Free Binary Decision Diagram Based Synthesis of Compact Crossbars for In-Memory Computing

Amad Ul Hassen, *Student Member, IEEE,* Dwaipayan Chakraborty, *Student Member, IEEE,*
Sumit Kumar Jha, *Member, IEEE,*

***Abstract*—We introduce a new computer-aided design approach based on Free Binary Decision Diagrams (FBDDs) for implementing Boolean functions on crossbars using flow-based computing. Our crossbar synthesis procedure uses generalized FBDDs to design crossbars for a Boolean formula such that there is a flow of current from an input nanowire to an output nanowire through the sneak paths in the crossbar if and only if the Boolean formula evaluates to true. Generalized FBDDs are more succinct representations of Boolean formulae than traditional Reduced Ordered Binary Decision Diagrams (ROBDDs) because they do not require the same variable ordering along all paths of the decision diagram. Our experimental results with the middle bit of a multiplier show that our designs are 69.9% more succinct than flow-based crossbar computing approaches designed using ROBDDs.**

***Index Terms*—FBDD, BDD, Memristor, Crossbar, Flow-based Computing, Non-volatile Memory.**

I. Introduction

JOHN von Neumann's "First Draft" defining a computer architecture for the EDVAC system [1] has survived for seven decades due to an exponential decrease in feature sizes over this period. The end of Dennard scaling and the rise of big data have led to a renewed interest in More-than-Moore devices [2] and novel computer architectures [3], including in-memory computing systems [4]. The ability to compute without moving data across the von Neumann barrier between the processor and the memory reduces both the energy and the time needed to perform the computations.

A two-dimensional crossbar of nanoscale memristors forms a desirable fabric for in-memory computing as memristors can serve as non-volatile storage devices and the values stored in the memristors can control the flow of current through sneak paths in the nanoscale crossbar. We can perform arbitrary Boolean computations on a nanoscale crossbar using the flow of current through sneak paths in the crossbar [5]–[8]. The critical step in this design process is the mapping of memristors in a crossbar to the variables in the Boolean formula being computed.

In earlier works [8], [9], it has been shown that Reduced Ordered Binary Decision Diagrams (ROBDDs) can be used to design nanoscale memristor crossbars capable of implementing Boolean formulae using flow-based computing. However, there exist Boolean formulae such that the size of their most succinct ROBDD representations with the best variable ordering is exponential in the number of variables. In this paper, we make the following new contributions:

1) We show how a bipartite variant of a Free Binary Decision Diagram (FBDD) can be used to synthesize a nanoscale crossbar that implements flow-based computing for a given Boolean formula.
2) We demonstrate the efficacy of our approach by synthesizing a nanoscale memristor crossbar for the middle bit of a 4-bit multiplier that takes 69.9% less area than a crossbar designed using ROBDDs [9].

A 4-bit multiplier designed using our approach needs 8.4% less area than the ROBDD based approach [9], while a multiplier designed using the best of both approaches needs 42.8% less area than the approach based only on ROBDDs.

A.U. Hassen and S. K. Jha are with the Electrical Engineering and Computer Science Department, University of Central Florida, Orlando, FL, 32816 USA e-mail: {amad, jha@eecs.ucf.edu}.

Manuscript received March 2, 2018; revised March 2, 2018.

II. Background

A. Memristors

A memristor is a two-terminal device that keeps track of how much current has flowed through it. Leon Chua postulated the existence of this fourth fundamental circuit element in 1971 [10]. In 2008, the first nanoscale memristor was created from doped titanium oxide by HP Labs [11]. Empirically, the resistance of a memristor may be given by the following equation: $R_{Memristor} = R_{ON}\frac{d}{L} + R_{OFF}\frac{L-d}{L}$. Here, R_{ON} is the resistance when the entire channel consists of the doped layer and R_{OFF} is the resistance when the entire channel consists of the undoped layer. L is the total length of the conductive channel which consists of doped and undoped layers, and d is the length of the doped layer. As current flows through the memristor in one direction, the doped channel length increases and its resistance drops. If the current flows in the opposite direction, its doped channel length decreases and the resistance of the device increases. The resistive state of the device remains unaltered when no current is flowing. Hence, a memristor can be used as a non-volatile memory element. We note that R_{OFF} and R_{ON} are the maximum and the minimum values of resistance for this memristor. We refer to a memristor with maximum resistance as a device in its OFF state; conversely a memristor with minimum resistance is considered as being in the ON state.

