Computation-Oriented Fault-Tolerance Schemes for RRAM Computing Systems

Wenqin Huangfu[1], Lixue Xia[1], Ming Cheng[1], Xiling 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, USA
[3]Department of Electrical and Computer Engineering, University of California, Santa Barbara, USA
[1]e-mail: yu-wang@mails.tsinghua.edu.cn

Abstract—**The emerging metal-oxide resistive switching random-access memory (RRAM) devices and RRAM crossbar arrays have demonstrated their potential in enormously boosting the speed and energy-efficiency of analog matrix-vector multiplication. Unfortunately, due to the immature fabrication technology, commonly occurring Stuck-At-Faults (SAFs) seriously degrade the computational accuracy of RRAM crossbar based Computing System (RCS). In this paper, we propose a Mapping Algorithm with inner fault-tOlerant ability (MAO) to convert matrix parameters into RRAM conductances in RCS by providing larger mapping space and fully exploring the available mapping space. Furthermore, we present two computation-oriented redundancy schemes - 'Redundant Crossbars' (RX) and 'Independent Redundant Columns' (IRC) to alleviate the loss of computational accuracy due to SAFs. RX adds redundant RRAM crossbar arrays and IRC introduces independent redundant RRAM columns to compensate the computational errors brought by SAFs.**

I. Introduction

With the coming of 'Big Data' era, traditional CMOS-based computing platforms, such as CPU, GPU and FPGA, cannot meet the requirements of processing data in real-time [1], [2]. For example, the highest power-efficiency of CMOS-based systems is only 10 GFLOPS/W now [3]. The emerging metal-oxide resistive switching random-access memory (RRAM) has demonstrated tremendous potential in significantly boosting the speed and energy efficiency of next-generation computing systems. For example, a low-power approximate computing system based on RRAM has demonstrated the energy-efficiency of 400 GFLOPS/W [4].

Unfortunately, due to the limitation of the immature fabrication technology, manufacturing yield is still a major concern in a RRAM crossbar based Computing System (RCS). Many types of faults exist in RCS and they can be divided into two categories, i.e. hard faults, like Stuck-At-Faults (SAFs), and soft faults, like Read-One-Disturb (R1D) and Read-Zero-Disturb (R0D). Researchers have recently paid attention to soft faults [5], but there has been no attempts to address hard faults that affect computation.

Hard faults, especially SAFs - the phenomenon of which is that faulty devices get stuck at high-resistance state (HRS) or low-resistance state (LRS) and cannot change their resistance states, are common in RCS. For example, researchers reported that only 63% RRAM devices are fault-free for a fabricated 4-Mb HfO_2-based RRAM test chip and SAFs appear rather frequently - about 10% RRAM devices in the chip contain SAFs [6]. However, all existing RRAM-related redundancy schemes focus exclusively on memory applications [7]. Because the basic storage unit of RRAM-based memory is a single RRAM device, fault-free RRAM devices in memory-oriented redundant structures, a few redundant rows or columns, are sufficient to replace faulty RRAM devices. Note that the basic computational unit of RCS is a whole RRAM column, consider a 512 × 512 RRAM crossbar array, even if the yield of a single RRAM device is 0.99, there is only a 0.0058 probability that a randomly chosen column is fault-free. Even when we use memory-oriented redundancy schemes for fault tolerance, redundant columns and rows can contain faulty devices, thereby rendering them ineffective as fault-free replacements. With these differences, memory-oriented redundancy schemes for RRAM cannot be used in RCS. An even more serious concern is that we need to get access to all RRAM devices during computation, thus we cannot shut down the faulty ones during computation. So, errors introduced by SAFs will inevitably be included in the final output. Therefore, computation-oriented fault-tolerance schemes are indispensable for RCS.

Previously proposed mapping algorithms, which convert matrix parameters into RRAM conductances of RCS, use simplified assumptions and target at ideal RRAM devices without SAFs [8] [9] [10]. In this paper, for low (below 5%) defect rate (Percentage of faulty devices among all fabricated devices), we present an alternative mapping algorithm to provide larger mapping space for RCS. Because the overhead of RRAM crossbar arrays in RCS is low [11], we further use extra RRAM crossbar arrays or parts of RRAM crossbar arrays as novel redundant structures to design computation-oriented redundancy schemes to tolerate high (above 5%) defect rate.

