2025 Design, Automation & Test in Europe Conference (DATE 2025)

Optimal Synthesis of Memristive Mixed-mode Circuits

Ilia Polian[1] Xianyue Zhao[2] Li-Wei Chen[1] Felix Bayhurst[1] Ziang Chen[2] Heidemarie Schmidt[2] Nan Du[2]

[1]*University of Stuttgart*
Institute of Computer Architecture and Computer Engineering
Stuttgart, Germany
{ilia.polian | li-wei.chen | bayhurst.felix}
@informatik.uni-stuttgart.de

[2]*Friedrich Schiller University Jena*
Institute for Solid State Physics
Leibniz Institute of Photonic Technology
Department of Quantum Detection
Jena, Germany
{Xianyue.Zhao | Ziang.Chen | Heidemarie.Schmidt | Nan.Du}
@leibniz-ipht.de

***Abstract*—Memristive crossbars are attractive for in-memory computing due to their integration density combined with compute and storage capabilities of their basic devices. However, yield and fidelity of emerging memristive technologies can make their reliable operation unattainable, thus raising interest in simpler topologies. In this paper, we consider synthesis of Boolean functions on 1D memristive line arrays. We propose an optimal procedure that can fully utilize the rich electrical behavior of memristive devices, mixing stateful (resistance-input) and non-stateful (voltage-input) operations as desired by the designer, leveraging their respective strengths. The synthesis method is based on Boolean satisfiability (SAT) solving and supports flexible constraints to enforce, e.g., restrictions of the available peripherals. We experimentally validate memristive logic circuits beyond individual logic gates by demonstrating the operation of a Galois field multiplier using a 1D line array of 10 memristors in parallel, highlighting the robust performance of our proposed mixed-mode circuit and its synthesis procedure.**

***Index Terms*—Logic synthesis, Memristive circuits, Logic-in-memory computing, Voltage-input and resistance-input logic operations, Universality**

I. Introduction

MEMRISTIVE devices combine compute and storage capabilities and thus enable post-von-Neumann logic-in-memory computer architectures [1]. Their rich electrical behavior allows the designer to realize diverse functionality, from neuromorphic computing [2] to arbitrary Boolean functions. For the latter purpose, several memristive logic families have been proposed, falling into stateful [3]–[5] and unstateful [6] operations. Memristors are often arranged on a 2D crossbar, and several synthesis methods to map Boolean functions onto a memristive crossbar have been suggested [7]–[10].

Memristors are still an emerging technology associated with low yield, limited endurance and reliability. Most of the studies mentioned have no experimental validation, or their experiments are restricted to a single logic gate, whereas larger circuits are modeled and simulated. In particular, multiple consecutive (cascaded) stateful operations on the same device can have insufficient fidelity. While analog neuromorphic operations have some natural resilience, digital logic gates needed for realizing Boolean functions have higher demands on the devices used for computations. This makes it difficult to assume that an entire crossbar has a sufficient number of working devices.

As a remedy, we consider a simpler topology: a 1D line-array, which can be integrated or composed of individual discrete devices that can be easily replaced after manufacturing or upon failure in operation. Our mixed-mode (MM) circuit design leverages the full electrical potential of memristors by allowing a flexible combination of stateful (resistance-input, R-ops) and non-stateful (voltage-input, V-ops) operations on each device. We formulate a synthesis framework that enables designers to optimize the use of R-ops and V-ops based on specific technological constraints. While R-ops require simpler peripheral circuitry, they are less reliable, particularly when cascaded. V-ops generally require fewer devices but are not universal in the readout-free version employed here, limiting their ability to realize arbitrary functions. If the line array is sufficiently large and R-ops are reliable, the entire circuit can be synthesized using only stateful operations. However, our proposed MM circuit, which strategically incorporates V-ops among R-ops, reduces the required number of devices, improves overall reliability, and broadens the range of functions that can be implemented, offering a distinct advantage over stateful-only approaches.

In contrast to existing synthesis procedures, which are gate-oriented and use data structures known from classical synthesis, such as binary decision diagrams [7], [8], and-inverter-graphs [9], [10] or majority-inverter graphs [11], the proposed synthesis method is based on creating and solving a monolithic Boolean satisfiability formula $\Phi(f, N_V, N_R)$. In this formula, f is the (multi-output) Boolean function being realized, and N_V and N_R are the numbers of allowed V-ops and R-ops, respectively. The formula contains variables encoding the control signals to be applied to the line array's top and bottom electrodes over a range of cycles; a solution of Φ yields a valid schedule to execute f on the line array. The procedure is optimal in the sense that an unsatisfiable Φ proves lack of solution for given parameters. To find a minimal realization, the designer calls the procedure repeatedly, with decreasing N_V and N_R until solving fails.

This work is supported by the German Research Foundation (DFG) project MemCrypto (grant PO 1220/15-2 and DU 1896/2-2) within the Proirity Program Nano Security and DFg Project MemDPU (grant DU 1896/3-1).

The key contributions of our work, highlighting the novelty of our approach, are as follows:

- We introduce and experimentally validate mixed-mode (MM) memristive operation by applying both V-ops and R-ops to the same device. Our approach uniquely combines the strengths of these operations while mitigating their individual drawbacks, enabling, for the first time, the optimal realization of complex small-scale functions, such as 3- and 4-input logic gates.
- We propose a new SAT-based synthesis method specifically designed for MM circuits, departing from traditional CMOS-inspired memristive synthesis techniques. This method achieves optimality in circuit design, setting a new benchmark for memristive synthesis.
- We synthesize various arithmetic primitives, demonstrating 3–5× improvement in latency and area compared to non-MM versions. Experimental validation of the synthesized circuits is conducted using $BiFeO_3$ memristors, confirming the practical benefits of our approach.

