1-2018

A Time-Efficient CMOS-Memristive Programmable Circuit Realizing Logic Functions in Generalized AND-XOR Structures

Muayad Aljafar
Portland State University

Marek Perkowski
Portland State University, marek.perkowski@pdx.edu

John M. Acken
Portland State University

Robin Tan
Portland State University

A Time-Efficient CMOS-Memristive Programmable Circuit Realizing Logic Functions in Generalized AND-XOR Structures

Muayad J. Aljafar, Marek A. Perkowski, John M. Acken, and Robin Tan

***Abstract*— This paper describes a CMOS-memristive Programmable Logic Device connected to CMOS XOR gates (mPLD-XOR) for realizing multi-output functions well-suited for two-level {NAND, AND, NOR, OR}-XOR based design. This structure is a generalized form of AND-XOR logic where any combination of NAND, AND, NOR, OR, and literals can replace the AND level. For mPLD-XOR, the computational delay, which is measured as the number of clock cycles, equals the maximum number of inputs to any output XOR gate of a function assuming that the number of XOR gates is large enough to calculate the outputs of the function simultaneously. The input levels of functions are implemented with novel programmable diode gates, which rely on the diode-like behavior of self-rectifying memristors, and the output levels of functions are realized with CMOS modulo-two counters. As an example, the circuit implementation of a 3-bit adder and a 3-bit multiplier are presented. The size and performance of the implemented circuits are estimated and compared with that of the equivalent circuits realized with stateful logic gates. Adding a feedback circuit to the mPLD-XOR allows the implementation of a multilevel XOR logic network with any combination of sums, products, XORs, and literals at the input of any XOR gate. The mPLD-XOR with feedback can reduce the size and number of computational steps (clock cycles) in realizing logic functions, which makes it well suited for use in communication and parallel computing systems where fast arithmetic operations are demanding.**

***Index Terms*— ESOP structure, parity circuit, programmable logic device (PLD), self-rectifying memristor, stateful logic, volistor logic.**

I. Introduction

IN today's computer architecture, a key problem limiting system speed is the amount of time spent on moving data between the processor and memory. A way to address this problem is to have memory within the calculation circuit. The invention of memristors [1-2] opens the door for changing the computing paradigms of separating calculation from memory. Memristors, as non-volatile devices, enable stateful logic [3-7], hence, saving time spent moving the data between memory and processor. One approach is to use memristor-based stateful logic. The basic operations of IMPLY and FALSE can be implemented in stateful logic. They form a functionally complete logic set and can implement any logic function in any logic system such as AND-OR or SOP (Sum-of-Products), AND-XOR or ESOP (Exclusive-OR-Sum-of-Products), TANT or Three-level-AND-NOT-Network with True inputs [8], AND-OR-XOR Three-Level Networks [9], and so on.

Alternatively, our motivation in this work is to realize arithmetic and communication functions in ESOP form. The ESOP realization of such functions decreases the number of products and literals compared to their realizations in SOP form [10], as described in Section III. The key point in realizing such functions in ESOP form is to implement multi-input XOR gates, which has been a challenge in conventional CMOS technology, i.e., a multi-input XOR gate is slow, consumes large amounts of power, and occupies larger areas compared to other combinational gates. Realizing a multi-input XOR gate with stateful logic would also require a long sequence of stateful operations or a large number of memristors. This disadvantage is due to the fact that XOR gates are only available in two logic levels (such as NAND-OR), for example, when implemented with IMPLY and FALSE operations. Consequently, realizing an n-input XOR gate requires realizing an exponential number of products, i.e., 2^{n-1}. Implementation examples of multi-input XOR gate based on stateful logic are shown in [6][11]. An n-input XOR of literals can be implemented in the same manner described in [6] in a $(2^{n-1}+1) \times (n+1)$ crossbar array with approximately $2n$ computational steps. It can also be realized in the NAND-OR synthesis method [11] in a $1 \times 2n$ crosspoint array in almost 2^{n-1} computational steps.

In addition to the *computational* crossbar arrays considered in [6][11], there are crossbar arrays used as Resistive Random Access Memory (ReRAM) [12] to store control data required for driving the computational arrays. While in our paper we used the ReRAM for control storage, in general it can have other uses. The size of the ReRAM is determined by the number of control bits used for driving the computational array per clock cycle and the number of clock cycles for realizing a function. For example, the size of the ReRAM for driving a $(2^{n-1} + 1) \times (n + 1)$ computational array can be estimated as $((2^{n-1} + 1) + (n + 1)) \times \alpha \times m$ where α is the number of control bits for driving each nanowire of the

computational array per clock cycle, and *m* is the number of computational steps.

This paper proposes a memristive programmable PLA-like circuit for realizing multi-output functions well-suited for two-level {NOR, AND, NAND, OR}-XOR based design. This circuit is called mPLD-XOR, which is memristor-based Programmable Logic Device connected to a CMOS modulo-two counters. The computational delay of the mPLD-XOR, which is measured as the number of clock cycles, equals the maximum number of inputs to any output XOR gates of a multi-output function, assuming that the number of modulo-two counters is large enough to calculate the XOR gates of the function simultaneously. The size of each computational array in mPLD-XOR is at most $1 \times 2n$ where *n* equals the number of primary inputs. The control data required for driving the computational arrays are stored in ReRAM whose size depends on the number of primary inputs *n*, the number of computational steps *m*, and the number of outputs of the function, as described in Section VII.

The mPLD-XOR approach has some advantages over the stateful approach. In the stateful approach, one would first calculate a function by populating the inputs in a computational array and then implement stateful gates. As a result, to calculate the function for a new set of inputs, the computational array must be *cleared* (by the FALSE operation), the new set of inputs must be populated, and the stateful gates must be implemented again. In this process, most of the computational steps are assigned for populating the inputs in the computational array [6]. In contrast, in mPLD-XOR implementation, this long process does not exist since the circuit uses voltage as input. In addition, the computational arrays are programmed only once for calculating a multi-output function for any new set of inputs, as described in Section IV. And the XOR gates are realized in CMOS modulo-two counters as described in Section V.

The mPLD-XOR implements multi-output functions in generalized ESOP forms, i.e., two-level structures where input levels consist of any combination of NAND, AND, NOR, OR, and literals, and output levels are made of only XOR gates. Input levels are implemented using novel programmable diode gates. These programmable gates, which rely on self-rectifying memristors [13-15], have no practical limit on the number of inputs, as described in Section IV. The output levels are realized in CMOS *memory*, i.e., modulo-two counters, and permanently stored in resistive memory using volistor NOT gate [16]. The generalized ESOP structures are good candidates for arithmetic and communication applications.

The paper is organized as follows. In Section II, a review of self-rectifying memristors is provided. In Section III, a generalization to the ESOP structure is introduced. A new approach to realize memristor-based programmable diode gates is proposed in Section IV. In Section V, the mPLD-XOR for realizing circuits in the generalized ESOP structure is proposed, followed by a brief description of mPLD-XOR read and write configuration in Section VI. Section VII shows how to implement a 3-bit adder and 3-bit multiplier in the mPLD-XOR. In addition, the size and performance of the circuits are estimated and compared to their corresponding circuits realized with stateful logic operations. In Section VIII, power, area, and delay of the programmable diode gates and mPLD-XOR are evaluated and compared to previously proposed memristive logic styles for *N*-bit addition operation. Section IX concludes the paper.

II. Self-Rectifying Memristors

In the early 1970's, Leon Chua originally conceived the concept of a memristor (short for memory-resistor) to link electric charge and magnetic flux [1]. The memristor, which is the fourth two-terminal basic element (in addition to resistor, capacitor, and conductor), remained a theoretical concept until researchers at Hewlett-Packard Laboratories successfully created the first device to be called a memristor in 2008 [2]. Unlike a conventional three-terminal transistor, a memristor is a two-terminal non-volatile passive device and retains the advantages of small size, low power consumption, and fast switching time. Crossbar structures have been investigated for a number of applications such as high-density ReRAM, programmable logic, and adaptive neuromorphic circuits [17-20]. The practical issue of the crossbar structures is the crosstalk due to the sneak path currents. These sneak currents are a result of reversely biased cells of the crossbar arrays and should be eliminated. To resolve this problem, Kim *et al.* [13] proposed a self-rectifying memristor or a memristor with intrinsic diode behavior. The self-rectifying memristors suppress the sneak path currents below 1pA due to the large rectifying ratio of the memristors, i.e., from three to six orders of magnitude (10^3-10^6) [13-15]. As a result, the use of self-rectifying memristors decreases the leakage power in crossbar arrays [14][21-22].The multi-bit storage capability of the cells was also reported [13-14]. In this work, a simplified model of self-rectifying memristor described by (1) and (2) is used for performance evaluation [21].

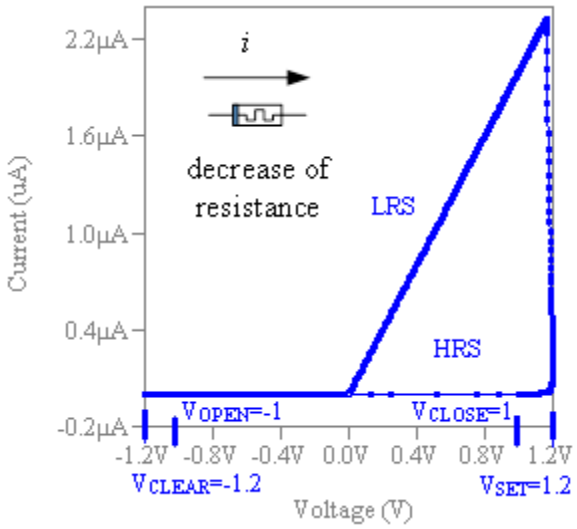
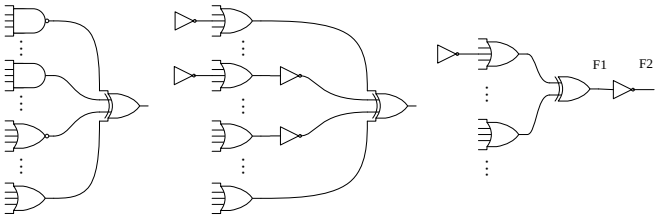


Fig. 1. *i-v* characteristic of the self-rectifying memristor. The device is initialized to HRS and driven by a sinusoidal signal with 1.2V amplitude and 25MHz frequency. The inset shows a symbolic diagram of a memristor. When the voltage exceeds V_{CLOSE}, the flow of current into the device decreases the resistance of the memristor.



(a) (b) (c)

Fig. 2. (a) Schematic of {NAND, AND, NOR, OR}-XOR logic structure. (b) Logical equivalence of {NAND, AND, NOR, OR}-XOR structure as realized in mPLD-XOR. (c) mPLD-XOR realizes functions in NOT-OR-XOR-NOT logic structure.

