Circuits, Systems, and Signal Processing (2020) 39:2822–2840
https://doi.org/10.1007/s00034-019-01300-0

Design of High-Speed Logic Circuits with Four-Step RRAM-Based Logic Gates

Xiaole Cui[1] · Xiao Ma[1] · Qiujun Lin[1] · Xiang Li[1] · Hang Zhou[1] · Xiaoxin Cui[2]

Received: 22 March 2019 / Revised: 27 October 2019 / Accepted: 29 October 2019 /
Published online: 11 November 2019

Abstract
The RRAM (resistive random-access memory) is one of the most competitive candidates of the emerging non-volatile memory devices. In recent years, the RRAM has been used as memory device and also to build logic circuit. However, the design method of the RRAM-based logic circuit is still an open issue. This paper proposes a design method of logic circuit based on the four-step RRAM logic gates. The design rules are studied for both the combinational and the sequential logic circuits in a parallel style. The design practices and synthesis results show that the proposed methods generate the high-performance circuits for the arbitrary logic functions. Moreover, the four-step RRAM-based logic gates are suitable for designing the circuits with pipelined architecture, since the different RRAM logic blocks have the uniform working speed. The pipelined N-bit ripple carry adder and the N-bit multiplier outperform the other RRAM base counterparts; the output results are obtained only with $2N+2$ and $6N-4$ working cycles, respectively.

Keywords Four-step RRAM-based logic gates · Combinational logic · Sequential logic · Pipelined architecture

1 Introduction

The RRAM (resistive random-access memory) is a two-terminal memory device that represents the logic states with its resistance values. The RRAM device turns to the low resistance state (LRS) if a positive voltage $V_{\text{set}} > V_{\text{on}}$ is applied to it; this operation is named as the set operation. The reset operation is conducted by applying a negative voltage $V_{\text{reset}} < V_{\text{off}}$. It changes the RRAM device to the high resistance state (HRS). V_{on} and V_{off} are the threshold voltages of the RRAM device. The

✉ Xiaoxin Cui
cuixx@pku.edu.cn

[1] Key Lab of Integrated Microsystems, Peking University Shenzhen Graduate School, Shenzhen 518055, China

[2] Institute of Microelectronics, Peking University, Beijing 100871, China

resistance value is readout without modifying the resistance state because of the weak read voltage.

The RRAM device is recently found useful not only as the memory device, but also as the logic device to build logic circuits. The first RRAM-based logic family is the IMPLY logic (material implication) published in 2010 [1]. Afterward, various RRAM logic gate families have been proposed, such as CRS (complementary resistive switches) [18], MRL (memristor ratioed logic) [12], LTG (linear threshold gate) [5], MAGIC (memristor aided logic) [13], MPLA (memristive programmable logic array) [19], iMemComp (intelligent memory and computing) [8, 15], MAD (memristor as drivers) [6], Scouting logic [29], MOS-like (metal–oxide–semiconductor device-like) [25], and four-step logic [31]. Most of the early research works of RRAM-based logic circuit are about the gate-level circuit design and device-level improvements, while the focus has been shifted to the circuit architectures and design methods of RRAM-based logic circuits in recent years.

The computation in memory (CIM) architecture is of great concern, since it is regarded as one of the feasible methods to implement the non-von Neumann computer architecture. However, only few of the RRAM logic gate families can be implemented in the crossbar array. The discussions on the design methods of RRAM-based logic circuit are limited to few of the RRAM logic gate families. The IMPLY logic circuit is widely studied and has most fruitful results on the circuit architectures and circuit synthesis methods [3, 14, 16, 20, 26, 27]. Generally, the IMPLY logic circuit consumes relatively long computation time, due to the serialization of its operations. Some design practices of logic circuits with MAGIC logic gates have been reported [9, 22], and the synthesis methods have been developed [10, 23]. For some Boolean functions, the performance of the MAGIC logic circuit is relatively low because only the NOR and the NOT gates in the MAGIC logic gate family can be implemented in the crossbar array. The CRS logic is useful to implement IMPLY functionality and other arithmetic functions, and the CRS-CMOS circuits have been verified to be able to implement PLA (programmable logic array) functions [28, 30]. However, the CRS circuit needs special complementary cells, and it is destructive-read prone [30].

The motivation of this work is to systematically study the general design method of logic circuit based on the four-step RRAM logic gates. The design rules for the high-performance logic circuits are the aim of this work. The main contributions of this work include:

(1) We propose the organization rules of the single logic block and the multiple logic blocks based on the four-step RRAM logic gates. The design constraints of the single logic block are investigated based on its structure. The constraints are helpful for the designers to decide whether to adopt the single logic block to build the circuit for the specific required Boolean functions.
(2) We propose the general design methods of arbitrary logic circuit based on the four-step RRAM logic family. The proposed design method of combinational logic circuit results in high-speed circuit due to the parallelism of the logic block based on the four-step RRAM logic gates. With the proposed design method of

sequential logic circuit, the designers can build high-performance sequential logic circuit by further exploiting the parallelism between the logic blocks. To the best of the authors' knowledge, the design method of sequential logic circuit based on RRAM devices has rarely systematically studied before, and the proposed method in this work is a practical attempt on this topic.

(3) The four-step logic family realizes different Boolean functions in four working cycles if the logic functions conform the design constraints of the single logic block. We found out that this feature is useful to build high-performance pipelined circuits. Utilizing the advantage of the parallelism between the operations in different logic blocks, the design method is proposed to build high-performance pipelined circuits. The design practices of the N-bit ripple carry adder and the N-bit multiplier verify the advantage on the circuit performance come from the proposed design method.

The rest of the paper is organized as follows. The four-step RRAM-based logic gates are introduced in Sect. 2. The design methods of the combinational and the sequential logic circuits are described in Sect. 3 along with the synthesis results on the benchmark circuits and some designs examples. Section 4 discusses the design method of the pipelined circuits, and the high-performance N-bit rippled carrier adder (RCA) and N-bit multiplier circuits are taken as the design instances. Lastly, the conclusions are drawn in Sect. 5.

