Mapping of Image and Network Processing Tasks on High-Throughput CMOL FPGA Circuits

Advait Madhavan and Dmitri B. Strukov
Department of Electrical and Computer Engineering
University of California Santa Barbara, Santa Barbara, CA 93106, USA
{advait, strukov}@ece.ucsb.edu

Abstract— **A simple two-terminal memristive device has excellent scaling properties. For example, devices with footprint below 10×10 nm^2 have been recently demonstrated and crossbar structures provide means of sustaining memristor density in large-scale circuits. While taking advantage of high density memristive devices is relatively straightforward in crossbar memory circuits, doing so efficiently in digital logic circuits still remains challenging. For example, only a small fraction (less than 1% on average) of memristive devices is actively utilized, i.e. turned to highly conductive state, in CMOL FPGA circuits which are configured to implement representative benchmark circuits. The main contribution of this paper is to demonstrate that such utilization can be much higher, more than 12%, in certain variety of CMOL FPGA circuits which are specifically designed for high throughput processing of streaming data. The high memristor device utilization is demonstrated by performing detailed mapping of network and image processing tasks and is mainly due to efficient use of high fan-in logic gates implementing exact and approximate pattern matching operations with streaming data. As a result of high utilization proposed circuits are estimated to have much higher computational throughput as compared to traditional approaches and represent a killer application which capitalizes efficiently on the density advantages of memristive devices.**

Keywords: Memristor, Programmable circuits, Hybrid circuits, Image Processing, Network Processing, CMOL FPGA.

I. INTRODUCTION

In its simplest form, a memristive device consists of three layers: top and bottom (metallic) electrodes and a thin film of some material which can undergo resistance switching [1]. By applying a voltage bias across the electrodes of such a device, the electrical conductivity of the thin film can be changed reversibly and retained for a sufficiently long time. Because of an active area which is potentially very small, just few atoms wide, for devices based on ion migration mechanism [2, 3], and a simple structure, which is conducive to aggressive lithographic and other patterning techniques, these memristive devices have excellent scaling properties. For example, several groups have recently shown memristors with device area below 10×10 nm^2 [4, 5], which is defined by the overlap area of bottom and top electrodes.

The most straightforward application for memristive devices is in crossbar passive memories [6]. The crossbar structure, which is based on mutually perpendicular layers of parallel wires (electrodes) with integrated thin-film devices at the cross-points, allows sustaining small footprints of single devices in large-scale circuits, thus fully capitalizing on the density of memristive devices.

Similarly, logic circuits based on crossbar structures, e.g. based on CMOL hybrids [6, 7], are the most practical. However, taking full advantage of memristor density in logic circuits is less straightforward. To see why this is the case it is natural to divide proposed crossbar-based logic circuits into two groups. (While most of these concepts do not explicitly rely on memristive devices, they share the same idea of capitalizing on the density of simple programmable two terminal crosspoint nanodevices and hence relevant for this discussion.)

In the first type of circuits, crosspoint nanodevices implement programmable interconnect among CMOS logic gates [8-13]. The nanodevices in the second type of crossbar circuits are employed more efficiently, either as a part of diode logic [14-16], Goto pair [17, 18], to implement look-up-table or PLA like computation [17, 19, 20], or material implication logic [21, 22]. While all these concepts were estimated to improve performance of conventional circuits, in all the cases only a small fraction of nanodevices are utilized efficiently. For the former circuits, this is mostly due to limited interconnect needs, guided by Rent's rule. For the latter, as the optimal gate fan-in is typically small for general purpose computations, i.e. < 10 for PLA or LUT based designs and even smaller for sea-of-gates approaches, only a small fraction of crosspoint devices are actively utilized (while the majority of crosspoint devices are turned to the high resistance state, effectively turned "off", and do not participate in computation). For example, less than 1% of cross-point devices are actively used when mapping representative benchmarks on CMOL FPGA circuits [14]. In circuits relying on keeping logic computation in crossbar memory or material implication logic all crosspoint devices might be utilized but only small fraction of them can be used at a time (i.e. during a clock cycle) because of decoder bottleneck.

The purpose of this paper is to show that utilization of crosspoint (memristive) devices can be much higher - the fraction of devices which are *actively and simultaneously utilized* at any given cycle can be as high as 12%. Such dramatic improvement of utilization is due to (i) choice of information processing applications which rely on extensive data processing between streaming data and fixed infrequently changing patterns; (ii) use of modified CMOL FPGA which is specifically customized for streaming data; and (iii) mapping

scheme optimizing utilization of memristive devices and maximizing information processing throughput.

