Computation-oriented Fault-tolerance Schemes for RRAM-based Computing Systems

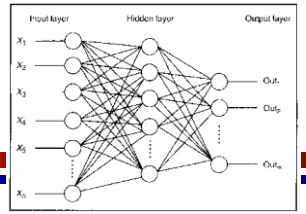
Wenqin Huangfu[1], Lixue Xia[1], Ming Cheng[1], Xilin Yin[1], Tianqi Tang[1], Boxun Li[1], Krishnendu Chakrabarty[2], Yuan Xie[3], Yu Wang[1], Huazhong Yang[1]

1 Dept. of E.E., Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing, China

2 Department of Electrical and Computer Engineering, Duke University North Carolina, USA

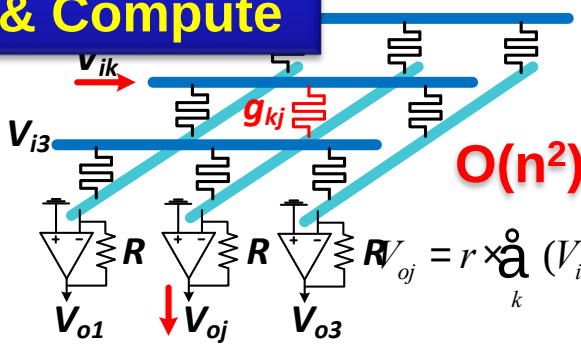
3 Department of Electrical and Computer Engineering, University of California at Santa Barbara, California, USA

RRAM-based Computation



- Emerging Devices, such as RRAM devices, provide a promising solution to realize better implementation of brain inspired circuits and systems in a high energy efficiency way

Merge Mem. & Compute



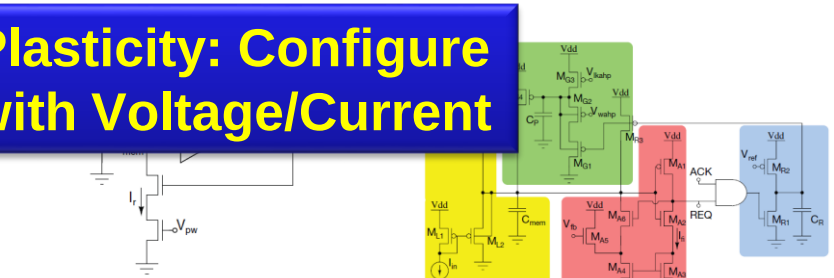
$O(n^2) \rightarrow O(n^0)$

$$V_{oj} = r \times \sum_k (V_{ik} \times g_{kj}),\ g_{kj} = \frac{1}{M_{kj}}$$

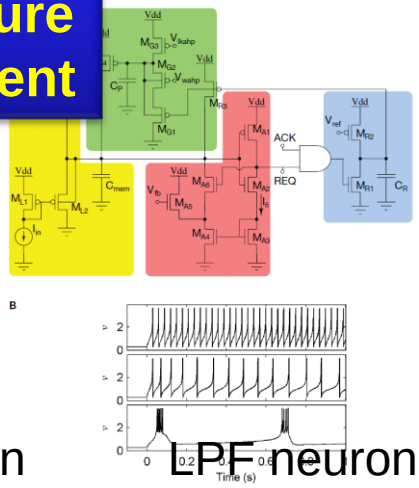
High Density

Fig. 1 Path Towards Ultra High Density NVM
(Samsung, 5th Workshop on Innovative Memory Technologies)

Plasticity: Configure with Voltage/Current



I&F neuron



LPF neuron

Non Volatile

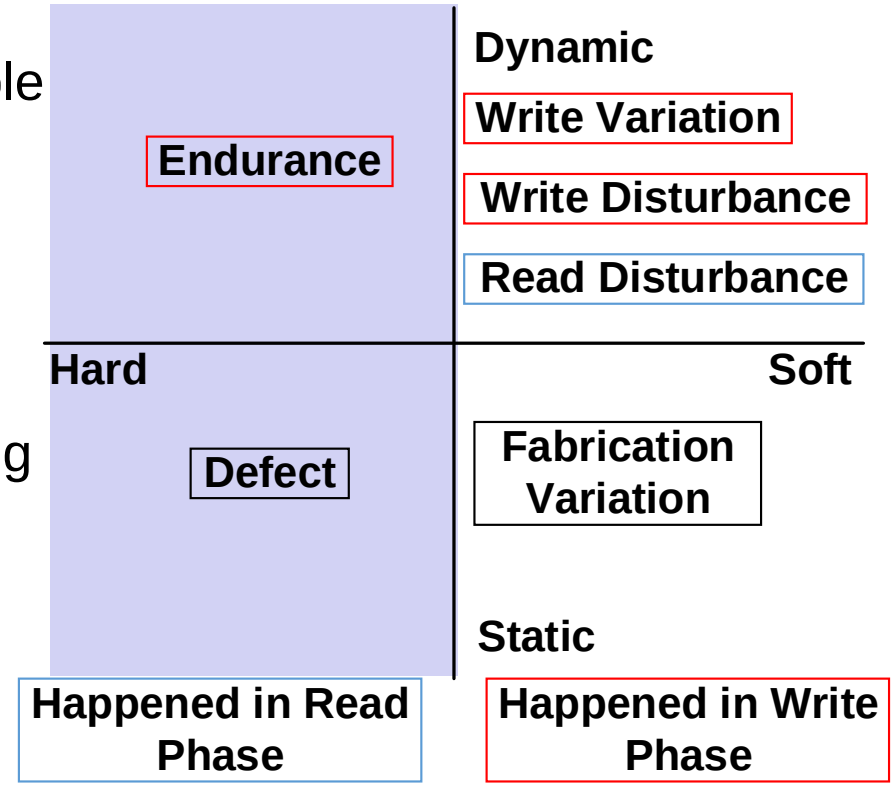
	Memristor	PCM	STT-RAM	DRAM	Flash	HD
Chip area per bit (F^2)	4	8–16	14–64	6–8	4–8	n/a
Energy per bit (pJ)[2]	0.1–3	2–100	0.1–1	2–4	10^1–10^4	10^6–10^7
Read time (ns)	<10	20–70	10–30	10–50	25,000	5–8x10^6
Write time (ns)	20–30	50–500	13–95	10–50	200,000	5–8x10^6
Retention	>10 years	<10 years	Weeks	<Second	~10 years	~10 years
Endurance (cycles)	~10^{12}	10^7–10^8	10^{15}	>10^{17}	10^3–10^6	10^{15} ?
3D capability	Yes	No	No	No	Yes	n/a

Related Work about RRAM-based NNs

- Architecture and circuit designs
  - RRAM-based Deep Neural Networks [Hu_DAC_2012,Chi_ISCA_2016,Liu_IEEE TCAS I_2016, etc]
  - RRAM-based Spiking NNs [Tang_DATE_2015,Hu_IEEE TCAD_2016]
  - RRAM-based Convolutional NNs [Wang_ISCAS_2016, Shafiee_ISCA_2016]
  - Training NN on RRAM [Hasan_IJCNN_2014]
- EDA tools
  - Simulator [Xia_DATE_2016]
  - Design optimization tool [Gu_ASPDAC_2015]
- Demonstration
  - NN/Hopfield Network [Prezioso_Nature_2015,Lee_IEDM_2016]
  - Convolver [Gao_IEEE Electron Device Letters_2016]

RRAM Device Meets Kinds of Faults

- **Fault Classification of RRAM Device:**
  - Hard Fault (Stuck-At Faults)
    - The resistance is unchangeable
  - Soft Fault
    - The resistance is changeable but not correct
  - Static Fault
    - Already happened before using
  - Dynamic Fault
    - Happened during using