The main contributions of this paper include:

1) We provide the first case study using the Mixed National Institute of Standards and Technology (MNIST) dataset [12] to demonstrate that SAFs have a significant adverse impact on RCS. Experimental results show that 10% SAFs, reported for RRAM test chip, will degrade the recognition accuracy of the MNIST dataset from 97.86% to 27.65%.
2) To reduce the drop in computational accuracy due to SAFs without incurring additional overhead, we propose a general Mapping Algorithm with inner fault-tOlerant ability (MAO), which enlarges the available mapping space for RCS. Experimental results show that with 5% SAFs, compared with previous mapping algorithm [10], MAO can improve the recognition accuracy of the MNIST dataset from 47.89% to 95.99%.
3) To ensure that RCS is effective for high defect rate, we propose two computation-oriented redundancy schemes, namely 'Redundant Crossbars' (RX) and 'Independent Redundant

This work was supported by 973 Project 2013CB329000, National Natural Science Foundation of China (No.61622403, 61373026, 61261160501), Brain Inspired Computing Research, Tsinghua University and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award number DE-SC0013553 with disclaimer at http://seal.ece.ucsb.edu/doe/. It was also supported in part by U.S. NSF (1500848 and 1533933), and a grant from Qualcomm.

978-1-5090-1558-0/17/$31.00 ©2017 IEEE

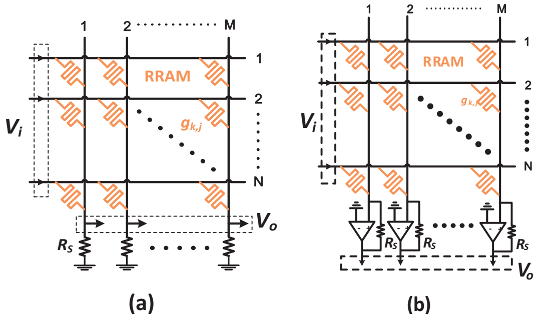

Fig. 1. Two typical sensing modes of RRAM crossbar array. (a).Using sensing resistors. (b).Using TIA.

Columns' (IRC), for RCS. We define redundancy ratio R for RX, IRC, and analyze the impact R has on the performance and energy overhead of RCS. With 10% SAFs, reported for RRAM test chip, RX can improve the recognition accuracy of the MNIST from 25.83% to 97.17% with energy overhead of 37.58% (R = 1) and IRC can improve that from 25.83% to 96.13% with energy overhead of 27.22% (R = 2) .

The rest of this paper is organized as follows: Section II provides the background knowledge of RRAM devices and the sensing mode we choose. Motivation to design computation-oriented fault-tolerance schemes is introduced in section III. Section IV presents MAO, RX and IRC. In Section V, experimental results and detailed analysis are provided. Finally, Section VI concludes this work.

II. Preliminaries

A. RRAM Devices and RRAM Crossbar Array

RRAM is an emerging, passive two-ports metal oxide device based on metal oxide materials such as HfO$_x$, TiO$_x$ and WO$_x$ [13]. RRAM devices can be used to build RRAM crossbar arrays, which can be used to perform matrix-vector multiplication very efficiently by merging storage and computation together, reducing the computation complexity from $O(n^2)$ to $O(1)$ [4], [14]. Due to the conductances of sensing resistors and RRAM devices can only be positive, two RRAM crossbar arrays are needed to represent an application matrix with both positive and negative parameters.

B. Choice of Sensing Mode

There are two widely used sensing modes for the output voltages:

1) R_s as the sensing component, as shown in Fig. 1(a).
2) Trans-impedance amplifier (TIA) as the sensing component, as Fig.1 (b) shows.

The mapping relations between matrix parameters and RRAM conductances can be represented as follows:

$$c_{i,j} = \begin{cases} \dfrac{g_{i,j}}{g_s + \sum_{k=1}^{N} g_{i,k}} & (R_s) \\ -\dfrac{g_{i,j}}{g_s} & (TIA) \end{cases} \tag{1}$$

where $c_{i,j}$ is the matrix parameter, i (i = 1,2,..,M) and j (j = 1,2,..,N) are the index numbers of input and output ports, $g_{i,k}$, $g_{i,j}$ and g_s are the conductances of RRAM devices and sensing resistors.

We choose the second sensing mode due to the following reasons:

1) Enable one-to-one mapping: With TIAs, $c_{i,j}$ only depends on $g_{i,j}$ and g_s. While with R_s, $c_{i,j}$ not only depends on $g_{i,j}$ and g_s, but also depends on $g_{i,k}$.