In the next section we briefly describe CMOL FPGA circuits specifically designed for high-throughput computation. Such circuits have been introduced earlier [23, 24] and here are extended to include linear threshold functionality. The description of CMOL FPGA circuits is followed by their architectural abstraction which is especially useful when performing application mapping. The main results of this paper, a specific mapping for representative network and image processing of low-level tasks is described next and the paper is concluded by discussion of potential performance of considered circuits.

II. High Throughput CMOL FPGA Architecture

Similar to the original CMOL circuits [6, 14, 25, 26], high-throughput CMOL FPGA is based on the following key features: (i) an area interface between the CMOS layer and nano subsystem (Fig. 1d); (ii) segmented crossbar array(s) layer which is rotated by a specific angle with respect to the mesh of CMOS-controlled vias (Fig. 1d); and (iii) a double decoding scheme that provides unique access to each cross-point memristive device.

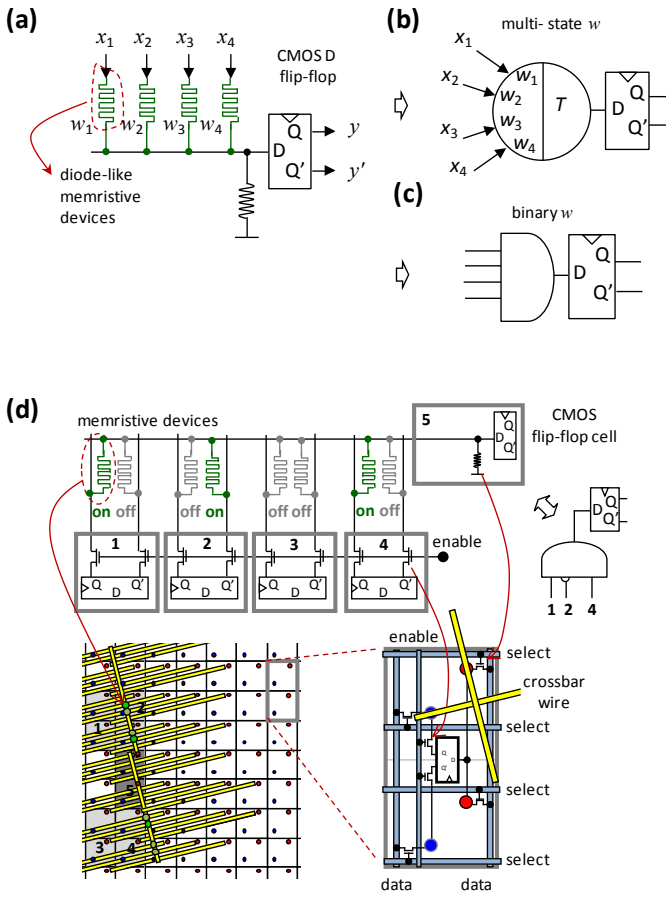

Figure 1. High throughput CMOL FPGA circuits: (a) Equivalent circuit implementation (b) linear threshold gate or (c) diode-resistor AND logic combined with D flip-flop gate, and (d) top view illustration of CMOL FPGA circuits with one crossbar layer and specific connectiveity domain (r = 4) showing a particular mapping example. On panel d, for clarity only nanodevices turned to the ON state and participating crossbar wire segments, i.e. those which overlap with two wire segments accessed by the input of cell "5", are shown at the left part of the figure.

The basic atomic cell in traditional CMOL consists of two vias that interface to the crossbar layer above the CMOS stack. Two such cells are required to implement the unit cell in proposed CMOL FPGAs, which hosts a CMOS D-flip-flop (Fig. 1d). The flip-flop is connected to the crossbar by four vias: two output vias (shown with a blue circle) connected to the normal and complementary outputs of the flip-flop, and two input vias (shown with a red circle) that are connected to the D input. Clock signals are assumed to be routed using the CMOS subsystem (note: to avoid confusion in referring to different types of cells, the basic two-via cell will be called "basic cell" and the larger four-via cell will be called "flip-flop cell" or simply "cell").

