The Synthesis Method of Logic Circuits Based on the NMOS-Like RRAM Gates

XIAOLE CUI\textsuperscript{1,}, YE MA\textsuperscript{1,}, FENG WEI\textsuperscript{1,}, AND XIAOXIN CUI\textsuperscript{2,} (Member, IEEE)

\textsuperscript{1}Key Lab of Integrated Microsystems, Peking University Shenzhen Graduate School, Shenzhen 518055, China
\textsuperscript{2}Institute of Microelectronics, Peking University, Beijing 100871, China

Corresponding author: Xiaoxin Cui (cuixx@pku.edu.cn)

This work was supported in part by the National Key Research and Development Project under Grant 2018YFB2202600, in part by the Shenzhen Science and Technology Innovation Committee under Grant JCYJ20170412150411676, in part by the National Natural Science Foundation of China under Grant U1613215, and in part by the TSV 3D Integrated Micro/Nano system Lab under Grant ZDSYS201802061805105.

ABSTRACT The synthesis method of logic circuits based on the RRAM (Resistive Random Access Memory) devices is of great concern in recent years. Inspired by the CMOS-like RRAM based logic gates, this work proposes a NMOS-like RRAM gate family. The advantages of the proposed NMOS-like RRAM gates include: (1) all the gate circuits are array-implementable; (2) the gate family is logic complete; (3) the NOR, AND and NOT gates only consume single cycle respectively in the computation phase; (4) the gate circuits save half number of RRAM devices compared with the CMOS-like RRAM based counterparts. Furthermore, the synthesis method of logic circuits based on the NMOS-like RRAM gates is proposed and discussed. The single-cycle NMOS-like RRAM gates are utilized in priority under the constraints of the logic block. The features of the proposed synthesis method are: (1) it generates the high-performance logic circuits because the NMOS-like RRAM based gates work in parallel; (2) the logic block based on the proposed NMOS-like gates can be realized in the RRAM array; (3) the large-scale logic functions can be implemented by cascading the logic blocks. The in-array full-adder circuit generated by the proposed synthesis method only consumes three cycles in minimum, which outperforms the previous RRAM based counterpart. The synthesis results on the benchmark circuits show that the proposed synthesis method is able to generate the high-performance circuit in RRAM arrays for arbitrary logic functions.

INDEX TERMS RRAM array, NMOS-like RRAM gates, computation in memory, synthesis method.

I. INTRODUCTION

The computation in memory (CIM) is an emerging circuit style. The design method of logic circuits based on the memory devices has attracted great attention from the academia and industries, because it is a feasible way to implement the Non-von Neumann computer architecture. Meanwhile, the memory technologies progress rapidly in decade. The memristive memory are developed as the important memory products in the future, and the RRAM (Resistive Random Access Memory) [1] is one of the emerging memristive devices. The typical RRAM device is made of a metal oxide switching layer and two electrodes. It represents logic states with two stable resistance states, i.e. the low resistance state (LRS) and the high resistance state (HRS). The ideal I-V curve of the RRAM device is presented in Fig.1.

The SET operation is conducted if a positive voltage $V_{\text{set}}$, which is stronger than the threshold voltage $V_{\text{on}}$, is applied on the device. It turns the RRAM device to the LRS. The RESET operation changes the RRAM device into the HRS. It is conducted by applying a negative voltage $V_{\text{reset}}$, which is lower than the threshold voltage $V_{\text{off}}$. The read operation does not destroy the resistance state because of the weak readout voltage. The RRAM is regarded as an important competitor of the non-volatile memory devices because of the advantages such as the small device size, high performance, low static power, high integration density, and compatibility with the CMOS technology. However, the RRAM based design method of logic circuits is still an open issue.

Recently, researchers found out that the RRAM device is also a useful logic device. Several array-implementable RRAM based logic gate families were proposed [2] for the CIM applications, e.g. IMPLY (material implication) [3], CRS (complementary resistive switches) [4], MAGIC

54466 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
FIGURE 1. The ideal I-V curve of the RRAM device.

FIGURE 2. The CMOS-like RRAM gates [18].

The rest of the paper is organized as follows. The NMOS-like RRAM logic gates are proposed in section II. Then the NMOS-like RRAM gates are applied to the design of logic circuits in section III. Section IV proposes the synthesis method of the array-implementable logic circuits, and presents the synthesis results on some benchmark circuits. Finally, conclusions are drawn in section V.

II. THE NMOS-LIKE RRAM GATES

A. THE BASIC NMOS-LIKE RRAM GATES

The proposed basic NMOS-like RRAM logic gates are presented in Figs. 3 (a)-(e), respectively, wherein the set end of the RRAM device is marked with the black line. The resistance value $R$ of the resistor satisfies $R_{\text{on}} \ll R \ll R_{\text{off}}$, where $R_{\text{on}}$ and $R_{\text{off}}$ are the resistance values of RRAM device in the LRS and HRS, respectively. It is named as the “NMOS-like gate”, because it only retains the pull-down network of the CMOS-like counterpart.