The remainder of this paper is organized as follows. The next section introduces mixed-mode memristive circuits, including the necessary background on memristive technology and studies voltage- and resistance-input operations theoretically. Synthesis of MM circuits is introduced in Section III. Synthesis results are reported in Section IV, while experimental results of physical measurements are shown in Section V. Section VI concludes the paper.

II. Mixed-Mode (MM) Memristive Circuits

A. Memristive devices and their operation

Memristors are two-terminal devices that support (at least) two internal states, called high- and low-resistance states (HRS, LRS). The voltages applied to the two terminals, called top and bottom electrodes (TE, BE), can induce transitions between the device's states. Typically in bipolar memristive devices (used in this work), for realization of Boolean functions considered in this paper, LRS and HRS are associated with logical values of 1 and 0, respectively. A memristor's state s can be read out by applying small-magnitude read voltage pulses to its TE and BE. In contrast, modifying its state requires higher-magnitude write pulses, where the presence or absence of a write pulse is indicated by logic 1 or 0, respectively. The logical behavior of a single device with the described encoding is summarized in Table I [6], [12]. This behavior is called *voltage-input operation* or V-op.

Beyond V-ops, which leverage the intrinsic switching behavior inherent to all two-terminal memristive devices, memristors also support additional operations known as stateful or resistance-input operations (R-ops). An example of an R-op is the MAGIC gate [4], which operates on a configuration of three interconnected memristors—two functioning as inputs and one as the output. By applying a carefully chosen voltage V_0 across these connected devices, the voltage-divider effect causes the output memristor to adopt the state $s_3 = \overline{s_1 \vee s_2}$, where s_1 and s_2 represent the states of the input memristors, effectively implementing a NOR gate. It is important to note that R-ops, in contrast to V-ops, are technology-dependent.

TABLE I
Voltage-input behavior V-op$(s, \mathrm{TE}, \mathrm{BE})$ of a memristive device

State s	TE	BE	Next-state	State s	TE	BE	Next-state
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	**0**
0	1	0	**1**	1	1	0	1
0	1	1	0	1	1	1	1

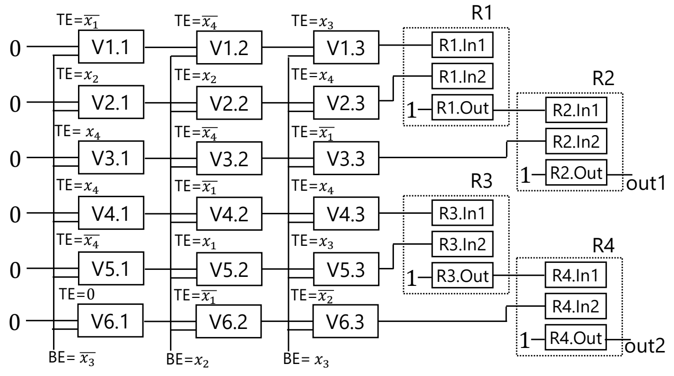

Fig. 1. Mixed-mode circuit to realize multiplication in $GF(2^2)$

For example, $BiFeO_3$ based memristive devices are well-suited for performing the above-mentioned MAGIC NOR gates, but, e.g., Ta_2O_5 memristors exhibit negated-implication (NIMP) functionality [13], compatible with the IMPLY logic family [3].

B. Mixed-mode composition

Circuits composed of several memristive devices are often organized in a 2D crossbar, with TEs (BEs) sharing the crossbar's columns (rows). In this work, we consider simpler line arrays, where the BEs of all devices are shared but TEs are independent, and arbitrary voltage pulses can be applied to them. Table II illustrates the V-op realizations for four basic 4-input logic operations AND, NAND, OR and NOR, running on a line array of four memristive devices with a shared BE (same input is always applied to all BEs). Its rows show alternating the states of four devices, the values applied to TEs and the shared BE of the four devices, from the set $L_4 = \{\text{const-0}, \text{const-1}, x_1, \overline{x_1}, x_2, \overline{x_2}, x_3, \overline{x_3}, x_4, \overline{x_4}\}$. We assume that all devices were initialized to 0, expressed through s_0 being the const-0 function `0000000000000000` in all cases.

The table shows the evolution of states assumed for the 16 possible inputs and the functions applied to each TE and BE in each cycle. For example, consider the transition from s_1 to s_2 for the NAND function $f_2 = \overline{x_1 x_2 x_3 x_4}$. For the third input from the left (the third entry of f's truth table, or $(x_1, x_2, x_3, x_4) = (0, 0, 1, 0)$), BE = 1 and TE = 0, so the memristor's state changes from 1 (LRS) to 0 (HRS). For the third input from the right, TE = 1 and BE = 0, and the state transitions from 0 to 1. It can be seen that all four functions are successfully constructed by V-ops only. Note that having shorter sequences for NAND and OR is unproblematic on a line array because we can always enforce "dummy cycles" by putting on TE the same value as is required (for neighboring devices) on BE.

TABLE II
REALIZATION OF 4-INPUT, 4-OUTPUT FUNCTION $f = (f_1, f_2, f_3, f_4) = (x_1x_2x_3x_4, \overline{x_1x_2x_3x_4}, x_1 + x_2 + x_3 + x_4, \overline{x_1 + x_2 + x_3 + x_4})$ BY VOLTAGE-INPUT OPERATIONS WITH SHARED BOTTOM ELECTRODE BE. EXAMPLE TRANSITIONS $s_1 \rightarrow s_2$ DISCUSSED IN TEXT ARE ENCIRCLED.