TABLE I
Performance Degradation due to SAFs for Three Experiments

SAFs (%)	0	1	5	10	20
Mapping Error (%)	0.21	16.60	37.04	52.55	73.72
Computational Error (%)	0.21	16.56	36.80	52.62	73.70
MNIST Error (%)	2.14	12.42	52.11	72.35	82.25

2) Isolate different layers of RCS: Without voltage followers or buffer amplifiers to isolate different circuit stages and guarantee the driving force, the parameters of neighboring layers in RCS will influence each other through R_s [15], [16].

III. Motivation

A. Fault Injection Experiment

To evaluate the performance degradation of RCS caused by SAFs, we conduct three experiments:

1) We use normalized mapping error to qualify the adverse effect SAFs have on the mapping results of RCS.
2) We use normalized computational error to qualify the negative effect SAFs have on the computational results of RCS.
3) We used the MNIST dataset as a case study to demonstrate the adverse impact that SAFs on a real-world application.

The evaluation metrics, i.e. mapping error (relative 2-norm distance between the target matrix and the mapped matrix), computational error (relative 2-norm distance between the idealized output and the output of configured RCS), recognition accuracy of the MNIST dataset and experimental setup are in Section V.A.

From Table.I, we can observe that, with 1% SAFs, the normalized mapping error and normalized computational error increase from about 0.20% to about 16.60%, the recognition accuracy of the MNIST drops from 97.83% to 87.58% and, with 20% SAFs, the normalized mapping error and normalized computational error increase to about 73.70% and the recognition accuracy of the MNIST drops to 17.75%.

Obviously, performance of the original design of RCS is totally unacceptable, even only with a small amount of SAFs. Considering that RCS has a great advantage of energy-efficiency over CMOS-based computing systems (tens and hundreds of times), it's worth sacrificing acceptable amount of energy (37.58% for RX and 27.22% for IRC in our experiments) to design redundancy schemes with the ability of tolerating SAFs in RCS.

B. Design Challenges

Assume that the yield of a single RRAM device is P_{FF} and the yield of different RRAM devices are independent, and the size of RRAM crossbar array is $M \times N$. Then the probability that a specific column of RRAM devices is fault-free is:

$$P_{Fault\ free} = P_{FF}^{N} \tag{2}$$

According to above equation, even if the yield of a single RRAM device is 0.99, for an RRAM crossbar array of size 512 × 512, there is only a 0.0058 probability that any given column is fault-free. Hence it is unlikely that there are many fault-free computational units in RCS, including the redundant computational units. As a result, traditional memory-oriented redundancy schemes are ineffective for RCS. A more serious concern is that it is not feasible to avoid SAFs by shutting down faulty devices and still achieve the desired functionality. Because even in an 1T1R (One-Transistor One-Resistor) structure, an unacceptable overhead of $M \times N$ control lines is needed to control every transistor independently. Thus errors caused by SAFs inevitably contribute to the final output of RCS.

IV. Mapping Algorithm and Redundancy Schemes

We propose a mapping algorithm with inner fault-tolerance ability and no extra overhead, i.e. MAO, in Section.A and two redundancy schemes - ‘Redundant Crossbars’ (RX) and ‘Independent Redundant Columns’ (IRC) with stronger fault-tolerance ability, in Section.B and Section.C. The overview of using RCS with RX and RCS with IRC to implement multi-layers Neural Networks (NNs) is shown in Fig.2.

A. Mapping Algorithms with inner fault-tOlerance ability (MAO)

In the original design of RCS, recall that matrix is realized by one positive RRAM crossbar array and one negative RRAM crossbar array. The relation between the input and output voltages in this Pos-Neg RRAM crossbar arrays system can be expressed as:

$$\begin{aligned}\vec{V_{out}} &= C^{+} \cdot \vec{V_{in}} + C^{-} \cdot -\vec{V_{in}} \\ &= (C^{+} - C^{-}) \cdot \vec{V_{in}} \\ &= C \cdot \vec{V_{in}} \end{aligned} \tag{3}$$

where C^{+} and C^{-} are the matrices represented by the positive and negative RRAM crossbar arrays, and $\vec{V_{in}}$ and $\vec{V_{out}}$ are the input and output voltages.

So, how to divide matrix parameters into the positive and negative RRAM crossbar arrays is a critical issue in mapping algorithm.

To simplify the division of matrix parameters in two RRAM crossbar arrays, original mapping algorithms usually allocate the positive and negative parameters in the target matrix into the positive and negative RRAM crossbar arrays correspondingly [10].

However, if we consider the physical limitations and SAFs in real-world RRAM devices, this simple method of dividing matrix parameters into RRAM crossbar arrays can result in errors:

1) The resistances of real-world RRAM devices cannot reach infinity, so zeros in divided application matrices cannot be reached in RRAM crossbar arrays, which causes error.
2) If an RRAM device contain SAFs, it is highly likely that this faulty device will lead to error, even though the error may be avoided by adjusting the other RRAM device on corresponding position of the other RRAM crossbar array.

