High throughput N-modular redundancy for error correction design of memristive stateful logic

Xi Zhu(朱熙), Hui Xu(徐晖), Weiping Yang(杨为平), Zhiwei Li(李智炜)[†],
Haijun Liu(刘海军)[‡], Sen Liu(刘森), Yinan Wang(王义楠), and Hongchang Long(龙泓昌)

College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, *China*

(Received 25 May 2022; revised manuscript received 29 August 2022; accepted manuscript online 5 September 2022)

Memristive stateful logic is one of the most promising candidates to implement an in-memory computing system that computes within the storage unit. It can eliminate the costs for the data movement in the traditional von Neumann system. However, the instability in the memristors is inevitable due to the limitation of the current fabrication technology, which incurs a great challenge for the reliability of the memristive stateful logic. In this paper, the implication of device instability on the reliability of the logic event is simulated. The mathematical relationship between logic reliability and redundancy has been deduced. By combining the mathematical relationship with the vector-matrix multiplication in a memristive crossbar array, the logic error correction scheme with high throughput has been proposed. Moreover, a universal design paradigm has been put forward for complex logic. And the circuit schematic and the flow of the scheme have been raised. Finally, a 1-bit full adder (FA) based on the NOR logic and NOT logic is simulated and the mathematical evaluation is performed. It demonstrates the scheme can improve the reliability of the logic significantly. And compared with other four error corrections, the scheme which can be suitable for all kinds of R–R logics and V–R logics has the best universality and throughput. Compared with the other two approaches which also need additional complementary metal–oxide semiconductor (CMOS) circuits, it needs fewer transistors and cycles for the error correction.

Keywords: memristor, stateful logic, logic reliability, in-memory computing

PACS: 85.35.–p, 85.25.Hv, 87.85.Qr, 84.32.–y **DOI:** 10.1088/1674-1056/ac8f38

1. Introduction

Traditional computation system based on the von Neumann architecture separates the computation from storage. Data stored in the storage unit need moving to the processor unit to accomplish the computation tasks. It brings significant costs in time and energy.[1] With the advent of artificial intelligence (AI) and the internet of things (IoT), it is harder for the traditional computation system to meet the requirements of low latency and low energy costs. An in-memory computing system that computes within the storage unit is a promising approach to overcome the challenge.[2]

Memristive stateful logic[3–8] is one of the most promising candidates to realize the in-memory computing. Memristor is a two-terminal nonvolatile electronic device.[9–12] Its resistance state can be switched between the low resistance state (LRS) and the high resistance state (HRS). The switching rule depends on the magnitude and the polarity of the applied voltage on it. The memristor in the memristive stateful logic can work as the computation unit and the storage unit within the integrated crossbar array,[13,14] which can remove the costs for the data movement.

However, current fabrication technology cannot eliminate the randomness of conductive filament (CF) formation/dissolution in the cation-based memristor, which induces that randomness including resistance variations and voltage variations in the SET/RESET operation for the memristor exits inevitably.[15] In the resistance variations, LRS and HRS variation rates σ/μ can reach 0.01 and 0.07, respectively.[16] And in the voltage variations, set and reset voltage variations σ/μ can reach 0.034 and 0.027, respectively.[17] These variations may bring the memristive stateful logic with the reliability problems.[18–22] The authors in Refs. [20,21] proposed multiple operations to improve the reliability. But the reliability of the logic may decrease when the device characteristics and the operation times cannot meet the specific conditions. Reference [22] proposed a universal logic error scheme, which requires customization for specific error control including unintended nonswitching of the output cell, unintended switching of the output cell, and unintended switching of the input cell. Besides, when the target logic operation is complex, a balanced gate needs to be designed carefully for the optimal time, area, and energy. Reference [18] proposed an error monitor scheme for an IMPLY-based memristive in-memory computing system. But it can only monitor and correct the specific stateful logic family of IMPLY logic. Reference [19] proposed a scheme which corrects the logic error with the additional memristive logic operations. But the reliability of the additional memristive logic must be credible to improve the reliability of the expected logic.

Therefore, we propose an n-modular redundancy logic error correction scheme with the best universality and throughput. And in comparison with the other two approaches which

[†]Corresponding author. E-mail: lizhiwei@nudt.edu.cn
[‡]Corresponding author. E-mail: liuhaijun@nudt.edu.cn

© 2023 Chinese Physical Society and IOP Publishing Ltd

also need additional CMOS circuits, it needs fewer transistors and cycles for the error correction. Moreover, synthesis and in-memory mapping of logic execution in a single row (SIMPLER)[23] can map a given logical function into a single column (or row) with maximum cell reuse automatically, which can be combined with the full parallel stateful logic computation to realize the scheme design from logical primitives to logical functions.

This paper extends our work in the conference paper,[24] which only proposes the basic n-modular redundancy error correction design with preliminary mathematic simulation results. The major contributions of this paper include:

(i) The reliability of the logic event and the logic function is analyzed, and the implication of the device instability on the logic reliability is discussed. The principle of n-modular redundancy is systematically analyzed. The design of the high throughput n-modular redundancy error correction scheme with a universal design paradigm is proposed.

(ii) An n-modular redundancy error correction circuit is put forward to improve the reliability of the stateful logic. The operational process is presented. And the circuit is simulated with Cadence Virtuoso to demonstrate its function.

(iii) Based on the properties of the tested devices, a case study of the scheme is carried out with HSPICE to evaluate the performance of the 1-bit full adder (FA) based on the stateful neural network.

In this paper, Section 2 presents the full parallel stateful logic based on the stateful neural network in the memristive crossbar array and the definition of the throughput. The reliability of the logic event and the logic function is analyzed, and the implication of the device instability on the logic reliability is discussed in Section 3. The principle and the design of the n-modular redundancy error correction scheme are proposed in Section 4. The 1-bit FA based on the NOR logic and the NOT logic is demonstrated for an example and the analysis on the scheme is presented in Section 5. This paper is concluded in Section 6.

2. Full parallel stateful neural network

Figure 1(a) shows an $m \times n$ memristive crossbar array, where m WLs and n BLs are connected with the synaptic weights. The configurations of the synaptic weights include one-transistor-one-resistor (1T1R), one-selector (or one-diode)-one-resistor (1S(D)1R), and one-resistor (1R). Sneak path[25] through unselected cells in the crossbar array may incur unexpected resistance switching in the logic operation. Therefore, 1T1R configuration is adopted in this paper, to alleviate the implication of the sneak path. 1R crossbar arrays presented in this paper are just used to show the function of the crossbar array. The practical realization of the logic operation is based on the 1T1R crossbar array. The voltages can be applied on WLs and BLs. Then the crossbar array can realize the vector-matrix multiplication (VMM) through Ohm's law and Kirchhoff's current law (KCL) through a read operation.[26] Figure 1(b) shows the realization of the VMM with the input voltages applied on the WLs and the output currents through BLs. And figure 1(c) shows the realization of the VMM with the input voltages applied on the BLs and the output currents through WLs. In Fig. 1(b), when BLs are connected to the ground directly, we can obtain

$$I_j = \sum_i V_i G_{ij}. \tag{1}$$

When BLs are connected to the grounded resistors R_{L}, respectively, we can gain

$$I_j = \frac{G_{\mathrm{L}} \sum_i V_i G_{ij}}{\sum_i G_{ij} + G_{\mathrm{L}}}, \tag{2}$$

where V_i represents the voltage applied on the i-th WL; I_j represents the current through the j-th BL; G_{ij} represents the conductance of the memristor at the intersection of the i-th WL and the j-th BL; G_{L} represents the conductance of the load resistor R_{L}.

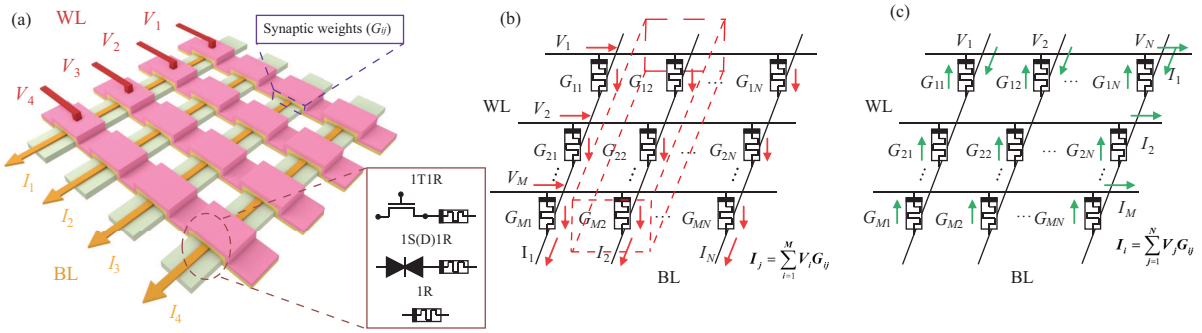


Fig. 1. (a) Memristive crossbar array, including 1T1R configuration and 1S(D)1R configuration. (b) Realization of the VMM with the input voltages applied on the WLs and the output currents through BLs. (c) Realization of the VMM with the input voltages applied on the BLs and the output currents through WLs.