Step	$f_1 = x_1x_2x_3x_4$		$f_2 = \overline{x_1x_2x_3x_4}$		$f_3 = x_1 + x_2 + x_3 + x_4$		$f_4 = \overline{x_1 + x_2 + x_3 + x_4}$	
s_0		0000000000000000		0000000000000000		0000000000000000		0000000000000000
TE	x_4	0101010101010101	$\overline{x_4}$	1010101010101010	x_2	0000111100001111	const-1	1111111111111111
Shared BE	const-0	0000000000000000	const-0	0000000000000000	const-0	0000000000000000	const-0	0000000000000000
s_1		0101010101010101		10$\boxed{1}$0101010101$\boxed{0}$10		0000111100001111		0000000000000000
TE	x_2	0000111100001111	x_1	00$\boxed{0}$0000011111$\boxed{1}$11	x_4	0101010101010101	$\overline{x_2}$	1111000011110000
Shared BE	$\overline{x_3}$	0011001100110011	$\overline{x_3}$	00$\boxed{1}$1001100110$\boxed{0}$11	$\overline{x_3}$	0011001100110011	$\overline{x_3}$	0011001100110011
s_2		0100110101001101		10$\boxed{0}$0100011101$\boxed{1}$10		0100110101001101		1100000011000000
TE	x_3	0011001100110011	x_2	0000111100001111	x_3	0011001100110011	const-0	0000000000000000
Shared BE	x_1	0000000011111111	x_1	0000000011111111	x_1	0000000011111111	x_1	0000000011111111
s_3		0111111100000001		1000111100001110		0111111100000001		1100000000000000
TE	const-0	0000000000000000	$\overline{x_2}$	1111000011110000	x_1	0000000011111111	const-0	0000000000000000
Shared BE	const-0	0000000000000000	const-0	0000000000000000	const-0	0000000000000000	const-0	0000000000000000
s_4		0111111100000001		1111111111111110		0111111111111111		1100000000000000
TE	x_1	0000000011111111					$\overline{x_4}$	1010101010101010
Shared BE	const-1	1111111111111111	const-1	1111111111111111	const-1	1111111111111111	const-1	1111111111111111
s_5		0000000000000001						1000000000000000

An example mixed-mode (MM) circuit is shown in Fig. 1. It realizes multiplication in Galois field and has been obtained by the synthesis procedure introduced later on. It has four inputs x_1, x_2, x_3 and x_4 and two outputs. This function was chosen because it has an earlier memristive implementation in [14]; details on the function can be found in that paper. The circuit has four MAGIC NOR R-ops R1, R2, R3 and R4, each constructed out of two input and one output memristors, with cascaded R-ops R1/R2 and R3/R4 sharing one device each, thus totaling 10 memristors. Four of them are initialized to 1 while the remaining six are fed by the V-op part of the circuit. We call these six inputs *V-legs* and observe that each has three V-ops (18 in total) with TE and shared BE driven by literals from set L_4 defined above. We can see that an R-op (e.g., R2) can be driven by another R-op (cascading), a V-op, or a literal.

The number of memristive devices in a MM circuit is determined by the number of R-ops, due to its voltage-divider effect, as V-ops do not consume extra memristors. For example, V-ops V4.1, V4.2 and V4.3 in Fig. 1 are executed on R3's first memristor. However, their TE and BE values must be provided by the peripherals, whereas the control of an R-op needs less data (the peripherals need to provide the same voltage V_0 to the three devices involved in an R-op). At the same time, R-ops suffer from high sensitivity to non-ideal electrical behavior, especially device-to-device (D2D) and cycle-to-cycle (C2C) variations during the voltage divider operation, leading to higher error rates than for V-ops.

The latency of an MM circuit is determined by the used peripherals. All V-ops drawn in parallel in Fig. 1 are executed in parallel, resulting in 3 cycles for the 18 V-ops. R-ops can only be parallelized if their shared parts are not connected to each other. This does not happen for the considered line arrays, even though it is possible for a crossbar. As a consequence, the total number of cycles for the example circuit is 3 for V-ops, 4 for R-ops, 7 in total.

C. Universality

R-ops in isolation are universal for both the MAGIC NOR operation considered here and NIMP gates for Ta_2O_5 memristors. That is, it is possible to take an arbitrary function, map it to NOR or NIMP gates, and then to R-ops in the respective technologies without using any V-ops; by extension, any MM architecture which allows V-ops and R-ops in any arrangement is universal as well. The universality question is more intricate for V-ops. From Table II, we have seen that V-ops can realize NAND and NOR operations, which are universal. However, this holds only if the output of a V-op can be input to other V-ops. This scheme is known under the name "CRS-R" (complementary resistive switching with readout) [6]. However, reading resistance states from the memristive crossbar is highly undesirable, as it requires amplification and transfer to valid writing biases for input-dependent writing operations, which consumes additional power compared to memory and stateful logic operations. Consequently, we restrict values admitted on all TE and BE terminals to the literal set $L_n = \{\text{const-0}, \text{const-1}, x_1, \overline{x_1}, x_2, \overline{x_2}, \dots, x_n, \overline{x_n}\}$ for an n-input function, which is much easier to realize.

In general, given a function f, a V-op with the restriction above can compute $f \cdot l$ for any literal $l \in L_n$ by either applying l to TE and const-1 to BE, or const-0 to TE and $\bar{l}$ to BE. Similar properties hold for OR:

$$f \cdot l = V(f, l, \text{const-1}) = V(f, \text{const-0}, \bar{l}) \quad (1)$$
$$f + l = V(f, l, \text{const-0}) = V(f, \text{const-1}, \bar{l}) \quad (2)$$

Consequently, all functions that can be decomposed into a sequence of ANDs and ORs with literals are realizable by V-ops. However, V-ops are not universal: all functions of shape $x_1x_2 + x_3x_4$ with pairwise different x_1, x_2, x_3, x_4 are not realizable by V-ops alone. An important special case is the XOR function $x_1 \oplus x_2 = x_1\overline{x_2} + \overline{x_1}x_2$.

