Reconfiguration on nanocrossbar using material implication

PRAVIN MANE*, NISHIL TALATI, AMEYA RISWADKAR, RAMESH RAGHU and
C K RAMESHA

Department of Electrical and Electronics Engineering/Electronics and Instrumentation Engineering, BITS
Pilani, K K Birla Goa Campus, Zuarinagar, Goa 403726, India
e-mail: pravinmane@goa.bits-pilani.ac.in

MS received 2 January 2015; revised 1 April 2016; accepted 30 August 2016

Abstract. Reconfigurable architectures (FPGA) with embedded memory elements face problems in retaining data for longer duration due to leakage current as it becomes dominant in nanometer devices in conventional technologies. Also, LUT-based designs consume more area, are slow in processing data and dissipate more power because of complex interconnection network. Memristor, a new nonvolatile memory element, can be used to overcome these limitations. In this paper, we propose an implication-NOR logic-gate-based FPGA architecture using memristors for implementation of logic functions and with embedded memory for storing data. The automation algorithm for the same is presented.

Keywords. Memristor; FPGA; ASIC; CMOL; FPNI; implication.

1. Introduction

Despite being the fastest, the densest and the most power efficient method for implementation of a given function, application-specific integrated circuits (ASICs) lag in terms of their cost and development time. Microprocessor-based approach is highly flexible, ubiquitous and require very short development time but may result in large processor-memory bottlenecks and computing overheads, which limit the system's performance and energy efficiency. Field-programmable gate arrays (FPGAs) combine some of the best properties of microprocessors and ASICs. Like microprocessors, they are highly flexible, cost-effective and require short development time; on the other hand, similar to ASICs, fine-grain customization can be done for FPGA. In applications where massive parallelism is required, FPGAs are the attractive choices. Due to the ineluctable advantages of FPGA over ASIC and microprocessor-based designs, reducing the performance gap between FPGA and ASIC in terms of area, speed and power utilization has become an interesting arena for the researchers [1–6].

In this paper, we exploit the possibility of a memristor-based FPGA architecture. Memristor is the fourth fundamental passive circuit element, theoretically first discovered by Leon Chua in 1971 [7], and came into physical existence in 2008 as a result of the research carried out at HP Labs [8]. It naturally arises in thin-film semiconductors, for which electronic and dopant equations of motion are

coupled in the presence of an applied electric field [8]. It is a two-terminal element whose resistance (also known as memristance) depends on the value and direction of current flown through it. It has an excellent capacity to preserve its state even when unpowered. Thus it can be used to replace SRAM cells in FPGA.

To combine the advantages of CMOS technology, including its high flexibility, functionality and yield, with the extremely high density of nanowires, nanodevices and interface hybrid CMOS–Memristor circuits have been built (figure 1). They combine a CMOS subsystem with monolithically integrated several layers of nanowire crossbars, consisting of arrays of two-terminal memristors, all connected by an area-distributed interface between the CMOS subsystem and the crossbars [9]. The most straightforward application of described hybrid circuits is embedded and stand-alone memories with their simple matrix structure. In the proposed architecture, memristors will be used to store single bit of data as well as for performing logic operation and CMOS is used for coding, decoding, line driving, sensing and input/output functions as in Cuniberti *et al* [1] and Strukov and Likharev [10].

These hybrid circuits are compatible in terms of the materials and processing technologies employed to fabricate them [11]. Manifold nanocrossbar layers are stacked onto a CMOS layer as shown in figure 1 in order to obtain superior scaling potential (10 GB/cm^2), high yield (99%), high on/off ratio (10^3), high endurance (10^6), large programming speed (5 ns) and retention time (5 months) [11], stackability [12], scalability, etc.

*For correspondence

Figure 1. CMOS-memristor hybrid circuits [13].