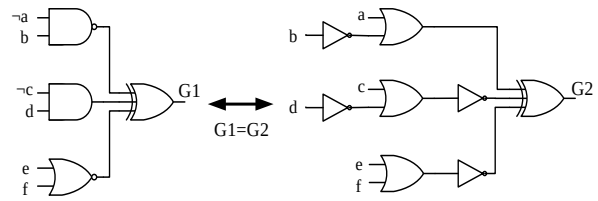


Fig. 3. Function G1 is implemented as its logical equivalence, G2.

$$R = \begin{cases} \mathrm{R_{OPEN}} \left(\frac{\mathrm{R_{CLOSED}}}{\mathrm{R_{OPEN}}}\right)^{s} & v \geq 0 \\ \mathrm{R_{OPEN}} & v < 0 \end{cases} \quad (1)$$

In Equation (1), R represents the resistance of a self-rectifying memristor, s is a state variable normalized to a real number between [0, 1], v is the voltage applied across a memristor, and $\mathrm{R_{OPEN}}$ represents the resistance when a memristor operates at HRS (high resistance state) while $\mathrm{R_{CLOSED}}$ represents the resistance when a memristor operates at LRS (low resistance state). According to the empirical results reported in [14], it is assumed that typical $\mathrm{R_{OPEN}}$ = 500MΩ and $\mathrm{R_{CLOSED}}$ = 500KΩ. The dynamic behavior of the state variable s is described by (2).

$$\frac{ds}{dt} = \begin{cases} \alpha(v - v_{CLOSE}) & v \geq v_{CLOSE} \\ \alpha(v - v_{OPEN}) & v \leq v_{OPEN} \\ 0 & elsewhere \end{cases} \quad (2)$$

Where v_{CLOSE} is a positive threshold voltage, v_{OPEN} is a negative threshold voltage, and $v_{CLOSE} = -v_{OPEN} = 1\mathrm{V}$, as used in [21]. In Equation (2), α is a positive constant associated with the programming rate of memristor and is assumed to be $125 \times 10^7\ (\mathrm{V.s})^{-1}$. When v equals $\mathrm{V_{SET}}$ (a positive programming voltage), the state transition in memristors from HRS to LRS occurs in 4ns, which is comparable with empirical programming rates reported in [23-24]. $\mathrm{V_{SET}}$ is defined as 1.2V, and $\mathrm{V_{CLEAR}}$, which is a negative programming voltage, is defined as –1.2V. Fig. 1 shows the hysteresis behavior of a self-rectifying memristor as found by the LTspice simulator. It illustrates the $i - v$ characteristic of a self-rectifying memristor as described in (1) and (2). The inset shows the symbolic diagram of a memristor. When the voltage exceeds $\mathrm{V_{CLOSE}}$, the current flowing into the device decreases the resistance of the memristor in HRS. The results of the power analysis should be considered an estimation using this simplified model. In this work, the same SPICE model of the self-rectifying memristor proposed in [21] is used for performance evaluation.

TABLE I
NUMBER OF PRODUCTS AND LITERALS OF BENCHMARK FUNCTIONS [10]

	Number of Products				Number of Literals	
	In	**Out**	**SOP**	**ESOP**	**SOP**	**ESOP**
5xp1	7	10	63	32	278	120
9sym	9	1	84	51	504	372
addm4	9	8	189	91	1225	521
adr3	6	4	31	15	116	44
adr4	8	5	75	31	340	112
clip	9	5	117	67	631	402
cm82a	5	3	23	13	80	33
f51m	8	8	76	32	326	112
inc8	8	9	37	15	100	43
life	9	1	84	49	672	311
log8	8	8	123	104	730	550
mlp4	8	8	121	62	736	305
nrm4	8	5	120	69	716	391
rd53	5	3	31	14	140	39
rd73	7	3	127	35	756	134
rd84	8	4	255	59	1774	267
rdm8	8	8	76	32	325	112
rot8	8	5	57	36	305	197
sqr8	8	16	180	112	1068	546
squar5	5	8	25	20	95	57
z4	7	4	59	29	252	111

III. A GENERALIZED ESOP STRUCTURE

Arithmetic functions are well-suited for ESOP-based design. These functions are implemented with a smaller number of products, interconnections, and literals than their counterparts implemented in AND-OR based design [10][25-26]. Table I shows the number of products and literals of benchmark functions implemented in SOP and ESOP based designs using the Quine-McCluskey algorithm for multi-output functions [27] and the EXMIN2 algorithm [25], respectively. The numbers clearly show the advantage of ESOP based design over SOP design for arithmetic functions.

In two-level AND-XOR networks, realizing multi-input XOR gates is essential. A multi-input XOR gate can be implemented as a cascade (or a tree) of two-input XOR gates. However, using a traditional CMOS technology, this approach results in a slow XOR gate [28]. In this study, we use the hybrid CMOS-memristive technology and propose a memristive programmable logic device connected to modulo-two counters (mPLD-XOR) to realize multi-output functions in ESOP based design. The mPLD-XOR can implement the XOR gates with a practically unlimited number of fan-in, which is an advantage over existing technologies (except quantum reversible circuits). The mPLD-XOR also allows the implementation of logic functions in two–level {NAND, AND, NOR, OR}-XOR based design, which is a generalized

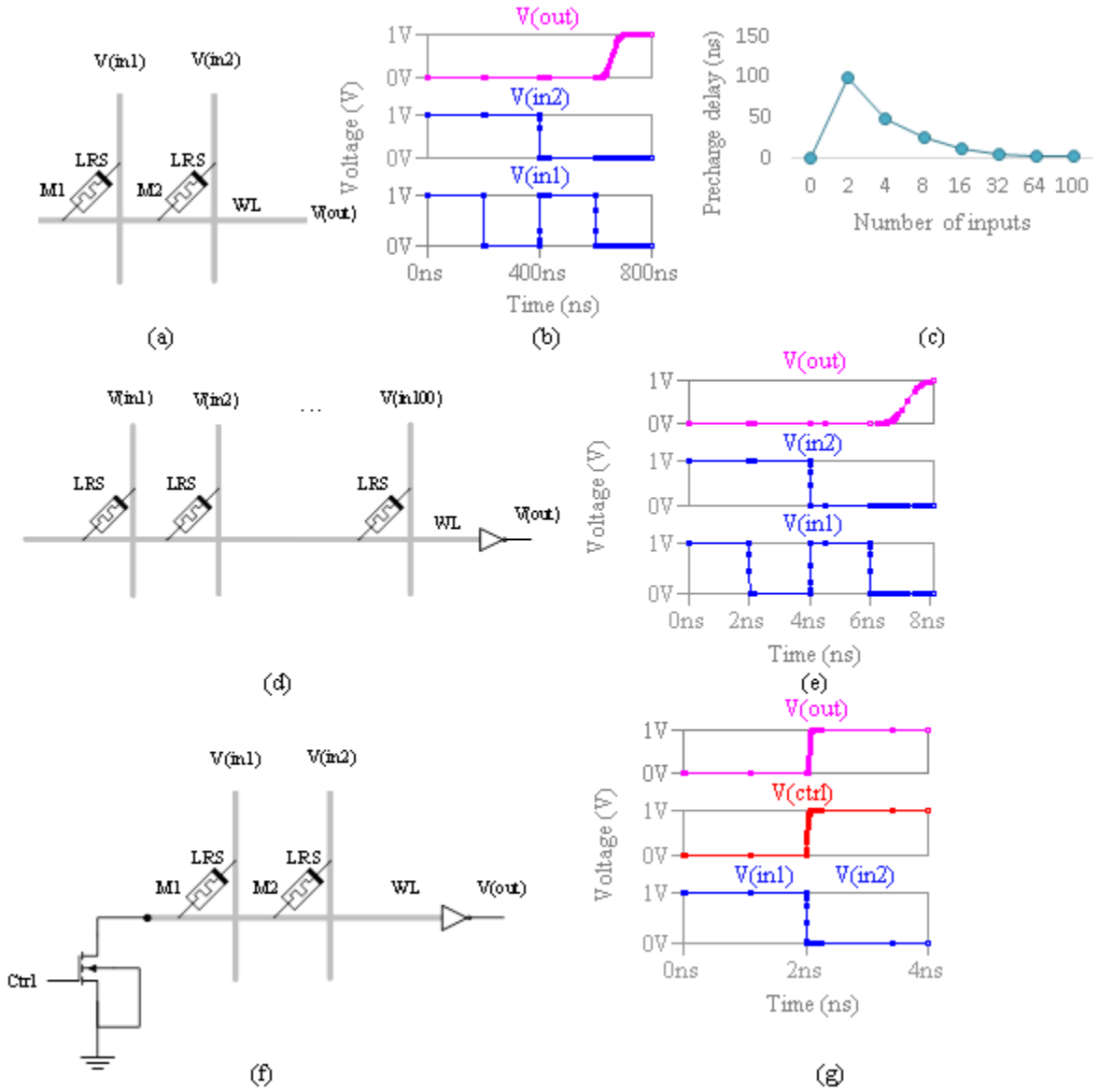


Fig. 4. Memristive programmable diode OR gate. (a) Schematic of a two-input diode OR gate implemented with self-rectifying memristors. (b) Behavior of a two-input diode NOR. (c) Relation between the size of a diode NOR and its RC delay during the precharge interval. (d) Schematic of a 100-input diode NOR gate. (e) Behavior of a 100-input diode NOR gate. (f) Schematic of a two-input diode NOR with pull-down transistor. (g) Behavior of a two-input diode NOR with pull-down transistor.

ESOP logic design. Fig. 2a shows the schematic of a two-level {NAND, AND, NOR, OR}-XOR structure. The mPLD-XOR implements the input level of functions using OR and NOT gates, i.e., NAND is realized as NOT-OR using De Morgan law, $\neg(ab) = (\neg a) + (\neg b)$, NOR as OR-NOT, and AND as NOT-OR-NOT, as shown in Fig. 2b. Note that $\neg a$ denotes the negation of variable a. Inverters at the input of XOR gates can be moved to the outputs. If the number of inverters is odd, the outputs must be inverted , i.e., $\neg a \oplus b = \neg(a \oplus b)$ (output F2 in Fig. 2c). If the number of inverters is even, the outputs are non-inverted, i.e., $(\neg a) \oplus (\neg b) = a \oplus b$ (output F1 in Fig. 2c). Fig. 3 shows an example of a single-output function (G1) in the generalized ESOP structure and its logical equivalence (G2) as implemented using mPLD-XOR.

IV. Memristive Programmable Diode Logic

Since the invention of memristors [2], there have been multiple proposals for memristive logic calculations. Some of them use resistance as input and output [3-7][21][29]. Some others use a mix of voltage and resistance as input and resistance as output [16][30]. Other proposals use voltage as input and output [31-35]. In this work, we propose a novel approach for realizing a diode logic OR and a diode logic AND gate [36], which rely on self-rectifying memristors and use voltage as input and output. These diode gates are used in mPLD-XOR to implement the input level of functions. In contrast to the earlier reported work [34], we propose a significantly different operation of the structurally similar and conceptually different gates.