D. MM realizations of 3- and 4-input functions

To quantify the capability of MM architectures to realize arbitrary functions property, we evaluated different combinations of R-ops and V-ops for 3-input and 4-input functions. Given integers k_{pre} and k_{post}, we counted how many functions are realizable by taking L_n as the set of possible initial functions; applying up to k_{pre} R-ops (NOR gates) to these functions; applying an arbitrary number of V-ops until a fixed point is

TABLE III
NUMBERS N_3 AND N_4 OF 3-BIT AND 4-BIT FUNCTIONS REALIZABLE BY k_{PRE} R-OPS, AN ARBITRARY NUMBER OF V-OPS AND k_{POST} R-OPS. k_{TEBE} IS THE NUMBER OF R-OPS ALLOWED AT TE AND BE.

k_{pre}	k_{post}	k_{TEBE}	N_3	N_4	k_{pre}	k_{post}	k_{TEBE}	N_3	N_4
0	0	0	104	1850	1	1	0	104	1850
1	0	0	104	1850	2	1	0	158	3590
2	0	0	158	3590	3	1	0	186	6170
3	0	0	186	6170	1	2	0	246	32178
4	0	0	256	63424	1	3	0	256	65536
5	0	0	256	65536	2	2	0	256	53278
0	1	0	104	1850	0	0	1	254	57558
0	2	0	246	32178	0	0	2	256	65534
0	3	0	256	65536	**Total # functions**			**256**	**65536**

reached; and applying up to k_{post} additional R-ops to all pairs of functions calculated by V-ops. Moreover, we considered the effects of allowing R-ops on top and bottom electrodes. Note that a physical realization of such a feature is possible but costly because it involves readout of the memristive device's state during computation and sending the outcome to TE/BE.

The result of this experiment is reported in Table III. One can see that the number of functions realizable by V-ops only is not very large, but relatively few R-ops provide universality. Applying R-ops seems to be more effective after V-ops than before them, and applying R-ops before as well as after V-ops does not show any advantages. For this reason, we will focus on MM architectures with V-ops followed by R-ops.

III. OPTIMAL SYNTHESIS OF MIXED-MODE CIRCUITS

The procedure described in this section takes an n-input, N_o-output Boolean function f, numbers N_V and N_R, and either produces a valid memristive realization of f with N_V V-ops followed by N_R R-ops, or proves that such a realization does not exist. For example, the circuit in Fig. 2 was synthesized by specifying f_{GFMUL} according to the Boolean functions of the Galois field multiplier's two outputs, N_R = 4 and $N_V = 18$ (note that the number of devices $N_{\text{Dev}} = 12$ and the number of V-legs $N_L = 6$ are implied by N_R and N_V). Synthesis is accomplished by constructing and solving a Boolean satisfiability (SAT) formula $\Phi(f, N_V, N_R)$ in conjunctive normal form (CNF) with a solution that specifies N_V V-ops and N_R R-ops of a valid MM circuit realizing f.

Optimality in terms of the number of required operations is achieved by iteratively calling the procedure with decreasing N_V and N_R. For instance, we verified that synthesizing f_{GFMUL} with N_R = 4 and $N_V < 18$ results in an unsatisfiable $\Phi(f, N_V, N_R)$, and the same was observed for $N_R < 4$, suggesting that the circuit in Fig. 2 is optimal. In general N_V and N_R offer the designer a trade-off: there may exist an MM circuit with less V-ops and more R-ops. The choice of N_R can be driven by the number of available devices (considering that not all of them may be functional due to yield or endurance issues) and by technology properties.

Less R-ops at the expense of additional V-ops will tend to reduce latency because V-ops are parallelizable and R-ops are not, reduce power consumption due to a lower number of devices, and improve reliability. An excessive number of V-ops will increase the complexity of peripherals and power consumed by them. For technologies with low endurance, V-ops are problematic because, in the worst case, every V-op switches the cell (in practice, many cells will retain their old values). For technologies with excessive variability, R-ops can be unreliable especially in cascading scenarios. Therefore, having N_V and N_R as knobs available to the designer makes the approach flexibly applicable to improved memristive technologies of the future.

A. SAT formula construction

Formula $\Phi(f, N_V, N_R)$ incorporates two restrictions of the generic MM synthesis problem. First, it considers circuits with a V-op part followed by an R-op part (as in Fig. 1), rather than allowing arbitrary mixtures of R-ops and V-ops which would technically be possible. This is motivated by our observations in Section II-D. Second, it uses NOR as the R-op, because the MAGIC NOR operation is best-suited for BFO memristors used for experimental demonstration. When using different devices, the encoding of the fitting R-op would have to be changed.

We start with describing the model's set of variables and then present its equations. $N_T := 2^n$ is the number of rows in f's truth table.

