

Synthesizing Mixed-Mode Operations for Memristors using Majority Decomposition

Felix Bayhurst¹, Li-Wei Chen¹, Kefeng Li², Heidemarie Krüger², Nan Du², Ilia Polian¹

¹Institute of Computer Architecture and Computer Engineering, University of Stuttgart, Stuttgart, Germany

²Institute for Solid State Physics Leibniz Institute of Photonic Technology and University of Jena, Jena, Germany

Abstract—Memristive technologies can enable novel mixed-mode (MM) circuit architectures, where diverse stateful and non-stateful logic operations are executed by the same physical device. Recently introduced optimal synthesis procedures for MM circuits have achieved 3-5X area and latency improvements compared with single-mode memristive logic families, yet such methods are not scalable. In this paper, we present a synthesis approach for MM circuits that leverages synthesis techniques for majority-inverter graphs (MIGs). MIG vertices are natural descriptions of non-stateful voltage-input (V-op) and stateful resistance-input (R-op) logic operations. Our synthesis can handle circuits with up to 27 inputs and achieves an average reduction of 80% in required devices and 65% delay when compared to a state-of-the-art approach for R-ops.

I. BACKGROUND

Memristors are emerging nano-devices that combine compute and memory capabilities. They promise decisive benefits in domains ranging from neuromorphic computing to security [1], [2]. A number of *memristive logic families* [3]–[6] have been suggested, each associated with advantages and disadvantages. Recently, a *mixed-mode* (MM) paradigm combining different types of operations within the same circuit [7] has been introduced, enabling circuits that are 3–5× more compact than their conventional counterparts. The considered circuits mix two types of digital operations: non-stateful *voltage-input operations* (V-ops) and stateful *resistance-input operations* (R-ops). V-ops, which correspond to the CRS logic family [4] without reading, apply logic values to top and bottom electrodes (TE/BE) of an individual memristive device. To be universal on their own, V-ops require a readout [8]. R-ops can be instantiated by different logic families, but here we are using two examples: MAGIC [3], which implements 2-input NOR on three connected devices, and its 3-input extension “controlled-NOR” (CNOR) [7], [9]. Both types of R-ops are universal, but CNOR offers higher synthesis efficiency. Extensive synthesis work has been done for memristor-based in-memory computing using R-ops on crossbar architectures, extending CMOS synthesis with memristor-aware optimizations [10], [11]. In contrast, our work maps circuit functionality to both R-ops and V-ops, prioritizing V-ops as they are less sensitive to device variability and can realize logic functions with substantially fewer operational cycles. The first MM-synthesis method [8] provided optimality guarantees, but was restricted to small circuits with ≤ 7 inputs. Motivated by its results, we consider architectures where V-ops are concentrated in the beginning of the circuit, followed by R-ops. We also restrict ourselves to one-dimensional line arrays in which all memristors are connected in parallel.

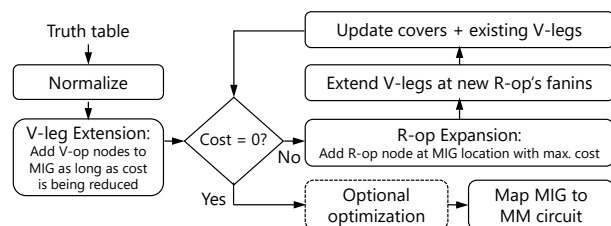


Fig. 1: Synthesis methodology for a single circuit output

Our scalable MM-synthesis approach is based on a decomposition algorithm for *majority-inverter graphs* (MIGs) [12], which offer a natural representation and manipulation of V-ops and R-ops alike. In [12] a tree-shaped MIG with no internal inversions is generated from a truth table. The considered divisors are input-literals normalized (XNOR) with the target function. The decomposition is based on *care functions*, where cf of an edge indicates care-bits (truth table positions) that need to have the same function as the target function. The procedure in [12] iteratively inserts vertices into the MIG and distributes the cf over their fanin edges, choosing divisors such as to maximize its *cover*. Synthesis concludes when all vertices cover all their care-bits.

We extend this approach to a MIG where vertices correspond to V-ops and R-ops using two connected vertices for CNOR R-ops. To achieve this, we find sequences of V-ops called V-legs optimized to cover a given cf . The procedure then introduces R-ops, splitting up the cf and allowing for further introduction and optimization of V-legs. Through this, we improve the number of MAGIC gates required from mapping with the standard logic-synthesis tool ABC [13] by over 90%, reducing both delay and area of synthesized circuits.

II. MIG-BASED SYNTHESIS OF MIXED-MODE CIRCUITS

Our synthesis procedure takes the truth table of an N -input multi-output function and produces its mixed-mode memristive circuit realization. It proceeds one output after the other, following the steps outlined in Fig. 1. The procedure first attempts to realize the target function using only V-ops. To this end, the procedure will add MIG vertices with pairs of literals to be applied at TE/BE, thus gradually extending the V-leg. We evaluate each feasible pair for its effect on the maximum (that is, the already achieved) and the minimum achievable *cover* after its addition to a V-leg. In combination with search

space pruning and different levels of filtering, we can efficiently explore possible V-legs and find the best fitting one for a cf .