The gate circuit works with two phases.

Phase 1. Input. It initializes the RRAM cells by the input voltages. If the input is logic 1, the top end of the corresponding RRAM cell is connected to GND, and the voltage $V_p$ is applied to the bottom end of the corresponding RRAM cell, where $V_p > \max \{|V_{\text{off}}|, V_{\text{on}}\}$. If the input is logic 0, the top end of the corresponding cell is also connected to GND, whereas the voltage $-V_p$ is applied to the bottom end of the corresponding cell. The RRAM cells in the same row can be initialized simultaneously.

However, the RRAM devices are usually organized into arrays. The CMOS-like RRAM gates as shown in Fig.2 are not array-implementable. It means that they are not able to be applied to the computation in memory. The motivation of this work is to overcome this drawback, and keep the high-performance characteristic of the CMOS-like RRAM gates. We propose the NMOS-like RRAM based gates by pruning the pull-up network of the CMOS-like RRAM based counterparts. Furthermore, this work studies and discusses the synthesis methods of logic circuits based on the proposed NMOS-like RRAM gates.

The gate circuit works with two phases.

Phase 1. Input. It initializes the RRAM cells by the input voltages. If the input is logic 1, the top end of the corresponding RRAM cell is connected to GND, and the voltage $V_p$ is applied to the bottom end of the corresponding RRAM cell, where $V_p > \max \{|V_{\text{off}}|, V_{\text{on}}\}$. If the input is logic 0, the top end of the corresponding cell is also connected to GND, whereas the voltage $-V_p$ is applied to the bottom end of the corresponding cell. The RRAM cells in the same row can be initialized simultaneously.

Phase 2. Computation. The voltage $V_{\text{in}}$ is applied to the left terminal LT, where $V_{\text{in}} \leq \min \{|V_{\text{off}}|, V_{\text{on}}\}$, and the bottom end of the corresponding column lines is grounded. The other nodes are floating during this phase. The output of the gate is generated by the voltage division between the RRAM devices and the resistor, and the RT is the output terminal. The gates output logic 1 with a high voltage, and logic 0 with a low voltage.

In the NOT gate in Fig.3 (a), the set end of the RRAM cell is the bottom end. The RRAM cell is in LRS if the input is
FIGURE 3. The proposed basic NMOS-like RRAM gates (a) NOT gate (b) two-input NOR gate (c) two-input NAND gate (d) two-input AND gate (e) two-input OR gate.

logic 1, and it is in HRS if the input is logic 0. The simulations on the output voltage are conducted with the SPICE tool and the RRAM model in [19]. The equivalent circuit model of the RRAM device is presented in Fig. 4 [19], where $R_L$ is the parasitic contact resistance of electrodes, $R_H$ is the parasitic resistance between electrodes, $C_P$ is the parasitic capacitance between electrodes, $R_S$ is resistance of the switching layer of the RRAM device. $R_S = R_{on}$ if the RRAM device is in LRS, and $R_S = R_{off}$ if the RRAM device is in HRS. The simulation conditions include: $V_{pp} = 1.4V$, $V_{in} = 0.5V$, $V_{on} = 0.8V$, $V_{off} = -1.1V$, $R = 7 \times 10^4 \Omega$, $R_L = 2052$, $R_H = 200M\Omega$, $C_P = 20fF$, the $R_{off}$ and $R_{on}$ is about $2.8 \times 10^6 \Omega$ and $5 \times 10^5 \Omega$, respectively. The waveform in Fig.3 (a) presents that, the output voltage is about 0.442V, which approaches $V_{in}$, if the input is logic 0, the rise time of the output voltage is about 352ps. The output voltage is about 0.034V, which is close to GND, if the input is logic 1. The NOT function is realized correctly, and the NOT gate consumes only one input cycle and one computation cycle.

The polarities of the RRAM cells in the NOR gate and the NAND gate are the same as that in the NOT gate, as presented in Fig.3 (b) and Fig.3 (c), respectively. The NOR gate in Fig.3 (b) also consumes two cycles, because the two RRAM cells in parallel are initialized in the same cycle. Whereas, the NAND gate in Fig.3 (c) consumes three cycles, because the RRAM cells in series have to be initialized in different input steps, and each step consumes single cycle. The simulation waveforms in Fig.3 (b) and Fig.3 (c) show that the logic functions of the gates are realized correctly, and the maximum rise times of the output voltages in the NOR gate and NAND gate are 300ps and 170ps, respectively.