2. Background

2.1 CMOL and FPNI architectures

Hybrid CMOS/molecular (CMOL) architecture of Strukov and Likharev is one of the probable ways to extend Moor's Law beyond the limits of transistor scaling. Density, simplicity and clean separation of configuration and data communication are some of the appreciable merits of CMOL architectures [14]. Field-programmable nanowire interconnect (FPNI) is another generalized hybrid architecture, which trades off some of the speed, density and defect tolerance of CMOL in exchange for easier fabrication, lower power dissipation and greater freedom in the section of nanodevices in the crossbar junctions (figure 2). In FPNI architecture, logic is performed at CMOS layer and routing operation is implemented through the nanowires. This reduces the static power dissipation significantly, and allows us to use

linear antifuses for the nanowire junctions [4] (<http://public.itrs.net/>).

2.2 Memristor: models and its resistance switching

A current-controlled time-invariant memristive system is represented by

$$\frac{dw}{dt} = f(w, i), \quad (1)$$

$$v(t) = R(w, t) i(t), \quad (2)$$

where w is an internal state variable, $i(t)$ is the memristive device current, $v(t)$ is the memristive device voltage, $R(w, i)$ is the memristance and t is time [16]. Based on the various mathematical expressions available for different variables, there are four major memristor models. They are

Figure 3. Memristor characteristics evaluation circuit.

Figure 2. Fabricated FPNI architecture with a nearly defect-free region (left) and highly defective region (right) in a 34 nm pitch nanowire crossbar [15].

Figure 4. Resistance variation of a memristor when the current across parallel combination of memristor and resistance is varied from -2 to $2 \mu\text{A}$.

Table 1. Imply gate truth table.

P	Q	$P \rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

(i) linear ion drift model, (ii) non-linear ion drift model, (iii) Simmons tunnel barrier model and (iv) threshold-adaptive memristor (TEAM) model.

TEAM is the model with simple expressions as well as an adequate accuracy. The state variable derivative and current–voltage relationship in this model can be given as below:

$$\frac{dx}{dt} = \begin{cases} k_{\text{off}} \left(\frac{i(t)}{i_{\text{off}}} - 1 \right)^{\alpha_{\text{off}}} f_{\text{off}}(x), & 0 < i_{\text{off}} < i \\ 0, & i_{\text{on}} < i < i_{\text{off}} \\ k_{\text{on}} \left(\frac{i(t)}{i_{\text{on}}} - 1 \right)^{\alpha_{\text{on}}} f_{\text{on}}(x), & i < i_{\text{on}} < 0 \end{cases} \quad (3)$$

$$v(t) = \left[R_{\text{ON}} + \frac{R_{\text{OFF}} - R_{\text{ON}}}{x_{\text{off}} - x_{\text{on}}} (x - x_{\text{on}}) \right] \cdot i(t)$$

where k_{off} , k_{on} , α_{off} and α_{on} are constants; i_{off} and i_{on} are current thresholds, and internal state variables, which represent the effective electric tunnel width [16]. The functions $f_{\text{off}}(x)$ and $f_{\text{on}}(x)$ represent the dependence on the state variable x . R_{off} and R_{on} represent the off and on state

Figure 5. Imply gate using memristor.

Figure 6. Implication-based NOR logic block.

resistances of the memristor, respectively. To simulate the proposed design, we have used the TEAM model along with Kvatinsky window.

Memristor shows pinched hysteresis characteristics, a property of memory elements. To study its switching behaviour, a resistance and a memristor are connected in a fashion shown in figure 3. The threshold limits are set to $I_{\text{off}} = -1 \text{ nA}$ and $I_{\text{on}} = 1 \text{ nA}$ in the model. The input current (i) figure 3 is changed from -2 to $2 \mu\text{A}$. The resulting current and resistance of the memristor are plotted in (figure 4). It is observed that when the current through the memristor is above I_{on} (i.e., greater than 1 nA in this case), it switches to on state (low resistance R_{on}) and when it is less than I_{off} (i.e., lower than -1 nA in this case), it switches to

Figure 7. Symbolic representation of NOR logic block using implication logic.

Figure 8. Evaluation logic block circuit.

off state (high resistance R_{off}). When current is between I_{off} and I_{on} it retains its previous state or resistance.

2.3 Switching equations for memristors

Most of the applications involving memristors concern with its switching speed. The transition delays of a memristor are given by [17]

$$t_{on} = \frac{C_0^2}{2\gamma_i \beta V \lambda} \quad (4)$$

where t_{on} is time for R_{off} to R_{on} transition and

Figure 9. Write logic block Circuit.

