SCIENCE CHINA
Information Sciences

• LETTER •

October 2020, Vol. 63 209401:1–209401:3
https://doi.org/10.1007/s11432-019-2684-9

A synthesis method for logic circuits in RRAM arrays

Xiaole CUI[1], Xiao MA[1], Feng WEI[1] & Xiaoxin CUI[2*]

[1]*Shenzhen Graduate School, Peking University, Shenzhen* 518055*, China;*
[2]*Institute of Microelectronics, Peking University, Beijing* 100871*, China*

Received 6 July 2019/Revised 9 September 2019/Accepted 14 October 2019/Published online 19 May 2020

Citation Cui X L, Ma X, Wei F, et al. A synthesis method for logic circuits in RRAM arrays. Sci China Inf Sci, 2020, 63(10): 209401, https://doi.org/10.1007/s11432-019-2684-9

Dear editor,

Resistive RAM (RRAM) based logic circuits have attracted much attention in recent years. The IMPLY (material implication) gates are the first published RRAM logic gate family [1], and many synthesis methods based on the IMPLY gates have been proposed and discussed in literature. These synthesis methods are all concerned with logic circuits in a single RRAM row. In this study, we propose a synthesis method for logic circuits in an RRAM array based on the multi-input IMPLY and OR gates.

The multi-input IMPLY gate [2], which represents logic states with the resistance states of the RRAM cells, as presented in Figure 1(a), realizes the function $Q = \overline{A_1 + \cdots + A_n} + Q$. This represents a multi-input NOR gate when the output cell Q is in the high resistance state (HRS). Figure 1(b) presents the multi-input OR gate [2], it can be used together with the IMPLY gate in the RRAM array because this gate also represents logic states with the resistance states of the RRAM cells, and the applied voltages are compatible. It is noted that the IMPLY and OR functions can be realized if the polarities of the RRAM devices and the applied voltages are reversed simultaneously. The equivalent IMPLY and OR gates are presented in Figures 1(c) and (d), respectively. The equivalent gates share the same constraints of input counts, whose computation and simulation are presented in the supplementary file, because of the same circuit structure. All the IMPLY and OR gates in Figures 1(a)–(d) only require one cycle for operation if the resistance states of the RRAM cells have been set correctly. Further, if the gates in Figures 1(a) and (d) are deployed in the RRAM row, the gates in Figures 1(b) and (c) realize the same logic functions in the column of the same array. This property enables designers to implement in-array logic circuits. The proposed synthesis method of the in-array logic circuit consists of two phases.

Phase 1. Circuit mapping.

The mapping process works as follows.

Step 1. Transform each minterm in the standard SOP (sum of product) expression of the logic function into the NOR form using de Morgan's law.

Step 2. Sort the NOR cubes in the order of number of variables. The NOR cubes are partitioned into two groups. Group A contains the NOR cubes that cannot be mapped into a single RRAM row, and all remaining NOR cubes are in group B.

Step 3. Map the NOR cubes in group A. Select the NOR cube with the maximum number of variables in the group and partition the variables of the selected NOR cube into several subgroups while considering the constraints of array size. The variables in each subgroup are mapped to a single RRAM row. Map the variables in each subgroup to the RRAM array row by row, and delete the

* Corresponding author (email: cuixx@pku.edu.cn)

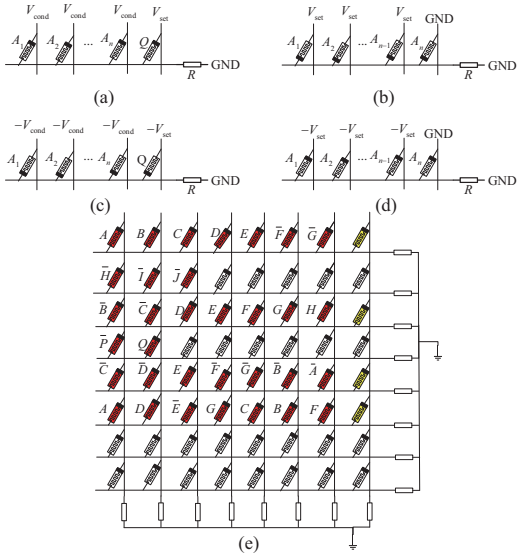

Figure 1 (Color online) RRAM based circuits. (a) Multi-input IMPLY gate [2]; (b) multi-input OR gate [2]; (c) equivalent IMPLY gate; (d) equivalent OR gate; (e) in-array circuit for the example circuit.

current NOR cube in group A. Then, select the NOR cube with the maximum number of variables in the current group and repeat these operations. This process continues until group A is empty or there is no available RRAM row space.