In order to configure such a circuit to implement custom logic, first, the CMOS flip-flop is disabled in all cells by de-asserting "enable" line (Fig. 1d). As a result, any crosspoint memristive device (shown with a green circle in Fig. 1d) in the crossbar structure can be programmed to the ON or OFF state by utilizing the double decoding scheme of CMOL memory [6], which is implemented using CMOS pass transistors and data/select wires (Fig. 1d). Here we assume that the ON state of the device is nonlinear (e.g., having either diode or symmetric tunnel barrier-like I-V [1]).

After the programming stage, logic operations are implemented with either linear threshold gate (Figs. 1a, b) [27] by taking advantage of the analog properties of memristive devices, or diode-resistor logic formed by ON-state nanodevices and CMOS pass transistors (Fig. 1a, c, d) [14]. The signal restoration, inversion, and latching is performed by the D-flip-flop. For example, Figure 1d shows a particular example of implementing function $Q_1 \bullet Q_2' \bullet Q_4$ with the cell "5", where signals Q and Q' are routed from the output of the corresponding cells. This operation is equivalent to performing "exact" matching operation between the current values in cells 1, 2, 3, and 4 and pattern "10X1", which is stored by memristive devices. Note that the state of the memristive devices remain unchanged during the operation and only change during programming stage. It is also worth stressing that the specific functionality of the flip-flop cell and their connectivity is governed by the state of memristive devices while CMOS sub-layer only implements the D-flip-flop gate and memristor programming circuitry.

III. Circuit Architecture Abstraction

Let us now consider abstraction of the CMOL FPGA circuit architecture which is very convenient for further discussion of mapping.

One important characteristic of the CMOL FPGA circuit is its connectivity domain (Fig 2a), which is defined by topological parameter r [14]. In particular, r is proportional to the ratio of linear size of basic cell to the crossbar wire pitch. The total number of crosspoint memristive devices attached to crossbar wire fragment is always of the order of r^2. Every memristive device provides a means of connecting given basic cell to surrounding basic cells via nanowire-memristive

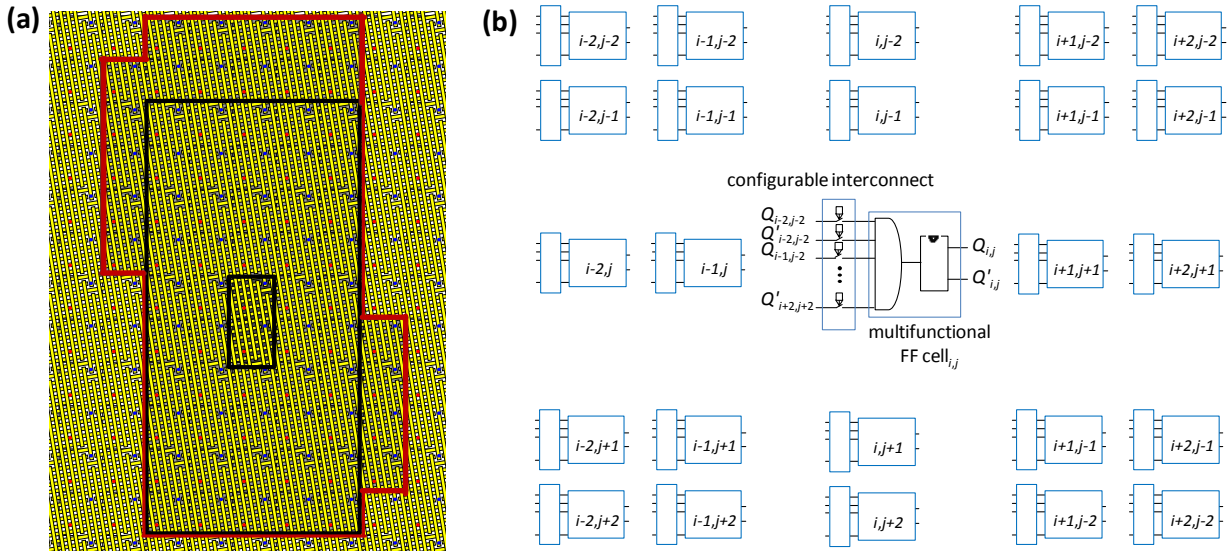

Figure 2. (a) Top view of crossbar circuit and input connectivity domain of flip-flop cell for specific topological parameter $r = 6$, and (b) equivalent sea-of-gate circuit architecutre for binary memristive devices implementing diode-resistor AND logic. On panel a, the set of basic cells bounded by red lines is a physically permitted domain, while the somewhat smaller number of cells surrounded by black lines is used in mapping examples. On panel b, AND gate should be replaced with linear threshold gate when analog properties of memristive devices are exploited.

device-nanowire link, so that the number of basic cells (M) in the connectivity domain is of the order of $M = r^2$.