Figure 10. Read logic block circuit.

$$t_{off} = \frac{\left[\left(1 - \frac{n_c}{n_i} \right) D + 2C_0 \right] \left(1 - \frac{n_c}{n_i} \right) D}{2\gamma_i \beta \lambda V} \quad (5)$$

where t_{off} is time for R_{on} to R_{off} transition, V is voltage across memristor,

$$C_0 = \left(\frac{\lambda}{2} \right) + \left(\frac{n_c}{n_i} \right) \left(D - \frac{\lambda}{2} \right), \quad (6)$$

λ is transition/conduction/insulation region length, n_c , n_t and n_i are conduction, transition and insulation region

Figure 11. The three-input logic block (LB3) architecture using implication-NOR operation.

concentrations, respectively, D is the thickness of insulating material (TiO_2) in the memristor and, γ_t is electron generating coefficient in the transition region:

$$\frac{d}{dt}n_t = \gamma_t(n_c - n_i)E_t(t) \quad (7)$$

Here, $E_t(t)$ is the electric field in the transition region and

$$\beta = \frac{1}{4} \left(\frac{n_c}{n_i} - 1 \right)^3 + \frac{2}{3} \left(\frac{n_c}{n_i} - 1 \right)^2 + \frac{1}{2} \left(\frac{n_c}{n_i} - 1 \right). \quad (8)$$

The voltage across memristor in the write cycle is fixed. However, the voltage in the evaluate cycle varies with the number of inputs to the block as given below:

$$V = V_{SET} - V_{COND} \left(\frac{R_G}{R_G + \left(\frac{R_{off}}{n-k} \parallel \frac{R_{on}}{k} \right)} \right) \quad (9)$$

where k is the number of memristors in on state and n is the maximum number of inputs.

When $k = 0$, $V = V_{max}$. Therefore minimum switching times for memristor are given by

$$t_{onmin} = \frac{C_0^2}{\gamma_t \beta V_{max} \lambda}. \quad (10)$$

Figure 12. Timing diagram of logic architecture.

and

$$t_{\text{off}_{\min}} = \frac{\left[\left(1 - \frac{n_c}{n_i} \right) D + 2C_0 \right] \left(1 - \frac{n_c}{n_i} \right) D}{2\gamma_t \beta \lambda V_{\max}}. \quad (11)$$

2.4 Material implication

Implication logic is the most natural operation implemented using a memristor. The truth table and the circuit diagram of this logic are given in table 1 and figure 5, respectively.

3. Generalized architecture

In this paper, we propose a new generalized architecture for implementing any logic function. It is based on the material implication operation performed using memristors. NOR gate can be realized with material implication, which is a universal gate and is the smallest block of proposed architecture.

3.1 Implication-based NOR block

In the previous section, imply logic has been discussed. One key point to note here is the required inequality between the applied voltages (i.e. $V_{\text{SET}} > V_{\text{COND}}$) for

Figure 13. Symbolic representation of three-input logic block.

evaluation. Thus, the only transition possible for the memristor Q is from off state to on state. This observation has been utilized to build the NOR gate.

Figure 6 represents the circuitry for n -input NOR gate using material implication by memristors. There are n -Input memristors and 1-evaluation memristor. Three major steps are involved in order to evaluate NOR operation using this logic. They are

1. *Write phase*: $[(M_1, M_2, \dots, M_n) \leftarrow \text{data}], (M_{\text{Eval}} \leftarrow \text{clear}), (\text{WRITE} = 1, \text{EVAL} = \text{READ} = 0)$.
2. *Evaluation phase*: $[V(M_1, M_2, \dots, M_n) \leftarrow V_{\text{COND}}], [V(M_{\text{Eval}}) \leftarrow V_{\text{SET}}], (\text{EVAL} = 1, \text{WRITE} = \text{READ} = 0)$.
3. *Read phase*: $(M_{\text{Eval}} \leftarrow V_{\text{Read}}), (\text{READ} = 1, \text{WRITE} = \text{EVAL} = 0)$.

After executing these operations correctly, the result of NOR operation for the n -input assembly is achieved on the evaluation memristor. A symbolic representation of the same block with all the control signals required is shown in figure 7.