- $l_{i,q}, 1 \leq i \leq 2 + 2n, 1 \leq q \leq N_T$ represent the literals of the synthesized functions: $l_{i,q} = 1$ iff the truth table of literal function i has 1 in row q, 0 otherwise.
- $v_{i,q}, 1 \leq i \leq N_V, 1 \leq q \leq N_T$ represent the outputs of V-ops (we use a consecutive index i for all V-legs).
- $r_{i,q}, 1 \leq i \leq N_R, 1 \leq q \leq N_T$ represent R-op outputs.
- $g^{\text{TE}}_{i,j}$ and $g^{\text{BE}}_{i,j}, 1 \leq i \leq N_V, 1 \leq j \leq 2 + 2N$ represent the connectivity of V-ops: $g^{\text{TE}}_{i,j} = 1$ ($g^{\text{BE}}_{i,j} = 1$) iff TE (BE) of V-op i is driven by literal j.
- $g^{\text{In1}}_{i,j}$ and $g^{\text{In2}}_{i,j}, 1 \leq i \leq N_R, 1 \leq j < 2 + 2n + N_V + i$ represent the connectivity of R-ops: $g^{\text{In}k}_{i,j} = 1$ iff the k-th input of R-op i is connected to function j, which can be a literal, a V-op, or a preceding R-op. For simplicity, we refer to function j's variables by $\alpha_{j,1}, \ldots, \alpha_{j,N_T}$.
- $o_{i,q}, 1 \leq i \leq N_O, 1 \leq q \leq N_T$ represent the outputs (truth tables) of the user-specified synthesized functions.
- $g^{\text{O}}_{i,j}, 1 \leq i \leq N_O, 1 \leq j \leq 2{+}2n{+}N_V{+}N_R$ represent the connectivity of outputs: $g^{\text{O}}_{i,j} = 1$ iff output i is connected to function j (R-op, V-op or literal).

The constructed formula Φ uses in several places the k-variable mutual exclusion (mutex) expression makes sure that exactly one of the variables $y_1, \ldots, y_k$ assumes the value 1:

$$\mu(y_1, \ldots, y_k) := (y_1 \vee \cdots \vee y_k) \wedge \bigwedge_{i<j} (\overline{y_i} \vee \overline{y_j}). \tag{3}$$

The formula Φ is given next; Eqs. 4–10 are explained below:

$$\bigwedge_{i,j} \begin{cases} (l_{i,q}), & \text{if entry } q \text{ of literal } i\text{'s truth table is 1} \\ (\overline{l_{i,q}}), & \text{if entry } q \text{ of literal } i\text{'s truth table is 0} \end{cases} \tag{4}$$

$$\wedge \bigwedge_{i,j,k,q} \left((g^{\text{TE}}_{i,j} \wedge g^{\text{BE}}_{i,k}) \rightarrow \left(v_{i,q} \equiv \begin{cases} V(l_{1,q}, l_{j,q}, l_{k,q}), \\ \quad \text{if } i \text{ at beginning} \\ \quad \text{of a V-leg} \\ V(v_{i-1,q}, l_{j,q}, l_{k,q}), \\ \quad \text{otherwise} \end{cases} \right) \right) \tag{5}$$

$$\wedge \bigwedge_i \mu(g^{\text{TE}}_{i,1}, \ldots, g^{\text{TE}}_{i,2+2n}) \wedge \bigwedge_i \mu(g^{\text{BE}}_{i,1}, \ldots, g^{\text{BE}}_{i,2+2n}) \quad (6)$$

$$\wedge \bigwedge_{i,j,k,q} \left((g^{\text{In1}}_{i,j} \wedge g^{\text{In2}}_{i,k}) \rightarrow (r_{i,q} \equiv R(\alpha_{j,q}, \alpha_{k,q})) \right) \quad (7)$$

$$\begin{aligned} &\wedge \textstyle\bigwedge_i \mu(g^{\text{In1}}_{i,1}, \ldots, g^{\text{In1}}_{i,2+2n+N_V+i-1}) \\ &\wedge \textstyle\bigwedge_i \mu(g^{\text{In2}}_{i,1}, \ldots, g^{\text{In2}}_{i,2+2n+N_V+i-1}) \end{aligned} \quad (8)$$

$$\wedge \bigwedge_{i,j} \begin{cases} (o_{i,q}), & \text{if entry } q \text{ of output } i\text{'s truth table is 1} \\ (\overline{o_{i,q}}), & \text{if entry } q \text{ of output } i\text{'s truth table is 0} \end{cases} \quad (9)$$

$$\wedge \bigwedge_{i,j,q} \left(g^{\text{O}}_{i,j} \rightarrow (o_{i,q} \equiv \alpha_{i,q}) \right) \wedge \bigwedge_i \mu(g^{\text{O}}_{i,1}, \ldots, g^{\text{O}}_{i,N_O}) \quad (10)$$

Eq. 4 defines the truth tables of the $(2+2n)$ literals of f by a set of unit clauses. For example, literal $i=3$ for the 2-input case is function $\overline{x_1}$ whose truth table has entries 1; 1; 0; and 0. For this literal, unit clauses $(l_{3,1}), (l_{3,2}), (\overline{l_{3,3}})$ and $(\overline{l_{3,4}})$ are generated. Eq. 5 captures the V-ops: for each row q of the truth table, the value v_i after V-op i must be derived from $v_{i-1,q}$ after application of literal j at TE and literal k at BE iff $g^{\text{TE}}_{i,j}$ and $g^{\text{BE}}_{i,k}$ are set. An exception applies when the V-op is in the beginning of a V-leg; then its input is const-0, represented by the first literal $l_{1,1}, \ldots, l_{1,N_T}$.

Eq. 6 makes sure that only one variable $g^{\text{TE}}_{i,j}$ is set for a given i, i.e., there is a uniquely defined function feeding the TE during every V-op, and the same holds for $g^{\text{BE}}_{i,j}$. Eq. 7 defines R-ops (recall that α can stand for an l, v or s variable). When $g^{\text{In1}}_{i,j}$ and $g^{\text{In2}}_{i,k}$ are set, R-op i is driven by functions α_j and α_k. Eq. 8 guarantees unique inputs to R-ops. Eq. 9 defines the truth tables of the output functions to be synthesized following the same principle as Eq. 4. Finally, Eq. 10 decides for each output which V-op, R-op or literal produces it.

