Automated synthesis of compact crossbars for sneak-path based in-memory computing

Dwaipayan Chakraborty Sumit Kumar Jha
Computer Science Department
University of Central Florida
Orlando, Florida 32816
Email: {dchakra, jha}@eecs.ucf.edu

***Abstract*—The rise of data-intensive computational loads has exposed the processor-memory bottleneck in Von Neumann architectures and has reinforced the need for in-memory computing using devices such as memristors. Existing literature on computing Boolean formula using sneak-paths in nanoscale memristor crossbars has only focussed on short Boolean formula. There are two open questions: (i) Can one synthesize sneak-path based crossbars for computing large Boolean formula? (ii) What is the size of a memristor crossbar that can compute a given Boolean formula using sneak paths? In this paper, we make progress on both these problems. *First*, we show that the number of rows and columns required to compute a Boolean formula is at most linear in the size of the Reduced Ordered Binary Decision Diagram representing the Boolean function. *Second*, we demonstrate how Boolean Decision Diagrams can be used to synthesize nanoscale crossbars that can compute a given Boolean formula using naturally occurring sneak paths. In particular, we synthesize large logical circuits such as 128-bit adders for the first-time using sneak-path based crossbar computing.**

I. Introduction

Crossbars of nanoscale non-volatile memory devices are being actively investigated as building blocks for high-density next-generation memory systems [1], [2]. Nanoscale memristive crossbars have also been used as synthetic substitutes for neuronal synapses, and a variety of neuromorphic hardware [3], [4] have been designed that replicate neural networks on this fabric. Low power, non-volatility, and the analog dynamics of the memristive crossbar all drive their use in such brain-like bio-inspired circuits. In this paper, we focus on the use of emerging nanoscale memristor fabrics for building computers that can execute software systems originally developed for existing CMOS circuits . The ability to perform Boolean computations on nanoscale memristor crossbars is crucial to the development of a digital computation fabric capable of executing legacy software systems. Such an approach will facilitate an in-memory computing revolution without disrupting the existing software ecosystem.

Prior work has shown that sneak paths in nanoscale crossbars can be used to evaluate Boolean formula. However, earlier approaches suffer from two different kinds of problems: either they create extremely large crossbars [5], or they involve reasoning with first-order logic formula and hence create compact crossbars [6], [7] but only for tiny Boolean formula. Thus, there are two open questions with the use of sneak paths in memristor crossbars for performing logical computations:

1) What is the size of the memristor crossbar that can compute a given Boolean formula using sneak paths?
2) Can one algorithmically synthesize compact crossbars for computing large Boolean formula using sneak paths?

TABLE I: Comparison of the size of the memristor crossbar for n-bit addition produced by our method with existing sneak-path based in-memory crossbar computing approaches. The entries for negation normal form (NNF) method [8] are lower bounds. The satisfiability modulo theory (SMT) approach in [6], [9] times out and does not produce a design in one week for even moderate-size adders. Our method produces crossbars that are exponentially more succinct than those designed by the approach discussed in [8].

#bits / input	NNF-based [8]	SMT-based [6], [9]	Our method
8	$10^4 \times 10^4$	Time Out	32×17
16	$10^6 \times 10^6$	Time Out	64×33
32	$10^{11} \times 10^{11}$	Time Out	128×65
64	$10^{21} \times 10^{21}$	Time Out	256×129
128	$10^{41} \times 10^{41}$	Time Out	512×257

Our paper makes progress on both these questions through the following contributions:

1) We demonstrate that the number of rows and columns required to compute a Boolean formula is at most linear in the size of the Reduced Ordered Binary Decision Diagram (ROBDD) representing the Boolean function. Our results provide the first characterization of formula that can be computed efficiently using sneak paths in nanoscale memristor crossbars.
2) We are the first to suggest the use of ROBDD for synthesizing memristor crossbars. We design sneak-path based memristor crossbars for circuits as large as 128-bit adders. A survey of the existing literature shows that sneak-path based crossbars have only been used to build small circuits such as one-bit adders [6], [7], [9].

Section II summarizes the background on Binary Decision Diagrams and memristor crossbar circuits required to understand our approach presented in Section III. In Section IV, we discuss our experimental results on examples that are too large to be synthesized using existing methods. We conclude by summarizing our investigations and highlighting avenues for future research in Section V.

