Memristive devices: Technology, Design Automation and Computing Frontiers

Mario Barbareschi, Alberto Bosio, Hoang Anh Du Nguyen, Said Hamdioui, Marcello Traiola, Elena Ioana Vatajelu

2017 12th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)

Memristive devices: Technology, Design Automation and Computing Frontiers

Mario Barbareschi[1], Alberto Bosio[2], Hoang Anh Du Nguyen[4], Said Hamdioui[4], Marcello Traiola[2], Elena Ioana Vatajelu[3]
[1]DIETI, University of Naples Federico II – Naples, Italy
[2]LIRMM CNRS/UM – Montpellier, France
[3]TIMA Grenoble Alpes University – Grenoble, France
[4]Delft University of Technology – Delft, The Netherlands

***Abstract*— The memristor is an emerging technology which is triggering intense interdisciplinary activity. It has the potential of providing many benefits, such as energy efficiency, density, reconfigurability, nonvolatile memory, novel computational structures and approaches, massive parallelism, etc. These characteristics may lead to deeply revise existing computing and storage paradigms. This paper presents a comprehensive overview of memristor technology and its potential to design a new computational paradigm.**

Keywords-component; emerging technologies; computer architecture; memory; CAD tool; logic synthesis.

I. INTRODUCTION

Today's computing devices are based on the CMOS technology, that is the subject of the famous Moore's Law [1], predicting that the number of transistors in an integrated circuit will be doubled every two years. Despite the advantages of the technology shrinking, we are facing the physical limits of CMOS. Among the multiple challenges arising from technology nodes lower than 20 nm, we can highlight the high leakage current (i.e., high static power consumption), reduced performance gain, reduced reliability, complex manufacturing process leading to low yield, complex testing process, and extremely costly masks [2][3][4][5].

Additionally, the expected never-ending increasing of performances is indeed no longer true. Looking in more detail, the classical computer architectures, either Von Neumann or Harvard, divide the computational unit (i.e., CPU) from the storage element (i.e., memory). Therefore, data have to be transferred inside the computational element in order to be processed and then transferred back to be stored. The main problem of this paradigm is the bottleneck due to the data transfer time limited by the bandwidth. For instance, transferring one TeraByte at the rate of 1Gbit/second requires more than two hours.

Many new technologies are under investigation, among them the memristor is a promising one [6]. Indeed, the memristor is a non-volatile device able to act as both storage and information processing unit that presents many advantages:

CMOS process compatibility, lower cost, zero standby power, nanosecond switching speed, great scalability, high density and non-volatile capability [7][8]. Thanks to its nature (i.e., computational as well as storage element), the memristor is exploited in different kind of applications, such as neuro-morphic systems [9], non-volatile memories [10], computing architecture for data-intensive applications [11].

This paper presents a comprehensive overview of memristor technology and its potential to design a new computational paradigm. The remainder of the paper is structured as following. Section II presents the basic background about the memristor and its potential. Section III discusses the design flow of memristor-based computing devices by presenting a synthesis flow and the design exploration framework. Finally, Section 0 discusses the real impact of memristor-based computing devices.

II. MEMRISTIVE DEVICES AND THEIR POTENTIAL

The continuous technology scaling, as well as the emergence of new technologies, favor increasing of the system complexity and performance, opening the scientific community to exotic applications and computation paradigms which had been unfeasible a few years back due to technological limitations of the hardware. The emergence of new low power, highly scalable, CMOS compatible memory devices (such as memristive devices) is tries to address the technical constraints of today's memories.

The memristive devices have great characteristics in terms of area, power and speed when used as memory or data storage devices, but, in addition, they are promising solutions for logic implementation. Thanks to the relative easiness of massive parallelism, computing in memristive memory becomes trending topic in current research activities. Moreover, other fields, such as brain-inspired computing, benefit from the unique features of this technology. In continuation, a comprehensive overview of the memristive technology landscape with special emphasis on the most desirable features and dire shortcomings in memory and logic design is presented.

