Systematic Generation of Memristor-Transistor Single-Phase Combinational Logic Cells

Baishakhi Rani Biswas, *Member, IEEE*, Claire Yuan, Fangzhou Wang, and Sandeep Gupta, *Senior Member, IEEE*

***Abstract*—The objective of our research is to create efficient methods and tools for the quick and thorough assessment of emerging digital circuit devices, facilitating the adoption of promising ones. In this work, we develop methods and tools for hybrid technology that combines memristors with MOS transistors and demonstrates their effectiveness. Although several types of memristor-transistor logic have been proposed, 15 years of research has created a small set of logic cells. We propose a systematic method for generating new and efficient memristor-transistor single-phase combinational logic cells. At the core of our approach is a cell enumerator, which enables us to explore a wide range of cell designs, including nonintuitive ones, and a data-driven inductive learning method, which identifies new properties of such cells and scales up our explorations. In conjunction with other completely new tools, these create a comprehensive and definitive library of logic cells. Our new cells provide significant improvements or significantly distinct Pareto-optimal alternatives for the few logic functions for which prior research has created cells. Importantly, our methods enable us to discover a previously unknown synergistic operation between memristors and transistors that occurs for specific cell topologies. We harness this synergy to develop a method for adding memristors to low-area pass-transistor circuits such that they produce strong output voltages and low power, including for patterns that cause ratioed operation. We have also developed a new memristor-transistor logic family, namely controlled-AND (cAND)/controlled-OR (cOR), which includes many of the best cells. We have also developed a constructive method for designing such cells.**

***Index Terms*—Data-driven inductive learning (IL), enumeration, logic cells, memristors.**

I. Introduction

It is now widely believed that we are reaching the limits of physical scaling of CMOS devices [1]. This has led to the emergence of several promising devices, such as memristors [2], [3], magnetic tunneling junctions [4], ferro-electric field-effect transistors (FeFETs) [5], tunnel field-effect transistors (TFETs) [6], and others.

Wide adoption of a new device and technology requires a rigorous evaluation. To conduct a rigorous evaluation, we must identify logic and memory cells that can best utilize the strengths of the new devices. Past experience has shown that this is challenging, especially when the new devices are significantly different from the prevailing technologies. In such cases, the most efficient cells are often very different, nonintuitive, and hence typically take years to discover.

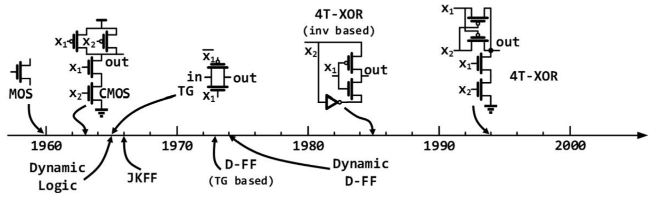


Fig. 1. Time-line of discoveries of new logic cells in MOS.

For example, consider the timeline of the discoveries of basic logic cells during the first two decades of MOS shown in Fig. 1. After the MOS device was made possible in 1960 [7], designers discovered the CMOS logic family in 1963 [8], the transmission gate (TG) in 1965 [9], and the JK flipflop in 1966 [10]. However, 13 years elapsed between the emergence of MOS devices and the discovery of TG-based D flipflops, in 1973 [11] and 1974 [12]. These flipflops were very *nonintuitive designs* at that time, since bipolar technologies prevailed and the characteristics of MOS devices were not well understood. Yet, these flip-flops are highly efficient in MOS and replaced all other types of flip-flops soon after their discovery.

Our research aims to create methods and tools that can speed up the process of discovering the most efficient cells, even those that are completely nonintuitive for new devices and technologies. This will be followed by a thorough evaluation of the new cell library. In this research work, we develop our new methods and tools by targeting a technology that combines memristors with MOS transistors.

The primary technical challenge is to search for cells, including those with nonintuitive topologies, since uniquely efficient memristor-transistor cells may be very different from those for the prevailing technology, namely MOS. Second, based on the track record of the devices widely used today (MOS) and in the past (bipolar), we expect that for any emerging device, only a tiny fraction of cell designs will be efficient. Consequently, we anticipate that the likelihood of discovering efficient cells through a random search approach will be extremely low. Third, our goal is not only to discover efficient cells but also to discover the properties that explain their operation and efficiency, and use these properties to develop constructive methods for the design of cells and circuits.

Manuscript received 3 September 2023; revised 2 February 2024; accepted 3 March 2024. Date of publication 1 April 2024; date of current version 20 September 2024. This work was supported in part by the National Science Foundation. This article was recommended by Associate Editor I. OConnor. *(Corresponding author: Baishakhi Rani Biswas.)*

The authors are with the Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: bbiswas@usc.edu).

Digital Object Identifier 10.1109/TCAD.2024.3384012

To address the above challenges, we develop an enumerative approach for the creation of cells. To scale this approach to cell sizes where desired cells can be found, we develop a data-driven inductive learning (IL) method and unique tools to make IL significantly more effective and efficient. We demonstrate that our approach dramatically increases the number of discovered cells over the past 15 years and significantly enhances or complements the previous set of cells. It is important to note that we identify key properties of memristor-transistor cells, including a previously unknown synergistic operation between memristors and transistors. This synergistic operation is at the core of our best cells' ability to provide surprisingly low-area as well as nearly ideal output voltage and low power consumption.

We first introduce the memristor-transistor logic style used for developing our methods and tools. The proposed data-driven exploration methods and tools are described in Section III. The results of our exploration, especially our library of new and efficient cells, a new logic family, its key properties, and a new constructive method for its design are presented in Section IV. Finally, Section V concludes the article.

II. Target Technology and Prior Research

Memristor [2], [13] is an emerging nonvolatile device that can be used, by itself or with MOS transistors [14], to perform logic operations. Memristor-MOS circuits can be efficient, especially in terms of the layout area, since memristors can be implemented at very low areas [14]. This two-terminal device acts like a variable resistor, with minimum and maximum resistance R_{on} and R_{off}. In a voltage-controlled memristor [15], the state of the device depends on the time integral of the voltage difference across its terminals.

A. Memristor-Transistor Single-Phase Combinational Logic

There are two major styles for using memristors to build logic circuits. The first style consists of multiphase IMPLY [16] and MAGIC [17] implementations used for in-memory computation in memristor crossbars. The input logic values are stored as memristor states (e.g., R_{off} as logic-0 and R_{on} as logic-1) in one phase, and memristor states are used to evaluate the output in the next phase. The crossbar also requires MOS decoders, READ/WRITE circuitry, and logic to control the execution of the logic operation.

In contrast, a memristor-resistor logic (MRL) [18] cell takes logic inputs as voltages, computes outputs in a *single-phase*, and produces output logic values as voltages. Also, an MRL logic cell does not require any additional MOS circuitry for decoding, reading/writing, or control. Importantly, MRL logic is compatible with static MOS logic. Hence, we target MRL-style logic for the development of our new methods and tools.

B. MRL Cells and Operation

MRL AND and OR [18] (Fig. 2) are the two basic MRL cells that can be implemented by memristors. As we focus on voltage-controlled memristors, the state of each memristor in a cell depends on the direction of the voltage across the memristor. For MRL AND as well as OR, when identical logic values are applied to both inputs (both high or both low), the resistance values of memristors remain unchanged, and the output voltage is equal to the voltage applied to the inputs. For an MRL AND with different logic values at its two inputs, the resistance of the memristor with logic-0 input decreases, i.e., becomes R_{on}, and the resistance of the memristor with logic-1 input increases, i.e., becomes R_{off}. Since $R_{\mathrm{off}} \gg R_{\mathrm{on}}$, the output is evaluated as logic 0 for AND. The OR's output is 1 for such input values.

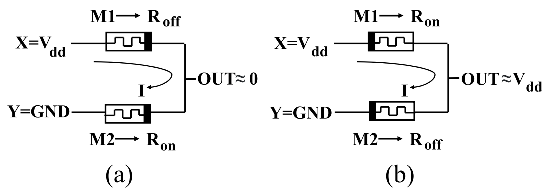


Fig. 2. Operation of (a) MRL AND, and (b) MRL OR cells.

As a memristor is a passive device, an inverter (INV) cannot be implemented using only memristors. However, by combining memristors and MOS transistors, all types of logic functions can be implemented in the MRL style. We target such memristor-MOS MRL combinational cells in this research and refer to these as *mem-MOS MRL cells.*

C. Prior Research in mem-MOS MRL

Mem-MOS MRL logic cells have been developed [19], [20], [21], [22], [23], [24] to implement AND, OR, XOR, and some other logic functions. As no systematic method had emerged prior to our research, the pace of discoveries was slow: 15 years of active research produced about two dozen cells. In Sections IV-B and IV-C, we compare these cells with the comprehensive and definitive library of cells generated by our method.

Moreover, only a handful of fundamental principles underlying the previous cells have been recognized. In Sections IV-D and IV-E, we derive new properties and constructive methods for the design of the best cells in our definitive mem-MOS MRL library.

III. Data-Driven Exploration of Logic Cells

The *objective* of this research is to identify *the definitive set of mem-MOS MRL cells,* i.e., cells that combine memristors and MOS transistors, provide single-phase operation, are compatible with static MOS logic, have low areas, and provide high performance. Further, in this study, we focus on cells that implement combinational logic functions and are acyclic.

To identify the *definitive* set of cells, it is necessary to explore all possible cell structures. This includes completely new and nonintuitive structures that go well beyond incremental modifications of MOS cells (e.g., complementary and pass-transistor logic), mem-only MRL cells, and combinations.