978-3-9815370-8-6/17/$31.00

II. Background and Related Work

In this section, we first recall the notions of Reduced Order Binary Decision Diagrams and sneak paths in nanoscale memristor crossbars. We then survey the existing research on sneak-path based in-memory crossbar computing and briefly describe other memristive computing architectures.

A. Boolean Decision Diagrams

A Boolean Decision Diagram (BDD) [12], [13] is a rooted directed acyclic graph that consists of two types of nodes: terminal nodes representing the logical values 0 and 1 and non-terminal decision nodes labeled by Boolean variables. Each decision node has a high edge to another node that corresponds to the variable at the decision node having the logical value 1. Each decision node also has a low edge to another node that is taken when the associated variable has the logical value 0. An Ordered Binary Decision Diagram (OBDD) is a Binary Decision Diagram where the variables appear in the same order on all paths from the root of the BDD to any of the leaves. A Reduced Ordered Binary Decision Diagram (ROBDD) is an ordered binary decision diagram where isomorphic subgraphs have been merged and nodes whose both children are isomorphic to each other have been eliminated. Several synthesis engines and verification packages have been developed using robust BDD manipulation libraries.

B. Memristors and Crossbars

The memristor is a passive, nonlinear, two-terminal electrical device that relates the fundamental quantities of electric charge and magnetic flux linkage. Besides the traditional TiO_{2-x} memristor, several other recent memristive technologies provide a number of advantages, including switching speed, energy or compatibility with CMOS. Memristors have been used for performing arithmetic operations, implementing logical operators, processing of images, nonlinear circuits and neuromorphic computing, to name a few. Resistive memory devices also lend themselves naturally to the design of memory architectures, including those for approximate computing.

C. Computing using Structurally Unconstrained Memristors

Boolean Decision Diagrams have been used to compute Boolean functions using memristors [10] by mapping each function into a netlist consisting of 2-to-1 multiplexers and NOT gates. The multiplexers are implemented using memristors while the NOT gates are implemented using CMOS. Majority Inverter Graphs (MIGs) [11] coupled with two different operations - material implication (IMP) and the majority operation (MAJ) - have also been used to compute Boolean functions. It has been argued that the MAJ-based implementation of a MIG is more space and time efficient than an IMP-based approach. These approaches differ from our proposed methodology in two ways:

1) Existing methods assume that the memristors are unconstrained by the structure of a memristor crossbar and can be arranged in any given topology. Hence, they cannot be applied directly to the synthesis of logic circuits using memristor crossbars. Some of these approaches also assume the ability to fabricate networks of memristors connected to CMOS devices. Our focus is on exploiting memristor crossbars being developed for memory applications to perform logical computations.
2) These approaches rely heavily on repeated sequential switching of memristors - often referred to as "micro-operations". In our experimental results, we show that our sneak-paths based method leads to fewer memristor switchings reducing both delay and power involved in Boolean computations.

D. Computing using Structurally Constrained Crossbars

Memristors have naturally been assembled into high-density crossbars through a variety of synthesis mechanisms, including biological self-assembly. The approaches discussed in Sec. II-C are not directly applicable to such structurally constrained crossbars. However, Velasquez & Jha [6] and Alamgir et al. [9] have suggested and experimentally demonstrated the ability to compute logical formula using sneak paths in memristor crossbars.

A "sneak path" [14] is an inadvertent path for current passing only through turned-on low resistance memristors parallel to a specified path that goes through a desired turned-off high resistance memory element. In their crossbar design formalism, Velasquez & Jha [6] and Alamgir et al. [9] have exploited sneak paths as design primitives for evaluating Boolean formulas using suitably designed memristive crossbars. A sneak path from one or more source wires to a destination wire in their memristive crossbar designs is present *if and only if* the corresponding Boolean formula evaluates to true. Thus, this approach transforms sneak paths – traditionally regarded as a curse [14] – into a boon for designing memristive computational circuits.