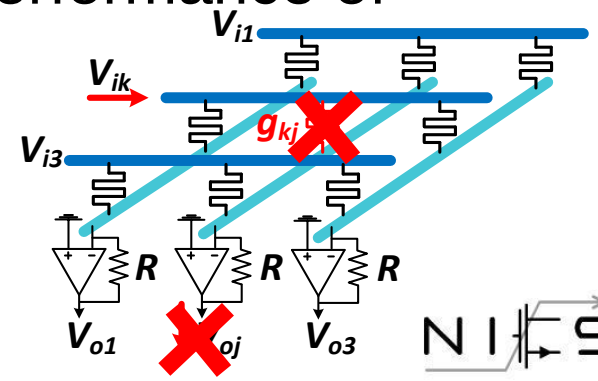
- Soft faults can be tolerant by inner fault tolerance of NN [Gu_ASPDAC_2015]
- Previous designs ignore the influence of hard faults

Impact of Hard Faults

- The yield (the percentage of available cells) of RRAM device varies from 60% to more than 90% in different materials and technologies [Chen_IEEE TC_2015]
- We test the classification accuracy of a 784x100x10 NN on *MNIST* test bench with random stuck-at-0 and stuck-at-1 faults:

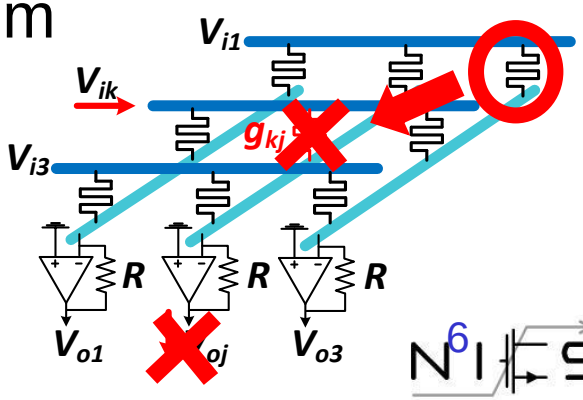
Yield	Ideal	95%	90%	80%
Accuracy	97.8%	26.7~60.4%	15.5~38.6%	10.6~28.0%
Reduction	-	>37%	>59%	>69%

- Hard faults can obviously influence the performance of RRAM-based computation



Fault-tolerant Research

- Memory-oriented methods use redundant cells and arrays to substitute faulty cells [Wang_DAC_12,Koh_ICCD_09]
- These methods are not available for RRAM-based computation
  - Redundant cell is helpless: the basic unit is the whole column instead of single cell
  - Unable to directly shut down faulty cells: if the size of RRAM crossbar array is M * N, M * N control lines are needed to control every cell independently, which is unacceptable
- We need computation-oriented research to tolerant the hard faults in RRAM-based computation system

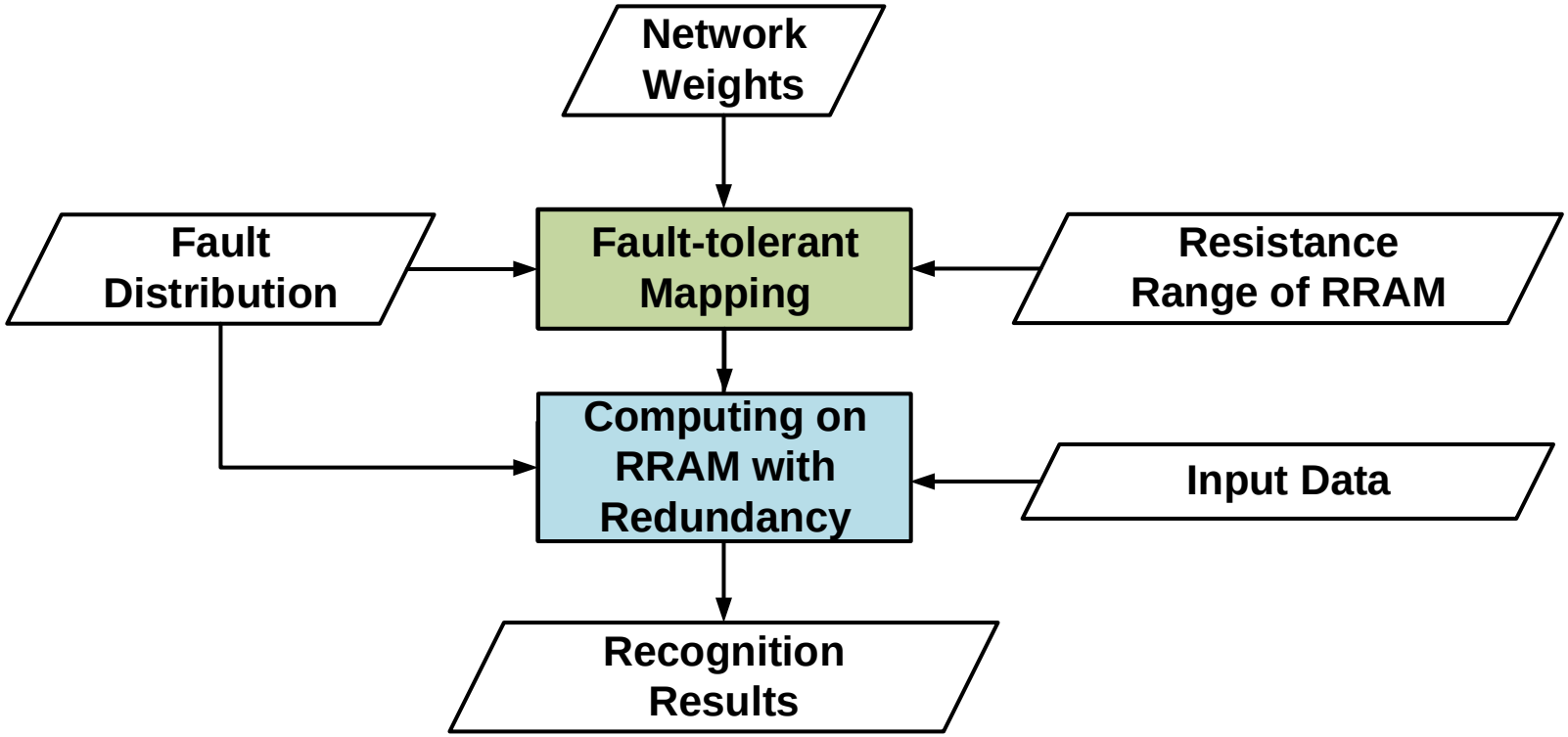


Fault-tolerant Method for RRAM-based Neural Computing System

- Background
  - NNs on RRAM
  - RRAM Faults
- Proposed Methods
  - Fault-tolerant mapping and redundant scheme
- Future Work

Proposed Methods

- Workflow of RRAM-based Computation

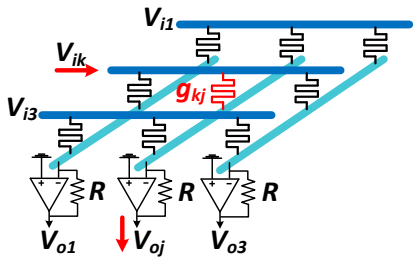


- Mapping algorithm with inner fault-tolerant ability
- Redundant schemes and circuits
  - Redundant crossbars
  - Independent Redundant Columns

Mapping Algorithm

- Traditionally, two RRAM crossbars are needed to represent the positive and negative values of matrix:

$$\begin{aligned}\vec{V_o} &= C^+ \cdot \vec{V_i} + C^- \cdot -\vec{V_i} \\ &= (C^+ - C^-) \cdot \vec{V_i} \\ &= C \cdot \vec{V_i}\end{aligned}$$

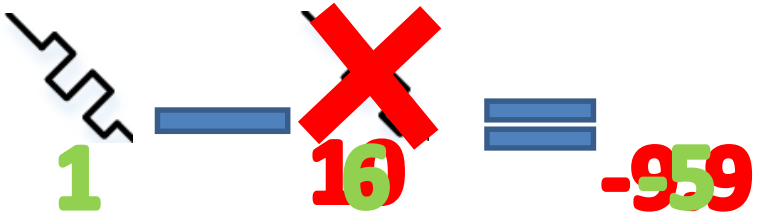


- Original mapping algorithm accords to the sign of weights to determine the RRAM values:

$$c^+_{k,j} = \begin{cases} c_{k,j} & c_{k,j} > 0 \\ 0 & c_{k,j} \leq 0 \end{cases}$$