A. Initial Explorations and the Emergence of Our Method

We started our exploration with a naïve method: We could enumerate the *exhaustive set of*, i.e., all possible, netlists of cells with k or fewer devices. We could then simulate each cell, and select the cells that are acyclic, implement combinational logic functions, and have low area and good performance, in terms of output voltage, delay, and power.

1) Key Challenge: We knew that this naïve version would not be scalable to higher values of k. We also knew that this was in direct opposition to the following sage advice:

> "Invention consists in avoiding the constructing of useless contraptions and in constructing the useful combinations which are in infinite minority."
> -Henri Poincare

Indeed, our naïve enumerator generated extremely large numbers of cells and could only be used for 1-to-3 device cells, i.e., for $k \leq 3$ (see Table II). After simulations and a quick review of randomly selected cells, we confirmed that most of the cells created were undesirable. Despite this, our naïve enumerator provided the starting point for us to develop our new method for the discovery of new logic cells, i.e., a method for "constructing the useful combinations which are in infinite minority."

We first focus on completely new aspects of our method, namely data-driven IL and our new tools to support this IL method.

For now, assume that we have our naïve cell enumerator (exhaustive) and a circuit-level, i.e., SPICE-like, circuit simulator. The simulator is used to characterize each cell by computing the logic function (F) it implements along with its delay (D), power (P), and voltage degradation (δV) relative to ideal output voltages. It is also used to classify the cells into *undesirable (und)* and *potentially desirable (pd)*. Also, all the created cells and their simulation results are stored in a database. Finally, Our database, along with an extensible toolkit, supports all types of queries, including an extensible set of graph queries on the netlists of cells.

2) Emergence of Data-Driven Inductive Learning: Our first challenge for moving past our naïve method was to develop an approach for "avoiding the constructing of useless contraptions." IL emerged as the natural approach when we started reviewing a handful of randomly selected cells, most of which were found to be undesirable via simulations. During the review of each cell, we noted the feature that made a specific cell undesirable. For example, in a few cells, one device was disconnected from others. In one cell we reviewed, an nMOS transistor's gate was driven by GND. In another cell reviewed, the two diffusion terminals of a pMOS transistor were connected to V_{dd} and GND. In yet another cell, the cell's output was driven by a single memristor, and so on.

For each of the above example cells, we were able to show that the specific device we had noted could indeed be removed from the cell without affecting its functionality. As we removed this device, the resulting cell would have one less device and would already be in the set we generated. Therefore, we labeled this cell with k-device as undesirable. We could also subsequently update our enumerator to avoid the generation of this cell.

Our review of undesirable cells raised questions like: Is a device with this topological feature removable only in this specific cell? Or, is it removable in *every cell that we have created* for $k = 1$ to 3? Is it also removable in *every cell with a larger value of k, which we have not even created yet?* Does this topological feature imply other related features that

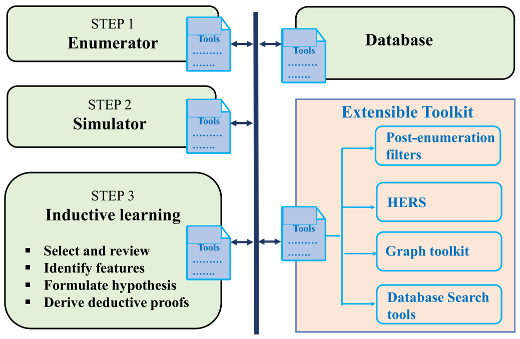


Fig. 3. Overview of our method and tools for systematically exploring logic cells. The key modules are the enumerator, simulator, and data-driven IL. Each is supported by an extensible set of tools distributed throughout (all boxes and text in blue). The enumerator is supported by tools to avoid many undesirable cells and is complemented by post-enumeration filters. Also, the simulator is complemented by tools such as low-complexity analyzers, and IL is supported by tools for cell selection, HERS, and so on. On the right side, we show our two foundational and extensible toolkits, namely database search and graph toolkit.

we have not observed in the small sample of cells we have reviewed?

A positive response to any of these questions would expand the scope of our review for each cell, since *it would enable us to not only eliminate the specific cell we reviewed but also a large number of other cells that are undesirable for the same reason.* Every one of these questions requires *inductive reasoning,* where one starts with a specific example and derives a general principle.

3) Method for Data-Driven Inductive Learning: As we started using inductive reasoning, we added steps to derive the following *IL* method, which heavily relies on data, namely our cells and simulation results.

1) *Select and review* one or more cells and identify whether they have some undesirable or desirable characteristics.
2) *Identify features* that may potentially be making these cells undesirable or desirable.
3) Via inductive reasoning, *formulate a hypothesis* that captures the potentially general cause of desirability or undesirability.
4) Develop a *deductive proof* for each hypothesis to derive a property.

Fig. 3 provides an overview of our method and tools. IL can identify *negative properties* and *positive properties*. Negative properties enable us to avoid or eliminate undesirable cells, while positive properties help us identify or create pd cells. Our IL first focused on negative properties (e.g., all the above examples) before identifying positive properties. Also, most of the early properties we identified were *topological properties*, i.e., were related to the topology of the cell's netlist graph. However, as described ahead, later in our research, we identified powerful *general properties*.

4) Early Examples of Inductive Learning: For our early observations, IL was easy and effective. For the above examples, we were able to rapidly carry out considerable generalizations. For instance, by applying IL, the specific undesirable feature of having both channel terminals of an nMOS device connected to V_{dd} and GND can be generalized to cover all three types of devices, namely nMOS, pMOS, and memristor, with both channel terminals connected to any combination of V_{dd}, GND, or inputs. (To simplify our presentation, we use the term *channel terminal* for the source and drain of MOS transistors as well as the p and n terminals of memristors. Further, we use the term *channel* for the source-to-drain path via any MOS device as well as for the p-to-n path via any memristor.) We were also able to develop hypotheses that these topological features would cause any cell of any size to be undesirable. We derived deductive proofs for these hypotheses to identify a wide range of negative topological properties summarized in Section III-C.

5) Using the Properties Identified by Inductive Learning: The early identified negative topological properties were *local*, meaning they pertained to the combinations of inputs, such as V_{dd} and *GND*, which should not be assigned to device terminals. It is easy to modify our enumerator to avoid generating cells with undesirable device configurations.

For example, our naïve enumerator first enumerates all possible sets of k devices. (e.g., for $k = 2$: 2 nMOS, 2 pMOS, 2 mem, 1 nMOS + 1 pMOS, 1 nMOS + 1 mem, and 1 pMOS + 1 mem.) Once we have all possible sets, we then proceed to create all possible ways in which these devices can be connected to form a circuit. Finally, it enumerates all possible ways in which we can assign inputs, e.g., X and Y, and special signals, namely V_{dd} and *GND*. It is easy to modify the last step to *avoid* assigning to the terminals of any device the combinations of inputs and special signal values already proven as undesirable by negative topological properties. Thus, each such property, derived from a review of a handful of undesirable cells, dramatically reduced the complexity of our enumerator and the number of cells it generates.

In addition, we describe ahead how we identify negative general properties which are more powerful than the negative topological properties. To utilize their benefits, we *filter* (i.e., remove) any cells deemed undesirable by a general property from the set of cells generated by the above enumerator (more ahead).

Hence, by incorporating the negative properties for avoidance or filtering of cells, we updated our naïve enumerator and created an *implicitly exhaustive* enumerator, i.e., an enumerator that generates all possible cells except the undesirable ones. Due to the effectiveness of the negative properties we identified via data-driven IL, we created dramatically smaller numbers of cells. Thus, it enables us to scale our study to cells with more devices, i.e., for higher values of k.

In summary, our data-driven IL method dramatically reduced the number of undesirable cells generated, allowed us to scale up our exploration, and enabled the discoveries we describe ahead. Thus, our approach and tools achieve the key goal suggested by Poncaire: "constructing the useful combinations which are in infinite minority."

B. Completely New Tools to Facilitate Data-Driven Inductive Learning

As above, a review of a small number of randomly selected cells readily provided hypotheses and these hypotheses were easy to prove. Although highly effective, this engineering approach tends to yield diminishing returns, as is often the case with most such approaches. In particular, as we increase k, the number of cells that we can potentially review increases exponentially. Also, in an undesirable cell, the numbers of potential features and hypotheses that capture the general cause of the cell being undesirable grow rapidly with k. This led us to ask: *Can we build tools to enable more comprehensive IL?*

We have developed novel tools to enhance IL while reducing manual effort. These tools include features for cell selection, identification of general causes of observed undesirable/desirable outcomes, hypothesis creation, and more. Due to space limit, we present only our most powerful, and completely new, method and tool which enables feature and *hypothesis exploration, refinement, and selection (HERS)*.

1) Background (Extensible Toolkit): We have built all our key tools around a database and an extensible library of graph functions. The database holds the cell netlists, the information created by the simulators (logic function (F), delay (D), power (P), and voltage degradation (δV)), labels and categories for each cell, relations between pairs of cells, and so on.

To enable rapid implementation of HERS for a wide range of hypotheses and post-enumeration filters for general properties, we have developed a toolkit that includes basic functions, especially for graph operations on cells. Key types of functions check whether a cell's netlist graph has a given topological property, check whether two netlists are isomorphic, and so on. Importantly, the toolkit is extensible. For example, we have already added more complex functions for modifying cells, e.g., for modifying a cell C_i by removing a specific device d_j, either by simply removing d_j or by short-circuiting its two channel terminals before removing the device.

2) HERS: Our new approach automates major inductive reasoning tasks, such as feature identification and hypothesis creation, thereby improving the effectiveness of data-driven IL. As mentioned above, the number of potential features and hypotheses can grow exponentially with cell size. In existing practice, the exploration of each hypothesis requires considerable manual analysis, hence only a handful of the most promising hypotheses can be explored. HERS differs from traditional approaches by introducing data-driven tools that explore various hypotheses, identify the most promising ones, offer guidance for refining these hypotheses, and generate insights that simplify the process of deductive proofs.