Fig. 4a shows the schematic of a two–input memristive programmable diode OR gate realized with self-rectifying memristors. The proper operation of the programmable diode OR requires initializing the memristors to LRS. In addition, input voltage v must satisfy the inequalities $0V \leq v \leq V_{CLOSE}$

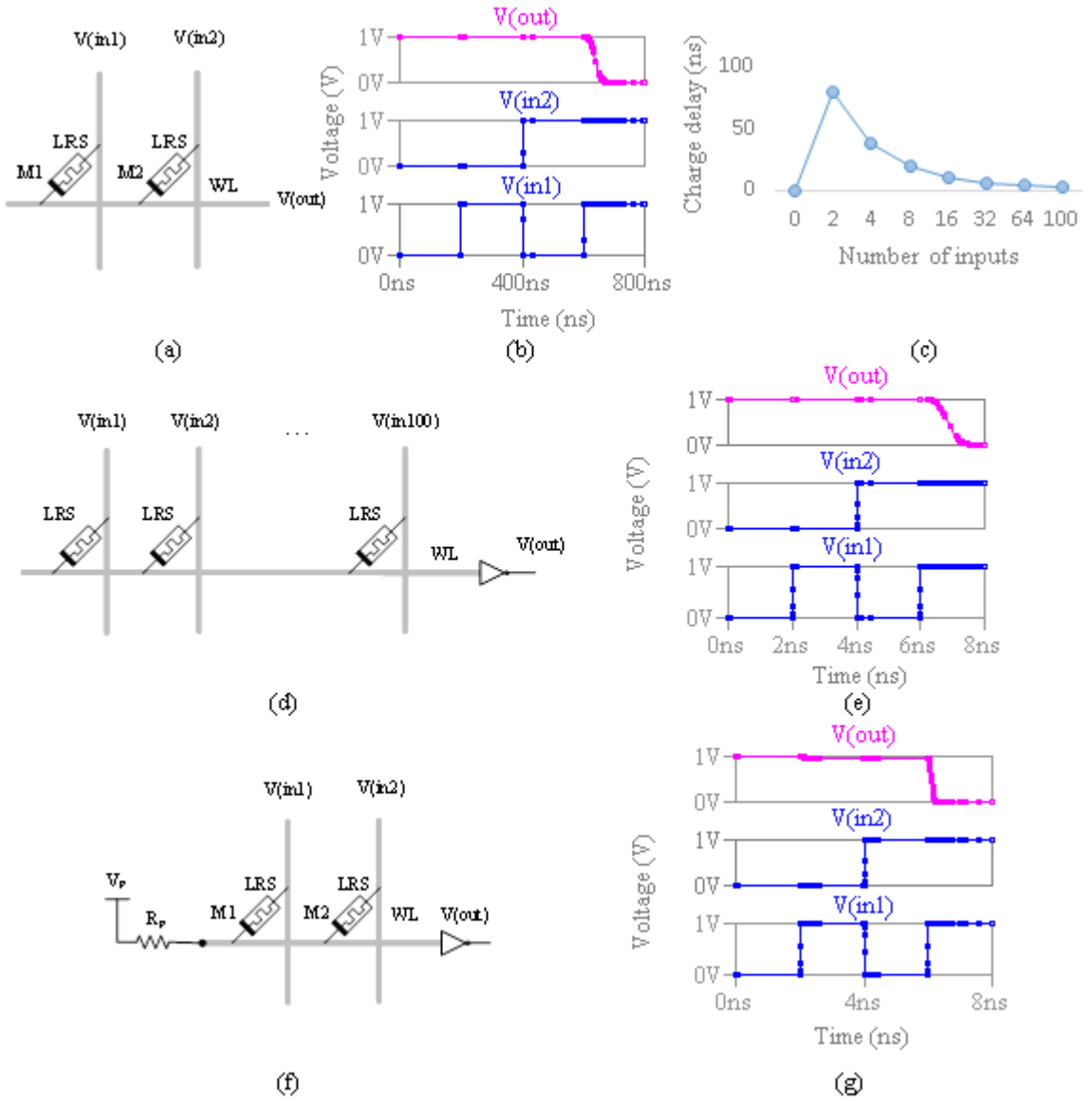


Fig. 5. Memristive programmable diode AND gate. (a) Schematic of a two-input diode AND gate implemented with self-rectifying memristors. (b) Behavior of a two-input diode NAND. (c) Relation between the size of a diode NAND and its RC delay during the charge interval. (d) Schematic of a 100-input diode NAND gate. (e) Behavior of a 100-input diode NAND gate. (f) Schematic of a two-input diode NAND with pull-up resistor. (g) Behavior of a two-input diode NAND with pull-up resistor.

where V_{CLOSE} is the positive threshold voltage (see Fig. 1). This voltage constraint is to ensure that the resistance states of self-rectifying memristors remain unchanged, as described by (2). The input voltage levels are defined as 0V and 1V encoding logic '0' and '1', respectively. Assuming that memristors M1 and M2 in Fig. 4a are connected to high and low input voltages, respectively, the behavior of the programmable diode OR gate can be described by $\Delta v = (R_{OPEN} + R_{CLOSED}) \cdot i$ where Δv is the input voltage difference, and i is the current through memristors. The current i is limited below 1pA by the reverse biased memristor M2. Therefore, the voltage drop across memristor M1, which is in a forward bias situation, is very small, and the output approximately equals V(in1). The behavior of an n-input programmable diode OR can be explained by (3) where j and k are the numbers of memristors set to low and high voltage levels, respectively. While all memristors are initialized to LRS, they exhibit different resistance states depending on their input voltages.

$$\begin{cases} \Delta v = \left(\frac{R_{OPEN}}{j} + \frac{R_{CLOSED}}{k}\right) \cdot i \\ n = j + k \end{cases} \quad (3)$$

Memristors set to high input voltage are forward biased and exhibit LRS, while memristors set to low input voltage are reverse biased and exhibit HRS.

The programmable diode OR without an output load operates in the range of 3ps. However, to be realistic, the behaviors of a diode NOR (diode OR connected to a CMOS inverter) is shown in Fig. 4b where V(in1) and V(in2) are the inputs and V(out) is the output. Connecting the output of the diode OR (WL) to a CMOS inverter causes unequal RC delays during the charge and precharge intervals. During the charge interval, memristors are forward biased and the RC delay is in the range of 100ps. However, during the precharge interval,

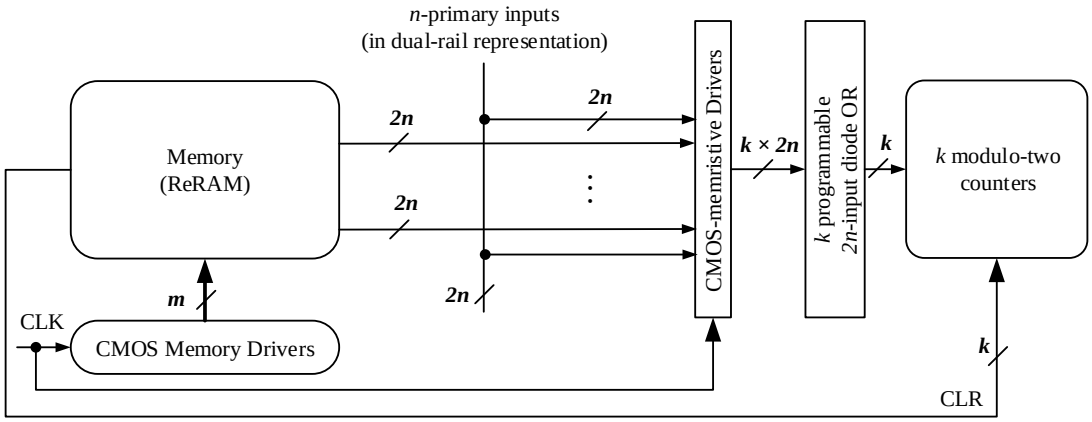


Fig. 6. Schematic of mPLD-XOR.

memristors are reverse biased, and the RC delay is in the range of 125ns. For large diode NORs, this delay reduces to a few nanoseconds as shown in Fig. 4c. Fig. 4d shows the schematic of a 100-input diode OR connected to a CMOS inverter (diode NOR). The behavior of the gate is shown in Fig. 4e where half of the inputs are the same as V(in1), and the other half are the same as V(in2). The precharge delay is in the range of 2ns. In addition, the use of a pull-down network further decreases the precharge delay. Fig. 4f shows the schematic of a two-input diode NOR with pull-down transistor connected to the output of the diode-OR gate, WL. The behavior of the NOR gate is shown in Fig. 4g where signal Ctrl controls the pull-down transistor. The precharge delay of the NOR gate is in the range of 125ps.

The diode gate memristors, which are initialized to LRS, act as binary switches while maintaining their resistance states. A memristor driven by the high input voltage acts as a *closed* switch, while a memristor driven by the low input voltage acts as an *open* switch. Clearly, this role cannot be played by standard memristors.

Similar to the programmable diode OR gate, a programmable diode AND gate can be implemented with self-rectifying memristors. Fig. 5a shows the schematic of a two–input programmable diode AND gate. Fig. 5b illustrates the behavior of the diode AND connected to a CMOS inverter (diode NAND). Fig. 5c shows the relation between the size of a diode NAND gate (the number of inputs) and its RC delay during the charge interval. Fig. 5d shows the schematic of a 100-input programmable diode AND gate. Fig. 5e illustrates the behavior of a 100-input programmable diode NAND gate where half of the inputs are the same as V(in1), and the other half are the same as V(in2). Fig. 5f shows the schematic of a two-input diode NAND with a pull-up resistor connected to the output of the diode AND gate, WL. The behavior of the NAND gate is shown in Fig 5g. Note that the RC delay during the charge interval depends on the value of pull-up resistor Rp, e.g., for Rp=1.25MΩ the delay is in the range of 400ps.

The i-*v* characteristic of a self-rectifying memristor can be modeled as a 1D1R (1diode in series with 1 resistive RAM) structure incorporated into a single two-terminal device [13]. A LRS memristor in reverse biased situation acts as an open switch and exhibits a high resistance state. Therefore, a self-rectifying memristor reduces the reverse bias leakage, as such it allows to implement diode gates with many inputs as shown in Fig. 4d and Fig. 5d. Clearly, the PN junction or Schottky diode in CMOS process does not act as open switches when they are in reverse bias situation, hence they cannot take the role of the self-rectifying memristors in large diode gates used in the mPLD-XOR (see Section V).