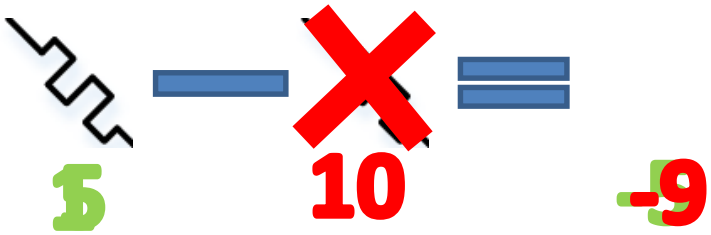
$$c^-_{k,j} = \begin{cases} -c_{k,j} & c_{k,j} < 0 \\ 0 & c_{k,j} \geq 0 \end{cases}$$

- This mapping method doesn't consider RRAM faults
  - Example
  - Target Value: -5
  - Range of value represent by RRAM devices: [1, 10]

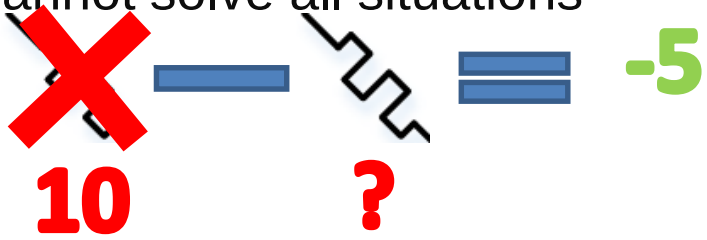


Mapping Algorithm

- However, in some situations, the influence of hard faults can be directly tolerant in mapping phase



- Proposed Mapping Method
  - Target: minimize the difference between the ideal value of matrix and the practical value represented by RRAM
  - Step 1: Initializing
  - Step 2: Greedily mapping the weights cell-by-cell
  - Cannot solve all situations



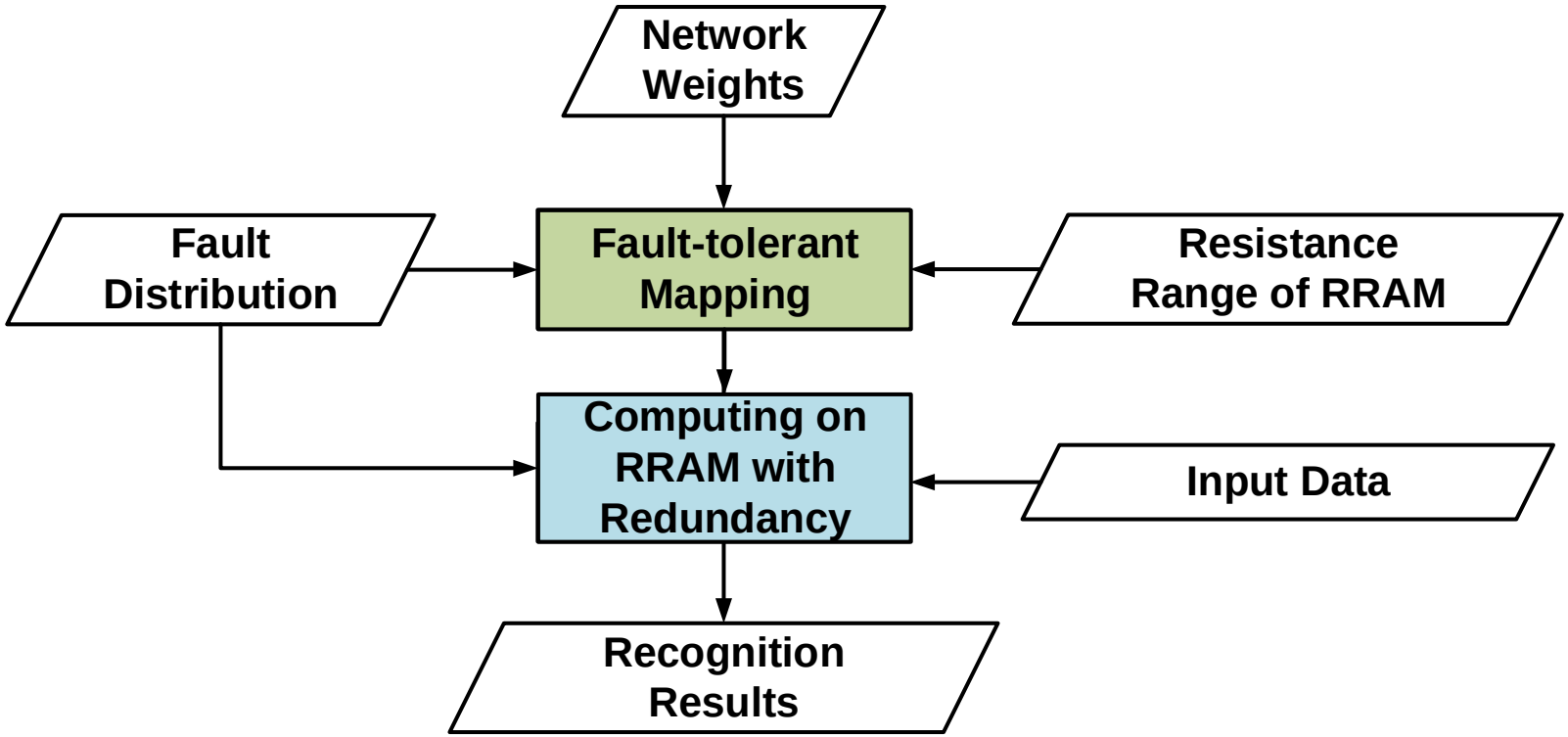
Algorithm 1: Mapping Algorithms with Inner Fault-Tolerant Ability

Input: C, G_{max}, G_{min} g_s, $StuckAtMap$, $Available\_Rdundant\_Cells$
Output: G^{+}_{Output}, G^{-}_{Output}

for $i = 1 : C.cells$ **do**
  if $G^{+}_{Output}(i)$ *doesn't get stuck* **then**
    $G^{+}_{Output} = Gmax$
  end
  if $G^{-}_{Output}(i)$ *doesn't get stuck* **then**
    $G^{-}_{Output} = Gmin$
  end
end
for $i = 1 : C.cells$ **do**
  for $j = 1 : Available\_Rdundant\_Cells(i)$ **do**
    $Diff = C(i) - (G^{+}_{Output}(i,:)/Gs - G^{-}_{Output}(i,:)/Gs)$;
    Adjust $G^{+}_{Output}(j)$ and $G^{-}_{Output}(j)$ to minimize $Diff$;
    $Diff = C(i) - (G^{+}_{Output}(i,:)/Gs - G^{-}_{Output}(i,:)/Gs)$;
    if $Diff == 0$ **then**
      break;
    end
  end
end
return G^{+}_{Output}, G^{-}_{Output}