Step 4. Map the NOR cubes in group B. The NOR cubes in group B are mapped to the rows that are not occupied by the NOR cubes in group A. The mapping of each row in the RRAM array consists of three sub-steps.

Sub-step 1. Select the NOR cube with the maximum number of variables in the current group B and map its variables to an empty row.

Sub-step 2. Map the NOR cubes containing the redundant variables with respect to the mapped NOR cubes in the current row.

First, select the unmapped NOR cube containing the maximum number of redundant variables with respect to the mapped NOR cube in the current row.

If there are enough available RRAM cells in the current row, then map the variables other than the redundant variables in the selected NOR cube into the current row. Then, select the unmapped NOR cube containing the maximum number of redundant variables with respect to the mapped NOR cubes in the current row, and map all variables other than the redundant variables in this cube to the current row, if there are available RRAM cells in the current row.

Otherwise, select the unmapped NOR cube containing the sub-maximum number of redundant variables with respect to the mapped NOR cubes in the current row, and map the non-redundant variables in this cube to the current row if there are available RRAM cells in the row.

This process continues until the current row is full or none of the NOR cubes in group B contain the redundant variables with respect to the mapped NOR cubes in the current row. Then, delete such mapped NOR cubes in group B.

Sub-step 3. Fill in the remainder of the available RRAM cells in the current row.

If there are still available cells in the current row, it selects the unmapped NOR cube with the maximum number of variables that can be mapped to the current row in the current group one by one, and maps their variables to the current row. This process continues until all the available cells in the current row are filled. These mapped NOR cubes are then deleted from group B.

Select and map the NOR cubes in the current group using the three sub-steps above in a row-wise order, if empty rows are available. This process continues until the current group B is empty or none of the rows in the RRAM array are available. The circuit mapping is completed successfully if the process stops with the former condition, whereas it fails if the process stops with the latter condition.

The circuit with the logic function $F = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}FGHIJ + BC\overline{D}\,\overline{E}\,\overline{F}\,\overline{G}\,\overline{H}P\overline{Q} + CD\overline{E}FG + B\overline{E}FG + \overline{A}\,\overline{D}E\overline{G} + \overline{A}\,\overline{C}E\overline{G} + \overline{A}\,\overline{B}E\overline{G} + A\overline{E}FG + E\overline{F}$ is considered as an example to illustrate the mapping process.

The target here is to map the logic function F to an 8×8 RRAM array.

Step 1. The logic function above is transformed into

$$\begin{aligned} F' = {} & \overline{A + B + C + D + E + \overline{F} + \overline{G} + \overline{H} + \overline{I} + \overline{J}} \\ & + \overline{\overline{B} + \overline{C} + D + E + F + G + H + \overline{P} + Q} \\ & + \overline{\overline{C} + \overline{D} + E + \overline{F} + \overline{G}} + \overline{\overline{B} + E + \overline{F} + \overline{G}} \\ & + \overline{A + D + \overline{E} + G} + \overline{A + C + \overline{E} + G} \\ & + \overline{A + B + \overline{E} + G} + \overline{\overline{A} + E + \overline{F} + \overline{G}} + \overline{\overline{E} + F} \end{aligned}$$

by the de Morgan's law.

Step 2. In the 8×8 RRAM array, a maximum of seven variables can be mapped to one row because the rightmost cell in each row is reserved for the intermediate value. For the transformed F' in Step 1, the first two NOR cubes belong to group A and the other NOR cubes belong to group B.

Step 3. $\overline{A+B+C+D+E+\overline{F}+\overline{G}+\overline{H}+\overline{I}+\overline{J}}$ is selected first as it contains the maximum number of variables. The variables $A, B, C, D, E, \overline{F}$, and $\overline{G}$ are mapped to Cell$(1,1)$–Cell$(1,7)$, and the variables $\overline{H}$, $\overline{I}$, and $\overline{J}$ are mapped to

Cell(2, 1)–Cell(2, 3), respectively. Then the NOR cube $\overline{\overline{B}+\overline{C}+D+E+F+G+H+\overline{P}+Q}$ is selected, and its variables are mapped to the third and fourth rows, as presented in Figure 1(e).