V. The mPLD-XOR: Circuit Structure and Functionality

The programmable diode logic gates introduced in Section IV are used in a new circuit structure called mPLD-XOR to realize multi-output functions in {NAND, AND, NOR, OR}–XOR based design. Fig. 6 shows the schematic of mPLD-XOR. The device consists of a ReRAM connected to CMOS drivers, memristive programmable diode OR gates connected to CMOS-memristive drivers, and CMOS modulo-two counters. The ReRAM is a crossbar array of self-rectifying memristors, which stores all control data required to drive the circuit. Inputs are applied to programmable diode OR gates through CMOS-memristive drivers controlled by data stored in the ReRAM. The output of each programmable diode OR gate is applied to a modulo-two counter, which acts as a parity circuit (or a T flip-flop). The counter is functionally equivalent to an XOR gate with an arbitrary number of inputs. The output of the counter is applied to a crosspoint array through a transmission gate and stored as the state of a memristor, as illustrated in Fig. 10 in Section VII. The assertion of signal CLR resets the counter and makes it available for calculating another output of a function.

Fig. 7 includes the symbolic diagram of a modulo-two counter as a D flip-flop where D = ¬Q. The counter is clocked by an *l*-input programmable diode OR gate. The circuit implements an *n*-input single-output function in {NAND, AND, NOR, OR}–XOR based design. Inputs In**i** where **i** ∈ $\{1,\cdots,n\}$ and their complements are applied to a 1×*l* programmable diode OR through CMOS-memristive drivers where *l*=2*n*. The drivers consist of 3-input programmable diode AND gates connected to CMOS buffers. When CLK is high, a combination of inputs determined by high C**j** where **j** ∈ $\{1,\cdots,l\}$ is propagated to the diode OR gate; C**j** are control

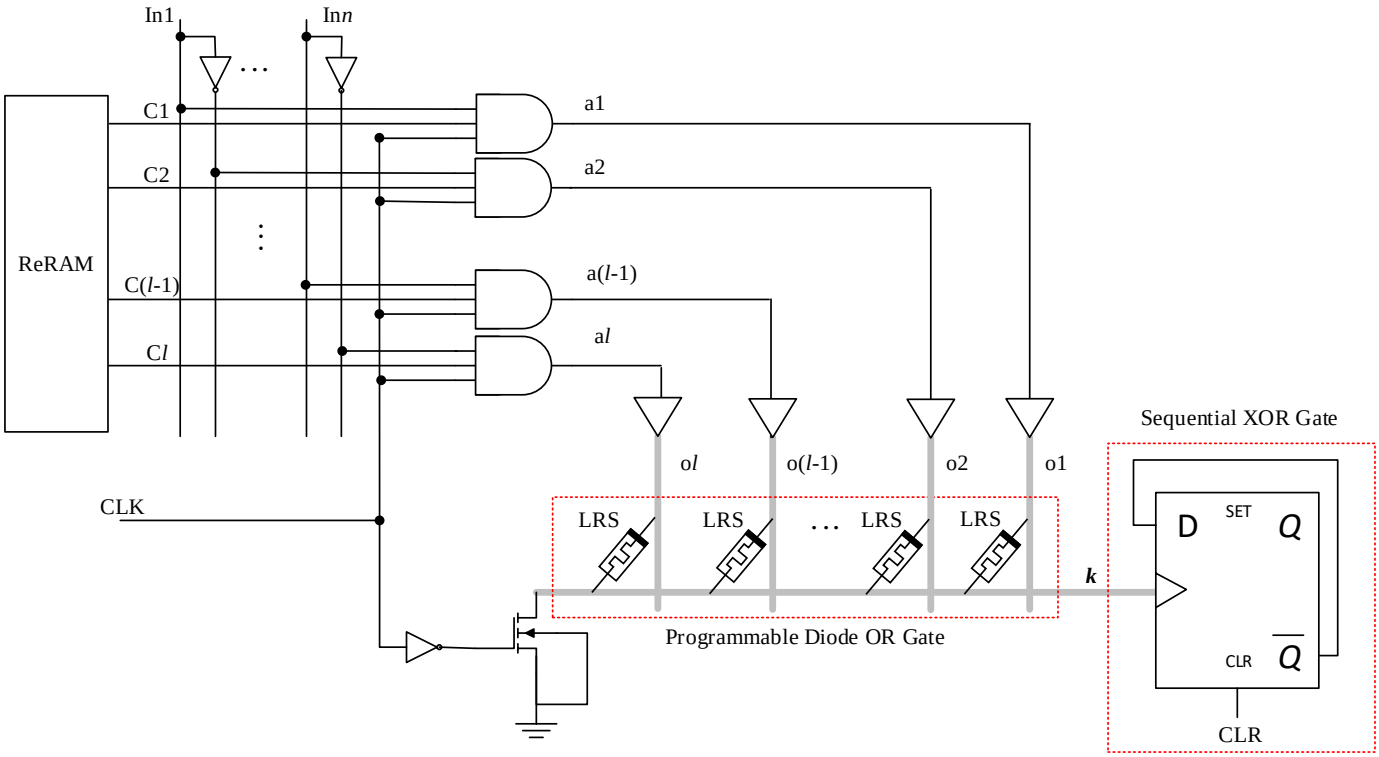


Fig. 7. Schematic of an mPLD-XOR for realizing n-input single-output functions with l lines diode OR where l=2n to allow for complemented inputs. In**i** are the primary inputs where **i** $\in \{1,\cdots,n\}$, and C**j** are the control signals stored in the ReRAM where **j** $\in \{1,\cdots,l\}$. The output of the circuit is either Q or $\overline{Q}$ depending on the function being implemented.

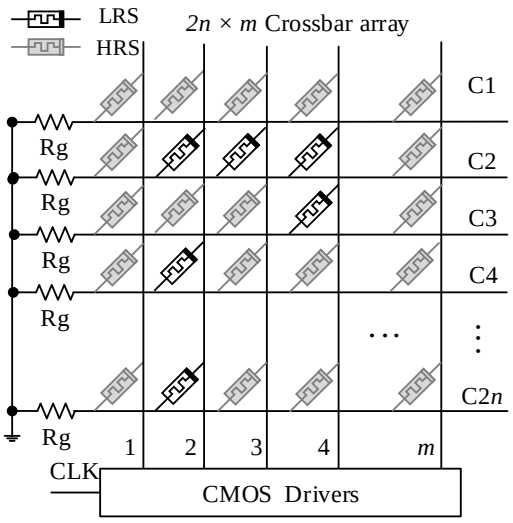


Fig. 8. Schematic of the ReRAM used in Fig. 7 with reference resistors Rg.

signals stored in each column of the ReRAM and applied to the AND gates at positive edge of signal CLK. The output of the circuit is Q, the current state of the counter (or ¬Q depending on the function being implemented) and is available after m clock cycles where m also denotes the number of XOR terms, i.e., m-input XOR of sums, products, or literals.

Fig. 8 shows the schematic of ReRAM with reference resistors Rg. The ReRAM is connected to CMOS drivers made of an m-bit shifter. The output of the shifter, (Q_1, Q_2… Q_m), at the first, second, and m^{th} clock cycle is (1, 0, …, 0), (0, 1, 0, …, 0), and (0, 0, …, 0, 1), respectively, where logic '1' and '0' denote high and low voltage levels, which are 0.6V

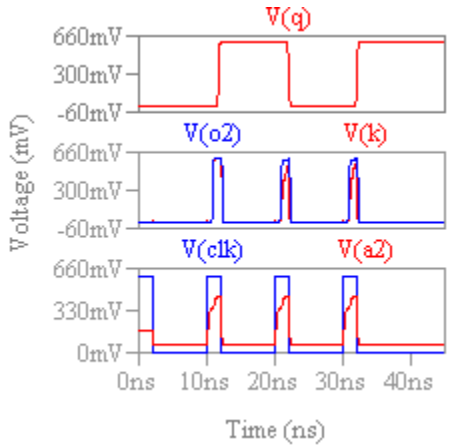


Fig. 9. Simulation results of the mPLD-XOR circuit shown in Fig. 7 with the ReRAM configuration shown in Fig. 8.

and 0V. The programmable circuit shown in Fig. 7 is scalable, and it can be a base of future memristive programmable fabrics to realize large arithmetic circuits and circuits with high XOR component. Examples of multi-output mPLD-XOR circuits are shown in Section VII.

The circuit in Fig. 7 is designed using a 50nm TSMC process BSIM4 models where the number of inputs is 32, and the size of ReRAM is 64 × 16. For this example, all of the inputs In**i** are assigned a value of '0'.¬In**i** represents the output of inverters with input In**i**. The ReRAM memristors are programmed as shown in Fig. 8. The pale shaded memristors are at HRS, and clear ones are at LRS. Fig. 9 shows the simulation results where V(clk) represents the clock cycle, V(a2) represents the output of the second AND gate, V(k) represents the output of the programmable diode OR, and V(q)

represents the output of the modulo-two counter (see Fig. 7). During the first clock cycle, the first column of the ReRAM shown in Fig. 8 is connected to '1' to read the control data. Since all memristors are in HRS, none of the inputs are applied to the programmable diode OR. At the second clock cycle, the second column is set to '1' and inputs ¬In**i** are applied to the diode OR. During the second, third, and fourth clock cycles, V(k) toggles the counter three times. As a result, the output of the counter remains high. The simulation results in Fig. 9 are based on Rg = 1MΩ, R_P = 3.5MΩ, V_P =0.6V, and input voltages defined as 0V and 0.6V where Rp is a pull-up resistor used in diode AND gates and is connected to voltage Vp.

VI. PROGRAMMING THE MPLD-XOR

The general writing to a ReRAM is described in many papers [14-15]. Programming the memristors requires the application of V_{SET} or V_{CLEAR} across the devices. This programming step is performed only once since the memristors maintain their resistance states during logical operations. The results of our simulations show that the sneak path current during the read operation is small due to the diode-like behavior of the memristors. For example, the sneak path current is between 80pA and 240pA depending upon its location in the array, which is small enough that it has no negative impact on memristors in sneak paths during the read operation. More analysis about the sneak path current during the read operation can be found in [22]. For vary large crossbar arrays, Opamp threshold logic can be used for the read operation [32]. In addition, using asymmetric voltage scheme further decreases the leakage power during the programming [15].

VII. IMPLEMENTATION EXAMPLES IN MPLD-XOR

In previous sections, the generic fabric of the mPLD-XOR and its programming were explained. As described in Section V, the mPLD-XOR can implement any logic functions. As an example, this section shows how a 3-bit adder and a 3-bit multiplier are mapped into this fabric. The size and performance of these circuits are also estimated. For comparison purposes, the same adder and multiplier are also implemented with stateful gates, and the size and performance of the circuits are estimated. Our calculations show that the size of the ReRAM and the number of computational steps for realizing a 3-bit adder in the mPLD-XOR circuit to the stateful circuit are 280:957 and 8:29, respectively. For a 3-bit multiplier, these ratios are 564:5720 and 12:65, respectively. In both approaches, the size of the ReRAM (including the CMOS drivers) dominates the area when large circuits are implemented. The implementation details can be found in Section VII.A and VII.B.