B. Crossbars

Nanoscale memristors are naturally assembled in the form of uniform two-dimensional arrays or crossbars. Memristive crossbars may be the architecture of choice for in-memory

computing as nanoscale memristors can be packed together in a crossbar with high density. An $n \times m$ crossbar consists of n horizontal nanowires and m vertical nanowires. Each horizontal nanowire is connected with all vertical nanowires through m distinct memristors. Similarly, each vertical nanowire is connected with all of the n horizontal nanowires through n different memristors. If a memristor is ON, the horizontal and the vertical nanowires connected to its terminal will be shorted; for an OFF memristor, the corresponding nanowires will not be connected.

C. Flow-based Computing using Sneak Paths

A nanoscale memristor crossbar of n rows and m columns has nm memristors. The plurality of memristive connections among horizontal and vertical nanowires gives rise to the phenomenon of sneak paths [12]. Sneak paths are trails of low resistance paths between two nanowires which are not directly connected with each other through an ON memristor. The probability of sneak-path-based disturbance increases exponentially with the length of the sneak path [13].

Our counter-intuitive approach leverages the abundance of sneak paths in nanoscale memristive crossbars for implementing Boolean functions. Our memristive crossbar design creates a one-to-one correspondence between the value of the Boolean function and the existence of a sneak path between the bottom and the topmost nanowires of the crossbar.

Definition 1 (Crossbar Designs for Boolean Formula)**.** *Let $f : \{0,1\}^k \rightarrow \{0,1\}$ be a k-bit Boolean function over variables $v_1, v_2 \ldots v_k$ and $D : R \rightarrow \{v_1, v_2, \ldots v_k\}$ be the design of the crossbar mapping memristors $R = \{r_{11}, r_{12} \ldots r_{1n}, r_{21}, \ldots, r_{mn}\}$ to the values of the variables V. A crossbar design D is said to implement the Boolean formula f if and only if the following two conditions hold:*

- *There exists a flow of current or a sneak path from the bottom nanowire to the topmost nanowire of the crossbar design D for a valuation of variables V if the Boolean formula f evaluates to true for the given valuation of the variables V.*
- *There is no sneak path connecting the bottom nanowire to the topmost nanowire of the crossbar design D for a valuation of variables V if the Boolean formula f evaluates to false for this valuation.*

The presence of a sneak path between the bottom and the topmost nanowire may be verified by applying a small voltage at the bottom nanowire and detecting the flow of current through the topmost nanowire. The flow of current in the topmost nanowire symbolizes that function is *true* while the absence of a significant flow of current implies that the function is *false*.

Figure 1(a) illustrates a simple flow-based in-memory computing design that implements a 4-input AND gate on a 3×2 crossbar. Let A, B, C and D be the four inputs to the AND gate. If A is *true*, the current flows from the bottom row to first column through the memristor labeled by A in figure 1(a). If B is also *true*, the current reaches the second row. If C and D are both *true*, the current eventually reaches the top nanowire through the sneak path shown in Fig. 1(a). Unlabeled memristors in Fig. 1 are always turned off in our design. Similarly, figure 1(b) shows a crossbar and its two sneak paths that implement a 2-input XOR gate.