2 The Four-Step Basic Logic Gate

2.1 The Two-Input Basic Logic Gates

Figures 1a–c present the circuits of the four-step RRAM NOT, AND, OR gates, respectively [31]. The rightmost bit lines in Figs. 1a–c are the output bit lines, and the other bit lines are the input bit lines. The RRAM cells connected to the input bit lines are the working cells, and those connected to the output bit lines are the output cells. The resistance value R of the resistors in the word lines satisfies $R_{on} \ll R \ll R_{off}$, where R_{on} and R_{off} are the resistance values of the RRAM device at LRS and HRS, respectively. All the basic logic functions are implemented in four steps, i.e., initialization, input, computation and output. These logic gates have the unified input voltages in each step as listed in Table 1. The programming voltage V_p satisfies $\max\{|V_{reset}|, V_{set}\} < V_p < 2 \min\{|V_{reset}|, V_{set}\}$. The inputs of the proposed gates are presented by voltages. The input voltage is GND for logic 0 and is V_p for logic 1. The resistance state of each RRAM cell may or may not change depending on the voltage difference between its set and reset ends in the initialization, input and computation steps. The resistors in the word lines are useful to generate different voltages on the word lines, depending on the resistance states of the RRAM cells, for the same set of applied voltages to the circuits. The changes of resistance states of RRAM cells in each step are specified in the rightmost column of Table 1. The output voltage in the output

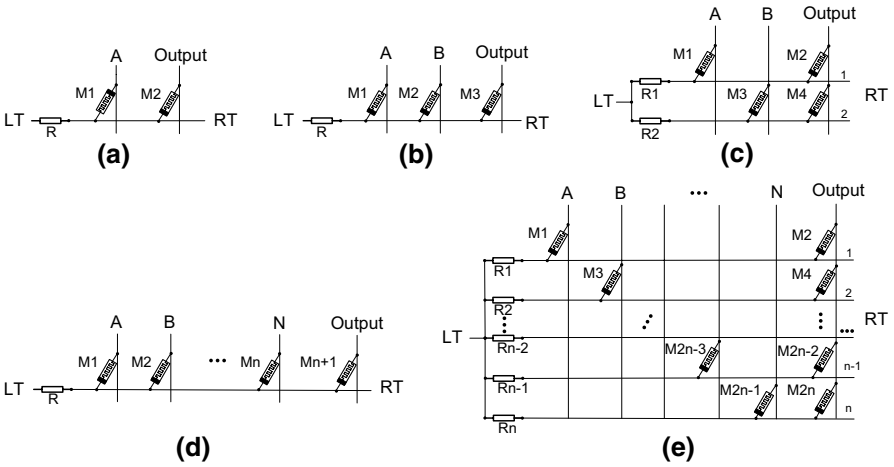

Fig. 1 The four-step logic gates **a** NOT gate **b** Two-input AND gate **c** Two-input OR gate **d** Multi-input AND gate **e** Multi-input OR gate

bit line is generated by the voltage division of the RRAM cells and resistors in the output step. The resistance states of RRAM cells do not change in the output step, so the circuit output is represented by the output voltage as well as the resistance states of the output cells. The four-step RRAM logic family [31] is logic complete and fully parallel in nature.

The functionalities of the two-input basic logic gates are simulated using the SPICE tool with RRAM model [17]. In the simulation, the resistance value of the resistor in the word lines is about 4×10^4 Ω; $V_{set}=1$ V; $V_p=1.2$ V; R_{HRS} and R_{LRS} of the RRAM cell are about 2.8×10^6 Ω and 5×10^3 Ω, respectively; $V_{on}=0.8$ V, and $V_{off}=-1.1$ V. The output waveforms of the NOT, AND and OR gates in the four steps are presented in Figs. 2a–c, respectively. Each step lasts about 100 ns. In the last step, the output voltage varies between approximately 0.485 V to 0.494 V for logic 1 (approaches to $V_{set}/2$) and approximately 0.039 V to 0.072 V for logic 0 (near to 0 V). The functionalities of the gates are verified.

2.2 The Multi-Input AND Gate and OR Gate

The parallelism is a key factor to enhance the circuit performance. In order to implement the complex combinational logic functions in parallel, the multi-input AND and OR gates are required. Figures 1d–e present the proposed multi-input AND and OR gates, respectively [31]. For the multi-input AND gate, one additional input bit line and one corresponding working cell are added in the basic two-input AND gate for an additional input variable in the AND Boolean function. For the multi-input OR gate, one additional independent word line and the corresponding working cell and output cell are added in the basic two-input OR gate if there is an additional input variable in the OR Boolean function. The proposed *n*-input AND

Table 1 The operation steps of the basic two-input logic gates

Basic logic gate	Steps	Input lines	Output line	Left terminal LT	Right terminal RT	Functionality
NOT Gate	① Initialization	V_p	V_p	GND	GND	Set M_1 to LRS; Reset M_2 to HRS.
	② Input	Input voltage	V_p	V_p	V_p	M_1 switches to HRS if input is logic 0.
	③ Computation	GND	$-V_{set}/2$	$V_{set}/2$	Floating	Presents the NOT function result by the resistance value of M_2.
	④ Output	GND	Output	$V_{set}/2$	Floating	The output voltage $\approx V_{set}/2$ for logic 1; the output voltage ≈ 0 for logic 0.
AND Gate	① Initialization	V_p	V_p	GND	GND	Reset all RRAM cells to HRS.
	② Input	Input voltage	V_p	V_p	V_p	Switch the corresponding working cells with input logic 0 to LRS.
	③ Computation	GND	$-V_{set}/2$	$V_{set}/2$	Floating	Presents the AND function result by the resistance value of output cell M_3.
	④Output	GND	Output	$V_{set}/2$	Floating	Voltage output.
OR Gate	① Initialization	V_p	V_p	GND	GND	Reset all the RRAM cells to HRS.
	② Input	Input voltage	V_p	V_p	V_p	Switch the corresponding working cells with input logic 0 to LRS.
	③ Computation	GND	$-V_{set}/2$	$V_{set}/2$	Floating	For each row, the output cell turns to LRS if the corresponding input cell is in HRS; while the output cell is in HRS if the corresponding input cell is in LRS.
	④ Output	GND	Output	$V_{set}/2$	Floating	Voltage output.