The NMOS-like NOT, NOR and NAND gates are logic complete. Conceptually, the AND gate can be implemented by cascading a NAND gate and a NOT gate, similar as the CMOS counterpart. However, the voltage loss on the gate circuit is not negligible. It requires a buffer between the cascading gates for the voltage recovery purpose. And a switch is required to control the timing of the two gates. The AND gate consumes four working cycles, because the computation step of the AND gate and the input step of the NOT gate are executed in the same cycle. This CMOS-like solution has disadvantages on both the performance and area consumption. The similar drawbacks exist in the OR gate, if it is implemented by cascading a NOR gate and a NOT gate with the switch and buffer.

Fig.3 (d) and Fig.3 (e) present the proposed NMOS-like AND gate and OR gate, respectively. These gates use the RRAM cells with the reversed polarities with respect to that in the NOT, NOR, and NAND gates. The RRAM cells are connected in parallel for the AND gate, and in series for the OR gate. The RRAM cell is in LRS if the corresponding input is logic 0, and it is in HRS if the corresponding input is logic 1, which are opposite to that in the NOT, NOR, and NAND gates. The simulations are conducted with the same tools and simulation conditions above. The waveforms in Fig.3 (d) and Fig.3 (e) show that the functions of these two gates are correct, and the maximum rise times of the output voltages in the AND gate and OR gate are 286ps and 818ps, respectively.

Table 1 compares the basic NMOS-like gates with the previous RRAM based counterparts. The proposed NMOS-
like gates are MOS-less RRAM gates. They are much faster than the IMPLY gates, and consume less RRAM cells than the MAGIC gates and iMemComp gates. The proposed gates are logic complete, but the MPLA is not. Table 1 shows that the NMOS-like RRAM gate family is a competitive candidate in the array-implementable RRAM gate families.

### B. THE MULTI-INPUT NMOS-LIKE GATES

The multi-input NMOS-like gates are implementable by extending the input cells of the two-input counterparts. However, the number of inputs is limited because the output of the gate circuit is obtained by the voltage division between the RRAM devices and the resistor.

For the m-input AND gate, as shown in Fig.5 (a), the output voltage drops with the increase of m. When the expected gate output is logic 1, the worst case occurs if all of the m RRAM cells are in HRS. In such a case, the condition (2-1) should be satisfied, where $V_{OH}$ is the lowest voltage to present logic 1. If $V_{OH} = 0.4V$, the constraint $m \leq 11$ is deduced from the same simulation conditions above. The same constraint is obtained for the m-input NOR gate, since it has the same topological structure as that of the AND gate, but with the opposite polarities of RRAM cells.

\[
m \leq \frac{(V_{in} - V_{OH}) \cdot R_{off}}{V_{OH} \cdot R} = \frac{V_{OH} \cdot R}{V_{OH} \cdot R}
\]

\[
n \leq \frac{(V_{in} - V_{OL}) \cdot R_{on}}{V_{OL} \cdot R} = \frac{V_{OL} \cdot R}{V_{OL} \cdot R}
\]

\[\text{(2-1)}\]

\[\text{(2-2)}\]

For the n-input OR gate presented in Fig.5 (b), the output voltage rises with the increase of n. When the expected output is logic 0, the worst case occurs if all of the RRAM cells are in LRS. The function of the gate is correct if the condition (2-2) is satisfied, where $V_{OL}$ is the highest voltage to present logic 0. The constraint $n \leq 6$ is obtained, if $V_{OL} = 0.15V$. The n-input NAND gate shares the same constraint of input count with the n-input OR gate.

### FIGURE 5. The multi-input gates (a) The m-input AND gate (b) The n-input OR gate.
The fluctuation of the voltage $V_{in}$ affects the output voltage of the gates. For the 11-input AND and NOR gates, the condition $V_{in} \geq 0.495V$ should be satisfied if $V_{OH} = 0.4V$, and for the 6-input OR and NAND gates, the condition $V_{in} \leq 0.51V$ is required if $V_{OL} = 0.15V$. The range of $V_{in}$ is between $[0.495V, 0.51V]$ accordingly for the worst cases of the proposed gates.

### III. THE SYNTHESIS METHOD OF LOGIC CIRCUITS BASED ON THE NMOS-LIKE RRAM GATES

#### A. THE LOGIC BLOCK WITH THE NMOS-LIKE RRAM GATES

The logic functionalities are realized in the logic block or logic blocks based on the proposed NMOS-like RRAM gates. The NMOS-like RRAM gates in the same logic block are able to work simultaneously. The RRAM cell in the AND gate and OR gate presents the opposite input states with respect to that in the NAND gate and NOR gate. It means that, if a logic state $x, x \in \{0,1\}$, is stored in the RRAM cell of the NOR gate or the NAND gate by an input operation, the corresponding RRAM cell of the AND gate or the OR gate stores $\overline{x}$ by the same input operation.

The logic block is designed as follows.