So, to overcome the drawbacks of original mapping algorithm, we design MAO to fully explore the available mapping space.

As shown in Algorithm 1, for each parameter in the target matrix, MAO will first find all RRAM devices that meant to form this target parameter (Line 1 to Line 2). Then, MAO will initiate all fault-free devices, among the devices found above, to default values, while leaving other devices with SAFs alone (Line 3 to Line 10). Next, for each parameter in the target matrix, MAO will adjust all operable devices that meant to form this target parameter one-by-one, until the combination value of all devices cannot be closer to the target matrix parameter (Line 11 to Line 19). So MAO can always get the best available mapping results.

Algorithm 1: Mapping Algorithm with inner fault-tOlerance ability (MAO)

```
   Input: $C$, $G_{max}$, $G_{min}$ $g_s$, $StuckAtMap$, $Available\_Devices$
   Output: $G^{+}_{Output}$, $G^{-}_{Output}$
1  for $i = 1 : C.parameter\_number$ do
2      for $j = 1 : Available\_Devices(i)$ do
3          if $G^{+}_{Output}(i,j)$ doesn't get stuck then
4              $G^{+}_{Output}(i,j) = Gmax$
5          end
6          if $G^{+}_{Output}(i,j)$ doesn't get stuck then
7              $G^{-}_{Output}(i,j) = Gmin$
8          end
9      end
10 end
11 for $i = 1 : C.parameter\_number$ do
12     for $j = 1 : Available\_Devices(i)$ do
13         if Both $G^{+}_{Output}(i,j)$ and $G^{-}_{Output}(i,:)$ get stuck then
14             continue;
15         end
16         $Diff$ = $||C(i) - (sum(G^{+}_{Output}(i,:))/Gs$
             $-sum(G^{-}_{Output}(i,:))/Gs)||$;
17         First adjust $G^{+}_{Output}(i,j)$ and then adjust $G^{-}_{Output}(i,j)$
             to minimize $Diff$;
18     end
19 end
20 return $G^{+}_{Output}$, $G^{-}_{Output}$
```

B. Redundant Crossbars (RX)

1) Hardware Structure: RX uses redundant crossbar arrays, which have the exactly the same size as the original RRAM crossbar arrays and also contain faulty devices, as the redundant structure. Fig. 2 (d) demonstrates the hardware structure of RX.

2) Data Flow and Mapping Method: With RX, the input will first go through the shared DACs (Digital-Analog-Converter) for the reason that redundant RRAM crossbar arrays share the same inputs with two original RRAM crossbar arrays. Then, data go directly into the corresponding input ports of different RRAM crossbar arrays. The outputs of different RRAM crossbar arrays will first go through ADCs (Analog-Digital-Converter), then go through adders to derive the final output. Multiple RRAM crossbar arrays form the final matrix:

$$C = (C^{+} + \sum_{i=0}^{R} C_{i}^{+}) - (C^{-} + \sum_{i=0}^{R} C_{i}^{-}) \tag{4}$$

where C, C^{+} and C^{-} are the final matrics formed by original positive and negative RRAM crossbar arrays, i, C_{i}^{+} and C_{i}^{-} are the indexes, matrics formed by positive and negative redundant crossbars. R is defined as the ‘redundancy ratio’, it’s the number of extra RRAM crossbar arrays we use to form one positive or negative RRAM crossbar array, besides the original two RRAM crossbar arrays.

MAO is used in RCS with RX. The difference between using MAO in original RCS and RCS with RX is, in the original RCS, the $Available\_Devices$ for each matrix parameter in Algorithm 1 are two RRAM devices in the positive and negative RRAM crossbars. While, in RCS with RX, the $Available\_Devices$ are $2R$ RRAM devices (Some with SAFs) in $2R$ RRAM crossbars.

3) Hardware Overhead: Larger redundancy ratio R means stronger fault-tolerance ability, while increasing the hardware overhead at the same time. So trade-off between computational accuracy and energy consumption exists. Assume the size of target matrix is $M \times N$, the defect rate of a single RRAM device is P and the redundancy ratio is R. $[X, O]$ stands for Decoder with X outputs and $[Y, I]$ stands for Multiplexer (MUX) with Y inputs. The hardware overhead of RX is in Table.II. What needs to be mentioned is that as redundant RRAM crossbars can share DACs with the original RRAM crossbar arrays, there is no overhead of DACs.

