights.<br>Nothing unusual found with aircraft environment during testing. | Very unlikely |
| EMI from aircraft systems | Extensive unit testing found no problems.<br>Measurement of the electromagnetic environment within the aircraft during ground and flight tests showed nothing unusual or excessive.<br>It was not possible to reproduce the exact conditions of the occurrence flights during testing.<br>Wiring integrity tests found no problems.<br>The aircraft configuration was not unique or unusual.<br>No problems with the other ADIRUs installed on same aircraft. | Unlikely |
| EMI from other onboard sources | No sources of concern were identified.<br>Extensive unit testing found no problems.<br>Measurement within the aircraft while PEDs were in use showed very minor effects on the electromagnetic environment. | Very unlikely |
| EMI from external sources | No sources of concern were identified.<br>Extensive unit testing found no problems.<br>The electromagnetic environment during flight tests showed nothing unusual or excessive.<br>No problems with other systems during the occurrence flights. | Very unlikely |
| SEE | The intensity of high-energy particles for the three occurrences was not unusual.<br>The ADIRU had limited mechanisms to detect and manage SEE (that is, no EDAC).<br>No SEE testing was performed on the occurrence units.<br>SEE testing on another unit did not induce the data-spike failure mode (although the testing was limited in scope).<br>Difficult to accurately estimate the likelihood of two SEEs occurring on the same ADIRU twice in its operational life. | Insufficient evidence to estimate likelihood |

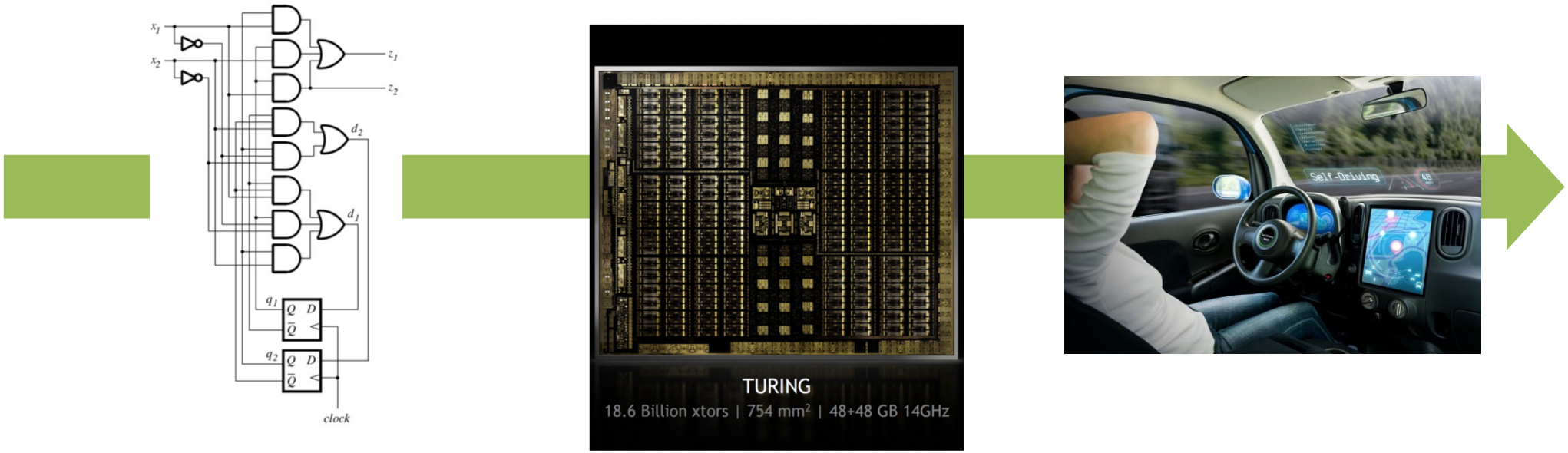
**Other factors (Very) unlikely**

**Soft error**
**Insufficient evidence since reproduction is very difficult**

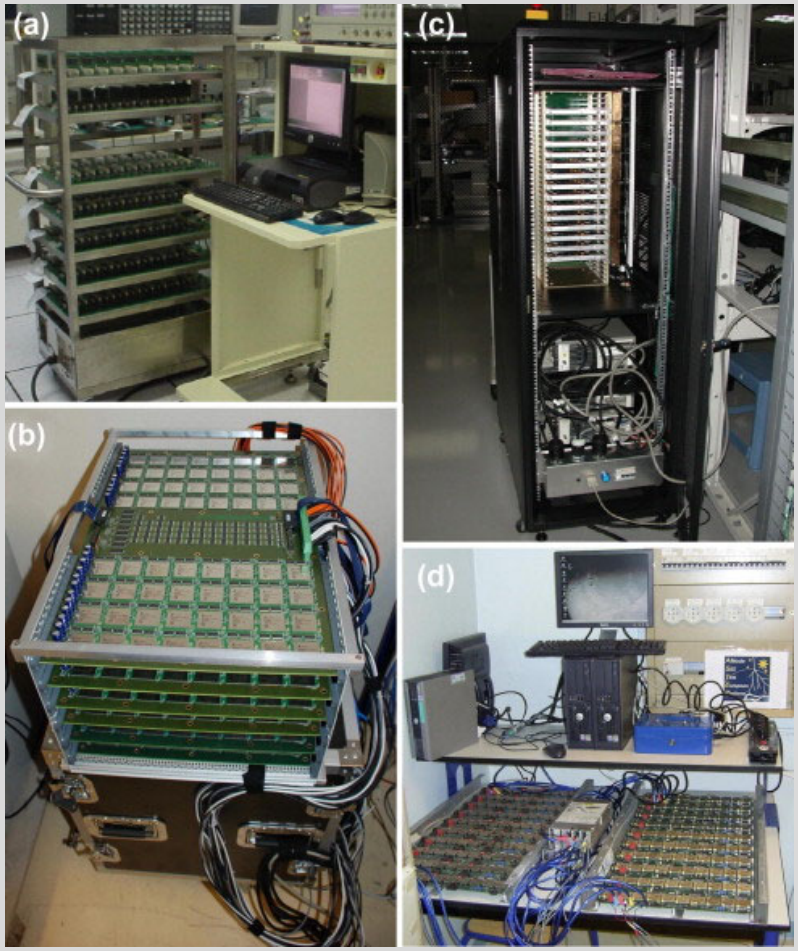
# Multi-physics multi-layer phenomena with diverse temporal and spatial scales

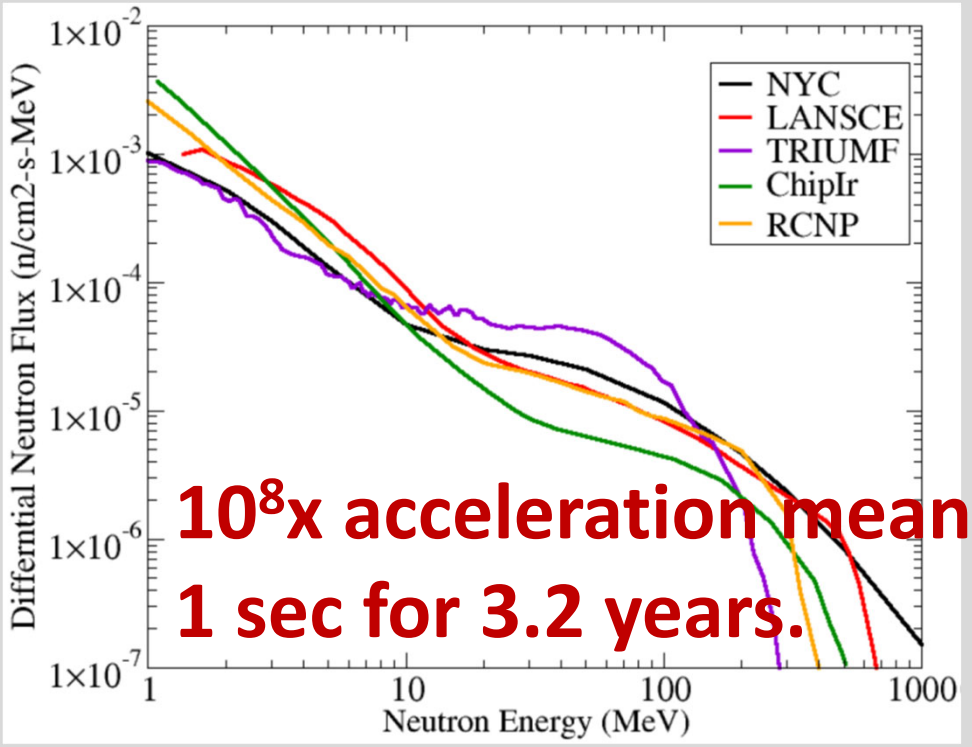




# Real-time & accelerated test




**$10^8$x acceleration means 1 sec for 3.2 years.**

Many devices are operated

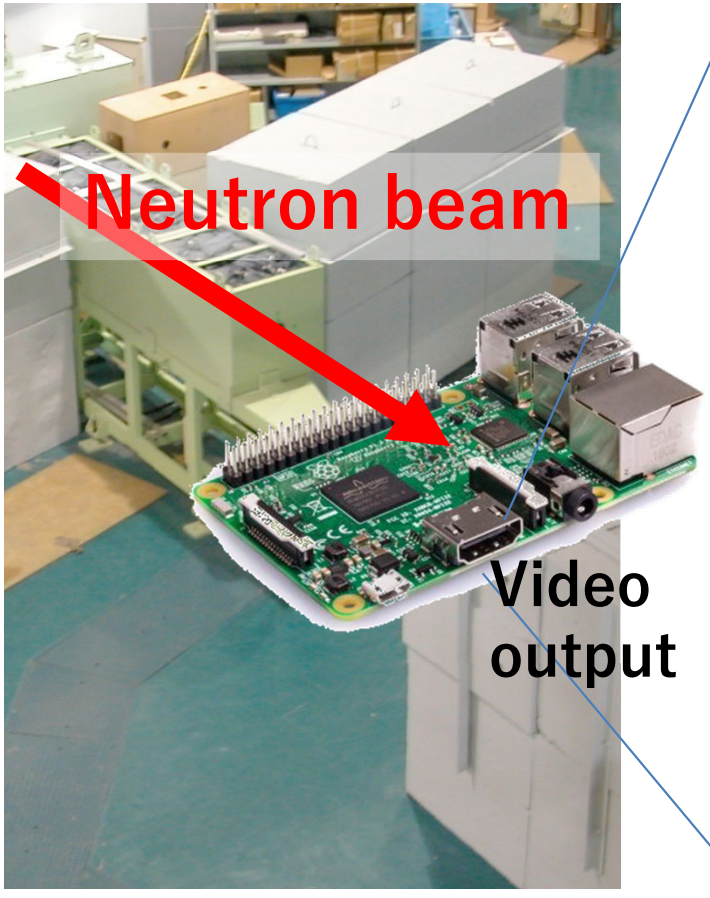
- Months to years are necessary to get enough # of errors

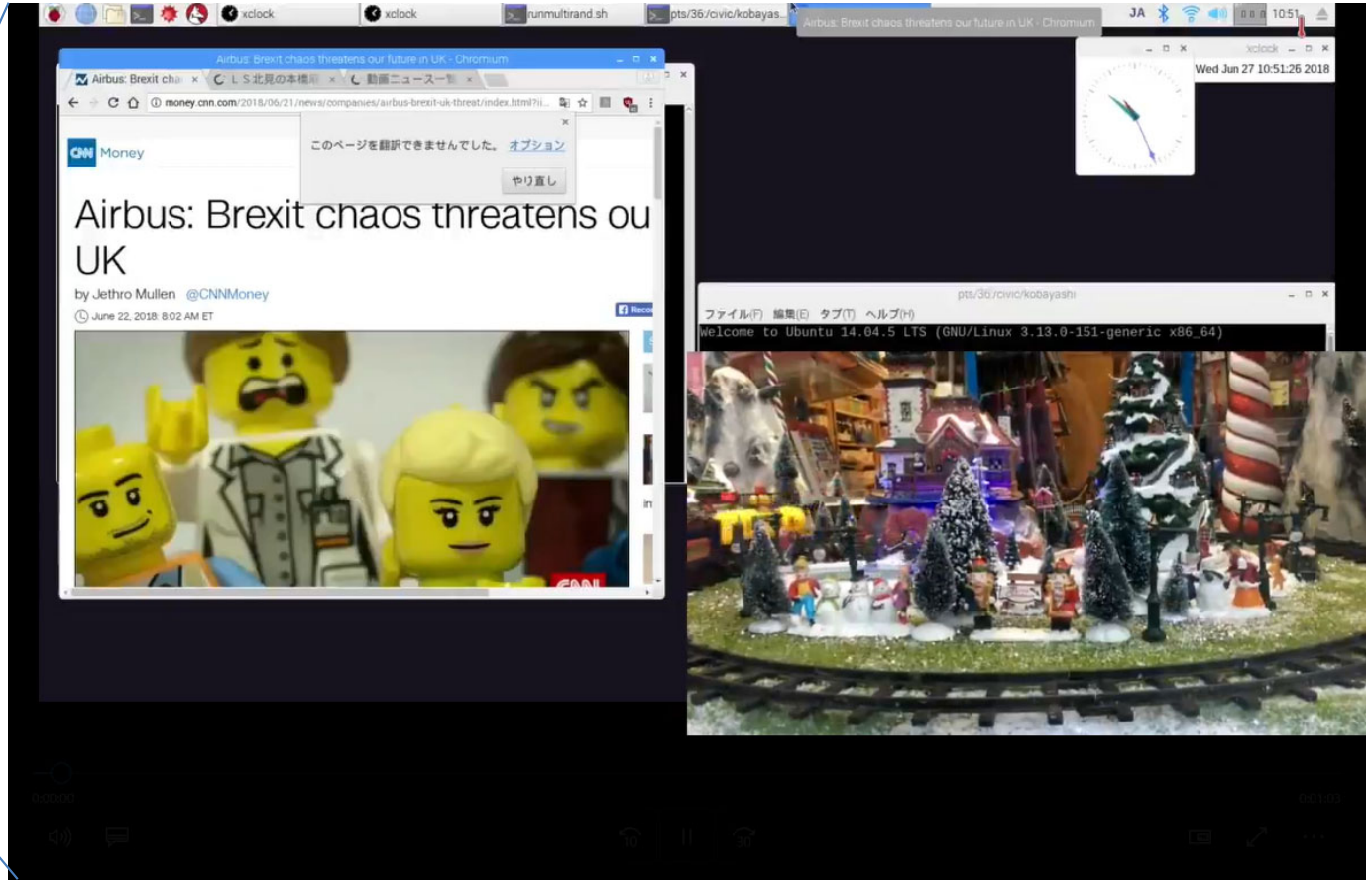
https://ars.els-cdn.com/content/image/1-s2.0-S0026271414000882-gr3.jpg

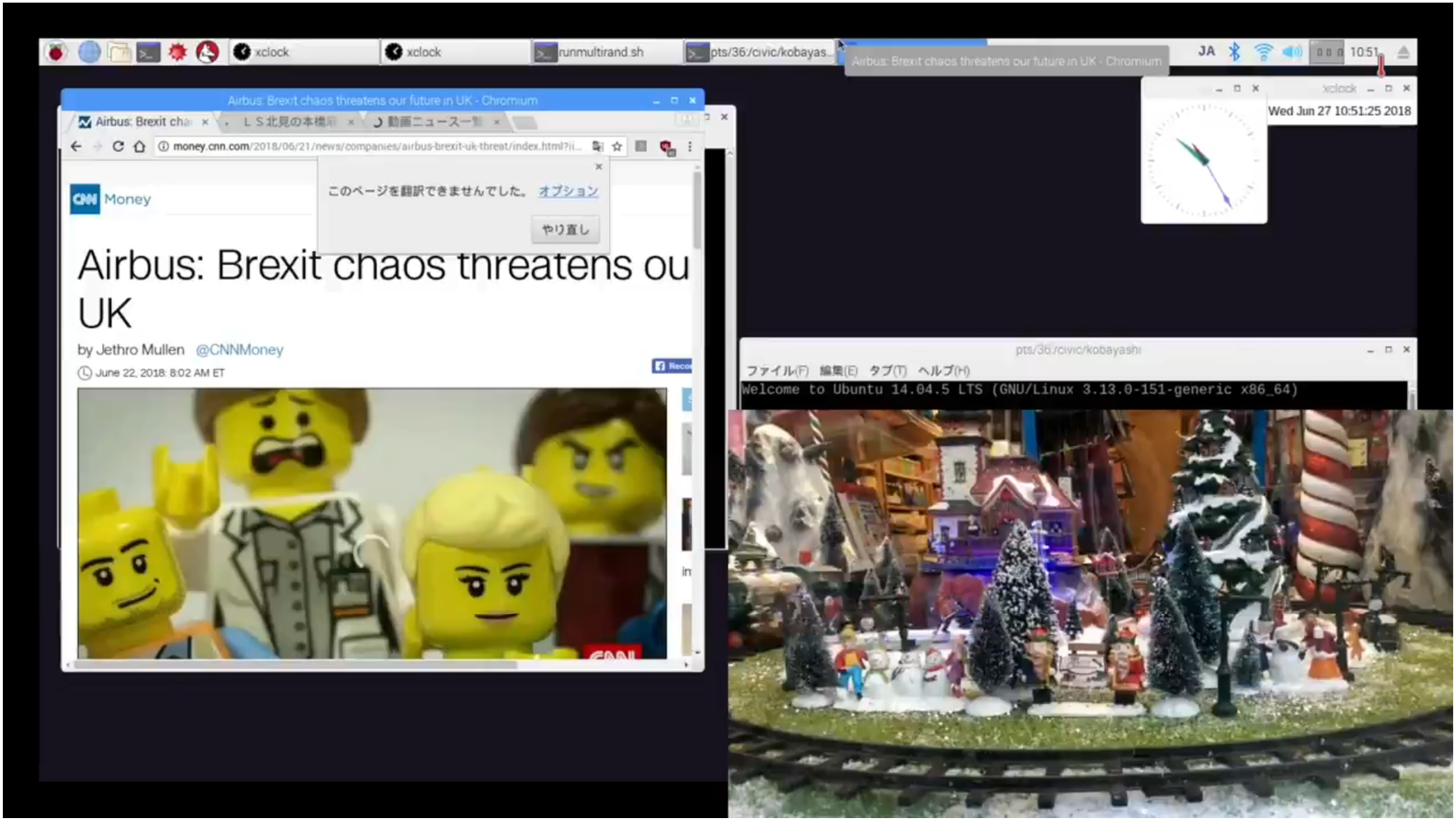
# Demonstration

Linux is running on Raspberry Pi

Courtesy to Prof. Kobayashi, Kyoto Institute of Technology.






Courtesy to Prof. Kobayashi, Kyoto Institute of Technology.

# Our life depends on AI applications running on integrated systems

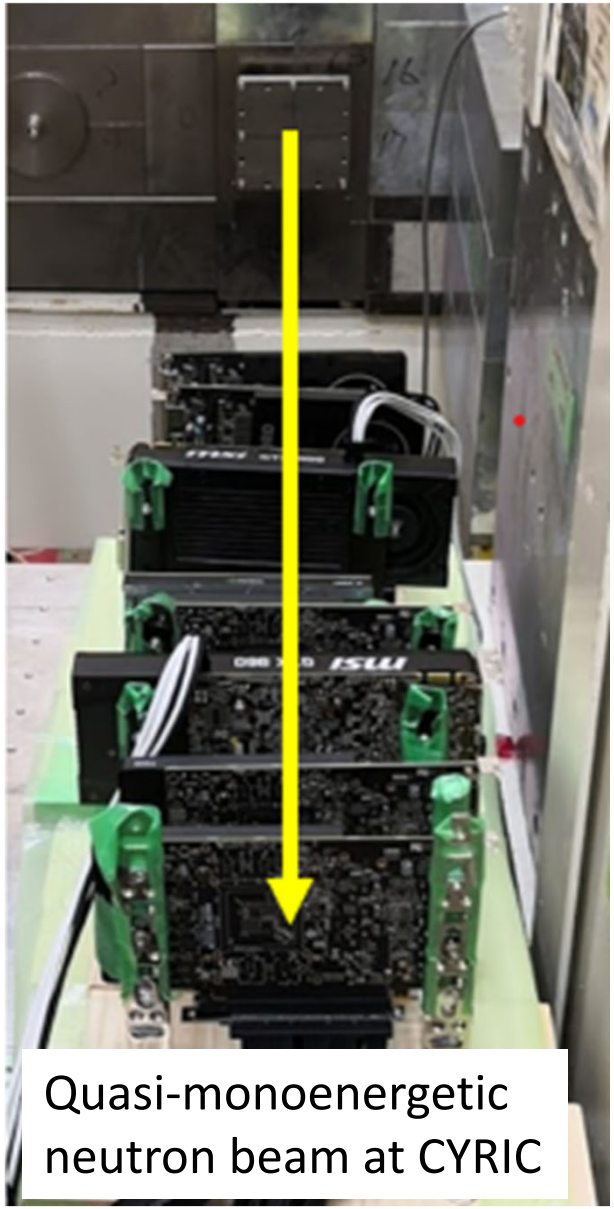

http://rtc.nagoya.riken.jp/ROBEAR/

**High reliability is demanded for AI-based safety-critical applications.**

**Rad-hard components sufficiently powerful to execute DNNs are not available, yet.**

# Preliminary experiment irradiating object detection running on GPUs

- Yolov3-tiny
- GPU cards
  - NVIDIA Quadro P2000
  - NVIDIA GeForce GTX960
  - Aligned in series on the beam track



Quasi-monoenergetic neutron beam at CYRIC

Y. Zhang, K. Ito, H. Itsuji, T. Uezono, T. Toba and M. Hashimoto, "Fault Mode Analysis of Neural Network-based Object Detection on GPUs with Neutron Irradiation Test," *RADECS*, 2020.