The first R-op is added if the initial V-leg cannot realize the function completely; its position is unique. A difference to [12] is the structural restriction from V-ops and R-ops, which necessitates the consideration of fixed inputs and inverted edges. For illustration, the OR function is obtained by fixing one input of a general MIG vertex to the normalized inverted constant-0 $\overline{nc0}$. We call the non-fixed, freely chooseable fanin edges fi_j . For R-ops that implement the NOR or CNOR function the *cover* c can be described by

$$c(\text{R-op}) = nc0 \wedge \left(\bigvee_{j=0}^m c(fi_j) \right) \vee \bigwedge_{j=0}^m c(fi_j) \quad (1)$$

with m being 2 for NOR and 3 for CNOR. The last 2 fanins of the considered R-ops are part of a NOR-vertex with an inverted fanout. These inversions need to be properly propagated through to all literals connected to these fanins. From Eq. 1 we can divide care-bits into two sets: those covered by the fixed input $nc0$ ($\overline{nc0}$ if an inversion was propagated to it) require only one of the fi_j to be covered on the output, whereas the remaining care-bits need all fi_j . These more relevant-to-cover care-bits need to be prioritized. Edges fi_j of a new R-op are extended to V-legs using the same procedure as above with the inversion pushed to all its literals if necessary. To balance structural priorities of the cf introduced through R-op restrictions and include both local, as well as global, properties in a comparable metric, we define a weighted cost-function over the two priority sets of care-bits. This allows the procedure to be adjusted e.g. to different R-op types.

After the first R-op, we choose to expand on the position with the highest cost, which offers the biggest potential for improvement. Like in [12], the new R-op is only added if it reduces cost, otherwise alternative positions are tried. If no position can successfully expand, a cost-balancing parameter is adjusted and another round of attempts is initiated. After successfully expanding on a fanin, the change in *cover* is propagated from the new R-op towards the output y . We then update all cfs in the tree down from the topmost reached vertex. All V-leg positions with modified, but not yet fully covered, cfs are repeatedly re-tried to achieve the best fitting combination of V-legs. When we reach a cost of 0, a MIG describing a valid MM circuit realization is produced. It can be optimized and mapped to a schedule of control pulses for an actual memristive line array. When handling a circuit with multiple outputs, we want to encourage logic sharing from early on. Therefore our method reuses intermediate solutions by re-introducing R-op functions from earlier outputs as divisors back into the decomposition.

Mapping to mixed-mode circuit: One additional requirement to execute the synthesized MM circuit on a line array of devices is given by its shared BE. That is, for all cycles i , each V-leg contains the same literal applied to the BE in this leg's i -th V-op. To achieve this, we start with a valid solution and calculate the minimum necessary care-function of each V-leg. We iteratively select the best literal for shared BE, i.e.,

the one minimizing the remaining cost, applying no-op (same literal on TE) where necessary. Although we focus here on one-dimensional line arrays (WL or BL), the same principles can be extended to two-dimensional crossbar architectures by generalizing the shared electrode assignment across both dimensions.

III. EXPERIMENTAL RESULTS

We applied our synthesis method to all MCNC benchmarks [14] with < 28 PIs and report results for selected circuits. All results have been obtained on a server with an Intel-Xeon E5-2687W CPU and 251GB RAM running AlmaLinux-8. Our synthesis method, reduces the number of MAGIC operations by roughly $10\times$ when compared to results from ABC. To show the advantage of our MM-synthesis, we compare results obtained with post-synthesis mapping (shared BE) to SIMPLE [10], which uses ABC for MAGIC-based synthesis and optimizes the mapping to a 2D crossbar array afterwards. We compare the area (number of devices) and the latency (number of cycles) in Table I. While both types of operations are assumed to have 1 cycle of latency, we note that V-ops on different V-legs can run in parallel on our 1D line-array, whereas R-ops cannot.

TABLE I: Comparison of SIMPLE [10] (on a 2D crossbar) and post-mapping Mixed-Mode circuits (on a 1D line array)

Circuit	PI	PO	SIMPLE [10]		CNOR-MM		Red. MM [%]	
			No. Dev	No. Cyc	No. Dev	No. Cyc	No. Dev	No. Cyc
5xp91	7	10	142	97	35	24	75.3	75.2
misex1	8	7	112	45	24	18	78.5	60
clip	9	5	184	136	33	33	82.0	75.7
x2	10	7	86	36	15	13	82.5	63.8
cm162a	14	5	92	46	17	18	81.5	60.8
cm163a	16	5	95	45	13	14	86.3	68.8
cm150a	21	1	87	51	22	25	74.7	50.9

Despite being restricted to 1D line arrays with no R-op parallelism altogether, MM circuits achieve an average reduction of 80% in required memristive devices and 65% in delay.