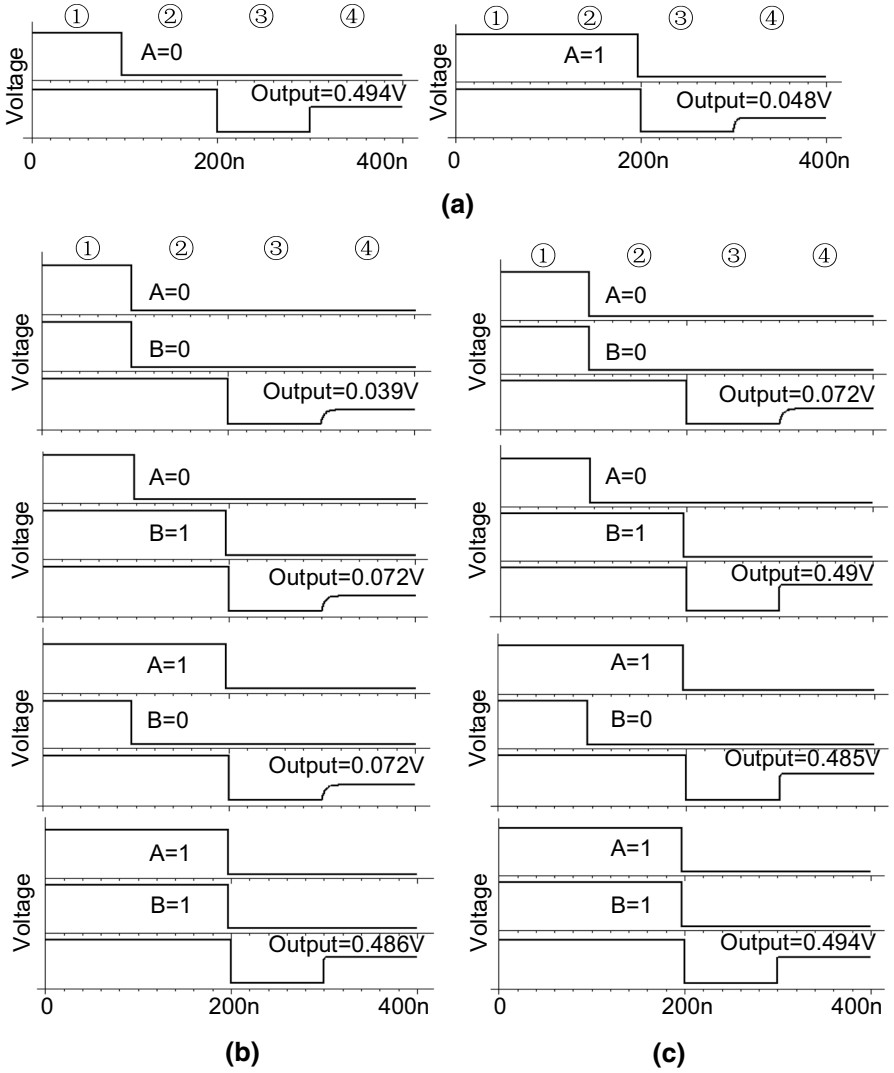

Fig. 2 The waveforms of the four-step logic gates **a** NOT gate **b** Two-input AND gate **c** Two-input OR gate (①, ②, ③ and ④ denote the four working steps of the gates)

gate is implemented with $n+1$ RRAM cells, and the n-input OR gate requires $2n$ RRAM cells. Both the n-input AND gate and the OR gate follow the operation steps in Table 1.

3 Design Method of Logic Circuits

3.1 Design of Logic Block

The logic functionalities are realized in the logic blocks based on the four-step RRAM logic gates. A full adder is taken as an example to illustrate the design of the logic block. The Boolean function of the full adder is defined in formulas (3.1) and (3.2).

$$S_i = A_i\bar{B}_i\bar{C}_{i-1} + \bar{A}_iB_i\bar{C}_{i-1} + \bar{A}_i\bar{B}_iC_{i-1} + A_iB_iC_{i-1} \tag{3.1}$$

$$C_i = A_iB_i + B_iC_{i-1} + A_iC_{i-1} \tag{3.2}$$

The conceptual full adder circuit with the four-step RRAM logic gate is shown in Fig. 3a. The sub-circuit in one row independently implements one AND cube in the Boolean function, and each OR function presents its result in the corresponding bit line of the output array on the right side. If the input variable is positive in the Boolean function, the set end of the corresponding working cell connects to the corresponding word line. The polarity of the working cell is inversed in the corresponding cross-point of the input array if the input variable is a negative one. In Fig. 3a, the sub-circuit with word line 1–4 realizes the Boolean function (3.1), and the sub-circuit with word line 5–7 realizes the function (3.2). The different logic gates in the array work simultaneously, and the full adder circuit outputs the computation results in four cycles following the operations defined in Table 1.

The RRAM cells with opposite polarities are difficult to fabricate in one array. This problem can be solved by assigning independent input bit line to both the positive and the negative variables; thus, the set end of all the RRAM cells in the array uniformly connects to the corresponding word line. This solution essentially moves the NOT function outside of the array. It works but destroys the completeness of the logic operations of the RRAM circuit. In order to retain the NOT

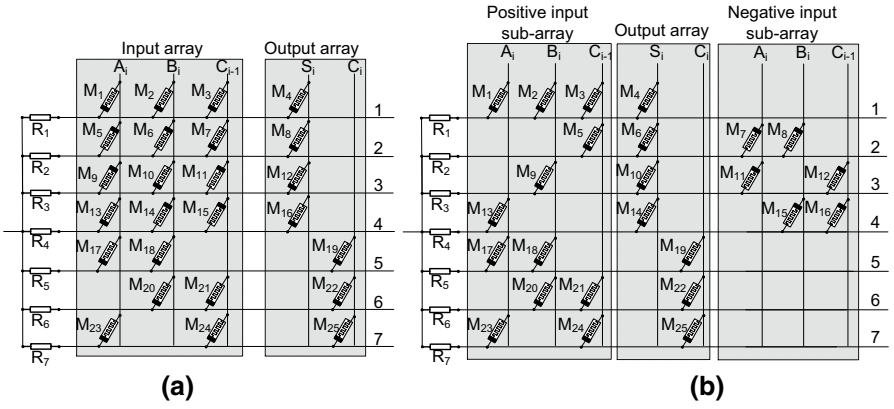

Fig. 3 The two-input full adder circuit **a** the conceptual design **b** the array implementable full adder circuit

function in the RRAM circuit, another solution is proposed as shown in Fig. 3b. All the RRAM cells in the input array in Fig. 3a are partitioned into two input sub-arrays. One input sub-array processes the positive input variables, and the polarities of RRAM cells in this sub-array are consistent with that of the output cells. The other input sub-array processes the negative input variables, and the polarities of RRAM cells in the sub-array are opposite to that of the output cells. The number of RRAM cells in Fig. 3b is equal to that in Fig. 3a, and the computation results are also obtained in four steps.

From logic point of view, any combinational circuit with arbitrary combinational logic function can be built in four steps, because the voltage across each RRAM cell of the proposed logic gate is exactly determined in each step, and each logic gate implements its logic function independently. The design method for the RRAM logic block is summarized based on the design practices described above.