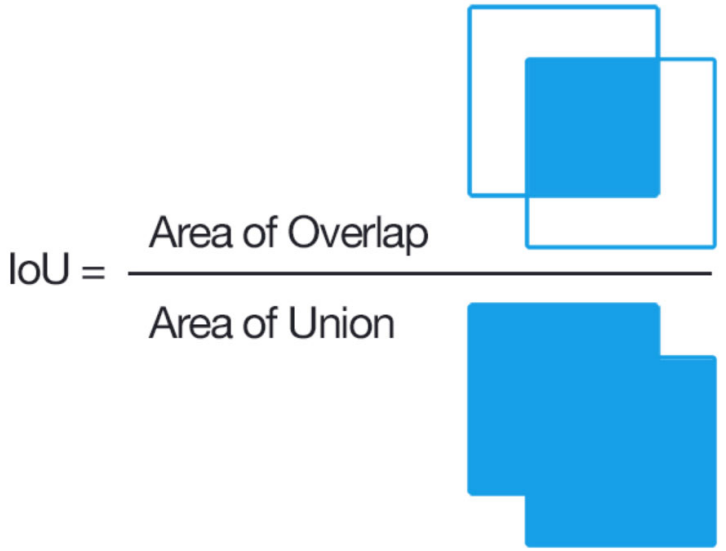
# Definitions of DUE, SDC and critical SDC

Impact of soft error on computation includes

- wrong computation result (**SDC**; Silent Data Corruption)
  - Harmful for all applications
- hang or halt (**DUE**; Detectable Uncoverable Error)
  - Harmful for real-time applications
- no effect (Mask)
  - Depends on both hardware and software

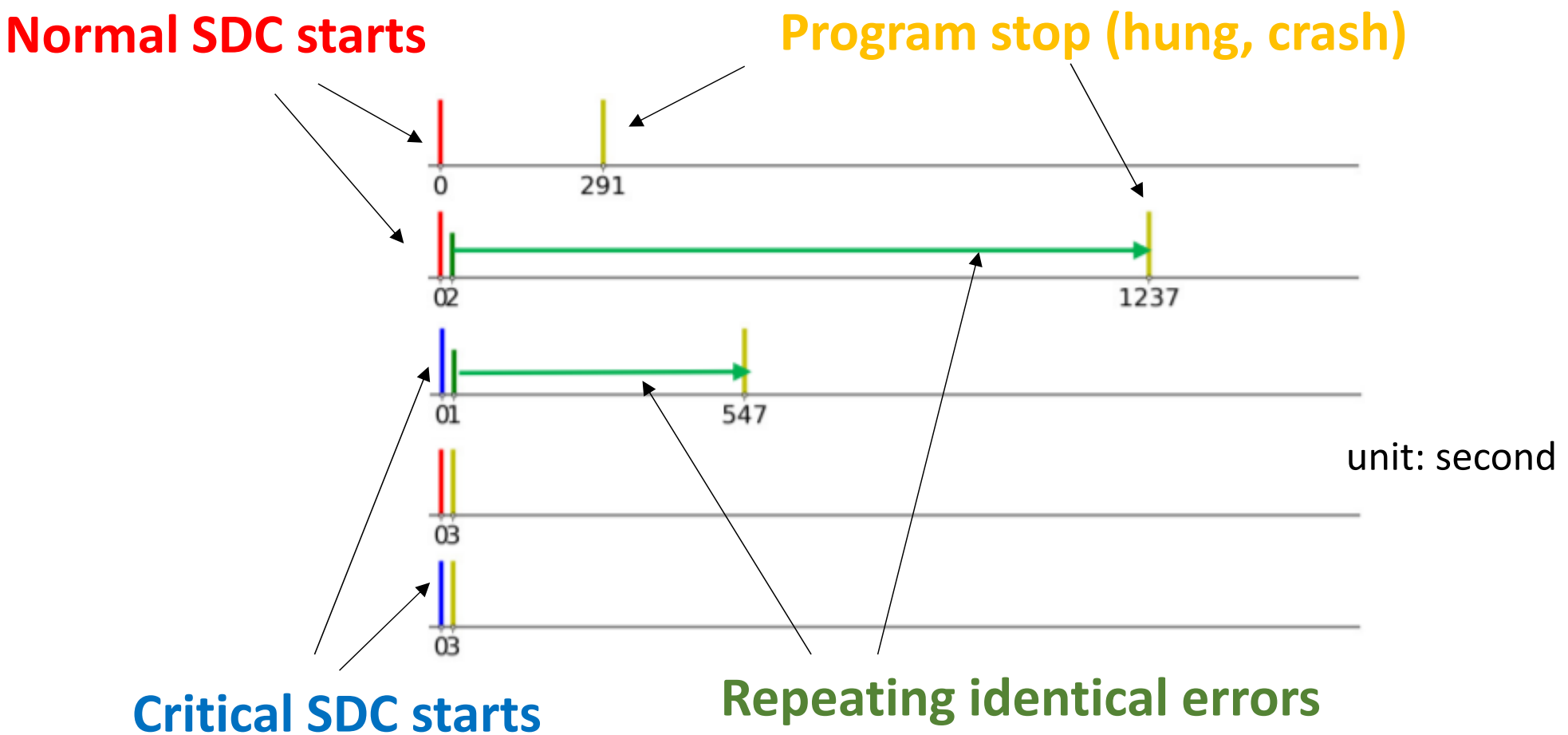
# Critical SDC in Object Detection

- SDCs that are critical to object detection
- IoU (intersection-over-union) is used to evaluate critical SDC
  - ***IoU*** : IoU of faulty and golden output
  - ***IoU*** > thresh: normal SDC (thresh: 0.8 in exp.)
  - ***IoU*** < thresh: critical SDC



# Temporal patterns of observed SDCs

- Each row corresponds to one sequence of error occurrence
- Some faulty outputs last for hundreds of seconds probably due to weight data corruption



# Fault mode categorization

Classify errors into two-by-two categories:

- Identical errors repeat or not
- SDC critical or not

# of faulty events

| Category | Identical errors | Variant errors |
|---|---|---|
| Critical SDC | **2** | **2** |
| Non-critical | **8** | **6** |

**Not all SDCs are critical.**

NNs are inherently redundant and robust to parameter perturbation.

**Error rate depends on underlaying hardware.**

# Current research status

- Some data in literature suggests radiation impact on DNN is so high, hindering safe large-scale use.
- COTS AI products exhibit a high error rate due to radiation [2][3], attributed to their large size and critical resource density.
- Effective hardening strategies against radiation is necessary.

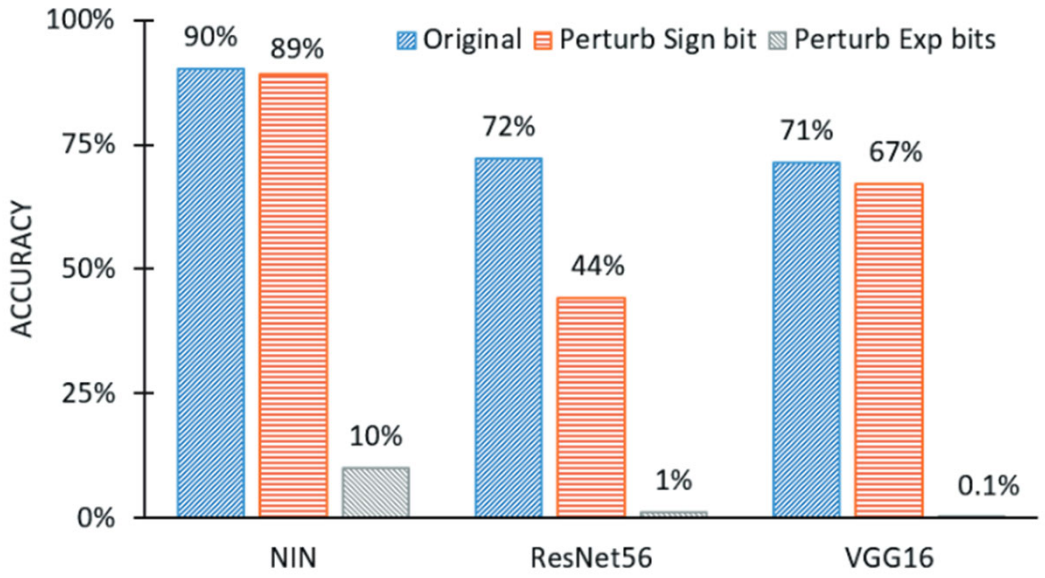
[2] Y. Ibrahim, H. Wang, M. Bai, Z. Liu, J. Wang, Z. Yang, and Z. Chen,"Soft error resilience of deep residual networks for object recognition," *IEEE Access*, 2020.
[3] D. A. G. Goncalves de Oliveira, L. L. Pilla, T. Santini, and P. Rech, "Evaluation and mitigation of radiation-induced soft errors in graphics processing units," *IEEE Trans. Computers*, 2016.

# Agenda

- **Robustness of NNs**
  - **Case study (FP)**
  - Identifying vulnerable weight parameters
  - Quantization
    - Multi-bit-width neural networks
- Robustness of hardware
  - Edge AI accelerator
  - GPU
- Countermeasures in literature

# NN robustness evaluation (FP case)

- **DNN robustness is important for soft error, hard error and security.**
  - **Malicious attack to DNN is another concern.**
- Maximum impact of single event upset in network parameters.
  - Among all the parameters, only one bit of one parameter is with fault and the others are fault-free.




Z. Yan, Y. Shi, W. Liao, M. Hashimoto, X. Zhou and C. Zhuo, "When Single Event Upset Meets Deep Neural Networks: Observations, Explorations, and Remedies," *ASP-DAC*, 2020.

# Observations in ResNet56

- Observation 1: the highest exponent bit has the highest impact across different layers while fraction bits have is very limited impact.

$$x = (-1)^{Sign} \times (1 + Fraction)^{Exponent - Bias}$$

- Observation 2: the first layer, which directly deals with the input stream, has higher impact.

**Maximum accuracy drop in ResNet56**

| $SSIPP$ | Input | Stack 1 | Stack 2 | Stack 3 | FC |
|---|---|---|---|---|---|
| Sign | 28.09% | 6.43% | 2.08% | 0.27% | 0.90% |
| Ex1 | 70.19% | 70.19% | 70.19% | 70.19% | 70.19% |
| Frac1 | 0.43% | 0.29% | 0.41% | 0.25% | 0.30% |

# Results in other networks

- Impact of bits are:
  exponent > sign >> fraction
- Impact of sign bit varies layer by layer.

| Layer | NIN/Sign W | NIN/Sign B | NIN/Ex W | NIN/Ex B | NIN/Fr W | NIN/Fr B | Res56/Sign W | Res56/Sign B | Res56/Ex W | Res56/Ex B | Res56/Fr W | Res56/Fr B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1.0% | 2.2% | 80% | 80% | 0.2% | 0.4% | 28.1% | 16.9% | 70% | 70% | 0.4% | 0.4% |
| 2 | 0.2% | 4.4% | 80% | 80% | 0.1% | 0.2% | 6.4% | 1.2% | 70% | 70% | 0.3% | 0.2% |
| 3 | 0.3% | 2.4% | 80% | 80% | 0.2% | 0.1% | 2.1% | 1.2% | 70% | 70% | 0.4% | 0.5% |
| 4 | 0.1% | 0.1% | 80% | 80% | 0.1% | 0.1% | 0.3% | 1.1% | 70% | 70% | 0.3% | 0.4% |
| 5 | 0.3% | 0.2% | 80% | 80% | 0.2% | 0.1% | - | - | - | - | - | - |
| Last | 2.4% | 0.2% | 80% | 80% | 0.7% | 0.1% | 1.0% | 0.1% | 70% | 70% | 0.3% | 0.1% |

| Layer | VGG16/Sign W | VGG16/Sign B | VGG16/Ex W | VGG16/Ex B | VGG16/Fr W | VGG16/Fr B |
|---|---|---|---|---|---|---|
| 1 | 4.2% | 2.4% | 70% | 70% | 0.9% | 0.9% |
| 2 | 0.1% | 0.4% | 70% | 70% | 0.1% | 0.2% |
| 3 | 0.1% | 0.1% | 70% | 70% | 0.1% | 0.1% |
| 4 | 0.1% | 0.1% | 70% | 70% | 0.1% | 0.1% |
| 5 | 0.1% | 0.1% | 70% | 70% | 0.1% | 0.1% |
| Last | 0.0% | 0.0% | 70% | 70% | 0.1% | 0.1% |

# Agenda

- **Robustness of NNs**
  - Case study (FP)
  - **Identifying vulnerable weight parameters**
  - Quantization
    - Multi-bit-width neural networks
- Robustness of hardware
  - Edge AI accelerator
  - GPU
- Countermeasures in literature

# Fault injection is too time-consuming

- Fault injection(FI) is a common method for estimating vulnerability of a network due to nonlinearity of NN. However, FI costs time prohibitively because NN has too many parameters.
- Contributions:  propose constructing a vulnerability model (VM) to predict vulnerability of DNN with fewer FIs in an acceptable time.
  - FI reproduces a bit flip supposing  soft error and malicious attack.

Y. Zhang, H. Itsuji, T. Uezono, T. Toba and M. Hashimoto, "Estimating Vulnerability of All Model Parameters in DNN with a Small Number of Fault Injections," *DATE*, 2022.
Y. Zhang, H. Itsuji, T. Uezono, T. Toba, M.Hashimoto, "Vulnerability Estimation of DNN Model Parameters with Few Fault Injections," *IEICE Trans. Fundamentals*, 2023.

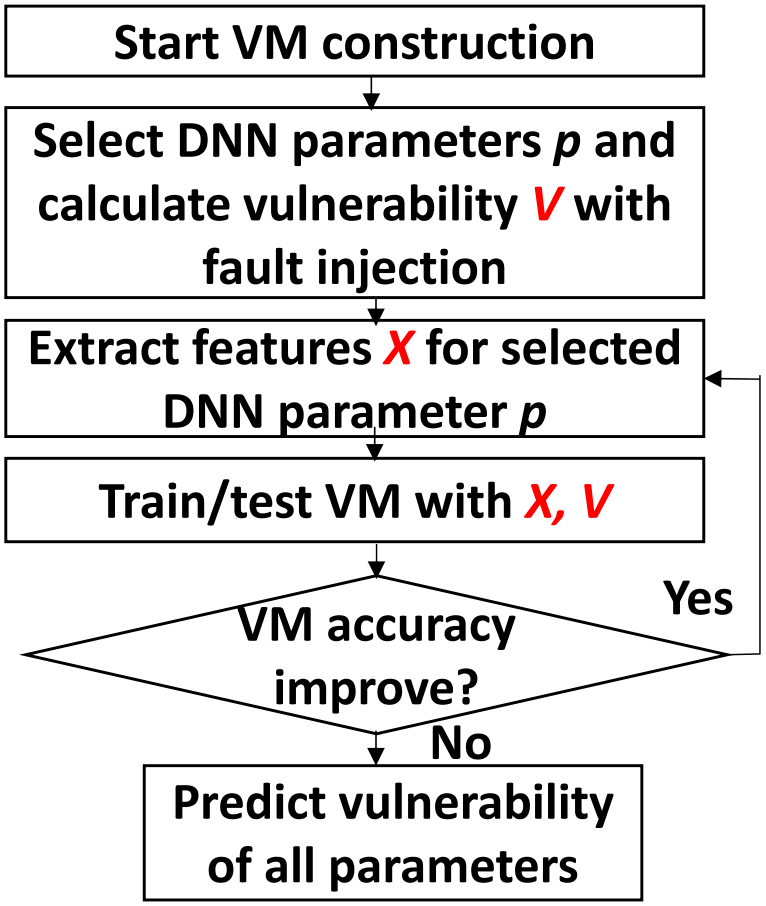
# Features of proposed VM construction

**Machine learning is used to construct VM.**

- Vulnerability definition:
  - sum of accuracy degradation for individual bit flips.
- Feature identification :
  - e.g., absolute value of parameters, gradient, calculation times, etc.
- Fast training:
  - Only conduct FI on important bits, e.g., exponent bits with value 0.
  - Iterative training to prepare minimum FI data for required accuracy

# VM construction flow

Input:
- Trained NN
- Image dataset




Output:
- Trained VM
- Vulnerability of all params.

Conventional method:
FI to **all/important** bits of **all parameters.**

Proposed method:
FI to **important bits of some parameters** and predict others' vulnerabilities

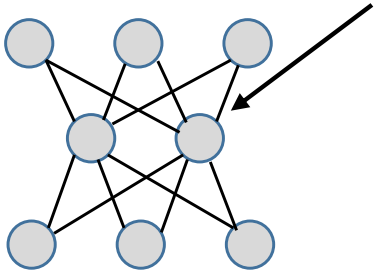
# Definition of vulnerability

$$V_i = \frac{1}{N_b}\sum_{j=1}^{N_b}(\Delta acc_{i,j})$$

$\Delta acc$: accuracy deviation between the original clean DNN and dirty DNN

$N_b$: the number of bits for vulnerability analysis in one DNN parameter

**For 32-bit floating point case:**



Neural network

parameter


sign exponent mantissa

$$V_i = \frac{1}{32}\left(\Delta acc + \sum_{j=1}^{8}\Delta acc + \sum_{j=1}^{23}\Delta acc\right)$$

# Efficient vulnerability calculation

- Approximation of vulnerability

$$V_i = \frac{1}{N_b}\sum_{j=1}^{N_b}(\Delta acc_{i,j}) \longrightarrow V_i' = \frac{1}{N_b}\sum_{j\in bits_i}(\Delta acc_{i,j})$$

$bits$: a set that may contain integer numbers from 1 to $N_b$


(a) 8 exponential bits of 32 bit floating-point


(b) 8 bit fixed-point positive number


(c) 8 bit fixed-point negative number

- $bits$ selection
  - Floating-point format
    - exponent bits whose values are 0.
  - Fixed-point format
    - Positive number: topmost ’1’ bit — ’0’ bits locating on the left side of the ‘1’ to ‘0’; the value change is at most 100%
    - Negative number: topmost ’0’ bits — ’1’ bits locating on the left side of the ‘0’ to ‘1’; the value change is at most 100%

# Feature extraction

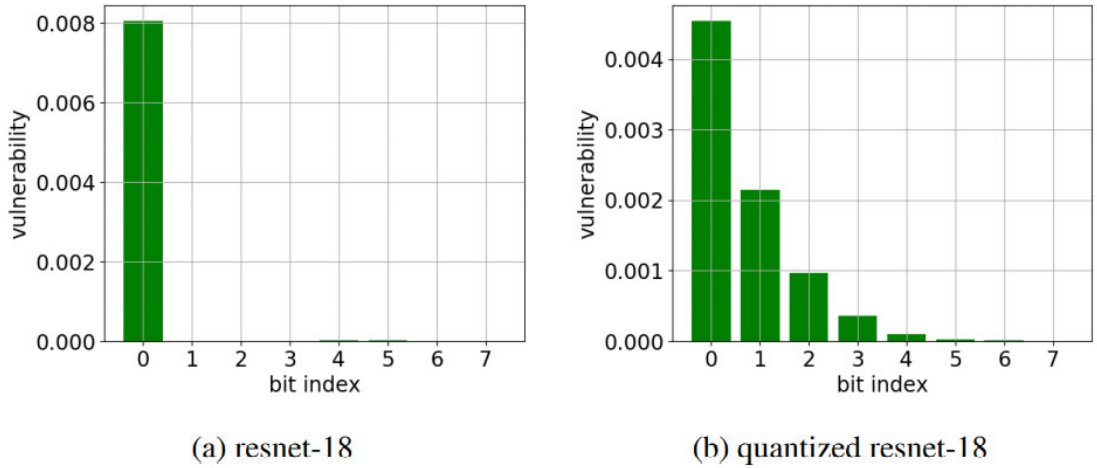
- Absolute value of param (A)
- Number of dangerous bits (D)
  - Number of $\boldsymbol{bits}$
- Gradient (G)
  - Larger gradient means larger impact on NN output
  - Available in NN training process
- Calculation time (CT)
  - How many times each param is used during one NN inference
- Layer location (ID, OD)
  - Location of each layer.