A. 3-bit adder

Below we will illustrate how a 3-bit adder can be realized in a simplified generic mPLD-XOR fabric, where inputs in a complementary form only are applied to the circuit. This simplification decreases the amount of the data stored in ReRAM to half. The size of the ReRAM is 64×16, and the number of modulo-two counters is three, which allows the calculation of three single-output functions simultaneously. The schematic of the mPLD-XOR for realizing the adder is shown in Fig. 10. Inputs are a**i** and b**i** where **i** ∈ {0, 1, 2}, and outputs are S0, S1, S2, and Co as described by (4). Instructions for realizing the adder are loaded into the ReRAM. Note that realizing any logic function requires loading the corresponding instructions into the ReRAM.

Fig. 11 shows a part of the ReRAM divided into subReRAMs where each stores instructions to calculate a 1-bit output, except the subReRAM shown in Fig. 11a, which stores control data for driving the CMOS part of the mPLD-XOR. The size of subReRAMs is 35 × 8.

Once an output is calculated, it is stored as a state of a memristor using volistor NOT gate [16]. Volistor NOT uses voltage as input and resistance as output. To describe the operation of a volistor NOT gate, consider the circuit shown in Fig. 4a. Let us assume that an input voltage is applied to memristor M1, and a negative bias defined as – 0.6V is applied to memristor M2. When the input is '1' (0.6V), the voltage across memristor M2 is –1.2V, i.e., V_{CLEAR}, which is sufficient to toggle the state of memristor M2 to HRS (logic '0'). When the input is '0' (0V), the state of memristor M2 remains at LRS (logic '1'). Memristor M1 is called 'source memristor' since it is driven by the input, and memristor M2 is called 'target memristor' since it stores the output [5]. Regardless of the input value, the state of the source memristor remains at LRS while the state of the target memristor may toggle to HRS depending on the input. Note that the role of each memristor is determined by the voltage applied to the memristor. The proper operation of the volistor NOT requires initializing memristors M1 and M2 to LRS. The crosspoint array at the bottom right of the mPLD-XOR in Fig. 10 is used to store the outputs of the counters using volistor NOT. The crosspoint memristors connected to counters' dual-rail outputs via transmission gates act as source memristors (SM), and the crosspoint memristors connected to TM drivers act as target memristors (TM). The TM drivers are made of a CMOS shifter with output voltage levels of 0V and –0.6V. The transmission gates connect the outputs of the counters to the source memristors only when the outputs have been calculated. The transmission gates are controlled by signals P**i** and ¬P**i** where **i** ∈{1, …, 2**j**} and **j** equals the number of counters, which is 3 (see Fig. 10). Recall that each counter has two outputs, Q and ¬Q, and thus, requires two transmission gates. The need for representing the outputs of the counters in dual-rail logic is described in Section III.

The first column of the subReRAM shown in Fig. 11a stores control data for initializing the CMOS subcircuits of the mPLD-XOR, e.g., by asserting signals CLR**i** where **i** ∈ {0, 1, 2, 3} to reset all modulo-two counters, or by asserting signals P**i** and ¬P**i** where **i** ∈ {1, …, 6} to disconnect the modulo-two counters connected to the crosspoint array via transmission

$$\begin{cases} S0 = a0 \oplus b0 \\ S1 = a1 \oplus b1 \oplus a0b0 \\ S2 = a2 \oplus b2 \oplus a1b1 \oplus a1a0b0 \oplus b1a0b0 \\ Co = a2b2 \oplus a2a1b1 \oplus a2a1a0b0 \oplus a2b1a0b0 \oplus b2a1b1 \oplus b2a1a0b0 \oplus b2b1a0b0 \end{cases} \quad (4)$$

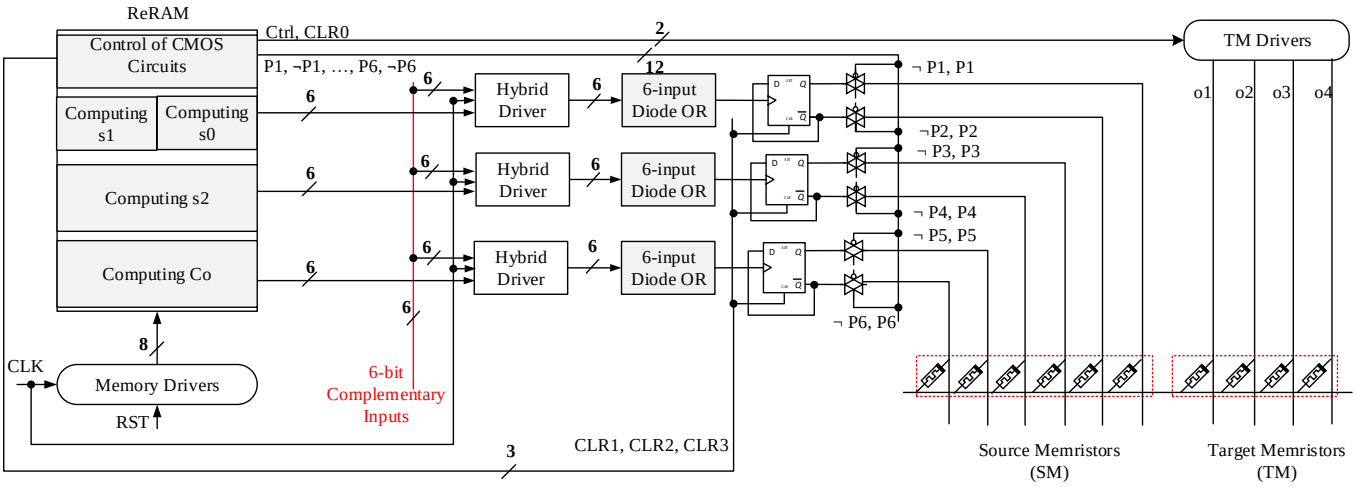


Fig. 10. Generic fabric of the mPLD-XOR for realizing a 3-bit adder. The grey rectangles correspond to memristive arrays, the white rectangles correspond to hybrid CMOS-memristive circuits, and the rest of the blocks correspond to the CMOS circuits.

$$\begin{cases} c_i = NAND\ (OR\ (\neg a, \neg \mathrm{b}\ ), OR(\neg a, \neg c_{i-1}), OR(\neg \mathrm{b}, \neg c_{i-1})) \\ s = NAND\ \big(OR\ (a, b, \neg c), OR(a, \neg \mathrm{b}, c), OR(\neg a, \neg \mathrm{b}, \neg c), OR(\neg a, b, c)\big) \end{cases} \quad (5)$$

gates. Once an output of the adder is calculated, signal Ctrl is asserted to store the output in a target memristor.

For example, upon the first assertion of signal Ctrl, the first output (S0) is stored in the leftmost target memristor of the crosspoint array shown in the bottom right of Fig. 10. In this operation, the outputs of TM drivers, (o1, o2, o3, o4), equal (–0.6, 0V, 0V, 0V). Ultimately all outputs will be stored in target memristors TM. Each column of subReRAMs in Fig. 11b-Fig. 11d controls the primary inputs applied to each programmable diode OR gate for realizing an XOR term such as NAND, AND, NOR, OR, or literal. (In particular, each row of the subReRAMs controls 1-bit input during computational steps where inputs are shown as a red vertical bus adjacent to the subReRAMs). The process of realizing the 3-bit adder are described below.

1) Initialization

CMOS subcircuits are initialized in the first clock cycle by asserting the control signals stored in the first column of ReRAM.

2) Calculating S0

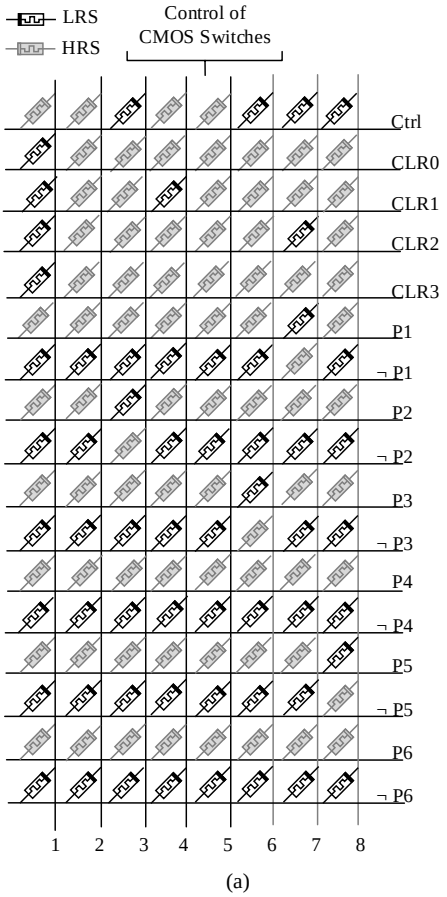
Sum bit S0 is calculated in the second and third clock cycles by driving the second and third columns of the crossbar array shown in Fig. 11b. In the third clock cycle, S0 is available at the non-inverted output of XOR1. At this moment, ¬S0 is applied to a source memristor via a transmission gate to store S0 in the leftmost target memristor of the crosspoint array. In this operation, signals P2 and Ctrl stored in the third column of the crossbar array in Fig. 11a are applied to the related transmission gate and TM drivers, respectively. In the fourth clock cycle, signal CLR1 is asserted to clear the output of XOR1 for the next use.

3) Calculating S2

Sum bit S2 is calculated in five clock cycles, as shown in Fig. 11c. In clock cycle number six, S2 is available at the inverted output of XOR2. At this moment, ¬S2 is applied to a source memristor, and S2 is stored next to S0. This operation requires asserting signals P3 and Ctrl, which are stored in column number six of the crossbar array in Fig. 11a.