Computation of Boolean formulas in normal forms using sneak paths: Jha et al. [8] have shown how Boolean formulae in negation normal form (NNF) can be computed using nanoscale memristor crossbars. This design is based on a structural induction over the Boolean formula and creates large designs; for example, a one-bit adder designed using this method requires a crossbar with 16 rows and 16 columns. The size of crossbars synthesized by this method (see Table I) for n-bit adders grows at least exponentially in n and hence, no attempt has been made at using this method for large Boolean circuits.

Automated synthesis of crossbars using formal methods: Velasquez & Jha [6] employ satisfiability modulo theories (SMT) to automatically synthesize crossbars for evaluating logical formula. A full adder designed using this method has been shown to be faster as well as more power-efficient than a popular CMOS design in [9]. This approach essentially searches the state space of all possible n-by-n crossbars to determine a design that computes the desired Boolean formula. Even when this exhaustive search is performed using symbolic methods such as satisfiability solving, the approach is not able to scale to large problems and no designs larger than the 1-bit adder have been published so far.

III. Designing Sneak Path based Compact Crossbars using Boolean Decision Diagrams

The ease of fabrication often dictates that we arrange a set of memristors into a nanoscale crossbar. In this section, we present our method for mapping a Boolean formula ϕ onto a nanoscale memristor crossbar using a Reduced Ordered Boolean Decision Diagram (ROBDD). In the first step, we compute the ROBDD G_ϕ for the Boolean formula ϕ using standard approaches [13]. In the next step, we compute the crossbar design corresponding to the ROBDD.

A. Inductive Mapping of ROBDDs to Crossbars

A ROBDD for a Boolean formula ϕ is a directed acyclic graph G_ϕ where the nodes are naturally divided into levels such that all outgoing edges from nodes at level i only connect to nodes at levels $i+1$ or higher. We will map subgraph $G_\phi[i:j]$ of a ROBDD containing all nodes from level i to level j onto a crossbar $\mathbb{M}(G_\phi[i:j])$ using induction.

Base Case: The smallest entity in our design is a subgraph $G_\phi[i:i]$ with nodes at the same i^{th} level in the ROBDD.

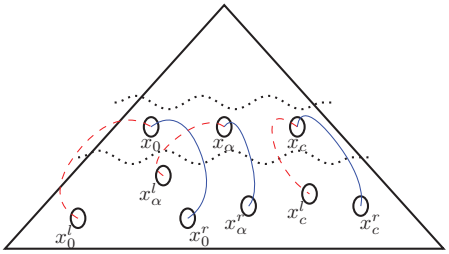

Fig. 1: Illustration of a single level of a Binary Decision Diagram and the children of the nodes at this level. The logical value of the children coupled with the logical value of the variable labeling this node produces the logical value computed by the nodes at this level.

As shown in Figure 1, let a single level $G_\phi[i:i]$ of a Boolean Decision Diagram contain c nodes x_0, x_1, …, x_c. Each node x_i has at most two children: if the variable x corresponding to the node x_i is true, the child is x_i^r; if the variable x is false, the child of the node x_i is x_i^l.

Since the nodes $x_0^l, x_0^r, \ldots x_c^l, x_c^r$ are at levels lower than the nodes x_0, x_1, …, x_c, these nodes compute functions that are independent of the variable labeling the i^{th} level or of variables in nodes at higher levels. As shown in Figure 2, we map the single level of a Binary Decision Diagram with c nodes to a crossbar with about $3c$ rows and c columns.

Our induction will proceed on the number of levels H in the fragment $G_\phi[i:j]$ of the Boolean Decision Diagram.

Inductive Hypothesis: For every subgraph $G_\phi[i:j]$ of a ROBDD containing all nodes from level i to level j such that $j \geq i$ and $j - i < H$, there exists a crossbar $\mathbb{M}(G_\phi[i:j])$ that contains horizontal nanowires $h_{v_0^{i-1}}, h_{v_1^{i-1}}, \ldots h_{v_{c-1}^{i-1}}$ computing the value of the Boolean formula corresponding to the nodes $v_0^i, v_1^i, \ldots v_{c-1}^i$ given that the values corresponding to the nodes $v_0^j, v_1^j, \ldots v_{c'-1}^j$ are available on the corresponding input nanowires. Figure 3 illustrates a crossbar that computes $G_\phi[i:j]$ along with its input and output nanowires.