# Setup

- Networks
  - ResNet-18, quantized ResNet-18, yolov3-tiny
- Datasets
  - CIFAR10: ResNet-18, quantized ResNet-18
  - COCO: yolov3-tiny
- VM algorithm
  - Random forest
  - Definition of $accuracy$
    - Top-k accuracy: ResNet-18, quantized ResNet-18
    - Mean average precision (mAP): yolov3-tiny

# Validating $\boldsymbol{bits}$ selection

- Vulnerability distribution on different bits



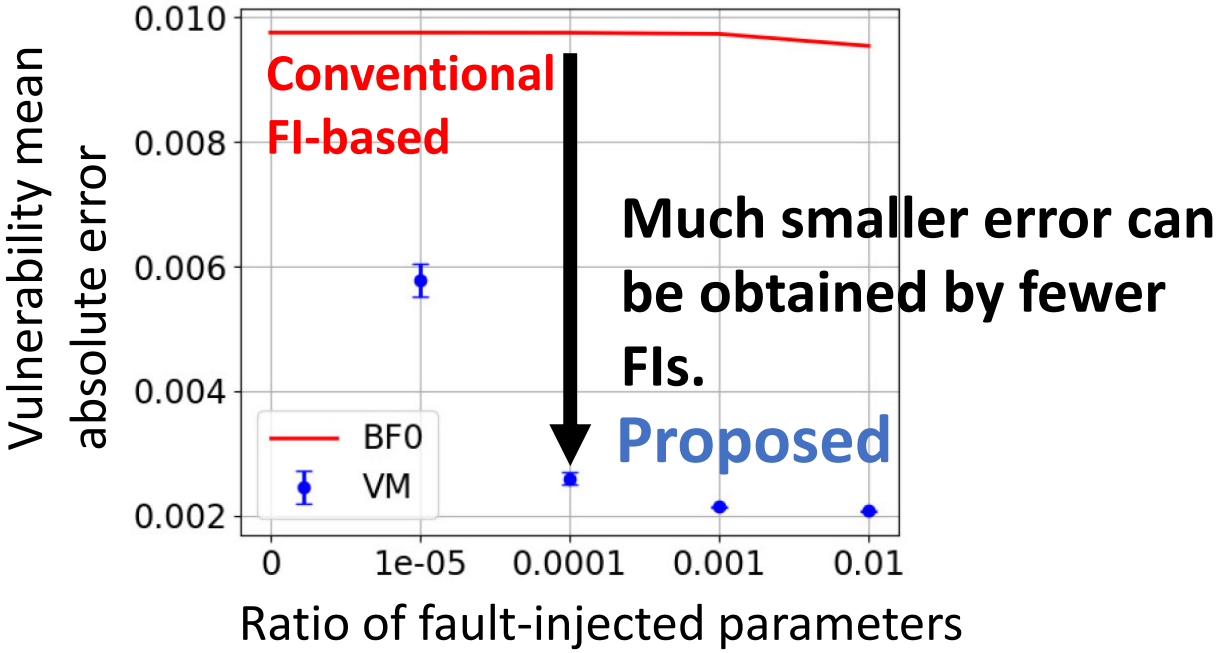

(a) resnet-18 (b) quantized resnet-18

*Vulnerability is centralized on MSB

- Assume for unimportant bits outside $\boldsymbol{bits}$, $\Delta acc = 0$
  - ResNet-18:
    - 99.9996% unimportant bits attain $\Delta acc$=0.
    - # of fault injection is reduced by 54.5%
  - Quantized ResNet-18:
    - 99.995% unimportant bits attain $\Delta acc$=0
    - # of fault injection reduces 27.1%

# Accuracy & time comparison with traditional FI

- VM can predict vulnerability accurately for resnet-18 and yolo-v3
- **>3000x speed-up** (733 to 0.21 hours) can be achieved compared with tradition FI.



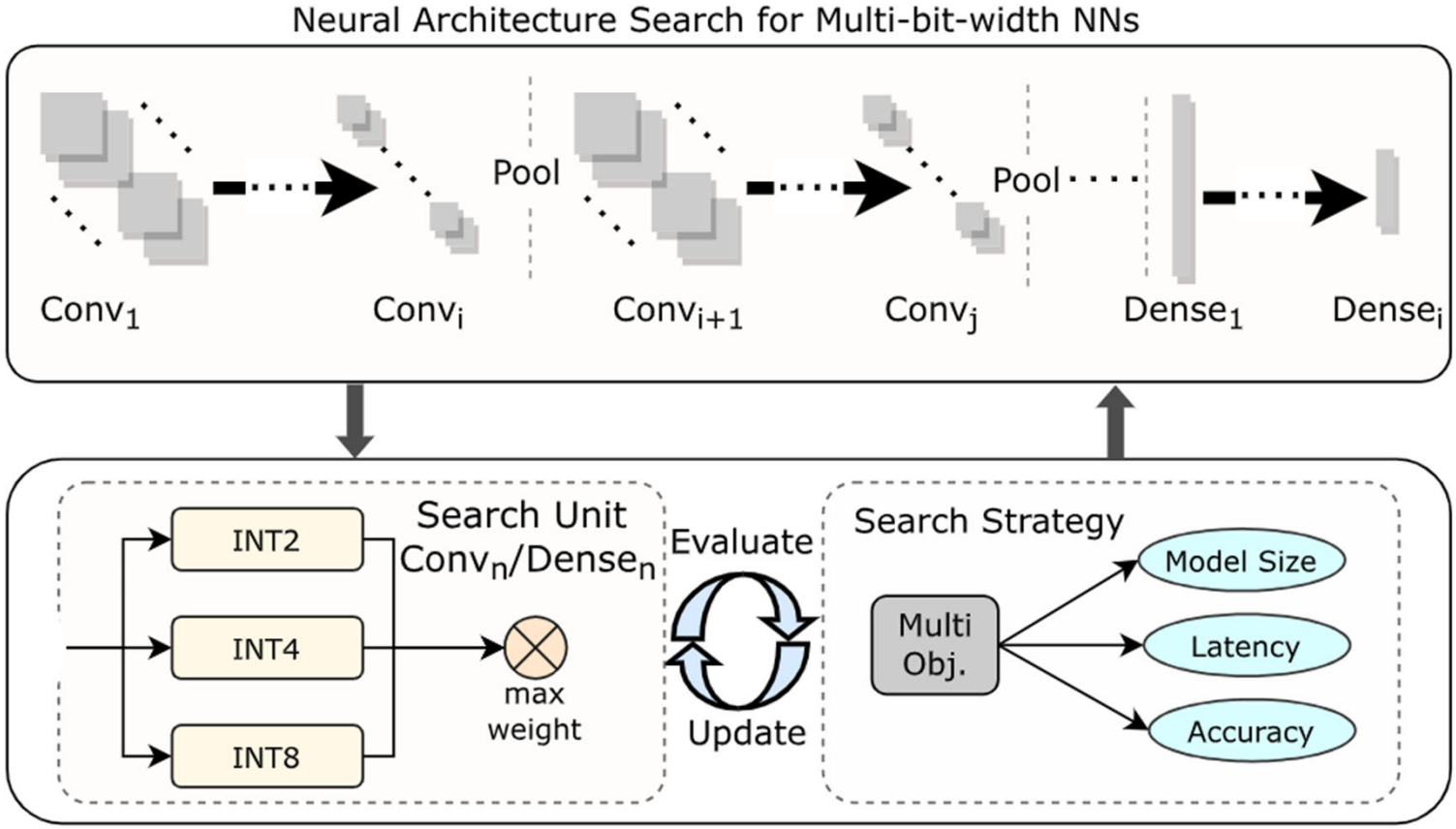

- Traditional fault injection: flip all bits for all parameters
- BF0: only flip important bits for all parameters

# Agenda

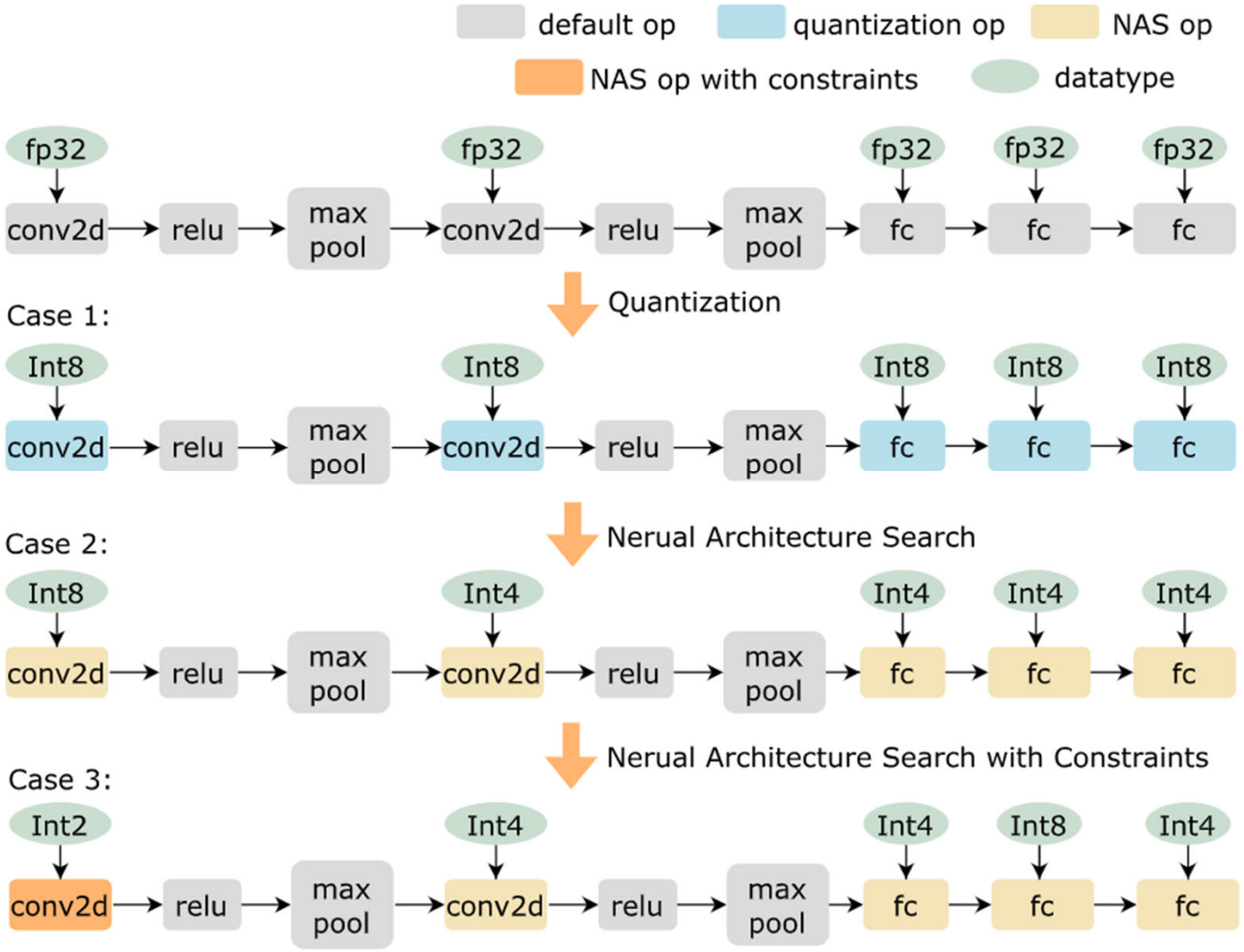
- **Robustness of NNs**
  - Case study (FP)
  - Identifying vulnerable weight parameters
  - **Quantization**
    - **Multi-bit-width neural networks**
  - Countermeasures in literature
- Robustness of hardware
  - Edge AI accelerator
  - GPU
  - Countermeasures in literature

# Neural architecture search (NAS) for multi-bit-width (MBW) NNs

Thanks to approximate and quantization-compatible features of CNNs, NAS is used for precision reduction with limited accuracy loss.



# Examples of MBW LeNet5 generation




Differential NAS approach in [4] is applied.

[4] M. Huang et al., "A high performance multi-bit-width booth vector systolic accelerator for NAS optimized deep learning neural networks," *IEEE Trans. CAS-I*, 2022.

# Reliability concern regarding MBW NNs

- In highly precision-reduced NNs, each bit needs to carry more information.

- Important to analyze the reliability of these multi-precision networks.

| Layer | Lenet5 Backbone Activation (NCHW) | Lenet5 Backbone Weight (OIHW) | INT8 model Precision (Bit) | INT8 model Weight Size (Byte) | INT4/8 model Precision (Bit) | INT4/8 model Weight size (Byte) | INT2/4/8 model Precision (Bit) | INT2/4/8 model Weight size (Byte) |
|---|---|---|---|---|---|---|---|---|
| Conv1 | 1×1×32×32 | 6×1×5×5 | 8 | 150 | 8 | 150 | 2 | 37.5 |
| Pooling1 | 1×6×28×28 | - | 8 | - | 8 | - | 2 | - |
| Conv2 | 1×6×14×14 | 16×6×5×5 | 8 | 2400 | 4 | 1200 | 4 | 1200 |
| Pooling2 | 1×16×10×10 | - | 8 | - | 4 | - | 4 | - |
| FC1 | 1×16×5×5 | 120×16×5×5 | 8 | 48000 | 4 | 24000 | 4 | 24000 |
| FC2 | 1×120×1×1 | 84×120×1×1 | 8 | 10080 | 4 | 5040 | 8 | 10080 |
| FC3 | 1×84×1×1 | 10×84×1×1 | 8 | 840 | 4 | 420 | 4 | 420 |
| Total | - | - | - | 61470 | - | 30810 | - | 35737.5 |
| Accuracy | - | - | 98.48% | | 95.42% | | 90.25% | |

Q. Cheng et al., "Reliability Exploration of System-on-Chip With Multi-Bit-Width Accelerator for Multi-Precision Deep Neural Networks," *IEEE Trans. CAS-I*, 2023,

# Increase in misclassification



| Layer | Bit Position | INT8 Model Samples | INT8 Model Errors | INT8 Model Error Ratio | INT4/8 model Samples | INT4/8 model Errors | INT4/8 model Error Ratio | INT2/4/8 model Samples | INT2/4/8 model Errors | INT2/4/8 model Error Ratio |
|---|---|---|---|---|---|---|---|---|---|---|
| Conv1 (1) | 0 | 15K | 0 | 0% | 15K | 0 | 0% | 15K | 120 | 0.8000% |
| | 1 | 15K | 0 | 0% | 15K | 7 | 0.0467% | 15K | 217 | 1.4467% |
| | 2 | 15K | 1 | 0.0067% | 15K | 13 | 0.0867% | - | - | - |
| | 3 | 15K | 0 | 0% | 15K | 57 | 0.3800% | - | - | - |
| | 4 | 15K | 1 | 0.0067% | 15K | 94 | 0.6267% | - | - | - |
| | 5 | 15K | 0 | 0% | 15K | 127 | 0.8467% | - | - | - |
| | 6 | 15K | 0 | 0% | 15K | 150 | 1.0000% | - | - | - |
| | 7 | 15K | 5 | 0.0333% | 15K | 143 | 0.9533% | - | - | - |
| **LAVS(esb) / Dur$^l$(ns) / RT(ns)** | | **8.2220e-5 / 1174570 / 299562.5** | | | **6.9417e-3 / 1174570 / 299562.5** | | | **3.9583e-3 / 1174570 / 301937.5** | | |
| Conv2 (2) | 0 | 240K | 0 | 0% | 240K | 408 | 0.1700% | 240K | 90 | 0.0375% |
| | 1 | 240K | 0 | 0% | 240K | 610 | 0.2542% | 240K | 206 | 0.0858% |
| | 2 | 240K | 1 | 0.0004% | 240K | 893 | 0.3721% | 240K | 302 | 0.1258% |
| | 3 | 240K | 11 | 0.0046% | 240K | 1415 | 0.5896% | 240K | 486 | 0.2025% |
| | 4 | 240K | 35 | 0.0146% | - | - | - | - | - | - |
| | 5 | 240K | 75 | 0.0313% | - | - | - | - | - | - |
| | 6 | 240K | 95 | 0.0396% | - | - | - | - | - | - |
| | 7 | 240K | 78 | 0.0325% | - | - | - | - | - | - |
| **LAVS(esb) / Dur$^l$(ns) / RT(ns)** | | **5.8437e-4 / 198090 / 155250.0** | | | **6.5885e-3 / 198090 / 155375.0** | | | **2.1473e-3 / 198090 / 156562.5** | | |
| FC1 (3) | 0 | 4800K | 0 | 0% | 4800K | 347 | 0.0072% | 4800K | 1394 | 0.0290% |
| | 1 | 4800K | 0 | 0% | 4800K | 766 | 0.0160% | 4800K | 2320 | 0.0483% |
| | 2 | 4800K | 0 | 0% | 4800K | 1778 | 0.0370% | 4800K | 4784 | 0.0997% |
| | 3 | 4800K | 0 | 0% | 4800K | 2688 | 0.0560% | 4800K | 9349 | 0.1948% |
| | 4 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 4800K | 1 | 0.0000% | - | - | - | - | - | - |
| | 7 | 4800K | 46 | 0.0010% | - | - | - | - | - | - |
| **LAVS(esb) / Dur$^l$(ns) / RT(ns)** | | **4.2300e-8 / 90 /163937.5** | | | **5.0211e-6 / 90 / 164562.5** | | | **1.6062e-5 / 90 /165625.0** | | |
| FC2 (4) | 0 | 1008K | 0 | 0% | 1008K | 64 | 0.0063% | 1008K | 7 | 0.0007% |
| | 1 | 1008K | 0 | 0% | 1008K | 200 | 0.0198% | 1008K | 38 | 0.0038% |
| | 2 | 1008K | 0 | 0% | 1008K | 436 | 0.0433% | 1008K | 67 | 0.0066% |
| | 3 | 1008K | 0 | 0% | 1008K | 699 | 0.0693% | 1008K | 81 | 0.0080% |
| | 4 | 1008K | 0 | 0% | - | - | - | 1008K | 93 | 0.0092% |
| | 5 | 1008K | 0 | 0% | - | - | - | 1008K | 102 | 0.0101% |
| | 6 | 1008K | 0 | 0% | - | - | - | 1008K | 178 | 0.0177% |
| | 7 | 1008K | 0 | 0% | - | - | - | 1008K | 378 | 0.0375% |
| **LAVS(esb) / Dur$^l$(ns) / RT(ns)** | | **0 / 90 / 114812.5** | | | **1.2082e-6 / 90 / 107062.5** | | | **8.1527e-7 / 90 /115437.5** | | |
| FC3 (5) | 0 | 84K | 0 | 0% | 84K | 22 | 0.0262% | 84K | 28 | 0.0333% |
| | 1 | 84K | 0 | 0% | 84K | 40 | 0.0476% | 84K | 42 | 0.0500% |
| | 2 | 84K | 0 | 0% | 84K | 74 | 0.0881% | 84K | 74 | 0.0881% |
| | 3 | 84K | 0 | 0% | 84K | 140 | 0.1667% | 84K | 161 | 0.1917% |
| | 4 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 84K | 2 | 0.0024% | - | - | - | - | - | - |
| | 7 | 84K | 56 | 0.0667% | - | - | - | - | - | - |
| **LAVS(esb) / Dur$^l$(ns) / RT(ns)** | | **3.4800e-8 / 90 / 101500.0** | | | **1.656e-7 / 90 / 100062.5** | | | **1.8300e-7 / 90 / 101437.5** | | |
| **ONAVS(esb) / Total RT(ns)** | | **6.6667e-4 / 835062.5** | | | **1.3537e-2 / 826625.0** | | | **6.1227e-3 / 841000.0** | | |