(1) Convert the logic functions of the circuit into the SOP (Sum of Production) expressions, i.e., $\text{Output}_j = \sum(\text{Input}_{j1} \bullet \text{Input}_{j2} \bullet \ldots \bullet \text{Input}_{jk})$.
(2) For each logic function, add n RRAM rows if there are $n-1$ OR operations. Add the RRAM rows for all the logic functions. A resistor R is connected in series on the left end of each word line.
(3) Add one output bit line for each output variable.
(4) For each AND cube in the SOP expression w.r.t. the output variable$_j$, add one input bit line for each input variable. The working RRAM cells for one AND cube are placed in the same RRAM row. For each input variable in one AND cube, place one working cell at the cross-point of the corresponding input bit line and the word line related to variable$_j$. The set end of the RRAM cell connects to the corresponding word line for the positive input variable in the AND cube; and vice versa for the negative input variable. Add the output RRAM cells at the cross-points of the corresponding output bit line and the corresponding word lines. The polarities of the output RRAM cells are the same as that of the working cells related to the positive input variables. Construct the input bit lines, the working cells and the output cells for each AND cube in the logic functions.
(5) Merge the bit lines and the corresponding working cells if the different AND cubes share the same input variable.
(6) Divide the input array into two sub-arrays for the positive and the negative input variables, respectively. Connect the corresponding bit lines in the positive and the negative input sub-arrays if the same input variable is shared. The positive/negative input sub-arrays and the output array share the same group of word lines. Reverse the polarities of the working cells at the corresponding cross-points in the negative sub-array.

This design method is applied to some small-scaled MCNC benchmark circuits using the synthesis tool ABC [2], and the results are presented in Table 2. It can be seen from the table that the proposed design method implements all the logic functions in four working cycles. The generated circuits clearly outperform the counterparts with the cost of the additional RRAM cells.

Table 2 The synthesis results of Boolean functions with different design methods of RRAM logic circuits [3, 16, 20, 26, 27]

Function	Input	RRAM count/cycle count					
		[16]	[3]	[20]	[26]	[27]	Prop.
rd53f1	5	18/26	17/27	–/30	–/26	14/22	25/4
rd53f3	5	18/42	18/32	–/125	–/130	18/26	50/4
con1f1	7	13/15	31/18	–/35	–/17	12/14	15/4

3.2 Design Constraints of Single Logic Block

Actually, the size of the logic block is limited, because the output voltage is determined by the voltage division of the RRAM cells and the resistors in the output step.

For the n-input AND gate, if m inputs are logic 1, where $m \leq n$, the output voltage in the fourth step is calculated as (3.3).

$$\begin{aligned} V_{\text{Mout}} &\approx \frac{V_{\text{set}}}{2} \times \frac{R_{\text{HRS1}}//\cdots//R_{\text{HRS}m}//R_{\text{LRS1}}//\cdots//R_{\text{LRS}(n-m)}}{\left[R + \left(R_{\text{HRS1}}//\cdots//R_{\text{HRS}m}//R_{\text{LRS1}}//\cdots//R_{\text{LRS}(n-m)}\right)\right]} \\ &= \frac{V_{\text{set}}}{2} \times \frac{R_{\text{HRS}}R_{\text{LRS}}}{\left\{\left[(n-m)R_{\text{HRS}} + mR_{\text{LRS}}\right]R + R_{\text{HRS}}R_{\text{LRS}}\right\}} \end{aligned} \tag{3.3}$$

The maximum output voltage, $V_{\text{Mout}} \approx \frac{V_{\text{set}}}{2} \times \frac{R_{\text{HRS}}}{(nR+R_{\text{HRS}})}$, is obtained if $m=n$. If n is not large, V_{Mout} approaches $V_{\text{set}}/2$. However, if n is large enough, V_{Mout} decreases below V_{OH}, which is the lowest output voltage to present logic 1. Thus, the AND function fails. For the worst case, i.e., $m=n$, the simulations on the output voltage of the AND gate with various number of inputs are conducted. According to [4], the resistance variation of HRS is larger than that of LRS. The ranges of resistance values of HRS and LRS are set to $7.709\times10^5\ \Omega \sim 3.624\times10^6\ \Omega$, and $4.841\times10^3\ \Omega \sim 5.802\times10^3\ \Omega$, respectively. The simulation tool is the same as that in Sect. 2. The results are presented in Fig. 4a. The results show that if the V_{OH} is set to 0.4 V, the upper bound of the number of inputs is 15.

For the n-input OR gate, if m inputs are logic 1, where $m \leqslant n$, m working cells are in HRS and $(n-m)$ working cells are in LRS consequently, then the corresponding m and $(n-m)$ output cells are in LRS and HRS, respectively. The output voltage in the fourth step is determined by the voltage division between the output RRAM cells and is calculated as (3.4).

$$V_{\text{Mout}} \approx \frac{V_{\text{set}}}{2} \times \frac{\frac{R_{\text{HRS}}}{n-m}}{\left(\frac{R_{\text{HRS}}}{n-m} + \frac{R_{\text{LRS}}}{m}\right)} = \frac{V_{\text{set}}}{2} \times \frac{mR_{\text{HRS}}}{\left[mR_{\text{HRS}} + (n-m)R_{\text{LRS}}\right]} \tag{3.4}$$

The lowest output voltage for logic 1 is obtained if $m=1$. In this worst case, $V_{\text{Mout}} \approx \frac{V_{\text{set}}}{2} \times \frac{R_{\text{HRS}}}{[R_{\text{HRS}}+(n-1)R_{\text{LRS}}]}$. The output voltage decreases with the increase of n. If n is large enough, then V_{Mout} drops below V_{OH} and the OR gate fails. The

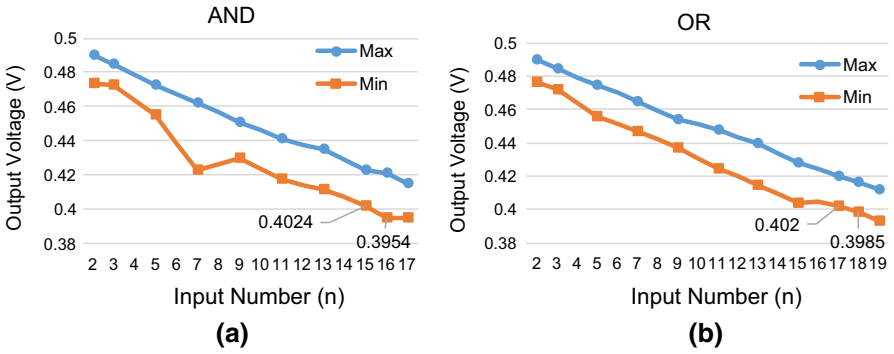

Fig. 4 The simulation results on the output voltage with various number of inputs **a** the AND gate **b** the OR gate

simulations on the output voltage of the OR gate with various number of inputs under the worst case are conducted. The results in Fig. 4b show that if the V_{OH} is set to 0.4 V, the upper bound of the number of inputs is 17.

For the SOP functions with various numbers of AND and OR operations, the simulations are conducted with various logic block sizes. V_{OH} is set to 0.4 V in the simulation. The maximum numbers of inputs for AND (p) and for OR (q) gates are varied. The cases in which the output voltage is not weaker than 0.4 V are listed in Table 3. The table shows that the logic block works if $p+q \leqslant 15$.

