A Compact 8-bit Adder Design using In-Memory Memristive Computing: Towards Solving the Feynman Grand Prize Challenge

Dwaipayan Chakraborty Sunny Raj Sumit Kumar Jha
Computer Science Department
University of Central Florida
4000 Central Florida Blvd
Orlando, Florida, 32816
Email: {dchakra, sraj, jha}@cs.ucf.edu

***Abstract*—We introduce a new compact in-memory computing design for implementing 8-bit addition using eight vertically-stacked nanoscale crossbars of one-diode one-memristor 1D1M switches. Each crossbar in our design only has 5 rows and 4 columns. Hence, the design may be used to fabricate a compact 8-bit adder that meets the size constraint of 50nm $\times$ 50nm $\times$ 50nm imposed by the electrical component of the Feynman Grand Prize. The potential availability of sub-5nm nanoscale memristors and single-molecule diode devices coupled with the ability to fabricate high-density nanoscale memristor crossbars suggests that our design may eventually be fabricated to meet the size constraints of the Feynman Grand Prize.**

I. Introduction

The design space of nanoscale computing systems can be explored using a variety of benchmarks and performance metrics, including energy, speed and robustness. An orthogonal approach to such a design exploration is to create innovative designs that can respond to well-known grand challenge problems. Several other areas of computing such as image processing, robotics and autonomous driving have seen leaps in technological maturity based on a sustained pursuit of such grand challenge problems.

The Feynman Grand Prize offered by the Foresight Institute requires the design, construction and demonstration of a nanoscale digital computing device capable of performing 8-bit addition in a cube of edge length 50 nm. Assuming that the device is being created using nanoscale memristors of size 3nm [1] and the inter-memristor gap is only 1nm, the 8-bit adder design can still only use crossbars of size 12 $\times$ 12. Such a compact implementation is not feasible using existing design approaches including our own recent related work [2]–[5].

A natural next step is to use a 3-dimensional stack of nanoscale memristor crossbars. For the sake of simplicity in fabrication, we may require that the different layers in the stack have limited and structured interactions. In an ideal setting, such interactions between different layers should be achieved by external connections among the layers. Each crossbar layer can implement a single full-adder and eight such crossbar layers can be stacked on top of each other. Each crossbar layer in this stack needs to implement a full-adder circuit within an area of 50nm $\times$ 50nm. We pursue this approach of assembling crossbars into 3-D stacks with limited external connections.

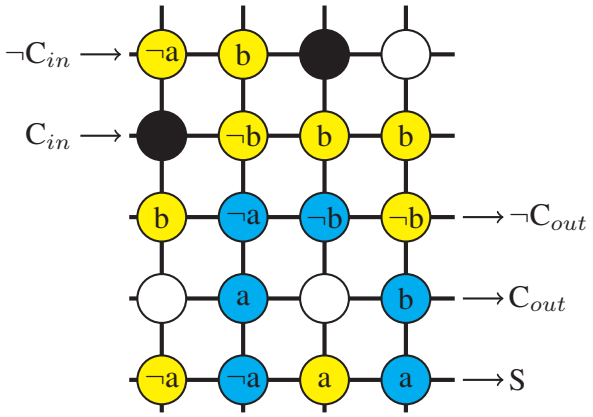


Fig. 1. Modular crossbar design for a 1-bit full adder using 1D1M switches. The crossbar accepts the values of the addends as stored values in the colored memristors. The carry-in inputs from the previous 1-bit adder are received on the top left of the crossbar and the carry-out bits for the next full adder are produced at the bottom right of the crossbar. The lowest nanowire computes the sum.

We make the following new contributions:

- We present a modular 1-bit full adder crossbar design (see Figure 1) composed of 1D1M devices [6]. The addends are stored on the memristors in the crossbar. The sum and (negated) carry-out bits flow out of the right side of the crossbar. The (negated) carry-in bit for the ripple adder is fed from the top-left of the crossbar.
- We demonstrate how a stack of 8 such nanoscale crossbars can be arranged in a 3-dimensional cube such that the (negated) carry-out from one crossbar can be fed to the (negated) carry-in of the crossbar just above it.