4) Calculating S1

Sum bit S1 is calculated in three clock cycles, as shown in Fig. 11b. In clock cycle number seven, S1 is available at the inverted output of XOR1. Simultaneously, ¬S1 is applied to a



(a)

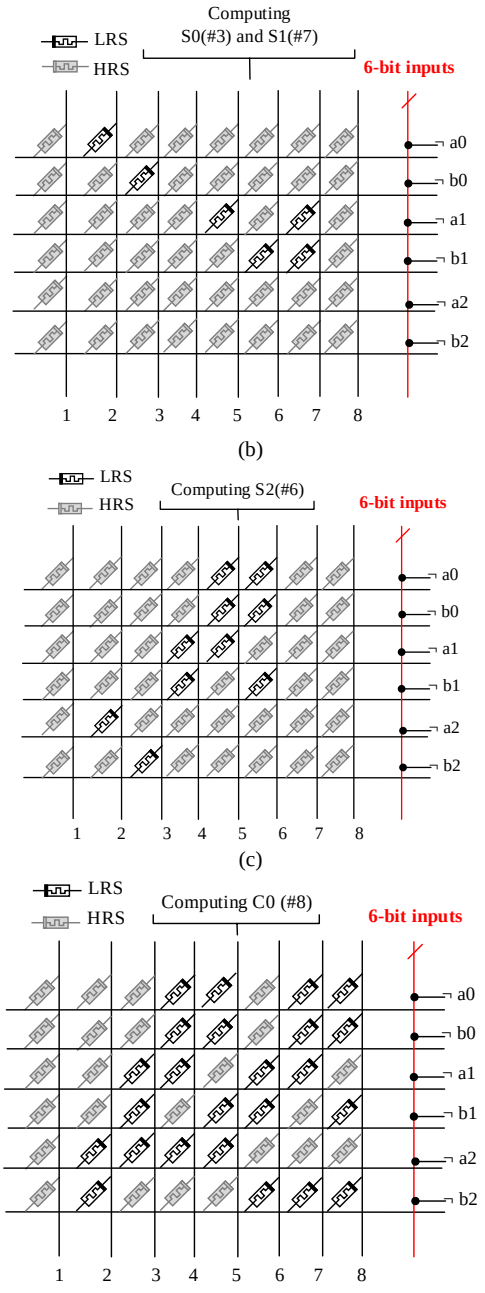


Fig. 11. Schematic of subReRAMs. SubReRAM (a) stores the control data for driving CMOS parts of the mPLD-XOR. SubReRAMs (b)-(d) store the control data for realizing S0, S1, S2, and Co. The number of clock cycles for computing each output is shown in parenthesis, e.g., (#8). The size of subReRAMs is 35 × 8.

source memristor to store S1 next to S2 by asserting signals P1 and Ctrl, stored in column number seven of the crossbar array in Fig. 11a.

5) Calculating Co

Carry bit Co is calculated in seven clock cycles, as shown in Fig. 11d. In clock cycle number eight, Co is available at the inverted output of XOR3. At this moment, ¬Co is applied to a source memristor to store Co next to S1 by asserting signals P5 and Ctrl, stored in column number eight of the crossbar array in Fig. 11a.

a0	b0	0	0	0	0	0	0
a1	b1	0	0	0	0	0	0
a2	b2	0	0	0	0	0	0

(a) 1 FALSE

a0	b0	¬a0	¬b0	0	0	0	0
a1	b1	¬a1	¬b1	0	0	0	0
a2	b2	¬a2	¬b2	0	0	0	0

(b) 2 IMPs

a0	b0	¬a0	¬b0	0	0	0	c1
a1	b1	¬a1	¬b1	0	0	0	¬c1
a2	b2	¬a2	¬b2	0	0	0	0

(c) 2 IMPs

a0	b0	¬a0	¬b0	0	0	0	0
a1	b1	¬a1	¬b1	c1	¬c1	0	0
a2	b2	¬a2	¬b2	0	0	0	0

(d) 2 IMPs + 1 FALSE

a0	b0	¬a0	¬b0	0	0	0	s0
a1	b1	¬a1	¬b1	c1	¬c1	0	0
a2	b2	¬a2	¬b2	0	0	0	0

(e) 2 IMPs

a0	b0	¬a0	¬b0	0	0	0	s0
a1	b1	¬a1	¬b1	c1	¬c1	0	c2
a2	b2	¬a2	¬b2	0	0	0	¬c2

(f) 4 IMPs

a0	b0	¬a0	¬b0	0	0	0	s0
a1	b1	¬a1	¬b1	c1	¬c1	0	0
a2	b2	¬a2	¬b2	c2	¬c2	0	0

(g) 2 IMPs + 1 FALSE

a0	b0	¬a0	¬b0	0	0	0	s0
a1	b1	¬a1	¬b1	c1	¬c1	0	s1
a2	b2	¬a2	¬b2	c2	¬c2	0	0

(h) 4 IMPs

a0	b0	¬a0	¬b0	0	0	0	s0
a1	b1	¬a1	¬b1	0	0	0	s1
a2	b2	¬a2	¬b2	c2	¬c2	c3	s2

(i) 7 IMPs

a0	b0	¬a0	¬b0	0	0	0	s0
a1	b1	¬a1	¬b1	0	0	0	s1
a2	b2	¬a2	¬b2	0	0	c3	s2

(j) 1 FALSE

Fig. 12. (a)-(j) Computational steps for realizing a 3-bit adder with stateful gates. The total number of IMP and FALSE operations is 29. The numbers of operations are shown in each step.

In summary, the 3-bit adder is realized in eight clock cycles in a 35 × 8 crossbar array.

The mPLD-XOR can be designed by programmable diode ANDs instead of programmable diode ORs to realize logic

functions in PPRM (Positive Polarity Reed-Miller) expressions that is AND-XOR expressions with uncomplemented literals only, as in (4). This realization also requires changing the diode AND gates used in CMOS-memristive drivers with diode OR gates. In addition, the control data stored in ReRAM (Fig. 11) must be substituted with their complements. In the AND-type mPLD-XOR, inputs in positive polarity are applied to the circuit, and there is no need for input CMOS inverters.

Using memristor-based stateful IMPLY gates, a 3-bit adder can be implemented with the NAND-OR logic structure [11] using (5). In this implementation, multi-input IMPLY gates are utilized. The process of realizing a 3-bit adder in a 3 × 8 computational array is shown in Fig. 12. More details can be found in [37].

The matrix shown in Fig. 12a is analogous to a 3 × 8 crossbar array where entities of the matrix represent the states of corresponding memristors. Inputs a**i** and b**i** where **i** ∈ {0, 1, 2} are populated in the computational array shown in Fig. 12a. In step **a**, the auxiliary memristors are initialized to HRS denoted by logic '0' in one clock cycle. In step **b**, copies of complementary inputs are stored in the third and fourth columns using stateful IMPLY gates. This step is realized in two clock cycles. In step **c**, carry bit c1 is calculated and stored in the rightmost memristor of the first row. Also, a copy of ¬c1 is stored in the rightmost memristor of the second row. In step **d**, c1and ¬c1 are copied to new locations, as shown in Fig. 12d. The rightmost column is cleared to be used in step **e**. In step **e**, sum bit s0 is calculated and stored in the rightmost memristor of the first row. The following steps, **f** through **j**, are illustrated in Fig. 12.

The number of computational steps is 29, and the size of the ReRAM can be estimated as $c \times m$ where c is the number of control bits driving the 3×8 crossbar array per computational step, and m equals the number of computational steps, which is 29. The number of control bits driving each nanowire cannot be smaller than three since each nanowire needs to be connected to multiple voltage levels, grounded through a reference resistor, or terminated to high impedance. Assuming that each nanowire is driven by three control bits, the size of the ReRAM can be estimated as 33× 29.

In an mPLD-XOR, the number of control bits for driving each nanowire of the computational arrays is only one because each nanowire must be connected to either '0' or '1' for computing logic. In other words, there is no need to terminate the nanowires to high impedance or to ground them through reference resistors. Moreover, the memristors maintain their resistance states during logic calculations and they don't need to be initialized repeatedly. Therefore, no additional voltage levels are required for implementing logic in mPLD-XOR. The programmable diode gates have simplified driving circuitries, which in turn decrease the size of ReRAM. The size of the ReRAM and the number of computational steps for realizing the adder in the mPLD-XOR circuit to the stateful circuit are 280:957 and 8:29, respectively.

B. 3-bit multiplier

A 3-bit multiplier can be realized with the mPLD-XOR shown in Fig. 10. This requires loading instructions of the multiplier into the ReRAM, which occupy 35×40 of the ReRAM size. The number of clock cycles (computational steps) for this realization is 40. Adding feedback CMOS D flip-flops to the mPLD-XOR can improve the size and performance of the circuit. For example, the multiplier can be implemented in twelve clock cycles using 47×12 of the ReRAM size. The enhanced mPLD-XOR stores the output of each counter in a D flip-flop and makes it available as a primary input (Fig. 13). This feedback also makes a counter available for calculating another output. The feedback D flip-flops are clocked by signal Sig**i** where **i** ∈ {1, 2, 3}. Adding the feedback to the mPLD-XOR allows the implementation of multilevel XOR logic networks with any combination of sums, products, XORs, and literals at the input of any XOR gate. Fig. 14 shows the schematic of a 3-bit multiplier as realized with the mPLD-XOR with feedback D flip-flops. In this realization, internal signals IP0, IC0, IC1, and IC2 in Fig. 14 are calculated, stored in D flip-flops, and used to implement the next levels of the circuit.

If the multiplier is implemented with stateful IMPLY gates, the number of computational steps will be 65. Fig. 15 shows the initial states of the crossbar array for realizing the multiplier. Assuming that each nanowire of the crossbar array is driven by three control bits, the size of the ReRAM is estimated as 66 × 65.

In summary, the size of the ReRAM and the number of computational steps for realizing a 3-bit multiplier in the mPLD-XOR circuit to the stateful circuit are 564:4290 and 12:65, respectively.

VIII. Evaluation and Comparison of Different Logic Styles

An extensive comparison of different architectures is beyond the scope of this paper, however, to get some level of comparison, here are some published numbers for other memristive styles of logic gates. The following is a direct comparison between the published threshold logic example [32], MAGIC NOR example [7], and our implementation, and as such does not include the driving circuitry to the memristor array (see Table II). The minimum and maximum power dissipations of 2, 10, and 100-input diode NAND/ NOR gates were calculated. The same power calculation was also performed for multi-input stateful NOR gates realized with self-rectifying memristors using converse non-implication gates (CNIMP) [6]. The simulation was performed in LTspice using 50nm TSMC process BSIM4 models and the memristor model explained by (1) and (2). The power dissipation and propagation delay are calculated when a square pulse with 10ns time period and a 50% duty cycle is applied. The initial states of memristors realizing the CNIMP NOR gates are assumed to be '0' (HRS). There are two steps for performing the stateful NOR gate. The first step is programming the memristors, and the second step is performing the logic. Most of the gate power consumption is related to the programming step (writing input values into memristors), which must be performed repeatedly for computing different logic functions. The initial states of the memristors in the programmable diode gates are assumed to be '1' (LRS). In contrast to the stateful gates, memristors in programmable diode gates maintain their resistance states during logic operations. The power

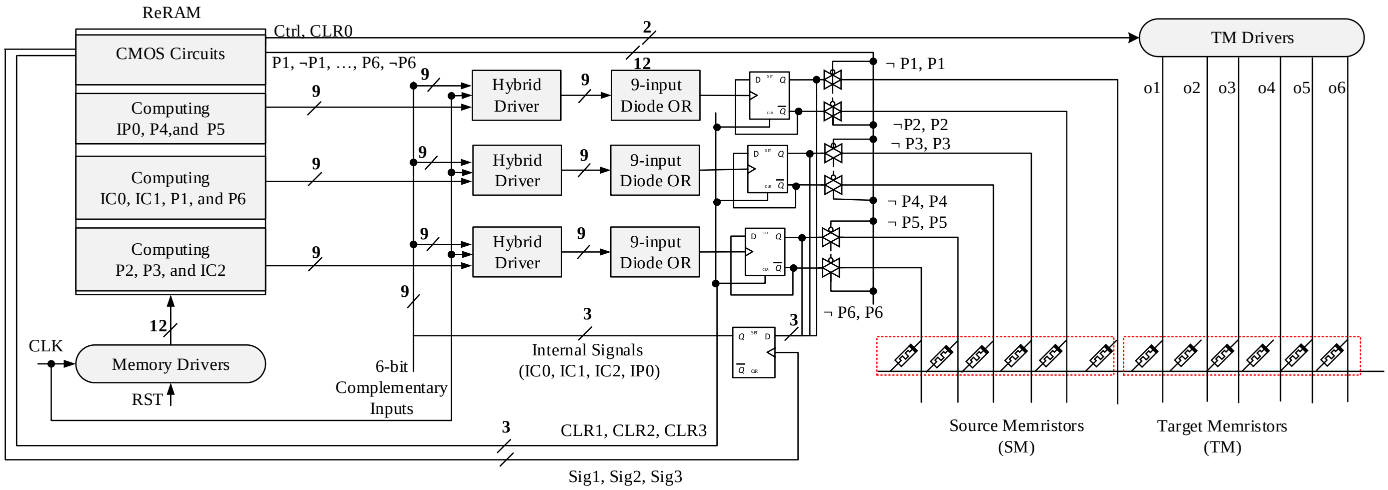


Fig. 13. Schematic of an mPLD-XOR with feedback D flip-flops for realizing a 3-bit multiplier. In this implementation, the number of computational steps is 12, and the instructions occupy 47×12 of the ReRAM.