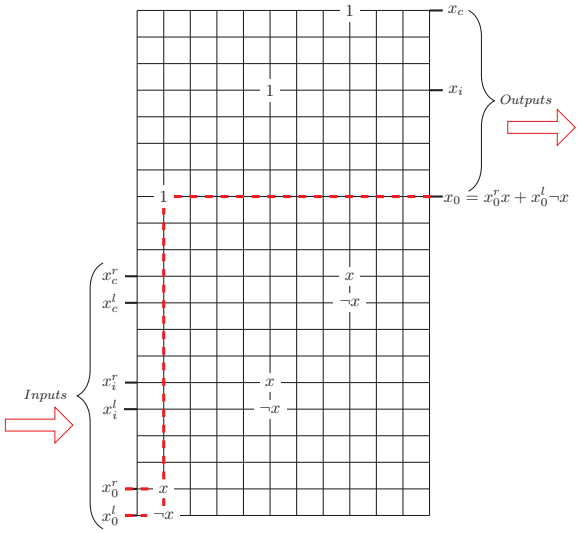

Fig. 2: Mapping a single level of nodes in a Binary Decision Diagram to a crossbar for sneak-path based computation. The inputs arrive on the left horizontal nanowires and the outputs are available on the top horizontal nanowires. An an illustration, the flow of current for one output nanowire computing x_0 from nanowires x_0^l and x_0^r using memristor values x and $\neg x$ is highlighted using dotted red lines.

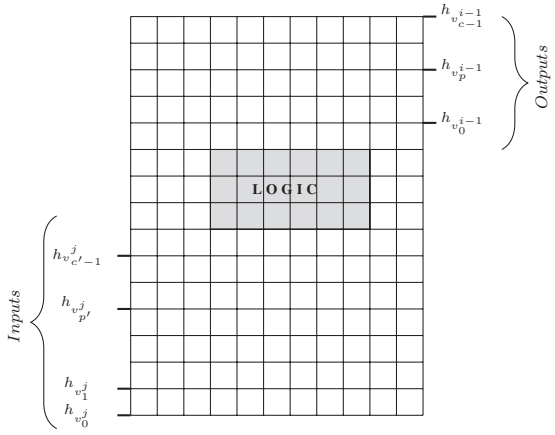

Fig. 3: The inductive hypothesis assumes that the subgraph $G_\phi[i:j]$ can be mapped to a crossbar with the inputs (corresponding to level j of the ROBDD) being fed along the lower rows and the outputs (corresponding to level i) leaving the crossbar along the higher rows. The grey section labelled LOGIC indicates that the detailed design of the crossbar implementing $G_\phi[i:j]$ is not illustrated here.

Induction: Consider a subgraph $G_\phi[i:j]$ containing all nodes from level i to level j ($j > i$ and $j - i = H$) of a ROBDD G_ϕ for the Boolean formula ϕ. We choose a level k between i and j, and divide the subgraph $G_\phi[i:j]$ into subgraphs $G_\phi[i:k]$ and $G_\phi[k+1:j]$ with directed edges $(u_0^k, v_0^k), (u_1^k, v_1^k), \ldots (u_{c-1}^k, v_{c-1}^k)$ from the subgraph $G_\phi[i:k]$ to $G_\phi[k+1:j]$. Similarly, $(u_0^{i-1}, v_0^{i-1}), (u_1^{i-1}, v_1^{i-1}), \ldots (u_{d-1}^{i-1}, v_{d-1}^{i-1})$ are directed edges coming into $G_\phi[i:k]$ and $(u_0^j, v_0^j), (u_1^j, v_1^j), \ldots (u_{e-1}^j, v_{e-1}^j)$ are directed edges coming out of $G_\phi[k+1:j]$

By the inductive hypothesis, for the subgraph $G_\phi[k+1:j]$, there exists a crossbar $\mathbb{M}(G_\phi[k+1:j])$ that

contains horizontal nanowires $h_{v_0^k}, h_{v_1^k}, \ldots h_{v_{c-1}^k}$ computing the value of the Boolean formula corresponding to the nodes $v_0^k, v_1^k, \ldots v_{c-1}^k$ given that the values corresponding to the nodes $v_0^j, v_1^j, \ldots v_{e-1}^j$ are available on the corresponding input horizontal nanowires. Similarly, by the inductive hypothesis, there exists a crossbar $\mathbb{M}(G_\phi[i : k])$ that contains horizontal nanowires $h_{v_0^{i-1}}, h_{v_1^{i-1}}, \ldots h_{v_{d-1}^{i-1}}$ computing the value of the Boolean formula corresponding to the nodes $v_0^{i-1}, v_1^{i-1}, \ldots v_{d-1}^{i-1}$ given that the values corresponding to the nodes $v_0^k, v_1^k, \ldots v_{c-1}^k$ are available on the corresponding input horizontal nanowires.