The underlying *principle* of HERS is that the correctness of a hypothesis for all cells we have generated is a necessary (but *not* sufficient) condition for the hypothesis to be correct for all cells, including large cells that we have not even generated. Through *implicitly exhaustive* enumeration, we have generated all cells that have $k \leq 5$ devices, except for the ones that we have proven to be undesirable. If our hypothesis holds for every cell in our database, then we can have a high level of confidence (but no guarantee) that the hypothesis is true for all values of k. Therefore, it is worthwhile to make the effort to develop a deductive proof.

We utilize the above principle as follows. First, we select and review some cells to identify a large number of potential features and hypotheses. We describe each hypothesis in terms of the topological features of the cell and the property that the device/cell is expected to satisfy. Second, we use our extensible graph toolkit to formulate every one of our hypotheses as a sequence of queries. Third, we use this sequence of queries on our database of cells to identify every cell that has the features specified in our hypothesis and to check whether the cell has the specified property. For each hypothesis, this sequence of queries either reports that every

Algorithm 1: Exploring Example Hypothesis Using Our New Tool HERS

```
Data: Query the database to identify all cells where only a memristor
      drives the output, say $C_i$, and note the logic function
      implemented by the cell, say as $F_i$
for each $C_i$ and $F_i$ do
    Short-circuit channel terminals of output memristor and remove
    the memristor from $C_i$ to obtain $C_i^r$;
    Query the database to identify the logic function implemented by
    $C_i^r$, say $F_i^r$;
    if $F_i^r != F_i$ then
        output $C_i$;
        return(false);
    end
end
return(true);
```

cell in the database satisfies the hypothesis, or returns one or more counterexamples of cells that do not satisfy the hypothesis.

Consider an example hypothesis: if the output of a cell is driven only by a single device which is a memristor (feature), then that memristor is redundant (hypothesis) and can be removed from the cell without changing the logic function the cell implements. HERS enables us to formulate this hypothesis into a compound query using our toolkit (see Algorithm 1): query the cell database to identify every cell C_i with the feature, i.e, the cell's output is driven only by a memristor; for every C_i that satisfies above, modify C_i by removing the memristor that drives the output to create a reduced version of the cell, C_i^r; query the database to search for cell C_i^r and return the logic function it implements; and compare the logic functions implemented by C_i and C_i^r.

For the above example hypothesis, the query reported that our hypothesis is satisfied by every cell in our database (and returned true). In contrast, for some other example hypothesis, that is not true for all cells in our database, the query would print one counterexample, i.e., a cell that does not satisfy the hypothesis, and return false. (Algorithm 1 can be easily modified to print all counter-examples.)

In the former case, we conclude that the hypothesis is strong, and we proceed to develop a deductive proof for the hypothesis. In the latter case, we review the counterexamples to determine whether to modify or abandon the hypothesis, i.e., we either undertake *hypothesis refinement* or *hypothesis abandonment*. If we decide to refine, the counterexamples provide guidance for hypothesis refinement and insights for our subsequent deductive proof. If we decide to abandon, HERS helps us by avoiding the considerable effort that would have been wasted on a search for a futile proof.

HERS has dramatically improved the effectiveness of our IL. As summarized in Table I, it has enabled the identification of our most impactful positive properties, namely the identification of generator graphs and our derivation of the sufficient topological conditions for the controlled-AND/controlled-OR (cAND/cOR) logic family which includes many of the best cells identified by this research. (More on these ahead.)

C. Results-1: Key Properties Identified via Inductive Learning

1) Negative Local Topological Properties and Their Use: In Section III-A, we presented some examples of negative topological properties we identified via straightforward IL. We have identified a much larger set of properties ranging from intuitive – e.g., many types of device redundancy, interchangeability of MOS device source-drain, interchangeability of input names, etc. – to less intuitive – e.g., redundancy of only a memristor driving the cell output, the undesirability of memristors in series, and so on. As most of these properties are local, we easily updated our enumerator to use these properties to *avoid* large numbers of cell netlists that are proven to be undesirable.

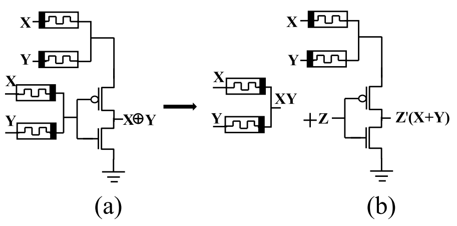


Fig. 4. (a) XOR cell proposed in [19]. (b) This cell is nonprimitive as it can be constructed using two smaller cells from our library, *and* its logic function is identical to the composition of the logic functions of the two smaller cells.

2) General Properties and Their Use: As we identified the above topological properties, we observed that sets of these shared a type of impact. For example, a set of properties caused one or more devices in a cell to be proven redundant. That is, any device in a cell that satisfied any one of these properties could be removed from the cell without affecting the logic function the cell implements. [In many (but not all) cases, the cell's performance metrics would also not change appreciably.] This observation enabled further IL to derive the *general property* that we call *reducibility*.

In this manner, IL from sets of negative topological properties we had already identified enabled us to identify several *negative general properties*: reducibility, degeneracy, isomorphism, nonprimitivity, and noncombinationality. Similarly, IL from other already identified sets of specific local properties enabled the identification of the *positive general property* of duality.

A cell is *reducible* if one or more devices in the cell can be removed without affecting the logic function implemented by the cell. When removing a device from the netlist, we can either simply remove the device, or we can remove the device *and* replace the removed device's channel with a wire.

A cell is *degenerate* if it has p inputs but implements a logic function of q variables where $q < p$. Two cells are *isomorphic*, if they have identical structures, once we consider device-level symmetries, e.g., the interchangeability of the two diffusion regions of MOS devices, or after permuting the names of cell inputs and/or the names of internal nodes in the cell's netlist.

A cell is *nonprimitive* if its netlist graph can be viewed as a composition of two or more smaller cells. This can occur when the cell has multiple disconnected components. This can also occur if: parts of the cell's netlist graph are smaller cells; *and* the logic function implemented by the given cell is a composition of the logic functions implemented by the smaller cells. For example, the XOR shown in Fig. 4 [19] is a nonprimitive cell since it is comprised of a memristor-only MRL AND and one of the cells we generated which implements $Z'(X + Y)$.

A cell is *noncombinational* if it does not implement a combinational logic function. This includes any cell whose

TABLE I
SUMMARY OF KEY PROPERTIES IDENTIFIED VIA IL. OUR NEW METHOD AND TOOL, HERS, HAS HELPED WITH THE IDENTIFICATION OF EVERY TOPOLOGICAL PROPERTY (AS INDICATED BY †)

	Topological properties	General properties
Negative	Interchangeability of MOS channel terminals and signal names;† Device redundancy†, etc. (see Section III-C)	Reducibility; isomorphism; degeneracy; non-primitivity; non-combinationality (see Section III-C)
Positive	Generator graphs† (see Figure 5); Sufficient conditions for new logic family cAND/cOR†	Duality (see Section III-C)

output is tri-state for one or more input patterns. Finally, *duality* is defined between a pair of cells, say C_i and C_j, where the structure *and* the logic function implemented by C_j can be derived by applying specific sets of transformations to those of C_i. For mem-MOS MRL, the set of circuit transformations is: reverse the p-n direction of every memristor; replace every nMOS transistor by a pMOS and vice versa; and replace every V_{dd} by GND and vice versa. The corresponding transformation of the logic function requires AND and OR to be interchanged in the logic expression. For example, if the original cell implements $X + YZ'$, then the cell obtained via duality transformation implements $X(Y + Z')$, i.e., $XY + XZ'$.

General properties are extremely useful for cell explorations in any technology, especially as *post-enumeration filters* in the early stages of the study of new technology.

Post-Enumeration Filters Based on General Properties: Each of the above negative general properties can be used to develop filters to identify undesirable cells in a given set of cells. For example, we can use our toolkit of graph functions and database to filter, i.e., to remove, all reducible cells created by the enumerator, as shown in Algorithm 2.

We have developed a set of such filters based on the above general properties and used these to eliminate extremely large fractions of cells generated by naïve enumerators. Even when we move beyond a naïve enumerator, filters continue to be valuable since it is difficult to identify topological properties that can avoid every undesirable cell which satisfies one of these general negative properties.

Also, ahead we use duality to reduce the number of cells we need to study to identify positive properties.

3) Positive Topological Properties and Their Use: We used our naïve enumerator for $k = 2$ and $k = 3$, identified negative properties, and used these to modify our enumerator to avoid the generation of large numbers of provably undesirable cells.

Generator Graphs: When we reviewed a few of the remaining cells for $k = 2$, we observed that all the cells we reviewed shared the general graph structure shown in Fig. 5(b), where the graph is obtained by replacing each device in the cell with a graph edge to denote the device's channel. Based on this observation, we formulated a hypothesis: all the pd cells for $k = 2$ share this graph structure. We used HERS, implemented using our extensible toolkit, to show that all pd cells for $k = 2$ indeed map to only one generator graph shown in Fig. 5(b). Further, for $k = 3$, all pd cells map to the two generator graphs shown in Fig. 5(c) and (d). (More ahead on deductive proofs.)

IL to Derive Larger Generator Graphs: We applied IL across the graphs for different values of k, to create all possible generator graphs for $k = 4$ to 6. For $k = 4$, we identified the four generator graphs shown in Fig. 5(e)–(h); for $k = 5$ and 6, we identified 10 and 17 generator graphs (not shown in the figure).