The performance of our design in terms of energy and switching speed has not been evaluated due to the relatively poor availability of computational models for state-of-the-art 1D1M devices [7].

978-1-5090-6037-5/17/$31.00

II. Related Work

The fourth fundamental electrical element, the memristor, was theoretically predicted by Chua [8] in 1971. There is evidence that memristors were being deployed in experimental practice during the early 20th century [9]. However, a physical nanoscale realization of the device was only popularized in 2007 [10]. This renewed interest in memristors coupled with the end of Dennard scaling for CMOS-based devices has opened up new avenues for the design of computational circuits [3], [11], [12].

A memristor is essentially a resistor with memory – a non-volatile memory element which also serves as a nanoscale switch. The switching property of memristors coupled with its small size and low power consumption [7] enable them to be used as junction switches in matrices of intersecting nanowires. This specific architecture is also termed as the *crossbar* architecture, and it allows more storage and computational power to be packed into smaller spaces.

Memristors have been used to implement neuromporphic computing architectures with great success. In some of these architectures, simple CMOS circuits are used to digitize the input as spikes and then spiking-based neuromorphic systems [13]–[15] are used to implement CNNs.

Memristor-based memories have also been used to accelerate general-purpose computing. Gaillardon et. al [16] have proposed Programmable Logic-in-Memory (PLiM) based on a ReRAM crossbar array. In this approach, an instruction similar to the native majority function is implemented on top of ReRAM. Other instructions can be compiled down to a sequence of majority operations [17] that must be performed sequentially. Bhattacharjee et al. have extended this approach to permit parallel execution of each word of a VLIW instruction set [18].

Boolean formulae have also been implemented using memristors. Building upon the design principles used to implement logical circuits, memristor circuits have indeed been designed to compute Boolean logical operations. Elementary logical operations [19]–[23], such as material implication and universal NAND gates, have been designed using repeated switching of nanoscale memristors. Circuits made of such elementary operations have been used to implement combinational circuits from the ISCAS89 benchmark using BDDs [24] as well as and-inverter graphs [11]. In principle, such simple logical circuits can be used to implement arbitrary loop-free programs. However, these designs still focus on computation and do not address the bottleneck between the memory and the processing units.

Since our modular one-bit adder design of 5 rows and 4 columns is driven by the spatial constraints of the Feynman Grand Prize, we highlight that device research has produced memristors with a feature size less than 5nm [7]. Memristors with dimensions as small as 1nm $\times$ 3nm have been recently fabricated [1]. This pursuit of atomic-scale memristors, combined with recent advances in molecular diodes [25], perhaps paves the way for the future development of unidirectional resistive switches that may be sufficiently small to enable our 8-bit adder design to be fabricated within the space constraints imposed by a cube of side 50nm.

A. Flow-based crossbar computing

Our results are most closely related to recent work on use of flows for performing computing on nanoscale crossbars. For computation using flows, a Boolean formula is first encoded into a crossbar design. The memristors in the crossbar can be permanently OFF (high resistance state), permanently ON (low resistance) and they can also take on values of the literals in the Boolean formula. The concept is illustrated by a very simple example in Fig. 2. A white circle represents a memristor in the OFF state, a black circle represents a memristor in the ON state and a yellow circle represents a memristor corresponding to the literal that labels the memristor. If a given set of inputs results in the formula being evaluated to true, then the flow introduced in the top row reaches the bottom row and we have a HIGH output. If a certain set of inputs results in the formula being evaluated to false, then the flow introduced in the top row does not reach the bottom row, and we have a LOW output.