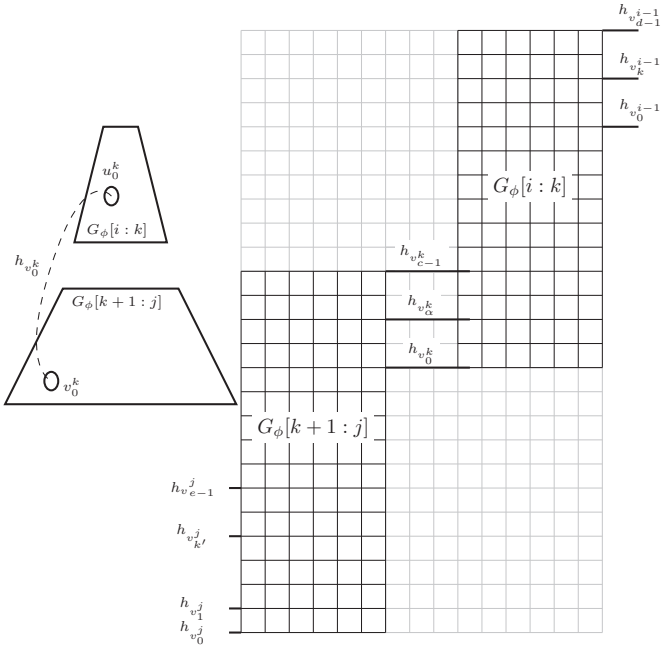

Fig. 4: The left image shows the subgraphs $G_\phi[i : k]$ and $G_\phi[k + 1 : j]$ of the BDD with one of the edges (u_0^k, v_0^k) shown for illustrative purposes. The right image shows the inductive synthesis of the crossbar for $G_\phi[i : j]$ from the crossbars corresponding to subgraphs $G_\phi[k+1 : j]$ and $G_\phi[i : k]$. In particular, observe how the nanowire $h_{v_0^k}$ carries the value corresponding to node v_0^k in the crossbar for $G_\phi[k + 1 : j]$ to the crossbar for $G_\phi[i : k]$.

As shown in Fig. 4, the crossbar corresponding to the subgraph $G_\phi[i : j]$ can be obtained by putting together smaller crossbars corresponding to the subgraphs $G_\phi[k + 1 : j]$ and $G_\phi[i : k]$ of the Boolean Decision Diagram. The lower crossbar may produce the outputs in a different order than the inputs expected by the upper crossbar but permutations of rows (or columns) do not change the computation being performed by a crossbar; hence, the rows of one crossbar can be rearranged to meet the desired rows of the other crossbar.

Theorem 1. *The number of rows $R(N)$ and the number of columns $C(N)$ of a memristive crossbar $M(G_\phi)$ that computes the Boolean formula ϕ corresponding to a ROBDD G_ϕ is linear in the number of nodes N of the ROBDD G_ϕ.*

Proof. We will show that $R(i) \leq 3i$ for all i using mathematical induction where i is the number of nodes between two levels of a ROBDD:

Base Case: An upper bound on the number of rows $R(i)$ for BDDs with one node can be computed as follows: $R(|G_{True}|) \leq 3$, $R(|G_{False}|) \leq 3$, $R(|G_x|) \leq 3$, $R(|G_{\neg x}|) \leq 3$. Hence, $R(1) \leq 3$.

Inductive Hypothesis: Assume that $R(i) \leq 3i$ for all $i < N$, where i is the number of nodes between two levels of a ROBDD.