Algorithm 2: Post-Enumeration Filter for Reducibility

```
Data: Query the database to identify all cells, say $C_i$, and note the
      logic function implemented by the cell, say as $F_i$
for each $C_i$ and $F_i$ do
    reducible = false;
    for each $d_j$ in $C_i$ do
        Remove $d_j$ from $C_i$ to obtain $C_i^{j,o}$;
        Query the database to identify the logic function implemented
        by $C_i^{j,o}$, say $F_i^{j,o}$;
        if $F_i^{j,o} == F_i$ then
            reducible = true;
        end
        Short-circuit channel terminals of $d_j$ and remove $d_j$ from $C_i$
        to obtain $C_i^{j,s}$;
        Query the database to identify the logic function implemented
        by $C_i^{j,s}$, say $F_i^{j,s}$;
        if $F_i^{j,s} == F_i$ then
            reducible = true;
        end
    end
    if reducible == true then
        label $C_i$ as undesirable;
    end
end
```

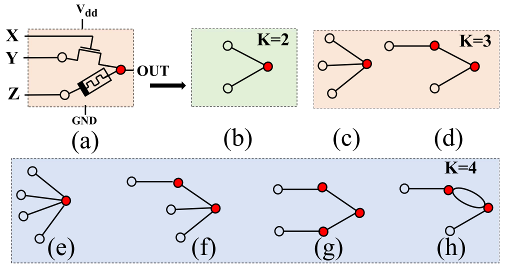


Fig. 5. Complete set of generator graphs for $k \leq 4$. The graph(s) for $k = 2$ and $k = 3$ are derived from the results of exhaustive enumeration, while those for $k = 4$ and $k = 5$ (ten graphs, not shown here), are derived via IL. (a) A sample cell, (b) generator graph for $k = 2$, (c)-(d) generator graph for $k = 3$, (e)-(h) generator graph for $k = 4$.

Collectively, these generator graphs are the most useful instance of positive topological properties to date. As we show ahead, the latest version of our enumerator avoids extremely large numbers of undesirable cells by limiting to the circuit netlists created using these graphs.

We derived deductive proofs for the genesis of these graphs. As the formal proofs are complex, we summarize the key properties that underlie our proofs.

Property: A cell where the gate of any transistor is driven by either of its diffusion regions (channel terminals), either directly or transitively via some subcircuit, is undesirable.

Property: Every transistor gate must be the cell's input.

Property: The output node of the cell must be driven by channels of at least two devices.

In summary, we only need to consider cells where: 1) the gate input of every transistor is connected to a primary input, called a *g-input* of the cell (we are considering primitive cells with independent gate inputs). 2) The memristor-transistor network has an acyclic structure (from inputs to the output) that we call a *channel-connected component*, as shown in the generator graphs. 3) The inputs shown on the left side of the graphs in Fig. 5 are called channel inputs, *c-inputs*.

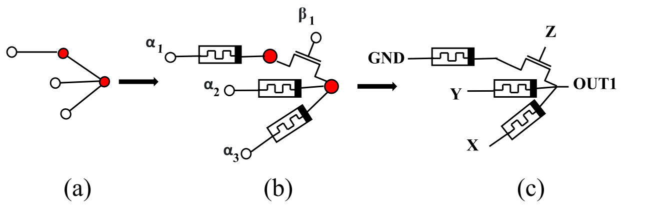


Fig. 6. Our implicitly exhaustive cell enumeration method: (a) Example generator graph, (b) one of the general netlists obtained at the end of stage-1, and (c) one of the final cell netlists obtained at the end of stage-2.

D. Our Key Methods, Tools, and General Properties

Fig. 3 shows our key methods and tools. We have already described our extensible toolkit and HERS above.

1) Cell Analyzers and Fast Simulator: Using our toolkit, we have developed tools for analysis and fast simulation for the initial characterization of our cells. These tools do not require specific transistor sizes.

Each of our analysis tools checks a cell's netlist graph for a specific negative topological property that renders the cell degenerate, nonprimitive, or noncombinational, especially tri-state. Thus, these tools eliminate large fractions of cells at extremely low complexities.

Our fast simulator generalizes the switch-level simulation algorithms for MOS [25], [26] for mem-MOS MRL cells, by adding technology-specific values, namely tri-state, weak 0/1, and ratioed, in addition to logic-0/1, and adding specialized algorithms for key tasks. At low complexity, this simulator eliminates large fractions of undesirable cells, especially most tri-state cells. It also computes the truth table entries for most input patterns, and the higher complexity simulation is required only for the small fraction of input patterns that cause ratioed voltage at the output.

2) Customized Spectre Simulator: For the characterization of cells, we use a Cadence Spectre simulator. We use CMOS 65 nm technology for transistors and the VTEAM model for memristors [15], [27] with $R_{\text{on}} = 1$ KΩ, $R_{\text{off}} = 100$ KΩ, $V_{\text{on}} = -0.05$ V, and $V_{\text{off}} = 0.05$ V. We selected the VTEAM model as it combines sufficient accuracy with low simulation time [15]. Also, we use $V_{dd} = 1.2$ V and $GND = 0$ V. Further, while our enumerator does not assign device sizes, our *default transistor sizes* are: nMOS with minimum length and width, and pMOS with minimum length and 2× the minimum width. Also, every logic cell is simulated with a capacitive load at its output, namely a CMOS INV with the above default W/L values.

In our first mode, we use this simulator to characterize the truth table of each cell. As mentioned above, we only use this simulator for input patterns for which the fast simulator has determined that the output value is ratioed. If the output voltage is between 0.4 and 0.8 V, we eliminate the cell; otherwise, depending on the voltage, we assign it logic-0/1.

In the second mode, for all the remaining cells, we use the Spectre simulator to characterize the power (P), voltage deviation from the ideal voltage (δV), and delay (D). During this process, we simulate all possible two-pattern sequences. This mode has higher complexity but is used only for the relatively small number of cells that are not eliminated near the end of our study.

Power (P)*:* From the above simulations, we compute the *average* of the power values over all possible patterns. Each pattern is applied for 10 ns and the reported average power includes both dynamic and static power within this duration.

Output Voltage Deviation (δV)*:* For each pattern, we compute the absolute value of the voltage deviation relative to the corresponding ideal voltage, namely, 0 V for logic-0 and V_{dd} for logic-1, compute the *worst-case* across all patterns.

Delay (D)*:* We compute the *worst-case* among the propagation delay values across all two-pattern sequences.

Finally, for any cell that has a ratioed output for one or more input patterns, we study whether the cell is *stable*. A ratioed cell is deemed stable if, for every pattern, the logic value at its output remains unchanged when we sweep all transistor widths over a *range* that spans 1× to 32× of the above default widths, while the lengths remain at the minimum. In most of our studies, we focus on stable cells.

3) Implicitly Exhaustive Cell Enumerators: By harnessing all the negative topological properties summarized above and the positive properties embodied in the generator graphs (Fig. 5), we have developed a highly optimized *implicitly exhaustive* enumerator for mem-MOS MRL cells. Our latest enumerator works in three stages.

Stage-1 (Create General Netlists): In the first stage, for each generator graph, we enumerate every possible combination of device type and polarity for each edge in the graph. Hence, for each edge, we enumerate a memristor and a MOS channel. In terms of polarity, for a memristor, we enumerate both directions (p-n and n-p), for a transistor we enumerate both types (nMOS and pMOS). We consider graph isomorphism during device assignment to avoid the creation of (many, but not all; more ahead) isomorphic netlists. We call each netlist generated at this stage a *general netlist.*

Each general netlist has two types of inputs: 1) g-inputs and 2) c-inputs, where each g-input drives the gate of a transistor, and each c-input drives a channel input, i.e., one of the terminals of a memristor (p or n) or one of the diffusion terminals of a MOS transistor. At this stage, for each general netlist, we assign each input a distinct input label: to the ith c-input we assign the label α_i and to the jth g-input we assign the label β_j. Fig. 6(b) shows one example general netlist created for the generator graph shown in Fig. 6(a). Finally, we use the above combination of fast and customized Spectre simulators to compute the truth table for every general netlist.

Stage-2 (Assign Primary Input Names): In this stage, for each general netlist, we enumerate versions with all possible primary input assignments to create a set of *final cell netlists.* Specifically, we enumerate all possible assignments of actual primary input names, e.g., X, Y, and so on, to the above-mentioned general input labels, namely α_i's and β_j's. During this, we consider the facts that multiple general input labels can be assigned the same primary input and that some (or all)

TABLE II
CELLS GENERATED BY OUR ENUMERATOR AND OTHER TOOLS

No. of devices (k)	No. of cells generated by naïve enumerator	No. of generator graphs	No. of general netlists with independent inputs	No. of circuit netlists	No. of cells after removing tristate, non-boolean, and degenerate cells	No. of final cells after post-enumeration filtering
2	2,216	1	10	179	20	16
3	905,467	2	56	3,828	572	173
4	-	4	487	120,759	19,392	4,522
5	-	10	4,778	5,181,368	771,355	134,449

of the c-inputs may also be assigned *special signals*, namely V_{dd} or GND, in addition to primary inputs. Fig. 6(c) shows one example primary input name assignment.

As we assign primary input names, we use the general netlist's truth table to compute the truth table for the final cell and discard degenerate and noncombinational cells, including tri-state cells.

Stage-3 (Use General Property Filters): Our above two stages avoid/eliminate all degenerate, nonprimitive, and noncombinational cells. In the third stage, we use a post-enumeration filter to avoid isomorphism and types of reducibility that occur frequently, including memristors in series or parallel, transistors in series/parallel whose g-inputs are assigned the same primary input name, and some commonly occurring types of logically equivalent cells.