1. Transform the logic function with the SOP (Sum of Production) form into an AND-NOR based expression by the De Morgan’s Law.
2. Map each first-level AND term in the logic function into one column. The set end of the corresponding RRAM cell is the bottom end for the positive variable, and the polarity of the RRAM cell is reversed for the negative variable.
3. Reorganize the RRAM cells representing the variables $x$ and $\overline{x}$ in different columns into the same row, where $x$ is a free variable of the logic function.
4. Add the I/O line at the top of the RRAM columns. A resistor $R$ is connected in series at one end of the I/O line. The terminal with the resistor $R$ is used as an input node, and the other terminal of the I/O line is used as the output node of the gate. All the columns are connected to the I/O line.
5. Add one terminal for each node in the circuit. The terminals connecting to the bottom end of the RRAM cells in the same row share the same input voltages.

We name this synthesis method as method A. The one-bit full-adder is taken as an example to illustrate the synthesis method of the logic block. Its logic function is defined as the equations (3-1) and (3-2).

$$\begin{align*}
S & = A \oplus B \oplus C = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + ABC \quad (3-1) \\
C & = AB + BC + AC = \overline{A}\overline{B} + BC + AC \quad (3-2)
\end{align*}$$

The proposed full-adder circuit is presented as Fig.6. The circuit consists of two logic blocks, since the function is expressed by two logic equations. The logic block shown in Fig.6 (a) implements the function (3-1). The logic block consists three rows and four columns, because $S$ contains four three-variable AND terms. The terminals 8~11, 12~15 and 16~17 share the corresponding applied voltages, respectively. It consumes four cycles to obtain the result $S$ because the input phase is executed row by row. Similarly, the logic block in Fig.6 (b) realizes the function (3-2). It consumes four cycles to obtain the output $C$. The full adder circuit consists of fifteen RRAM cells and two resistors, and it consumes four cycles in total, because the two logic blocks work in parallel.

The output voltages of the logic blocks in Figs. 6(a) and (b) are simulated with the conditions in section II. The waveforms presented in Fig.7 show that the NMOS-like RRAM gates based full-adder circuit works correctly.

Table 2 compares the MOS-less RRAM based one-bit full-adder circuits generated from different synthesis methods. The circuit performance is represented by the cycle counts, and the circuit area is represented by the number of
RRAM cells and resistors. It shows that the proposed one-bit full-adder circuit in Fig.6 has advantages on both the performance and the area consumption compared with the counterparts generated from the previous MOS-less RRAM based methods.

### B. THE SYNTHESIS METHOD BASED ON THE NMOS-LIKE RRAM GATES

According to the analysis in Section II (B), the design constraints of the logic block are summarized as follows.

The maximum number of RRAM cells which participant in the computation in one row does not exceed 11, and the maximum number of RRAM cells which participant in the computation in one column does not exceed 6, if the $V_{OH}$ and $V_{OL}$ are set as 0.4V and 0.15V, respectively. The constraints of the logic block change consequently, if the simulation condition changes.

The full-adder circuit in Fig.6 conforms the design constraints of logic block. However, if the logic function exceeds the design constraints of logic block, the multi-stage circuit architecture has to be introduced. We improve the synthesis method A with two rules for the cases that beyond the constraints of logic block.

The full-adder circuit in Fig.6 conforms the design constraints of logic block. However, if the logic function exceeds the design constraints of logic block, the multi-stage circuit architecture has to be introduced. We improve the synthesis method A with two rules for the cases that beyond the constraints of logic block. The full-adder circuit in Fig.6 conforms the design constraints of logic block. However, if the logic function exceeds the design constraints of logic block, the multi-stage circuit architecture has to be introduced. We improve the synthesis method A with two rules for the cases that beyond the constraints of logic block.

The logic function is expressed in an AND-NOR form. Rule 1. For the cases that the required number of RRAM cells for the NOR terms exceed the constraints, the logic function is partitioned by transforming it to the AND-NOR-AND form. Each term conforms the constraints, and the transformed logic function is implemented in a multi-stage circuit architecture accordingly.

The full-adder circuit in Fig.6 conforms the design constraints of logic block. However, if the logic function exceeds the design constraints of logic block, the multi-stage circuit architecture has to be introduced. We improve the synthesis method A with two rules for the cases that beyond the constraints of logic block.

Rule 2. For the cases that the required number of RRAM cells for the AND terms exceed the constraints, the logic function is partitioned by transforming it to the NOR-NOR circuit architecture accordingly.

The logic function is expressed in an AND-NOR form. Rule 1. For the cases that the required number of RRAM cells for the NOR terms exceed the constraints, the logic function is partitioned by transforming it to the AND-NOR-AND form. Each term conforms the constraints, and the transformed logic function is implemented in a multi-stage circuit architecture accordingly.

The logic function $F_1$ defined by (3-3) is taken as an example to illustrate the synthesis method using Rule 1.

F1 contains sixteen first-level AND terms. It requires sixteen RRAM cells in one row to realize the second-level NOR function, which is beyond the constraints of logic block. $F_1$ is transformed into an AND-NOR-AND form by Rule 1, as presented in formula (3-4).