All equations in the model are mapped to CNF and handed to a SAT solver. Its satisfying assignment provides information about the circuit's structure from the three types of variables g: variables $g^{\text{TE}}_{i,j}$ and $g^{\text{BE}}_{i,j}$ that are set to 1 for a given i indicate which literals are applied to TE and BE during V-op i; $g^{\text{In1}}_{i,j}$ and $g^{\text{In2}}_{i,k}$ specify both inputs for each R-op; and $g^{\text{O}}_{i,j}$ determine the circuit's outputs. Moreover, the results of V-ops, R-ops and outputs are contained in variables v, s and o for all possible inputs to the circuit.

It is possible to restrict the solution by additional constraints, e.g., forcing TE of V-op i to a specific literal j by adding a unit clause $(g^{\text{TE}}_{i,j})$. Since our experiments target a line array, we require that BE of V-ops during the same step are identical (e.g., V1.1, V2.1, ..., V6.1 in Fig. 1), by adding $2+2n$ pairs of CNF clauses $(g^{\text{BE}}_{i,k} \vee \overline{g^{\text{BE}}_{j,k}}) \wedge (\overline{g^{\text{BE}}_{i,k}} \vee g^{\text{BE}}_{j,k})$.

B. Example

The CNF formula $\Phi(f_{GFMUL}, 18, 4)$ is used to search for the mixed-mode circuit with 18 V-ops and 4 R-ops for the 4-input, 2-output Boolean function f_{GFMUL} of the multiplier in $GF(2^2)$. This formula, constructed using Eqs. 4–10, includes, for each of the 18 V-ops i, two sets of ten variables $g^{\text{TE}}_{i,1}, \ldots, g^{\text{TE}}_{i,10}$ and $g^{\text{BE}}_{i,1}, \ldots, g^{\text{BE}}_{i,10}$. In an assignment that satisfies Φ, precisely one variable of either set is set to 1. For example, the found solution that corresponds to the circuit in Fig. 1 has $g^{\text{TE}}_{2,9} = 1$, which means that the top electrode of V-op 2 (V1.2 in the figure) is driven by literal 9 out of the list of literals $L_4 = (\text{const-0}, \text{const-1}, \overline{x}_1, x_1, \ldots, \overline{x}_4, x_4)$.

In a similar manner, variables $g^{\text{In1}}_{4,31} = g^{\text{In2}}_{4,28} = 1$ specify the connectivity of the circuit's final R-op R4: its first input is driven by the output of R-op R3 (encoded by 31 because the first ten positions are reserved for L_4's ten literals and the following 18 for the V-ops), and its second input is connected to the last V-op V6.3, encoded as 28. Finally, the outputs of the circuit are connected to outputs of R2 (encoded by 30) and R4 (encoded by 32), specified by $g^{\text{O}}_{1,30} = g^{\text{O}}_{2,33} = 1$.

TABLE IV
RESULTS OF OPTIMAL SYNTHESIS FOR MIXED-MODE CIRCUITS (MM) AND CIRCUITS CONSISTING OF R-OPS (R-ONLY). N_{R}: NUMBER OF R-OPS; N_L: NUMBER OF V-LEGS; N_{VS}: NUMBER OF STEPS PER V-LEG; N_{ST}: TOTAL NUMBER OF STEPS; N_{DEV}: NUMBER OF DEVICES.

Circuit	Mode	n	N_{O}	N_R	N_L	N_{VS}	N_{St}	N_{Dev}	Vars	Clauses	T [s]
1-bit	MM	3	2	2	3	3	5	5	880	44.1K	3
adder	R-only			9	–	0	9	20	1394	34.2K	2
2-bit	MM	5	3	4	6	5	9	10	13.2K	1.6M	109
adder	R-only			$\leq$18	–	0	18	39	15.2K	784.8K	343233
3-bit	MM	7	4	5	8	6	11	14	93.0K	17.9M	24154
adder	R-only			$\leq$25	–	0	25	54	108.9K	8.1M	162433
GF(2^4)	MM	4	4	7	11	4	11	18	14.2K	1.1M	1539
inversion	R-only			$\leq$30	–	0	30	64	11.2K	997.6K	78187
GF(2^2)	MM	4	2	4	6	3	7	10	4544	347.5K	6
multipl.	R-only			$\leq$14	–	0	14	30	5106	199.0K	15

IV. Experimental Results

The optimal method from Section III was applied to small arithmetic functions with less than 7 inputs (8-input circuits are also considered out of reach for optimal gate-level synthesis in CMOS circuits). The smallest found mixed-mode circuit (MM, best combination of V-ops and R-ops) and the smallest circuit following the conventional paradigm and consisting of stateful R-ops only (R-only) are reported in Table IV. The table contains the numbers of R-ops N_R, the number of V-legs N_L, the number of steps, i.e., parallel operations on each V-leg N_{VS}, and the resulting number of steps for executing the entire circuit N_{St} and the number of required memristive devices N_{Dev}. The last three columns contain the number of SAT variables and clauses for Eqs. 4–10 and the run time. We used the SLIME 5 solver [15] on a 4.2 GHz AMD Ryzen 9 3950X 16-core processor with 128 GB RAM.