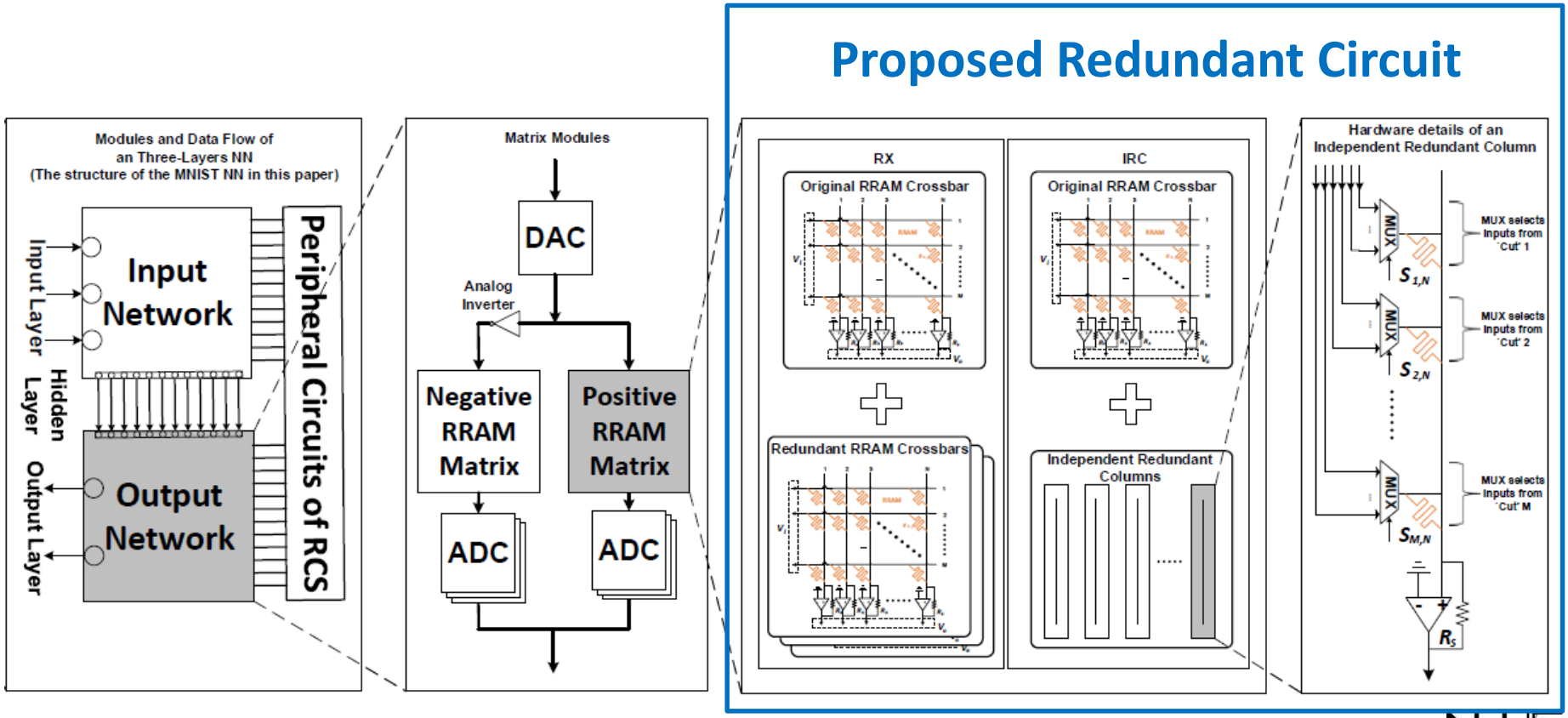
Proposed Method

- Mapping algorithm with inner fault-tolerant ability
- Redundant schemes and circuits



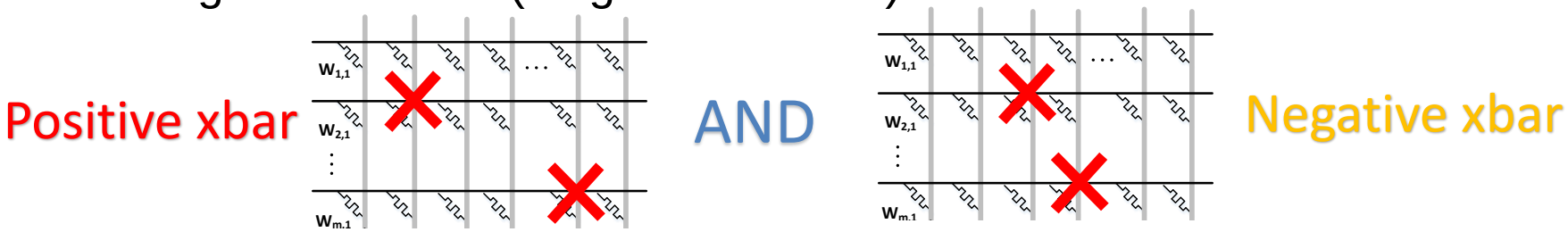
Hardware Structure

- Two Redundant schemes and corresponding redundant circuits are proposed
  - Redundant crossbars
  - Independent Redundant Columns

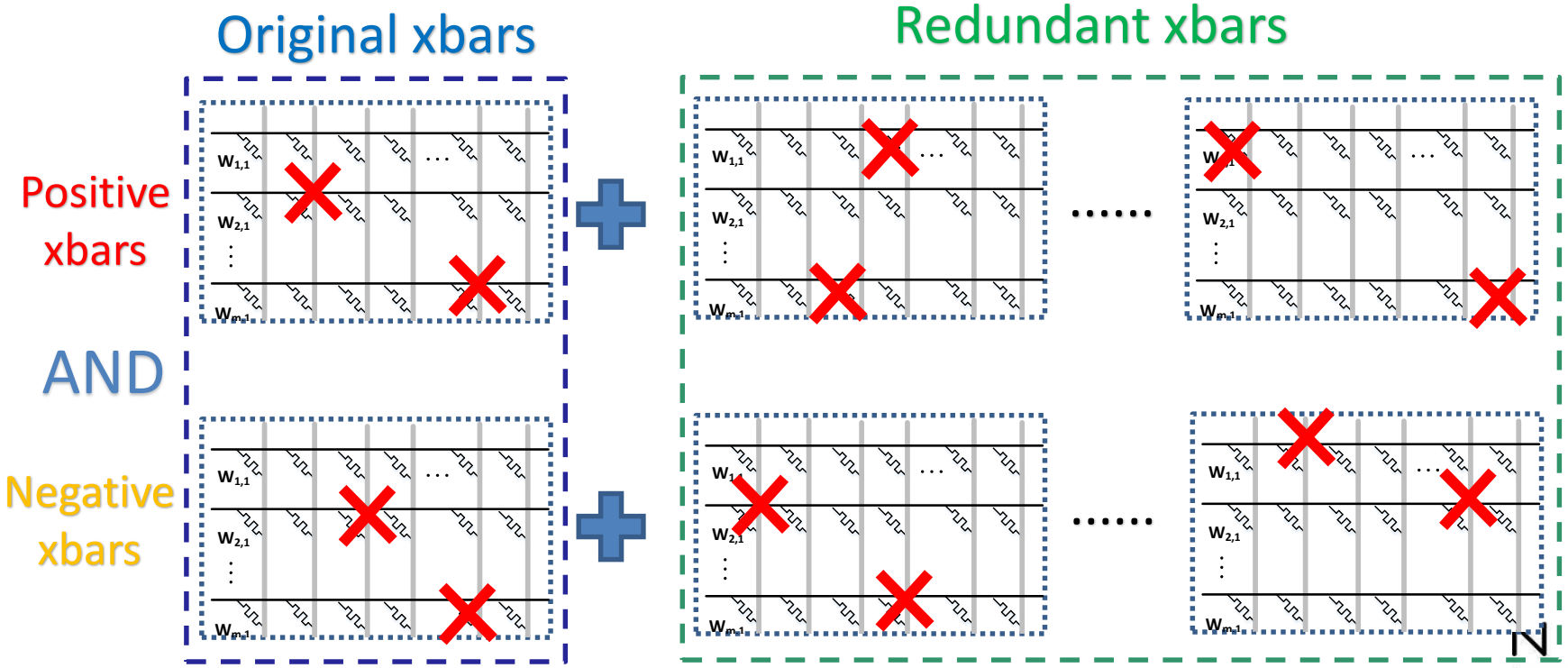


Redundant Crossbars

- Structure: Using multi-crossbars as redundant crossbars
  - Original structure (Original crossbar):



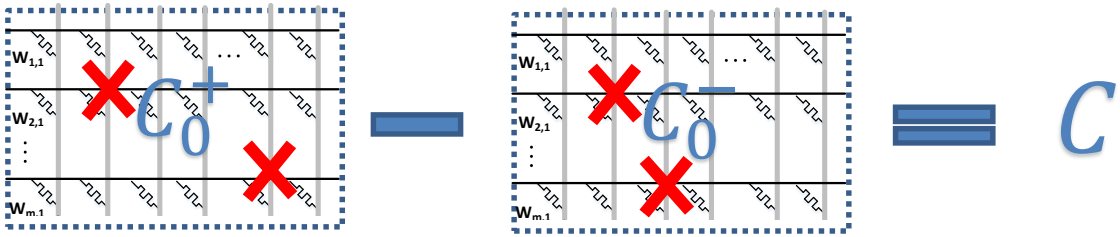
  - Modified structure (Original crossbar + Redundant crossbars):