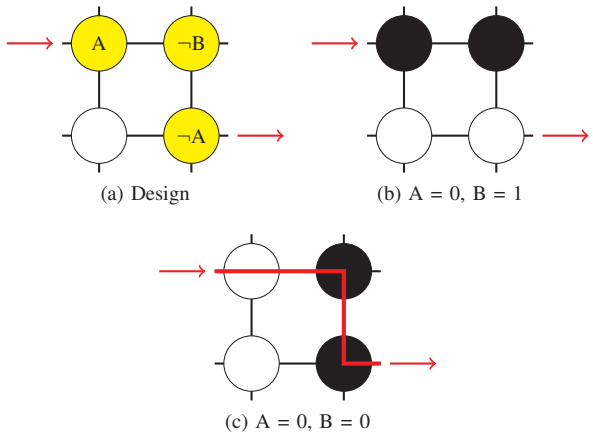


Fig. 2. Example for computation using sneak paths. The crossbar will cause a current to flow from the top to the bottom horizontal nanowire if and only if both A and B are false. Figure (c) shows the flow of current using the deep red line. Figure (b) shows an example where the flow cannot transfer from the top horizontal nanowire to the right vertical nanowire and hence does not reach the bottom horizontal nanowire.

The approach has been used to design a 1-bit full adder using nanoscale memristors that has been experimentally evaluated to be superior in power-product delay than a contemporary CMOS adder [26]. However, the earlier 1-bit adder design can not be used to achieve our 3-D stackable 8-bit adder design because the adder design is *not modular*. Hence, two such 1-bit adder designs cannot be put together to produce a 2-bit adder. The flow-based computing approach has also been shown to scale up to 128-bit adders without degradation of the flow currents [3].

III. Design

We propose a new crossbar design for a modular full adder that computes the sum, the carry-out and the negation of the carry-out bit for 1-bit Boolean addition. The design is shown in Figure 1 and consists of five horizontal nanowires (rows) and four vertical nanowires (columns). The top row of the crossbar receives the negation of the carry-in bit while the second row from the top receives the carry-in bit. The crossbar has two orientations of 1D1M switches. The switches that only allow current flow from row to column are marked in yellow and the switches which only allow current flow from column to row are marked in blue. Switches that are permanently ON and permanently OFF are represented by black and white circles respectively. The outputs are obtained from the bottom row, the second row from the bottom and the third row from the bottom. If the sum bit is true, there is flow in the bottom row. Similarly, if the carry-out bit is true, the second row from the bottom has a flow; otherwise, the third row from the bottom has a flow

In order to successfully build a compact 8-bit adder, eight replicas of the full adder described above are stacked on top of each other. Three such crossbar layers with external interconnects are shown in Figure 3. The inputs to the crossbars should be loaded into the memristors according to the design in Figure 1. The carry-in bit for the first crossbar should be set to 0 and the negation of the carry-in but should be set to 1. For all the other carry-in bits, the nanowire corresponding to the carry-out bit of the previous full-adder unit should be fed into the carry-in bit of the next full-adder. Similarly, the negation of the carry-out bit should be fed into the negation of the carry-in bit of the next full adder unit. The eight crossbars will produce the eight sum bits of 8-bit adder on their lowest nanowires.

The crossbar design has several interesting features when compared to earlier crossbar designs [3], [5], [26]:

(i) The one-bit full adder proposed in [26] is not modular and cannot be used to build a ripple-carry adder.

(ii) The 8-bit adder presented in [3] requires a single crossbar layer with more than a hundred memristors. It is highly unlikely that such a design can ever be fabricated within the space constraints of the Feynman Grand Challenge. The earlier design compares poorly in size with our approach that needs only 20 memristors in each layer.

(iii) The approach presented in [5] combines machine learning and formal methods to synthesize very compact 4-bit adders with 8 rows and 5 columns; but, the methodology has not been used to design 8-bit adders. However, it is possible that the machine learning approach may be extended to create compact 8-bit adders in the future.

Earlier crossbar approaches have often used memristor crossbars while our approach requires the use of 1D1M devices. Modularity requires our flow-based computing crossbar designs to receive two input flows: the carry-in bit and its negation. In a purely memristive design, both these flows get mixed up without the use of unidirectional 1D1M devices.