A. Working Principle and Classification

The memristor is a semiconductor device whose resistance is called memristance. The memristance is a charge dependent resistance and its value varies as a function of current and flux. The memristor technology has some great advantages such as

data non-volatility, CMOS compatibility, low switching power, no leakage power, high integration capability.

Memristors can be classified in two different types: (i) ionic thin film and molecular memristors, and (ii) magnetic and spin-based memristors. In this work we are focusing on the first category, since the second has developed independently as spintronic devices.

When used as a memory device, the first category of memristors is called resistive memory, more precisely Resistive Random Access Memory (RRAM) and it can act as a non-volatile memory. Its data storage element is a three-layer device, consisting of a dielectric sandwiched between two metal electrodes. There are many materials which can be used for the electrodes and dielectric, but the underling operation principle remains the same. The RRAM device switches between two resistive states, i.e., the high resistance state (HRS) and the low resistance state (LRS), when triggered by an electrical input.

RRAM relies on the formation (corresponding to low resistance) and the rupture (corresponding to high resistance) of conductive paths in the dielectric layer. Once the conduction path is formed, it may be RESET (the path broken, transition from LRS to HRS) or SET (the path re-formed, transition from HRS to LRS). Usually, right after fabrication (i.e., the pristine samples) the devices have a very high electrical resistance (~1GΩ) and a large voltage is required for the first SET operation, also known as the forming process; this drastically reduces the device resistance (to about 10KΩ) triggering the switching behavior in the subsequent cycles.

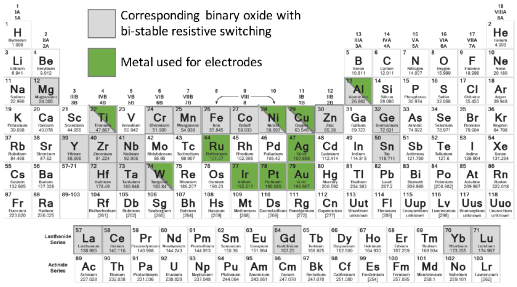
Classification: The memristive devices (RRAM) can be classified following different criteria. They can be classified according to the used materials, the switching mechanism, the conductive path, and the switching mode.

The different types of resistive devices according to different classifications are illustrated in Fig. 1. In Fig. 1a. the materials suitable for fabricating a resistive element are underlined, in Fig. 1b the types of conductive path formation are illustrated while in Fig. 1c the switching modes are sketched.

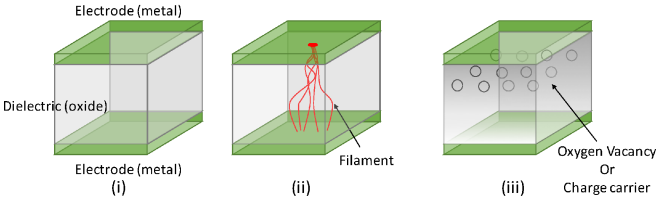
There are two possible RRAM *switching modes*: unipolar switching, which depends only on the amplitude of the applied voltage and not its polarity, i.e., the SET and the RESET operations are controlled by the same polarity; and bipolar switching in which the SET and the RESET operations are controlled by reverse polarities.

Depending on the *dominant physical switching mechanism*, the resistive devices can be classified in: Phase Change Memories, Electrostatic/ Electronic Effects Memories, and Redox Memories. Various resistive switching mechanisms have been proposed to efficiently perform the SET and RESET operations. They include the formation and rupture of conductive paths, charge trapping, electrode-limited conduction [12], [13]. The low-resistance *conductive path* can be either localized (filamentary) or homogeneous.