Furthermore, if the resistance states of the memristors in each WL are the same, the voltages applied on memristors in each WL are the same. And $I_1 = \cdots = I_j = \cdots = I_n$. When the memristors in the crossbar array are the bipolar memristors, namely, they only have two resistance states: R_{LRS} and R_{HRS}, memristors in each BL can be designed as a logic function based on the stateful logic, where R_{LRS} represents logic '1' and R_{HRS} represents logic '0'. We can make use of these

characteristics for realizing n same logic functions simultaneously.

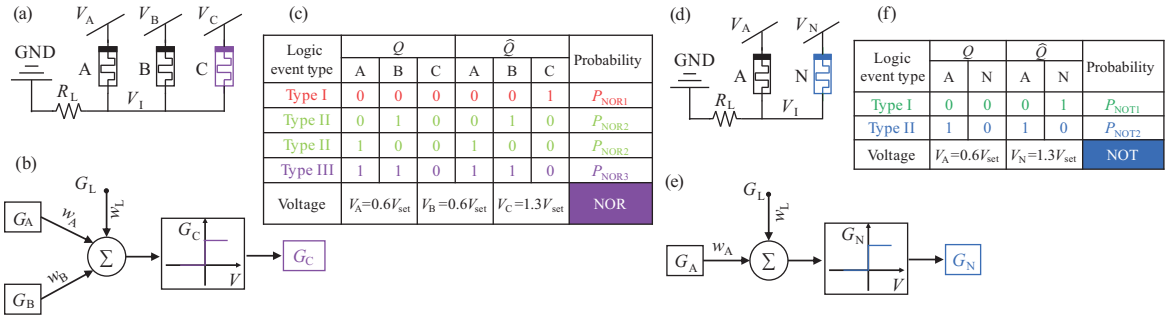
For example, V_A, V_B, and V_C are applied on WL_1, WL_2, and WL_3, respectively. And the other WLs are floating. It can realize the NOR logic based on the stateful neural network in Fig. 2(a). Memristors are connected to the grounded load resistor R_L. They would not be reset when the voltages between them are positive due to their polarity. The initial state of the output memristor C is R_{HRS}, representing logic "0". With a high resistance ratio of R_{HRS}/R_{LRS}, R_{LRS} conductance and R_{HRS} conductance can be normalized approximately as the real numbers 1 and 0, respectively, which allows the logical value and device conductance to be used interchangeably. According to Ohm's law and Kirchhoff's law, the node potential V_I can be deduced as

$$V_I = \frac{\sum_i V_i G_i}{\sum_i G_i}, \tag{3}$$

where $i = A, B, C$, and R_L, and $V_{RL} = 0$. When the voltage V_C–V_I exceeds the threshold voltage V_{set}, namely, V_C–V_I–$V_{set} > 0$, the output memristor C will be switched to R_{LRS} (logic "1").

$$\frac{(V_C - V_i - V_{set})\, G_i}{\sum_i G_i} = \frac{Y}{\sum_i G_i} > 0, \tag{4}$$

where Y can be viewed as the weighted sum in a neural network,[5] that is, $Y = \sum_i G_i w_i$. G_i values (the conductance values) work as the input variables and w_i values ($w_i = V_C$–V_i–V_{set}) work as the synaptic weights. Figure 2(b) shows the equivalent network model in Fig. 2(a). The w_i (voltage applied on the memristors) can be adjusted to obtain different outputs (logic output values). Figure 2(c) shows the truth table of the NOR logic. Similarly, figures 2(d), 2(e), and 2(f) show the NOT logic based on stateful neural network, single-layer perceptron network, and the truth table of the NOT logic, respectively.



(c)

Logic event type	Q			$\bar{Q}$			Probability
	A	B	C	A	B	C	
Type I	0	0	0	0	0	1	P_{NOR1}
Type II	0	1	0	0	1	0	P_{NOR2}
Type II	1	0	0	1	0	0	P_{NOR2}
Type III	1	1	0	1	1	0	P_{NOR3}
Voltage	V_A=0.6V_{set}			V_B=0.6V_{set}		V_C=1.3V_{set}	NOR

(f)

Logic event type	Q		$\bar{Q}$		Probability
	A	N	A	N	
Type I	0	0	0	1	P_{NOT1}
Type II	1	0	1	0	P_{NOT2}
Voltage	V_A=0.6V_{set}		V_N=1.3V_{set}		NOT

Fig. 2. (a) The NOR logic based on stateful neural network.[5] (b) Single-layer perceptron network for the NOR logic. (c) The truth table of the NOR logic. (d) The NOT logic based on stateful neural network.[5] (e) Single-layer perceptron network for the NOT logic. (f) The truth table of the NOT logic.

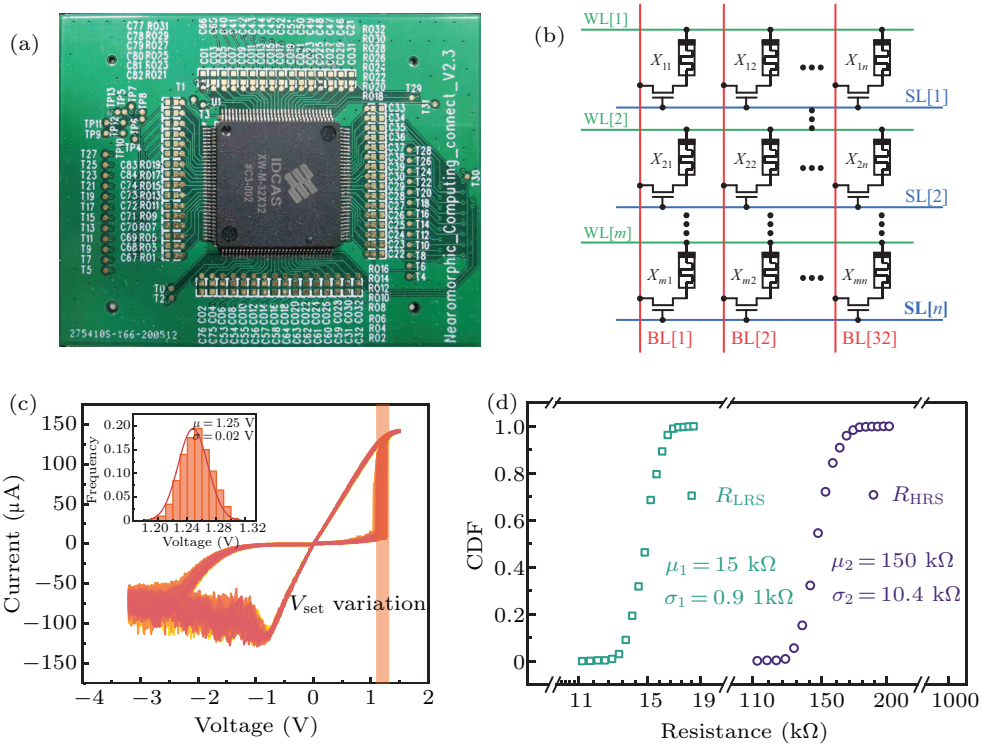


Fig. 3. Characterization of TiN/TaO_x/HfO_x/TiN memristor. (a) Optical image overview of the packaged chip. (b) The schematic of 32 × 32 1T1R crossbar array inside the chip. (c) I–V characteristics of the TiN/TaO_x/HfO_x/TiN memristor and the distribution of the set threshold voltage V_{set} measured in the direct current sweeping test. (d) Cumulative distribution function (CDF) of R_{LRS} and R_{HRS} measured in the pulse test.

To measure the device properties and verify the scheme, the circuit-level platform with the TiN/TaO_x/HfO_x/TiN 1T1R 32×32 crossbar array and its peripheral circuits are established. Figures 3(a) and 3(b) show the optical image overview of the packaged chip and the schematic of 32×32 1T1R crossbar array inside the chip, respectively. The transistor in the 1T1R array works as a selector to alleviate the implication of the sneak path and a current limiter to perform the compliance current. In the direct current sweeping test, when the selected memristor is set to the LRS, a voltage of 1.5 V is applied on the corresponding SL to open the selected memristor, and the corresponding WL sweeps from 0 V to 1.5 V, while the BL is grounded. When the selected memristor reset to the HRS, a voltage of 1.5 V is applied on the corresponding SL to open the selected memristor, and the corresponding BL sweeps from 0 V to 3.2 V, while the WL is grounded. The typical I–V characteristics of the bipolar memristor is shown in Fig. 3(c). And the normal distribution of the set threshold voltage $V_{\rm set}$ with the mean value $\mu = 1.25$ V and the standard deviation $\sigma = 0.02$ V is also presented in Fig. 3(c). In the pulse test, a voltage of 1.5 V is applied on the selected SL. And the set voltage and the reset voltage are 1.25 V and -2 V, respectively. The cycles are 1000. The cumulative distribution function (CDF) of $R_{\rm LRS}$ and $R_{\rm HRS}$ is shown in Fig. 3(d), where $R_{\rm LRS}$ follows the normal distribution with the mean value $\mu_1 = 15$ kΩ and the standard deviation $\sigma_1 = 0.91$ kΩ, and $R_{\rm HRS}$ follows the normal distribution with the mean value $\mu_2 = 150$ kΩ and the standard deviation $\sigma_2 = 10.4$ kΩ.