Since a flip-flop cell consists of two basic cells and has two input crossbar nanowires, the input connectivity domain of the flip-flop cell is larger (Fig. 2a). The largest flip-flop cell connectivity domain (~ to M flip-flop cells or equivalently $2M$ basic cells) is achieved by having the least overlap between individual connectivity domains of basic cells comprising the flip-flop cell. For example, this could be implemented with the blue via having contacts at the edges of the crossbar wires as shown on Figure 2a.

To simplify mapping, we will further use artificially smaller (than physically permissible) rectangular shaped domains – for example domain of 5×5 flip-flop cells for topological parameter $r = 6$ (Fig. 2a). In this case, the high throughput CMOL FPGA architecture can be thought of as an array of multifunctional flip-flop cells. Every flip-flop cell can pass its outputs to or accept inputs from any of the flip-flop cells in 5×5 connectivity domain which is always centered with respect to a given flip-flop cell (Fig. 2b). Moreover, every flip flop cell can be configured to perform AND or linear threshold functions with normal or complimented outputs of flip-flop cells in its connectivity domain. Note that due to De Morgan's law and the presence of complementary output Boolean OR functions can also be realized for every flip-flop cell.

IV. Mapping Case Studies

Though any Boolean logic circuit can be implemented with the modified CMOL FPGA fabric, it is especially attractive for data streaming applications relying on high fan-in computations. Here we consider two such applications - network and image information processing tasks. To simplify our analysis we omit details of the particular algorithm implementation for these tasks but rather focus on performing the most important bottleneck operation during low level processing. For both cases such operation is often exact or approximate pattern matching, which could be efficiently implemented with wide fan-in gates in high-throughput CMOL FPGA fabric.

A. Network Processing

Let's consider one dimensional (1D) stream of data pushed through a very deep pipeline. A very common task is to check if the stream of data has a particular bit sequence which matches or is similar to a given one, from a large set of sequences of interest. For example, in deep-packet network filtering [28] the sequence in question can be a known signature of a computer virus. Other examples for applications requiring pattern matching with 1D streaming data include data processing in bioinformatics, network routing and string processing [29]. A common challenge for all these applications is to perform high throughput pattern matching, which would match the speed of the gigabit network in case of network intrusion and detection systems, against rather large database of signatures.

The example of exact pattern matching which permits the use of don't care bits was demonstrated in Figure 1d. Note that while Figure 1d shows matching of two 4-bit patterns, (one pattern comprised by the state of flip-flop cells and another one programmed by memristive devices) the number of bits in a pattern that can be compared by one cell is typically much larger - on the order of a few hundred bits for practical values of topological parameter r. Likewise, a flip-flop cell can be configured to perform approximate pattern matching when analog properties of memristive devices are utilized to implement linear threshold gates. In particular, linear threshold gate can implement matching of two patterns based on the Hamming distance between them. The specific threshold for

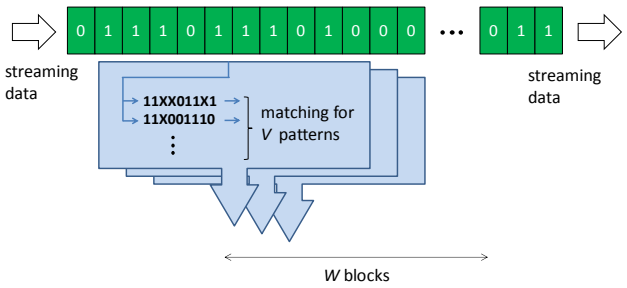

Figure 3. General idea for pattern matching with 1D streaming data.

Hamming distance can be programmed in-field by setting memristive devices to appropriate resistance states [27].

Conceptually, the idea of pattern matching for 1D streaming data is simple (Fig. 3). To improve throughput it is natural to perform multiple pattern matching operations in parallel. Because of fan-out restrictions (i.e. limited connectivity domain) pattern matching is performed simultaneously with several (W) portions of the pipeline data as shown on Figure 3, with V operations done concurrently in each block. With such a scheme, the total number of pattern matching operations performed in a given cycle is $V \times W$ and W cycles are needed to check a certain portion of the streaming data against all ($V \times W$) programmed patterns. Therefore, it is natural to allocate (configure) some flip flop cells in the homogeneous array to perform streaming data by forming long pipelines, while others to implement pattern matching. For simplicity, we assume that the results of pattern matching operations are summed together so that the circuit generates a single bit on the output at every cycle. (A more sophisticated processing would be straightforward given universality of the flip-flop cells and flexibility in mapping.)