Here WRC, Eval and RDC are the control signals given to enable the Write, Evaluate and Read circuitry, respectively, in corresponding phases. Data represent the bus containing the inputs. V_{COND} and V_{SET} are appropriate voltages to be applied across the memristors. For a fixed value of R_G , there is a constraint for the fan-in of this gate, which is

$$n \leq \frac{R_{\text{on}}}{\frac{V_{\text{COND}} R_G}{V_{\text{SET}} - i_{\text{on}} R_{\text{off}}} - R_G}. \quad (12)$$

3.2 Functional blocks in CMOS layer

To implement logic function in nanocrossbar layer, the following necessary blocks are involved in CMOS layer.

Figure 14. Architecture to implement any logic function without constraints in terms of fan-in.

1. *Evaluation block*: Figure 8 shows the circuit required to evaluate the imply logic. Eval is the control signal to enable this block. The input memristor should be connected in parallel between T_{IM1} and T_{IM2} . The evaluation memristor should be connected between T_{EM1} and T_{EM2} . This parallel combination will be connected to the ground through R_G . V_{COND} and V_{SET} are applied across the input memristors and the evaluation memristor, respectively.
2. *Write block*: Figure 9 shows the Write block internal circuit required to apply appropriate voltage across the memristors in order to write inputs to the input memristors and clear the evaluation memristor. WRC is the write control signal to enable this block. Data are the data signal, which can write either logic '0' or '1' to the memristors. Here, the memristor should be connected between T_1 and T_2 in an appropriate orientation.
3. *Read block*: Figure 10 shows the Read block circuit. The output of the implication logic is in terms of the

resistance of the memristor. To use it further, converting it to voltage is a must. This is the sole purpose of Read block. The evaluation memristor is connected between T_1 and T_2 . Appropriate voltage V_{READ} is applied to it. If the memristor is in on (off) state (i.e. R_{on} (R_{off})), V_{Result} will be logic '1' (logic '0').

4. *Priority block*: It is used to select logic '1' input with maximum priority from the outputs of NOR blocks.

3.3 Comprehensive description of the architecture

Figure 11 shows the detailed architecture to implement a three input function. The emblematic representation is shown in figure 13. A, B, and C are the data lines (inputs); EN is the enable control signal to enable this particular logic block; LEN is the level enable signal and WRC, EV and RDC are the write, evaluate and read control signals, respectively.

LEVEL1 consists of seven NOR blocks and a seven-input (output) priority circuit. For n -input operation, number of NOR blocks and the inputs (outputs) to (from) the priority block will be $2^n - 1$. Further, total number of memristors in LEVEL1 will be $n2^{n-1} + 2^n - 1$. Outputs of each NOR block are connected as the inputs to the priority circuit. LEVEL2 consists of m NOR blocks (where m represents the number of outputs). Each NOR block corresponds to one logical output. The number of inputs to these NOR blocks is equal to the number of input combinations having outputs different from all inputs being logic high (logic '1'). The timing diagram for architecture operation is given in figure 12. Various control signals (WR, EVAL, RD) and read/write data in LEVEL1/LEVEL2 are shown in the timing diagram along with results to be written in destination memristor.

Since there is a restriction on fan-in of NOR block, implementing the logic function having large number of inputs may give faulty result(s). In that case, the logic function should be divided into multiple parts depending upon constraint on fan-in and then small result blocks can be multiplexed to implement larger function. The sole purpose of EN signal here is to enable any b -input NOR blocks ($b \leq$ fan-in of NOR gate) in such situation. Thus, many such blocks can be used in cascade to produce the

correct output. Thus, the architecture containing any number of inputs is shown in figure 14.

4. Evaluating the performance of the architecture

This section deals with analysing the above architecture with respect to its nanodevices used and CMOS overhead. A more recent approach, CMOL (CMOS/molecular hybrid), proposes moving the most difficult functions necessary for logic inversion and gain-along with the demultiplexing into the CMOS layer, using the nanowires and selected configured junctions only for wired-OR logic and signal routing. We will henceforth be comparing the performance of our architecture with the CMOL FPGA architecture considering some benchmark circuits given by UCB.