The analysis and the simulation results show that the logic block size has upper bounds. The design constraints of single logic block are summarized according to the simulation results as follows:

The number of inputs for the AND gates should not exceed 15, while the number of inputs for the OR gates should not exceed 17, and the sum of the number of inputs of AND and OR gates in the single logic block should not exceed 15, if V_{OH} is set to 0.4 V.

The design constraints are technology dependent. They differ, if the resistance value of the RRAM cell varies or V_{OH} is set to another value. For the specific condition combinations, the design constraints of the logic block need to be recalculated following the above-described method.

3.3 Design Method of Combinational Circuit

The multiple logic blocks are required to implement the arbitrary combinational logic circuit, if the logic functionalities are beyond the design constraints of single logic block. The data transfer between the logic blocks is essential in the multiple logic block circuits. However, the proposed RRAM circuit works based on the voltage division between the resistances, and the behavior of the next logic block is affected by the output resistance of the previous logic block. Therefore, a switch and a buffer are introduced between the interconnected input and output terminals to isolate the cascading logic blocks, as shown in Fig. 5. The output voltages of the logic block are approximately $V_{set}/2$ and 0 V for logic 1 and 0,

Table 3 The simulation results on the output voltage of the logic block with various sizes

Number of inputs for AND gates (p)	/	2	2	3	4	5	6	6	7	8
Number of inputs for OR gates (q)	17	16	15	14	13	12	11	10	9	9
Output voltage (V)	0.402	0.4011	0.405	0.4009	0.401	0.408	0.402	0.401	0.408	0.4013
Number of inputs for AND gates (p)	8	8	9	10	11	12	13	14	15	/
Number of inputs for OR gates (q)	8	7	6	6	5	4	3	2	/	/
Output voltage (V)	0.4084	0.4061	0.4063	0.406	0.4048	0.4045	0.4037	0.4007	0.4024	/

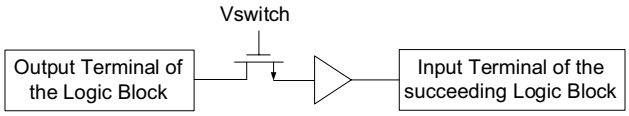

Fig. 5 The switch and the buffer between the logic block

respectively. The inserted MOS buffer makes it more reliable. The buffer outputs V_p or 0 V if its input voltage is $V_{set}/2$ or 0 V, respectively.

The design method for the arbitrary combinational logic functions is as follows.

(1) Express the original logic functions into the SOP form. If the numbers of variables do not exceed the design constraints of the single logic block size, design the single logic block circuit following the design method in Sect. 3.1.
(2) If the numbers of variables in the original logic functions exceed the design constraints of the single logic block, decompose the original functions into several sub-functions by introducing some intermediate variables. Each sub-function is expressed in the SOP form, and the numbers of variables in the sub-functions do not exceed the design constraints of single logic block.
(3) Design single logic block circuit for each sub-function following the design method in Sect. 3.1.

(4) Insert switch and buffer modules between the logic blocks with cascading relationships.

This design method is applied to some large-scale MCNC benchmark circuits, which are beyond the design constraints of the single logic block, and the results are listed in Table 4. Four more cycles are consumed if one additional logic block is inserted. However, it can be seen that all the circuits designed by the proposed method have better performance than that of the counterparts.

3.4 Design Method of Sequential Circuit

The sequential circuit consists of combinational circuits and flip flops. Fortunately, the logic block is able to store the intermediate states, and the arbitrary combinational functions can be realized by the design method in Sect. 3.3.

In some cases, the delay function is required for the correct timing of the circuits. In the proposed four-step based RRAM logic circuit, the single cycle is the minimum delay unit, since each step of the proposed RRAM logic block consumes one cycle. The single cycle delay can be generated by repeating the output step of the logic block, since the operation does not change the resistance state of RRAM cells. In order to implement a k-cycle delay for a logic block, additional k output steps are required, and the switch is enabled at the $(4+k)$th cycle.

The systematic design method of sequential logic circuit based on RRAM devices is rarely reported in the literature. Only some specific sequential circuits, such as LFSR (Linear Feedback Shift Register), have been designed based on RRAM devices [4, 24]. The four-step RRAM logic circuit can be used to design the sequential logic circuit with higher performance due to the advantage of parallelism. The proposed design method for the sequential logic circuit is as follows:

(1) Draw the state transition diagram of the sequential logic functions and list the state transition equations with SOP form.

Table 4 The synthesis results on some large-scale benchmark circuits [3, 16, 20, 26, 27]

Function	Input	RRAM count/cycle count					
		[16]	[3]	[20]	[26]	[27]	Prop.
rd53f2	5	32/42	30/57	–/56	–/62	20/37	98/8
rd73f1	7	116/223	140/238	–/142	–/404	68/137	304/24
rd73f2	7	26/97	253/46	–/257	–/–	25/37	528/36
rd73f3	7	59/118	210/104	–/210	–/–	36/70	181/16
sao2f1	10	59/87	33/102	–/67	–/94	34/61	102/8
sao2f2	10	63/104	31/112	–/49	–/208	41/77	228/20
sao2f3	10	133/199	22/380	–/20	–/130	89/188	111/12
sao2f4	10	128/202	33/252	–/34	–/–	77/165	130/12

(2) According to the state transition equations, build single or multiple logic blocks following the design method in Sect. 3.3. Insert delays for the correct timing of the circuit if needed. Name the circuit as module A.
(3) Duplicate the module A, and name it as module B. The module A processes the current states, and the module B processes the next states.
(4) Connect the pseudo-primary outputs (PPOs) of module A to the corresponding pseudo-primary inputs (PPIs) of module B, connect the PPOs of module B to the corresponding PPIs of module A, and add the switch and buffer to each PPO between the modules. Insert delays between modules A and B if necessary.

The LFSR is taken as the example circuit. A four-stage LFSR with a characteristic polynomial $g(x)=x^4+x^3+1$ is shown in Fig. 6a. The state transition equations are listed in formula (3.5).

$$\begin{aligned} D_0^{n+1} &= D_0^n \oplus D_3^n \\ D_1^{n+1} &= D_0^n \\ D_2^{n+1} &= D_1^n \\ D_3^{n+1} &= D_2^n \end{aligned} \tag{3.5}$$