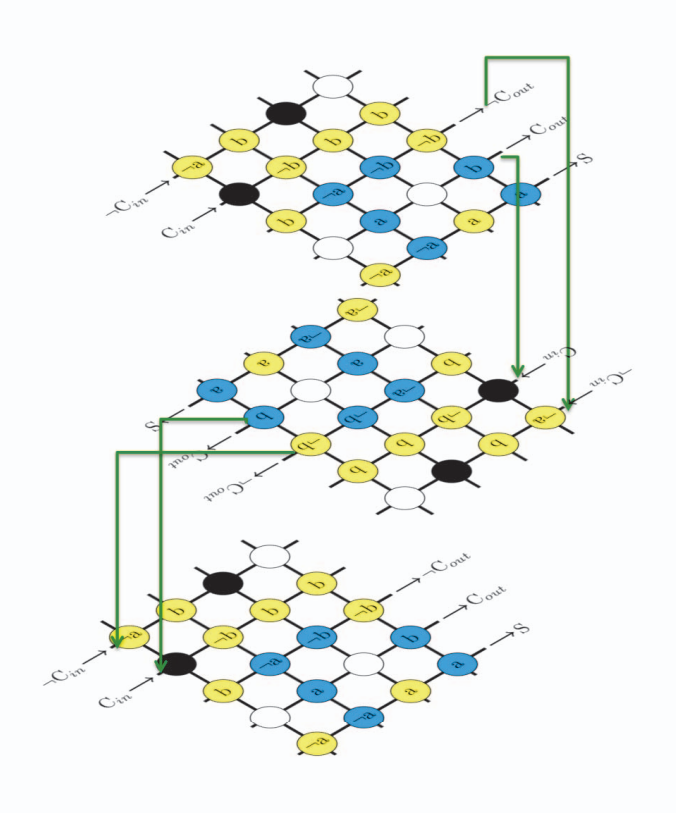


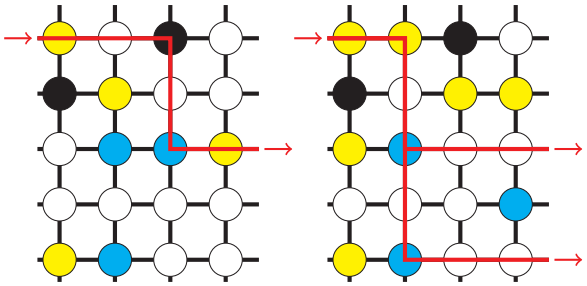
Fig. 3. Illustration of 3 crossbars being stacked in 3-D with only external connections. Every alternate crossbar design is flipped in order to ensure easy routing of the external interconnects. Each layer is loaded with its own addends and the carry-out of the previous layer. Each layer produces the sum for that bit and the carry-in of the next crossbar layer.

IV. Results

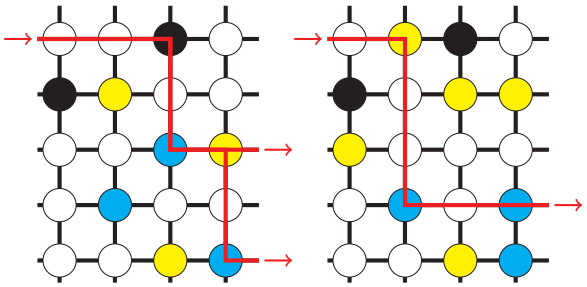
Three different studies have been performed on the correctness of the crossbar design presented in Figure 1. First, we show the flow of current through the modular 1-bit full adder crossbar that leads the crossbar to perform the correct arithmetic. For demonstrative purposes, we then show how the flow of current through 8 stacked crossbars leads to correct computation on one illustrative input. Finally, we discuss the feasibility of distinguishing a no-flow state of a crossbar from a flow state using quantitative arguments.

A. Validation

In order to clearly exhibit how the one-bit modular full adder performs the necessary computation, we enumerate the logical response of the crossbar for all possible inputs. The sneak paths and outputs in Figure. 4 show the four input patterns when the carry-in bit is false. The yellow circles represent the switches in the ON state that allow flow only from row to column while the blue circles are switches that only allow flow from coulmn to row. Similarly, white circles represent switches that are in the OFF state and black circles represent the switches that are permanently in the ON state.