4.1 The CMOS overhead

The dominant CMOS overhead comes from the decoding and multiplexing the outputs of our logic blocks.

1. *Decoder*: There are $n - 3$ inputs to the decoder. Each output has a NAND gate associated with it. Also, each

Table 2. Conventional LUT-based CLB and proposed implication-NOR-gate-based logic block comparison.

Parameters	Three-input LUT-based CLB	Three-input logic block using implication-NOR gate
Memristor count	—	33
Transistor count	342	188
Delay (ns)	3.8	2.43
Power dissipation (mW)	7.8	2.15

Table 3. Comparison of the architecture in terms of device count for benchmark circuits.

Benchmark circuit	CMOL		Proposed architecture		Conventional FPGA transistors
	Transistors	Memristors	Transistors	Memristors	
ALU4	22459	8157	3169	91	880
APEX2	21916	9907	1484	46	330
APEX4	4178	6257	6876	190	2090
BIGKEY	6339	8835	64840	1738	21010
CLMA	383123	70403	10583	289	3300
EXP5P	3043	4885	21704	586	6930
EX1010	63879	33298	3843	109	1100
MISEX	15061	7598	5191	145	1540
PDC	12336	35552	6140	379	4400
S298	52088	13321	2495	73	660
SEQ	24499	8897	12268	334	3850
SPLA	132792	29251	15975	433	5060

- input has a NOT gate associated with it. The number of transistors in the decoder are $2 \cdot (n - 3) + 4 \cdot (2^{n-3})$.
2. *Multiplexer*: This MUX is assumed to be made up of transmission gates since we already have the decoded enable signals available. It outputs two control signals corresponding to each destination memristor: the write signal to one of the final write blocks and the default value to be written to the destination memristors. Hence, $8(2^{n-3})$ transistors will be used here.
 3. *Logic block*: The CLB uses numerous function-specific as well as other units in the CMOS layer. It consists of circuits that write to a memristor, perform the imply operation on the memristor and read the resistance of the same. In addition, it consists of transmission gates that control the propagation of certain signals and act as bridges to the nanodevices. An estimated number of transistors that shape this block is $473 + 337m$.

required to store the input voltage and compute the NOR operation onto another memristor. The second level is also made of NOR blocks. It operates on the relevant minterms and gives the final output required. This output is stored in the destination memristor in the final step. As a result, there are $19 + 9O$ net memristors used. 'O' indicates the number of outputs of logic function. For example, 1-bit full adder has 2 outputs, namely sum and carryout; hence value of 'O' for 1-bit full adder is 2.

A comparison of conventional lookup table (LUT)-based three-input configuration logic block (CLB) and three-input implication NOR-based architecture in terms of device count, delay and power dissipation is given in table 2 while device count comparison for benchmark circuits is given in table 3.

4.2 Nanodevices used

Memristors are used on three levels in this architecture. The first level is made of NOR blocks. Memristors are

4.3 Automation algorithm

The automation algorithm for proposed architecture is shown in Algorithm-1.

Algorithm 1: Automation Algorithm

Input: Array $I(n, m)$ where $n:1$ to 2^y , y is the number of inputs; $m:1$ to z , z is the number of outputs and $I(n, m)$ are the minterm; x -input, z -output logic BLOCK; $y - x$ input decoder