According to the practical device properties and principles of the stateful neural network mentioned above, the circuit parameters would be configured: $R_{\rm L} = 1.25R_{\rm LRS}$, and $V_{\rm A} = V_{\rm B} = 0.6V_{\rm set}$, $V_{\rm C} = V_{\rm N} = 1.3V_{\rm set}$. The parameters of the device and the circuit are shown in Table 1. Note that the circuit parameters remain unchanged in the simulations and tests.

Table 1. Parameter of the device and the circuit.

Variable	Parameter	Default value
$V_{\rm set}$ (V)	set threshold voltage	1.25
$V_{\rm reset}$ (V)	reset threshold voltage	−2.0
$R_{\rm LRS}$ (kΩ)	LRS of the memristor	15
$R_{\rm HRS}$ (kΩ)	HRS of the memristor	150
$V_{\rm A}$ (V)	voltage applied on A	0.75
$V_{\rm B}$ (V)	voltage applied on B	0.75
$V_{\rm C}$ (V)	voltage applied on C	1.63
$V_{\rm N}$ (V)	voltage applied on N	1.63
$R_{\rm L}$ (kΩ)	resistance of the load resistor	18.75

Furthermore, the throughput, which is the maximum number of instances that can be executed within the crossbar array within a given time unit, is defined as

$$\text{Throughput} = \frac{\#\text{instances}}{\text{Latency}}. \tag{5}$$

It can be improved by parallelizing the execution of many instances of the same logic function: each instance is placed in a different column as shown in Fig. 1(b).

3. Logic reliability

The logic function can be described as a set of mapping from the logic sequence $\boldsymbol{Q} = [E^1, \ldots, E^r, E^{r+1}]$ ($\boldsymbol{Q} = [X_1, \ldots, X_r, Y]$) to the expected logic sequence $\hat{\boldsymbol{Q}} = [\hat{E}^1, \ldots, \hat{E}^r, \hat{E}^{r+1}]$ ($\hat{\boldsymbol{Q}} = [\hat{X}_1, \ldots, \hat{X}_r, \hat{Y}]$). E^j represents the j-th element in $\boldsymbol{Q}$, $j = 1, \ldots, r+1$, which could be X or Y. X, Y, and r represent the input, the output, and the number of inputs, respectively. A logic event can be defined as a mapping from a sequence $Q_i \in \boldsymbol{Q}$ to the sequence $\hat{Q}_i \in \hat{\boldsymbol{Q}}$. The reliability of the logic event $Q_i \to \hat{Q}_i$ can be defined as the probability that the logic sequence Q_i can be mapped to $\hat{Q}_i$ correctly. And the reliability of the logic function $\boldsymbol{Q} \to \hat{\boldsymbol{Q}}$ can be defined as the probability that any logic sequence Q_i can be mapped to corresponding $\hat{Q}_i$ correctly.

Considering that the initial states of the outputs are fixed, $\boldsymbol{Q}$ has 2^r combinations. For each combination $Q_i \in \boldsymbol{Q}$, the probability that any element E_i^j in Q_i can be mapped to $\hat{E}_i^j$ in $\hat{Q}_i$ is p_i^j, where, that is, $P\left(\hat{E}_i^j | Q_i\right) = p_i^j$, $i = 1, \ldots, 2^r$, $j = 1, \ldots r+1$. Only if $P\left(\hat{E}_i^1 | Q_i\right) = \cdots = P\left(\hat{E}_i^{r+1} | Q_i\right) = 1$, can the logic event $Q_i \to \hat{Q}_i$ be performed correctly. As the logic events are mutually independent, the reliability of the logic function $\boldsymbol{Q} \to \hat{\boldsymbol{Q}}$ can be deduced:

$$P_{\rm f} = \prod_{i=1}^{2^r} P_{\rm ei}, \tag{6}$$

where $P_{\rm ei}$ represents the reliability of the logic event $Q_i \to \hat{Q}_i$.

For example, as for 2 input NOR stateful logic gates, there are 4 logic events from the input logic sequences $\boldsymbol{Q}$: $Q_1 = [0,0,0]$, $Q_2 = [0,1,0]$, $Q_3 = [1,0,0]$, and $Q_4 = [1,1,0]$ to the output logic sequences $\hat{\boldsymbol{Q}}$: $\hat{Q}_1 = [0,0,1]$, $\hat{Q}_2 = [0,1,0]$, $\hat{Q}_3 = [1,0,0]$, $\hat{Q}_4 = [1,1,0]$. When $P\left(\hat{E}_1^1 = 0 | Q_1\right) = P\left(\hat{E}_1^2 = 0 | Q_1\right) = P\left(\hat{E}_1^3 = 1 | Q_1\right) = 1$, the reliability of the logic event I $P_{\rm e1}$ is 1. The characteristics of these memristors are consistent with each other, and the voltages applied on the A and B are the same. Therefore, inputs A and B are equivalent to each other. Moreover, we can divide these 4 logic events into 3 types for the NOR logic. Type I: $Q_1 \to \hat{Q}_1$, Type II: $Q_2 \to \hat{Q}_2$, and $Q_3 \to \hat{Q}_3$, Type III: $Q_4 \to \hat{Q}_4$ as shown in Fig. 2(c). Similarly, we can divide 2 logic events into 2 types for the NOT logic. Type I: $Q_1 \to \hat{Q}_1$, Type II: $Q_2 \to \hat{Q}_2$ as shown in Fig. 2(f).

The implication of device instability on the reliability of the logic events in the NOR logic and the NOT logic is simulated as shown in Figs. 4(a) and 4(b). The circuit parameters are shown in Table 1. The device parameters ($V_{\rm set}$, $V_{\rm reset}$, $R_{\rm LRS}$, and $R_{\rm HRS}$) fluctuate appropriately based on the tested result. Assuming variable X satisfies the normal distribution $N\left(X_m, (X_m \times \text{rate})^2\right)$, where X_m, and rate represent the mean value, and fluctuation rate (σ/μ), respectively. For example, rate $= 0.016$ in $V_{\rm set}$, considering that the mean value of $V_{\rm set}$

is 1.25 V, namely, $V_{\rm set} \sim N\ (1.25, 0.02^2)$ (V). Here the devices follow $V_{\rm set} \sim N\ (1.25, 0.02^2)$ (V), $R_{\rm LRS} \sim N\ (15, (15 \times \text{rate})^2)$ (kΩ), and $R_{\rm HRS} \sim N\ (150, (150 \times \text{rate})^2)$ (kΩ) independently, and the rate fluctuates from 0.01 to 0.09. The simulation is based on the TEAM model, which is simple, flexible, general, and sufficiently accurate for memristive device-based circuit design.[27]

Figure 4 shows that the reliability of these logic events will keep decreasing as the device instability increases. In Fig. 4(a), $Q_2 \to \hat{Q}_2$, and $Q_3 \to \hat{Q}_3$ decrease synchronously as the device instability increases, which validates that these two logic events can be put into logic event type II together. Type I ($Q_1 \to \hat{Q}_1$) and type III ($Q_3 \to \hat{Q}_3$) have the worst and the best reliability in the device instability, respectively. In Fig. 4(b), type I has better reliability than type II initially. And type I decreases faster than type II with the instability increasing.

It demonstrates that device instability has a great impact on the logic reliability. However, device instability is inevitable based on the current fabrication technology. In order to solve such an issue, an n-modular redundancy error correction scheme is proposed to help the memristive stateful logic design towards high reliability.

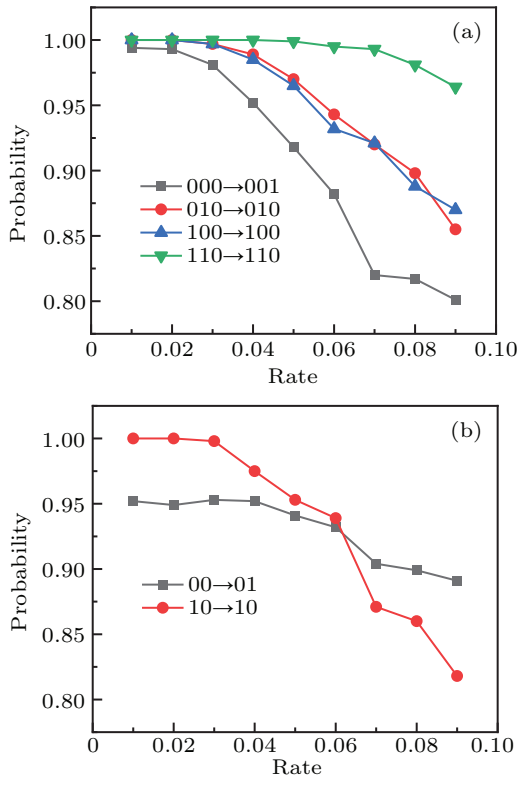


Fig. 4. The implication of device instability on the reliability of the logic events in the NOR logic (a) and the NOT logic (b).