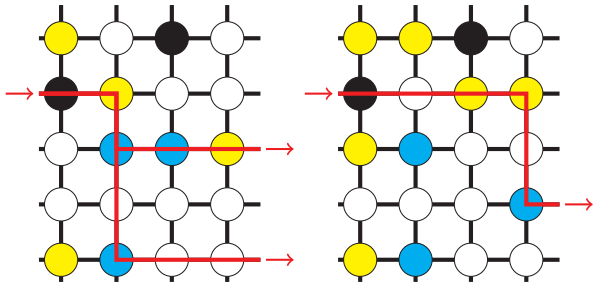


(a) A=0, B=0, S=0, C_{out}=0 (b) A=0, B=1, S=1, C_{out}=0

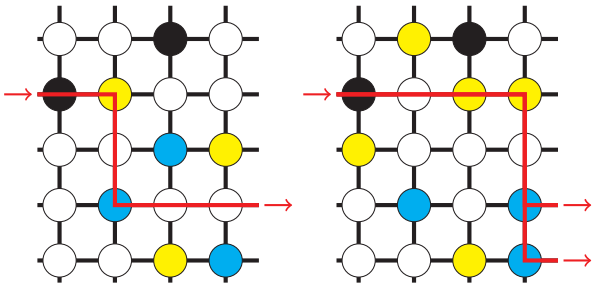


(c) A=1, B=0, S=1, C_{out}=0 (d) A=1, B=1, S=0, C_{out}=1

Fig. 4. Sneak paths and crossbar outputs when there is no carry-in input i.e. there is a flow on the topmost horizontal nanowire. The black devices are turned on while the white devices are turned off. The blue-colored devices only allow current to flow from a column to a row while the yellow-colored devices only allow current to flow from a row to a column. The flows from the input row ($\neg C_{in}$) to the output rows are shown with bold red lines.



(a) A=0, B=0, S=1, C_{out}=0 (b) A=0, B=1, S=0, C_{out}=1



(c) A=1, B=0, S=0, C_{out}=1 (d) A=1, B=1, S=1, C_{out}=1

Fig. 5. Sneak paths and crossbar outputs when there is a carry-in input on the second-highest horizontal nanowire. The black (white) devices are turned on (off). The blue (yellow) colored devices only allow current to flow from a column (row) to a row (column). The flows from the input row (C_{in}) to the output rows are shown with bold red lines. All flows to outputs labeled 0 are blocked by turned-off memristors.

There is a flow on the lowest horizontal nanowire if and only if the sum bit is true (see Figure 4(b) and (c)). Similarly, there is a flow on the second-lowest horizontal nanowire if and only if the carry-out bit is true (see Figure 4(d)). The flow caused by the negation of the carry-out bit can be observed on the third-lowest horizontal nanowire in Figure 4(a), (b), and (c).

Figure. 5 shows the behavior of the modular 1-bit adder when the carry-in bit is true. Again, we can verify that the presence of flow on the bottommost horizontal nanowire correctly corresponds to the output sum bit being true. Similarly, the next two horizontal crossbars correctly correspond to the carry-out and the negation of the carry-out bit respectively.

B. 8-bit adder

The modular one-bit adder can be stacked on top of each other to form a 8-bit adder as shown in Figure 3. We show how the addition of 10101111 with 11011001 can be performed using this 8-bit adder in Figure 6. First, we load the least significant bits of the inputs on the first crossbar layer shown in Figure 6(a) and load the subsequent bits on the next layers. The i^{th} crossbar layer is loaded with the i^{th} bit of the input.

Then, the carry-out bit of the i^{th} crossbar is connected to the carry-in bit of the $(i+1)^{th}$ crossbar. Similarly, the negation of the carry-out bit of the i^{th} crossbar is connected to the negation of the carry-in bit of the $(i+1)^{th}$ crossbar.

The whole 3-dimensional package receives an initial flow on the top nanowire of the first layer. This flow corresponds to the negation of the carry-in of this crossbar layer indicating that there is no carry-in at this stage. Then, this flow passes through the nanowires and memristor switches performing the required 8-bit addition computation. The i^{th} sum output bit can be read at the lowest nanowire of the i^{th} layer.