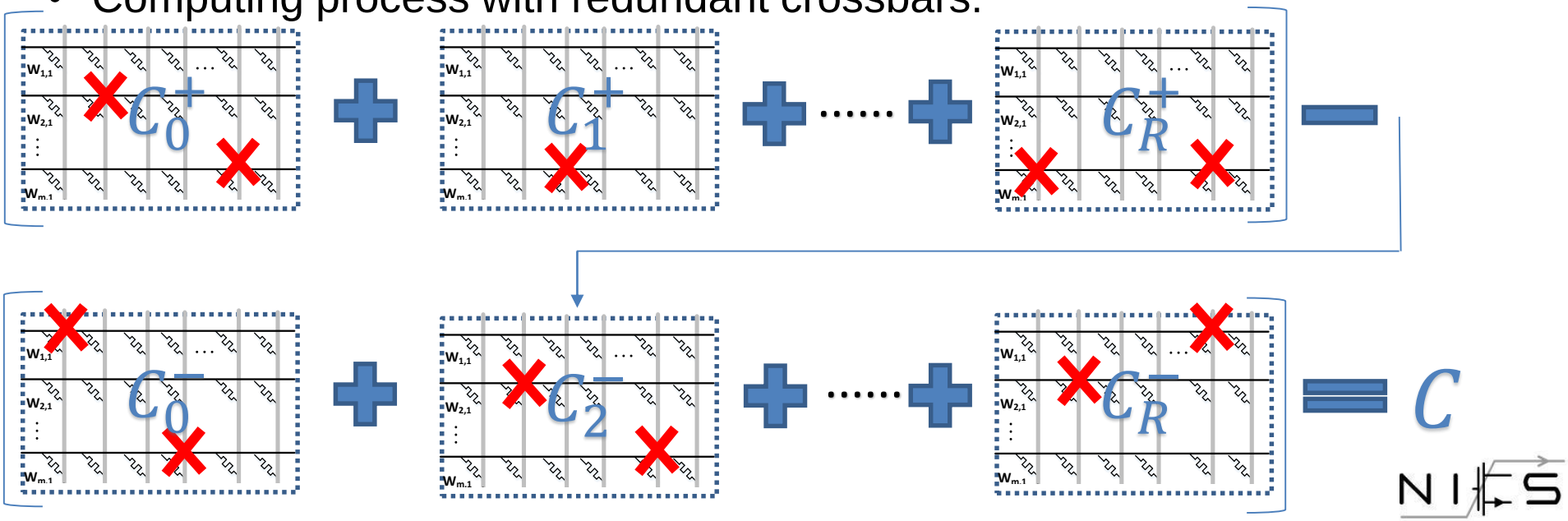
Redundant Crossbars

Mapping algorithm: Co-mapping different crossbars to get the best available results

- Target application matrix: **C**
- Original computing process:



- Computing process with redundant crossbars:



Redundant Crossbars

Cell-by-cell Mapping Process

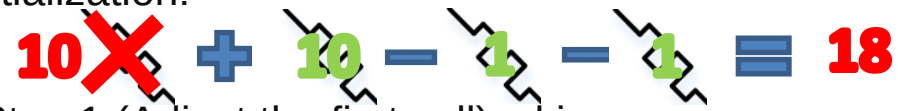
Target Value: -5

Range of value represent by

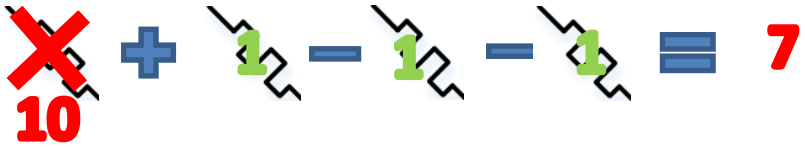
RRAM devices: [1, 10]

Redundancy Ratio (Number of Redundant Crossbars): 1

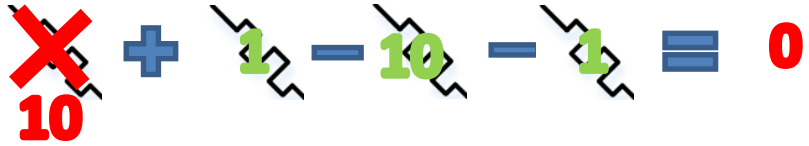
- Initialization:



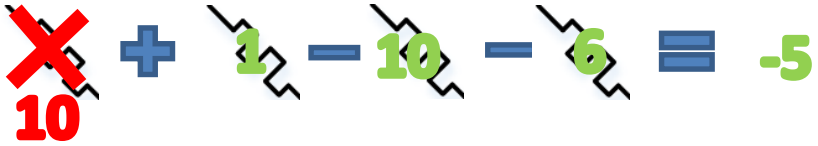
- Step 1 (Adjust the first cell): skip
- Step 2 (Adjust the second cell):



- Step 3 (Adjust the third cell):



- Step 4 (Adjust the fourth cell):



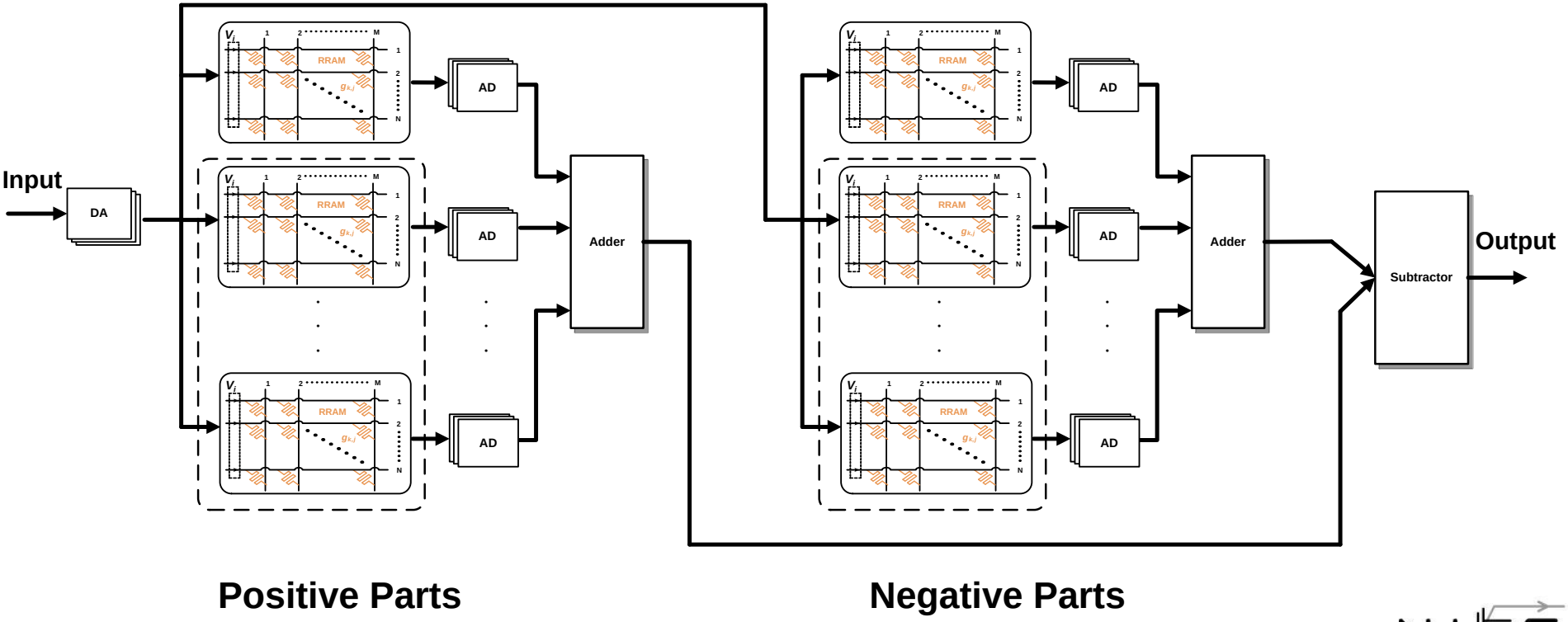
Algorithm 1: Mapping Algorithms with Inner Fault-Tolerant Ability