Inductive Step: Consider the number of rows $R(N)$ corresponding to the number of nodes between two levels of a ROBDD, say $G_\phi[i : j]$.

$$
\begin{aligned}
R(N) &= R(|G_\phi[i:j]|) \\
&\leq R(|G_\phi[i:k]|) + R(|G_\phi[k+1:j]|) \\
&\qquad \text{(by the design of the crossbar)} \\
&\leq 3|G_\phi[i:k]| + 3|G_\phi[k+1:j]| \\
&\qquad \text{(by inductive hypothesis)} \\
&= 3(|G_\phi[i:k]| + |G_\phi[k+1:j]|) \\
&= 3N \\
&\quad \text{(since, } |G_\phi[i:k]| + |G_\phi[k+1:j]| = |G_\phi[i:j]| = N\text{)}
\end{aligned}
$$

By symmetry, the number of columns $C(N)$ is also linear in the number of nodes of the ROBDD. We omit the identical argument for brevity. □

IV. Experimental Results

We pursued an experimental evaluation of our approach in order to answer three different queries:

1) Can we synthesize n-bit adders for $n = 1$ to $n = 128$ bits? Section IV-A summarizes our successful attempts at designing n-bit adders using sneak paths in memristor crossbars. To the best of our knowledge, we are the first to report the synthesis of circuits as large as 128-bit adders using sneak paths in nanoscale crossbars.
2) How does our approach compare to existing methods for sneak-path based crossbar computing? Table I shows that our approach produces designs that are exponentially more compact than existing techniques.
3) How does our approach compare against existing techniques for memristive computing that are not constrained by the structure of a nanoscale crossbar? Section IV-B shows that our method performs better than most of these approaches in both computing speed and power. The efficiency is achieved by avoiding the idea of "micro-operations" that plague other techniques based on the imply-logic and its successors.

For our experiments, we used the HSPICE model presented in [15]. We assume an input voltage of 1 V to our crossbar circuits and an external load resistance of 100Ω. The switching energies and the switching delay of memristors have been obtained from [16]. The switching delay is 85 picoseconds and the average power consumption for switching is $30\mu W$. We note that the nature of our results is not a function of these specific values as our approach reduces the number

of memristor switchings by avoiding the idea of performing multiple "micro-operations" that cause additional delay and energy usage in competing approaches.

A. Design of n-bit Adders

We expect our method to synthesize compact, fast and energy-efficient adders by leveraging an optimal ordering of the BDD variables. In Table I, we have studied the size of the nanoscale crossbar required to perform $n-$bit addition using sneak paths in nanoscale crossbars. Our results compare favorably with the work of Alamgir et al. [9] as that approach is not able to synthesize adders with four or more bits. The size of our crossbar is *exponentially more succinct* than those generated by Jha et al. [8].

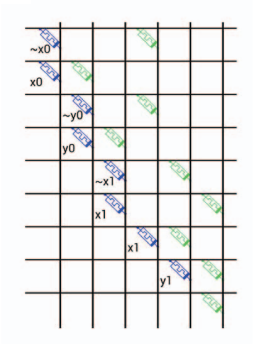

Fig. 5: Computation of the carry-bit for a 2-bit adder. Input data is stored in the blue memristors while the green memristors are turned-on. Memristors not shown in the figure are turned-off.

TABLE II: Total computation delay (in picoseconds) to compute the most-significant bit of n-bit addition.

#bits / input	BDD [10]	MIG -IMP [11]	MIG -MAJ [11]	Our Method	Speedup Prior best/ Our method
16	16,320	27,200	8,160	5,369	151.9%
32	32,640	54,400	16,320	10,823	150.7%
64	65,280	108,800	32,640	21,729	150.2%
128	130,560	217,600	65,280	43,543	149.9%

In Table II and Table III, we compare the speed and power of our crossbar-based approach with three different methods where the memristors are not constrained by the structure of a crossbar. The speed and power for competing approaches are approximate bounds computed by us using the number of required micro-operations. The switching delay and energy have been assumed to be identical for all methods. We see that prior BDD-based approach [10] requires about 300% of the energy and time required by our approach. However, the approach based on MIG-MAJ [11] performs better than prior approaches and may be particularly effective for large circuits.

TABLE III: Total power (in μW) required to compute the most-significant bit of a binary addition.