Figure 4 shows an example of the mapping where white, green, and blue flip-flops represent cells performing pattern matching, data streaming, and its processing, respectively. More specifically, in this example the streaming data are passed via two independent pipelines formed by green cells, and four exact pattern matching operations are performed with streaming data at each cycle by white cells. The results of pattern matching operations are summed in a pipeline comprised by blue cells.

In general, the largest number of bits in a pattern that can be matched with one cell, denoted by N_{bits}, is M - 1. This would correspond to the case when all cells in the connectivity domain of the given one are configured to stream data. In this case only $\alpha = 1/M$ fraction of the cells are performing the pattern matching operation. At the other extreme case, when all cells in the connectivity domain are configured to perform pattern matching except for one $\alpha = (M\text{-}1)/M$, but in this case $N_{\text{bits}} = 1$. More generally, the dependence of the number of bits in a pattern which can be compared by a cell configured to perform pattern matching is described by $N_{\text{bits}} = (1\text{-}\alpha)M$. Since the largest throughput performance (defined as the total aggregate number of bits in all patterns matched per one cycle) is achieved when the product αN_{bits} has the maximum value it is trivial to show that optimal value of α is 0.5, i.e. when half of the cells in the connectivity domain are configured to perform pattern matching and the other half to stream data. (The cells configured to process the results of pattern matching are neglected in this analysis, which is justified due to their relatively small number, at least in our considered mapping.) A similar observation for balancing streaming and processing resources has been made when mapping network processing tasks on conventional FPGA circuits [30].

Figure 5 shows an example of one such mapping. The streaming 1D data is duplicated and pushed through several pipelines. To ensure that any connectivity domains of white cells consists of always unique and contiguous streaming data from the green cells, the relative position of data in every second pipeline is shifted by five positions (Fig. 5b) for the considered connectivity domain size. (The cells that should be allocated to duplicate the streaming data are not shown though the overhead due to these cells should be negligible.)

Because of the limited size of the connectivity domain, the logical summation from all of the pattern matching cells is done in several steps. As Figures 4 and 5c show, at each cycle a particular blue cell latches the logical summation of the values from the two nearest white cells (which hold the results of pattern matches from the previous cycle), and one blue cell to the left of the given one. The partial sum is fully pipelined and propagates along a row of blue cells at the rate of one cell position per cycle, i.e. as fast as the streaming data. Once partial sums from different rows are propagated to the right edge of the chip, they are summed up in the similar fashion to get one single value. This value represents the logical summation of all results of the comparisons performed with the array at specific time window.

For a given value of $r = 6$, each white flip-flop cell performs $N_{\text{bits}} = 10$-bit wide pattern matching. Also note that with such

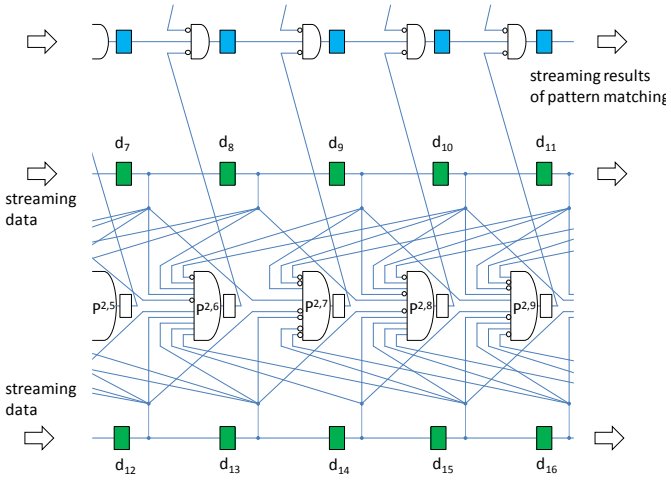

Figure 4. Example of exact matching operations for flip-flop values in green cells, with "0111011111", "1X10000X00", "1011010111", and "1100110110" patterns programmed in memristive devices performed by white cells $p^{2,6}$, $p^{2,7}$, $p^{2,8}$, and $p^{2,9}$, repectively. Note that green cells, are configured into a long pipeline chains to stream data, while the results of pattern matching operations are summed in a pipeline comprised by a chain of blue cells. Here the summation is performed with AND gate based on De Morgan's Law. This figure is a zoom-in of a particular section of Figure 5b.