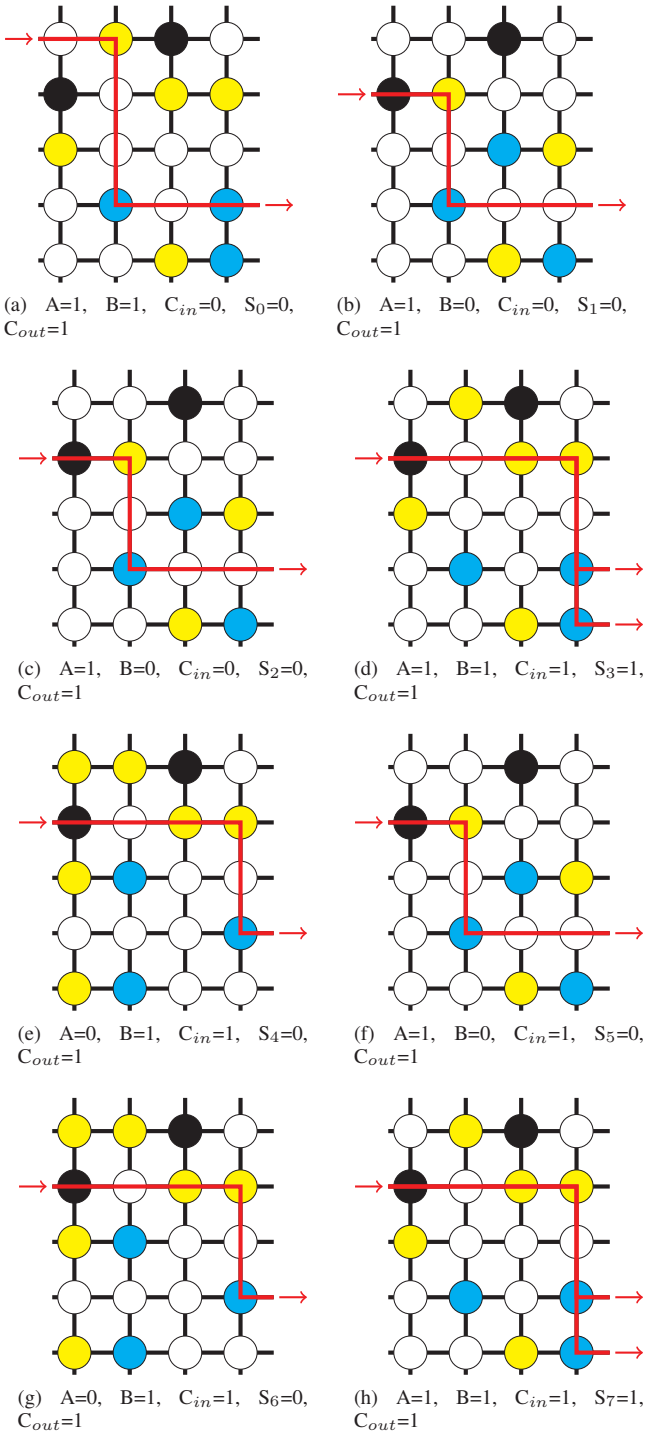
In our running example, the first crossbar layer computes the sum of A=1 and B=1 to obtain a sum of 0 and a carry-out of 1. See Figure 6(a). The carry-out then reaches the carry-in of the second crossbar. This crossbar performs a sum of A=1 and B=0 together with this carry-in to obtain a sum of 0 and a carry-out of 1. See Figure 6(b). The third crossbar in Figure 6(c) performs an identical computation.

The fourth crossbar receives a carry-in from the third crossbar and computes the sum of A=1 and B=1 with this carry-in. It produces a sum of 1 (on the lowest nanowire) and also produces a carry-out of 1. See Figure 6(d).

The process continues through all the 8 crossbar layers of the 8-bit adder and we obtain the sum of 10001000 with a carry-out bit of 1. The presence of both the carry-out and negated carry-out bits enables us to design a modular one-bit adder that can just be stacked on top of each other to form an 8-bit adder.