# Increase in misclassification

**Higher bits have larger impacts.**
**77.1% SDCs come from high two bits.**

| Layer | Bit Position | INT8 Model Samples | INT8 Model Errors | INT8 Model Error Ratio | INT4/8 model Samples | INT4/8 model Errors | INT4/8 model Error Ratio | INT2/4/8 model Samples | INT2/4/8 model Errors | INT2/4/8 model Error Ratio |
|---|---|---|---|---|---|---|---|---|---|---|
| Conv1 (1) | 0 | 15K | 0 | 0% | 15K | 0 | 0% | 15K | 120 | 0.8000% |
| | 1 | 15K | 0 | 0% | 15K | 7 | 0.0467% | 15K | 217 | 1.4467% |
| | 2 | 15K | 1 | 0.0067% | 15K | 13 | 0.0867% | - | - | - |
| | 3 | 15K | 0 | 0% | 15K | 57 | 0.3800% | - | - | - |
| | 4 | 15K | 1 | 0.0067% | 15K | 94 | 0.6267% | - | - | - |
| | 5 | 15K | 0 | 0% | 15K | 127 | 0.8467% | - | - | - |
| | 6 | 15K | 0 | 0% | 15K | 150 | 1.0000% | - | - | - |
| | 7 | 15K | 5 | 0.0333% | 15K | 143 | 0.9533% | - | - | - |
| LAVS(esb) / Dur'(ns) / RT(ns) | | 8.2220e-5 / 1174570 / 299562.5 | | | 6.9417e-3 / 1174570 / 299562.5 | | | 3.9583e-3 / 1174570 / 301937.5 | | |
| Conv2 (2) | 0 | 240K | 0 | 0% | 240K | 408 | 0.1700% | 240K | 90 | 0.0375% |
| | 1 | 240K | 0 | 0% | 240K | 610 | 0.2542% | 240K | 206 | 0.0858% |
| | 2 | 240K | 1 | 0.0004% | 240K | 893 | 0.3721% | 240K | 302 | 0.1258% |
| | 3 | 240K | 11 | 0.0046% | 240K | 1415 | 0.5896% | 240K | 486 | 0.2025% |
| | 4 | 240K | 35 | 0.0146% | - | - | - | - | - | - |
| | 5 | 240K | 75 | 0.0313% | - | - | - | - | - | - |
| | 6 | 240K | 95 | 0.0396% | - | - | - | - | - | - |
| | 7 | 240K | 78 | 0.0325% | - | - | - | - | - | - |
| LAVS(esb) / Dur'(ns) / RT(ns) | | 5.8437e-4 / 198090 / 155250.0 | | | 6.5885e-3 / 198090 / 155375.0 | | | 2.1473e-3 / 198090 / 156562.5 | | |
| FC1 (3) | 0 | 4800K | 0 | 0% | 4800K | 347 | 0.0072% | 4800K | 1394 | 0.0290% |
| | 1 | 4800K | 0 | 0% | 4800K | 766 | 0.0160% | 4800K | 2320 | 0.0483% |
| | 2 | 4800K | 0 | 0% | 4800K | 1778 | 0.0370% | 4800K | 4784 | 0.0997% |
| | 3 | 4800K | 0 | 0% | 4800K | 2688 | 0.0560% | 4800K | 9349 | 0.1948% |
| | 4 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 4800K | 1 | 0.0000% | - | - | - | - | - | - |
| | 7 | 4800K | 46 | 0.0010% | - | - | - | - | - | - |
| LAVS(esb) / Dur'(ns) / RT(ns) | | 4.2300e-8 / 90 /163937.5 | | | 5.0211e-6 / 90 / 164562.5 | | | 1.6062e-5 / 90 /165625.0 | | |
| FC2 (4) | 0 | 1008K | 0 | 0% | 1008K | 64 | 0.0063% | 1008K | 7 | 0.0007% |
| | 1 | 1008K | 0 | 0% | 1008K | 200 | 0.0198% | 1008K | 38 | 0.0038% |
| | 2 | 1008K | 0 | 0% | 1008K | 436 | 0.0433% | 1008K | 67 | 0.0066% |
| | 3 | 1008K | 0 | 0% | 1008K | 699 | 0.0693% | 1008K | 81 | 0.0080% |
| | 4 | 1008K | 0 | 0% | - | - | - | 1008K | 93 | 0.0092% |
| | 5 | 1008K | 0 | 0% | - | - | - | 1008K | 102 | 0.0101% |
| | 6 | 1008K | 0 | 0% | - | - | - | 1008K | 178 | 0.0177% |
| | 7 | 1008K | 0 | 0% | - | - | - | 1008K | 378 | 0.0375% |
| LAVS(esb) / Dur'(ns) / RT(ns) | | 0 / 90 / 114812.5 | | | 1.2082e-6 / 90 / 107062.5 | | | 8.1527e-7 / 90 /115437.5 | | |
| FC3 (5) | 0 | 84K | 0 | 0% | 84K | 22 | 0.0262% | 84K | 28 | 0.0333% |
| | 1 | 84K | 0 | 0% | 84K | 40 | 0.0476% | 84K | 42 | 0.0500% |
| | 2 | 84K | 0 | 0% | 84K | 74 | 0.0881% | 84K | 74 | 0.0881% |
| | 3 | 84K | 0 | 0% | 84K | 140 | 0.1667% | 84K | 161 | 0.1917% |
| | 4 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 84K | 2 | 0.0024% | - | - | - | - | - | - |
| | 7 | 84K | 56 | 0.0667% | - | - | - | - | - | - |
| LAVS(esb) / Dur'(ns) / RT(ns) | | 3.4800e-8 / 90 / 101500.0 | | | 1.656e-7 / 90 / 100062.5 | | | 1.8300e-7 / 90 / 101437.5 | | |
| ONAVS(esb) / Total RT(ns) | | 6.6667e-4 / 835062.5 | | | 1.3537e-2 / 826625.0 | | | 6.1227e-3 / 841000.0 | | |

# Increase in misclassification

| Layer | Bit Position | INT8 Model Samples | INT8 Model Errors | INT8 Model Error Ratio | INT4/8 model Samples | INT4/8 model Errors | INT4/8 model Error Ratio | INT2/4/8 model Samples | INT2/4/8 model Errors | INT2/4/8 model Error Ratio |
|---|---|---|---|---|---|---|---|---|---|---|
| Conv1 (1) | 0 | 15K | 0 | 0% | 15K | 0 | 0% | 15K | 120 | 0.8000% |
| | 1 | 15K | 0 | 0% | 15K | 7 | 0.0467% | 15K | 217 | 1.4467% |
| | 2 | 15K | 1 | 0.0067% | 15K | 13 | 0.0867% | - | - | - |
| | 3 | 15K | 0 | 0% | 15K | 57 | 0.3800% | - | - | - |
| | 4 | 15K | 1 | 0.0067% | 15K | 94 | 0.6267% | - | - | - |
| | 5 | 15K | 0 | 0% | 15K | 127 | 0.8467% | - | - | - |
| | 6 | 15K | 0 | 0% | 15K | 150 | 1.0000% | - | - | - |
| | 7 | 15K | 5 | 0.0333% | 15K | 143 | 0.9533% | - | - | - |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 8.2220e-5 / 1174570 / 299562.5 | | | 6.9417e-3 / 1174570 / 299562.5 | | | 3.9583e-3 / 1174570 / 301937.5 | | |
| Conv2 (2) | 0 | 240K | 0 | 0% | 240K | 408 | 0.1700% | 240K | 90 | 0.0375% |
| | 1 | 240K | 0 | 0% | 240K | 610 | 0.2542% | 240K | 206 | 0.0858% |
| | 2 | 240K | 1 | 0.0004% | 240K | 893 | 0.3721% | 240K | 302 | 0.1258% |
| | 3 | 240K | 11 | 0.0046% | 240K | 1415 | 0.5896% | 240K | 486 | 0.2025% |
| | 4 | 240K | 35 | 0.0146% | - | - | - | - | - | - |
| | 5 | 240K | 75 | 0.0313% | - | - | - | - | - | - |
| | 6 | 240K | 95 | 0.0396% | - | - | - | - | - | - |
| | 7 | 240K | 78 | 0.0325% | - | - | - | - | - | - |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 5.8437e-4 / 198090 / 155250.0 | | | 6.5885e-3 / 198090 / 155375.0 | | | 2.1473e-3 / 198090 / 156562.5 | | |
| FC1 (3) | 0 | 4800K | 0 | 0% | 4800K | 347 | 0.0072% | 4800K | 1394 | 0.0290% |
| | 1 | 4800K | 0 | 0% | 4800K | 766 | 0.0160% | 4800K | 2320 | 0.0483% |
| | 2 | 4800K | 0 | 0% | 4800K | 1778 | 0.0370% | 4800K | 4784 | 0.0997% |
| | 3 | 4800K | 0 | 0% | 4800K | 2688 | 0.0560% | 4800K | 9349 | 0.1948% |
| | 4 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 4800K | 1 | 0.0000% | - | - | - | - | - | - |
| | 7 | 4800K | 46 | 0.0010% | - | - | - | - | - | - |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 4.2300e-8 / 90 /163937.5 | | | 1.6062e-5 / 90 /165625.0 | | | | | |
| FC2 (4) | 0 | 1008K | 0 | 0% | 1008K | 64 | 0.0063% | 1008K | 7 | 0.0007% |
| | 1 | 1008K | 0 | 0% | 1008K | 200 | 0.0198% | 1008K | 38 | 0.0038% |
| | 2 | 1008K | 0 | 0% | 1008K | 436 | 0.0433% | 1008K | 67 | 0.0066% |
| | 3 | 1008K | 0 | 0% | 1008K | 699 | 0.0693% | 1008K | 81 | 0.0080% |
| | 4 | 1008K | 0 | 0% | - | - | - | 1008K | 93 | 0.0092% |
| | 5 | 1008K | 0 | 0% | - | - | - | 1008K | 102 | 0.0101% |
| | 6 | 1008K | 0 | 0% | - | - | - | 1008K | 178 | 0.0177% |
| | 7 | 1008K | 0 | 0% | - | - | - | 1008K | 378 | 0.0375% |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 0 / 90 / 114812.5 | | | 1.2082e-6 / 90 / 107062.5 | | | 8.1527e-7 / 90 /115437.5 | | |
| FC3 (5) | 0 | 84K | 0 | 0% | 84K | 22 | 0.0262% | 84K | 28 | 0.0333% |
| | 1 | 84K | 0 | 0% | 84K | 40 | 0.0476% | 84K | 42 | 0.0500% |
| | 2 | 84K | 0 | 0% | 84K | 74 | 0.0881% | 84K | 74 | 0.0881% |
| | 3 | 84K | 0 | 0% | 84K | 140 | 0.1667% | 84K | 161 | 0.1917% |
| | 4 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 84K | 2 | 0.0024% | - | - | - | - | - | - |
| | 7 | 84K | 56 | 0.0667% | - | - | - | - | - | - |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 3.4800e-8 / 90 / 101500.0 | | | 1.656e-7 / 90 / 100062.5 | | | 1.8300e-7 / 90 / 101437.5 | | |
| ONAVS(esb) / Total RT(ns) | | 6.6667e-4 / 835062.5 | | | 1.3537e-2 / 826625.0 | | | 6.1227e-3 / 841000.0 | | |

**Quantization induces larger impacts.**

# Increase in misclassification

| Layer | Bit Position | INT8 Model Samples | INT8 Model Errors | INT8 Model Error Ratio | INT4/8 model Samples | INT4/8 model Errors | INT4/8 model Error Ratio | INT2/4/8 model Samples | INT2/4/8 model Errors | INT2/4/8 model Error Ratio |
|---|---|---|---|---|---|---|---|---|---|---|
| Conv1 (1) | 0 | 15K | 0 | 0% | 15K | 0 | 0% | 15K | 120 | 0.8000% |
| | 1 | 15K | 0 | 0% | 15K | 7 | 0.0467% | 15K | 217 | 1.4467% |
| | 2 | 15K | 1 | 0.0067% | 15K | 13 | 0.0867% | - | - | - |
| | 3 | 15K | 0 | 0% | 15K | 57 | 0.3800% | - | - | - |
| | 4 | 15K | 1 | 0.0067% | 15K | 94 | 0.6267% | - | - | - |
| | 5 | 15K | 0 | 0% | 15K | 127 | 0.8467% | - | - | - |
| | 6 | 15K | 0 | 0% | 15K | 150 | 1.0000% | - | - | - |
| | 7 | 15K | 5 | 0.0333% | 15K | 143 | 0.9533% | - | - | - |
| LAVS(esb) / Dur[l](ns) / RT(ns) | | 8.2220e-5 / 1174570 / 299562.5 | | | 6.9417e-3 / 1174570 / 299562.5 | | | 3.9583e-3 / 1174570 / 301937.5 | | |
| Conv2 (2) | 0 | 240K | 0 | 0% | 240K | 408 | 0.1700% | 240K | 90 | 0.0375% |
| | 1 | 240K | 0 | 0% | 240K | 610 | 0.2542% | 240K | 206 | 0.0858% |
| | 2 | 240K | 1 | 0.0004% | 240K | 893 | 0.3721% | 240K | 302 | 0.1258% |
| | 3 | 240K | 11 | 0.0046% | 240K | 1415 | 0.5896% | 240K | 486 | 0.2025% |
| | 4 | 240K | 35 | 0.0146% | - | - | - | - | - | - |
| | 5 | 240K | 75 | 0.0313% | - | - | - | - | - | - |
| | 6 | 240K | 95 | 0.0396% | - | - | - | - | - | - |
| | 7 | 240K | 78 | 0.0325% | - | - | - | - | - | - |
| LAVS(esb) / Dur[l](ns) / RT(ns) | | 5.8437e-4 / 198090 / 155250.0 | | | 6.5885e-3 / 198090 / 155375.0 | | | 2.1473e-3 / 198090 / 156562.5 | | |
| FC1 (3) | 0 | 4800K | 0 | 0% | 4800K | 347 | 0.0072% | 4800K | 1394 | 0.0290% |
| | 1 | 4800K | 0 | 0% | 4800K | 766 | 0.0160% | 4800K | 2320 | 0.0483% |
| | 2 | 4800K | 0 | 0% | 4800K | 1778 | 0.0370% | 4800K | 4784 | 0.0997% |
| | 3 | 4800K | 0 | 0% | 4800K | 2688 | 0.0560% | 4800K | 9349 | 0.1948% |
| | 4 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 4800K | 1 | 0.0000% | - | - | - | - | - | - |
| | 7 | 4800K | 6 | 0.0010% | - | - | - | - | - | - |
| LAVS(esb) / Dur[l](ns) / RT(ns) | | 0 / 90 / 164062.5 | | | 1.0211e-6 / 90 / 164562.5 | | | 1.6062e-5 / 90 /165625.0 | | |
| FC2 (4) | 0 | 1008K | 0 | 0% | 1008K | 64 | 0.0063% | 1008K | 7 | 0.0007% |
| | 1 | 1008K | 0 | 0% | 1008K | 390 | 0.0387% | 1008K | 38 | 0.0038% |
| | 2 | 1008K | 0 | 0% | 1008K | 666 | 0.0661% | 1008K | 67 | 0.0066% |
| | 3 | 1008K | 0 | 0% | 1008K | 699 | 0.0693% | 1008K | 81 | 0.0080% |
| | 4 | 1008K | 0 | 0% | - | - | - | 1008K | 93 | 0.0092% |
| | 5 | 1008K | 0 | 0% | - | - | - | 1008K | 102 | 0.0101% |
| | 6 | 1008K | 0 | 0% | - | - | - | 1008K | 178 | 0.0177% |
| | 7 | 1008K | 0 | 0% | - | - | - | 1008K | 378 | 0.0375% |
| LAVS(esb) / Dur[l](ns) / RT(ns) | | 0 / 90 / 114812.5 | | | 1.2082e-6 / 90 / 107062.5 | | | 8.1527e-7 / 90 /115437.5 | | |
| FC3 (5) | 0 | 84K | 0 | 0% | 84K | 22 | 0.0262% | 84K | 28 | 0.0333% |
| | 1 | 84K | 0 | 0% | 84K | 40 | 0.0476% | 84K | 42 | 0.0500% |
| | 2 | 84K | 0 | 0% | 84K | 74 | 0.0881% | 84K | 74 | 0.0881% |
| | 3 | 84K | 0 | 0% | 84K | 140 | 0.1667% | 84K | 161 | 0.1917% |
| | 4 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 84K | 2 | 0.0024% | - | - | - | - | - | - |
| | 7 | 84K | 56 | 0.0667% | - | - | - | - | - | - |
| LAVS(esb) / Dur[l](ns) / RT(ns) | | 3.4800e-8 / 90 / 101500.0 | | | 1.656e-7 / 90 / 100062.5 | | | 1.8300e-7 / 90 / 101437.5 | | |
| ONAVS(esb) / Total RT(ns) | | 6.6667e-4 / 835062.5 | | | 1.3537e-2 / 826625.0 | | | 6.1227e-3 / 841000.0 | | |

- Conv. layers induce larger impacts due to multiple usage.
- Conv. Weight size is small and selective protection is meaningful.

# Increase in misclassification

| Layer | Bit Position | INT8 Model Samples | INT8 Model Errors | INT8 Model Error Ratio | INT4/8 model Samples | INT4/8 model Errors | INT4/8 model Error Ratio | INT2/4/8 model Samples | INT2/4/8 model Errors | INT2/4/8 model Error Ratio |
|---|---|---|---|---|---|---|---|---|---|---|
| Conv1 (1) | 0 | 15K | 0 | 0% | 15K | 0 | 0% | 15K | 120 | 0.8000% |
| | 1 | 15K | 0 | 0% | 15K | 7 | 0.0467% | 15K | 217 | 1.4467% |
| | 2 | 15K | 1 | 0.0067% | 15K | 13 | 0.0867% | - | - | - |
| | 3 | 15K | 0 | 0% | 15K | 57 | 0.3800% | - | - | - |
| | 4 | 15K | 1 | 0.0067% | 15K | 94 | 0.6267% | - | - | - |
| | 5 | 15K | 0 | 0% | 15K | 127 | 0.8467% | - | - | - |
| | 6 | 15K | 0 | 0% | 15K | 150 | 1.0000% | - | - | - |
| | 7 | 15K | 5 | 0.0333% | 15K | 143 | 0.9533% | - | - | - |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 8.2220e-5 / 1174570 / 299562.5 | | | 6.9417e-3 / 1174570 / 299562.5 | | | 3.9583e-3 / 1174570 / 301937.5 | | |
| Conv2 (2) | 0 | 240K | 0 | 0% | 240K | 408 | 0.1700% | 240K | 90 | 0.0375% |
| | 1 | 240K | 0 | 0% | 240K | 610 | 0.2542% | 240K | 206 | 0.0858% |
| | 2 | 240K | 1 | 0.0004% | 240K | 893 | 0.3721% | 240K | 302 | 0.1258% |
| | 3 | 240K | 11 | 0.0046% | 240K | 1415 | 0.5896% | 240K | 486 | 0.2025% |
| | 4 | 240K | 35 | 0.0146% | - | - | - | - | - | - |
| | 5 | 240K | 71 | 0.0296% | - | - | - | - | - | - |
| | 6 | 240K | 95 | 0.0396% | - | - | - | - | - | - |
| | 7 | 240K | 78 | 0.0325% | - | - | - | - | - | - |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 2.1473e-3 / 198090 / 155375.0 | | | | | | 2.1473e-3 / 198090 / 156562.5 | | |
| FC1 (3) | 0 | 4800K | 0 | 0% | 4800K | 347 | 0.0072% | 4800K | 1394 | 0.0290% |
| | 1 | 4800K | 0 | 0% | 4800K | 768 | 0.0160% | 4800K | 2320 | 0.0483% |
| | 2 | 4800K | 0 | 0% | 4800K | 1778 | 0.0370% | 4800K | 4784 | 0.0997% |
| | 3 | 4800K | 0 | 0% | 4800K | 2688 | 0.0560% | 4800K | 9349 | 0.1948% |
| | 4 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 4800K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 4800K | 1 | 0.0000% | - | - | - | - | - | - |
| | 7 | 4800K | 46 | 0.0010% | - | - | - | - | - | - |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 5.2083e-7 / 90 / 164562.5 | | | | | | 1.6062e-5 / 90 /165625.0 | | |
| FC2 (4) | 0 | 1008K | 0 | 0% | 1008K | 64 | 0.0063% | 1008K | 7 | 0.0007% |
| | 1 | 1008K | 0 | 0% | 1008K | 128 | 0.0127% | 1008K | 38 | 0.0038% |
| | 2 | 1008K | 0 | 0% | 1008K | 256 | 0.0254% | 1008K | 67 | 0.0066% |
| | 3 | 1008K | 0 | 0% | 1008K | 699 | 0.0693% | 1008K | 81 | 0.0080% |
| | 4 | 1008K | 0 | 0% | - | - | - | 1008K | 93 | 0.0092% |
| | 5 | 1008K | 0 | 0% | - | - | - | 1008K | 102 | 0.0101% |
| | 6 | 1008K | 0 | 0% | - | - | - | 1008K | 178 | 0.0177% |
| | 7 | 1008K | 0 | 0% | - | - | - | 1008K | 378 | 0.0375% |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 1.2082e-6 / 90 / 107062.5 | | | | | | 8.1527e-7 / 90 /115437.5 | | |
| FC3 (5) | 0 | 84K | 0 | 0% | 84K | 22 | 0.0262% | 84K | 28 | 0.0333% |
| | 1 | 84K | 0 | 0% | 84K | 40 | 0.0476% | 84K | 42 | 0.0500% |
| | 2 | 84K | 0 | 0% | 84K | 74 | 0.0881% | 84K | 74 | 0.0881% |
| | 3 | 84K | 0 | 0% | 84K | 140 | 0.1667% | 84K | 161 | 0.1917% |
| | 4 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 5 | 84K | 0 | 0% | - | - | - | - | - | - |
| | 6 | 84K | 2 | 0.0024% | - | - | - | - | - | - |
| | 7 | 84K | 56 | 0.0667% | - | - | - | - | - | - |
| LAVS(esb) / Dur[1](ns) / RT(ns) | | 3.4800e-8 / 90 / 101500.0 | | | 1.656e-7 / 90 / 100062.5 | | | 1.8300e-7 / 90 / 101437.5 | | |
| ONAVS(esb) / Total RT(ns) | | 6.6667e-4 / 835062.5 | | | 1.3537e-2 / 826625.0 | | | 6.1227e-3 / 841000.0 | | |

- Compared w/ FP case, the impact is limited.
- INT8 model is robust.
- When preventing error accumulation, the accuracy degradation is not significant even in MBW NNs.