4. Logic error correction scheme

4.1. Principle of the logic error correction scheme

We can make use of the vector–matrix multiplication in the memristive crossbar array to realize the error correction scheme based on n-modular redundancy efficiently. Figure 5 shows the r-input logic error correction circuit design based on n-modular redundancy. It is an $m \times n$ crossbar array whose WLs are connected to the corresponding current comparators CC. The red memristors are in incorrect states in the crossbar array. The green voltages $V_{\rm read}$ used for reading the memristors' states are applied on the BL and their current $I_{\rm D}$ is gathered to the corresponding comparator.

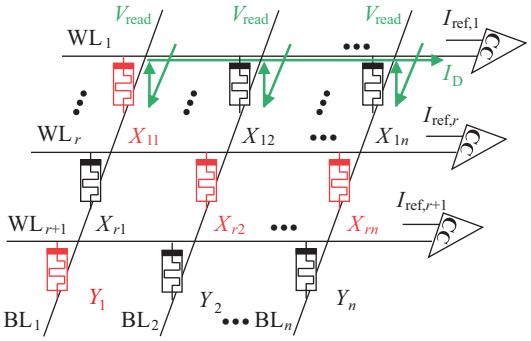


Fig. 5. The r-input logic error correction circuit design based on n-modular redundancy.[24]

There are n same r-input and logic functions ($r+1=m$) in the crossbar array. Through applying voltages on the WL$_{\rm S}$, n logic functions can be performed simultaneously as a result of the vector-matrix multiplication in the crossbar array. However, some memristors are in the incorrect resistance states due to the instability in memristors. If there are k memristors are in $R_{\rm LRS}$ and (mk) memristors are in $R_{\rm HRS}$, when $V_{\rm read}$ is applied on BL$_{\rm S}$, the current through WL $I_{\rm D}$ can be deduced according to Ohm's law and Kirchhoff's current law:

$$I_{\rm D}(k) = k \cdot \frac{V_{\rm read}}{R_{\rm on}} + (m-k) \cdot \frac{V_{\rm read}}{R_{\rm off}}. \tag{7}$$

Memristors in the crossbar array are fabricated in the same condition. Therefore, we can assume that they follow the same random distribution independently.

If $\hat{E}_i^j = 1$, it means that the corresponding memristor (the j-th memristor in a BL) in the corresponding logic combination (the i-th logic combination) will be $R_{\rm LRS}$ after the logic operation. For n-modular redundancy (n BLs in the crossbar array for the same logic operation with the same logic inputs), the probability that only k modules in the j-th WL are correct (k memristors are $R_{\rm LRS}$ in the j-th WL) can be deduced:

$$P(I_{\rm D}(k)) = C_n^k (p_i^j)^k (1-p_i^j)^{n-k}. \tag{8}$$

Moreover, the probability that $I_{\rm D} \geq I_{\rm D}(\lfloor n/2 \rfloor + 1)$ can be deduced:

$$\begin{aligned} P_{si}^j &= P\left(I_{\rm D} \geq I_{\rm D}\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)\right) \\ &= \sum_{k=\lfloor n/2 \rfloor + 1}^{n} C_n^k (p_i^j)^k (1-p_i^j)^{n-k}. \end{aligned} \tag{9}$$

Similarly, if $\hat{E}_i^j = 0$, it means that the corresponding memristor (the j-th memristor in a BL) in the corresponding logic combination (the i-th logic combination) will be $R_{\rm HRS}$

after the logic operation. The probability that I_D is lower than or equal to $I_D(\lfloor n/2 \rfloor)P_{si}$ can be deduced:

$$P_{si}^j = P\left(I_D \le I_D\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\right) = \sum_{k=\lfloor n/2 \rfloor+1}^{n} C_n^k (p_i^j)^k (1-p_i^j)^{n-k}. \quad (10)$$

In a single logic event, the relationship between the number of redundancy n for P_{si}^j reaching the expected reliability and p_i^j is shown in Fig. 6. It demonstrates that increasing n can increase reliability. Moreover, n will keep decreasing as p_i^j increases continuously and the change rate speeds down as p_i^j increases continuously.

If $p_i^j > 50\%$ for any p_i^j, this probability can keep increasing and close to 100% as the number of repeated logic operations n increases.

If $p_i^j < 50\%$ for any p_i^j, that is, $1 - p_i^j > 50\%$ for any p_i^j. the approach agrees with the condition $p_i^j > 50\%$ for any p_i^j. And the operation is presented in Table 2.

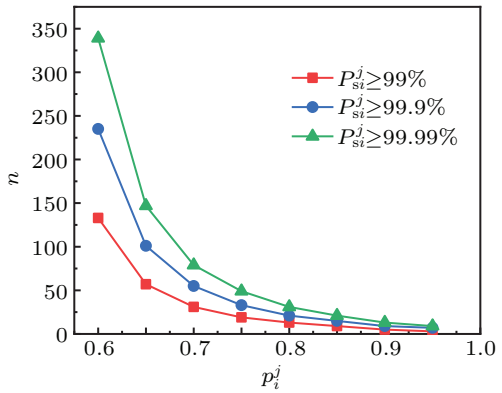


Fig. 6. The relationship between the number of redundancy n for reaching expected success rate and p_i.[24]

Table 2. The operation based on different conditions.

Condition	R_{LRS}, $p_i^j > 50\%$	R_{HRS}, $p_i^j > 50\%$	R_{LRS}, $p_i^j < 50\%$	R_{HRS}, $p_i^j < 50\%$
I_{ref}	$I_D\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)$	$I_D\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$	$I_D\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$	$I_D\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)$
Compare	$I_D \ge I_{ref}$	$I_D \le I_{ref}$	$I_D \le I_{ref}$	$I_D \ge I_{ref}$
Success rate	$\sum_{k=\lfloor n/2 \rfloor+1}^{n} C_n^k p_1^k (1-p_1)^{n-k}$		$\sum_{k=0}^{\lfloor n/2 \rfloor} C_n^k p_1^k (1-p_1)^{n-k}$	
Operation	set/reset	reset/set	set/reset	reset/set

Therefore, for each element in any logic combination, when the probability p_i^j can be evaluated, P_{si}^j can be promoted through n-modular redundancy bit by bit. It can enhance the reliability of the logic event. And then the reliability of the logic function can be enhanced according to Eq. (6).

4.2. Design of the error correction scheme

The inequations in Eqs. (9) and (10) can be performed with a current comparator. We can compare I_D with I_{ref} which represents the critical current when the number of R_{LRS} accounts for half of the n memristors. If I_D is $> I_{ref}$, we will apply V_{set} on these n memristors to make them in R_{LRS}, or we will apply V_{reset} on them to make them in R_{HRS}, under the condition $p_i^j > 50\%$ for any p_i^j.

Figure 7 shows the universal steps for the error correction scheme based on the n-modular redundancy. The red memristors represent the memristors that are in incorrect states. The blue current comparator CC represents the activated current comparator CC. The green and red arrows represent the directions of read voltage (V_{read}) and write voltage (V_{set} and V_{reset}), respectively.

There are $2r+2$ steps in the whole correction scheme for the r-input logic. It corrects the memristors WL by WL. In each WL including r inputs and 1 output, we compare the current I_D with I_{ref} in the first step. Then a write operation on all memristors in the same WL according to the output of the comparator is performed. If it is desired for the reset operation, V_{reset} is applied on BLs and the corresponding WL is connected to the ground as shown in Fig. 7(b). And if it is desired for set operation, V_{set} is applied on the corresponding WL, and BLs are connected to the ground as shown in Figs. 7(d) and 7(f). The other WLs are floating.

Figure 8 shows the schematic diagram for the logic error correction based on the n-modular redundancy. The green block and the red block represent the input memristor and the output memristor, respectively. And there are r inputs and 1 output in the logic operation. Figure 8(a) shows the universal design and can correct the errors including the input errors and the output errors, which corresponds to Fig. 5. The n same logic operation can be performed in 1 step. And the logic correction operation is $2r+2$, as shown in Fig. 7. It costs $(r+1)\times n$ memristors and $2r+3$ clock cycles.

When the r input memristors can be kept in the correct state in the logic operation, the design in Fig. 8(a) costs $(r+1)\times n$ memristors and 3 clock cycles. Therefore, it can be regarded as the time-optimal logic correction design. Furthermore, when the r input memristors can be kept in the correct state in the logic operation, the scheme can be expanded to another two designs. Figure 8(b) shows the area-optimal logic correction design. The logic operation can be performed step by step. It costs $r+n$ memristors and $n+2$ clock cycles. Figure 8(c) shows the trade-off logic correction design. Note that n/d is an integer generally. It costs $(r+n/d)\times n$ memristors and $n/d+2$ clock cycles. The time-optimal design and the area-optimal design represent the conditions $n=d$ and $d=1$ in the trade-off design, respectively.

