On Synthesizing Memristor-Based Logic Circuits in Area-Constrained Crossbar Arrays

Hsin-Tsung Lee, *Chia-Chun Lin, Yung-Chih Chen, and Chun-Yao Wang
* Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R.O.C.
Email: *chiachunlin@gapp.nthu.edu.tw

2021 22nd International Symposium on Quality Electronic Design (ISQED) | 978-1-7281-7641-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISQED51717.2021.9424249

***Abstract*—Memristors are considered as promising candidates for Computation-In-Memory due to their non-volatile storage and computing capabilities. In recent years, a growing number of general-purpose computing platforms based on different design styles at the memristor crossbar arrays have been proposed. In this paper, we present a comprehensive synthesis approach for performing arbitrary Boolean logic operations in area-constrained crossbar arrays. The approach exploits multicycle logic operations to deal with the delay and area constraints. The experimental results at larger EPFL benchmarks show that our approach is robust and scalable.**

I. INTRODUCTION

In recent years, researchers developed diverse logic design styles that use the crossbar arrays directly. The IMPLY design style performs material implication through the logic gate built with two memristors and one resistor [6]. For the MAGIC design style, NOR and NOT operations are mapped to the crossbar arrays directly. In [5], a single memristor cell has shown its capability to perform a 3-input majority operation. In [4], 14 types of logic functions can be implemented in three cycles by a single memristor.

And-Inverter Graphs (AIGs) [3], and Majority-Inverter Graphs (MIGs) [1] are common ways to represent Boolean functions under the memristor crossbar arrays during logic synthesis. Different cost functions considering area and latency in the physical implementation with memristor devices have also been proposed [5]. In these cost functions, the estimated latency is proportional to the depth of graph representation of circuit. Furthermore, the estimated area is highly related to the total number of nodes in the graph. However, these estimations are based on an assumption that all the operations at the same level of graph can be performed simultaneously. In fact, the actual latency and area might increase due to different crossbar array constraints in different design styles, which might limit the possibility of full parallelism.

To the best of our knowledge, none of these previous studies discussed the mapping flow of using multicycle logic operations under the crossbar array constraints. Hence, in this paper, we demonstrate how suitable the multicycle logic operation style is for the crossbar arrays, especially for parallel operations. We also utilize row and column parallelism under different scenarios to obtain a better result. Note that the extra registers for saving results are not necessary in our work, i.e., all the variables read out from memristors need to be used at the same cycle.

II. PROPOSED APPROACH

In our work, we propose an approach to synthesize a function on a 2D memristor crossbar array. The basic operations in a memristor crossbar array are as follows: The variables that are stored in one row can be read out in the first half-cycle. The variables that have been read out in the first half-cycle, constant 1, or constant 0, then can be applied at the selected control lines in the second half-cycle. Furthermore, the AND and NOT operations will be performed separately in our approach. This is because an AND operation receives its input from T1 and always applies logic 1 at T2, but a NOT operation receives its input from T2 and always applies logic 1 at T1. These operation mechanisms make these two types of operations not conducted at the same row simultaneously.

This work is supported in part by the Ministry of Science and Technology of Taiwan under MOST 108-2218-E-007-061, MOST 109-2221-E-007-082-MY2, MOST 109-2221-E-155-047-MY2, and MOST 109-2224-E-007-005.

TABLE I
EXPERIMENTAL RESULTS OF EPFL BENCHMARKS USING 256 × 256 CROSSBAR ARRAYS.

BENCHMARK	|PI|	|PO|	|AND|	|NOT|	|CYCLE|
adder	256	129	1020	1150	1375
div	128	128	20969	18722	14753
max	512	130	2865	2904	814
mult	128	128	26727	23302	41697
sin	24	25	5202	3441	1641
sqrt	128	64	19332	16487	20064

III. EXPERIMENTAL RESULTS

We implemented the proposed algorithm in C++ and conducted the experiments on Intel Xeon ®E5-2650V2 2.60GHz machine with 64 GB memory under CentOS 6.7. We have evaluated the proposed approach by using the EPFL benchmarks [2]. We conducted our approach on some larger designs from EPFL benchmarks, which showed the robustness and scalability of our approach. The experimental results are listed in TABLE I.

IV. CONCLUSION

We present a synthesis approach for computing arbitrary Boolean function on memristor crossbar arrays. We demonstrate the potential of the multicycle operations to reduce the number of reading cycles under the 2D crossbar arrays.

REFERENCES

[1] L. Amaru, P. E. Gaillardon, G. D. Micheli, "Majority-Inverter Graph: A Novel Data-Structure and Algorithms for Efficient Logic Optimization," *In Proc. of DAC*, pp. 1-5, June 2014.
[2] L. Amaru, P.E. Gaillardon, G.D. Micheli, "The EPFL Combinational Benchmark Suite," *In Proc. of IWLS*, June 2015
[3] A. Kuehlmann, V. Paruthi, F. Krohm, M. K. Ganai, "Robust Boolean Reasoning for Equivalence Checking and Functional Property Verification," *IEEE TCAD*, vol. 21, pp. 1377-1394, Dec. 2002.
[4] E. Linn, R. Rosezin, S. Tappertzhofen, U. Böttger, R. Waser "Beyond von Neumann—Logic Operations in Passive Crossbar Arrays Alongside Memory Operations," *Nanotechnology*, vol. 23, pp. 23-30, 2012.
[5] S. Shirinzadeh, M. Soeken, P. E. Gaillardon, R. Drechsler, "Logic Synthesis for RRAM-Based In-Memory Computing," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 1422-1435, Sep. 2017.
[6] 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 TVLSI*, vol. 26, pp. 2842-2852, Dec. 2018.