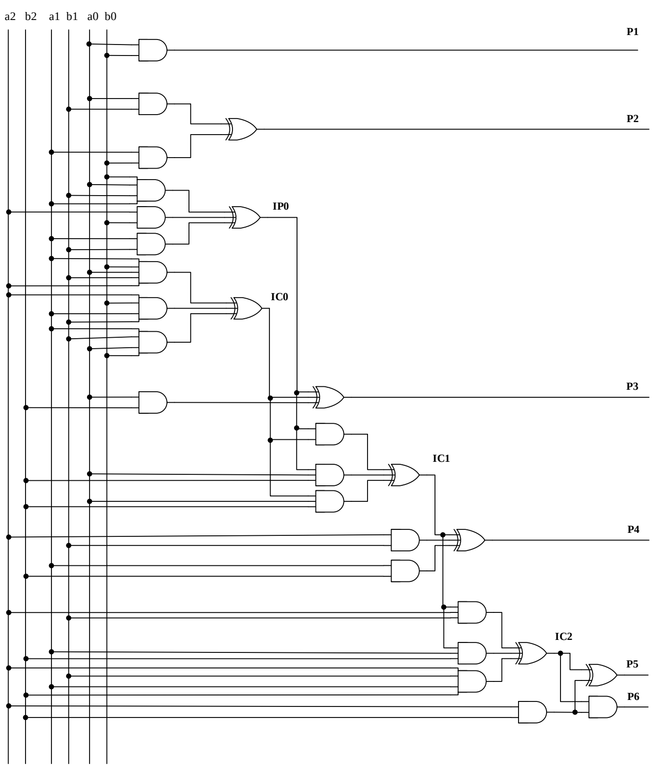


Fig. 14. Schematic of a 3-bit multiplier in multilevel XOR logic structure where XOR terms are sums, products, XORs, or literals. This circuit is implemented with an mPLD-XOR with feedback circuit. The internal signals (IP0, IC0, IC1, and IC2) are stored in memory cells (feedback circuit) to decrease the size of the ReRAM.

consumption of a 2-input MAGIC NOR gate based on published numbers [7] is shown in Table II. This power does not include the programming step where the programming of a memristor to HRS and LRS consumes 26.35uW and 169uW, respectively [7]. The power consumption of 10 and 100-input MAGIC NOR gates is estimated by scaling the power consumption of a 2-input MAGIC NOR gate. For example, the power consumption of a 10-input MAGIC NOR is estimated as 5× the power consumption of a 2-input MAGIC NOR, which is about 29.75uW-313.85uW. In terms of area, the numbers of computational memristors in all logic styles shown in Table II are similar, however, their CMOS parts are different. For instance, the output of a diode gate is connected

$\neg a0$	$\neg a1$	$\neg a2$	$\neg b0$	$\neg b1$	$\neg b2$	0	0	0	0	0	0	0	0	0	0
$\neg a0$	$\neg a1$	$\neg a2$	$\neg b0$	$\neg b1$	$\neg b2$	0	0	0	0	0	0	0	0	0	0
$\neg a0$	$\neg a1$	$\neg a2$	$\neg b0$	$\neg b1$	$\neg b2$	0	0	0	0	0	0	0	0	0	0
$\neg a0$	$\neg a1$	$\neg a2$	$\neg b0$	$\neg b1$	$\neg b2$	0	0	0	0	0	0	0	0	0	0
$\neg a0$	$\neg a1$	$\neg a2$	$\neg b0$	$\neg b1$	$\neg b2$	0	0	0	0	0	0	0	0	0	0
$\neg a0$	$\neg a1$	$\neg a2$	$\neg b0$	$\neg b1$	$\neg b2$	0	0	0	0	0	0	0	0	0	0

Fig. 15. 6×16 computational array for implementing a 3-bit multiplier with stateful logic gates. The matrix entities denote the initial states of the array.

to a CMOS inverter and a pull-up resistor (or a pull-down transistor). However, the output of a memristive resistance divider in a threshold gate is connected to an Opamp threshold circuit. In our simulation, it is assumed that Vdd=1V, V_{COND} =0.6V, and V_{PROG} =–0.6V where V_{COND} and V_{PROG} are bias voltages used to operate CNIMP NOR gates. In addition, inputs applied to the diode gates are defined as 0V and 1V.

Fig. 16 shows the area and delay of an N-bit adder realized with multiple approaches. The area and delay of the mPLD-XOR are explained by (6) and (7), respectively, where M_N is the number of computational memristors, and D_N is the number of clock cycles. These equations are derived for an mPLD-XOR with three modulo-two counters and feedback D flip-flops.

M_1=4, M_2=10, M_3=15, M_4=21, M_5=27, and M_N=M_{N-1}+4 for N>5. (6)

D_1=3, D_N=D_{N-1}+3 *if* (N-1 mod 3) ≠2, and D_N=D_{N-1}+2 *if* (N-1 mod 3) =2. (7)

In all approaches, only the numbers of auxiliary (computational) memristors are considered since the number of source and target memristors are equal. The mPLD-XOR is 1.31× faster in average than the FBLC approach [38], which is the second fastest approach shown in Fig. 16. In addition, the mPLD-XOR approach is 1.49× more area efficient in average than the IMPLY Parallel approach [29], which is the second most area efficient approach shown in Fig. 16.

Note that the number of memristors utilized in the FBLC approach increases dramatically [39] compared to other approaches and is not included in the graph (Fig. 16).

A 16-bit adder is also implemented in the mPLD-XOR with three modulo-two counters and feedback D flip-flops where A=01010101and B=00110011 are used as input vectors. The adder is performed in 42 clock cycles each of 4ns period and

TABLE II
COMPARISON OF CIRCUIT IMPLEMENTATION USING DIODE GATES WITH THAT OF OTHER LOGIC STYLES

Logic Style	Logic Function	Power Dissipation (uW) 2 inputs	10 inputs	100 inputs	Delay
Programmable Diode logic	NOR	0.055-0158	0.055-0.165	0.055-0.245	< 0.2ns
	NAND	0.426-0.638	0.446-0.680	0.437-0.880	<0.50ns
Threshold logic [32]	NOR	NP	10.6	11.49	0.60us
	NAND	NP	9.2	10.09	0.45us
CNIMP	NOR	0.457-1.381	0.459-5.028	0.466-64.053	2 cycles (20ns)
MAGIC [7]	NOR	5.95-62.77	NP	NP	1 cycle (1.3ns)

NP: Not provided (by the related papers)

50% duty cycle. The average power consumption is about 17uW based on our spice simulations where 97% of this power is consumed in CMOS parts of the circuit. The same adder is also implemented in Xilinx Artix-7 FPGA XC7A100T (28nm technology). The Xilinx vivado is used to estimate the data path delay and power consumption. The longest data path delay of the FPGA is estimated as 7.798ns. The total active circuit power is 416mW. In terms of energy, the mPLD-XOR circuit consumes 2.678pJ to execute the 16-bit adder, while the Xilinx FPGA consumes 3.244nJ for implementing the adder.

IX. CONCLUSION

The mPLD-XOR relies on the characteristics of self-rectifying memristors to allow voltages to be used directly as inputs. The mPLD-XOR is designed to implement multi-output functions well suited for XOR of sums or products structure such as arithmetic and communication functions. The input levels of such functions are realized with novel programmable diode gates and the output levels with CMOS modulo-two counters. The number of computational steps for calculating a multi-output function equals the maximum number of inputs to any output XOR gate when the number of modulo-two counters is large enough to calculate the outputs simultaneously. A 3-bit adder and a 3-bit multiplier are implemented with mPLD-XOR, and the size and performance of each circuit were compared with the implementation of stateful gates. The size of ReRAM and the number of clock cycles for realizing the adder in mPLD-XOR approach to the stateful approach are 280:957 and 8:29 respectively. For the 3-bit multiplier, these ratios are 564:4290 and 12:65. Adding feedback circuits to mPLD-XOR, as shown in Fig. 13, allows the implementation of a multilevel XOR logic network with any combination of sums, products, XORs, and literals at the input of any XOR gate. The mPLD-XOR with feedback circuit has the potential to decrease the number of computational steps and the size of ReRAM. In realizing the 3-bit multiplier, the size of ReRAM and the number of computational steps in mPLD-XOR with a feedback circuit to the mPLD-XOR without a feedback circuit are 564:1400 and 12:40, respectively. A 16-bit adder is also implemented in the mPLD-XOR and Xilinx Artix-7 FPGA XC7A100T. The power consumption of the mPLD-XOR is smaller than the

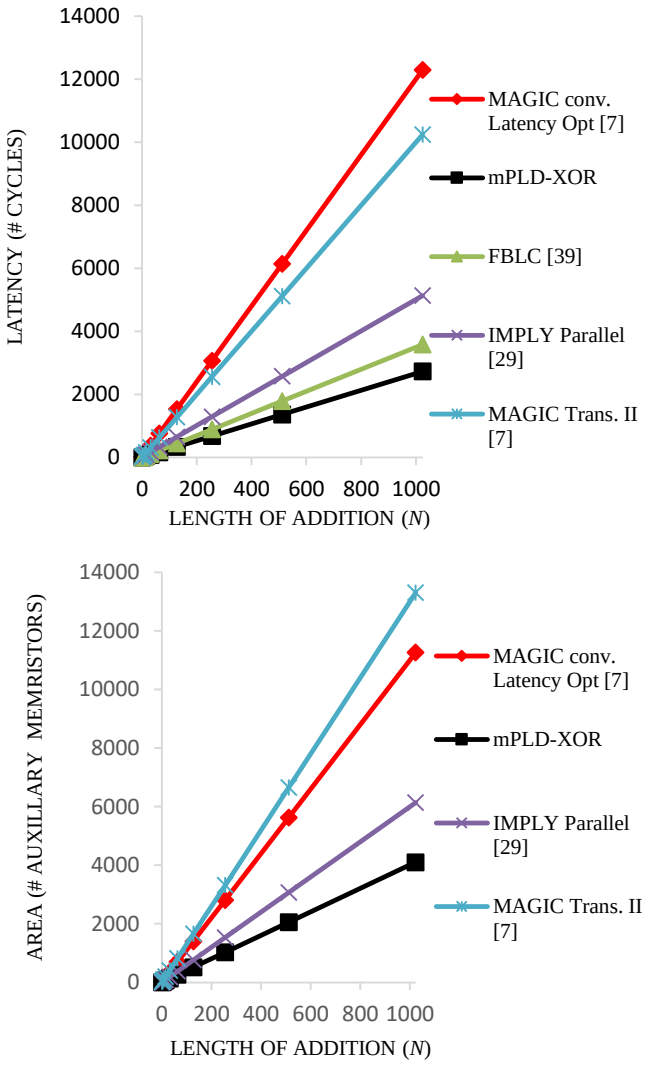


Fig. 16. Area and delay comparisons of an *N*-bit adder realized with multiple approaches.