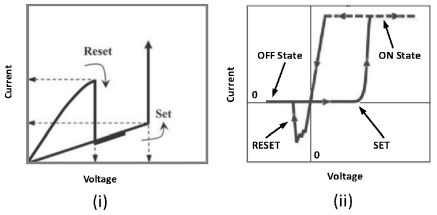
One of the most versatile resistive memories is the Redox RAM [1], [14], where the RESET and SET processes, breakdown and regrow of the conductive filaments, involve oxidation and reduction (i.e., redox reaction). These are Metal-Insulator-Metal (MIM) structures, in which the switching mechanism is electrochemical and it can occur in the insulator-layer, or at the insulator-layer/metal contact interfaces.



a) Overview of Memristive-compliant materials – adaptation after S.Yu, et al. "Metal oxide resistive switching memory," 2011 [41]



b) Resistive element classification according to the conductive path: (i) pristine sample, (ii) filamentary path, (iii) homogenous path



c) Resistive element classification according to the switching modes: (i) unipolar switching, (ii) bipolar switching

Figure 1. Resistive device classification

The MIM structures can be classified by their underlying switching mechanism as follows [15]:

The Valence Change Mechanism (VCM): here the dielectric layer can act as an electrolyte and the migration of oxygen vacancies within the applied electric field evolves in a bipolar manner. The conductive path is formed due to the oxygen anions (positively charged oxygen vacancies), while the electric current is defined by the electrostatic barrier in the band diagram. Appling negative bias voltage on the electrodes of the memristor the SET operation is performed due to a local redox reaction which increases the device conductivity. The RESET operation is performed by reversing the bias polarity and allowing the recombination of oxygen. The most common examples of VCM RRAMs use TaOx, HfOx and TiOx [16], [17] devices.

The Electrochemical Mechanism (ECM): uses an electrochemically active electrode metal such as Ag or Cu. The mobile metal cations drift in the ion conducting layer and discharge at the counter-electrode, leading to a growth of conductive metallic filaments in the isolation layer - i.e., the SET mechanism. The RESET mechanism is performed by

reversing the polarity of the applied voltage, resulting in the electrochemical dissolution of the conductive filaments [18].

The Thermochemical Mechanism (TCM): relies on a filament modification due to Joule heating. Conductive filaments, composed of the electrode metal transported into the insulator, are formed during the forming process prior to memory cyclic switching. The SET operation is achieved by Joule heating; it triggers local redox reactions that facilitate the formation of oxygen deficient ions and metallic filaments. The RESET operation is a thermally activated process resulting in a local decrease of the metallic species. TCMs are unipolar switching devices. NiO has emerged as the reference material for resistive switching based on the TCM [19].

As a reasonably representative example, in the subsequent sections, the focus will be on HfOx-based VCM RRAMs, as they seem the most promising. Note that the focus of the paper is on device test where the quality of the conductive path formation is relevant, regardless of the physical mechanism.

B. Opportunities and Challenges

Opportunities: Memristive devices are on the way to change the classical memory/storage architectures. They should meet the high demands of tomorrow applications, like high performance and high density, good endurance, small devices sizes, good integration, low power profile, resistance to radiation, and ability to scale below 20 nm [20], [21].

The most investigated use of the memristor is memory since a it can store data. When compared with traditional memories, such as SRAM or DRAM, this kind of memory has many benefits, such as, no leakage power, non-volatility and scalability, being in the same time superior to flash memory in terms of speed and scalability. In addition, the memristive device can be used in logic circuits, either as standalone logic gates, or used in hybrid CMOS-memristor circuits. Memristors can be used to do digital logic using implication instead of NAND.

The simple device structure (metal-insulator-metal) of a RRAM device, its compatibility with CMOS process, the scaling opportunities below 8nm, its large on/off ratio, and fast operating speed make the RRAM devices ideal candidates to eventually be used as embedded memories.

Challenges: Amongst the greatest challenges faced by today's RRAM devices is their relatively low endurance ($10^5 - 10^{10}$ cycles [22]) and poor uniformity. The low endurance limits their efficiency as embedded memories, while the poor uniformity causes extreme variability and limited reproducibility.

Another challenge is the large number of new materials (and combinations of materials) which can be used for the resistive stack formation (as seen in Fig. 1a) making difficult the standardization of the fabrication process. The introduction of new materials in RRAM fabrication does not allow enough time to collect and generate the data required to guarantee sufficient yield. These issues, which are common to all emerging technologies, introduce aggressive challenges on defect and fault modelling and possible test solutions.