The advantage of RX is that it maintains the RRAM crossbar array structure and no extra routing or control logics are needed.

C. Independent Redundant Columns (IRC)

1) Hardware Structure: IRC uses independent redundant columns, which usually have much fewer rows than that of the original RCS,

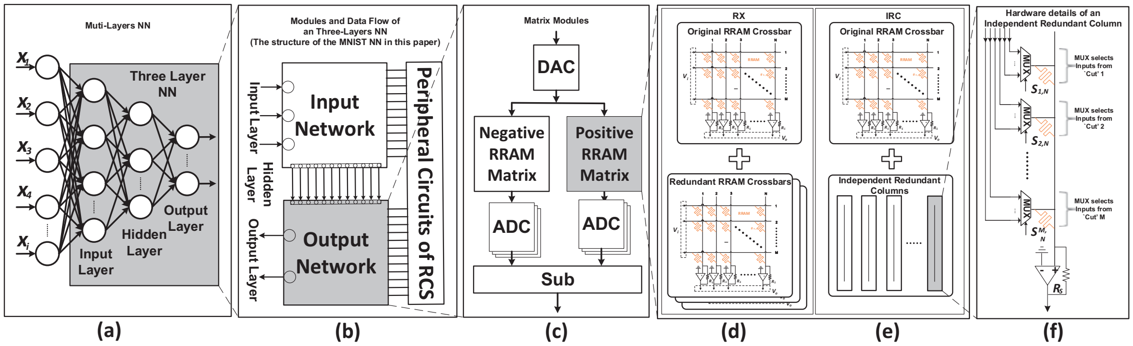

Fig. 2. Overview of using RCS with RX and RCS with IRC to implementing multi-layers NN. (a). Multi-layers NN. (b). Modules and data flow of an three-layers NN (The structure of the MNIST NN in this paper). (c). Modules to form a matrix in NN. (d). Hardware structure of RX. (e). Hardware structure of IRC. (f). Hardware details of an independent redundant column.

TABLE II
HARDWARE OVERHEAD OF THREE DESIGNS OF RCS
MATRIX SIZE: $M \times N$; DEFECT RATE: P; REDUNDANCY RATIO: R; $[X, O]$: DECODER WITH X OUTPUTS; $[X, I]$:MUX WITH X INPUTS;

	Original	RX	IRC
RRAM	$2MN$	$2(R+1)MN$	$2MN + 4\lceil RP \rceil MN$
ADC	$2N$	$2(R+1)N$	N
DAC	M	M	M
TIA	$2N$	$2(R+1)N$	$4N$
Decoder	$2 \times [M, O]$ + $2 \times [N, O]$	$2(R+1) \times [M, O]$ + $2(R+1) \times [N, O]$	$2 \times [M, O]$ + $2 \times [N, O]$ + $2 \times [2\lceil RP \rceil M, O]$
MUX	0	0	$4\lceil RP \rceil MN \times [\lceil \frac{1}{p} \rceil, I]$
Adder	0	RN	$2N$
Sub	$2N$	$2N$	$2N$

as the redundant structure. Assume the defect rate of a single device in RCS is P, the size of RRAM crossbar array is $M \times N$. Then the expectation of SAFs in each RRAM column is:

$$E(F) = P \times M = PM \tag{5}$$

Because of the uncertainty of SAFs (Even IRC contain SAFs), routing strategy and control logics are needed to select the inputs of different RRAM devices in redundant columns. To ease the hardware overhead of routing, we evenly divide RRAM devices in columns of original RRAM crossbar arrays into $E(F)$ parts, we call each part a 'Cut'. Thus the expectation of SAFs in each Cut is 1. Similar to RX, we define 'redundancy ratio' R to indicate the number of redundant devices in a redundant column for one Cut.

We find that two redundant devices in the positive and negative redundant columns can cancel the negative effect one faulty device have perfectly. For example, if an RRAM device in the negative RRAM crossbar array gets stuck at low resistance, to completely compensate the negative effect caused by this faulty device, we can first set its corresponding device on the positive RRAM crossbar array to low resistance, then add a pair of fault-free RRAM devices to make the mapping space the same as ideal. Considering that both the positive and negative RRAM crossbar arrays may contain SAFs, we should double the number of redundant devices. So the number of redundant devices in each independent redundant column should be:

$$M_{Redundancy} = 2 \times R \times E(F) = 2 \times R \times P \times M = 2RPM \tag{6}$$

In a word, IRC contains N independent redundant RRAM columns, each with $M_{Redundancy}$ RRAM devices. The expectation of SAFs in each Cut is 1. SAFs in each Cut is dealt with $2R$ redundant devices (some may with SAFs). The inputs of redundant devices are selected by MUXs (Controlled by signal $s_{i,j}$). The inputs number of MUXs are $\lceil \frac{1}{p} \rceil$ (number of devices each Cut). Fig. 2 (e) demonstrates the hardware structure of IRC and Fig. 2 (f) contains details of a single independent redundant column.