Artix-7 FPGA, but it performs slower. In addition, the size and delay comparisons of an *N*-bit adder realized with multiple approaches show that the mPLD-XOR adder is more area-delay efficient. This paper presents examples demonstrating the benefits of using mPLD-XOR for realizing logic functions.

REFERENCES

[1] L. Chua, "Memristor-The missing circuit element," in *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507-519, Sep 1971.

[2] D. Strukov *et al.*, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80-83, 2008.

[3] J. Borghetti *et al.*, "Memristive switches enable stateful logic operations via material implication," *Nature*, vol. 464, pp. 873–876, 2010.

[4] P. Kuekes, "Material implication: Digital logic with memristors," presented at the Memristor and Memristive Syst. Symp., Berkeley, CA, 2008.

[5] S. Shin, K. Kim and S. M. Kang, "Reconfigurable Stateful nor Gate for Large-Scale Logic-Array Integrations," in *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 58, no. 7, pp. 442-446, July 2011.

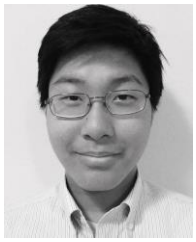
[6] E. Lehtonen, J. H. Poikonen and M. Laiho, "Applications and limitations of memristive implication logic," *2012 13th Int. Workshop on Cellular Nanoscale Netw. Appl.*, Turin, 2012, pp. 1-6.

[7] N. Talati, S. Gupta, P. Mane and S. Kvatinsky, "Logic Design Within Memristive Memories Using Memristor-Aided loGIC (MAGIC)," in *IEEE Trans. Nanotechnology*, vol. 15, no. 4, pp. 635-650, July 2016.
[8] J. F. Gimpel, "The Minimization of TANT Networks," in *IEEE Trans. Electronic Comput.*, vol. EC-16, no. 1, pp. 18-38, Feb. 1967.
[9] D. Debnath and T. Sasao, "An optimization of AND-OR-EXOR three-level networks," *Des. Automation Conf., 1997. Proc. of the ASP-DAC '97 Asia and South Pacific*, Chiba, 1997, pp. 545-550.
[10] D. Debnath and T. Sasao, "A heuristic algorithm to design AND-OR-EXOR three-level networks," *Proc. of 1998 Asia and South Pacific Des. Automation Conf.*, Yokohama, 1998, pp. 69-74.
[11] E. Lehtonen, J. Poikonen and M. Laiho, "Implication logic synthesis methods for memristors," *2012 IEEE Int. Symp. Circuits Syst.*, Seoul, 2012, pp. 2441-2444.
[12] H. Akinaga and H. Shima, "Resistive Random Access Memory (ReRAM) Based on Metal Oxides," in *Proc. of the IEEE*, vol. 98, no. 12, pp. 2237-2251, Dec. 2010.
[13] K. Kim, S. Hyun Jo, S. Gaba, and W. Lu, "Nanoscale resistive memory with intrinsic diode characteristics and long endurance," *Appl. Phys. Lett.*, vol. 96, no. 5, p. 053106, 2010.
[14] K. Kim *et al.*, "A Functional Hybrid Memristor Crossbar-Array/CMOS System for Data Storage and Neuromorphic Applications," *Nano Lett.*, vol. 12, no. 1, pp. 389-395, 2012.
[15] K. Kim *et al.*, "Low-Power, Self-Rectifying, and Forming-Free Memristor with an Asymmetric Programing Voltage for a High-Density Crossbar Application," *Nano Lett.*, vol. 16, no. 11, pp. 6724-6732, 2016.
[16] M. Aljafar, P. Long and M. Perkowski, "Memristor-Based Volistor Gates Compute Logic with Low Power Consumption," *BioNanoScience*, vol. 6, no. 3, pp. 214-234, 2016.
[17] A. Dehon, "Nanowire-based programmable architectures," *J. Emerg. Technol. Comput. Syst.*, vol. 1, no. 2, pp. 109-162, 2005.
[18] J. Borghetti *et al.*, "A hybrid nanomemristor/transistor logic circuit capable of self-programming," *Proc. of the National Academy of Sciences*, vol. 106, no. 6, pp. 1699-1703, 2009.
[19] S. Jo, K. Kim and W. Lu, "High-Density Crossbar Arrays Based on a Si Memristive System," *Nano Lett.*, vol. 9, no. 2, pp. 870-874, 2009.
[20] D. Jeong *et al.*, "Emerging memories: resistive switching mechanisms and current status," *Reports on Progress in Physics*, vol. 75, no. 7, p. 076502, 2012.
[21] E. Lehtonen *et al.*, "A cellular computing architecture for parallel memristive stateful logic," *Microelectronics Journal*, vol. 45, no. 11, pp. 1438-1449, 2014.
[22] Y. Gao *et al.*, "Read operation performance of large selectorless cross-point array with
[23] M. Lee *et al.*, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta2O5−x/TaO2−x bilayer structures," *Nature Materials*, vol. 10, no. 8, pp. 625-630, 2011.
[24] A. Torrezan, J. Strachan, G. Medeiros-Ribeiro, and R. Williams, "Sub-nanosecond switching of a tantalum oxide memristor", *Nanotechnology*, vol. 22, no. 48, p. 485203, 2011.
[25] T. Sasao, "EXMIN2: a simplification algorithm for exclusive-OR-sum-of-products expressions for multiple-valued-input two-valued-output functions," in *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 12, no. 5, pp. 621-632, May 1993.
[26] T. Sasao and P. Besslich, "On the complexity of mod-2l sum PLA's," in *IEEE Trans. Comput.*, vol. 39, no. 2, pp. 262-266, Feb 1990.
[27] S. Muroga, *Logic design and switching theory*, 1st ed. New York, NY, US: Wiley, 1979.
[28] N. Weste and K. Eshraghian, *Principles of CMOS VLSI design*, 1st ed. New York, NY, US: Addison-Wesley, 1985.
[29] S. Kvatinsky *et al.*, "Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies," in *IEEE Trans. Very Large Scale Integr. Syst*, vol. 22, no. 10, pp. 2054-2066, Oct. 2014.
[30] S. Shin, K. Kim, and S. Kang, "Memristive XOR for resistive multiplier," *Electronics Letters*, vol. 48, no. 2, p. 78, 2012.
[31] R. Rosezin *et al.*, "Crossbar Logic Using Bipolar and Complementary Resistive Switches," in *IEEE Electron Device Letters*, vol. 32, no. 6, pp. 710-712, June 2011.
[32] A. P. James, L. R. V. J. Francis, and D. S. Kumar, "Resistive Threshold Logic," in *IEEE Trans. Very Large Scale Integr. Syst*, vol. 22, no. 1, pp. 190-195, Jan. 2014.
[33] L. Gao, F. Alibart, and D. B. Strukov, "Programmable CMOS/Memristor Threshold Logic," in *IEEE Trans. Nanotechnology*, vol. 12, no. 2, pp. 115-119, March 2013.
[34] S. Kvatinsky *et al.*, "MRL — Memristor Ratioed Logic," *2012 13th Int. Workshop on Cellular Nanoscale Netw. Appl.*, Turin, 2012, pp. 1-6.
[35] J. R. Burger *et al*., "Variation-tolerant computing with memristive reservoirs," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, Jul. 2013, pp. 1–6.
[36] R. H. Wilkinson, "A Method of Generating Functions of Several Variables Using Analog Diode Logic," in *IEEE Trans. Electronic Comput.*, vol. EC-12, no. 2, pp. 112-129, April 1963.
[37] E. Lehtonen *et al.*, "Recursive Algorithms in Memristive Logic Arrays," in *IEEE Journal on Emerging and Selected Topics in Circuits Syst.*, vol. 5, no. 2, pp. 279-292, June 2015.
[38] Lei Xie *at al*., "Fast boolean logic mapped on memristor crossbar," *2015 33rd IEEE International Conference on Computer Design (ICCD)*, New York, NY, 2015, pp. 335-342.
[39] H. A. Du Nguyen *et al.*, "On the Implementation of Computation-in-Memory Parallel Adder," in *IEEE Trans. Very Large Scale Integr. Syst*, vol. 25, no. 8, pp. 2206-2219, Aug. 2017.

Muayad J. Aljafar received his M.Sc. degree in electrical engineering from Portland State University in 2016. Currently, he is a PhD student at Portland State University and his primary research area is a memristive parallel computation and machine learning.

Marek A. Perkowski (M'84-SM'04) obtained his M.S. degree in Electronics and Ph.D. Degree in automatic control from Institute of Automatic Control, Department of Electronics, Technical University of Warsaw, Warsaw, Poland. Since 1983 Dr. Perkowski works for Department of Electrical and Computer Engineering at Portland State University. Dr. Perkowski chaired the IEEE Technical Committee on Multiple-Valued Logic in years 2003-2005 and was a chair of IEEE Computational Intelligence Society Task Force on Quantum Computing 2006 - 2007.

John M. Acken He received his BS(76) and MS(78) in electrical engineering from Oklahoma State University and his PhD (88) in electrical engineering from Stanford University. He is a research faculty member in the Electrical and Computer Engineering Department, Portland State University, Portland, OR. Prior to PSU, he taught and guided research at Oklahoma State University. Dr. Acken has contributed to several technical activities including: Session Chair: CCCT 2008, Publicity Chair: IEEE IDDQ Testing Workshop 1996, Session Chair: ACM/SIGDA Workshop on Logic Level Modeling for ASICs 1995.



Robin Tan is currently an independent researcher at Portland State University. His primary research area is machine learning and applications and logic synthesis with memristive stateful logic.