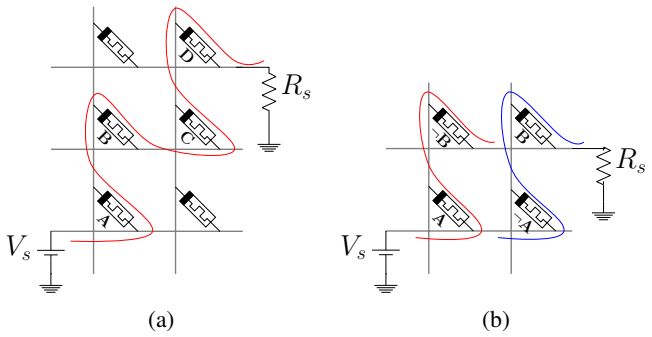

Fig. 1: (a) Crossbar design for flow-based in-memory computing of a 4-input AND gate on a 3×2 crossbar. The sneak path is highlighting the flow of current from the bottom nanowire to topmost nanowire. (b) Design for in-memory computing of a 2-input XOR gate on a 2×2 crossbar.

D. Binary Decision Diagrams

Binary decision diagrams (BDDs) are a natural choice for designing nanoscale memristive crossbars that implement flow-based computing using sneak paths. BDDs are compact structural representations of Boolean functions. Lee was the first to use them for representing switching circuits in 1959 [14]. Akers did a comprehensive study of binary decision diagrams in 1978 [15].

Let $f(x)$ be a k-bit function on the variable set $V = \{v_1, v_2, v_3 .... v_k\}$. The BDD representation for the function is a directed acyclic graph with one root node, two terminal nodes and possibly multiple intermediate nodes. All nodes except the terminal nodes have two outgoing edges. All non-terminal nodes of BDDs are labeled by a variable $v_i \in V$, terminal nodes are labeled as 0 or 1. Each non-terminal node is connected to either of its children depending on the value of the variable v_i. Each node of a BDD represents a Boolean function, the root node represents the original function $f(x)$, the terminal node 1 represents *true*, the terminal node 0 represents *false*, while non-terminal nodes represent functions which are co-factors of the function represented by their predecessor. If the original function $f(x)$ is *true* for some $x \in \{0,1\}^k$, there exists a path from the root node to the terminal node labeled as 1; if $f(x)$ is *false*, the path reaches the terminal node marked as 0.

In our earlier flow-based computing approach using sneak paths [9], we employed reduced ordered binary decision diagrams (ROBDDs) for implementing Boolean functions on nanoscale memristive crossbars. ROBDDs are a subclass of BDDs where variable ordering has to be maintained on each path from the root node to the terminal nodes. For example, if $\pi = \{v_1, v_2, \ldots v_k\}$ represents the variable ordering, v_1 should always appear before v_2 on each path from the root node to the terminal node. ROBDDs with a given variable ordering are canonical representations of Boolean

functions [16]. Efficient inductive implementations of basic Boolean operations using BDDs have been implemented in popular software packages [17]–[20].

III. FBDD based Synthesis of Crossbars

Our earlier approach that uses ROBDDs to synthesize flow-based computing circuits can lead to large memristor crossbars for functions whose ROBDDs are exponential in the number of variables. In this paper, we seek to exploit the fact that there are several interesting Boolean functions with exponential-size ROBDDs but only polynomial-size Free Binary Decision Diagrams (FBDDs) [21].

The requirement of a strict variable ordering along all paths of a ROBDD is relaxed in Free Binary Decision Diagrams (FBDDs); hence, different paths from the root to the terminal nodes of a FBDD may represent different orderings of the variables in the FBDD [21]. Like ROBDDs, FBDDs also do not allow repeated occurrences of variables along any path from the root node to the terminal nodes. In general, FBDDs are more compact than ROBDDs because FBDDs do not enforce the same strict variable ordering along all paths from the root node to the terminal node of the decision diagram.

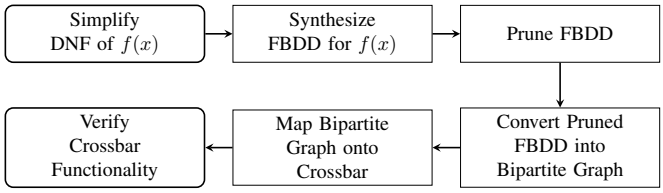

Fig. 2: Flow diagram of our FBDD-based synthesis approach.