Input: C, G_{max}, G_{min} g_s, $StuckAtMap$, $Available\_Rdundant\_Cells$
Output: G^{+}_{Output}, G^{-}_{Output}

for $i = 1 : C.cells$ **do**
  if $G^{+}_{Output}(i)$ *doesn't get stuck* **then**
    $G^{+}_{Output} = Gmax$
  end
  if $G^{-}_{Output}(i)$ *doesn't get stuck* **then**
    $G^{-}_{Output} = Gmin$
  end
end
for $i = 1 : C.cells$ **do**
  for $j = 1 : Available\_Rdundant\_Cells(i)$ **do**
    $Diff = C(i) - (G^{+}_{Output}(i,:)/Gs - G^{-}_{Output}(i,:)/Gs)$;
    Adjust $G^{+}_{Output}(j)$ and $G^{-}_{Output}(j)$ to minimize $Diff$;
    $Diff = C(i) - (G^{+}_{Output}(i,:)/Gs - G^{-}_{Output}(i,:)/Gs)$;
    if $Diff == 0$ **then**
      break;
    end
  end
end
return G^{+}_{Output}, G^{-}_{Output}

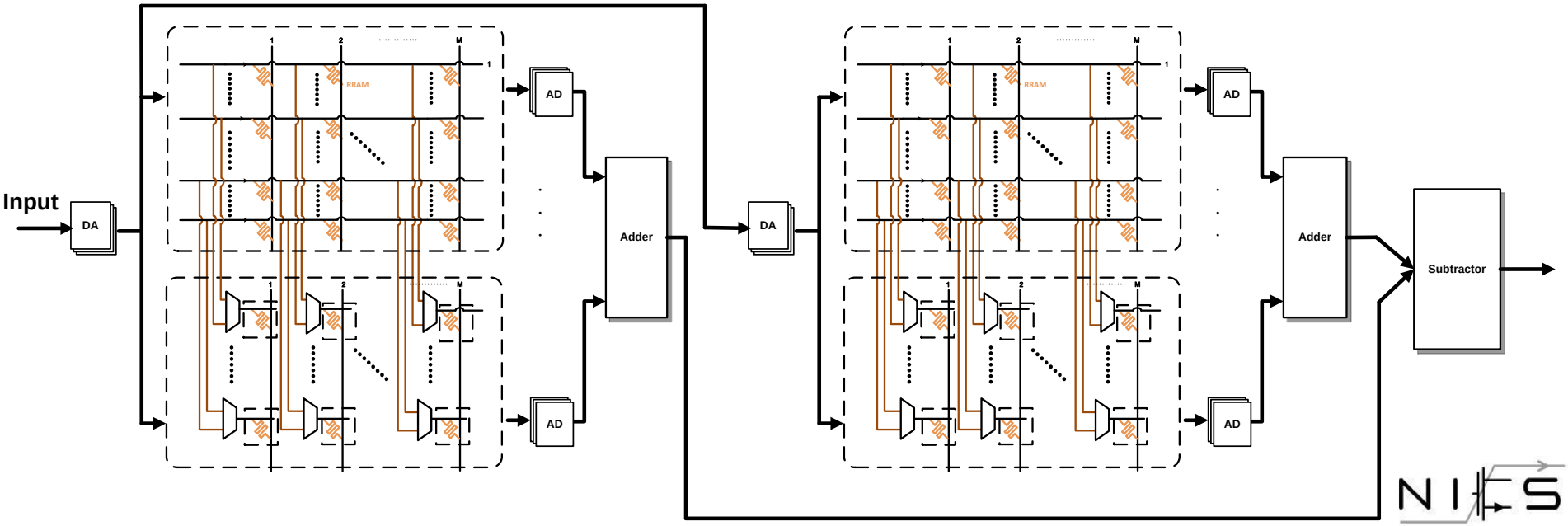
Redundant Crossbars

- Computing Process: merge the results from the original RRAM crossbar and the results from redundant crossbars together
- Definition: R is the redundant ratio, standing of extra RRAM crossbar arrays that used as redundant structure



Independent Redundant Columns

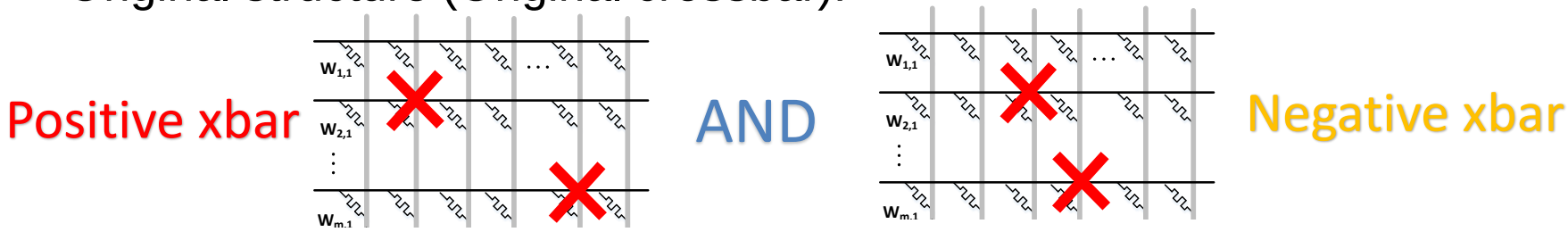
- Given the fault ratio of a single device is *P*, the size of RRAM crossbar array is *M* x *N*, the expectation of faults in each column is *P* x *M*
  - Solution: A smaller redundant unit which only contains *P* x *M* cells
  - Divide each column into *P* x *M* parts (called ‘cut’), while the expectation of faults in each cut is 1
  - *R* indicates the number of redundant devices in a redundant column for one cut



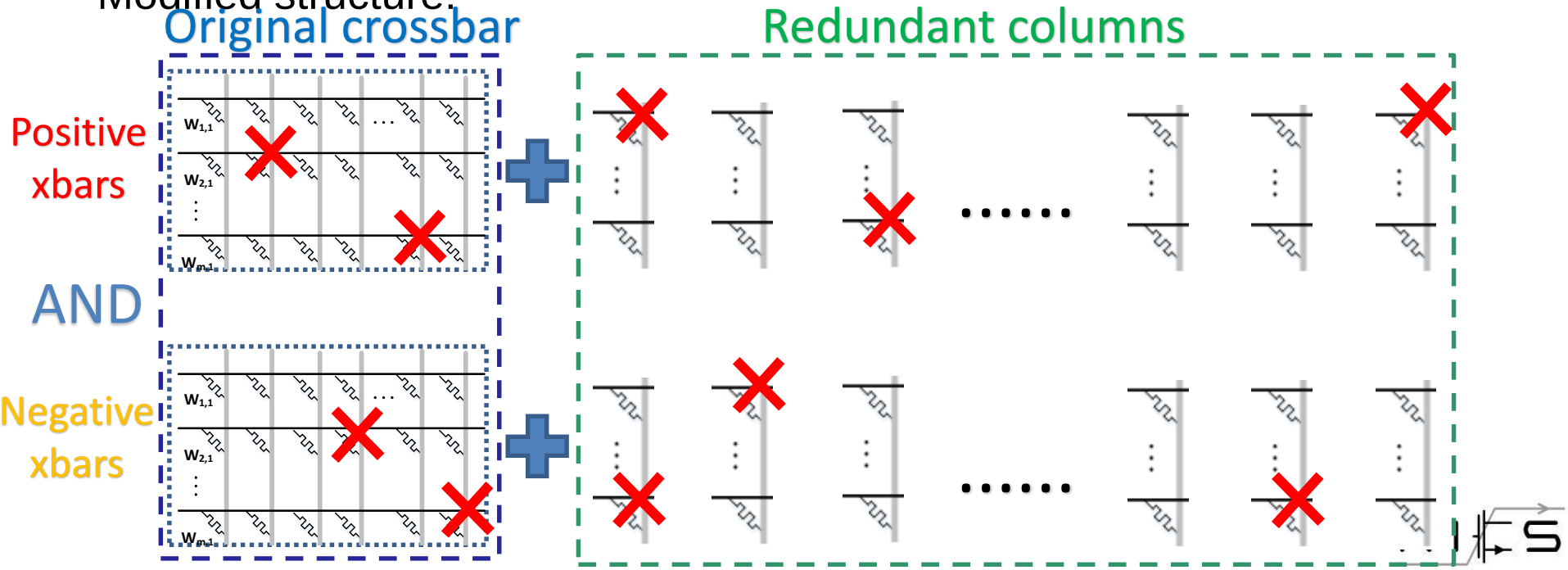
Independent Redundant Columns

Structure: Using small independent columns as Redundant Columns

- Original structure (Original crossbar):