# Agenda

- Robustness of NNs
  - Case study (FP)
  - Identifying vulnerable weight parameters
  - Quantization
    - Multi-bit-width neural networks
- **Robustness of hardware**
  - **Edge AI accelerator**
  - GPU
- Countermeasures in literature

# Demands of Edge AI chips

- Reduced Latency: Real-time data processing locally is crucial for applications like autonomous vehicles and robotics.
- Improved Privacy and Security: Local data processing on the device enhances data privacy and security, reducing the risk of data interception.
- Lower Bandwidth Requirements: The decrease in the need for data transmission to the cloud benefits areas with limited internet access and reducing connectivity dependence.

# Edge AI SoCs

- For AI applications, the hardware AI accelerator can be integrated into the SoC as a peripheral.
- Edge AI SoCs can be used for mission-critical and reliability-demanding applications.

- Essential to analyze weak points of the entire SoC with AI accelerator.

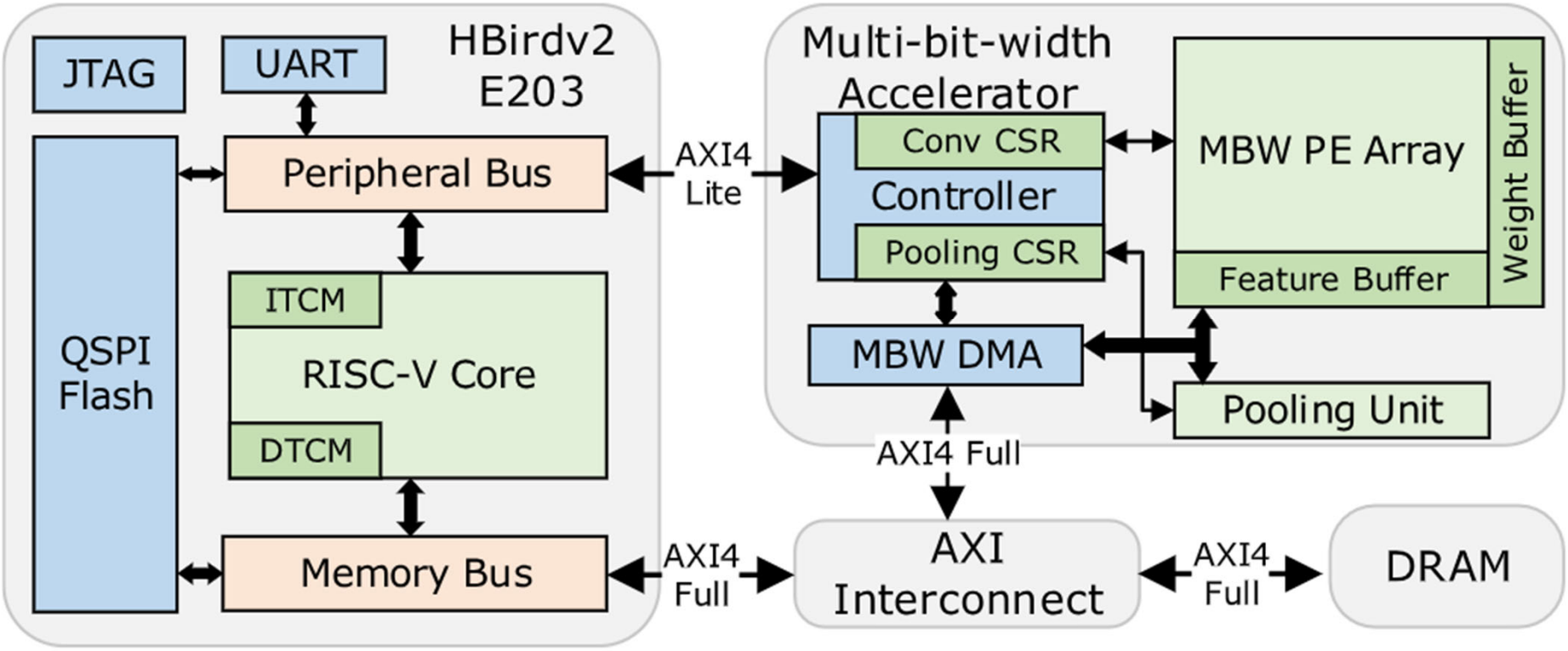
# Case study: Reliability assessment of an edge AI SoC

- We perform a case study using a SOTA SoC design that accepts NAS optimized LeNet5 with MNIST data set and implemented into a flash-based FPGA.
- We analyze the reliability of our SoC by fault injection (FI) and neutron irradiation experiments, aiming to provide valuable insights and serve as crucial references for future reliability-aware designs.
  - CRAM in the flash-based FPGA is robust to neutron irradiation compared with SRAM-based FPGA.
  - This FPGA-based SoC implementation reproduces the susceptibility of any dedicated SoC chips.

Q. Cheng et al., "Reliability Exploration of System-on-Chip With Multi-Bit-Width Accelerator for Multi-Precision Deep Neural Networks," *IEEE Trans. CAS-I*, 2023,

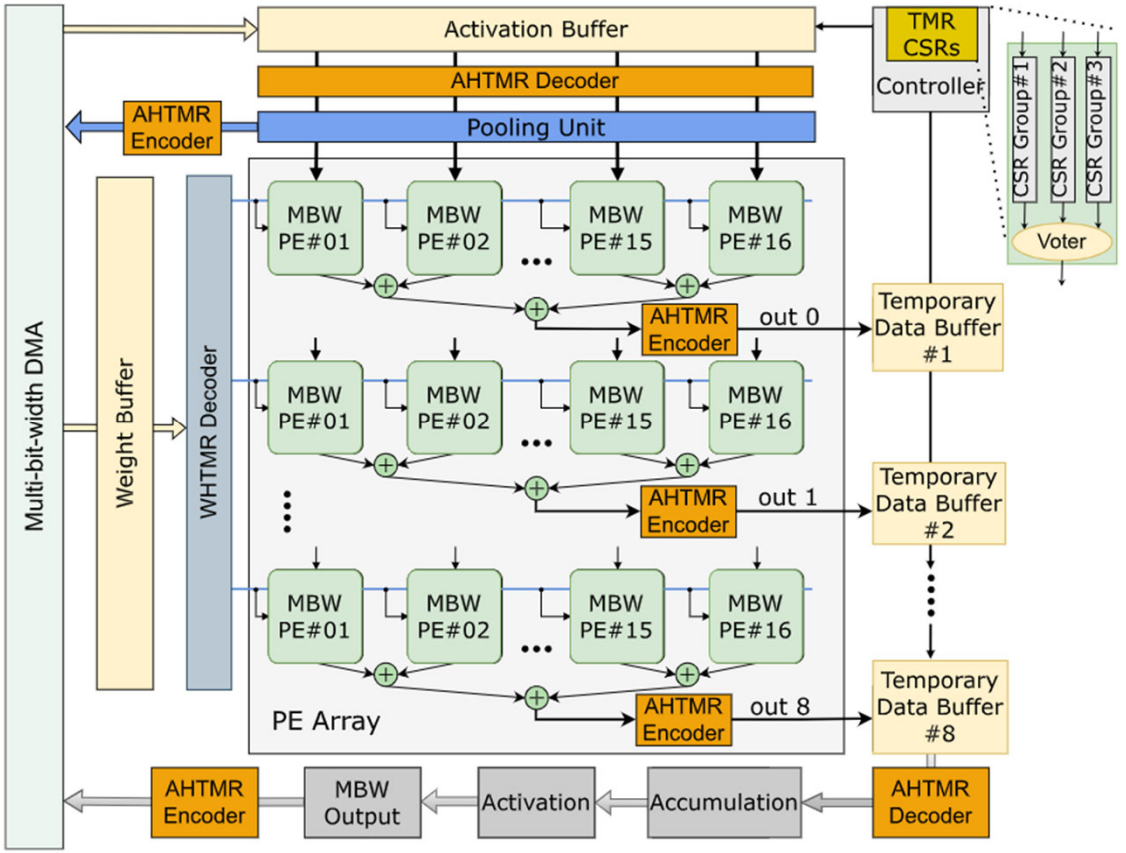
# Chip architecture

- SoC consists of 1) MBW accelerator, 2) lightweight 32-bit RISC-V processor, and 3) DDR4 DRAM.
- RISC-V core has 2-stage pipeline, instruction tightly coupled memory (ITCM) and data tightly coupled memory (DTCM).



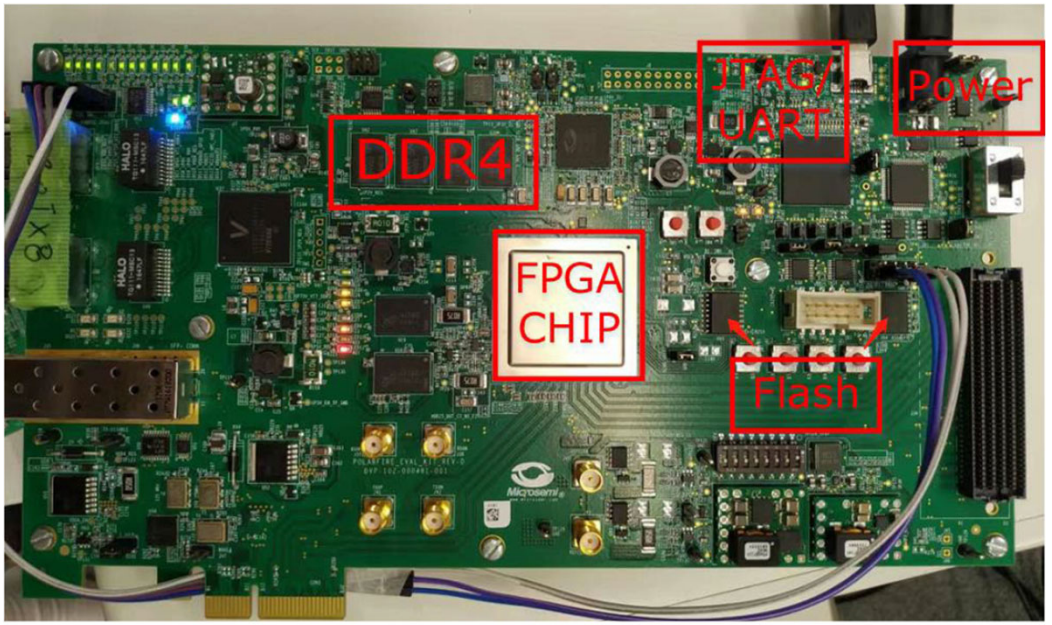
# MBW accelerator

- MBW vector systolic accelerator [4] w/ a 16x8 array
  - #inputs is 16, 32, and 64 for INT8, INT4, and INT2 respectively
  - # of output channel is 8.
- MAC is based on a multi-precision Booth multiplier.



# SoC implementation

- SoC is implemented for MPF300T Eval Kit.
- CRAM in this flash-based FPGA is robust to radiation, reproducing the susceptibility of SoC chips.

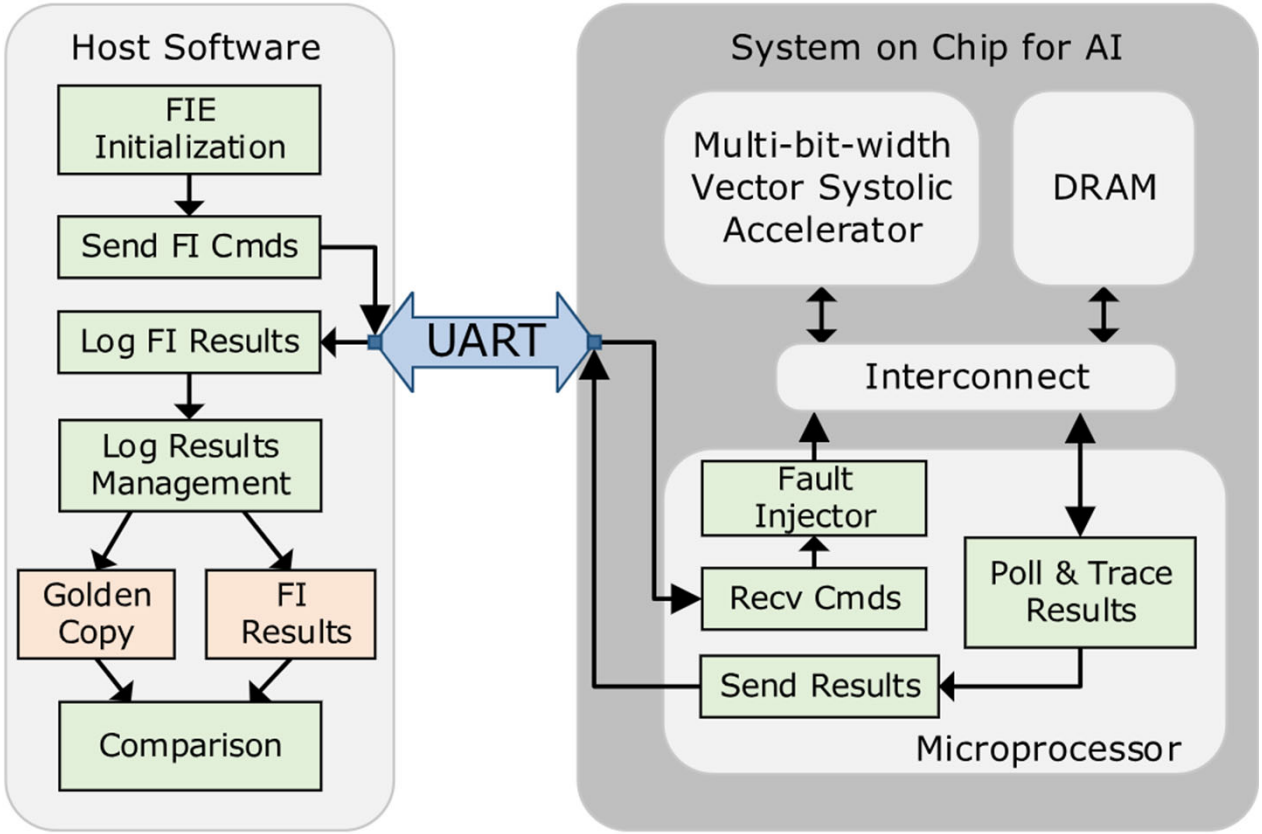


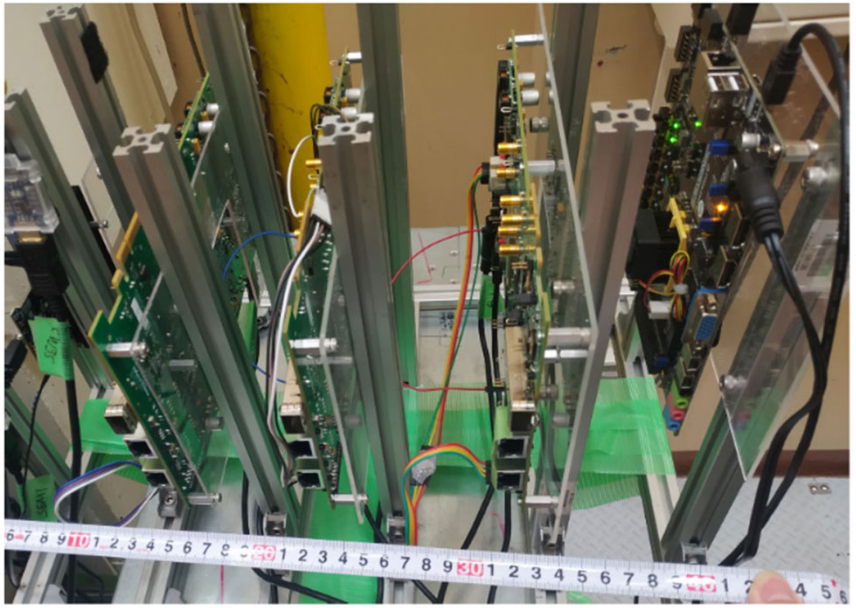

| Component | | Fabric 4LUT | Fabric DFF | Interface 4LUT | Interface DFF | Math (18×18) | uSRAM | LSRAM |
|---|---|---|---|---|---|---|---|---|
| Accelerator (100 MHz) | Conv Unit | 31339 | 21839 | 13032 | 13032 | 136 | 54 | 208 |
| | Pool Unit | 33250 | 26064 | 408 | 408 | 4 | 22 | 0 |
| | DMA | 375 | 75 | 336 | 336 | 0 | 28 | 0 |
| | Interface | 71 | 55 | 0 | 0 | 0 | 0 | 0 |
| | Total | 65035 | 48033 | 13776 | 13776 | 140 | 104 | 208 |
| RISCV E203 (16 MHz) | | 18591 | 9414 | 2448 | 2448 | 0 | 12 | 64 |
| PF_DDR4 (400 MHz) | | 18404 | 15219 | 1272 | 1272 | 0 | 43 | 21 |
| AXI4 Interconnect | | 3458 | 3138 | 636 | 636 | 0 | 53 | 0 |
| Others | | 61 | 36 | 0 | 0 | 0 | 0 | 0 |
| Total | | 105549 | 75840 | 18132 | 18132 | 140 | 212 | 293 |

* ITCM Size: 64KB LSRAM; DTCM Size: 64KB LSRAM; Accelerator Buffer Size: 512KB LSRAM; CSR Size: 79*32Bits uSRAM

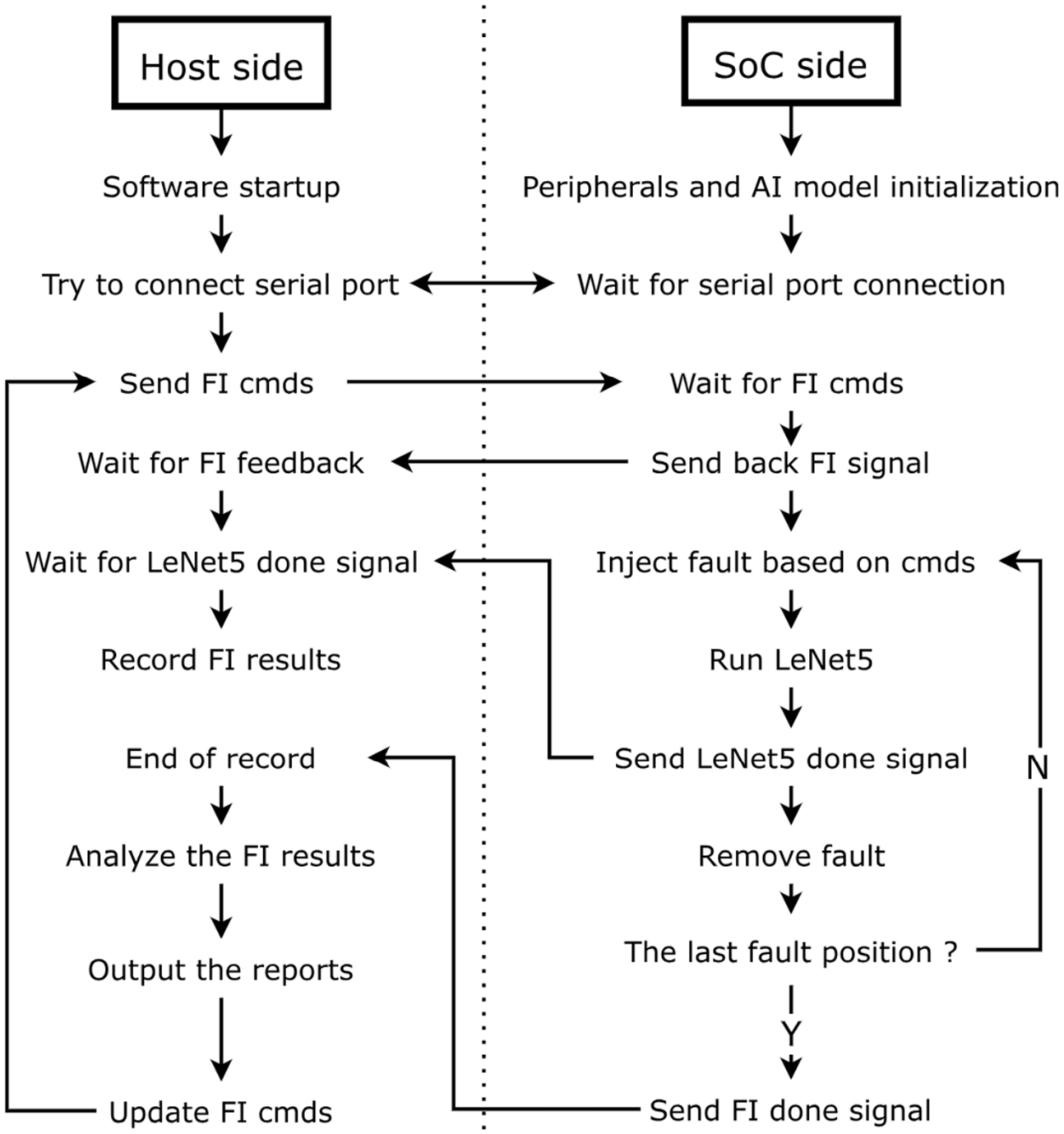
# Experiments

- Fault injection
  - Reproduces single bit upset in weights, activations, state registers of the controller, and CNN config. params.
  - Logs results and saves them via host software for analysis
- Neutron irradiation
  - Neutron beam is given to 3 FPGA boards at CYRIC, Tohoku Univ.