Figure 2 shows the flow diagram illustrating the steps of our synthesis process. The first step transforms the given Boolean formula f into a simplified Disjunctive Normal Form (DNF). In the next step, we synthesize a Free Binary Decision Diagram representation of the Boolean function f. By definition of a FBDD, the functions represented by a node and its children are related by the Shannon expansion: $f(x) = af(x|_{a=1}) + \neg af(x|_{a=0})$. Here, f is the function implemented by the parent node, $f(x|_{a=1})$ and $f(x|_{a=0})$ are the functions implemented by the children nodes and a is the binary variable around which $f(x)$ is decomposed. In our approach to the synthesis of FBDDs, the variable a is obtained using a greedy heuristic. A Boolean variable a is chosen such that it appears most often in the DNF representation of the function f. The intuition behind choosing a using this greedy heuristic is that the DNF of the resulting co-factors $f(x|_{a=0})$ and $f(x|_{a=1})$ would be small for such a choice of a. Here, we compute the size of a formula as the total number of conjunctions and disjunctions in its DNF representation.

Figure 3(a) shows the free BDD synthesized for the second-output-bit of a 4-bit multiplier using this heuristic. Incidentally, the resulting graph is same as a ROBDD for this particular function. As is clear from definition 1, we are interested in only those paths that end on the terminal node 1; therefore, we prune the FBDD and get rid of the edges that are connected to the terminal node 0.

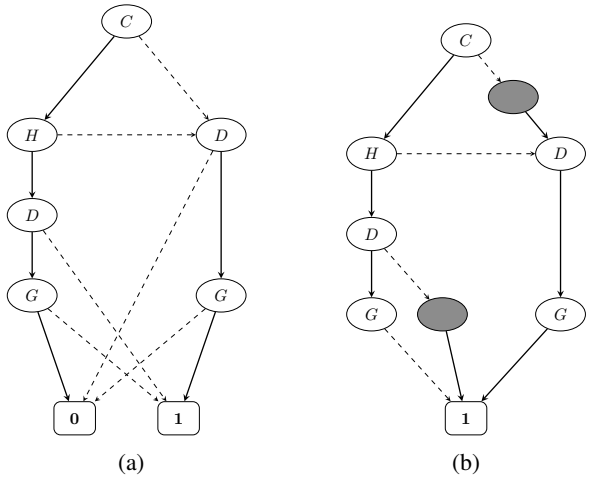

Fig. 3: (a) Free Binary Decision Diagram (FBDD) for second-output-bit of a 4-bit multiplier. A:D represent the first operand and E:H represent the second operand. (b) Bipartite graph of the pruned FBDD for the second-output-bit of a four bit multiplier synthesized using our approach. Dark nodes are dummy nodes used for converting the pruned FBDD into a bipartite graph.

However, the pruned FBDD is not yet ready for mapping onto crossbars. All memristors in crossbars establish connections between horizontal nanowires and vertical nanowires. There are no direct connections between two horizontal nanowires or two vertical nanowires in a crossbar. Hence, the underlying graph corresponding to a nanoscale memristor crossbar is bipartite. In the next step, we transform the pruned FBDD into a bipartite graph by inserting dummy nodes to eliminate odd-length cycles. It is well known that a graph without odd-length cycles is bipartite. Figure 3(b) shows a bipartite graph obtained after pruning and the introduction of dummy nodes into the FBDD of Fig. 3(a).

In the final step, we map the pruned bipartite graph obtained from the FBDD onto a nanoscale memristor crossbar. First, we measure the distance of each node from the root node. The root node is mapped onto the topmost nanowire, nodes with even numbered distance from the root node are mapped onto horizontal nanowires, and nodes with odd numbered distance from the root node are mapped onto the vertical nanowires. Since our graph is bipartite, no node can be at both even and odd distance from the root node. Figure 4 shows the synthesized crossbar for the second-output-bit of a 4-bit multiplier.