For the MM version, N_R is the smallest number for which a solution for $\Phi(f, N_V, N_R)$ could be found, and N_{VS} is the smallest value for that N_R. N_L is set to $N_R + N_O$ except for adders where it is set to $N_R + N_O - 1$, because the global carry output is realizable by V-ops and does not require an extra output. Note that 2-bit and 3-bit adders are not modular but are synthesized based on truth tables of all outputs. The number of steps for the MM version is $N_{\text{St}} = N_{VS} + N_R$ (no parallelization of R-ops), and the number of required memristors is $N_{\text{Dev}} = 2N_R + N_O$. For the R-only circuit, the minimum N_R with a soluble $\Phi(f, 0, N_R)$ was determined; the numbers of steps and devices were N_R and $2N_R + N_O$. For some circuits, the optimality proof could not be performed, i.e., the SAT solver timed out for $\Phi(f, 0, N_R - 1)$; these cases are marked by "$\leq$".

It can be seen from Table IV that MM circuits have significantly reduced latency and area requirements, 3–5 times

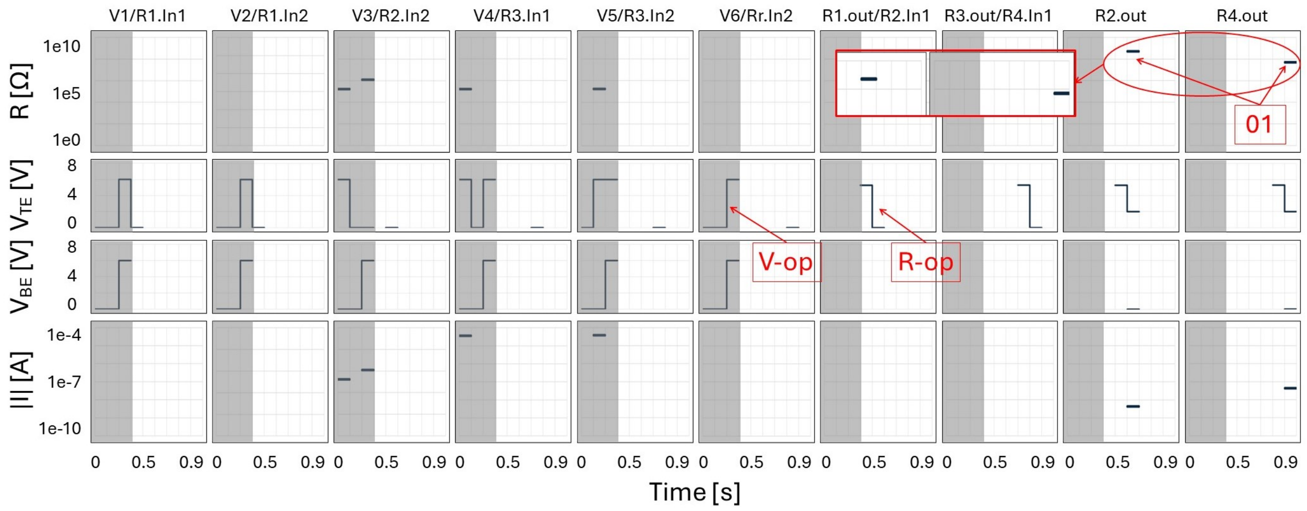

Fig. 2. Electrical measurement of all ten memristive cells of the GF multiplier of Fig. 1 for input $x_1x_2x_3x_4 = 1011$

less than the R-only versions. This is because integrating V-ops does not increase the number of required memristors, and V-ops are parallelizable, whereas R-ops are not. It seems that quite a large amount of computations can be offloaded to cheaper V-ops, whereas one or two R-ops per output are sufficient for computing even algebraically challenging functions such as GF inversion. Comparing with other works, the GF(2^2) multiplier in [14] uses 7 MRL gates with a total of 17 memristors, 11 transistors and 6 resistors, whereas our mixed-mode implementation needs just 4 memristors. Table V contains a comparison with representative published adder designs. Note that [18], [20] require smaller numbers of devices because they use IMPLY gates that are realizable with less memristors than the 3 devices per our R-op.

V. Experimental Demonstration

We implemented the MM $GF(2^2)$ multiplier, as shown in Fig. 1, on a line array using 10 BiFeO$_3$ (BFO) memristors in parallel [6], [21]. Each memristor consists of a BFO thin film, with a nominal thickness of 550 nm, sandwiched between an Au top electrode (TE) and a Pt bottom electrode (BE). The thin films were deposited via pulsed laser deposition, resulting in self-rectifying bipolar resistive switching behavior driven by an interface-induced switching mechanism with a tunable barrier height at the BE interface region. All electrical measurements were conducted using a Keithley 2400 source meter, controlled through a custom-made PCB board equipped with a switch unit for selecting and managing individual word lines (WLs) and bit lines (BLs) via a LabVIEW program on a PC.

Fig. 2 presents the experimental validation of multiplication in $GF(2^2)$ with the input sequence 1011, implemented using the optimal mixed-mode circuit synthesized and depicted in Figure Fig. 1. The experiment utilizes 10 memristive cells, displayed in columns, all initialized to state 0, with cells 7–10 pre-set to state 1, acting as output cells for R-ops R1–R4. The results showcase key parameters, including the resistance of each cell (first row), the voltages applied to individual top electrodes (TEs) and the shared bottom electrode (BE), and the absolute current $|I|$ across each cell (excluding the initialization phase). In cycles 1–3, all 18 V-ops are executed on cells 1–6, where, for example, the first device undergoes operations V1.1, V1.2, and V1.3. During cycles 4–5, R-ops are performed using cells 7–8 as output cells, with the readout of output 1 occurring in cycle 6. In cycles 7–8, R-ops utilize cells 9–10 as output cells, with the readout of output 2 in cycle 7. A total of 9 cycles are required to complete the $GF(2^2)$ multiplication, including readout cycles, which are highlighted in the diagram. Note that when the TE and BE are biased equally, resistance and current measurements are not observable. The successful experimental demonstration, resulting in correct readout values of out1 = 0 and out2 = 1, highlights the exceptional accuracy and efficiency of our proposed mixed-mode circuit design.