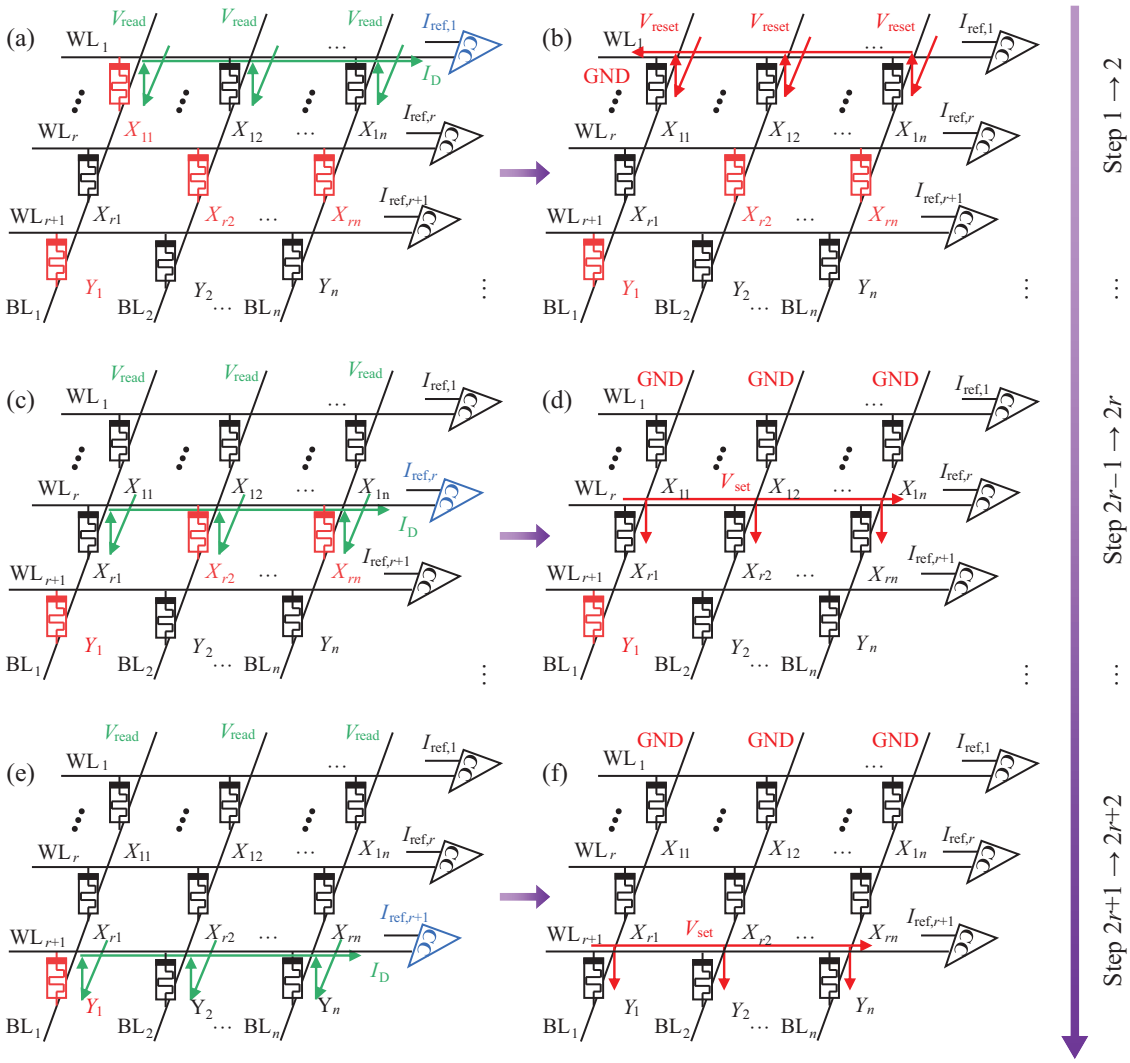


Fig. 7. The r-input logic error correction step based on n-modular redundancy.[24]

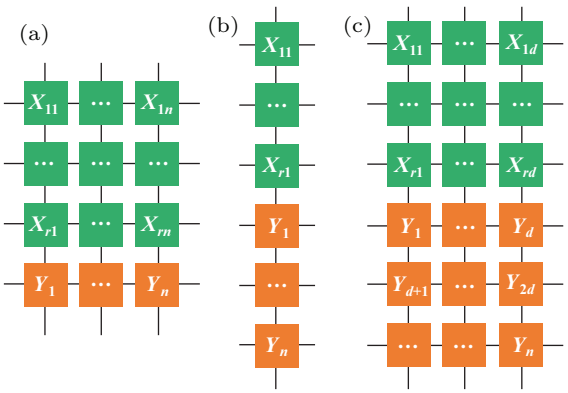


Fig. 8. The schematic diagram for the logic error correction based on the n-modular redundancy. (a) the time-optimal logic correction design, (b) the area-optimal logic correction design, (c) the trade-off logic correction design.

Figure 9 presents the schematic diagram of the proposed n-modular redundancy error correction circuit in the 1T1R array. It consists of ANALOG_TOP composed of the crossbar array and the error correction block, and DIGITI_TOP used for controlling the circuit. In the crossbar array, WLs, BLs, and SLs are connected with the WL drivers, BL drivers, and SL drivers, respectively. BLs and WLs are only connected with the 1T1R. Each SL is connected with the gate of the transistor in the corresponding line, which can control the transistor (open or close) through applying voltages (V_{open} and V_{close}) on the corresponding line. The WL drivers and the BL drivers can apply different voltages (V_x, V_{read}, V_{set}, and V_{reset}) and states ("G", "Z", and "F") on WLs and BLs, respectively. V_x represents the applied voltages on the memristors in the logic operations. And the states "G", "Z", and "F" represent that the node is grounded, the node is floating, and the node is connected with the grounded resistor R_{L}, respectively.

The error correction block consists of a multiplexer MUX, and a current comparator. And the error correction circuit is the current comparator. The circuit diagram of the current comparator which consists of 14 transistors is as shown in Fig. 10. It compares the read current I_{D} with the reference current I_{ref} through comparing the current of two branches out_1 and out_2. If $I_{\text{D}} > I_{\text{ref}}$, the output signal will be high level signal. Otherwise, the signal will be low level signal.

The current comparator is connected with MUX. Through the selection signal Ctrl_L of MUX, the current comparator can be connected with the specific BL or WL. Then the cur-

rent I_D will be compared with I_{ref}. The output of the current comparator will be transmitted into the DIGITI_TOP.

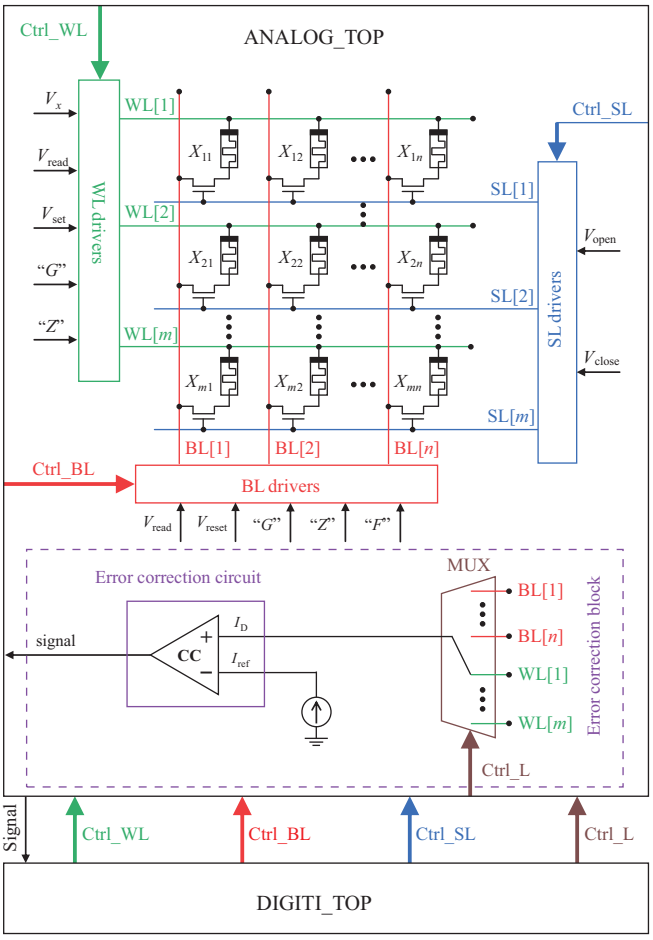


Fig. 9. Schematic diagram of the proposed N-modular redundancy error correction circuit in $m \times n$ 1T1R array.