IV. Experimental Results

We have synthesized a 4-bit multiplier using our approach. It has two input operands: the first operand is comprised of bits A:D and the second operand is comprised of bits E:H. Since the output of a 4-bit multiplier is an eight bit number, we have synthesized eight crossbars. Table I presents the sizes of the synthesized crossbars and configured memristors for each output bit. In order to verify the correctness of the synthesized crossbars, we have exhaustively applied all input combinations

TABLE I: Comparison of ROBDD based approach [9] with FBDD based approach for output bits of a 4-bit multiplier

	ROBDD based Synthesis			FBDD based Synthesis			Best of Both Approaches		
Bit Index	Crossbar Size	Area	Configured Memristors	Crossbar Size	Area	Configured Memristors	Crossbar Size	Area	Configured Memristors
1 (LSB)	2 by 2	4	3	2 by 2	4	3	2 by 2	4	3
2	4 by 5	20	11	4 by 5	20	11	4 by 5	20	11
3	19 by 19	361	51	8 by 7	56	21	8 by 7	56	21
4	66 by 60	3960	186	35 by 34	1190	103	35 by 34	1190	103
5	42 by 40	1680	124	47 by 49	2303	136	42 by 40	1680	124
6	27 by 28	756	82	47 by 45	2115	129	27 by 28	756	82
7	17 by 20	340	55	28 by 28	784	80	17 by 20	340	55
8 (MSB)	7 by 9	63	22	10 by 11	110	30	7 by 9	63	22
Total		**7184**	534		**6582**	513		**4109**	421

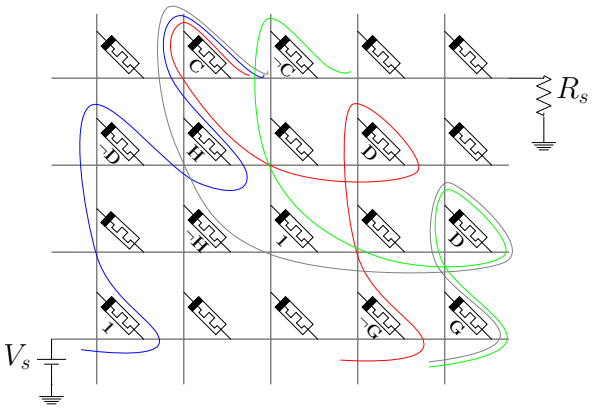

Fig. 4: Crossbar for the second-output-bit of a 4-bit multiplier with A:D and E:H as operands. Memristors are labeled with the values stored in them and unlabeled memristors are always turned off. The highlighted lines show four sneak paths emanating from the bottom nanowire and reaching the top nanowire. These sneak paths are responsible for computation of second-output-bit of the multiplier.

on the synthesized crossbar designs. We verified that the sneak paths between the bottom and topmost nanowires existed only when the corresponding output was *true*; there was no path whenever the function output was *false*.

In order to understand the impact of multiple paths of different lengths on the correctness of our designs, we perform quantitative SPICE resistive network simulations for all possible 256 input configurations of a 4-bit multiplier. We focus on the middle fourth-bit of the multiplier and used the values of $R_{ON} = 50\Omega$, $R_s = 100\Omega$, $V_s = 1V$ and $R_{OFF} = 500k\Omega$ for our simulations. Memristors with HRS (high resistance state) to LRS (low resistance state) ratio of 10^7 have been reported in literature [22]. Figure 5 summarizes our experimental observations. Flows corresponding to *true* formulae (shown in blue) are clearly distinguished from flows corresponding to the *false* formulae (shown in red). The minimum output voltage for a *true* formula was 0.177V while the maximum output voltage for a *false* formula was 0.053V; hence, the two truth values are clearly distinguishable in all cases.

We have investigated the impact of memristor variability on the correctness of our designs by changing the resistance values by 5%. We vary the resistance of each memristor in the crossbar corresponding to the middle fourth-bit of the 4-bit multiplier by increasing (decreasing) its value by 5%. The lowest voltage corresponding to *true* falls from 0.177V to 0.170V while the highest value corresponding to *false* rises from 0.053V to 0.056V; both the *true* and the *false* values are clearly distinct from each other in all cases.