E. Results-2: Effectiveness of Our Enumerator and Other Tools

Table II demonstrates the effectiveness of our above IL, HERS, enumerator, and other tools. It also shows the number of cells created by our naïve enumerator for $k \leq 3$. For $k = 3$, the naïve enumerator creates 905 467 cells. In contrast, our above enumerator starts with two generator graphs and creates 56 general netlists and 3828 circuit netlists. The number of circuit netlists that have Boolean outputs is 572. After using post-enumeration filters, it finally provides 173 mem-MOS MRL cells for our study.

Importantly, the naïve enumerator does not even work for $k > 3$ due to its exponential complexity. Yet we identify four and ten generator graphs for $k = 4$ and $k = 5$, respectively. Starting with these, our new enumerator finally provides 4522 and 134 449 mem-MOS MRL cells for these cases. The set of all cells reported in the last column of this table is the starting point of our study described next and results in discoveries that dramatically expand the last 15 years of research results.

IV. EXPERIMENTAL RESULTS: THE LIBRARY OF UNIQUELY EFFICIENT CELLS AND NEW LOGIC FAMILIES

We first identify the robust and uniquely efficient cells among the cells we have generated, to create our library of mem-MOS MRL cells. We introduce our library by first comparing the mem-MOS MRL cells proposed by prior research with the set of Pareto-optimal cells in our library, and demonstrating the benefits of our new cells. We then summarize how our library is a definitive library for circuit designers interested in mem-MOS MRL logic.

Subsequently, we use our data-driven IL method and tools to identify *positive properties* of our best cells. Importantly, we have discovered a new way in which memristors and MOS devices can work together to create low-area cells with excellent performance. We also identify one new logic family and develop a method to design high-performance cells with any number of devices.

A. Metrics and Methods for Selecting Robust and Efficient Cells

1) Performance Metrics: We focus on four metrics: 1) area (A); 2) power (P); 3) deviation from the ideal output voltage (δV); and 4) delay (D).

Memristors are considerably smaller than transistors [14], hence some previous studies have completely ignored the memristor area. However, to distinguish between two cells with the same number of transistors but different numbers of memristors, we assign a nonzero weight to the area of each memristor. We designate the *area of each memristor as 0.1, relative to 1 for the area of each transistor.* We have already described above how we compute the values of the other three metrics. In our studies, we use the values of these metrics for each cell *for the default transistor sizes presented in Section III-D.*

2) Methods for Identifying Uniquely Efficient Cells: We categorize the cells based on the logic functions they implement, as cells that implement the same logic are directly comparable.

Baseline Robustness (Cells With Low δV and P): Our exploration of mem-MOS MRL cells has identified a large number of low-cost cells with very low δV values and relatively low power, namely $\delta V < 0.1$ V and $P < 5$ μW. Low δV is correlated with better noise margin and lower short-circuit power. Hence, we mainly focus on cells that satisfy these constraints.

Baseline Robustness (Stable Cells): In most of our studies, we focus on cells that are *stable,* i.e., on cells that implement the same logic function over the wide range of transistor sizes provided in Section III-D. Such cells provide the same desirable property as a ratioless cell, namely the flexibility to size transistors to optimize other metrics, e.g., power or delay.

A Library for a Range of Users (Pareto-Optimal Cells): Subject to the above quantitative and qualitative robustness requirements (low δV-and-P and stable), we create a library that includes cells which a wide range of users would find optimal. It is important to remove only the cells that are inefficient for every possible user priority. This is captured by the notion of *Pareto-optimality:* For each logic function, we compare every pair of cells C_i and C_j that implement the function. If C_i is inferior or equal to C_j in every one of the four metrics, while being inferior in at least one metric, then C_i is marked as *dominated*. At the end of pair-wise comparisons across all cells that implement the function, the cells that are **not** marked as dominated constitute the set of Pareto-optimal cells. Each cell in this set excels in at least one metric, making it desirable for users to prioritize among A, D, P, or δV.

Selecting Cells to Study (n-Best Cells): In contrast to Pareto-optimality, here we focus on the cells that are *all-rounders*: by identifying n cells whose four metric values – A, P, δV, and D – are *all within* 100ϵ *percent* of their minimum values.

Let S denote the set of all robust cells that implement a given logic function. Also, let C_i denote a cell in the set S. Across all the cells in the set S, first, we identify the *minimum to maximum range* for each of the four metric values. That is, across all cells in S, for A we identify

$$A_{\text{range}}(S) = [A_{\min}(S), A_{\max}(S)].$$

Similarly, we identify the ranges for P, δV, and D.

Then, for a given value of a parameter ϵ, we identify all cells $C_i \in S$ that *simultaneously satisfy the following conditions for all four metrics:*

$$A(C_i) \le A_{\min}(S) + \epsilon[A_{\max}(S) - A_{\min}(S)], \quad \textbf{and}$$
$$\vdots$$
$$D(C_i) \le D_{\min}(S) + \epsilon[D_{\max}(S) - D_{\min}(S)].$$

That is, we select a cell if every one of its four metric values is within 100ϵ percent of the best value (i.e., minimum) for the corresponding metric across all the cells, where the percentage is taken over the range between the maximum and minimum value of the metric.

Finally, we compute the minimum value of ϵ such that exactly n cells in set S satisfy the above conditions. This identifies the n-best all-rounder cells for that function, and we designate the corresponding value of ϵ as ϵ_n. In particular, we study the *best cell* for every logic function and the corresponding ϵ_1. We also study the 5-best cells and ϵ_5.

B. Results-3: Our Cells and Comparison With Prior Research

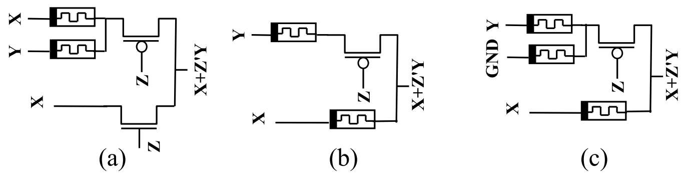
Over 15 years of prior research has created a relatively small number of mem-MOS MRL logic cells [18], [19], [20], [21], [22], [23], [24]. Here, we compare the prior cells with the cells we have created.

Previous studies have manually explored existing cells and logic families to design new cells for specific logic functions, deriving inspiration from a range of sources. The key question that motivated our research is: Are there efficient cell designs that are so far from existing cells and logic families that they may have been missed by such explorations? This has prompted us to create an exhaustive set of cells with ≤ 5 devices, except those that are certainly not desirable, to ensure that we do not miss any nonintuitive design that may be uniquely efficient.

Hence, the goal of these comparisons is to answer the following types of questions: Do there exist new cells that are *significantly superior* to the prior cells created via manual explorations? Alternatively, do there exist new cells that are Pareto-optimal and provide *sufficiently different tradeoffs* relative to the prior cells?

As mentioned earlier, for each logic function, we compare prior cells with the cells we have created, in terms of A, P, δV, and D.

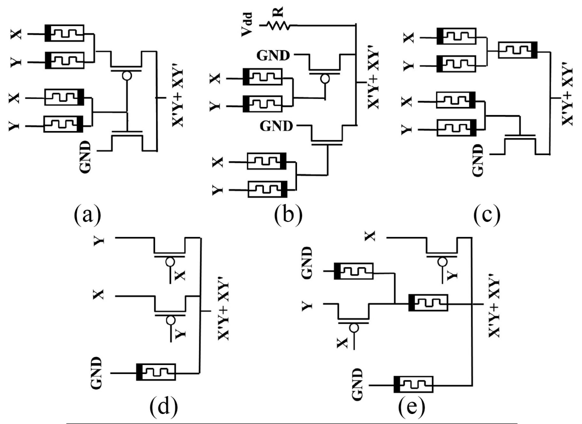
In [19], a mem-MOS MRL template has been proposed by: 1) starting with a MOS pass-transistor MUX and 2) using



Cell	A	P	δV	D
C_4	2.2	0.11μW	320mV	0.28ns
C_5	1.2	0.05μW	80mV	0.41ns
C_6	1.3	0.08μW	90mV	0.21ns

(d)

Fig. 7. Cells implementing the function $X + Z'Y$. (a) C_4: a prior cell [19], (b) C_5: a new Pareto optimal cell, (c) C_6: another new Pareto optimal cell which improves all metrics over the prior cell, and (d) corresponding metric values.



Cell	A	P	δV	D
C_7	2.4	0.08μW	250mV	0.26ns
C_9	1.5	0.06μW	80mV	0.47ns
C_{10}	2.1	0.16μW	10mV	0.61ns
C_{11}	2.3	0.27μW	90mV	0.29ns

(f)

Fig. 8. Cells implementing two-input XOR. (a) C_7: a prior cell [19], (b) C_8: a prior cell [20], (c) C_9: a prior cell [22], (d) C_{10}: a new Pareto optimal cell, (e) C_{11}: a new Pareto optimal cell, and (f) corresponding metric values.

mem-only MRL AND/OR gates to drive some/all of the three inputs of this MUX. This template can be used to create mem-MOS MRL cells for eight different logic functions of two and three input variables. Our method creates every one of the prior cells that are primitive. Importantly, our method creates additional new cells that *significantly improve* the values of some/all metrics over the corresponding prior cells.

The prior cell C_4 shown in Fig. 7 is an example of the above template that implements the function $X + Z'Y$. Two new cells C_5 and C_6 generated by our method also implement the same function. In this case, our method generates the prior cells as well. Our new cell C_6 *improves all four performance metrics* over the prior cell C_4, lower P (0.72×), lower δV (0.28×), and lower D (0.75×).