# Details in FI process



# FI results (control state registers (CSR))

- "Error Ratio": misclassification ratio in overall errors.
- "Acceptable": SoC can output the results
- "Unacceptable": SoC fails to complete CNN calculation, i.e. DUE

| CSR | Description | Error Ratio / Acceptable (%) | Unacceptable (%) |
|---|---|---|---|
| **STCONV** | Start conv operation | 0 / 0 | 100.00 |
| **CBWIN** | Bit-width of input data | 91.7557 / 100.000 | 0 |
| **CFADDR** | Activation base address | 75.5439 / 100.000 | 0 |
| **CFSHP** | Shape of activation | 76.8271 / 83.0952 | 16.9048 |
| **CCONF** | Padding, Stride and Kernel | 28.6260 / 35.0000 | 65.0000 |
| **CWADDR** | Weight base address | 69.8139 / 100.000 | 0 |
| **COADDR** | Output data base address | 77.5191 / 100.000 | 0 |
| **CDONE** | If conv operation done | 0 / 0 | 100.00 |
| **STPOOL** | Start pooling operation | 0 / 0 | 100.00 |
| **PCONF** | Pad, Stride and Kernel | 0 / 0 | 100.00 |
| **PIADDR** | Input data address | 66.0663 / 100.000 | 0 |
| **POADDR** | Output data address | 79.0672 / 100.000 | 0 |
| **PIOSHP** | Shape of activation | 59.3511 / 100.000 | 0 |
| **PBWIN** | Bit-width of input data | 88.5496 / 100.000 | 0 |
| **PDONE** | If pool operation done | 0 / 0 | 100.00 |

# Observations in FI to CSR

- The bit-flip of CSRs is far more sensitive than that of weights in NNs.
- Data errors in the acceptable range can lead to a high probability of misclassification.
- Error values in the unacceptable range cause the accelerator to enter into a deadlock or hang the AXI bus.

- Fortunately, the size of state registers is limited.
- Protecting state registers requires small overhead, but significantly contributes to reliability improvement.

# Reliability configurations in irradiation experiments

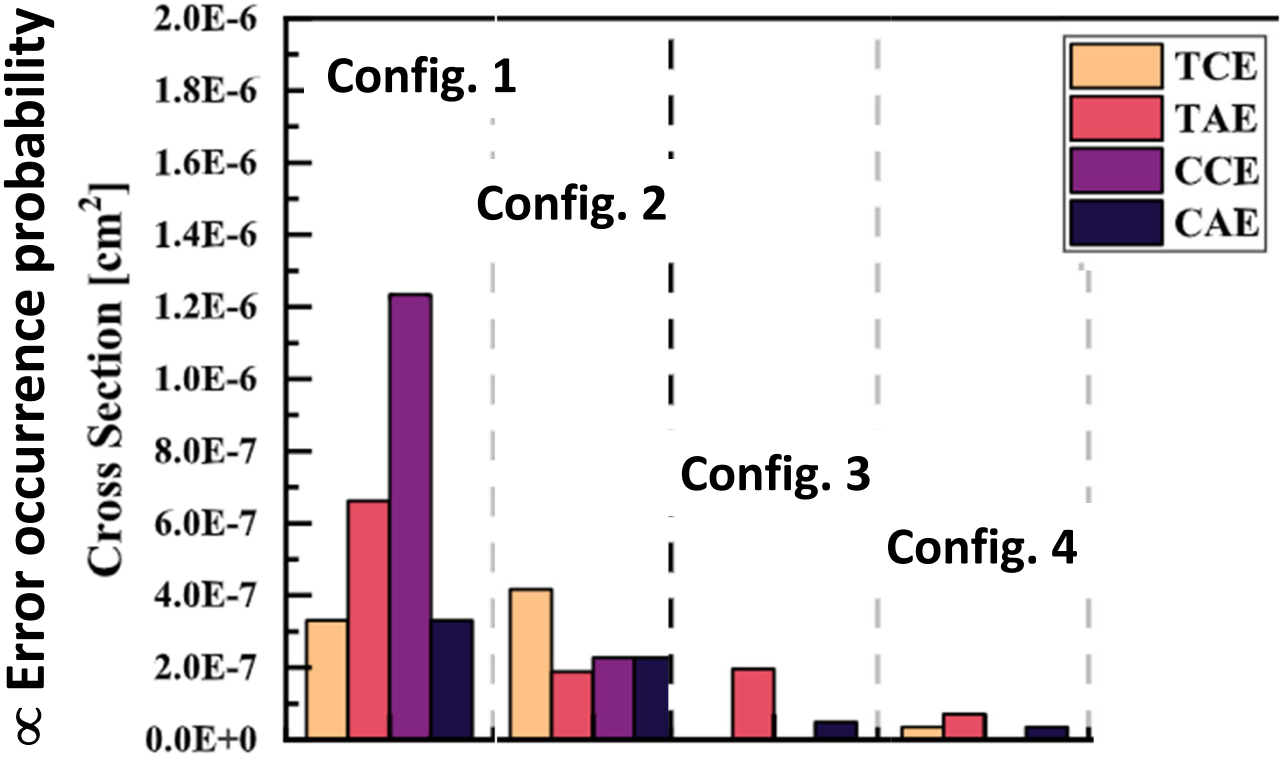
- Config. 1: Not refresh the contents of ITCM and DTCM frequently, resulting in error accumulation
- Config. 2: reset the after each round, preventing error accumulation in ITCM and DTCM
- Config. 3: replace normal SRAMs in RISC-V with TMRed SRAM
- Config. 4: replace CSRs with TMRed ones

| | Reset | TMR_I/DTCM | TMR_CSR |
|---|---|---|---|
| Config. 1 | No | No | No |
| Config. 2 | Yes | No | No |
| Config. 3 | Yes | Yes | No |
| Config. 4 | Yes | Yes | Yes |

# Irradiation results

| | Reset | TMR_I/DTCM | TMR_CSR |
|---|---|---|---|
| Config. 1 | No | No | No |
| Config. 2 | Yes | No | No |
| Config. 3 | Yes | Yes | No |
| Config. 4 | Yes | Yes | Yes |

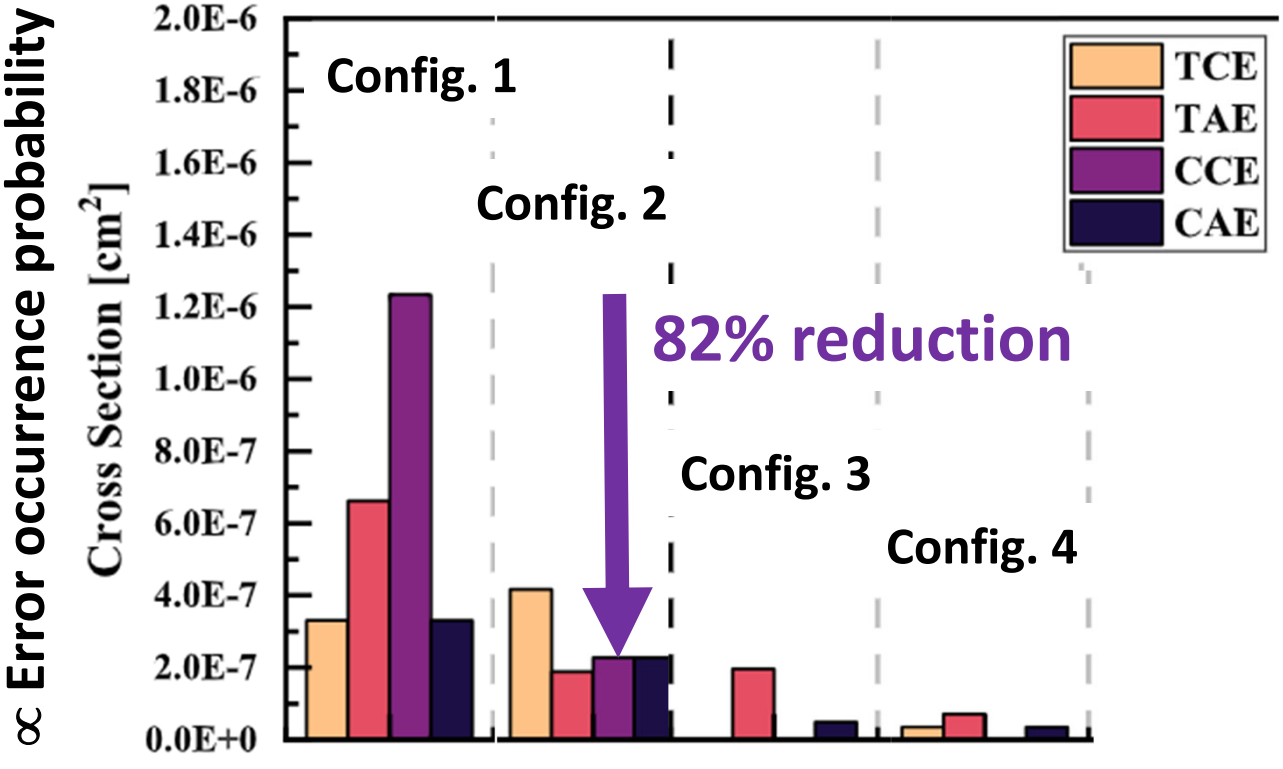
- **Tolerable Core Event (TCE)**: RISC-V core has some misbehavior, but not affect NN application
- **Tolerable Accelerator Event (TAE)**: classification is correct, but the middle outputs are not as expected
- **Critical Core Event (CCE)**: RISC-V core runs away or crashes
- **Critical Accelerator Event (CAE)**: accelerator has no correct response or correct classification result



# Irradiation results

| | Reset | TMR_I/DTCM | TMR_CSR |
|---|---|---|---|
| Config. 1 | No | No | No |
| Config. 2 | Yes | No | No |
| Config. 3 | Yes | Yes | No |
| Config. 4 | Yes | Yes | Yes |

- **Tolerable Core Event (TCE)**: RISC-V core has some misbehavior, but not affect NN application
- **Tolerable Accelerator Event (TAE)**: classification is correct, but the middle outputs are not as expected
- **Critical Core Event (CCE)**: RISC-V core runs away or crashes
- **Critical Accelerator Event (CAE)**: accelerator has no correct response or correct classification result

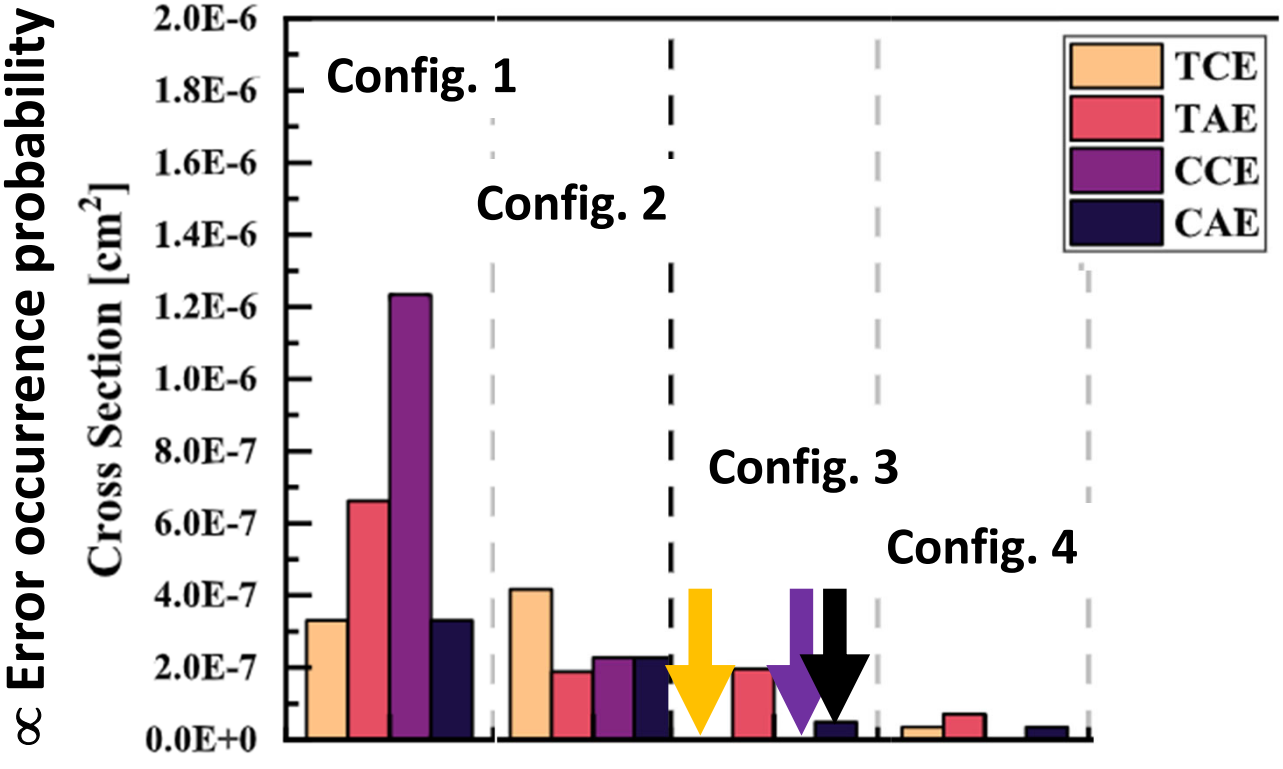



**RISC-V core is more sensitive to accumulated errors than accelerator.**
**When the data in I/DTCM is flipped and errors are accumulated, SoC could crash easily.**

# Irradiation results

| | Reset | TMR_I/DTCM | TMR_CSR |
|---|---|---|---|
| Config. 1 | No | No | No |
| Config. 2 | Yes | No | No |
| Config. 3 | Yes | Yes | No |
| Config. 4 | Yes | Yes | Yes |

- **Tolerable Core Event (TCE)**: RISC-V core has some misbehavior, but not affect NN application
- **Tolerable Accelerator Event (TAE)**: classification is correct, but the middle outputs are not as expected
- **Critical Core Event (CCE)**: RISC-V core runs away or crashes
- **Critical Accelerator Event (CAE)**: accelerator has no correct response or correct classification result

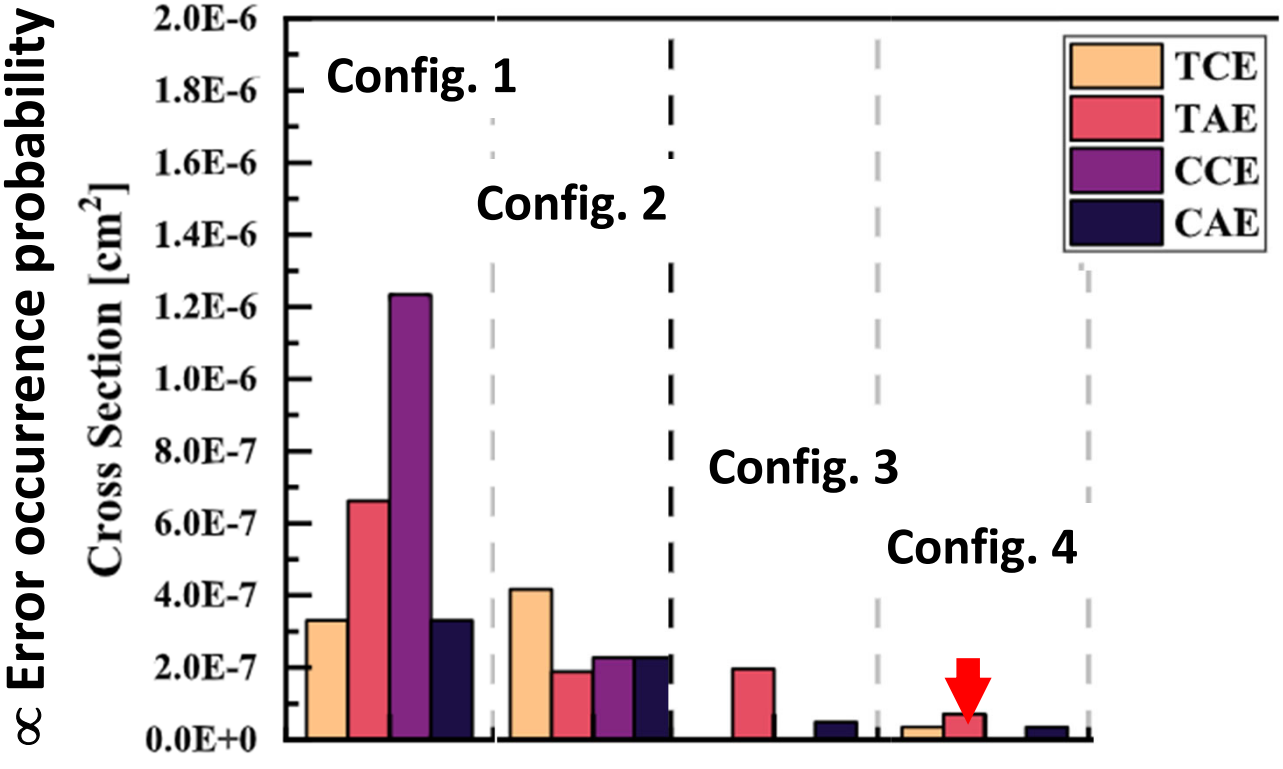



**After deploying TMRed I/DTCM in RISC-V core, CCEs almost decrease to zero and the other events also have a significant decrease. Errors in accelerator become dominant.**

# Irradiation results

| | Reset | TMR_I/DTCM | TMR_CSR |
|---|---|---|---|
| Config. 1 | No | No | No |
| Config. 2 | Yes | No | No |
| Config. 3 | Yes | Yes | No |
| Config. 4 | Yes | Yes | Yes |

- **Tolerable Core Event (TCE)**: RISC-V core has some misbehavior, but not affect NN application
- **Tolerable Accelerator Event (TAE)**: classification is correct, but the middle outputs are not as expected
- **Critical Core Event (CCE)**: RISC-V core runs away or crashes
- **Critical Accelerator Event (CAE)**: accelerator has no correct response or correct classification result



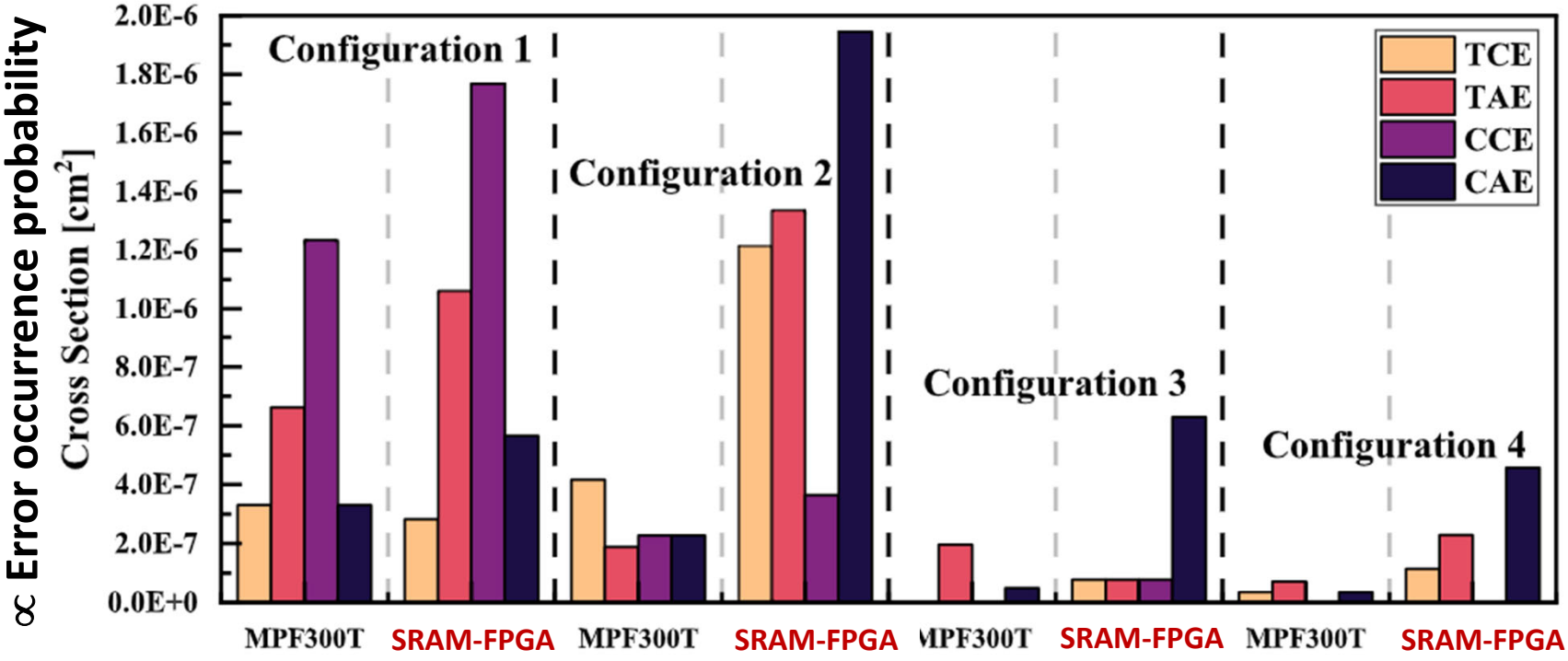

**After deploying TMRed, errors generated at accelerator decreased.**

**Overall cross section is reduced by 78.05% compared with Config. 1.**

# Comparison to SRAM FPGA

| | Reset | TMR_I/DTCM | TMR_CSR |
|---|---|---|---|
| Config. 1 | No | No | No |
| Config. 2 | Yes | No | No |
| Config. 3 | Yes | Yes | No |
| Config. 4 | Yes | Yes | Yes |

- **Tolerable Core Event (TCE)**: RISC-V core has some misbehavior, but not affect NN application
- **Tolerable Accelerator Event (TAE)**: classification is correct, but the middle outputs are not as expected
- **Critical Core Event (CCE)**: RISC-V core runs away or crashes
- **Critical Accelerator Event (CAE)**: accelerator has no correct response or correct classification result




**CRAM errors limit the reliability improvement in configs. 2, 3, and 4.**

# What we learned

- RISC-V core is more vulnerable than accelerator.
- Implementing mitigation techniques (e.g., Error Correcting Codes (ECC), TMR) is necessary for instruction and data memory to strengthen the SoC.
- After this, the vulnerability of the interface b/w processor and accelerator becomes visible.
- Impact of precision differences in NNs is limited.
- After deploying the above-mentioned countermeasures, the accelerator errors are dominant. Now, weight protection needs consideration.
- Specific application requirements decide whether to deploy the mitigation techniques above. Depending on the system criticality and required reliability level, a combination of these techniques are necessary to ensure the SoC's overall reliability to potential faults.