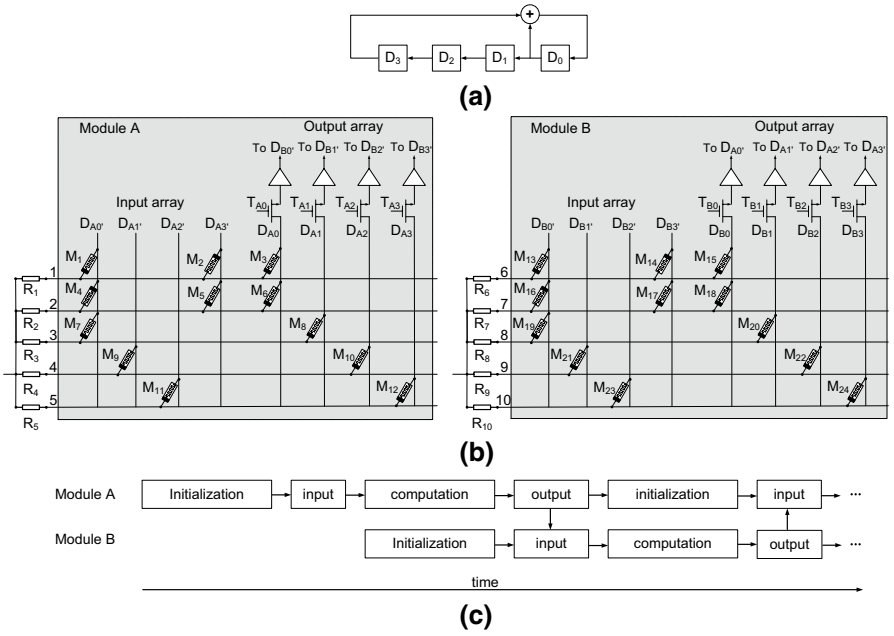

Fig. 6 Design of a four-stage LFSR **a** a conceptual four-stage LFSR **b** the proposed LFSR circuit **c** the working processes of modules A and B of the proposed LFSR circuit

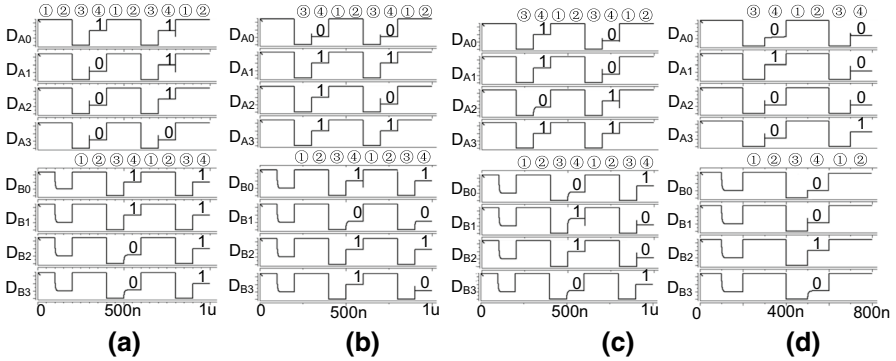

Fig. 7 The output waveforms of the proposed four-stage LFSR (①, ②, ③ and ④ denote the four working steps of the gates)

The functions in (3.5) are realized by the circuit in Fig. 6b, which consists of two interconnected duplicate circuit modules. The working operations of modules A and B are presented in Fig. 6c. The period of the LFSR is 15. The circuit is seeded with $D_0D_1D_2D_3 = 0001$. The output of each module is presented in Fig. 7, where $D_{A0} \sim D_{A3}$ are the outputs of module A, and $D_{B0} \sim D_{B3}$ are the outputs of module B. Figures 7a–d present the output of the LFSR in one period. Figure 7a illustrates the output waveforms in the first 10 working cycles, i.e., 1 μs. It can be seen that, the module A outputs its value 1000 at 400 ns and that output data is fed into module B at the same working cycle. Then the module B outputs its value 1100 at 200 ns later and that output is the input of module A, so the module A outputs its value 1110 at 200 ns later. These steps are consistent with that shown in Fig. 6c. This process continues until the seeding vector 0001 is generated as the output of module A in Fig. 7d. The functional waveforms verify the functionality of the LFSR circuit based on the proposed method.

It can be observed that each state transition consumes only two working cycles in average. As shown in Table 5, the proposed LFSR circuit has better performance than the counterpart in [24]. The proposed LSFR circuit only needs 24 RRAM cells, 10 resistors, and 40 MOSFET devices, it consumes less area compared with the

Table 5 Comparison of the four-stage LFSR circuits [21, 24]

Circuit	Average cycles for one state transition	Area
IMPLY LFSR [24]	55	8 RRAM cells + 1 resistor
Hybrid Memristor-CMOS LFSR [21]	1	64 RRAM cells + 128 MOSFET devices
Proposed LFSR	2	24 RRAM cells + 10 resistors + 40 MOSFET devices

hybrid RRAM-MOS logic gates based scheme in [21]. The average cycles for each state transition are constantly two cycles, if the proposed method is applied to the other LFSR functions with various characteristic polynomials. The designs of the LFSR circuits obtain stable and high-performance results.

4 Design of Pipelined Circuits

4.1 Design Method of Pipelined Circuit

The pipelining that utilizes the parallelism between the modules is one of the key techniques to increase the data throughput of the circuit. The four-step RRAM logic gates implement different combinational logic functions in four steps if the numbers of variables do not exceed the constraints discussed in Sect. 3.2. This feature is suitable for the organization of the pipelined architecture. The pipelined circuit has high data throughput if most of its modules have equal latency. Even if the latency of different modules is not equal, we can make the latency of modules equal by adding extra delays, as described in Sect. 3.4. It improves the efficiency of the pipelined circuit.

The design method of the pipelined circuit is proposed as follows.

(1) Partition the circuit into multiple modules according to its functionality.
(2) Implement each module into single or multiple logic blocks with the four-step RRAM logic gates, and analyze the timing of each input and output.
(3) Overlap or cascade the modules in the time-spacing diagram according to the data dependencies. Convert the pipeline into circuit and add the switches and buffers between the cascading logic blocks.
(4) Insert extra waiting cycles to some modules to satisfy the timing requirement, according to the timing of each module.

4.2 Case Study: The Pipelined N-Bit RCA

The pipelined ripple carry adder (RCA) is taken as an example circuit that consists of homogenous modules. The conceptual scheme of the pipelined N-bit RCA circuit consists of identical full adder modules, as illustrated in Fig. 8a. A_{ij}, B_{ij}, C_{ij}, and S_{ij} are the augend, addend, carry output and sum output of the jth full adder module for the ith input data, respectively. The circuit of each full adder module is same as Fig. 3b. Each module works simultaneously and consumes four cycles.