Fig. 8 shows several XOR implementations, three prior cells, C_7 [19], C_8 [20], and C_9 [22], and two new cells, C_{10} and C_{11}. Our method does not generate these prior cells, since all are nonprimitive. We ignore the prior cell C_8, since it does not work for the device parameters we use (its δV is 480 mV).

TABLE III
How Our Cells Improve Upon Logic Cells Proposed by Prior Research

Function	Prior cells	New cells	Improvements provided by new cells
$XY^{'}$	C_1, C_2	C_3	New Pareto cell, C_3: higher P (2.2×) and lower D (0.47×) compared to C_1; lower δV (0.17×) and higher D (1.6×) compared to C_2
$X + Z^{'}Y$	C_4 (Fig. 7)	C_5, C_6 (Fig. 7)	New optimal cell C_6: lower P (0.72×), δV (0.28×), D (0.75×) and A
$XY^{'} + X^{'}Y$	C_7-C_9 (Fig. 8)	C_{10}, C_{11} (Fig. 8)	New Pareto cell, C_{10}: lower δV (0.04×), but higher P (2.3×) and D (2.3×) compared to C_7; Lower δV (0.13×), but higher P (2×) and D (1.3×) compared to C_9; New Pareto cell, C_{11}: Lower δV (0.36×) and similar D, but higher P (3.38×) compared to C_7; Similar δV and lower D (0.62×), but higher P (4.5×) compared to C_9;
$XY^{'}Z$	C_{12}, C_{13}	C_{14}	New Pareto cell: lower P (0.4×) and lower D (0.4×) compared to C_{12}; Lower P (0.25×) and similar δV compared to C_{13}

TABLE IV
Our Definitive Library of mem-MOS MRL Cells

# inputs	# functions implemented	# cells generated	# cells with $\delta V < 0.1v$	# cells also with $P < 5\mu W$	# cells also stable	# cells also pareto-optimal	# functions implemented in Pareto set
2	8	42,805	17,186	14,963	14,963	565	8
3	55	69,721	12,131	9,974	9,974	853	40
4	391	38,793	3,016	2,288	2,288	584	106
5	755	8,628	304	255	255	168	90
6	341	943	0	0	0	0	0

Our new cells provide *significantly different* new Pareto alternatives for the prior cells. The new cell C_{10} *significantly improves* δV compared to prior cells but at the cost of higher P and D. Relative to C_9, C_{10} lowers δV to 0.13×, while our other new cell C_{11} lowers D to 0.61×.

Table III shows a summary of our above comparisons. It also summarizes a comparison of three cells implementing the logic function XY' (a prior cell C_1 [22], an MRL cell [18] C_2, and one of our new cells C_3) and three cells implementing the function $XY'Z$ (a prior cell C_{12} [22], an MRL implementation [18] C_{13}, and one of our new Pareto optimal cells C_{14}).

Our method has produced cells that either significantly improve all metrics over the cells reported by prior research, replacing them entirely, or adding new cells to the set of Pareto-optimal cells. These new cells significantly alter the tradeoffs by improving one or more metrics while trading off other metrics.

Hence, these comparisons clearly show that while prior research has indeed identified a few very good cells, the scope of manual explorations and design is limited. This clearly shows the benefits of our methods and tools, namely their ability to discover significantly better cells and hence enable a much more meaningful evaluation of a new technology.

Notably, our new cells provide significant improvements or significantly different tradeoffs, even though we compare our primitive cells with $k \leq 5$ devices with all the prior cells, including many nonprimitive and/or cells with $k \geq 6$ devices.

C. Results-4: Our Cell Library Versus Prior Research

Table IV summarizes the numbers of unique, robust, and pd cells. Column-3 shows the number of cells created by the latest version of our enumerator. Column-2 shows the number of logic functions for which our enumerator has created cells.

We apply our two quantitative criteria for robustness, namely $\delta V < 0.1$ V and $P < 5$ μW, to select cells, and the numbers of selected cells are shown in Columns-4 and 5. Then we apply our qualitative criterion for robustness, namely cell stability, to select cells. The resulting numbers of cells are shown in Column-6. We also use our approach to select only the Pareto-optimal cells and show the numbers of cells in our library in Column-7. Our library includes cells for the number of logic functions displayed in the last column.

Two important observations about the above steps. When we apply our two quantitative robustness criteria ($\delta V < 0.1$ V and $P < 5$ μW), we eliminate rapidly increasing fractions of cells as we move from two-input cells to six-input cells. This is because a cell with a limited number of devices that implements a logic function with a large number of inputs is likely to be not robust. Hence, our focus on robustness reduces the number of logic functions for which our library includes cells (compare Columns-2 and 8). Also, when we select only the stable cells we do not eliminate any cell that meets both our quantitative criteria for robustness, namely $\delta V < 0.1$ V and $P < 5$ μW. We will explain this ahead.

1) Definitive Library of Cells: The cells and logic functions summarized in Columns 7 and 8 of Table IV constitute *a definitive library of mem-MOS MRL cells.* The term definitive captures the fact that these cells collectively constitute a *set that guarantees Pareto-optimality relative to the set of all possible mem-MOS MRL cells which are robust*. Simply, if a cell is not in this library, then it cannot be implemented as a primitive mem-MOS MRL cell, or is not robust, or has performance metrics that are dominated by one of the other cells in our library.

In quantitative terms, 15 years of prior research has created a couple of dozen cells. Our method and tools have dramatically expanded the library of mem-MOR MRL cells to provide a

TABLE V
SUMMARY OF THE 5-BEST CELLS FOR TWO-INPUT LOGIC FUNCTIONS (ASTERISKS IDENTIFY THE TYPES OF THE BEST CELLS).

Logic function	Total # of cells with low del V and low P	ϵ_1	ϵ_5	The types of the 5-best cells (* denotes the type of the best cell)			
				cAND/cOR and derivatives	Pseudo-MOS and derivatives	MRL	Other
XY'	4637	0.08	0.08	5*	0	0	0
$X+Y$	3239	0.04	0.26	2	0	3*	0
$X+Y'$	4416	0.06	0.13	5*	0	0	0
XY	3261	0.04	0.08	0	0	5*	0
$(X+Y)'$	642	0.10	0.20	2	3*	0	0
$(XY)'$	637	0.20	0.22	0	5*	0	0
$X \oplus Y$	180	0.17	0.32	3*	0	0	2
$(X \oplus Y)'$	180	0.12	0.22	4*	0	0	1

choice of 2070 cells, counting only the cells that are robust and Pareto-optimal. Hence, most of our cells are the first known implementations of the corresponding logic functions. Also, our research has dramatically expanded the set of logic functions for which robust mem-MOS MRL cells are now known.

2) Best mem-MOS MRL Cells: We now select the best cells and apply our IL method to derive their properties and develop a constructive method for their design.

We have already presented above our method for identifying, for each logic function, the n-best cells and the corresponding ϵ_n values. Here, we use this n-best method with $n = 1$ to identify the best cell and compute the value of ϵ_1. We also use it with $n = 5$ and identify the 5-best cells and ϵ_5. Table V shows the key characteristics of the best and the 5-best cells for each logic function. The ϵ_1 value, for a logic function, shows how close the best all-rounder cell for the function comes to achieving the minimum value of *every one of the four metrics, simultaneously*.

Interestingly, for every one of the logic functions of two variables, an all-rounder cell exists whose performance is within a small range – between 1.04× and 1.20× – of the minimum values for all four metrics, *simultaneously*. The value of ϵ_1 is especially low for $X + Y$ and XY, followed by $X + Y'$ and XY'. Hence, there exists at least one all-rounder cell that is very close to being globally optimal.

Having identified the best and 5-best cells, we select these for review and apply our IL method.

3) General Structures of the 5-Best Cells: IL starting with a review of the 5-best cells enabled us to identify the general structures of these cells shown in Fig. 9. Each one of these structures constitutes a mem-MOS MRL logic style and is described next.

First, mem-only MRL cells (column labeled MRL in Table V) can only implement noninverting functions. For two-input logic functions, this limits such cells to AND and OR. For three-input functions, these are also useful for other inversion-free AND-ORs and OR-ANDs. Our study shows that, for many of these logic functions, even when compared to the set of all-possible mem-MOS MRL cells, mem-only MRL cells, shown in Fig. 9(a), are the best.

Second, due to their high areas, CMOS cells do not appear in the set of 5-best designs. However, our review of 5-best cells for NAND $(XY)'$ and NOR $(X + Y)'$, showed that

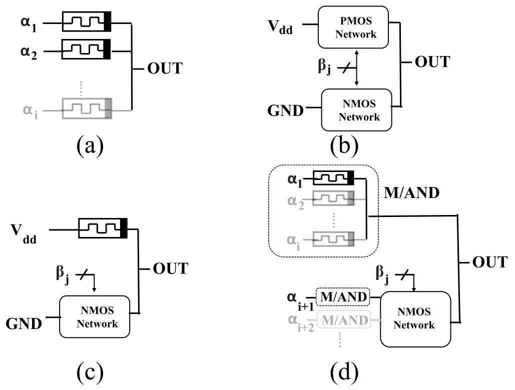
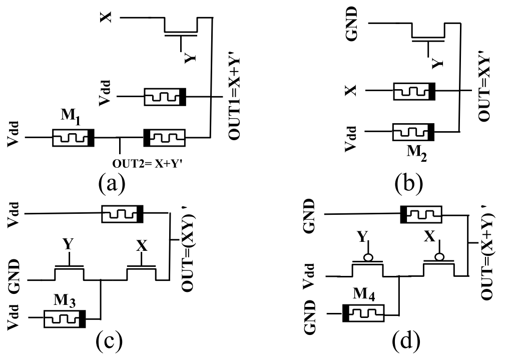


Fig. 9. Transformation from CMOS cell to controlled AND (cAND) cell: the general structure of (a) memristor AND cell, (b) CMOS cell, (c) mem-MOS MRL version of pseudo nMOS cell, where the pull-up of the CMOS cell is replaced by a load memristor, and (d) cAND cell. α_i and β_j are c- and g-inputs of the cell and will be replaced by input variables such as X and Y. c-inputs may also be replaced by GND or V_{dd}.

the use of memristors along with MOS devices enables a mem-MOS version of pseudo-MOS: the general structure for NAND is shown in Fig. 9(c); NOR has the dual structure. The specific NAND and NOR cells are shown in Fig. 10(c) and (d). The NAND can be viewed as the pull-down network of the corresponding CMOS cell, combined with a single memristor serving as the pull-up. In this cell, the memristor M_3 is logically redundant but it reduces the cell's delay and makes it one of the 5-best cells. Such mem-MOS versions of pseudo-MOS have significantly lower areas (about 0.5×) compared to the corresponding CMOS cells. Despite this, for two-input functions, such pseudo-MOS designs are the best only for NAND and NOR, i.e., the two-input functions for which a single CMOS cell implementation exists. For three-input logic functions, such cells are the best only for NAND, NOR, and OR-AND-INV functions.