Output: 'netlist'-like connection pattern

```

1 for  $n \leftarrow 1$  to  $y - x$  do
2   |  $DecoderI(i) \leftarrow Input(x + i)$ 
3 end
4 for  $j \leftarrow 1$  to  $2^{y-x}$  do
5   |  $logicBLOCK(j)(en) \leftarrow DecoderO(j)$ 
6 end
7 logic BLOCK(1 to  $2^{y-x}$ )(data(1 to  $x$ ))  $\leftarrow$  Input(1 to  $x$ )
8  $Mux1(1$  to  $z)I(1$  to  $2^{y-x}) \leftarrow$  logic BLOCK(1 to  $2^{y-x}$ )( $Wr(1$  to  $z)$ )
9  $Mux1(1$  to  $z)(En)(1$  to  $2^{y-x}) \leftarrow$  DecoderO(1 to  $2^{y-x}$ )
10  $Mux2(1$  to  $z)I(1$  to  $2^{y-x}) \leftarrow$  logic BLOCK(1 to  $2^{y-x}$ )( $Defaultvalue(1$  to  $z)$ )
11  $Mux2(1$  to  $z)(En)(1$  to  $2^{y-x}) \leftarrow$  DecoderO(1 to  $2^{y-x}$ )
12 Destination Write BLOCK( $Wr(1$  to  $z)$ )  $\leftarrow$   $Mux1(1$  to  $z)O(1$  to  $2^{y-x})$ 
13 Destination Write BLOCK( $Defaultvalue(1$  to  $z)$ )  $\leftarrow$ 
     $Mux2(1$  to  $z)O(1$  to  $2^{y-x})$ 
14 Destination memristor( $m$ )( $HIGH, LOW$ )  $\leftarrow$  Destination Write BLOCK $O(m)(HIGH, LOW)$ 
15 return connection pattern
    1.  $DecoderI(i)(1$  to  $y - x) \leftarrow Input(x$  to  $y)$   $\triangleright$  connection pattern between inputs and Decoder inputs
    2.  $logic BLOCK(1$  to  $2^{y-x})(en) \leftarrow DecoderO(i)(1$  to  $2^{y-x})$   $\triangleright$  connection pattern between outputs of Decoder and logic block inputs
    3.  $logic BLOCK(1$  to  $2^{y-x})(data) \leftarrow Input(1$  to  $x)$   $\triangleright$  connection pattern between input lines and logic BLOCK inputs
    4.  $Destination memristor(1$  to  $z)(HIGH, LOW) \leftarrow logic BLOCK(1$  to  $2^{y-x})O(1$  to  $z)(HIGH, LOW)$   $\triangleright$  connection pattern between outputs of logic BLOCKS and destination memristors

```

Figure 15. Simulation results for inputs = 000, Outputs: Sum = 0 (200 kΩ), and Carry = 0 (200 kΩ).

Figure 17. Simulation results for inputs = 011. Outputs: Sum = 0 (200 kΩ) and Carry = 1(100 Ω).

Figure 16. Simulation results for inputs = 001, Outputs: Sum = 1 (100 Ω), and Carry = 0 (200 kΩ).

Figure 18. simulation results for inputs = 111. Outputs: Sum = 1 (100 Ω) and and Carry = 1 (100 Ω).

5. Simulation results

One-bit full adder using the implication-NOR-based architecture is simulated. The results are shown in figures 15–18. The simulation results of one-bit pipelined full

adder along with the input voltage waveform used are shown in figures 19 and 20. The simulation is carried out using Virtuoso tool of Cadence and spectre simulator. TEAM model written in Verilog A is used for memristor while 65 nm CMOS technology is used for CMOS part of the circuit.

Figure 19. Inputs applied to one-bit pipelined full adder.

Figure 20. Output of one-bit pipelined full adder for different inputs corresponding to figure 19 : output (t_1) (Sum = 0, Carry = 0) for inputs ABC = 000, output (t_2) (Sum = 1, Carry = 0) for inputs ABC = 010, output (t_3) (Sum = 1, Carry = 1) for inputs ABC = 111, Output and (t_4) (Sum = 0, Carry = 1) for inputs ABC = 110.

6. Conclusion

In this paper, we propose 3D hybrid CMOS–memristor architecture using implication-NOR gate to implement logic functions having a number of inputs well within fan-in limits (which depend on the technology employed for fabrication). To remove the restrictions on fan-in, we also

propose a method to cascade these logic blocks. The memristor and transistor count for benchmark circuit is also presented, which shows considerable improvement in device utilization. Most of the circuit components in CMOS layer are reusable and are repeated. 3D architecture will improve the area utilization. The comparison of three-input reconfigurable architecture with three-input LUT-based

CLB has shown improvement in terms of delay, area and power consumption.