# Related work in edge AI SoCs

- EdgeAI devices, e.g., Google's Tensor Processing Unit [5] and NeuroShield [6], have undergone testing.
- Experiments with the NeuroShield and TPU indicate they have fewer errors compared to GPUs, along with a more straightforward error pattern where fewer outputs are affected and the erroneous values closely resemble the correct ones.
- Consequently, the rate of misclassification in neural networks on EdgeAI hardware is less than that on other platforms.

[5] R. L. Rech Junior, et al., “High energy and thermal neutron sensitivity of google tensor processing units,” *IEEE Trans. Nuclear Science*, 2022.
[6] S. Blower et al., “Evaluating and mitigating neutrons effects on COTS Edge AI accelerators,” *IEEE Trans. Nuclear Science*, 2021.

# Agenda

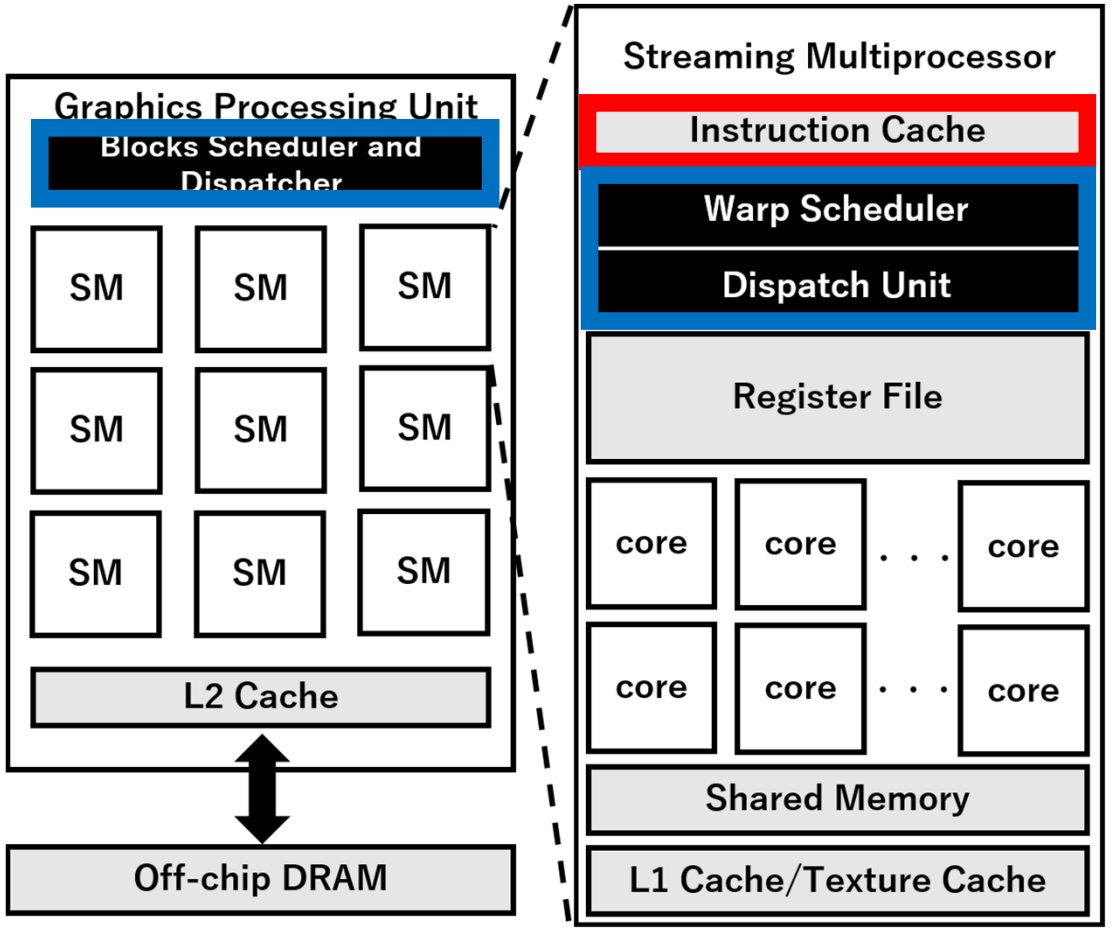
- Robustness of NNs
  - Case study (FP)
  - Identifying vulnerable weight parameters
  - Quantization
    - Multi-bit-width neural networks
- **Robustness of hardware**
  - Edge AI accelerator
  - **GPU**
- Countermeasures in literature

# Problem of application-level soft error evaluation in GPU

- **Circuit structure is not disclosed**
  - **Scheduler, dispatcher, etc.**
  - **A corruption in, e.g., scheduler, can impact multiple parallel processes.**
- **Instruction cache is invisible to users**
  - **Cache is accessed from multiple parallel processes.**

▼

- Target of fault injection is limited.
- Difficult to know part-wise SER contribution




not disclosed

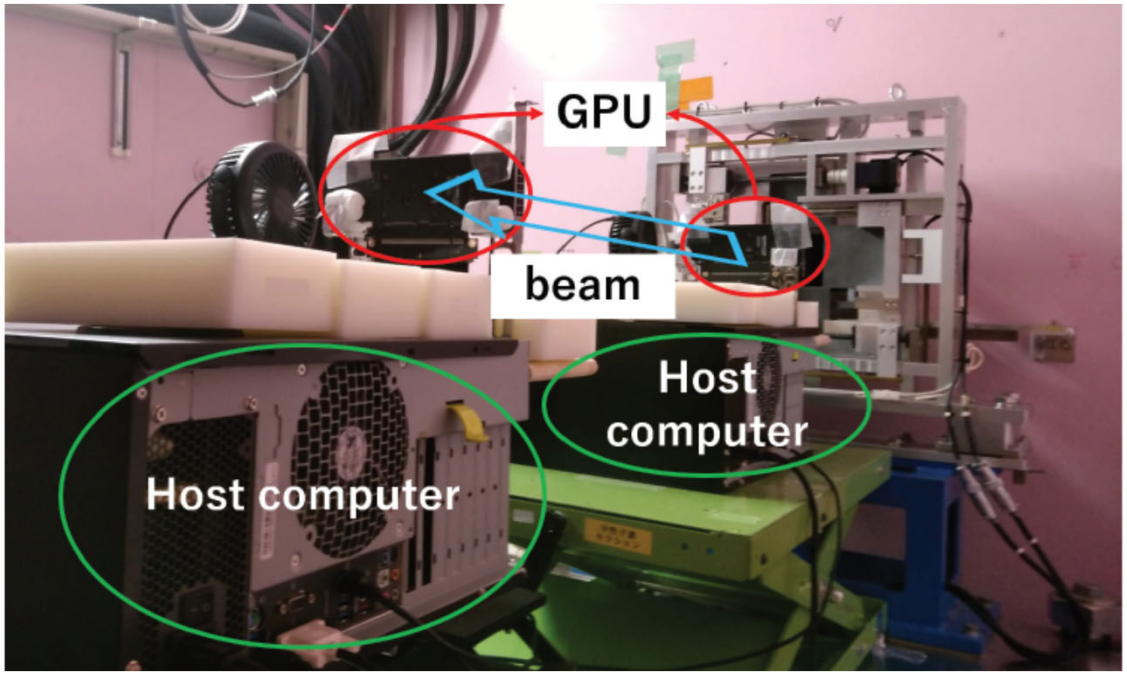
difficult to measure SER

# Assessing contribution from undisclosed components

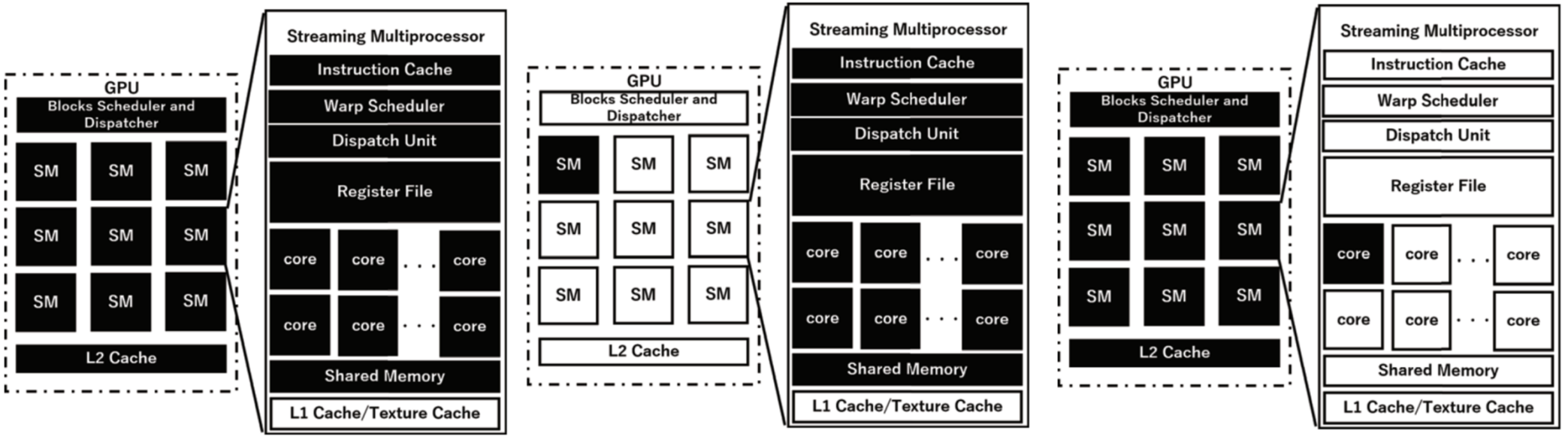
- **Carried out irradiation test for**
  - Error rates of disclosed memory components
  - SDC error rates of matrix multiplication programs
- **Compare measured SDC error rate and the one predicted only w/ disclosed memory components**
- **Difference is expected to come from undisclosed components**

K. Ito, et al., "Characterizing Neutron-Induced SDC Rate of Matrix Multiplication in Tesla P4 GPU," *RADECS*, 2019.

# GPU error rate measurement



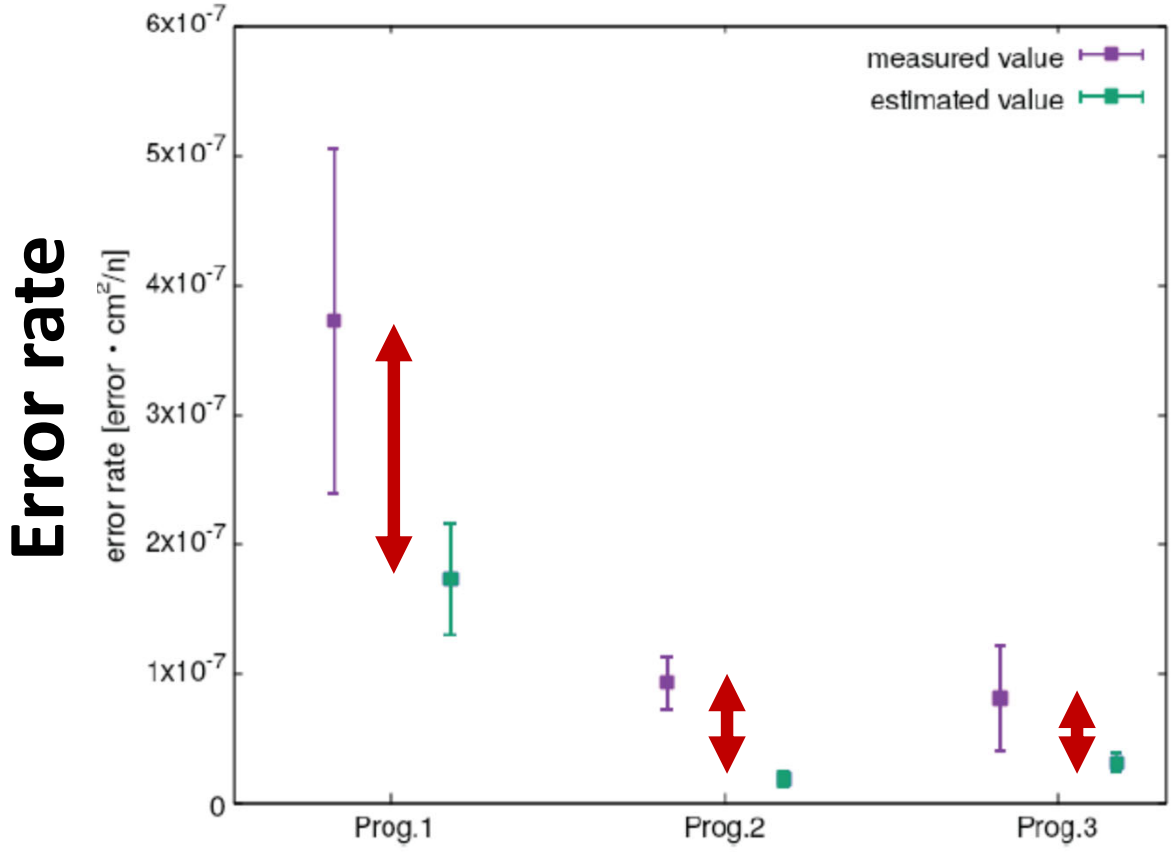

**Matrix multiplication programs w/ different resource usages were run**




(a) Program 1 (b) Program 2 (c) Program 3

# Comparison between measured and estimated error rates

- Estimated error rates using measured memory error rate and usage
  - Worst-case estimation assuming all errors induce SDC.
- Even with the worst-case estimation, there is a large discrepancy coming from errors in internal undisclosed hardware



# Difficulties in fault injection and radiation experiment

**Difficulties in fault injection**

- In high-level fault injection, faults can be injected only on that subset of resources which is visible to the programmer.
- Considering faults in computing resources (such as the pipelines, the control units, functional units, or scheduler), evaluating the impact on the software is not trivial.

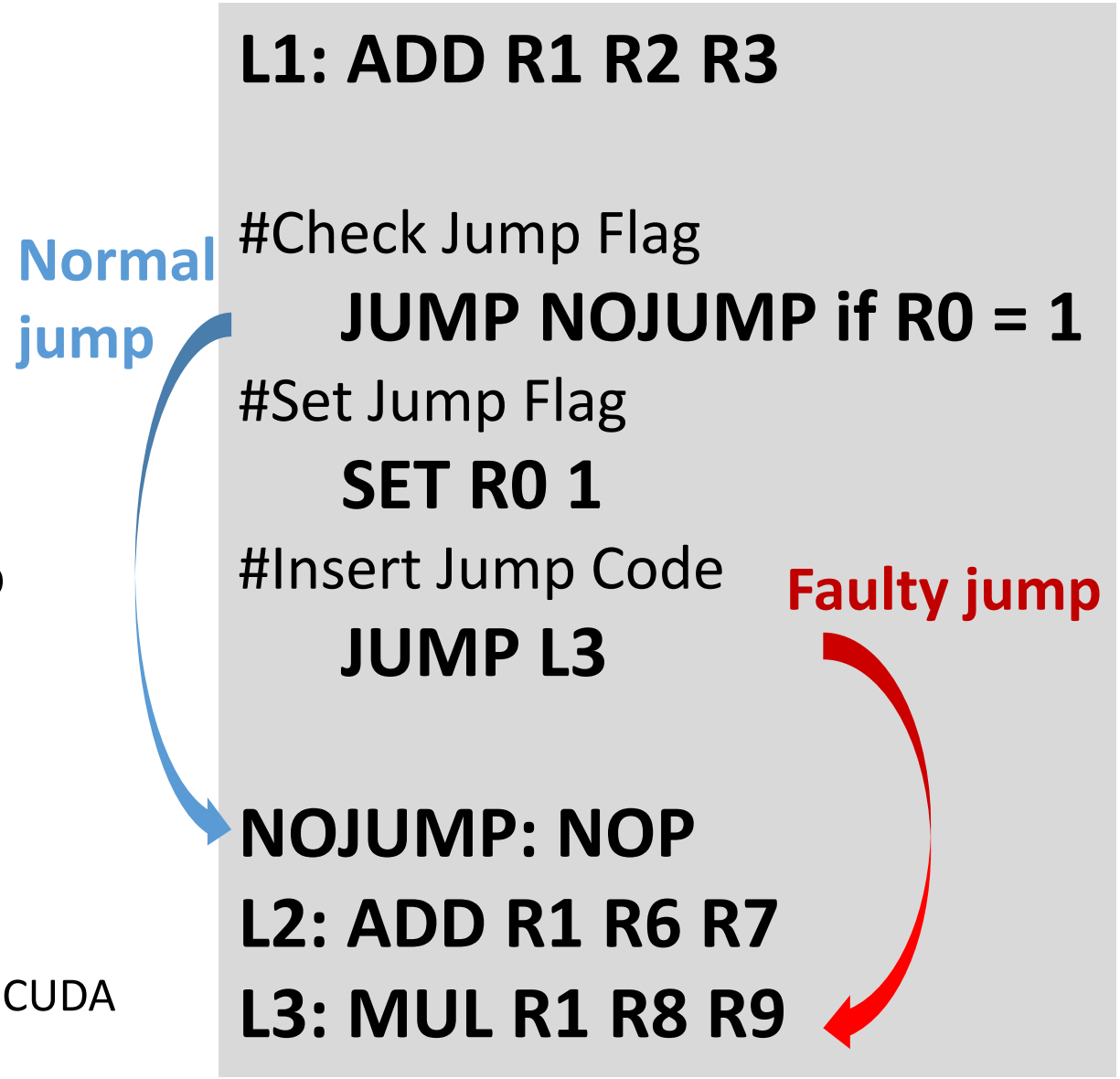
**Difficulties in radiation experiments**

- Radiation experiments do not allow to track faults propagation, preventing us from associating observed behaviors with the fault source and, thus, identifying the most vulnerable resources.
- Results are valid only for the particular codes and configurations that have been tested.