Third, for all other two-input logic functions, the 5-best cells all share the general structure shown in Fig. 9(d), or the structure of its dual. Fig. 10(a) and (b) show the implementations of $X + Y'$ and XY'. (We have discussed XOR earlier.) Fig. 10(e) shown the performance metrics for the four cells above. The input–output waveforms for cell C_{15} (OUT1) are shown in Fig. 11 and illustrate what is typical of our best cells: low delays and low δV values, despite much lower areas compared to CMOS.



Cell	A	P	δV	D
C_{15}	1.3	0.11μW	12mV	0.32ns
C_{16}	1.2	0.11μW	71mV	0.18ns
C_{17}	2.2	0.24μW	18mV	0.39ns
C_{18}	2.2	0.32μW	30mV	0.44ns

(e)

Fig. 10. One of the five-best cells implementing the function. (a) C_{15}: $X+Y'$, (b) C_{16}: XY' function, (c) C_{17}: two-input NAND, (d) C_{18}: two-input NOR, and (e) corresponding metric values.

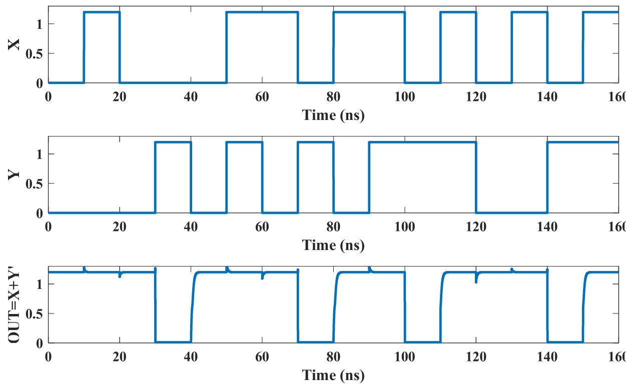


Fig. 11. Timing diagram for the input–output signal of the cell C_1 implementing the function $X + Y'$.

Hence, our review of 5-best cells enabled us to identify *new types of mem-MOS MRL cells which we call cAND and cOR.* Finally, the cells mentioned in the column labeled "Other" in Table V are cascaded combinations of two cells of the above three types (All types of cascaded structure are not logically viable [28]). Interestingly, cAND/cOR are also key components of these cells, especially for most of the three-input logic functions.

Hence, *one of the most important discoveries enabled by our research is the identification of a new cell type in mem-MOS MRL: namely the cAND, and its dual, cOR.* Detailed analysis of such cells is presented next.

D. Results-5: Synergistic Operation of Memristors and MOS

Our cAND and cOR cells universally provide low δV. This is surprising given their structures. In this section, we investigate the causes of this and discover that memristors and MOS devices operate synergistically, and this is the key to the performance of these cells.

Almost every cAND and cOR cell uses MOS transistors in configurations where paths from one or more c-inputs to the output pass via transistor channels. From MOS device

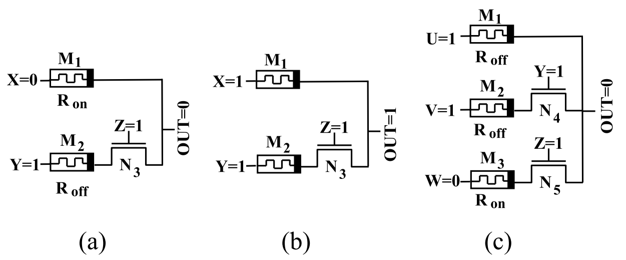


Fig. 12. (a) cAND cell with two-channel inputs with memristor states for $ZYX = 110$. (b) Same cell for $ZYX = 111$. In this case, memristor states are determined by the previous pattern. However, the states do not affect the output voltage, since both inputs are identical. (c) cAND cell with three channel inputs with memristor states for $ZYUVW = 11110$.

characteristics we know that, in such a *pass-transistor configuration*, while a logic-0 (0) passes perfectly via an nMOS transistor, a logic-1 (1) is degraded from V_{dd} to a voltage less than $V_{DD} - V_{tn}$, which is called weak-1 (w1). A pMOS in a similar configuration passes weak-0 (w0). At a cell output w1 corresponds to $\delta V > V_{tn}$, which is much higher than the δV for all cAND/cOR, as well as for most of our Pareto cells.

Hence, we faced the question: *How do all cAND cells use only nMOS transistors in the pass-transistor configuration, and still avoid weak values and achieve dramatically lower δV values?* Also, many cAND cells have paths from multiple c-inputs that pass via nMOS devices in the pass-transistor configuration and converge at the output. Hence, the question: *How do cAND cells provide low δV for patterns where the MOS devices in the mem-MOS paths have a ratioed operation?*

We used IL to answer these questions and discovered key positive properties. Consider the cell shown in Fig. 12(a) and (b). This cell has two c-inputs, X and Y, and one g-input, Z. The path from X to the output is a *mem-only path* with memristor M_1, while that from Y is a *mem-MOS path* with memristor M_2 in series with nMOS transistor N_3. These figures show this cell for input patterns where the mem-MOS path is *active,* i.e., N_3 is on, and its c-input, Y, is trying to pass a logic-1 via the nMOS device's channel. Hence, the question is: Why do we not get a weak value, w1, at the output?

First, for pattern $ZYX = 110$, nMOS N_3 is on and both input–output paths are active. Due to $X = 0$ and $Y = 1$, memristors M_1 and M_2 achieve the resistance values shown in Fig. 12(a). nMOS N_3, in series with M_2, tries to pass 1 via the mem-MOS path and would have produced a w1 at the output. However, for this pattern, mem M_1 in the mem-only path is passing 0 and is in the low resistance state (R_{on}). In contrast, mem M_2 in the mem-nMOS path is in a high resistance state (R_{off}). Hence, the mem-only path overrides the w1 from the mem-nMOS path and drives the output to 0. Hence, w1 is avoided by the *mem-only path overriding w1 to produce 0.* Very low δV is achieved at the output due to the high R_{off} to R_{on} ratio.

Next, for pattern $ZYX = 111$, nMOS N_3, in series with M_2, passes w1 via the mem-MOS path. At the same time, mem M_1 in the mem-only path supplements this by also passing 1. Hence, w1 is avoided by the *mem-only path supplementing w1 to produce 1.* Again, this achieves δV close to $0v$.

Consider another cell shown in Fig. 12(c), which shows the states of the memristors for the pattern $ZYUVW = 11110$, which causes a ratioed operation between the mem-MOS path with mem M_3 in series with nMOS N_5 trying to pass a 0; the mem-MOS path with mem M_2 in series with nMOS N_4 trying to pass a w1; and the mem-only path with M_1 trying to pass 1. Each of the memristors in the paths passing a 1 is in the high resistance state (R_{off}), while the memristor in the path passing a 0 is in low resistance state (R_{on}). Hence, the path passing 0 decisively overrides the other two paths and drives 0 at the output, thus avoiding high δV that could have been caused either due to MOS-MOS ratioed operation, or due to nMOS passing a w1, or both.

Our review of the above example cAND cells, and a few others, enabled us to *discover a synergistic operation between memristors and MOS devices in cAND cells. In these cells, if a pattern tries to pass w1 via nMOS device(s) or degrade the output voltage via MOS-MOS ratioed operation, the memristors switch states to avoid voltage degradation, either by overriding the weak value, or by supplementing the weak value, or by decisively resolving the ratioed operation.*

E. Results-6: A Constructive Method to Create cAND/cOR Cells

In this section, we present a method to construct our new family of mem-MOS MRL cells, namely cAND. cOR can be derived using duality (see Section III-C).

1) Topology of cAND: We have outlined the general structure of a cAND cell in Fig. 9(d). Now we present the requirements we impose on its topology, as well as the variations we allow. Then we present key properties satisfied by any cAND that meets the above requirements.

Requirement 1 (cAND): The block labeled nMOS Network in Fig. 9(d) is any channel-connected configuration (CCC) of nMOS devices, All the gate inputs are driven by g-inputs of the cell, labeled β_j. On the right side of this network, all the channels are combined into a single port that is connected to the cell output. The channels of the nMOS network on the left side are combined into one or more ports.

Requirement 2 (cAND): Each of the port(s) on the left side of the nMOS Network is driven either by one c-input via a memristor with the polarity shown in the figure, or by two or more c-inputs via a memristor AND.

Requirement 3 (cAND): Finally, the memristor-only network at the top either has one c-input that drives the output via one memristor with the polarity shown in the figure, or has two or more c-inputs that drive the output via a memristor AND.