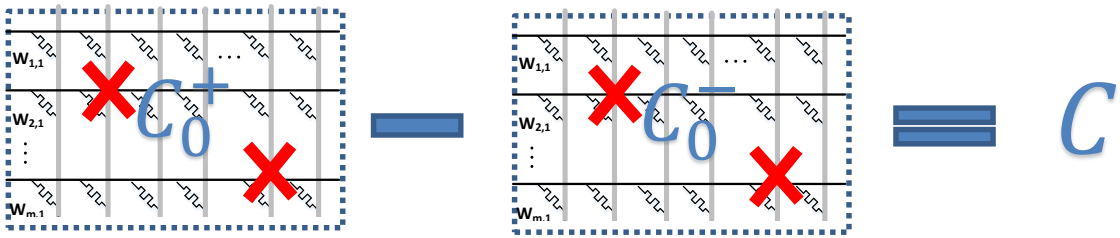
- Modified structure:



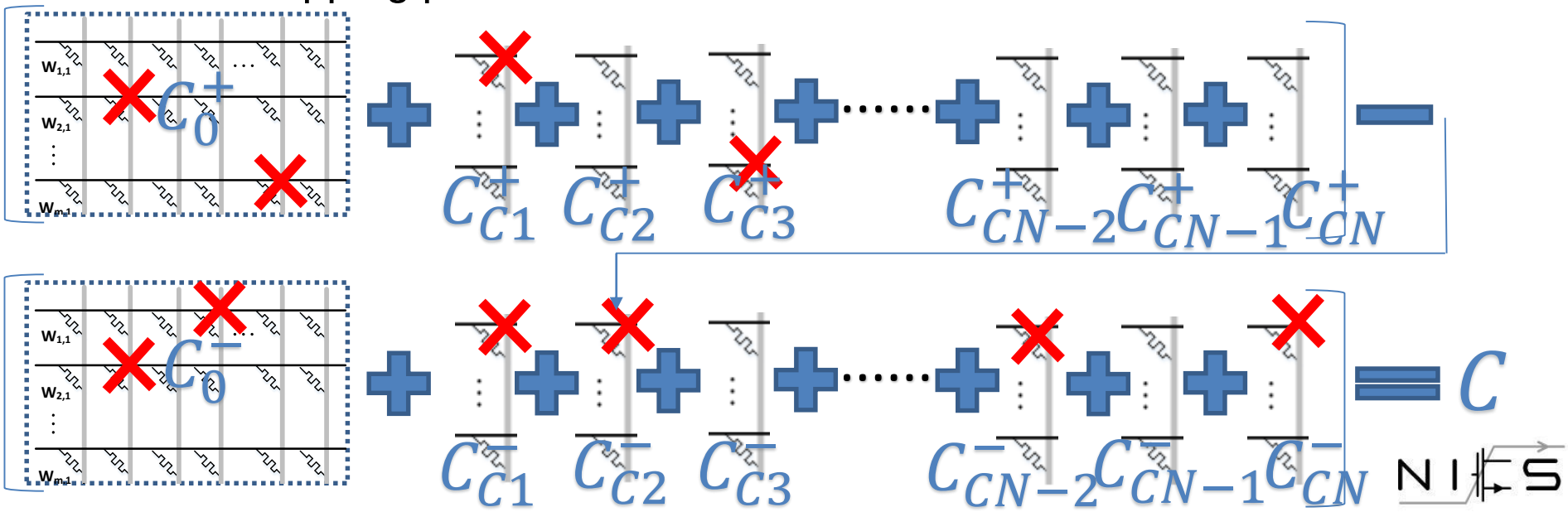
Independent Redundant Columns

Mapping algorithm: Co-mapping corresponding columns to get best available result

- Target application matrix: **C**
- Original Mapping process:



- Modified Mapping process:



Hardware Overhead

- The proposed redundant methods requires additional RRAM cells and corresponding function/control modules
- The detailed comparison between original method and proposed two methods is as follows:

TABLE III
HARDWARE OVERHEAD OF ORIGINAL RCS DESIGN AND TWO REDUNDANT SCHEMES WE PROPOSED

	Original	Redundant Xbar	Redundant Columns
RRAM	$2 \times M \times N$	$2 \times (R+1) \times M \times N$	$2 \times M \times N + 2 \times 2 \times \lceil R \times P \rceil \times M \times N$
ADC	$2 \times N$	$2 \times (R+1) \times N$	N
DAC	M	M	M
Op Amps	$2 \times N$	$2 \times (R+1) \times N$	$2 \times 2 \times N$
Decoder	$2 \times M$ outputs + $2 \times N$ outputs	$2 \times (R+1) \times M$ outputs + $2 \times (R+1) \times N$ outputs	$2 \times M$ outputs + $2 \times N$ outputs + $2 \times (2 \times \lceil R \times P \rceil \times M$ outputs)
Mux	0	0	$2 \times 2 \times \lceil R \times P \rceil \times M \times N \times (\lceil \frac{1}{p} \rceil \times$ inputs
Adder	0	$R \times N$	$2 \times N$
Sub	$2 \times N$	$2 \times N$	$2 \times N$

Experimental Results

Three case studies

1. Mapping an 128x128 matrix onto RRAM-based computing systems
2. 128x128 Matrix-vector multiplication on RRAM-based computing system
3. 784x100x10 NN for MNIST on RRAM-based computing system;

Three evaluation metrics

1. Mapping Error: l_2 norm distance between ideal matrix and mapped matrix
2. Computing Error: l_2 norm distance between ideal output vector and RRAM computed output vector
3. MNIST Error: error rate of MNIST classification accuracy. The error rate of MNIST on CPU platform (ideal case) is 2.17%

Experimental Results

- Performance of Mapping Algorithm
  - Great improvement for all three evaluation metrics under different faults
  - Especially effective for testing cases with low (below 5%) faults
  - For 5% SAFs, MAO reduce MNIST error from 52.11% to 4.01%
  - No extra overhead is introduced by MAO

SAFs (%)	Mapping Error (%)			Computational Error (%)			MNIST Error (%)		
	Origin [10]	MAO	Percentage Improvement (%)	Origin [10]	MAO	Percentage Improvement (%)	Origin [10]	MAO	Percentage Improvement (%)
1	16.60	10.10	6.50	16.56	10.14	6.42	12.42	2.24	10.18
3	28.82	17.71	11.11	28.93	17.53	11.40	34.60	2.89	31.71
5	37.04	23.11	13.93	36.80	23.11	13.69	52.11	4.01	48.10
7	43.89	27.70	16.19	43.40	27.21	16.19	62.31	7.55	54.76
8	46.83	29.88	16.95	46.88	29.71	17.17	66.52	10.99	55.53
10	52.83	34.81	18.02	52.39	34.88	17.51	74.17	15.38	58.79
15	63.86	42.80	21.06	63.89	42.60	21.29	79.67	41.57	38.10
20	73.72	53.15	20.57	73.70	53.31	20.39	82.25	58.19	24.06