#bits / input	BDD [10]	MIG -IMP [11]	MIG -MAJ [11]	Our Method	Power Ratio Prior best/ Our method
16	5,760	9,600	2,880	1,901	151.5%
32	11,520	19,200	5,760	3,832	150.3%
64	23,040	38,400	11,520	7,693	149.7%
128	46,080	76,800	23,040	15,415	149.5%

B. Other Benchmarks from Existing Literature

We investigate the performance of our synthesis algorithm on benchmarks reported in [17]. Our performance varies substantially based on the benchmark being investigated as shown in Table IV and Table V. Some Boolean formula have large representations as Boolean decision diagrams and our approach is challenged by such examples. The speed and power estimates for competing approaches were obtained by using the number of micro-operations reported in the respective publications. The memristor switching delay and energy have been assumed to be identical for all approaches.

TABLE IV: Total computation delay (in picoseconds) for benchmarks reported in [17].

#bits / input	BDD	MIG -IMP	MIG -MAJ	Our Method	Prior best/ Our method
5xp1	6,205	8,415	3,060	1,109	275.9%
9sym_d	5,270	14,875	5,100	2,898	200.1%
misex3	15,725	14,025	5,695	5,369	106.1%
sym10_d	5,950	15,895	6,120	4,192	145.9%
clip	7,585	9,350	3,400	1,706	199.3%

It should be noted that the competing approaches require the ability to build networks of memristor circuits in arbitrary topologies while our approach respects the structural constraints posed by a nanoscale memristor crossbar.

TABLE V: Total power consumption (in μW) for benchmarks in [17].

#bits / input	BDD	MIG -IMP	MIG -MAJ	Our Method	Prior best/ Our method
5xp1	2,190	2,970	1,080	393	274.8%
9sym_d	1,860	5,250	1,800	1,027	175.3%
misex3	5,550	4,950	2,010	1,901	105.7%
sym10_d	2,100	5,610	2,160	1,479	146.0%
clip	2,670	3,300	1,200	605	198.3%

TABLE VI: The minimum value of a True output (in volts) is at least 10 times larger than the corresponding maximum value of a False output (in volts). Hence, it is feasible to distinguish false values from true values unambiguously.

#bits/ input	Maximum value of false signal	Minimum value of true signal
2	4×10^{-5}	0.1
4	8×10^{-5}	0.056
8	16×10^{-5}	0.031
16	31×10^{-5}	0.017
32	22×10^{-5}	0.009
64	20×10^{-5}	0.005
128	17×10^{-5}	0.002

We present the quality of the outputs for a representative selection of n-bit adders in Table VI. As the number of bits double, we observe that the minimum value of the true signal is approximately reduced by a factor of 2. For adders upto 128 bits, the ratio of the true signal to the false signal is more than 10 and hence, it is clearly feasible to distinguish the true output signal of the crossbar from the false output signal. However, for larger circuits, one can readily envision a scenario where the false and the true signals have similar and possibly indistinguishable values.

V. Conclusions and Future Work

Emerging high-density non-volatile memory technologies such as memristive RAMs provide a new opportunity for coupling computation with memory on the same fabric. Earlier approaches to computing using crossbars have suffered from one or more of the following drawbacks: (i) the requirement to fabricate memristors in forms other than dense nanoscale crossbars, (ii) the need to synthesize enormously large crossbars that make the approach impractical, (iii) the requirement of solving first-order logic synthesis problems that are too large for existing decision procedures, or (iv) the inability to reuse the existing software stack. We focus on computation of Boolean formula using structurally constrained crossbars and demonstrate the following:

1) The size of our crossbar circuits can be exponentially more succinct than the size of those designed using structural induction on negation normal forms [8].
2) We have synthesized crossbar circuits for Boolean formula that are two orders of magnitude larger than those designed using satisfiability-solving based automated synthesis by [6].
3) We have shown that Boolean formula can be computed using crossbars at most linear in the size of the ROBDD representation of the formula. Our results provide the first characterization of Boolean formula that can be computed efficiently using compact nanoscale crossbars.

Several exciting directions for future work remain open. First, the use of decision diagrams creates nanoscale crossbars that are largely empty. For large Boolean formula, more than 90% of the crossbar is turned off at all times. Does there exist a crossbar design for n-bit addition that is more compact than the design based on its Boolean decision diagram? We conjecture that the answer is in the affirmative.