The main concern regarding the RRAM is the variability of its switching parameters. It has been demonstrated that lowering the switching power of RRAM will induce large variability of the filament formation. This is due to movement of only countable number of atoms [1]. This movement can cause device to device resistance variation but also cycle to cycle variation [23]. This has resulted in the need for enhancing the read/write power consumption in RRAM [24] along with designing adaptive sensing circuits to mitigate this effect [25]. Moreover, it has motivated the utilization of other types of RRAMs such as the conductive bridging RAM (CBRAM) which has bigger high to low resistance ratio and non-filamentary RRAM devices in which the ions drift across the whole aperture of RRAM [26].

In terms of scaling RRAM has shown to be promising device as its data storage is based on atomic movement. Theoretically, RRAM can scale down to size of a conductive filament and scaled devices down to 2nm has been reported. Nonetheless extensive scaling can make the filament so small and might induce retention problems [27].

Note that as a result of filamentary switching, RRAM scaling will not be accompanied by scaling of the operating voltage and currents because the filament conduction is governed by the electrical programming conditions, therefore to achieve low power constrains proper material selection and optimized programming conditions is required [28].

From reliability perspectives, RRAMs have recently improved a lot from their early stage appearances. Their endurance cycles have increased from 10^3 up to 10^{12} cycles and there has been some attempts to remove the initial forming step which is one of the sources for their resistance variability [29]. A consolidation of material engineering along with optimizing the device operating parameters and novel techniques at circuit level are under research to further improve their reliability.

Due to attractive potential scaling and fast operation properties of RRAM they are considered as strong replacement for flash memories. Several prototyped chips have been presented for RRAM devices. As examples Panasonic has presented an 8Mb fast RRAM memory [30], also Scandisk/Toshiba have presented a 32 Gb chip for high density applications.

III. TOWARD AUTOMATION OF MEMRISTIVE DEVICE BASED CIRCUIT DESIGN

A fundamental component of any kind of computing architecture is the implementation of boolean logic functions thus, an automated tool for the synthesis of memristor-based circuits is mandatory [39,40]. In [31], the authors proposed a methodology for the synthesis of boolean logic function on a memristor-based crossbar. Their work showed that is possible to implement any kind of boolean function on a memristor-based crossbar. In [32], we illustrated a methodology to automatically map an arbitrary boolean function to a memristor-based crossbar implementation. By applying different minimization tools and different synthesis parameters, we also showed that each obtained architecture is strongly dependent on them. Design Space Exploration (DSE) is

therefore mandatory to help and guide the designer to select the best architecture.

Bearing in mind such consideration, in this section, we present a formal DSE approach that aims to calculate interesting circuits attributes avoiding simulation campaigns. We propose an algorithmic method to estimate both workload independent attributes (e.g. performance, area, etc.) and workload dependent ones. In particular, we estimate the power consumption of a given memristor-based crossbar architecture (the Fast Boolean Logic Circuit [31]) providing both a lower and an upper bound for the power consumption and an error estimation.

A. Synthesis Flow and DSE

As described in [31], FBLC approach implements a boolean function as a Sum-of-Product (SoP). Thus, the resulting crossbar has to be configured accordingly to the function's minterms. The proposed synthesis flow is depicted in the 0The input of the flow is the target boolean function that is minimized by using two different synthesis tools (i.e., ABC [33] and SIS [34]). Actually, we exploited two different tools to estimate the impact of different synthesis parameters and algorithms on the circuit characteristics (i.e., performance, area, power consumption, etc.). More in detail, SIS is employed for generating 2-levels logical networks while ABC is exploited for generating multi-levels logical networks. The result is the boolean function minimized and described as a set of minterms. As described above, different descriptions can be obtained. The subsequent step is the mapping of the minimized boolean function onto a crossbar-based memristor circuit. The tool XbarGen [32] can extract the function's minterms from the generated representation in order to analyze them and to build the corresponding FBLC circuit. The result is the set of VHDL files modelling the crossbar circuit. Finally, the crossbar VHDL model can be simulated by using any available logic simulator.