We also compare the performance of the current FBDD based heuristic approach with our previous ROBDD based approach [9]. For the first four output bits, heuristic based variable ordering has produced either similar sized or smaller crossbars. But for the last four bits, ROBDD based variable ordering has has resulted in smaller crossbars. A 4-bit multiplier generated using our approach needs 8.4 percent less area than the ROBDD based approach, while a multiplier designed using the best of both approaches needs 42.8 percent less area than an approach based only on ROBDDs.

V. Conclusions and Future Work

We have presented a new FBDD-based computer-aided design approach for synthesizing compact crossbars that implement Boolean formulae using flow-based computing [5], [6], [23]. Free Binary Decision Diagrams are often more succinct than ROBDDs as they do not enforce the requirement of a strict variable ordering along all paths from the root to the terminal nodes of a decision diagram. We have taken advantage of this increased representational power of FBDDs for designing compact nanoscale memristor crossbars. In our experimental investigations, FBDDs designed using a simple greedy heuristic have resulted in identical or more compact crossbars for the first four output bits of a 4-bit multiplier.

The number of configured memristors in our crossbars designed using the best of the ROBDD and FBDD approaches varies according to the complexity of the Boolean function being synthesize. The configured memristors occupy 75%, 55%, 37.5%, 8.66%, 7.38%, 10.85%, 16.18% and 34.92% of the crossbar space for the first through eighth output bits. Thus, our decision-diagram based approach produces sparse crossbar designs. We have used model counting and simulated annealing to synthesize highly compact flow-based computing adders [7] where more than 90% of the crossbar has been configured during the design process. An interesting direction of future work would be to design compact

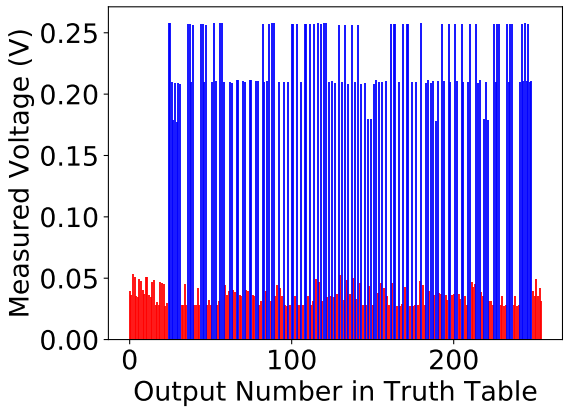

Fig. 5: Output for the fourth-output-bit of the multiplier. The X-axis represents the index of truth table entries, and the Y-axis is the voltage across R_s. Blue lines correspond to input combinations with *true* outputs, red lines represent *false* outputs for the Boolean function. Output voltage is at least 0.177 V when the Boolean formula is *true*. Voltage is no more than 0.053 V when the Boolean formula is *false*.

multipliers that configure and employ a large fraction of the available memristors on a crossbar. Our current FBDD-based approach relies on the availability of memristors with high HRS-LRS ratios. An approach that uses smaller and more dense crossbars is likely to reduce the need for memristors with high HRS-LRS ratios. A deeper theoretical investigation into the computational capability of flow-based computing on crossbars and the size of Boolean formula that can be computed on a memristor crossbar is merited.

Our FBDD based in-memory crossbar computing approach is not specific to memristor crossbars. The methodology can also be employed to design circuits using other resistive-RAM devices [24], [25]. In future, we intend to explore decision diagrams such as fixed type FBDDs [26], [27] that may result in more scalable synthesis of compact crossbars for Boolean functions with higher bit-widths.

Acknowledgement

The authors would like to thank the US Air Force for support provided through the AFOSR Young Investigator Award to Sumit Jha. The authors acknowledge support from the National Science Foundation Software & Hardware Foundations #1438989 and Exploiting Parallelism & Scalability #1422257 projects. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0255.

References

[1] J. Von Neumann, "First draft of a report on the EDVAC," *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27–75, 1993.

[2] M. M. Waldrop, "The chips are down for Moore's law," *Nature*, vol. 530, no. 7589, pp. 144–147, 2016.

[3] L. Ceze, M. D. Hill, and T. F. Wenisch, "Arch2030: A vision of computer architecture research over the next 15 years," *arXiv preprint arXiv:1612.03182*, 2016.