# Example of fault injection to control flow

- Inject error into one warp by editing PTX code.
  - PC in one of active warps is changed
  - Faulty jump can go to any labels
  - Jump flag is for jumping only once
  - Loops are unrolled

PTX: pseudo assembly language for CUDA




K. Ito, et al., "Analyzing DUE Errors on GPUs with Neutron Irradiation Test and Fault Injection to Control Flow," *IEEE Trans. Nuclear Science*, 2021.

# Possible direction for reliability assessment

- Low-level fault injection to RTL could reproduce the hardware behavior.
- However, COTS devices do not provide RTL designs. Also, considering the slow RTL simulation, various hardware configurations and many software applications, the low-level fault injection suffers from simulation time.

- Fault injection complemented with beam experiments is one possible direction when dealing with complex hardware.

# Constructing a model capable of estimating various GPU applications

Target model: $y = \sum_{k=1}^{n} a_k x_k + b$

$y$: SER (response variable)
$x_k$: app. info. (Explanatory variable)
$a_k, b$: constants to be obtained

**Irradiation exp. w/ various apps.**
**Prepare a number of app. metrics**

**Select primary $x_k$ according to correlation b/w $x_k$ and measured SER $(y)$**

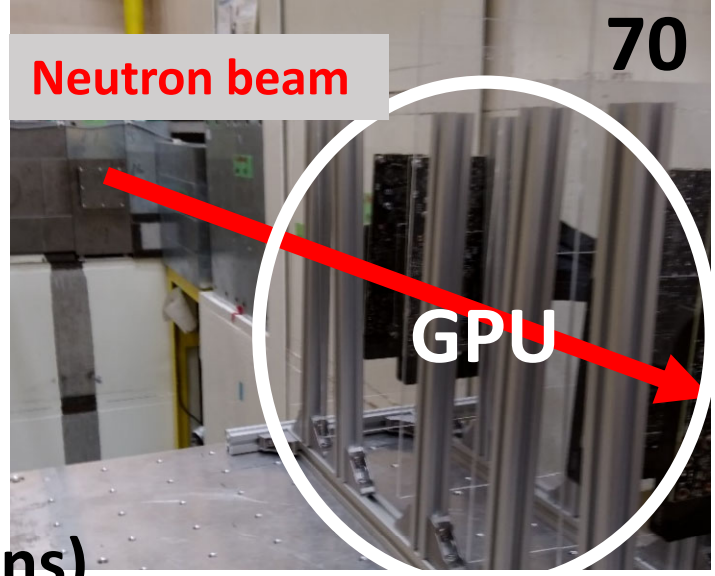
**Regression to obtain $a_k$(k=1,2,..,n), and $b$**

K. Ito, et al., "Constructing Application-Level GPU Error Rate Model with Neutron Irradiation Experiment," *RADECS*, 2021.

# AVF/PVF

- The probability for an error to propagate from memory elements to software visible state and modify the software execution (thus becoming a failure, such as an SDC or DUE) is called Architectural/Program Vulnerability Factors (AVF/PVF).
  - Depending on papers, AVF and PVF are differently defined.
- AVF/PVF from errors in memory visible to programmers can be easily obtained via high-level fault injection.

# Programs used in experiments




Exp. site: Tohoku Univ. CYRIC
GPU: NVIDIA Quadro P2000

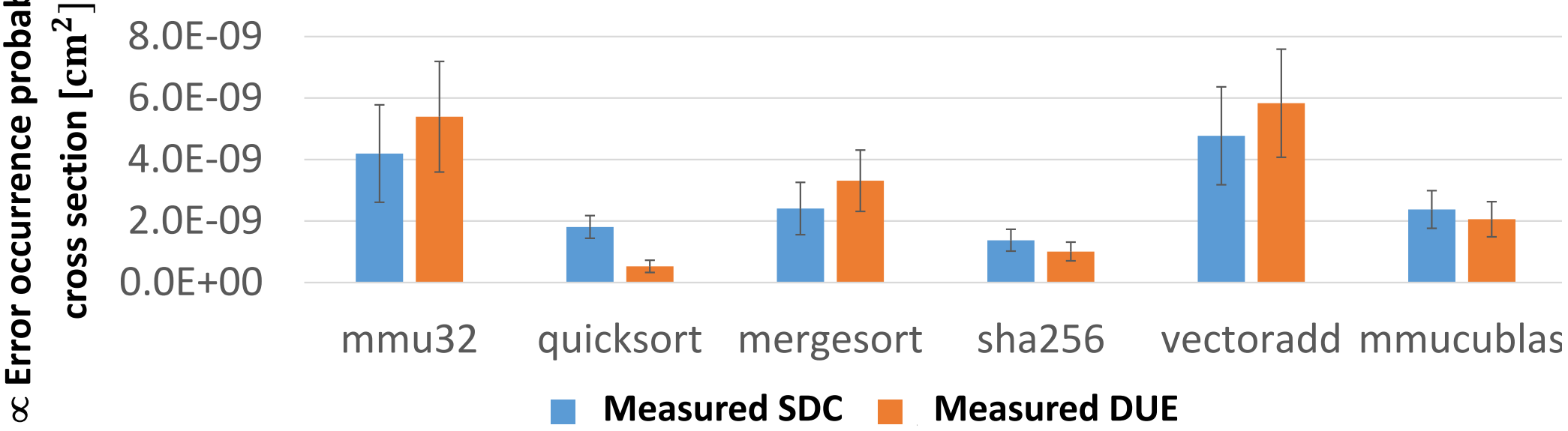
**Prepared applications w/ different behaviors (e.g., mem. size, #blocks, #threads, instructions)**

**Instruction proportion**

| | INT/FLOAT | LDG/STG | LDS/STS | BRANCH | CONTROL | MOVE |
|---|---|---|---|---|---|---|
| Mmu32 (matrix multiplication) | 46.4% | 2.2% | 43.5% | 1.1% | 3.3% | 0.4% |
| Quicksort (sort) | 43.1% | 11.6% | 0.0% | 14.2% | 8.6% | 19.7% |
| Mergesort (sort) | 27.7% | 0.5% | 13.1% | 15.9% | 10.8% | 11.4% |
| Sha256 (hash) | 72.7% | 10.2% | 0.0% | 4.7% | 7.2% | 4.8% |
| Vectoradd (parallel add) | 52.2% | 13.0% | 0.0% | 4.4% | 8.7% | 4.4% |
| Mmucublas (mat. mul. library) | 88.3% | 2.2% | 7.6% | 1.3% | 0.5% | 0.1% |

# Measured error rates

- SDC (silent data corruption)
  - Wrong output
  - **Detection is difficult**
- DUE (detectable but uncorrectable error)
  - Crash, hang, etc.
  - **Necessary to reboot GPU or host PC**

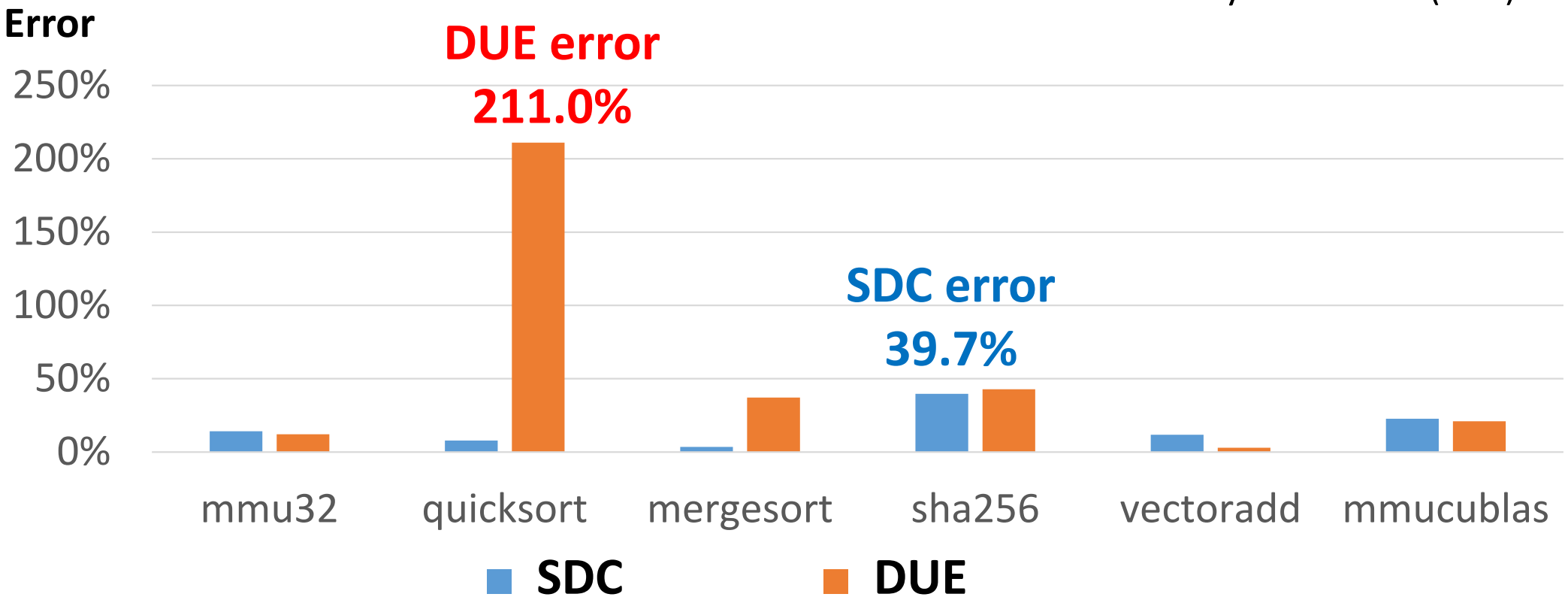


# Single-variable model

- "Warps per block" is primary explanatory variable
- $y = a_1 \times (Warps\ per\ block) + \mathbf{b}$
- Error for SDC is up to **39.7%**
- Error for DUE is up to **221.0%**

| | Corr. Coeff. (SDC) | Corr. Coeff. (DUE) |
|---|---|---|
| Warps per block | 0.98 | 0.96 |
| Dispatched warps | 0.80 | 0.89 |
| Gld_efficiency | 0.76 | 0.80 |
| Gst_efficiency | 0.75 | 0.80 |
| Warp exec. efficiency | 0.75 | 0.80 |

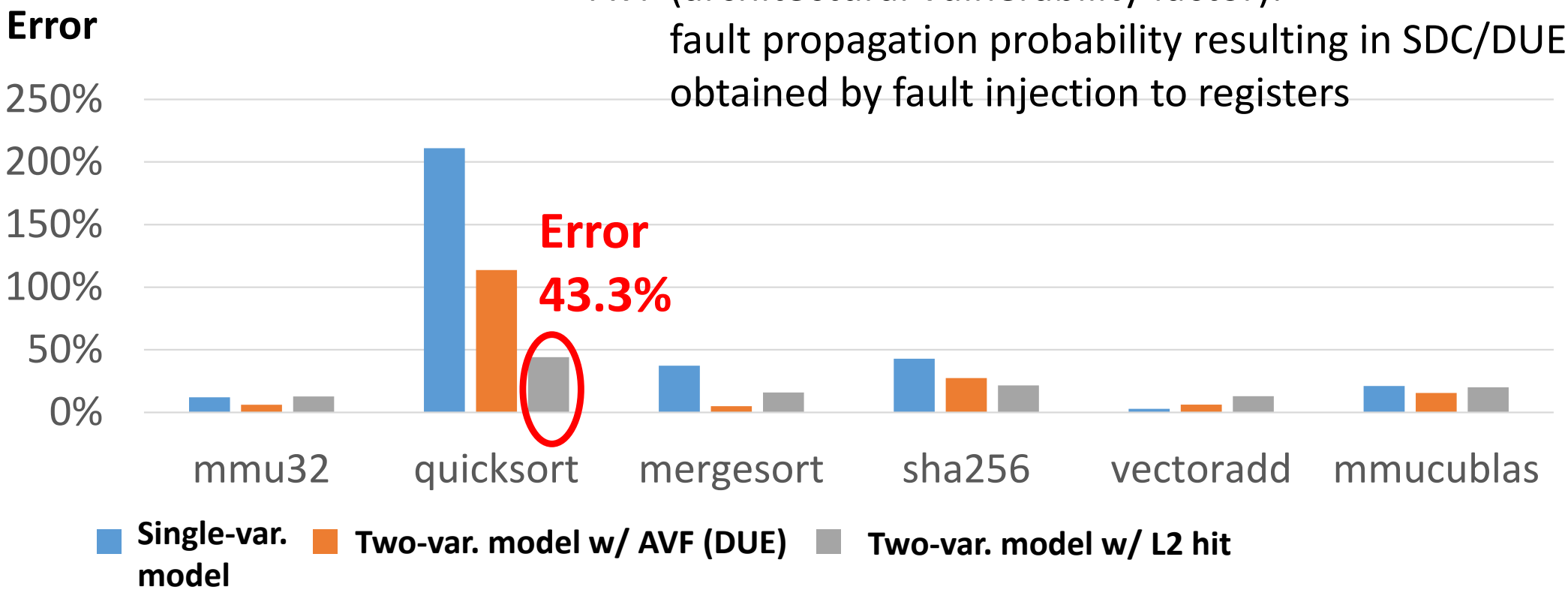
Sorted by corr. coeff. (SDC)



# Two-variable model

- $y = a_1 \times (Warps\ per\ block) + a_2 \times (new\ var.) + \mathbf{b}$
- **SDC model does not improve w/ any new variables**
- **DUE model improves w/ AVF (DUE) and L2 hit**
  - L2 hit reduced the maximum error to **43.3%**

AVF (architectural vulnerability factor):
fault propagation probability resulting in SDC/DUE obtained by fault injection to registers



# Related work in GPUs (1/2)

- ECC (Error-Correcting Code) can lower the GPU error rate by an order of magnitude, but it is less effective at decreasing the number of radiation-induced misclassifications in CNNs [7], [8].
- Reducing execution speed does not affect the Fault In Time (FIT) rate, whereas utilizing more parallel resources or larger hardware cores can increase the FIT rate, albeit with a possible performance advantage. Metrics such as Mean Executions Before Failure (MEBF), could be suitable to balance error rate with performance [9], [10], [11].
- Neutron beam experiments in [9] indicate that a higher # of parallel processes can overburden the scheduler, leading to increased error rates in GPUs [45]. The use of GPU resources more intensively raises the susceptibility to errors [12].

[7] F. F. d. Santos, et al., "Analyzing and increasing the reliability of convolutional neural networks on GPUs," *IEEE Trans. Reliability*, 2019.
[8] D. A. G. Goncalves de Oliveira, et al., "Evaluation and mitigation of radiation-induced soft errors in graphics processing units," *IEEE Trans. Computers*, 2016.
[9] P. Rech, L. L. Pilla, P. O. A. Navaux, and L. Carro, "Impact of GPUs parallelism management on safety-critical and HPC applications reliability," *DSN*, 2014.
[10] C. Weaver, et al., "Techniques to reduce the soft error rate of a high-performance microprocessor," *ISCA*, 2004.
[11] G. Reis, et al., "Design and evaluation of hybrid fault-detection systems," *ISCA*, 2005.
[12] J. M. Badia, et al., "Reliability evaluation of LU decomposition on GPU-accelerated system-on-chip under proton irradiation," *IEEE Tran. Nuclear Science*, 2022.

# Related work in GPUs (2/2)

- Algorithms that are slower and memory-bound tend to be more susceptible to errors, whereas the most efficient algorithms exhibit a smaller error cross section [13].
- Corruption of shared resources such as caches or the scheduler can disrupt multiple parallel processes [14][15].
- The severity of corruption (value difference) is influenced by the parallel architecture and the specific algorithm being run [16], [17], [18].

[13] J. M. Badia, et al., “Comparison of parallel implementation strategies in GPU-accelerated system-on-chip under proton irradiation,” *IEEE Trans. Nuclear Science*, 2022.
[14] P. Rech, et al., “An efficient and experimentally tuned software-based hardening strategy for matrix multiplication on GPUs,” *IEEE Trans. Nuclear Science*, 2013.
[15] L. L. Pilla, et al., “Software-based hardening strategies for neutron sensitive FFT algorithms on GPUs,” *IEEE Trans. Nuclear Science*, 2014.
[16] D. Oliveira, et al., “Experimental and analytical study of Xeon Phi reliability,” SC, 2017.
[17] D. A. G. D. Oliveira, et al., “Radiation-induced error criticality in modern HPC parallel accelerators,” *HPCA*, 2017.
[18] V. Fratin, et al., “Code-dependent and architecture-dependent reliability behaviors,” *DSN*, 2018.

# Agenda

- Robustness of NNs
  - Case study (FP)
  - Identifying vulnerable weight parameters
  - Quantization
    - Multi-bit-width neural networks
- Robustness of hardware
  - Edge AI accelerator
  - GPU
- **Countermeasures in literature**

# Software countermeasure

- Implement cheap concurrent replication with idle hardware
  - Selective replication for protecting only the most critical layers or portion of the neural network [19], [20]–[22].
- Stop errors propagating in CNNs by checking whether the propagated values during MaxPooling layers, detecting up to 85% of critical errors in CNNs [7].

[19] F. Libano, et al., “Selective hardening for neural networks in FPGAs,” *IEEE Trans. Nuclear Science*, 2019.
[20] L. Weigel, et al., “Kernel vulnerability factor and efficient hardening for histogram of oriented gradients,” *DFT*, 2017.
[21] A. Ruospo, et al., “Selective hardening of critical neurons in deep neural networks,” *DDECS*, 2022.
[22] C. Bolchini, et al., “Selective hardening of CNNs based on layer vulnerability estimation,” *DFT*, 2022.

# Algorithm-Based Fault-Tolerant (ABFT)

- ABFT for matrix multiplication [23] detects and corrects more than 80% of errors. When applied to CNNs, ABFT outperformed ECC and duplication [7]. Smart light-ABFT [24] further reduces the overhead for GPUs.
- Concurrent signature calculations and signature comparison for matrix multiplication [25]

[23] P. Rech, et al., “An efficient and experimentally tuned software-based hardening strategy for matrix multiplication on GPUs,” IEEE Trans. Nuclear Science, 2013.
[24] S. Hari, et al., “Making convolutions resilient via algorithm-based error detection techniques,” IEEE Trans. Dependable and Secure Computing, 2022.
[25] H. Itsuji, et al., "Concurrent Detection of Failures in GPU Control Logic for Reliable Parallel Computing," ITC, 2020.

# Algorithm countermeasure

- Assess and contrast consecutive input frames against their corresponding detection outputs. Similar frames should yield similar detection results. A discrepancy may raise an error alert. 70% of critical errors are detected though producing some false positives [26].
- Reduced Precision Duplication With Comparison (RD-DWC) has been applied to GPUs and has demonstrated error detection rates of 75% in average with acceptable additional overhead [27].

[26] L. K. Draghetti, et al., “Detecting errors in convolutional neural networks using inter frame spatio-temporal correlation,” *IOLTS*, 2019.
[27] F. F. dos Santos, et al., “Reduced precision DWC: An efficient hardening strategy for mixed-precision architectures,” *IEEE Trans. Computers*, 2022.

# Fault aware training

- If the DNN is trained to classify objects correctly even w/ transient faults, it is possible to produce a more reliable model while maintaining the original accuracy.

▼

Training is performed by injecting transient faults.

- The model is expected to autonomously learn how to properly deal with faults to reduce mispredictions [28][29].

Proposed vulnerability model can improve the noise injection efficiency during the training.

[28] G. Gambardella, N. J. Fraser, U. Zahid, G. Furano and M. Blott, "Accelerated Radiation Test on Quantized Neural Networks trained with Fault Aware Training," *AERO*, 2022.
[29] N. Cavagnero, F. D. Santos, M. Ciccone, G. Averta, T. Tommasi and P. Rech, "Transient-Fault-Aware Design and Training to Enhance DNNs Reliability with Zero-Overhead," *IOLTS*, 2022.

# Conclusions

- Robustness of NNs
  - Case study (FP)
  - Identifying vulnerable weight parameters
  - Quantization
  - Multi-bit-width neural networks
- Robustness of hardware
  - Edge AI accelerator
  - GPU
- Countermeasures in literature

# For your reference

- Some excellent and complete surveys of the available reliability studies have been published.
  - F. Su, C. Liu, and H.-G. Stratigopoulos, “Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives,” *IEEE Design & Test*, vol. 40, no. 2, pp. 8–58, 2023.
  - Y. Ibrahim, H. Wang, J. Liu, J. Wei, L. Chen, P. Rech, K. Adam, and G. Guo, “Soft errors in DNN accelerators: A comprehensive review,” *Microelectronics Reliability*, vol. 115, p. 113969, 2020.
  - P. Rech, "Artificial Neural Networks for Space and Safety-Critical Applications: Reliability Issues and Potential Solutions," *IEEE Trans. Nuclear Science*, to appear.