C. Flow Quality

The flow of current through several turned-on memristors may be sufficiently impeded to look like the flow through a turned-off memristor. This can be a problem when very deep



(a) A=1, B=1, C_{in}=0, S_0=0, C_{out}=1

(b) A=1, B=0, C_{in}=0, S_1=0, C_{out}=1

(c) A=1, B=0, C_{in}=0, S_2=0, C_{out}=1

(d) A=1, B=1, C_{in}=1, S_3=1, C_{out}=1

(e) A=0, B=1, C_{in}=1, S_4=0, C_{out}=1

(f) A=1, B=0, C_{in}=1, S_5=0, C_{out}=1

(g) A=0, B=1, C_{in}=1, S_6=0, C_{out}=1

(h) A=1, B=1, C_{in}=1, S_7=1, C_{out}=1

Fig. 6. Tracing two 8-bit inputs through the ripple-carry adder. Addition of 10101111 with 11011001 produces 10001000 with a carry-out bit of 1.

computations involving thousands of memristors are being performed without buffers that pull up the voltage of a degraded signal. In our setting, a careful study of Figures 4 and 5 shows that each flow passes through only two memristors. Hence, our flow through 8 crossbar layers of the 8-bit adder only pass through 16 memristors. Since the ratio of the high resistance to the low resistance states of the memristor can exceed 10^3 [27] and has been reported to be as high as 10^7 [28], the resistance of 16 turned-on memristor in series is at least two orders of magnitude smaller than a single turned-off memristor. Hence, the flow and no-flow states of the nanowires in our design are readily distinguishable.

V. Conclusions and Future Work

We suggest a new in-memory 8-bit adder design using nanoscale crossbars of 1D1M (1-diode, 1-memristor) junction switches and computation using flows of current through nanowires. Our design of the modular full adder that can be used to build a compact 8-bit adder requires the use of one diode-one memristor 1D1M devices instead of pure memristors in order to separate the two logically different inputs – carry-in and the negation of the carry-in. Given the 3nm size of state-of-the-art memristors [1] and the advances in single molecule diode technologies, our design may be fabricated within the space constraints imposed by the Feynman Grand Prize in the near future.

Several directions for future research remain open. Can we fabricate a smaller modular 1-bit full adder using fewer rows or columns? Can we build 2-bit adders that are only slightly larger than the 1-bit adder but require us to put together only 4 different crossbars on a stack to form a compact 8-bit adder? We believe that the answer to the latter question should be in the affirmative.

In [5], it has been shown that a 4-bit adder can be designed using 8 rows and 5 columns. Is it possible to design a modular 4-bit adder using 1D1M devices in a similar crossbar area budget? We conjecture that the answer should be in the affirmative but might require different synthesis algorithms.

An interesting direction for theoretical research would be to study the size of crossbars. What is the smallest crossbar that implements a given Boolean formula? In [3], it has been shown that the size of a crossbar can be upper bounded by a linear function of the size of the Boolean Decision Diagram (BDD). However, it has not been shown that the bound is tight and there may exist formula for which crossbars are much smaller than the corresponding BDDs. The design of compact crossbars for implementing multiplication would be of practical interest.

Given the current maturity of the work on flow-based memristive computing, it should be feasible to translate loop-free C programs into one or more crossbar designs algorithmically and verify such a translation using modern parallel circuit simulation software. Our team has started building such a cyber-infrastructure at http://sumitkumarjha.com/www/jha/software.

REFERENCES

[1] K.-S. Li, C. Ho, M.-T. Lee, M.-C. Chen, C.-L. Hsu, J. Lu, C. Lin, C. Chen, B. Wu, Y. Hou *et al.*, "Utilizing sub-5 nm sidewall electrode technology for atomic-scale resistive memory fabrication," in *VLSI Technology (VLSI-Technology): Digest of Technical Papers, 2014 Symposium on*. IEEE, 2014, pp. 1–2.

[2] A. Velasquez and S. K. Jha, "Parallel boolean matrix multiplication in linear time using rectifying memristors," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1874–1877.

[3] D. Chakraborty and S. K. Jha, "Automated synthesis of compact crossbars for sneak-path based in-memory computing," in *Design Automation and Test in Europe (DATE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 770–775.