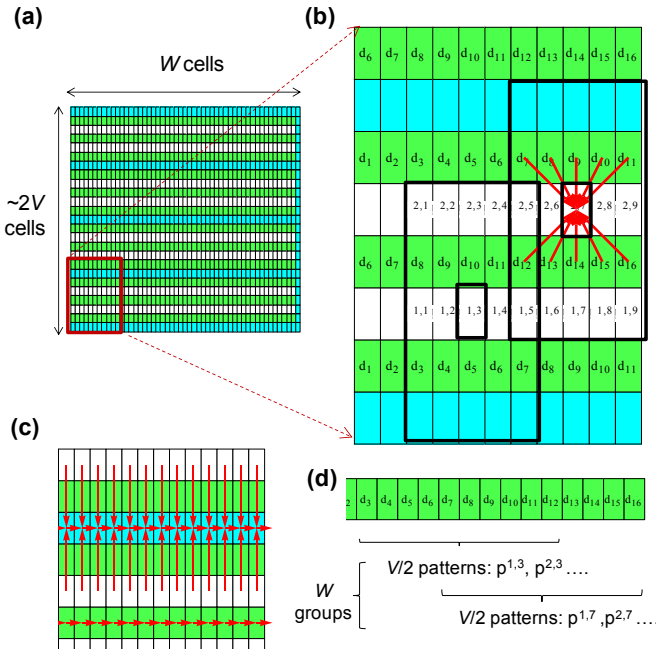

Figure 5. Mapping of pattern matching task to high-throughput CMOL FPGA with r = 6: (a) general mapping scheme; (b) zoom in showing mapping of the streaming data and cells performing pattern matching; (c) the scheme for getting logical summation operation of pattern matching and data propagation in a pipeline; and (d) the relative window for pattern matching operations in the data stream. Red arrows show schematically data movement/logical operation performed by each type of cell. Here the index for d denotes the order of data in pipeline, while pair of indexes for p represents specific pattern index within the block and block id.

mapping, white cells in the same column are performing matching with the same block of data at any given cycle (Fig. 5d). Therefore, the number of pattern matching operations V per block (Fig. 3) is given by the number of white cells in a column, which is roughly equal to half of the total number of cells per column. The number of different blocks W is given by the number of cells in a row, i.e. width of the cell array, so that the total number of patterns which can be matched by an array is roughly half of the total number of cells in the array.

B. *Image Processing*

The main difference for mapping of image processing tasks is that both streaming data and patterns are two dimensional (2D). For example, in automatic target recognition (ATR) systems 128 by 128 array of 8-bit pixels is searched for potential targets which are described by 16 by 16 binary pixel templates [31]. The bottleneck operation in ATR algorithm is 1-bit correlations between input image and template which has to be done for all possible relative offsets and for rather large number of templates. (It should be noted that contemporary ATR systems work with even larger input and template images sizes and potentially higher data rate, e.g. required for hyperspectral image processing [32].)

Naturally, a correlation operation which produces multi-bit output value cannot be done with a single cell in digital CMOL FPGA circuits. On the other hand, a combined operation of correlation with thresholding is straightforward if cell is configured to implement a linear threshold gate. Such an operation might be sufficient for eliminating bottleneck processing in ATR and other image processing tasks in which the image is correlated with template information by performing approximate pattern matching.

Similarly to the previously considered mapping the maximum utilization is achieved with balanced number of white and green cells. Let's assume that connectivity domain is large enough that each cell performs matching between streaming data of input image and the whole template (The proposed scheme can be extended to the case when the domain is smaller, by performing matching operations for the parts of the template instead). Let's also assume that $V \times V$ 2D image is pushed through a pipeline formed by green cells, e.g. from left to right, by one cell position each cycle. To perform correlation for one template for all possible vertical offsets in one cycle requires programming a column of V white cells to perform matching with the same template. Evaluation of all horizontal offsets would just require V cycles to push data (from left to right) past the column of particular white cells. The next column of white cells can be programmed to perform matching for a second template and so on. The result of the pattern matching operations might be summed all together and pushed to the bottom of the array and propagated to the left. For more details see extended version of this paper in Ref. [33].