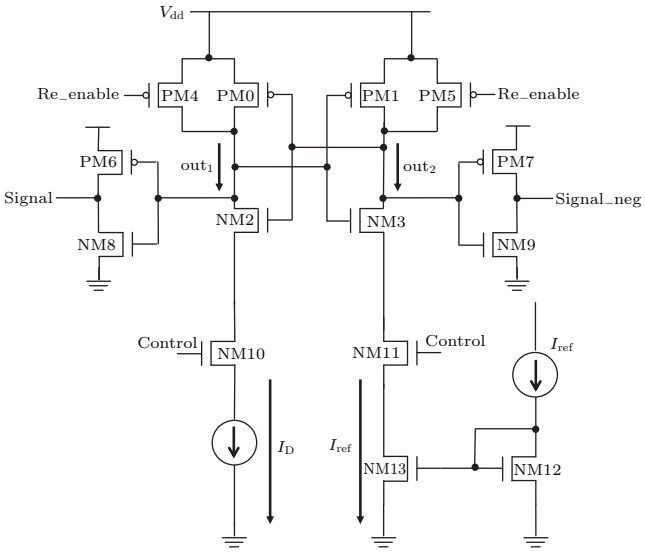


Fig. 10. Circuit diagram of the current comparator.

DIGITI_TOP can control the crossbar array and the error correction block simultaneously through Ctrl_WL, Ctrl_BL, Ctrl_SL, and Ctrl_L.

We take the NOR logic based on the stateful neural network with 3-modular redundancy for an example to demonstrate the process. When the input memristors can be kept in the correct state in the logic operation, the error correction needs to be performed on the output memristors denoted by X_{31}, X_{32}, and X_{33}.

(i) Perform the logic operation in parallel: WL[1]–WL[3] are applied on V_A, V_B, and V_C, respectively. And the other WLs are in the "Z" state. BL[1]–BL[3] are in the "F" state and the other BLs are in "Z" state. V_{open} is applied on SL[1]–SL[3] and V_{close} is applied on the other SLs.

(ii) Read the logic states in parallel: WL[3] is connected with the current comparator. And the other WLs are in the "Z" state. BL[1]-BL[3] are applied on V_{read} and other BLs are in the "Z" state. V_{open} is applied on SL[3] and V_{close} is applied on the other SLs.

(iii) Write the logic states in parallel: V_{open} is applied on SL[3] and V_{close} is applied on the other SLs. If $I_D > I_{ref}$, V_{set} is applied on WL[3] and the other WLs are in the "Z" state. BL[1]–BL[3] are in the "G" state and the other BLs are in "Z" state; If $I_D < I_{ref}$, V_{reset} is applied on BL[1]–BL[3], and the other BLs are in "Z" state. WL[3] is in the "G" state and the other WLs are in the "Z" state.

The schematic plot is implemented in the time-optimal logic correction design here. And it also can be implemented in the area-optimal logic correction design. Note that the positions of the applied voltages need to be changed synchronously.

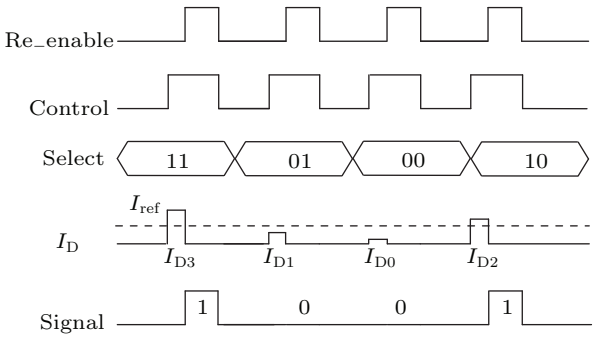


Fig. 11. The output signal of four data patterns with 3-modular redundancy in the NOR logic.

Cadence Virtuoso is utilized to complete the simulation of analog-mixed signal. The memristor model is TEAM model, and the other circuits are based on Semiconductor Manufacturing International Corporation (SMIC) 40-nm technology. Figure 11 shows the output signal of four data patterns with 3-modular redundancy in the NOR logic. re_enable and control are the control signal of the current comparator. select represents the different data patterns for 3-modular redundancy. "00" represents that 3 output memristors' resistance is all in HRS. "01" represents that two output memristors' resistance is in HRS. "10" represents that one output memristor's resistance is in HRS. And "11" represents that 3 output memristors' resistance is all in LRS. I_D represents the read current of the three output memristors. It shows I_D is larger than I_{ref} when

select is "11" or "10". And its corresponding output signal is 1. It demonstrates the circuit can work well for the logic error correction.

5. Simulation and discussion

In this section, we take the 1-bit FA as an example to simulate the implication of the device instability on the reliability of the combinational logic circuit. Then the scheme is carried out to improve the reliability of the logic. The 1-bit FA based on the stateful neural network with two steps, whose probability based on the tested result in Fig. 3 is 11.2%, is hard to be realized. Therefore, the 1-bit FA is designed with the NOR logic and NOT logic based on the stateful neural network through multiple steps. With the help of SIMPLER,[23] the 1-bit FA is mapped into a single column within a crossbar array. SIMPLER, short of synthesis and in-memory mapping of logic execution in a single row, can reduce the minimum required number of cells for the execution of the given logical function and reduce the execution latency using a column (or row), which can be combined with the full parallel stateful logic computation to realize the scheme design from logical primitives to logical functions.

5.1. The 1-bit FA simulation

The 1-bit FA based on the NOR logic and the NOT logic within a crossbar array in a single column is proposed according to the SIMPLER. Figure 12(a) shows the 1-bit FA netlist which consists of seven NOR logic gates (g_4, g_5, g_6, g_8, g_9, g_{10}, and g_{11}) and five NOT logic gates (g_1, g_2, g_3, g_7, and g_{12}). Figure 12(b) shows the execution of the 1-bit FA netlist using a single column of the memory with nine memristors. The first three memristors serve as inputs A, B, and C_i and will not be destroyed. The other six memristors are used for the logic operation and the storage of the temporary logic value. There are six steps including 15 clock cycles for the complete 1-bit FA. Step I: initialize the six memristors to logic '0' (R_{HRS}) for the following logic operation. Step II: serial logic operation of the six gates. Step III: initialize the three memristors not required as inputs for following logic operations. Step IV: serial logic operation of the next three gates. Step V: initialize the three memristors not required as inputs for following logic operations. Step VI: serial logic operation of the last three gates.

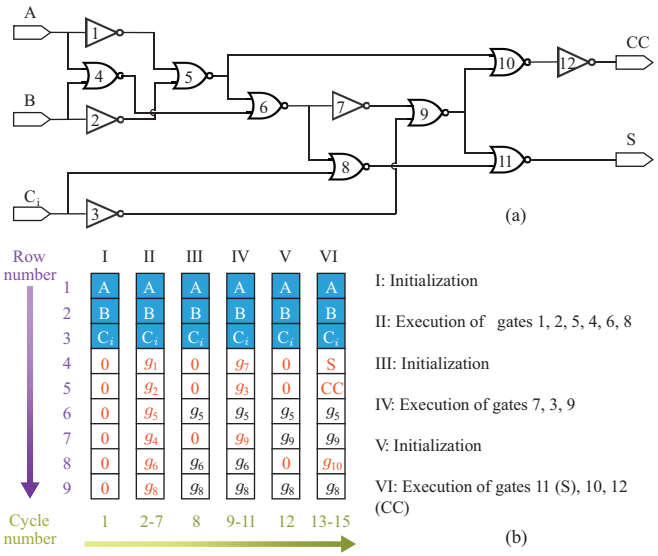


Fig. 12. (a) 1-bit FA netlist based on the NOR logic gate and NOT logic gate. (b) Execution of the 1-bit FA netlist using a single column of the memory with nine memristors.

The simulation algorithm of n-modular redundancy based on the 1-bit FA is demonstrated in Algorithm 1. Note that the states of the input memristors in the logic gates won't be destroyed in this device characteristics in Table 1. Therefore, the error correction only needs to be carried out for the output cell.

(i) The input such as "000" is obtained from the list logic_inputs consisting of ["000", "001", "010", "011", "100", "101", "110", "111"]. A, B, and C_{in} are initialized according to the input. The other six memristors are initialized as logic "0" for the logic operation.

Algorithm 1. Simulation algorithm of n-modular redundancy based on the 1-bit FA.

```
logic_inputs = ["000", "001", ..., "111"]
logic_gates  = ["g1", "g2", ..., "g12"]
for i in redundancy do
    repeat N_S do
        logic_outputs = []
        for input = "000", "001", ..., "111" do
            initialize nine memristors' states according to the input
            for logic_gate = "g1", "g2", ..., "g12" do
                get the input state in_gate and the output state out_gate in the logic_gate
                V_D = 0
                repeat i do
                    generate the random distribution
                    write logic netlist file according to the in_gate and out_gate including their respective properties
                    run the netlist file and read the resistance of the out_gate R_out
                    V_D += V_read/R_out × R_f
                out_gate = logic "1" if V_D > V_ref else logic "0"
            get the final logic output and put it into logic_outputs
        compare logic_outputs with different logic operation truth table and count logical times N_L
    get P_L^i = N_L/N_S
```

(ii) The 1-bit FA based on the i-modular redundancy circuit is designed, whose memristors' states are determined by the input. Each memristor follows the same random normal distribution independently. And 12 logic operations are performed in sequence. Accumulate the voltage $V_{\rm D}$ for i times and compare the voltage with the reference voltage $V_{\rm ref}$. The logic error correction can be executed for the output memristor in every logic operation. After 12 logic operations are finished, the output sequence can be obtained.