A compact pipelined circuit is designed, shown in Fig. 8b, since each module of the N-bit RCA is identical. It only uses two full adder modules, one computes the odd bits, and the other computes the even bits. Similar as the working process shown in Fig. 6c, the initialization and the input operations of the full adder 1 are executed firstly. Then the computation of full adder 1 and the initialization of full adder 2 are executed in the same working cycle. The output data of the full adder 1 is fed as input data to the full adder 2 in the next working cycle. The full adder 1

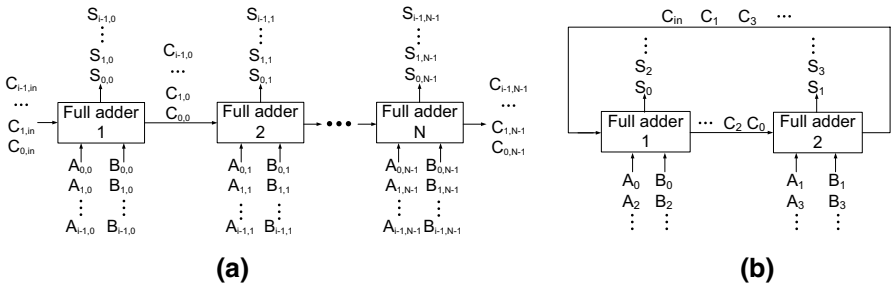

Fig. 8 The pipelined N-bit RCA **a** conceptual circuit **b** compact circuit

always leads the full adder 2 two working cycles. The input and the output steps of each full adder operate simultaneously. Each module works in parallel. There are no idle cycles from the third working cycle till the end. For the N-bit RCA, the compact pipelined RCA circuit output the final result in $2N+2$ working cycles, and it consumes 50 RRAM cells, 14 resistors and 10 MOSFET devices.

4.3 Case Study: The Pipelined N-Bit Multiplier

A 4×4 multiplier circuit is used to demonstrate the pipelined circuit consisting of different modules and non-simultaneous inputs. It consists of full adder (FA) modules, half adder (HA) modules and AND gates, as illustrated in Fig. 9a. The FA module is designed as Fig. 3b in Sect. 3.1, and the HA module is designed as Fig. 9b. In Fig. 9a, $Y_0 \sim Y_3$ and $P_0 \sim P_3$ are the inputs and the outputs of the multiplier, respectively, and $X_0 \sim X_3$ are the inputs of the circuit modules. The cycle number of the logic block output for the first input data of each circuit module is presented on the paths, and that for the second input data is presented in the brackets. The outputs of the successor logic blocks are two cycles later than the current logic blocks in the same row, because the input steps of the successor logic blocks execute simultaneously with the output steps of the current logic blocks, similar as the case in Fig. 6c. Some modules require waiting cycles for the synchronization of the pipelined circuit. The output of the AND gate is two cycles later than the right neighboring AND gate in the same row. And for the leftmost HA and FA modules, the carry input is two cycles later than the sum input. Thus, for the 4×4 pipelined multiplier circuit, the input data is fed into $Y_0 \sim Y_3$ every ten cycles. This multiplier circuit outputs the first result at the 20th cycle and then outputs one result in every ten cycles till the end.

For a $N\times N$ multiplier circuit with the similar structure, it outputs the first result at the $(6N-4)$th cycle and outputs the following results every $2N+2$ cycles. Table 6 compares the proposed pipelined multiplier circuit with the previously published RRAM-based multipliers. The proposed scheme has significant advantage on the performance with additional cost of area. Table 6 shows that the four-step RRAM-based logic block is suitable for the design of the high-performance pipelined circuit.

The number of RRAM cells can be further reduced. It can be observed from Fig. 9a that the modules on the right side are idle after they finish their work. If

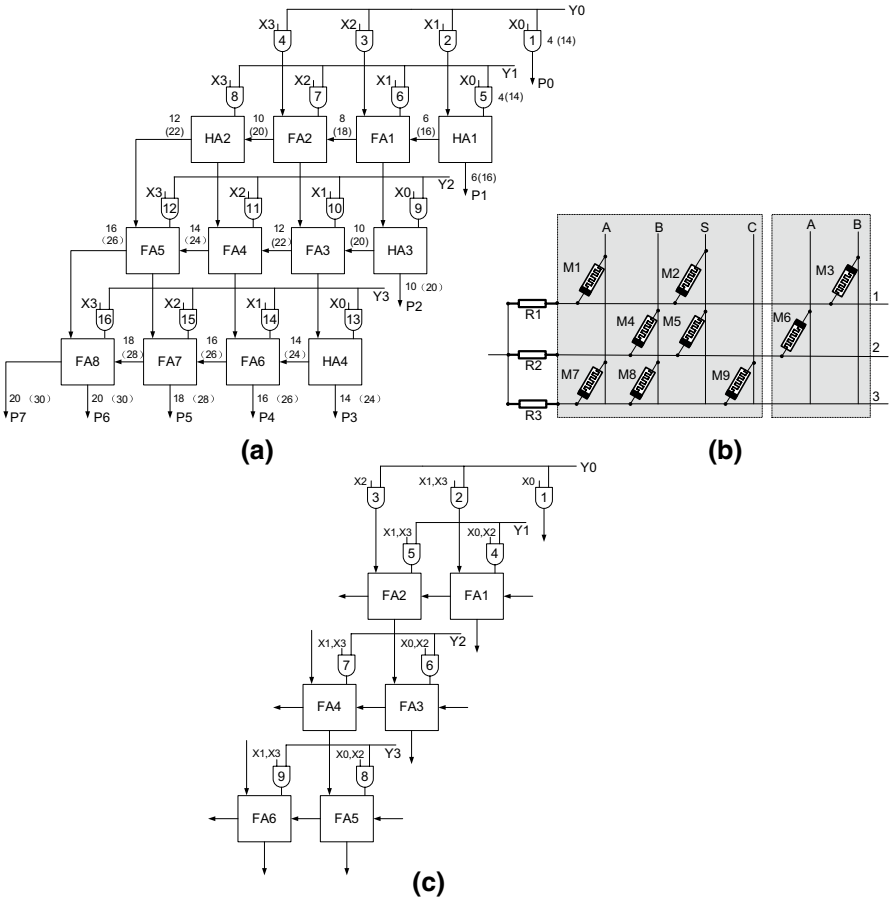

Fig. 9 **a** The 4×4 multiplier circuit **b** the two-input half adder **c** the column-folded 4×4 multiplier circuit

these modules are reused, the modules on the left side can be removed. The compacted circuit is presented in Fig. 9c. This scheme consumes only 177 RRAM cells, 51 resistors and 90 MOSFET devices. This column folding technique can also be applied to $N \times N$ multiplier circuit to further reduce the area.