2) Data Flow and Mapping Method: With IRC, the input will first go through the shared DAC, then it will go into different rows in the original RRAM crossbar array and corresponding redundant devices in the independent redundant columns. The output of different columns in the original RRAM crossbar arrays will merge with output from corresponding independent redundant columns to get the final output:

$$C_{i,j} = \begin{cases} C_{i,j}^{+} - C_{i,j}^{-} & (FF) \\ (C_{i,j}^{+} + \sum_{k=1}^{A^{+}} V_{i,j,k}^{+}) - (C_{i,j}^{-} + \sum_{k=1}^{A^{-}} V_{i,j,k}^{-}) & (SAFs) \end{cases} \tag{7}$$

where i (i = 1,2,..,M) and j (j = 1,2,..,N) are the index numbers of the input and output ports, $C_{i,j}$, $C_{i,j}^{+}$ and $C_{i,j}^{-}$ are the final matrix parameter and matrix parameters formed by RRAM devices in the positive and negative RRAM crossbar arrays, k is a counter index, A^{+} and A^{-} are numbers of available redundant devices in the corresponding positive and negative independent redundant columns, $V_{i,j,k}^{+}$ and $V_{i,j,k}^{-}$ are the compensatory parameters formed by RRAM devices in the positive and negative independent redundant columns.

MAO is used in RCS with IRC. In RCS with IRC, the $Available\_Devices$ in Algorithm 1 for each matrix parameter are either two RRAM devices in positive and negative RRAM crossbar arrays or multiple RRAM devices in both the original RRAM crossbar arrays and corresponding independent redundant columns.

3) Hardware Overhead: Similar to RX, determined by redundancy ratio R, trade-off between computational accuracy and energy consumption exists. The hardware overhead of IRC is in Table.II. Parameters in Table.II have the same meanings as indicated above. And, for the same reason, IRC doesn't need extra DACs, too.

The advantage of IRC is that it needs fewer ADCs/DACs and RRAM devices than RX.

TABLE III
ERROR DUE TO PREVIOUSLY PROPOSED MAPPING ALGORITHM AND MAO FOR THREE TASKS (IDEALIZED MNIST ERROR: 2.17%)

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

V. EXPERIMENTAL RESULTS

A. Evaluation Metrics

Three evaluation metrics and reasons why we choose these evaluation metrics are as follows:

1) Mapping Error: To see the negative effects SAFs have on the mapping results, mapping error is defined as the relative 2-norm distance between the target matrix and the mapped matrix:

$$Error_{Mapping} = \frac{||C_{Mapped} - C_{Idealized}||_2}{||C_{Idealized}||_2} \times 100\% \quad (8)$$

where $C_{Idealized}$ is the target matrix and C_{Mapped} is the mapped matrix.

2) Computing Error: To see the negative effects SAFs have on the results of matrix-vector multiplication, computational error is defined as the relative 2-norm distance between the idealized output and the output of configured RCS:

$$Error_{Computational} = \frac{||R_{Mapped} - R_{Idealized}||_2}{||R_{Idealized}||_2} \times 100\% \quad (9)$$

where $R_{Idealized}$ is the idealized results and R_{Mapped} is the output of configured RCS.

3) MNIST Recognition Error: To see the negative effects SAFs have on real-world applications and to evaluate the performance of MAO, RX and IRC on real-world application, we use the well-known MNIST digital recognition dataset as a case study.

B. Experimental Setup

The RRAM devices we use have a precision of 8-bit, which is important to guarantee good performance for many NN applications, a high resistance of 1MΩ and a low resistance of 1kΩ. The yield of a single RRAM device is set to 0.99, 0.95, 0.90 and 0.80. We assume the faulty devices randomly get stuck at their HRS or LRS.

In the first two experiments, the size of application matrix is set to 128×128. The matrix parameters are random values ranging from -1 to 1 and, due to the reason that input voltages of RCS should always be positive, the parameters of the input vector are random values ranging from 0 to 1. We run each simulation 100 times and calculate the average error rate as the final error rate.