Runtime: The runtimes tend to increase with the circuit size and the number of R-ops (expansions). Similar to CMOS synthesis, larger circuits will need peephole optimization, which works with submodules of around 8 PIs. Our procedure requires a median runtime of 62 ms among all 8-PI MCNC benchmarks.

IV. CONCLUSION AND FUTURE WORK

The mixed-mode paradigm significantly improves memristive realizations of logic functions, reaching up to over $10\times$ reductions in area and time requirements (as opposed to $3-5\times$ reported in [8] for smaller circuits). We presented the first scalable synthesis method for MM circuits, which leverages MIG-based decomposition, but substantially extends earlier procedures to support V-ops and two types of R-ops. The method is effective for modules with up to 27 inputs with runtimes of smaller circuits being in a suitable range for peephole optimization.

In the future, we plan to explore synergies with well-developed methods for CMOS. A further question is the extension of the method from line-arrays to 2D crossbars. This could leverage known mapping approaches such as [10] [11].

REFERENCES

- [1] W. Huang, X. Xia, C. Zhu, P. Steichen, W. Quan, W. Mao, J. Yang, L. Chu, and X. Li, "Memristive artificial synapses for neuromorphic computing," *Nano-Micro Letters*, vol. 13, no. 1, p. 85, 2021.
- [2] N. Du, H. Schmidt, and I. Polian, "Low-power emerging memristive designs towards secure hardware systems for applications in internet of things," *Nano Materials Science*, vol. 3, no. 2, pp. 186–204, 2021, nano Energy Materials and Devices for Miniaturized Electronics and Smart Systems. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2589965121000015>
- [3] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Magic—memristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.
- [4] T. You, Y. Shuai, W. Luo, N. Du, D. Bürger, I. Skorupa, R. Hübner, S. Henker, C. Mayr, R. Schüffny *et al.*, "Exploiting memristive BiFeO₃ bilayer structures for compact sequential logics," *Advanced Functional Materials*, vol. 24, no. 22, pp. 3357–3365, 2014.
- [5] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL—memristor ratioed logic," in *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*. IEEE, 2012, pp. 1–6.
- [6] S. Gupta, M. Imani, and T. Rosing, "Felix: Fast and energy-efficient logic in memory," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–7.
- [7] N. Du, I. Polian, C. Bengel, K. Li, Z. Chen, X. Zhao, U. Hubner, L.-W. Chen, F. Liu, M. D. Ventra, S. Menzel, and H. Krueger, "Mixed-mode in-memory computing: Towards high-performance logic processing in a memristive crossbar array," *Communications Engineering*, 2025, accepted. Preprint available under <https://arxiv.org/abs/2506.19063>.
- [8] I. Polian, X. Zhao, L.-W. Chen, F. Bayhurst, Z. Chen, H. Schmidt, and N. Du, "Optimal synthesis of memristive mixed-mode circuits," in *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, pp. 1–7. [Online]. Available: <https://ieeexplore.ieee.org/document/10993268/>
- [9] N. Peled, R. Ben-Hur, R. Ronen, and S. Kvatinsky, "X-MAGIC: Enhancing PIM using input overwriting capabilities," in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, pp. 64–69. [Online]. Available: <https://ieeexplore.ieee.org/document/9344095>
- [10] R. Ben-Hur, N. Wald, N. Talati, and S. Kvatinsky, "Simple magic: Synthesis and in-memory Mapping of logic execution for memristor-aided logic," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, pp. 225–232. [Online]. Available: <http://ieeexplore.ieee.org/document/8203782/>
- [11] R. Ben-Hur, R. Ronen, A. Haj-Ali, D. Bhattacharjee, A. Eliahu, N. Peled, and S. Kvatinsky, "SIMPLER MAGIC: Synthesis and Mapping of In-Memory Logic Executed in a Single Row to Improve Throughput," vol. 39, no. 10, pp. 2434–2447. [Online]. Available: <https://ieeexplore.ieee.org/document/8781866/>
- [12] S.-Y. Lee, H. Riener, and G. D. Micheli, "Logic resynthesis of majority-based circuits by top-down decomposition," in *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, pp. 105–110. [Online]. Available: <https://ieeexplore.ieee.org/document/9417058/>
- [13] R. Brayton and A. Mishchenko, "ABC: an academic industrial-strength verification tool," in *Proceedings of the Computer Aided Verification (CAV 2010)*, ser. Lecture Notes in Computer Science, vol. 6174. Springer, 2010, pp. 24–40. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-14295-6_5
- [14] K. Koźmiński *et al.*, "Benchmarks for layout synthesis (mcnc benchmark suite)," in *Proceedings of the MCNC Workshop on Layout Synthesis*, 1991, distributed by the Microelectronics Center of North Carolina.