During the mapping process, XbarGen extracts the crossbar attributes that will be exploited by the proposed formal DSE approach. Let us first detail those attributes before moving to the DSE description. They can be divided in two main categories, namely the workload independent and workload dependent. Next subsections describe both of them and last subsection details the formal DSE.

B. Workload independent attributes

The workload independent attributes do not need any simulation (i.e., we do not have to simulate the crossbar VHDL model) to be evaluated. They are extracted by XbarGen during the mapping process and they are formalized as follows:
Number of memristors in the circuit defined by the following equation:

$$N_m = \sum_j \left[2 * N_{in}(l_j) + \sum_i (N_{occ}(m_i, l_j)) + \sum_i (N_{lit}(m_i) * p_{ij}) + 2 * N_{out}(l_j) \right] \quad (1)$$

Total area of the circuit defined by the following equation:

$$Area = \sum_j \left\{ [2 * N_{in}(l_j) + 2 * N_{out}(l_j)] * \left[1 + \sum_i (p_{ij}) + N_{out}(l_j) \right] \right\} \quad (2)$$

Number of crossbars, i.e. NC

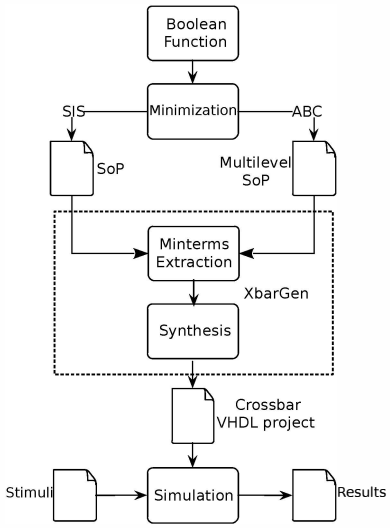


Figure 2. Synthesis flow

Response time of the circuit defined by the following equation:

$$RespTime = T_C * N_C \quad (3)$$

given that:

- Indexes *i* and *j* run on minterms and crossbars respectively;
- N_{in} and N_{out} are the number of inputs and outputs respectively;
- N_{occ} (m_i , l_j) is the number of occurrence of i-th minterm in j-th crossbar;
- N_{lit} (m_i) is the number of literals of i-th minterm;
- p_{ij} is equal to 1 if the *i-th* minterm is present in the *j-th* crossbar, otherwise it is equal to 0;
- T_C is the 'Latency' of a Crossbar;
- N_C is the Number of Crossbars in the circuit.

C. Workload dependent attributes

The workload dependent attributes require the simulation of the generated VHDL circuits to be evaluated. In this work, we consider the **power consumption** as workload dependent attributes formalized as:

$$P = \sum_j \left[N_{up_j} \cdot C_{up} + N_{down_j} \cdot C_{down} \right] \quad (4)$$

given that:

- index *j* runs on crossbars;
- N_{upj} and N_{downj} are the number of memristors in the *j-th* crossbar that switch from '0' to '1' and from '1' to '0' respectively;

- C_{up} and C_{down} are the power consumption of a memristor switching from '0' to '1' and from '1' to '0' respectively.

It is worth to note that N_{upj} and N_{downj} depend on the applied workload.

D. Formal DSE

The main goal of the proposed DSE is the characterization of the synthesized crossbars w.r.t. the above identified attributes. The idea is to avoid any simulation to speed up the DSE. Clearly, for the workload independent attributes the formal DSE is straightforward since it is enough to exploit the equations (1), (2) and (3).

The challenging issue is determining the actual power consumption. Even if the power consumption is a workload dependent attribute, we will show how to compute two bounds that cannot be exceeded by the actual power consumption: a *worst* case bound and a *best* case bound. It is worth to emphasize that such bounds will be computed without any simulation.

Referring to the equation (4), the idea that we exploit is to identify within the crossbar the elements that do not depend on actual inputs and manage those which are dependent on actual inputs. Thus, we can observe - as 0 shows - that the architecture has a first RESET stage (INA) in which all the memristor in the circuit are set to '1'.