Each NOR term in (3-4) conforms the design constraints of logic block. The circuit is implemented by the two-stage circuit as presented in Fig.8 (a). The two AND-NOR functions in (3-4) are implemented by the two first-stage blocks. The switches and inverters are inserted between the cascading stages for the isolation and voltage recovery purpose. The switches are in the OFF state when the first-stage blocks are in the input phase, and the switches are in the ON state during the computation phase of the first-stage blocks. The inverters output voltage $V_p$ and $-V_p$ for logic 1 and 0, respectively. The second-stage block implements the NOR function, instead of the AND function in (3-4), because the inverters are inserted between stages. The first-stage blocks consume six cycles. The outputs of the first-stage blocks are fed to the inputs of the second-stage block in the same cycle. The circuit totally consumes seven cycles.

The logic function $F_2$ as presented in (3-5) is taken as the example to illustrate the application of Rule 2.

All the first-level AND terms consist of seven variables, which exceed the constraints presented above. To implement $F_2$ in the logic blocks, the logic function is transformed to the NOR-NOR expression as (3-6). Each NOR term conforms the design constraints of the logic block.

Fig.8 (b) implements the function $F_2$. It consumes five cycles accordingly.

The large-scale circuits are designed by applying the two rules in combination, even if all the NOR terms and the AND terms exceed the design constraints. The transformed logic function consumes the same number of RRAM cells as that in the original logic function, but it requires more MOSFET devices. Improved by the two rules, the synthesis method A is applicable for arbitrary logic functions.

The polarities of RRAM cells in Fig.8 (a) are not uniform. The hybrid-polarity logic block is implemented if the number of input variables do not exceed the design constraints. The method A is applied to some small-scale benchmark circuits. The synthesis results are collected in Table 3. It is seen that the

| Method A | # of Cycles | # of Resistors | # of RRAMs |
|----------|-------------|---------------|------------|
| IMPLY [3] | 19          | 5             | 28         |
| CRS [4]  | 14          | 1             | 16         |
| MAGIC [5] | 16          | 0             | 28         |
| MAGIC [17] | 10          | 0             | 48         |
| MAGIC [22] | 13          | 0             | 36         |
| MAGIC [23] | 18          | 0             | 24         |

Table 2. Comparisons of different MOS-less RRAM based full-adder circuits.
IV. THE SYNTHESIS METHOD OF IN-ARRAY LOGIC CIRCUITS

A. A CASE STUDY

The logic block with the uniform RRAM polarity is array-implementable. The logic function \( F_3 \), which is expressed in the NOR-NOR expression, is taken as an example to illustrate the design method of the in-array circuit.

\( F_3 \) contains sixteen first-level NOR terms, and the first NOR term contains twelve variables. \( F_3 \) in (4-1), as shown at the bottom of the next page, exceeds the constraints of the logic block, and it has to be transformed to (4-2), as shown at the bottom of the next page, by applying Rule 1. The transformed logic function only consists of the NOR and AND operations, so Rule 2 is not applied.

The logic function in (4-2) conforms the constraints of the logic block, and it is realized by the circuit in Fig. 9. The first-stage block computes all the first-level NOR terms. The first NOR term in (4-1) occupies two RRAM rows, because it is partitioned into two parts by the NOR-AND operations in (4-2). The number of rows in the first-stage block equals to the number of NOR terms in (4-2). The NOR results of the first and second rows are NORed together in the second-stage block, because the inverters are inserted between stages. The value of \( A + B + C + D + E \cdot F + G + H + I + J + K + L \) are obtained.

The third-stage and the fourth-stage blocks compute the second-level NOR operations in (4-1). The number of columns in the third-stage block equals to the number of the first-level NOR terms in (4-1). The third-stage block contains two rows because the \( F_3 \) in (4-1) contains sixteen NOR terms, which exceeds the constraints of logic block. The fourth-stage block computes the final result.

All the RRAM cells in the first-stage block are initialized in the same cycle. The seventeen row-based NOR gates execute simultaneously. So the first-stage block consumes two cycles. Each successor logic block consumes one more cycle, because the output operations of the current-stage block share the same cycle with the input operations of the next-stage block. The circuit consumes five cycles in total.

B. THE SYNTHESIS METHOD OF THE ARRAY-IMPLEMENTABLE CIRCUIT

The proposed synthesis method of logic circuits based on the array-implementable blocks is as follows.
Step 1. Transform the logic function into the NOR-NOR form. If the NOR-NOR expression of the logic function exceeds the design constraints of the logic block, retransform the logic function by applying Rule 1 iteratively, until the transformed logic function conforms the constraints of logic block. The transformed logic function only consists of the NOR and AND terms. In general, the terms in the logic function can be partitioned into four parts: the first-level NOR terms, the second-level AND terms if any, the second-level NOR terms, and the final AND operations if any.