In the third experiment, we use MNIST digital recognition dataset as a case study. MNIST is a widely used dataset for optical character recognition with 60,000 handwritten digits in training set and 10,000 in testing set. In our experiment, we use all samples of handwritten digits to train the NN and use all testing samples for testing. The size of the NN we train is 784 × 100 × 10. Experimental results show that the recognition accuracy is 97.83% on the CPU platform and also 97.83% on the ideal RCS. Due to the randomness of SAFs in RCS, we run each simulation for 100 times and calculate the average recognition accuracy as the final recognition accuracy.

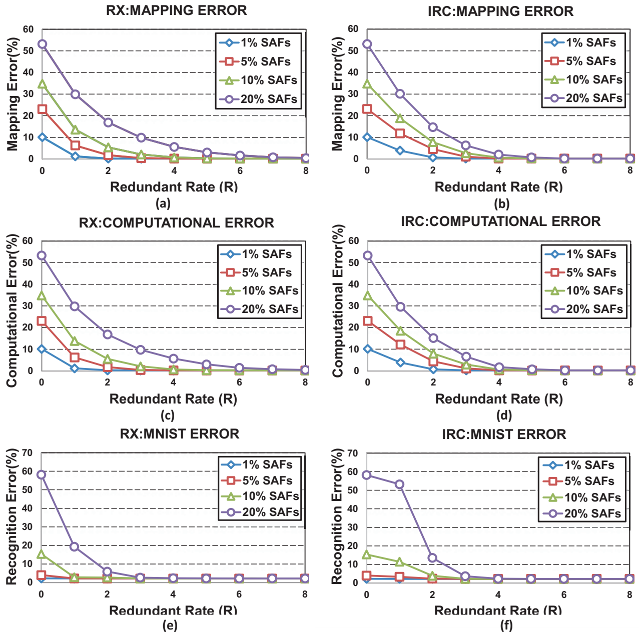

Fig. 3. The performance of RX and IRC with different defect rate of RRAM devices and different redundancy ratio. (a).Mapping error with RX. (b).Mapping error with IRC. (c).Computational error with RX. (d).Computational error with IRC. (e).Recognition error of MNIST dataset with RX. (f).Recognition error of MNIST dataset with IRC.

C. Performance of MAO, RX and IRC

1) Mapping Algorithm with inner fault-tOlerance ability (MAO): With SAFs, the performance of previous mapping algorithm and MAO is in Table.III. We can see that MAO shows very good performance under all tests. To be specific, with 1% and 5% SAFs, MAO improves the recognition accuracy of the MNIST from 87.58% and 47.89% to 97.76% and 95.99%, with no extra overhead.

2) Redundant Crossbars (RX): With SAFs and using MAO, the performance of RX is in Fig. 3 (a), (c), (e). We can see that, with RX, by increasing R, the performance of all three experiments can be improved to near-perfect. To be specific, with 20% SAFs, RX can improve the recognition accuracy of MNIST dataset from 17.75% to 97.35% with $R = 3$.

3) Independent Redundant Columns (IRC): With SAFs and using MAO, the performance of IRC is in Fig. 3 (b), (d), (f). We can see that, with IRC, by increasing R, the performance of all three experiments can be improved to near-perfect. To be specific, with

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

20% SAFs, IRC can improve the recognition accuracy of MNIST from 17.75% to 96.35% with $R = 3$.

4) Energy, Performance and Parameters Analysis: According to the hardware overhead in Table.II, we calculate the energy overhead of RCS with RX and RCS with IRC in Table. IV. We can see that RX and IRC can achieve huge improvement in the recognition accuracy of MNIST dataset on average at the price of acceptable energy overhead. By increasing R, we not only suppress the average MNIST error, but also narrow the range of MNIST error effectively.

R-Energy relations of RCS with RX and RCS with IRC are presented in Fig.4 (a), (b). With RX, the R-Energy relation is linear and defect rate independent. The energy overhead of RX will increase 37.58%, which is mainly caused by the presence of extra RRAM devices and ADCs, as R increases by one. With IRC, when R increases from 0 to 1, the energy overhead increases quickly (about 20%), mainly contributed from the presence of extra ADCs. Then, further increasing R only brings about extra RRAMs, the number of which is defect rate dependable as indicated in Equ. (6), the increasement of energy overhead ranges from 1.18% to 8.32%, with SAFs ranges from 1% to 20%.

In the MNIST application, the energy-accuracy trade-offs of RCS with RX and RCS with IRC are in Fig. 4 (c), (d). We can see that, using MAO as the mapping method, RX and IRC can suppress the recognition error from 74.17% to 2.83% and 3.87% with an energy overhead of 37.58% (R = 1) and 27.22% (R = 2). While further increasing R introduces more energy overhead and achieves little improvement in recognition accuracy.