References

- [1] Cuniberti G, Fagas G and Richter K (Eds.) 2005 Introducing molecular electronics, 1st edn. Springer Publishing Company, Incorporated
- [2] Kim K, Shin S and Kang S-M S 2011 Field programmable stateful logic array. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 30(12): 1800–1813. doi:10.1109/TCAD.2011.2165067
- [3] Kuon I and Rose J 2009 Quantifying and exploring the gap between FPGAs and ASICs, 1st edn. Springer Publishing Company, Incorporated, ISBN: 1441907386, 9781441907387
- [4] Snider G S and Williams R S 2007 Nano/CMOS architectures using a field-programmable nanowire interconnect. *Nanotechnology* 18(3): 035204. <http://stacks.iop.org/0957-4484/18/i=3/a=035204>
- [5] Strukov D B, Stewart D R, Borghetti J, Li X, Pickett M, Ribeiro G M, Robinett W, Snider G, Strachan J P, Wu W, Xia Q, Yang J J and Williams R S 2010 Hybrid CMOS/memristor circuits. In: *Proceedings of 2010 IEEE international symposium on circuits and systems (ISCAS)*, pp. 1967–1970 doi:10.1109/ISCAS.2010.5537020
- [6] Strukov D B and Likharev K K 2005 CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. *Nanotechnology* 16(6): 888–890. <http://stacks.iop.org/0957-4484/16/i=6/a=045>
- [7] Chua L O 1971 Memristor—the missing circuit element. *IEEE Trans. Circuit Theory* 18(5): 507–519. doi:10.1109/TCT.1971.1083337
- [8] Strukov D B, Snider G S, Stewart D R and Williams R S 2008 Memristor—the missing circuit element. *Nature* 453: 80–83
- [9] Strukov D B 2012 3D hybrid CMOS/memristor circuits: basic principle and prospective applications. In: *Proceedings of the 2012, Conference on Optoelectronic and Microelectronic Materials Devices (COMMAD)*, pp. 21–22. doi:10.1109/COMMAD.2012.6472340
- [10] Strukov D B and Likharev K K 2007 Defect-tolerant architectures for nanoelectronic crossbar memories. *J. Nanosci. Nanotechnol.* 7(1):151–167
- [11] Jo S H and Lu W 2008 CMOS compatible nanoscale non-volatile resistance switching memory. *Nano Lett.* 8(2): 392–397. doi:10.1021/nl073225h PMID: 18217785
- [12] Yang J J, Zhang M-X, Strachan J P, Miao F, Pickett M D, Kelley RD, Medeiros-Ribeiro G and Williams R S 2010 High switching endurance in TaO_x memristive devices. *Appl. Phys. Lett.* 97(23). doi:10.1063/1.3524521. <http://scitation.aip.org/content/aip/journal/apl/97/23/10.1063/1.3524521>
- [13] Xia Q, Robinett W, Cumbie M W, Banerjee N, Cardinali T J, Yang J J, Wu W, Li X, Tong W M, Strukov D B, Snider G S, Medeiros-Ribeiro G and Williams R S 2009 Memristor/CMOS hybrid integrated circuits for reconfigurable logic. *Nano Lett.* 9(10): 3640–3645 doi:10.1021/nl901874j. PMID: 19722537
- [14] Strukov D B and Likharev K K 2006 A reconfigurable architecture for hybrid CMOS/nanodevice circuits. In: *Proceedings of the 2006 ACM/SIGDA 14th international symposium on field programmable gate arrays*. FPGA '06, ACM, New York, NY, USA, pp. 131–140 ISBN1-59593-292-5. doi:10.1145/1117201.1117221
- [15] Jo S H, Chang T, Ebong I, Bhadviya B B, Mazumder P and Lu W 2010 Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10(4): 1297–1301. PMID: 20192230. doi:10.1021/nl904092h
- [16] Kvatinisky S, Friedman E G, Kolodny A and Weiser U C 2013 TEAM: ThrEshold adaptive memristor model. *IEEE Trans. Circuits I Regular Papers* 60(1): 211–221. doi:10.1109/TCSI.2012.2215714
- [17] Zhang L, Chen Z, Joshua Yang J, Wysocki B, McDonald N and Chen Y 2013 A compact modeling of TiO₂-TiO_{2x} memristor. *Appl. Phys. Lett.* 102(15). doi:10.1063/1.4802206. <http://scitation.aip.org/content/aip/journal/apl/102/15/10.1063/1.4802206>