Step 2. Map each first-level NOR term into one row with the NMOS-like NOR gate. The RRAM cells representing the same input variable in different rows are reorganized into the same column. Insert one MOSFET switch and one MOSFET inverter at each output terminal of the array. The number of rows of the first-stage block equals to the number of first-level NOR terms in the logic function, and the number of columns equals to the number of different input variables in the first-level NOR terms.

Step 3. The array which contains one NOR gate in each row is named as the NOR array. The NOR array based successor logic block implements the AND term in the logic function if any, because the inverters are inserted between the cascading blocks. Each RRAM cell in the second-stage NOR gate stores the intermediate value of the corresponding first-level NOR-NOT operation. If the number of the intermediate values exceeds the constraints of the logic block, the multi-stage structure is introduced. The switches and inverters are also inserted between stages. The NOR arrays are cascaded until all the second-level AND operations in the logic function are performed. The number of stages of the NOR array for the second-level AND operations is noted as M.

Step 4. The second-level NOR terms are implemented by the AND arrays, which contains one AND gate in each row. The AND gates are used because the inverters are inserted between stages. Each RRAM cell in the (M+1)th stage stores the intermediate value of the corresponding NOR-NOT operation in the predecessor logic blocks. The number of rows in the (M+1)th stage equals to the number of corresponding NOR operations in the predecessor logic blocks. The multi-stage structure is required, if the number of the intermediate values exceeds the constraints of the logic block. The switches and inverters are also inserted between stages. The number of stages of the AND array for the NOR operations is noted as N.

Step 5. If the number of intermediate values of the second-level NOR terms exceeds the design constraints of the logic block, the NOR arrays are required for the final AND operations. Each RRAM cell in the (M+N+1)th NOR array stores the intermediate value of the corresponding NOR operation in the predecessor logic blocks. If the number of intermediate values exceeds the constraints of the logic block, it needs the multi-stage structure to implement the final AND operations. The switches and inverters are inserted between stages. The number of stages of NOR array for the final AND operations is noted as K.

This synthesis method is named as method B. The circuit generated from the method B implements the logic functions with M+N+K+1 cascading arrays, and it consumes M+N+K+2 cycles. The full-adder circuit generated from method B is presented in Fig.10. It consumes only three cycles, which is faster than the counterpart in Fig.6. But it requires twenty-five RRAM cells, nine resistors and twenty-one MOS transistors.

C. THE SCHEMES WITH MULTIPLE OUTPUT STEPS

The method B generates the high-performance circuits because all the outputs of the current-stage block are fed to

\[
F_3 = \begin{bmatrix}
A + B + C + D + E + F + G + H + I + J + K + L + \overline{A} + B + C + D + E \\
+ \overline{A} + B + C + D + E + A + \overline{B} + C + D + E + \overline{A} + B + C + D + E \\
+ A + \overline{B} + C + D + E + A + B + C + D + E + \overline{A} + B + C + D + E \\
+ \overline{A} + B + C + D + E + A + B + C + D + E + \overline{A} + B + C + D + E \\
+ A + \overline{B} + C + D + E + A + B + C + D + E + \overline{A} + B + C + D + E \\
+ A + B + C + D + E + A + \overline{B} + C + D + E + A + B + C + D + E + \overline{A} + B + C + D + E \\
\end{bmatrix}
\]

(4-1)

\[
F_3 = \begin{bmatrix}
A + B + C + D + E + F + G + H + I + J + K + L \\
A + B + C + D + E \\
A + B + C + D + E \\
A + B + C + D + E \\
A + B + C + D + E \\
A + B + C + D + E \\
\end{bmatrix}
\]

(4-2)
the next-stage block in the same cycle. However, it results in the heavy area consumption, because it requires one switch and one inverter at each I/O line between stages.

The MOS transistor consumes much more area than the RRAM cell does. In order to reduce the number of MOS transistors, the scheme with multiple output steps is proposed. The main idea of the scheme is that, all the outputs of an array share the same inverter, and the output operations are performed sequentially. The output sequence of the block is controlled by the switches. For the n-output array which conforms the constraints of logic block, it only consumes n + 3 MOS transistors, but it requires n + 1 cycles. This method is named as method C.

The example F3 is implemented by the method C, as presented in Fig.11 (a). It saves seventeen MOS inverters, and it consumes the same number of RRAM cells and resistors as that in Fig.9. But it consumes nineteen cycles to obtain the final result, since the output operations through the paths $V_{g1} - V_{g3}$ and $V_{g2} - V_{g3}$ can be executed in parallel with the output operations of the first-stage logic block through $V_{g24} - V_{g6}$ and $V_{g23} - V_{g6}$, respectively. The circuit in Fig.11 (a) is much slower than the circuit in Fig.9.