Requirement 4 (cAND): Each g-input is assigned to one of the cell's primary inputs, e.g., $X, Y, \ldots$; and each c-input is assigned either to one of the primary inputs, or V_{dd}, or GND.

2) Properties Guaranteed by Above cAND Construction: The above requirements for cAND are *sufficient* to derive the following properties of cAND. These properties explain why cAND cells, and their duals, cOR cells, appear in the 5-best sets for so many logic functions.

Property 1 (cAND): For any input pattern where nMOS devices in active mem-MOS paths are all passing logic-1, any cAND cell is guaranteed to provide low δV at the cell output, as well as negligible short-circuit power dissipation.

Property 2 (cAND): When a mix of 0 and a 1 are applied to the c-inputs of active mem-MOS paths in any cAND cell, then the cell always achieves low δV and negligible short-circuit power dissipation.

Any such pattern causes an MOS-MOS ratioed operation and could have degraded the output voltage and caused high power due to high short-circuit current. However, in any cAND cell, the memristor in each active path (including the mem-only path(s), which are always active) with 0 at its c-input goes into R_{on} while the memristor in every active path with 1 at its c-input goes into R_{off}. We have already explained (end of Section IV-D) how this avoids high δV. Further, the power dissipation is reduced *since each active mem-MOS path with logic-1 at its c-input has a memristor in* R_{off} *state* and hence limits the short-circuit current and hence limits the short-circuit power.

In the past, low δV and low P were provided by CMOS cells, which have high areas as they require both nMOS pull-down and pMOS pull-up networks. While pass-transistor logic required much fewer transistors and hence had low areas, these typically had much higher δV and P.

In contrast, cAND cells only require nMOS networks and cOR cells require pMOS networks, plus a few memristors which have negligible areas. *Hence, cAND and cOR cells, for the first time, combine the low areas of pass transistor logic with provably low* δV *and* P.

V. Contributions and Ongoing Research

IL enabled us to identify that in many of our best cells, memristors and MOS devices operate synergistically to avoid high δV and P. Our key discovery is that this phenomenon arises when the characteristics of memristors and MOS devices are aligned with specific cell topologies. Importantly, this enables low δV and P, which were previously confined to CMOS cells, in a new family of cells – cAND and cOR – with half as much area. We have also identified key requirements for the construction of cAND/cOR cells.

We have developed an IL method and a completely new tool, namely hypothesis exploration, refinement, and selection (HERS), which automates critical tasks for IL. We have extensively used these to develop a highly streamlined implicitly exhaustive cell enumerator to enable the above discovery. These tools have also generated a definitive library of mem-MOS MRL cells.

Ongoing research spans device-cell co-optimization (i.e., co-optimization of the values of memristor parameters and our cell designs), further optimization of cAND/cOR cells, and new methods for logic synthesis using our definitive library of mem-MOS MRL cells. We are also expanding our method for the discovery of sequential mem-MOS MRL cells as well as uniquely efficient cells for other types of emerging devices.

References

[1] A. Razavieh, P. Zeitzoff, and E. J. Nowak, "Challenges and limitations of CMOS scaling for FinFET and beyond architectures," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 999–1004, Sep. 2019, doi: 10.1109/TNANO.2019.2942456.

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

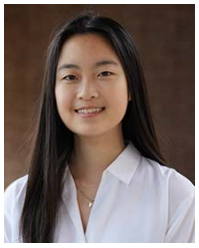
[3] H. D. Nguyen, J. Yu, L. Xie, M. Taouil, S. Hamdioui, and D. Fey, "Memristive devices for computing: Beyond CMOS and beyond von Neumann," in *Proc. IFIP/IEEE Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, 2017, pp. 1–10.

[4] S. Tehrani et al., "Recent developments in magnetic tunnel junction MRAM," *IEEE Trans. Mag.*, vol. 36, no. 5, pp. 2752–2757, Sep. 2000.
[5] S. Gupta, M. Steiner, A. Aziz, V. Narayanan, S. Datta, and S. K. Gupta, "Device-circuit analysis of ferroelectric FETs for low-power logic," *IEEE Trans. Electron Devices*, vol. 64, no. 8, pp. 3092–3100, Aug. 2017.
[6] J. Hutchby, G. Bourianoff, V. Zhirnov, and J. Brewer, "Extending the road beyond CMOS," *IEEE Circuits Devices Mag.*, vol. 18, no. 2, pp. 28–41, Mar. 2002.
[7] Electric field controlled semiconductor device, by K. Dawon. (1963, Aug. 27). U.S. Patent 31 022 30A. [Online]. Available: https://patents.google.com/patent/US3102230A/en
[8] Low stand-by power complementary field effect circuitry, by F. M. Wanlass. (1967, Dec. 5). U.S. Patent 33 568 58A. [Online]. Available: https://patents.google.com/patent/US3356858A/en
[9] Complementary field-effect transistor transmission gate, by J. J. G. G. R. Burns. (1969, Jul. 22). U.S. Patent 34 574 35A. [Online]. Available: https://patents.google.com/patent/US3457435A/en
[10] Jk flip-flop, by N. J. Miller. (1969, Sep. 16). U.S. Patent 34 678 39A. [Online]. Available: https://patents.google.com/patent/US3467839A/en
[11] Set-reset flip-flop, by G. Skorup. (1974 May). U.S. Patent 38 123 84A. [Online]. Available: https://patents.google.com/patent/US3812384A/en
[12] Logic circuit for bistable d-dynamic flip-flops, by C. Piguet. (1977 Nov. 8). U.S. Patent 40 577 41A. [Online]. Available: https://patents.google.com/patent/US4057741A/en
[13] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.
[14] J. Borghetti et al., "A hybrid nanomemristor/transistor logic circuit capable of self-programming," *Proc. Nat. Acad. Sci. United States Am.*, vol. 106, no. 6, pp. 1699–1703, Feb. 2009.
[15] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.
[16] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "'Memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, no. 7290, pp. 873–876, Apr. 2010.
[17] S. Kvatinsky et al., "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.
[18] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL—Memristor ratioed logic," in *Proc. 13th Int. Workshop Cell. Nanoscale Netw. Appl.*, 2012, pp. 1–6.
[19] M. Teimoori, A. Ahmadi, S. Alirezaee, and M. Ahmadi, "A novel hybrid CMOS-memristor logic circuit using memristor ratioed logic," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, 2016, pp. 1–4.
[20] X. Wang, R. Yang, Q. Chen, and Z. Zeng, "An improved memristor-CMOS XOR logic gate and a novel full adder," in *Proc. 9th Int. Conf. Adv. Comput. Intell. (ICACI)*, 2017, pp. 7–11.
[21] X. Xu, X. Cui, M. Luo, Q. Lin, Y. Luo, and Y. Zhou, "Design of hybrid memristor-MOS XOR and XNOR logic gates," in *Proc. Int. Conf. Electron Devices Solid-State Circuits (EDSSC)*, 2017, pp. 1–2.
[22] G. Liu, S. Shen, P. Jin, G. Wang, and Y. Liang, "Design of memristor-based combinational logic circuits," *Circuits, Syst., Signal Process.*, vol. 40, pp. 5825–5846, Jun. 2021.
[23] C. Ji, T. Li, and X. Zou, "Multifunctional module design based on hybrid CMOS-memristor logic circuit," in *Proc. Int. Conf. Service Sci. (ICSS)*, 2022, pp. 112–116.
[24] B. Su, J. Cai, Y. Zhang, Y. Wang, S. Wang, and K. Wen, "A 1T2M memristor-based logic circuit and its applications," *Microelectron. J.*, vol. 132, Feb. 2023, Art. no. 105674.
[25] Bryant, "A switch-level model and simulator for MOS digital systems," *IEEE Trans. Comput.*, vol. C-33, no. 2, pp. 160–177, Feb. 1984.
[26] A. Salz and M. Horowitz, "Irsim: An incremental MOS switch-level simulator," in *Proc. 26th ACM/IEEE Des. Autom. Conf.*, 1989, pp. 173–178. [Online]. Available: https://doi.org/10.1145/74382.74412
[27] K. A. Ali, M. Rizk, A. Baghdadi, J. Diguet, and J. Jomaah, "MRL crossbar-based full adder design," in *Proc. 26th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, 2019, pp. 674–677.
[28] B. R. Biswas and S. Gupta, "Memristor-specific failures: New verification methods and emerging test problems," in *Proc. 40th IEEE VLSI Test Symp.*, 2022, pp. 1–6.

Baishakhi Rani Biswas (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2015 and 2018, respectively. She is currently pursuing the Ph.D. degree with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA.

Her research interest includes circuit design with MOS and emerging technologies, and testing.

Ms. Biswas is a recipient of the Annenberg Fellowship.



Claire Yuan is pursuing the B.S. degree in electrical and computer engineering with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California (USC), Los Angeles, CA, USA.

She has conducted undergraduate research projects with CURVE and is currently specializing in circuit design. Outside of research, she is involved in multiple engineering clubs and volunteering.

Ms. Yuan is a recipient of Provost Fellowship in USC.

Fangzhou Wang received the B.S. degree in electrical engineering from Tianjin University, Tianjin, China, in 2012, the M.S. degree in electrical engineering, computer science, and financial engineering from the University of Southern California, Los Angeles, CA, USA, in 2014, 2015, and 2018, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, in 2019.

He is currently a Research Scientist with Meta Inc.

Sandeep Gupta (Senior Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology at Kharagpur, Kharagpur, India, in 1985, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, MA, USA, in 1989 and 1991, respectively.

He is currently a Professor with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA. His current research interests include the areas of design, testing, and validation of digital hardware.