TABLE V
Comparison of MM adders with adder designs from literature

Adder	$n=1$		$n=2$		$n=3$		Adder	$n=1$		$n=2$		$n=3$	
	N_{St}	N_{Dev}	N_{St}	N_{Dev}	N_{St}	N_{Dev}		N_{St}	N_{Dev}	N_{St}	N_{Dev}	N_{St}	N_{Dev}
[16]	8	29	11	58	14	87	[17]	12	17	18	19	24	21
[18]	5	22	7	44	9	66	[19]	11	12	22	18	33	24
[20]	5	17	7	34	9	51	**Ours**	5	5	10	9	14	11

VI. Conclusions and Future Work

We demonstrated that mixed-mode circuits provide tremendous advantages compared to known memristive designs while at the same time being feasible experimentally. The proposed synthesis procedures allow the designer to focus on critical aspects of a specific technology (e.g., low reliability, limited device count, excessive peripherals). Their optimality guarantees make them ideal for, e.g., creation of library primitives.

Compared with the current solution for 1D memristive line arrays, 2D memristive crossbars offer new possibilities (e.g, potentially parallel R-ops) but also new complexities (restrictions on TEs in addition to BEs). Our future work will focus on extending optimal synthesis methods to crossbars, but also on developing scalable heuristic methods for larger functions, leveraging exact soluitions as much as possible.

References

[1] Y. Xiao, B. Jiang, Z. Zhang, S. Ke, Y. Jin, X. Wen, and C. Ye, "A review of memristor: material and structure design, device performance, applications and prospects," *Science and Technology of Advanced Materials*, vol. 24, no. 1, p. 2162323, 2023.

[2] S. Kumar, X. Wang, J. P. Strachan, Y. Yang, and W. D. Lu, "Dynamical memristors for higher-complexity neuromorphic computing," *Nature Reviews Materials*, vol. 7, no. 7, pp. 575–591, 2022.

[3] J. Borghetti *et al.*, "Memristive switches enable stateful logic operations via material implication," *Nature*, vol. 464, pp. 873–6, 04 2010.

[4] S. Kvatinsky *et al.*, "MAGIC—Memristor-aided logic," *IEEE Trans Circ and Syst II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.

[5] ——, "MRL—Memristor Ratioed Logic," in *Int'l Workshop Cellular Nanoscale Networks*, 2012, pp. 1–6.

[6] T. You *et al.*, "Exploiting memristive BiFeO3 bilayer structures for compact sequential logics," *Adv Funct Mater*, vol. 24, 06 2014.

[7] P. L. Thangkhiew *et al.*, "An efficient memristor crossbar architecture for mapping Boolean functions using binary decision diagrams (BDD)," *Integr.*, vol. 71, pp. 125–133, 2020.

[8] S. Chakraborti *et al.*, "BDD based synthesis of Boolean functions using memristors," in *Int'l Design & Test Symp. (IDT)*, 2014, pp. 136–141.

[9] J. Bürger, C. Teuscher, and M. Perkowski, "Digital logic synthesis for memristors," *Reed-Muller Workshop*, pp. 31–40, 2013.

[10] A. Fayyazi, A. Esmaili, and M. Pedram, "HIPE-MAGIC: a technology-aware synthesis and mapping flow for highly parallel execution of memristor-aided logic," in *ISLPED*. ACM, 2020, pp. 235–240.

[11] L. G. Amarù, P. Gaillardon, and G. D. Micheli, "Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization," in *Design Automation Conf.*, 2014, pp. 194:1–194:6.

[12] N. Du *et al.*, "Field-driven hopping transport of oxygen vacancies in memristive oxide switches with interface-mediated resistive switching," *Phys Rev Applied*, vol. 10, 11 2018.

[13] B. Hoffer *et al.*, "Experimental demonstration of memristor-aided logic (MAGIC) using valence change memory (VCM)," *IEEE Trans Elec Dev*, vol. 67, no. 8, pp. 3115–3122, 2020.

[14] X. Yang *et al.*, "Novel techniques for memristive multifunction logic design," *Integr.*, vol. 65, pp. 219–230, 2019.

[15] O. Riveros, "SLIME: a minimal heuristic to boost SAT solving," *SAT race*, p. 38, 2019.

[16] S. Kvatinsky *et al.*, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans VLSI*, vol. 22, pp. 2054–2066, 10 2014.

[17] A. Siemon *et al.*, "Stateful three-input logic with memristive switches," *Scientific reports*, vol. 9, no. 1, 2019.

[18] S. G. Rohani and N. TaheriNejad, "An improved algorithm for IMPLY logic based memristive full-adder," in *IEEE Canadian Conf. Elec and Comput Engineering*, 2017, pp. 1–4.

[19] L. Cheng *et al.*, "Functional demonstration of a memristive arithmetic logic unit (MemALU) for in-memory computing," *Adv Func Mat*, vol. 29, no. 49, 2019.

[20] S. Ganjeheizadeh Rohani, N. Taherinejad, and D. Radakovits, "A semi-parallel full-adder in IMPLY logic," *IEEE Trans. VLSI Systems*, vol. 28, no. 1, pp. 297–301, 2020.

[21] H. Schmidt, "Prospects for memristors with hysteretic memristance as so-far missing core hardware element for transfer-less data computing and storage," *Journal of Applied Physics*, vol. 135, no. 20, 2024.