The tradeoff between the solutions in Fig.9 and Fig.11 (a) results in the method D. It divides the outputs of the block into several groups. Each group shares one inverter, and the outputs in different groups execute simultaneously. This technique can be applied to each stage or the selected stages, according to the requirements of circuit area and performance. The method D is applied to the example F3 between the third logic block and the first/second logic blocks, as presented in in Fig.11 (b). The input operations of the third-stage array require eight cycles, and the circuit consumes fourteen cycles in total.

Generally speaking, the circuit performance increases with the increase of transistor count. However, the capability of performance improvement varies, if the location of the
inserted inverter changes. For the array with a large number of outputs, the scheme has better efficiency on the performance improvement, if the outputs are partitioned uniformly.

The method B and C are applied to some MCNC benchmark circuits. The synthesis results are compared with the counterparts generated from the previous methods in Table 4. It is seen that the method B results in the circuits with the best performance, but consumes more area. The circuits generated by the method C also outperform those previous counterparts for the most cases.

The application of method D is flexible. The best performance result from method D is the same as that generated from the method B. And the most area efficient result from the method D is the same as that generated from the method C. So the synthesis results from the method D are not listed in the Table 4.

**D. THE SCHEME WITH THE UNIFORM-POLARITY ARRAYS**

The circuit generated from method B ~D consists multi-stage logic blocks. The RRAM polarities in each logic block are

![Figure 11](image)

**FIGURE 11.** The schemes with multiple output steps (a) circuit of F3 by the method C (b) circuit of F3 by the method D.

**TABLE 4.** The comparison of different synthesis methods for the RRAM based logic circuits.

| Function  | Input | RRAM count / MOS count /Resistor count/Cycle count |
|----------|-------|-----------------------------------------------|
|           |       | [7]   | [8]   | [9]   | [10]  | [12]  | Method B | Method C |
| rd73f1    | 7     | 116/- | 140/- | 142   | 404   | 68/- | 298/414 | 298/654 |
|           | 223   |       |       |       |       | 137  |           |           |
| rd73f2    | 7     | 26/-  | 253/- | -     | 25/-  | 37   | 518/210 | 518/817 |
|           | 97    |       |       |       |       |      |           |           |
| rd73f3    | 7     | 59/-  | 210/- | -     | 210   | 70   | 179/117 | 179/484 |
|           | 138   |       |       |       |       |      |           |           |
| sao2f1    | 10    | 59/-  | 33/-  | 67    | 94    | 34/- | 69/57 | 69/262 |
|           | 87    |       |       |       |       |      |        |           |
| sao2f2    | 10    | 63/-  | 31/-  | 49    | 208   | 41/- | 140/102 | 140/423 |
|           | 104   |       |       |       |       |      |        |           |
| sao2f3    | 10    | 133/- | 22/-  | 20    | 130   | 89/- | 92/42 | 92/215 |
|           | 199   |       |       |       |       |      |        |           |
| sao2f4    | 10    | 128/- | 33/-  | 34    | -     | 77/- | 69/33 | 69/141 |
|           | 202   |       |       |       |       |      |        |           |
uniform, but the RRAM polarities in different arrays may be different. The reason is that the circuit uses the NOR arrays and the AND arrays, and the polarities of RRAM cells in these two types of array are opposite. The AND operations are introduced in the cases that the number of NOR terms exceeds the constraints of logic block.

Actually, the scheme with the uniform-polarity arrays is implementable by using the NOR arrays in each stage. It is achieved by substitute the AND arrays with the NOR arrays, and substitute the inverters at the input terminals of the corresponding arrays with the buffers, simultaneously, based on the circuit generated from the method B→D. Each buffer is realized by two cascading inverters. The circuit area increases after the substitutions. However, the substitutions have little impact on the circuit performance.

V. CONCLUSION
This paper proposes the NMOS-like RRAM gate family, and studies the synthesis methods based on these voltage-input voltage-output RRAM based logic gates. The advantages of the NMOS-like RRAM based logic circuit include: (1) the circuit performance is higher than those of the previous counterparts, thanks for the parallelism of the NMOS-like gates; (2) the arbitrary large-scale logic functionalities can be implemented by cascading the logic blocks; (3) the circuit is array-implementable, which supports the computation in memory. The design practices and synthesis results show that the proposed synthesis methods are able to generate the relatively high-performance circuits, and these methods enable the designers to tradeoff the area and performance of the circuits by sharing the inter-block buffers or inverters.