Second, Boolean decision diagrams are not suitable for succinctly representing several interesting Boolean formula, such as multiplication. Our method does not exploit the canonical nature of BDDs and hence other more succinct but non-canonical representations like Free Boolean Decision Diagrams may serve as adequate representations for synthesizing nanoscale memristor crossbars from Boolean formula.

Third, Table VI shows how the quality of the true signal computed by the crossbar deteriorates as the size of the crossbar increases. We are investigating the synthesis of N parallel sneak paths for Boolean formula that reduce the effective resistance of the crossbar by N. Such an approach will permit us to restore the logic values to a range where true values can be distinguished from false values even for crossbars corresponding to very large Boolean formula.

Finally, several approaches have been suggested for circumventing naturally-occurring sneak paths [18] that provide new design opportunities to suppress some sneak paths while exploiting the rest of these naturally-occurring paths for computation.

Acknowledgment

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] I. Vourkas and G. C. Sirakoulis, "Memristive crossbar-based nonvolatile memory," in *Memristor-Based Nanoelectronic Computing Circuits and Architectures*. Springer, 2016, pp. 101–147.
[2] X. A. Tran, E. H. Toh, and E. K. B. Quek, "Non-volatile resistive random access memory crossbar devices with maximized memory element density and methods of forming the same," Mar. 24 2016, US Patent 20,160,087,197.
[3] R. Tetzlaff, *Memristors and Memristive Systems*, 1st ed. Springer-Verlag New York, 2014.
[4] A. Adamatzky and L. Chua, *Memristor Networks*, 1st ed. Springer Science & Business Media, 2013.
[5] A. Velasquez and S. K. Jha, "Parallel computing using memristive crossbar networks: Nullifying the processor-memory bottleneck," in *9th International Design & Test Symposium (IDT), 2014*. IEEE, 2014, pp. 147–152.
[6] ——, "Automated synthesis of crossbars for nanoscale computing using formal methods," in *Nanoscale Architectures (NANOARCH), 2015 IEEE/ACM International Symposium on*. IEEE, 2015, pp. 130–136.
[7] ——, "Fault-tolerant in-memory crossbar computing using quantified constraint solving," in *Computer Design (ICCD), 2015 33rd IEEE International Conference on*. IEEE, 2015, pp. 101–108.
[8] S. K. Jha, D. Rodriguez, J. E. V. Nostrand, and A. Velasquez, "Computation of boolean formulas using sneak paths in crossbar computing," Patent US 9,319,047, April 19, 2016.
[9] Z. Alamgir, K. Beckmann, N. Cady, A. Velasquez, and S. K. Jha, "Flow-based computing on nanoscale crossbars: Design and implementation of full adders," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 1870–1873.
[10] S. Chakraborti, P. V. Chowdhary, K. Datta, and I. Sengupta, "BDD based synthesis of boolean functions using memristors," in *2014 9th International Design and Test Symposium (IDT)*. IEEE, 2014, pp. 136–141.
[11] S. Shirinzadeh, M. Soeken, P.-E. Gaillardon, and R. Drechsler, "Fast logic synthesis for RRAM-based in-memory computing using majority-inverter graphs," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 948–953.
[12] C.-Y. Lee, "Representation of switching circuits by binary-decision programs," *Bell System Technical Journal*, vol. 38, no. 4, pp. 985–999, 1959.
[13] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *Computers, IEEE Transactions on*, vol. 100, no. 8, pp. 677–691, 1986.
[14] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 156–160.
[15] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Rogers, "A memristor device model," *IEEE electron device letters*, vol. 32, no. 10, pp. 1436–1438, 2011.
[16] B. J. Choi, A. C. Torrezan, J. P. Strachan, P. Kotula, A. Lohn, M. J. Marinella, Z. Li, R. S. Williams, and J. J. Yang, "High-speed and low-energy nitride memristors," *Advanced Functional Materials*, 2016.
[17] S. Yang, *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC), 1991.
[18] A. Levisse, B. Giraud, J. No, M. Moreau, J. Portal *et al.*, "Sneakpath compensation circuit for programming and read operations in rram-based crosspoint architectures," in *2015 15th Non-Volatile Memory Technology Symposium (NVMTS)*. IEEE, 2015, pp. 1–4.