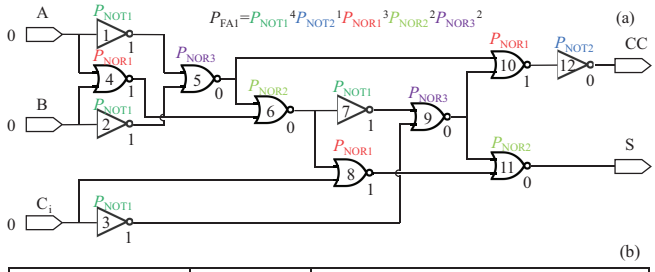
(iii) Walk through the input combinations and get the logic_outputs. We can compare it with the standard 1-bit FA logic output sequence and count the right logical times $N_{\rm L}$.

(iv) Repeat the above steps 1–3 for $N_{\rm S}$ times, and the reliability of the 1-bit FA based on i-modular redundancy can be calculated: $P_{\rm L}^{i} = N_{\rm L}/N_{\rm S}$.

(v) Increase the redundancy i and repeat the above steps 1–4. The reliability of the 1-bit FA for different modular redundancies can be obtained.

5.2. Discussion

The probability of the 1-bit FA can be evaluated based on the parameters of the devices and the circuits. Figure 13(a) shows the probability of the logic event [0, 0, 0, 0, 0] → [0, 0, 0, 0, 0] in the 1-bit FA. When the memristor A is the logic "0", the probability of the logic event [0, 0] → [0, 1] in the NOT gate g_1 is $P_{\rm NOT1}$. Similarly, the probability of the logic events in 12 logic gates and the logic event [0, 0, 0, 0, 0] → [0, 0, 0, 0, 0] in the 1-bit FA can be obtained. Furthermore, the other seven logic events in the 1-bit FA are obtained. And the probability of the 1-bit FA can be deduced according to Eq. (6) as shown in Fig. 13(b).



Input			Output		Probability
A	B	C_i	CC	S	SUM
0	0	0	0	0	$P_{\rm FA1}=P_{\rm NOT1}{}^{4}P_{\rm NOT2}{}^{1}P_{\rm NOR1}{}^{3}P_{\rm NOR2}{}^{2}P_{\rm NOR3}{}^{2}$
0	0	1	0	1	$P_{\rm FA2}=P_{\rm NOT1}{}^{3}P_{\rm NOT2}{}^{2}P_{\rm NOR1}{}^{3}P_{\rm NOR2}{}^{3}P_{\rm NOR3}{}^{1}$
0	1	0	0	1	$P_{\rm FA3}=P_{\rm NOT1}{}^{2}P_{\rm NOT2}{}^{3}P_{\rm NOR1}{}^{3}P_{\rm NOR2}{}^{4}P_{\rm NOR3}{}^{0}$
0	1	1	1	0	$P_{\rm FA4}=P_{\rm NOT1}{}^{2}P_{\rm NOT2}{}^{3}P_{\rm NOR1}{}^{3}P_{\rm NOR2}{}^{2}P_{\rm NOR3}{}^{1}$
1	0	0	0	1	$P_{\rm FA5}=P_{\rm NOT1}{}^{2}P_{\rm NOT2}{}^{3}P_{\rm NOR1}{}^{3}P_{\rm NOR2}{}^{4}P_{\rm NOR3}{}^{0}$
1	0	1	1	0	$P_{\rm FA6}=P_{\rm NOT1}{}^{2}P_{\rm NOT2}{}^{3}P_{\rm NOR1}{}^{2}P_{\rm NOR2}{}^{4}P_{\rm NOR3}{}^{1}$
1	1	0	1	0	$P_{\rm FA7}=P_{\rm NOT1}{}^{3}P_{\rm NOT2}{}^{2}P_{\rm NOR1}{}^{2}P_{\rm NOR2}{}^{3}P_{\rm NOR3}{}^{2}$
1	1	1	1	1	$P_{\rm FA8}=P_{\rm NOT1}{}^{2}P_{\rm NOT2}{}^{3}P_{\rm NOR1}{}^{2}P_{\rm NOR2}{}^{4}P_{\rm NOR3}{}^{1}$
Logic function 1-bit FA					$P_{\rm FA}=P_{\rm NOT1}{}^{20}P_{\rm NOT2}{}^{20}P_{\rm NOR1}{}^{20}P_{\rm NOR2}{}^{28}P_{\rm NOR3}{}^{8}$

Fig. 13. (a) The probability of the logic event [0, 0, 0, 0, 0] → [0, 0, 0, 0, 0] in the 1-bit FA. (b) Statistics on the number of the NOR and the NOT logic events required by the 1-bit FA logic events.

The probability of the 1-bit FA with n-modular redundancy is simulated and mathematically evaluated, and the results are shown in Fig. 14. The devices follow $V_{\rm set} \sim N$ (1.25, 0.02^2) (V), $R_{\rm LRS} \sim N$ $(15, (15 \times {\rm rate})^2)$ (kΩ), and $R_{\rm HRS} \sim N$ $(150, (150 \times {\rm rate})^2)$ (kΩ) independently. S1, S3, S5, and S7 represent the simulation probability of the 1-bit FA with 1, 3, 5, and 7 modular redundancies, respectively. E1, E3, E5, and E7 represent the mathematical evaluation results of the 1-bit FA with 1, 3, 5, and 7 modular redundancies, respectively. The total simulation times $N_{\rm S} = 1000$. The rate of $R_{\rm LRS}$ and $R_{\rm HRS}$ fluctuates from 0.01 to 0.09. It shows the probability of the 1-bit FA decreases as the rate of $R_{\rm LRS}$ and $R_{\rm HRS}$ increases. And n-modular redundancy can improve the probability significantly. What is more, the evaluation results can fit simulation results well, which can verify the evaluation methodology.

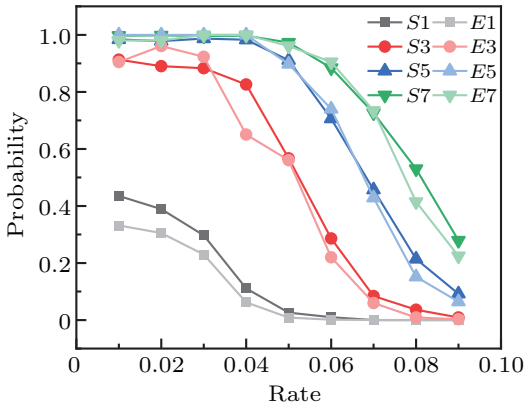


Fig. 14. The simulation and evaluation probability of the n-modular redundancy based on the 1-bit FA.

The whole steps can be evaluated in the simulation. The 1-bit FA takes 15 steps, three of which are initialization for following logical operation in Fig. 12(b). The error correction only needs to be carried out for the output cell according to the device characteristics. Therefore, the logical error corrections need to be performed 12 times. If the time-optimal logic correction scheme is carried out where the error correction performs in parallel, the whole steps require 39. And it takes three times of 9, 27 in total, devices among memristive crossbar array.

The throughput of the logic operation with the logic correction can be evaluated. Figure 15 shows all aligned logic column operations with n-modular redundancy of t different row-aligned instances are executed in parallel. And it takes the time-optimal logic correction design. We assume there are a steps for logic operations and b steps for logic corrections. Then there are $a + t \times b$ steps in parallel with the design in Fig. 9. And there are $(a + b) \times t$ steps for the other correction schemes.[17–22] The throughput of this work is $t/(a + t \times b)$, which is larger than that of the other correction schemes $1/(a + b)$ according to Eq. (5).

There are four kinds of error correction schemes[17–22] except for this work. We compare this work with the others from four angles. Table 3 shows the comparison of the error correction scheme.

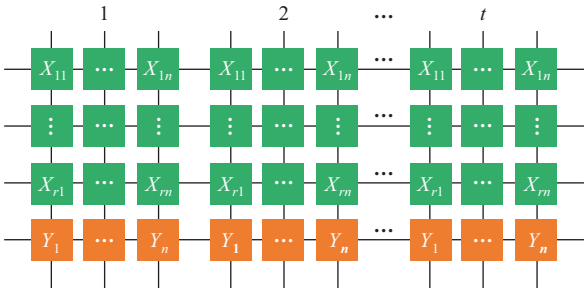


Fig. 15. All aligned logic column operations with n-modular redundancy of t different row-aligned instances are executed in parallel.

The authors Refs. [19,20] improved the logic reliability through reasonable multiple operations. Therefore, it does not need additional CMOS circuits. The cycles for error corrections of the primary logic vary from 2–4 according to the devices' properties. It can work for IMP and MAGIC. The logic reliability with it may not be able to reach 100% when the resource consumption is not considered.