REFERENCES
[1] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found,” Nature, vol. 453, no. 7191, pp. 80–83, May 2008.
[2] I. Vourkas and G. C. Sirakoulis, “Emerging memristor-based logic circuit design approaches: A review,” IEEE Circuits Syst. Mag., vol. 16, no. 3, pp. 15–30, 3rd Quart., 2016.
[3] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, “Memristive” switches enable ‘stateful’ logic operations via material implication,” Nature, vol. 464, no. 7290, pp. 873–876, Apr. 2010.
[4] R. Rosezin, E. Linn, C. Kugeler, R. Bruchhaus, and R. Waser, “Crossbar logic using bipolar and complementary resistive switches,” IEEE Electron Device Lett., vol. 32, no. 6, pp. 710–712, Jun. 2011.
[5] S. Kvatsinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, “MAGIC—Memristor-aided logic,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 61, no. 11, pp. 895–899, Nov. 2014.
[6] G. S. Rose, J. Rajendran, H. Manem, R. Karri, and R. E. Pino, “Leveraging memristive systems in the construction of digital logic circuits,” Proc. IEEE, vol. 100, no. 6, pp. 2033–2049, Jun. 2012.
[7] F. Lalchhandama, B. G. Sapui, and K. Datta, “An improved approach for the synthesis of Boolean functions using memristor based IMPLY and INVERSE-IMPLY gates,” in Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI), Jul. 2016, pp. 319–324.
[8] J. Bürger, C. Teuscher, and M. Perkowski, “Digital logic synthesis for memristors,” in Proc. Reed-Muller Workshop, 2013, pp. 1–10.
[9] A. Raghavanshi and M. Perkowski, “Logic synthesis and a generalized notation for memristor-realized material implication gates,” in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2014, pp. 470–477.
[10] X. Wang, R. Tan, and M. Perkowski, “Synthesis of memristive circuits based on stateful IMPLY gates using an evolutionary algorithm with a correction function,” in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2014, pp. 2842–2852, Dec. 2018.
[11] Y. Yang, J. Mathew, S. Pontarelli, M. Ottavi, and D. K. Pradhan, “Complementary resistive switch-based arithmetic logic implementations using material implication,” IEEE Trans. Nanotechnol., vol. 15, no. 1, pp. 94–108, Jan. 2016.
[12] H.-P. Wang, C.-C. Lin, C.-C. Wu, Y.-C. Chen, and C.-Y. Wang, “On synthesizing memristor-based logic circuits with minimal operational pulses,” IEEE Trans. Very Large Scale Intege (VLSI) Syst., vol. 26, no. 12, pp. 2842–2852, Dec. 2018.
[13] S. Kvatsinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, “Memristor-based material implication (IMPLY) logic: Design principles and methodologies,” IEEE Trans. Very Large Scale Intege (VLSI) Syst., vol. 22, no. 10, pp. 2054–2066, Oct. 2014.
[14] R. Ben Hur, N. Wald, N. Talati, and S. Kvatsinsky, “Simple magic: Synthesis and in-memory mapping of logic execution for memristor-aided logic,” in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2017, pp. 225–232.
[15] I. Vourkas and G. C. Sirakoulis, “A novel design and modeling paradigm for memristor-based crossbar circuits,” IEEE Trans. Nanotechnol., vol. 11, no. 6, pp. 1151–1159, Nov. 2012.
[16] H. Li, P. Huang, B. Gao, B. Chen, X. Liu, and J. Kang, “A SPICE model of resistive random-access memory for large-scale memory array simulation,” IEEE Electron Device Lett., vol. 35, no. 2, pp. 211–213, Feb. 2014.
[17] S. Kvatsinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, “MRL—Memristor-aided logic,” in Proc. 13th IEEE Int. Workshop Cellular Nanos. Netw. Appl., Turin, Italy, Aug. 2012, pp. 1–6.
[18] P. Huang, J. Kang, Y. Zhao, S. Chen, R. Han, Z. Zhou, Z. Chen, W. Ma, M. Li, L. Liu, and X. Liu, “Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits,” Adv. Mater., vol. 28, no. 44, pp. 9758–9764, Nov. 2016.
[19] N. Talati, S. Gupta, P. Mane, and S. Kvatsinsky, “Logic design within memristive memories using memristor-aided iLOGIC (MAGIC),” IEEE Trans. Nanotechnol., vol. 15, no. 4, pp. 635–650, Jul. 2016.
[20] R. Gharpinde, P. L. Thangkhiew, K. Datta, and I. Sengupta, “A scalable in-memory logic synthesis approach using memristor crossbar,” IEEE Trans. VLSI Syst., vol. 26, no. 2, pp. 355–366, Feb. 2018.
FENG WEI received the B.Eng. degree in electronic science and technology from the University of Electronics Science and Technology of China, Chengdu, China, in 2016, and the M.Eng. degree in electronic science and technology from the Southern University of Science and Technology, Shenzhen, China, in 2019. He is currently a Research Assistant with the Peking University Shenzhen Graduate School. His research interest is VLSI design.

XIAOXIN CUI (Member, IEEE) received the B.Sc. degree in cybernation from Beihang University, Beijing, China, in 2002, and the Ph.D. degree in microelectronic from Peking University, Beijing, in 2007. She is currently an Associate Professor with Peking University, Beijing. Her research interests include low-power IC design and hardware security.