[4] 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 *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1870–1873.

[5] D. Chakraborty and S. K. Jha, "Design of compact memristive in-memory computing systems using model counting," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 2655–2658.

[6] L. Zhang, L. Zhu, X. Li, Z. Xu, W. Wang, and X. Bai, "Resistive switching mechanism in the one diode-one resistor memory based on p+-Si/n-ZnO heterostructure revealed by in-situ TEM," *Scientific Reports*, vol. 7, 2017.

[7] W. Kim, A. Chattopadhyay, A. Siemon, E. Linn, R. Waser, and V. Rana, "Multistate memristive tantalum oxide devices for ternary arithmetic," *Scientific Reports*, vol. 6, 2016.

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

[9] G. Gandhi, V. Aggarwal, and L. Chua, "Coherer is the elusive memristor," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 2245–2248.

[10] 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, 2008.

[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 *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 2016, pp. 948–953.

[12] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2014.

[13] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[14] C. Liu, B. Yan, C. Yang, L. Song, Z. Li, B. Liu, Y. Chen, H. Li, Q. Wu, and H. Jiang, "A spiking neuromorphic design with resistive crossbar," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[15] C. Liu, Q. Yang, C. Zhang, H. Jiang, Q. Wu, and H. H. Li, "A memristor-based neuromorphic engine with a current sensing scheme for artificial neural network applications," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017, pp. 647–652.

[16] P.-E. Gaillardon, L. Amarú, A. Siemon, E. Linn, R. Waser, A. Chattopadhyay, and G. De Micheli, "The programmable logic-in-memory (plim) computer," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 2016, pp. 427–432.

[17] M. Soeken, S. Shirinzadeh, P.-E. Gaillardon, L. G. Amarú, R. Drechsler, and G. De Micheli, "An MIG-based compiler for programmable logic-in-memory architectures," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*. IEEE, 2016, pp. 1–6.

[18] D. Bhattacharjee, R. Devadoss, and A. Chattopadhyay, "ReVAMP: ReRAM based VLIW architecture for in-memory computing," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017*, 2017.

[19] E. Gale, B. de Lacy Costello, and A. Adamatzky, *Boolean Logic Gates from a Single Memristor via Low-Level Sequential Logic*. Springer, 2013, pp. 79–89.

[20] S. Kvatinsky, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Memristor-based IMPLY logic design procedure," in *Computer Design (ICCD), 2011 IEEE 29th International Conference on*. IEEE, 2011, pp. 142–147.

[21] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *Proceedings of the 2009 IEEE/ACM International Symposium on Nanoscale Architectures*. IEEE Computer Society, 2009, pp. 33–36.

[22] X. Sun, G. Li, L. Ding, N. Yang, and W. Zhang, "Unipolar memristors enable "stateful" logic operations via material implication," *Applied Physics Letters*, vol. 99, no. 7, p. 072101, 2011.

[23] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie, "Design implications of memristor-based RRAM cross-point structures," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 2011, pp. 1–6.

[24] S. Chakraborti, P. V. Chowdhary, K. Datta, and I. Sengupta, "BDD based synthesis of boolean functions using memristors," in *Design & Test Symposium (IDT), 2014 9th International*. IEEE, 2014, pp. 136–141.

[25] A. C. Aragonès, N. Darwish, S. Ciampi, F. Sanz, J. J. Gooding, and I. Díez-Pérez, "Single-molecule electrical contacts on silicon electrodes under ambient conditions," *Nature Communications*, vol. 8, 2017.

[26] A. Velasquez and S. K. Jha, "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.

[27] F. Gul and H. Efeoglu, "ZnO and ZnO_{1-x} based thin film memristors: The effects of oxygen deficiency and thickness in resistive switching behavior," *Ceramics International*, 2017.

[28] C. Hu, M. D. McDaniel, J. G. Ekerdt, and T. Y. Edward, "High on/off ratio and quantized conductance in resistive switching of tio_2 on silicon," *IEEE Electron Device Letters*, vol. 34, no. 11, pp. 1385–1387, 2013.