Reference [22] divided the logic gates into NOR-type gates and non-NOR-type gates. As for NOR-type gates such as NOR, it makes use of zero checking module (ZCM) consisting of a three-input NOR CMOS logic gate. Three-input NOR CMOS logic gate needs about 6 transistors. As for non-NOR-type gates such as NAND, it designs a balance gate consisting of a memristive logic gate to combine these two gates into one logic gate for special detections. Note that the reliability of the balance gate must be 100%. Therefore, before the error correction of the non-NOR-type gate, its balance gate must be corrected first. If the balance gate belongs to non-NOR-type logic gate, it also needs another balance gate to correct the balance gate... When the reliability of the balance gate is 100%, the odd checking module (OCM) consisting of two three-input XOR and one XOR CMOS logic gates is performed to start the error correction. The OCM needs about $16 \times 2 \times 2 + 16 = 80$ transistors. And the total number of transistors is $6 + 80 = 86$. And the cycles for error corrections of the primary logic are 3 for NOR-type gates, and at least 6 for non-NOR-type gates. It can work for all kinds of R–R logics. The logic reliability with it always can be able to reach 100% when the resource consumption is not considered.

The authors in Ref. [18] detected two kinds of error types through comparing the node voltages twice, then corrects them with different voltages. The proposed error detection circuit totally requires 72 transistors, where the digital sub-circuits including flip-flops, MUX, and XOR gate, are all designed with the minimized size.[18] And the cycles for error corrections of the primary logic are 4. It can work for IMP. The logic reliability with it always can be able to reach 100% when the resource consumption is not considered.

The authors in Ref. [19] corrected the logic error with the additional memristive logic operations. Therefore, it does not need additional CMOS circuits. The cycles for error corrections of the primary logic are 5. It can work for IMP. The logic reliability with it may not be able to reach 100% when the resource consumption is not considered.

This work judges the correct logic according to the share of different logic values. And the judgment is realized with a current comparator. Therefore, all kinds of logics whose logic values are represented by resistance states are suitable for this work. The current comparator needs 14 transistors. And the cycles for error corrections of the primary logic are 3. It can work for all kinds of R–R logics and V–R logics. The logic reliability with it always can be able to reach 100% when the resource consumption is not considered.

In summary, although these three approaches (Refs. [18, 22], and this work) need additional CMOS circuits for improving reliability, the logic reliability with them always can reach 100% when the resource consumption is not considered. The other two approaches cannot. The number of transistors this work needs is observably less than the other two approaches. And the cycles for error corrections of the primary logic of this work needs are less than those of the other two approaches, too. What is more, this work can be suitable for all kinds of R–R logics and V–R logics. Its universality is more powerful than the others.

Table 3. Comparison of the error correction scheme.

	The number of transistors in the error correction circuit	The cycles for error corrections of the primary logic	What types of logics the scheme can work	Whether the reliability always can reach 100% when the scheme can work
Zhu *et al.*[20,21]	0	2–4	IMP, and MAGIC	No
In *et al.*[22]	86	3 or 6	R–R logic	Yes
Xu *et al.*[18]	72	4	IMP	Yes
Li *et al.*[19]	0	5	IMP	No
This work	14	3	R–R logic, and V–R logic	Yes

6. Conclusion

In this paper, we simulated the implication of device instability on the reliability of the logic event with the TEAM model in HSPICE. The mathematical relationship between logic reliability and redundancy has been deduced. Then through combining the mathematical relationship with the vector-matrix multiplication in a memristive crossbar array, the logic error correction scheme has been proposed. It can be implemented in any stateful logic and the logic based on them. Moreover, a universal design paradigm has been put

forward for the complex logic with SIMPLER. And the circuit schematic and the flow of the scheme have been raised. Finally, 1-bit FA based on the NOR logic and NOT logic within a crossbar array in a single column is simulated and the mathematical evaluation according to the device properties is performed. It demonstrates the scheme can improve the reliability of the logic significantly. And compared with other four error corrections, the scheme which can be suitable for all kinds of R–R logics and V–R logics has the best universality and throughput. Although it needs additional CMOS circuits, the logic reliability with it always can reach 100% when the resource consumption is not considered. And compared with the other two approaches which also need additional CMOS circuits, it needs fewer transistors and cycles for the error correction.

Acknowledgements

Project supported by the National Key Research and Development Plan of the Ministry of Science of Technology of China (Grand Nos. 2019YFB 2205100 and 2019YFB 2205102), the National Natural Science Foundation of China (Grant Nos. 61974164, 62074166, 61804181, 62004219, and 62004220), and the Science Support Program of the National University of Defense and Technology (Grand No. ZK20-06).

References

[1] Wang Z, Wu H, Burr G W, Hwang C S, Wang K L, Xia Q and Yang J J 2020 *Nat. Rev. Mater.* **5** 173
[2] Mutlu O, Ghose S, Gómez-Luna J and Ausavarungnirun R 2019 *Microprocess. Microsyst.* **67** 28
[3] Borghetti J, Snider G S, Kuekes P J, Yang J J, Stewart D R and Williams R S 2010 *Nature* **464** 873
[4] Kvatinsky S, Member S, Belousov D, Liman S, Satat G, Member S, Wald N, Friedman E G, Kolodny A, Member S and Weiser U C 2014 *IEEE Trans. Circuits Syst. II Express Briefs* **61** 895
[5] Sun Z, Ambrosi E, Bricalli A and Ielmini D 2018 *Adv. Mater.* **30** 2
[6] Hu X, Schultis M J, Kramer M, Bagla A, Shetty A and Friedman J S 2019 *IEEE Trans. Circuits Syst. I Regul. Pap.* **66** 263
[7] Li Y, Zhou Y X, Xu L, Lu K, Wang Z R, Duan N, Jiang L, Cheng L, Chang T C, Chang K C, Sun H J, Xue K H and Miao X S 2016 *ACS Appl. Mater. Interfaces* **8** 34559
[8] Huang P, Kang J, Zhao Y, Chen S, Han R, Zhou Z, Chen Z, Ma W, Li M, Liu L and Liu X 2016 *Adv. Mater.* **28** 9758
[9] Yang B, Xu N, Zhou E, Li Z, Li C, Yi P and Fang L 2020 *Chin. Phys. B* **29** 48505
[10] Li Z, Chen P Y, Xu H and Yu S 2017 *IEEE Trans. Electron Dev.* **64** 2721
[11] Xu N, Yoon K J, Kim K M, Fang L and Hwang C S 2018 *Adv. Electron. Mater.* **4** 1
[12] Jiang W, Li J, Liu H, Qian X, Ge Y, Wang L and Duan S 2022 *Chin. Phys. B* **31** 040702
[13] Chen B, Cai F, Zhou J, Ma W, Sheridan P and Lu W D 2015 *Tech. Dig. - Int. Electron Devices Meet. IEDM* 2016-February 17.5.1–17.5.4
[14] Ielmini D and Wong H S P 2018 *Nat. Electron.* **1** 333
[15] Liu S, Lu N, Zhao X, Xu H, Banerjee W, Lv H, Long S, Li Q, Liu Q and Liu M 2016 *Adv. Mater.* **28** 10623
[16] Koroleva A A, Chernikova A G, Chouprik A A, Gornev E S, Slavich A S, Khakimov R R, Korostylev E V., Hwang C S and Markeev A M 2020 *ACS Appl. Mater. Interfaces* **12** 55331
[17] Jang J, Gi S, Yeo I, Choi S, Jang S, Ham S, Lee B and Wang G 2022 *Adv. Sci.* **9** 2201117
[18] Xu J, Zhan Y, Li Y, Wu J, Ji X, Yu G, Jiang W, Zhao R and Wang C 2021 *IEEE Trans. Circuits Syst. I Regul. Pap.* **69** 309
[19] Li Z, Long H, Zhu X, Wang Y, Liu H, Li Q, Xu N and Xu H 2022 *Adv. Intell. Syst.* **4** 2100234
[20] Zhu X, Li Z, Long H, Liu H, Wang Y and Xu H 2020 *IEEE International Symposium on Circuits and Systems (ISCAS)* (Sevilla) pp. 1–5
[21] Zhu X, Long H, Li Z, Diao J, Liu H, Li N and Xu H 2020 *Microelectronics J.* **103** 104866
[22] In J H, Kim Y S, Song H, Kim G M, An J, Jeon J B and Kim K M 2020 *Adv. Intell. Syst.* **2** 2000081
[23] Ben-Hur R, Ronen R, Haj-Ali A, Bhattacharjee D, Eliahu A, Peled N and Kvatinsky S 2019 *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **39** 2434
[24] Zhu X, Xu H, Long H, Li Q, Li Z, Liu H and Wang Y 2021 *IEEE Electron Devices Technology and Manufacturing Conference (EDTM)* (China) pp. 1–3
[25] Kim S, Zhou J and Lu W D 2014 *IEEE Trans. Electron Dev.* **61** 2820
[26] Zhu X, Li Z, Liu H, Li Q, Liu S, Li N and Xu H 2020 *IET Circuits, Dev. Syst.* **14** 498
[27] Kvatinsky S, Friedman E G, Kolodny A and Weiser U C 2013 *IEEE Trans. Circuits Syst. I Regul. Paper* **60** 211