Experimental Results

- Performance and overhead under some typical cases

TABLE IV
PERFORMANCE AND ENERGY OVERHEAD OF RX AND IRC ON MNIST
SIMULATE 100 TIMES WITH RANDOM SAFS

SAFs (%)	Design of RCS	R	Range of MNIST Error (%)	Average MNIST Error (%)	Average Percentage Improvement (%)	Energy Overhead (%)
5	Original	0	[39.56, 73.42]	52.11	-	-
	MAO	0	[2.88, 9.16]	4.01	48.10	-
	RX	1	[2.13, 6.75]	2.22	49.89	37.58
	RX	2	[2.13, 2.27]	2.18	49.93	75.17
	IRC	1	[2.46, 4.07]	3.32	48.79	20.85
	IRC	2	[2.14, 2.85]	2.32	49.79	23.55
10	Original	0	[61.44, 84.47]	74.17	-	-
	MAO	0	[8.53, 36.15]	15.38	58.79	-
	RX	1	[2.37, 7.85]	2.83	71.34	37.58
	RX	2	[2.15, 7.02]	2.70	71.47	75.17
	IRC	1	[4.97, 26.06]	11.44	60.03	22.68
	IRC	2	[2.25, 9.87]	3.87	70.30	27.22
20	Original	0	[71.99, 89.38]	82.25	-	-
	MAO	0	[49.82, 70.46]	58.19	24.06	-
	RX	1	[10.84, 37.8]	19.30	57.50	37.58
	RX	2	[3.33, 13.37]	5.91	77.95	75.17
	RX	3	[2.38, 6.46]	2.65	79.60	112.75
	IRC	1	[35.96, 69.55]	54.21	28.04	26.46
	IRC	2	[6.00, 29.82]	13.52	68.73	34.78
	IRC	3	[2.36, 4.00]	3.65	78.60	43.09

Experimental Results

- Energy-Performance Trade-Off (Task: MNIST)

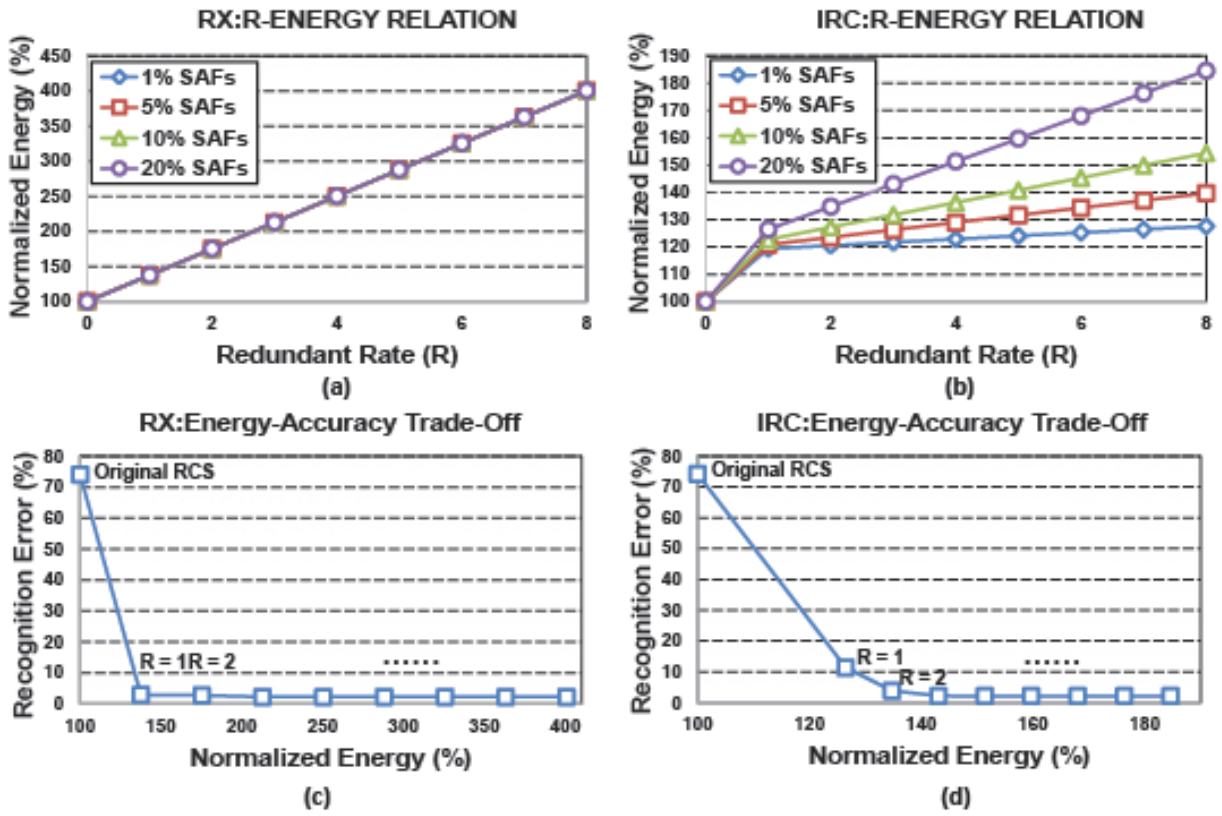


Fig. 4. (a).R-Energy relation of RX. (b).R-Energy relation of IRC. (c).Energy-Error trade-off of RX (10% SAFs). (d).Energy-Error trade-off of IRC (10% SAFs).

References

[1] Prakash, Aravind, and S. Maikap. "Improved resistance memory characteristics and switching mechanism using TiN electrode on TaOx/W structure." *Proc. Winter Simul. Conf*. 2013.

[2] Chen, C.; Shih, H.; Wu, C.; Lin, C.; Chiu, P.; Sheu, S.; Chen, F.T., "RRAM Defect Modeling and Failure Analysis Based on March Test and a Novel Squeeze-Search Scheme," *Computers, IEEE Transactions on* , vol.64, no.1, pp.1,11, Jan. 1 2015

[3] Lee D, Park J, Moon K, et al. Oxide based nanoscale analog synapse device for neural signal recognition system[C]//2015 IEEE International Electron Devices Meeting (IEDM). IEEE, 2015: 4.7. 1-4.7. 4.

[4] Prezioso M, Merrikh-Bayat F, Hoskins B D, et al. Training and operation of an integrated neuromorphic network based on metal-oxide memristors[J]. Nature, 2015, 521(7550): 61-64.

[5] Wang, Jue, Xiangyu Dong, and Yuan Xie. "Point and discard: a hard-error-tolerant architecture for non-volatile last level caches." *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012.

[6] Cheng-Kok Koh; Weng-Fai Wong; Yiran Chen; Hai Li, "The salvage cache: A fault-tolerant cache architecture for next-generation memory technologies," *Computer Design, 2009. ICCD 2009. IEEE International Conference on* , vol., no., pp.268,274, 4-7 Oct. 2009

[7] Hasan R, Taha T M. Enabling back propagation training of memristor crossbar neuromorphic processors[C]//2014 International Joint Conference on Neural Networks (IJCNN). IEEE, 2014: 21-28.

[8] Degraeve R, Fantini A, Raghavan N, et al. Causes and consequences of the stochastic aspect of filamentary RRAM[J]. Microelectronic Engineering, 2015, 147: 171-175.

[9] Shih, Hsiu-Chuan, et al. "Training-based forming process for RRAM yield improvement." *VLSI Test Symposium (VTS), 2011 IEEE 29th*. IEEE, 2011.

[10] Chen Y Y, Roelofs R, Redolfi A, et al. Tailoring switching and endurance/retention reliability characteristics of HfO 2/Hf RRAM with Ti, Al, Si dopants[C]//VLSI Technology (VLSI-Technology): Digest of Technical Papers, 2014 Symposium on. IEEE, 2014: 1-2.

[11] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars[C]//Proc. ISCA. 2016.

[12] Liu X, Mao M, Liu B, et al. Harmonica: A framework of heterogeneous computing systems with memristor-based neuromorphic computing accelerators[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2016, 63(5): 617-628.

[13] Xia L, Li B, Tang T, et al. MNSIM: Simulation platform for memristor-based neuromorphic computing system[C]//2016 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2016: 469-474.

[14] Gu P, Li B, Tang T, et al. Technological exploration of rram crossbar array for matrix-vector multiplication[C]//The 20th Asia and South Pacific Design Automation Conference. IEEE, 2015: 106-111.

[15] Tang T, Xia L, Li B, et al. Spiking neural network with rram: Can we use it for real-world application?[C]//Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium, 2015: 860-865.

[16] Chi P, Li S, Qi Z, et al. PRIME: A Novel Processing-In-Memory Architecture for Neural Network Computation in ReRAM-based Main Memory[C]//Proceedings of ISCA. 2016, 43.

Thank you!

Q&A