VI. CONCLUSION

In this paper, we propose a mapping algorithm with inner fault-tolerance ability, i.e. MAO, and two computation-oriented redundancy schemes, i.e. RX and IRC, for RCS. MAO can reduce the mapping error and computational error for RCS with no extra overhead. In our case study, with 1% SAFs, MAO can improve the recognition accuracy of the MNIST dataset from 88.52% to 97.76%, which is only 0.07% lower than the result on CPU platform. RX and IRC demonstrate their excellent fault-tolerance ability. Compared with the original design of RCS, with 10% SAFs, RX can improve the recognition accuracy of the MNIST from 25.83% to 97.17% with energy overhead of 37.58% and IRC can improve the recognition accuracy of the MNIST from 25.83% to 96.13% with energy overhead of 27.22%.

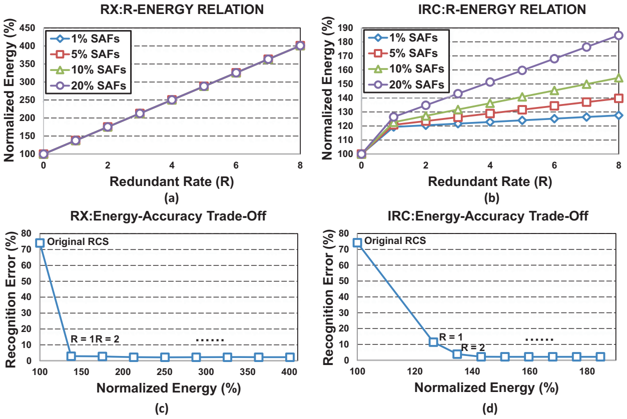

Fig. 4. (a).R-Energy relation of RX. (b).R-Energy relation of IRC. (c).Energy-Error trade-off of RX (10% SAFs). (d).Energy-Error trade-off of IRC (10% SAFs).

REFERENCES

[1] C. Guger, H. Ramoser *et al.*, "Real-time eeg analysis with subject-specific spatial patterns for a brain-computer interface (bci)," *IEEE Trans on Rehabilitation Engineering*, vol. 8, no. 4, pp. 447–456, 2000.

[2] G. A. Carpenter, S. Grossberg *et al.*, "Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural networks*, vol. 4, no. 5, pp. 565–588, 1991.

[3] P. A. Merolla, J. V. Arthur *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[4] B. Li, Y. Shan *et al.*, "Memristor-based approximated computation," in *ISLPED*. IEEE Press, 2013, pp. 242–247.

[5] B. Li, Y. Wang *et al.*, "Ice: inline calibration for memristor crossbar-based computing engine," in *DATE*. European Design and Automation Association, 2014, pp. 1–4.

[6] C.-Y. Chen, H.-C. Shih *et al.*, "Rram defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *TC*, vol. 64, no. 1, pp. 180–190, 2015.

[7] S. Kannan, N. Karimi *et al.*, "Detection, diagnosis, and repair of faults in memristor-based memories," in *VTS*. IEEE, 2014, pp. 1–6.

[8] M. S. Tarkov, "Mapping neural network computations onto memristor crossbar," in *SIBCON*. IEEE, 2015, pp. 1–4.

[9] M. Hu, H. Li *et al.*, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 10, pp. 1864–1878, 2014.

[10] P. Gu, B. Li *et al.*, "Technological exploration of rram crossbar array for matrix-vector multiplication," in *ASP-DAC*. IEEE, 2015, pp. 106–111.

[11] B. Li, L. Xia *et al.*, "Merging the interface: Power, area and accuracy co-optimization for rram crossbar-based mixed-signal computing system," in *DAC*. ACM, 2015, pp. 1–6.

[12] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[13] H.-S. P. Wong, H.-Y. Lee *et al.*, "Metal–oxide rram," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[14] B. Li, P. Gu *et al.*, "Rram-based analog approximate computing," *TCAD*, vol. 34, no. 12, pp. 1905–1917, 2015.

[15] S. O. Cannizzaro, A. D. Grasso *et al.*, "Design procedures for three-stage cmos otas with nested-miller compensation," *TCAS I*, vol. 54, no. 5, pp. 933–940, 2007.

[16] W. Oh and B. Bakkaloglu, "A cmos low-dropout regulator with current-mode feedback buffer amplifier," *TCAS II*, vol. 54, no. 10, pp. 922–926, 2007.