Step 4. The NOR cube $\overline{\overline{C}+\overline{D}+E+\overline{F}+\overline{G}}$, which contains the maximum number of variables in the current group B, is selected. The variables $\overline{C}$, $\overline{D}$, E, $\overline{F}$, and $\overline{G}$ are mapped to Cell(5, 1)–Cell(5, 5), respectively. Both $\overline{\overline{B}+E+\overline{F}+\overline{G}}$ and $\overline{\overline{A}+E+\overline{F}+\overline{G}}$ have three redundant variables, i.e., E, $\overline{F}$ and $\overline{G}$ with respect to the mapped NOR cube, and the variables $\overline{B}$ and $\overline{A}$ are mapped to Cell(5, 6) and Cell(5, 7), respectively. All the available cells in the fifth row are filled. Following the same process, the remaining four NOR cubes are mapped to the sixth row. The circuit mapping is thus completed successfully because the circuit only occupies six rows in the 8×8 array.

Phase 2. Computation.

The result of the logic function is obtained by four computation steps.

Step 1. Input: First, all the RRAM cells in the array are reset to HRS simultaneously. Then, the corresponding cells are set into the low resistance state (LRS) in each row if their input state is logic 1. The set operations consume at most k cycles if the circuit occupies k rows.

Step 2. Compute the NOR cubes that occupy multiple rows. For each NOR cube that occupies multiple RRAM rows, each corresponding bit in different rows is OR-ed, and the intermediate results are stored in the RRAM cells in the first row they occupy. This requires only a single cycle. All the rows storing the intermediate results perform NOR operations in the same cycle, and the results of the NOR cubes are stored in the corresponding rightmost RRAM cells.

Step 3. Compute the NOR cubes that are mapped in a single row. For each row occupied by a complete NOR cube or several NOR cubes, compute the first NOR cube and store the results in the rightmost RRAM cell of this row. Then, the other NOR cubes are OR-ed, one by one, by the IMPLY operations, and the results are stored in the rightmost cell in the row iteratively until all the NOR cubes in this row are computed.

Step 4. Output: The intermediate results stored in the cells in the rightmost column are OR-ed and the final result is stored in one of these cells.

In the example presented in Figure 1(e), Step 1 requires at most seven cycles because the set operations have to be executed row by row. In Step 2, it executes $\text{Cell}(1, i) = \text{Cell}(1, i) + \text{Cell}(2, i)$ and $\text{Cell}(3, i) = \text{Cell}(3, i) + \text{Cell}(4, i), i = 1$–$7$, and the results of the two NOR cubes are computed and stored in Cell(1, 8) and Cell(3, 8), simultaneously. In Step 3, for the fifth row in Figure 1(e), the NOR operation is applied to the cells $\overline{C}$, $\overline{D}$, E, $\overline{F}$, $\overline{G}$ and Cell(5, 8), and the result of $\overline{\overline{C}+\overline{D}+E+\overline{F}+\overline{G}}$ is stored in Cell(5, 8). Then $\overline{\overline{B}+E+\overline{F}+\overline{G}}+\text{Cell}(5, 8)$ is computed by the multi-input IMPLY operation, and the intermediate result is stored in Cell(5, 8). Finally, $\overline{\overline{A}+E+\overline{F}+\overline{G}}+\text{Cell}(5, 8)$ is computed and stored in Cell(5, 8). Similar computations are performed in the sixth row. In Step 4, it computes $\text{Cell}(1, 8)+\text{Cell}(3, 8)+\text{Cell}(5, 8)+\text{Cell}(6, 8)$. The final result is stored in Cell(1, 8) as the output of the logic function.

Experiments. The proposed array-oriented synthesis method of the logic circuit is applied to some benchmark circuits with different number of columns. The synthesis results show that, for most cases, the circuits generated by our proposed two-dimensional synthesis method outperform those counterparts that are generated by the previous one-dimensional synthesis methods.

The details of the data and the discussion can be found in the supplementary file.

Conclusion. This study proposes a synthesis method for logic circuit in the RRAM array. Assuming a logic function which has m and n NOR cubes in the original group A and B, respectively, it is mapped to k rows in the RRAM array. The generated circuit totally requires $m + n + k + 3$ cycles, including one initialization operation, k set operations, $m+1$ OR operations, and $n+1$ multi-input IMPLY operations, to obtain the final result.

Acknowledgements This work was supported by Shenzhen Science and Technology Innovation Committee (Grant No. JCYJ20170412150411676).

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Borghetti J, Snider G S, Kuekes P J, et al. 'Memristive' switches enable 'stateful' logic operations via material implication. Nature, 2010, 464: 873–876

2 Xie L, Nguyen H A D, Taouil M, et al. Boolean logic gate exploration for memristor crossbar. In: Proceedings of International Conference on Design and Technology of Integrated Systems in Nanoscale Era, Istanbul, 2016. 1–6