5 Conclusion

This paper proposes the circuit design method based on the four-step RRAM-based logic gates. The proposed method is applicable for the arbitrary logic circuit design, including both the combinational and the sequential circuits. The advantages of the

Table 6 Comparison of different RRAM-based multiplier circuits [7, 11]

	Proposed multiplier	IMPLY [7]	MAD [7]	MAGIC [11]
Latency (cycles)	$6N-4$	$2N^2+21N$	N^2+N	$13N^2-14N+6$
Average cycles per output result (cycles)	$2N+2$	$24N$	$4N$	–
Area (number of RRAM cells, resistors and MOSFET devices)	$28N^2-41N$ RRAM cells $15N^2-20N$ MOSFET devices $8N^2-11N$ resistors	$7N+1$ RRAM cells $8N-1$ MOSFET devices $N+1$ resistors	$5N$ RRAM cells $15N$ MOSFET devices $4N+2$ resistors	20 N-5 RRAM cells

four-step RRAM-based logic circuit include: (1) the logic blocks can easily be cascaded to implement large-scale logic functionalities; (2) all the logic gates work in parallel, producing high-performance circuits; (3) the computation in memory is realized, since the circuit is organized into RRAM array. The design practices and synthesis results show that the proposed design method is able to generate the high-performance circuit schemes. Furthermore, due to the uniform working speed of different logic blocks, the four-step RRAM-based logic circuit is suitable for building the pipelined circuits. The design practices show that the pipelined circuits effectively enhance the performance and data throughput of the circuit.

Acknowledgement This work was supported by Shenzhen Science and Technology Innovation Committee under Grant No. JCYJ20170412150411676, National Natural Science Foundation of China under Grant No. U1613215, and TSV 3D Integrated Micro/Nano system Lab under Grant No. ZDSYS201802061805105.

References

1. J. Borghetti, G.S. Snider, 'Memristive' switch enable 'stateful' logic operations via material implication. Nature **464**, 873–876 (2010)
2. Berkeley Logic Synthesis and Verification Group, ABC: a system for sequential synthesis and verification. http://www.eecs.berkeley.edu/~alanmi/abc/. Accessed 8 Nov 2019
3. J. Bürger, C. Teuscher, M. Perkowski, Digital logic synthesis for memristors, in *Proceedings of the Reed-Muller Workshop* (2013), pp. 1–10
4. A. Chen, Comprehensive assessment of RRAM-based PUF for hardware security applications, in *IEEE International Electron Devices Meeting, Washington DC USA* (2015)
5. L. Gao, F. Alibart, Programmable CMOS/memristor threshold logic. IEEE Trans. Nanotechnol. **12**, 115–119 (2013)
6. L. Guckert, E.E. Swartzlander, MAD gates- memristor logic design using driver circuitry. IEEE Trans. Circuits Syst. II Express Briefs. **64**, 171–175 (2017)
7. L. Guckert, E.E. Swartzlander, Optimized memristor-based multipliers. IEEE Trans. Circuits Syst. I Regul. Pap. **64**, 373–385 (2017)
8. P. Huang, J. Kang, Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits. Adv. Mater. **28**, 9758–9764 (2016)
9. R.B. Hur, S. Kvatinsky, Memory processing unit for in-memory processing, *in IEEE/ACM International Symposium on Nanoscale Architecture (NANOARCH)* (Beijing, 2016), pp. 171–172

10. R.B. Hur, N. Wald, Simple magic: synthesis and in-memory mapping of logic execution for memristor-aided logic, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (Irvine, 2017), pp. 225–232
11. A. Haj-Ali, R. Ben-Hur, Efficient algorithms for in-memory fixed point multiplication using MAGIC, in *IEEE International Symposium on Circuits and Systems (ISCAS)*, (Florence, 2018), pp. 1–5
12. S. Kvatinsky, N. Wald, MRL-memristorratioed logic*, in IEEE International Workshop on Cellular Nanascale Networks and Their Applications* (Turin, 2012), pp. 1–6
13. S. Kvatinsky, D. Belousov, MAGIC-memristor-aided logic. IEEE Trans. Circuits Syst II Express Briefs. **61**, 895–899 (2014)
14. S. Kvatinsky, G. Satat, Memristor-based material implication (IMPLY) logic: design principles and methodologies. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **22**, 2054–2066 (2014)
15. H. Li, B. Gao, Z. Chen, A learnable parallel processing architecture towards unity of memory and computing. Scientific Rep. **5**(1), 13330 (2015)
16. F. Lalchhandama, B. Gopal, An improved approach for the synthesis of Boolean functions using memristor based IMPLY and INVERSE-IMPLY gates*, in IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (Pittsburgh, 2016), pp. 319–324
17. H. Li, P. Huang, A SPICE model of resistive random access memory for large-scale memory array simulation. IEEE Electron Device Lett. **35**, 211–213 (2014)
18. R. Rosezin, E. Linn, Crossbar logic using bipolar and complementary resistive switches. IEEE Electron Device Lett. **32**, 710–712 (2011)
19. G.S. Rose, J. Rajendran, Leveraging memristive systems in the construction of digital logic circuits. Proc. IEEE. **100**, 2033–2049 (2012)
20. Ragjivamsjo, A., Perkowski, M., Logic synthesis and a generalized notation for memristor-realized material implication gates, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, (San Jose, 2014), pp. 470–477
21. Sasi, A., Amirsoleimani, A., Hybrid memristor-CMOS based linear feedback shift register design, in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, (Batumi, 2017), pp. 62–65
22. P.L. Thangkhiew, R. Gharpinde, Area efficient implementation of ripple carry adder using memristorcorssbar array*, in IEEE International Symposium on Design & Test (IDT)* (Hammanet, 2016), pp. 142–147
23. P.L. Thangkhiew, R. Gharpinde, Efficient mapping of Boolean functions to memristor crossbar using MAGIC NOR gates. IEEE Trans. Circuits Syst I Regul. Pap. **65**, 2466–2476 (2018)
24. M. Teimoory, A. Amirsoleimani, Memristor-based linear feedback shift register based on material implication logic, in *IEEE European Conference on Circuit Theory and Design*, (Trondheim, 2015), pp. 1–4
25. I. Vourkas, G.C. Sirakoulis, A novel design and modeling paradigm for memristor-based crossbar circuits. IEEE Trans. Nanotechnol. **11**, 1151–1159 (2012)
26. X. Wang, R. Tan, Synthesis of memristive circuits based on stateful IMPLY gates using an evolutionary algorithm with a correction function*, in IEEE/ACM International Symposium on Nanoscale Architecture (NANOARCH)* (Beijing, 2016), pp. 97–102
27. H.P. Wang, C.C. Lin, On synthesizing memristor-based logic circuits with minimal operational pulses. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **26**, 2842–2852 (2018)
28. X. Wang, S. Li, Configurable logic operations using hybrid CRS-CMOS cells. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **26**, 2641–2647 (2018)
29. L. Xie, H.A. Du, Scouting logic: a novel memristor-based logic design for resistive computing*, in IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (Bochum, 2017), pp. 176–181
30. Y. Yang, J. Mathew, Complementary resistive switch-based arithmetic logic implementations using material implication. IEEE Trans. Nanotechnol. **15**, 94–108 (2016)
31. K. Zhang, X. Cui, A design of high performance full adder with memristors, in *IEEE International Conference on ASIC (ASICON)*, (Guiyang, 2017), pp. 746–749