V. Discussion and Summary

Let us now estimate utilization efficiency of memristive devices by counting the number of memristive devices which participate (i.e. turned to the ON state) in mapping. It is more convenient to count all the devices attached to the wires leading to the flip-flop gate inputs, i.e. quasi vertical wires segments on Figures 1d and 2a).

For both considered examples, roughly half of all cells is utilized to perform pattern matching (white cells) and the other half is providing streaming data (green cells). Therefore, only half of the crosspoint devices attached to the input wire of the pattern matching cells is used to program patterns. (The other half of the devices is always turned off because it is serving connections to cells performing pattern matching). Because each bit of a pattern is represented by two memristive devices (Figs. 1a, d) and one of the two devices is always turned on (if we neglect don't care bits) the number of memristive devices participating in pattern matching is always close to 25%. Because there are 50% pattern matching cells, the memristive device utilization is at least 12.5%.

The high utilization of memristive devices might lead to unprecedented performance for pattern matching tasks. Our earlier estimates (for somewhat aggressive parameters of memristive devices and crossbar wire pitch) indicate that pattern matching throughput for network processing tasks might be more than 4 orders of magnitude higher as compared to the traditional approaches [23, 24]. That preliminary performance estimation was performed taking into account latency increase due to high fan-in gates and power hungry diode logic by enforcing practical power consumption density. The noise margins were not included in the model; however, it should be noted that occasional errors in operation of pattern

matching cells might be acceptable, while other, low fan-in, cells, used for streaming and post processing of the data will be less affected by noise margins.

It is also worth mentioning that the proposed circuits could potentially offer much higher pattern capacity without any performance penalty. Because the number of storage elements in existing hardware-based exact and approximate pattern matchers is limited by the 2D chip area, they must be dynamically reconfigured to accommodate additional patterns that are beyond their storage capabilities. Dynamic reconfiguration is a relatively slow process, and thus the throughput for a fixed area will be considerably smaller than the ideal value. Alternatively, the bit capacity of the high-throughput CMOL FPGA can be increased by integrating more crossbar layers using the multilayer CMOL idea [7]. Thus, this circuit architecture can support more patterns without large penalty in throughput.

More generally, we note that the performance advantages of the proposed circuit architecture are mainly due tight and synergetic integration of memory and logic functionalities. Though CMOL circuits are essential for getting high bandwidth between memory and logic subsystems, other stacking schemes with area distributed connectivity, such as through silicon via technology, and different memory devices, such as flash memory, can be utilized to implement the proposed concept.

ACKNOWLEDGMENT

This work is supported by the Air Force Office of Scientific Research (AFOSR) under the MURI grant FA9550-12-1-0038 and by National Science Foundation grant CCF-1017579.

REFERENCES

[1] D.B. Strukov and H. Kohlstedt, "Resistive switching phenomena in thin films: Materials, devices and applications", MRS Bulletin 37(02), pp. 108-114, February 2012.

[2] W. Lu, D.S. Jeong, M. Kozicki, and R. Waser, "Electrochemical metallization cells – blending nanoionics and nanoelectronics", MRS Bulletin 37(02), pp. 124-130, February 2012.

[3] J.J. Yang, I.H. Inoue, T. Mikolajick, and C.S. Hwang, "Metal oxide memories based on thermochemical and valence change mechanisms", MRS Bulletin 37(02), pp. 131-137, February 2012.

[4] F. Miao et al., "Anatomy of a nanoscale conduction channel reveals the mechanism of a high-performance memristor", Advanced Materials, vol. 23(47), pp. 5633 – 5640, 2011.

[5] B. Govoreanu et al., "10×10 nm^2 Hf/HfO_x crossbar resistive RAM with excellent performance, reliability and low-energy operation", in: Proc. IEDM, pp. 31.6.1-31.6.4, 2011.

[6] K. K. Likharev and D. B. Strukov, "CMOL: Devices, circuits, and architectures," Lecture Notes in Physics, vol. 680, pp. 447-477, 2006.

[7] D. B. Strukov and R. S. Williams, "Four-dimensional address topology for circuits with stacked multilayer crossbar arrays," P Natl Acad Sci USA, vol. 106, pp. 20155-20158, Dec 1 2009.

[8] D. Strukov and A. Mishchenko, "Monolithically stackable hybrid FPGA," in: Proc. DATE'10, Dresden, Germany, Mar. 2010, pp. 661-666.

[9] Q. F. Xia, et al., "Memristor-CMOS hybrid integrated circuits for reconfigurable logic," Nano Lett, vol. 9, pp. 3640-3645, Oct 2009.