[4] H.-S. P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature Nanotechnology*, vol. 10, no. 3, pp. 191–194, 2015.

[5] S. K. Jha, D. E. Rodriguez, J. E. Van Nostrand, and A. Velasquez, "Computation of boolean formulas using sneak paths in crossbar computing," Apr. 19 2016, uS Patent 9,319,047.

[6] D. Chakraborty, S. Raj, J. C. Gutierrez, T. Thomas, and S. K. Jha, "In-memory execution of compute kernels using flow-based memristive crossbar computing," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, Nov 2017, pp. 1–6.

[7] D. Chakraborty and S. K. Jha, "Design of compact memristive in-memory computing systems using model counting," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2017, pp. 1–4.

[8] ——, "Automated synthesis of compact crossbars for sneak-path based in-memory computing," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. IEEE, March 2017, pp. 770–775.

[9] A. U. Hassen, "Automated synthesis of compact multiplier circuits for in-memory computing using ROBDDs," in *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, July 2017, pp. 141–146.

[10] L. Chua, "Memristor – the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, September 1971.

[11] 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.

[12] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, and K. N. Salama, "Memristor-based memory: The sneak paths problem and solutions," *Microelectronics Journal*, vol. 44, no. 2, pp. 176–183, 2013.

[13] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Information-theoretic sneak-path mitigation in memristor crossbar arrays," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 4801–4813, Sept 2016.

[14] C.-Y. Lee, "Representation of switching circuits by binary-decision programs," *The Bell System Technical Journal*, vol. 38, no. 4, pp. 985–999, July 1959.

[15] S. B. Akers, "Binary decision diagrams," *IEEE Transactions on computers*, vol. C-27, no. 6, pp. 509–516, June 1978.

[16] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, Aug 1986.

[17] R. M. Jensen, *A comparison study between the CUDD and BuDDy OBDD package applied to AI-planning problems*. School of Computer Science, Carnegie Mellon University, 2002.

[18] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," in *27th ACM/IEEE Design Automation Conference*. IEEE, Jun 1990, pp. 40–45.

[19] F. Somenzi, "CUDD: CU decision diagram package release 3.0.0," *University of Colorado at Boulder*, 2015.

[20] K. Milvang-Jensen and A. J. Hu, "BDDNOW: A parallel BDD package," in *Proceedings of the Second International Conference on Formal Methods in Computer-Aided Design*, ser. FMCAD 98. Springer, 1998, pp. 501–507.

[21] I. Wegener, *Branching programs and binary decision diagrams: theory and applications*, ser. Monographs on Discrete Mathematics and Applications. SIAM, 2000, vol. 4.

[22] A. Bricalli, E. Ambrosi, M. Laudato, M. Maestro, R. Rodriguez, and D. Ielmini, "Resistive switching device technology based on silicon oxide for improved on x2013;off ratio x2014;part ii: Select devices," *IEEE Transactions on Electron Devices*, vol. 65, no. 1, pp. 122–128, Jan 2018.

[23] D. Chakraborty, S. Raj, and S. K. Jha, "A compact 8-bit adder design using in-memory memristive computing: Towards solving the Feynman grand prize challenge," in *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, July 2017, pp. 67–72.

[24] W. Kang, Z. Wang, H. Zhang, S. Li, Y. Zhang, and W. Zhao, "Advanced low power spintronic memories beyond STT-MRAM," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17. ACM, 2017, pp. 299–304.

[25] K. Suzuki and S. Swanson, "A survey of trends in non-volatile memory technologies: 2000-2014," in *2015 IEEE International Memory Workshop (IMW)*. IEEE, May 2015, pp. 1–4.

[26] B. Becker, R. Drechsler, and R. Werchner, "On the relation between BDDs and FDDs," pp. 72–83, 1995.

[27] J. Gergov and C. Meinel, "Efficient boolean manipulation with OBDDs can be extended to FBDDs," *IEEE Transactions on Computers*, vol. 43, no. 10, pp. 1197–1209, Oct 1994.