[10] G.S. Snider, R.S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect", Nanotechnology, vol. 18, art. 035204, 2007.

[11] J. Cong and B. Xiao, "mrFPGA: A novel FPGA architecture with memristor-based reconfiguration", in: Proc. NanoArch'11, San Diego, CA , June 2011, pp. 1-8.

[12] S. Onkaraiah *et al*., "Using OxRRAM memories for improving communicatios of reconfigurable FPGA architectures", in: Proc. NanoArch'11, San Diego, CA, June 2011, pp. 65-69.

[13] A. Gayasen, N. Vijaykrishnan, M.J. Irwin, "Exploring technology alternatives for nano-scale FPGA interconnects", in: Proc. DAC'05, Anaheim, CA, June 2005, pp. 921-926.

[14] D. B. Strukov and K. K. Likharev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," Nanotechnology, vol. 16, pp. 888-900, Jun 2005.

[15] A. DeHon, "Nanowire-based programmable architectures," J. Emerg. Technol. Comput. Syst., vol. 1, pp. 109-162, 2005.

[16] S.C. Goldstein and M. Budiu, "NanoFabrics: Spatial computing using molecular electronics", in: Proc. ISCA'01, Goteborg, Sweden, June-July 2001, pp. 178-189.

[17] M. Stan et al., "Molecular electronics: From devices and interconnect to circuits and architectures", Proc. IEEE, vol. 91, pp. 1940-1957, 2003.

[18] D.B. Strukov and K.K. Likharev, "All-NDR crossbar circuits", in: Proc. Nano'11, Portland, OR, August 2011, pp. 865 – 868.

[19] A.J. KleinOsowski and D.J. Lilja, "The NanoBox project: Exploring fabrics of self-correcting logic blocks for high defect rate molecular device technologies", in: Proc. VLSI'04, Las Vegas, NV, June 2004, pp. 19-24.

[20] S. Paul and S. Bhunia, "Computing with nanoscale memory: Model and architecture", in: Proc. NanoArch'09, San Francisco, July 2009, pp. 1 – 6.

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

[22] S. Shin, K. Kim, and S.-M. Kang, "Reconfigurable stateful nor gate for large scale logic array integrations", IEEE Trans. Circuits and Systems – II, vol. 58 (7), pp. 442 – 446, 2011.

[23] F. Alibart, T. Sherwood, and D.B. Strukov, "Hybrid CMOS/nanodevice circuits for high throughput pattern matching applications", in: Proc. AHS'11, San Diego, CA, Jun. 2011, pp. 279-286.

[24] D.B. Strukov, "Hybrid CMOS/nanodevice circuits with tightly integrated memory and logic functionality", in: Proc. Nanotech'11, vol. 2, Boston, MA, June 2011, pp. 9-12.

[25] D.B. Strukov and K.K. Likharev, "CMOL FPGA circuits", in: Proc. CDES'06, Las Vegas, NV, June 2006, pp. 213-219.

[26] D.B. Strukov and K.K. Likharev, "Reconfigurable hybrid CMOS/nanodevice circuits for image processing", IEEE Trans. Nanotechnology 6, pp. 696-710, 2007.

[27] L. Gao, F. Alibart, and D.B. Strukov, "Configurable CMOS/memristor threshold gate", accepted to IEEE Trans. Nanotechnology, 2012.

[28] Y. H. Cho and W. H. Mangione-Smith, "Deep network packet filter design for reconfigurable devices," ACM T Embed Comput S, vol. 7, pp. 21:1-21:26, 2008.

[29] S. Hauck and A. DeHon, *Reconfigurable computing: The theory and practice of FPGA-based computation*, Morgan Kaufmann, 2008.

[30] Z. K. Baker and V. K. Prasanna, "Time and area efficient pattern matching on FPGAs," in: Proc. FPGA'04, Monterey, CA, Feb. 2004, pp. 223-232.

[31] J. Villasenor, et al., "Configurable computing solutions for automatic target recognition," in: Proc. FCCM'96, Napa Valley, CA, Apr. 1996, pp. 70-79.

[32] S. M. Chai et al., "Focal-plane processing architectures for real-time hyperspectral image processing," Appl. Optics, vol. 39, pp. 835-849. 2000.

[33] A. Madhavan and D. B. Strukov, "Mapping of image and network processing tasks on high-throughput CMOL FPGA circuits, extended version, available online at https://sites.google.com/site/strukov/, 2012.