Therefore, during this stage, we have only the contribution of N_{upj} * C_{up} while, during the rest of the computation, only N_{downj} * C_{down} contributes to the power consumption. Moreover, in both worst and best case scenarios, we consider a concatenation of executions providing inputs which trigger the worst and the best case respectively. Bearing in mind this, we can observe that, whether N_{downj} memristors switch from '1' to '0' during the computation, the RESET phase has to switch the same number of memristors from '0' to '1'. Therefore, considering both worst and best case, we can assume that the two contributions are equal:

$$N_{up_j} = N_{down_j} \tag{5}$$

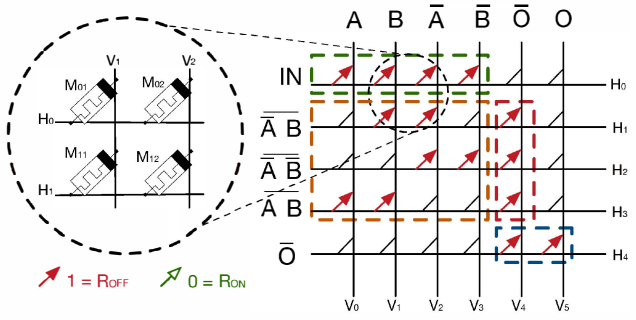


Figure 3. Reset stage

In order to estimate this contribution, let us consider that a crossbar can be divided in 4 parts, as detailed in [32]. Hence we can assume the following:

- Concerning the green IN box, it is clear that half of the input memristors are going to switch during each execution of the circuit.
- The same consideration is true for the output memristors, in the blue OL box.

Therefore, such two blocks of memristors can be evaluated, in terms of switching memristors, independently from the actual input values. Thus, we are able to rewrite the equation 5 as follows:

$$N_{up_j} = N_{down_j} = N_{in}(l_j) + N_{out}(l_j) + N_{int_j} \tag{6}$$

where:

- N_{intj} is the number of memristors that belong to the minterm boxes NAND and AND within the *j-th* crossbar that switches during the computation:

$$N_{int_j} = (N_{m_{NAND}} + N_{m_{AND}})_j \tag{7}$$

Let us now discuss about the remaining orange NAND and the red AND boxes. The memristors of these parts switch accordingly with the actual input values. For them, the goal is to find the two bounds. It is worth highlighting that the number of switching memristors in the AND box depends on which memristors switch in the NAND box. Therefore, the best and worst cases are computed by considering only the NAND box. Bearing in mind that half of the input memristors will eventually switch $1 \rightarrow 0$ during each execution of the circuit, to find the best and worst input vectors we count, for each vertical nanowire, the number of memristors in the NAND box. For each couple of literals xi and $\bar{x}i$ (vertical nanowires) we consider, as for the best case, the one that leads to the lowest number of memristors and, as for the worst case, the one that leads to the biggest one.

Finally, we compute how many AND memristors will switch using the selected input vectors: since the minterms box is performing a NAND operation, if a minterm has at least one literal among those in the selected input vectors, the corresponding memristor in the AND box will not switch, otherwise it will.

As can be seen, we do not need the truth table of the function for retrieving $(N_{mNAND}, N_{mAND})_{worst/best}$, indeed we only count the number of memristors in the NAND box in order to find the best and the worst combinations of inputs and then we verify if each minterm will be whether '0' or '1'. Therefore, the algorithm has a *linear* complexity. This is a great improvement compared to doing a simulation with all the combinations of inputs that would lead to a complexity $\theta(2n)$, with n the number of inputs

E. Experimental Results

This section provides experimental results achieved by the proposed flow. A bunch of combinatorial circuits are used as benchmarks, details about circuits characteristics are available in [35].

TABLE I. EXPERIMENTAL RESULTS

Exp.	IN	Multiple Crossbar										
		Minterms	N_m	Area	Time	Xbars	P_{worst}	E_{worst}	P_{best}	E_{best}	DSE Time (ms)	SimTime Overhead
xor5	5	22	156	894	42	6	80	20	70	14	7,20985	536,45%
rd53	5	58	414	4790	56	8	215	56	183	40	20,3347	468,59%
squar5	5	71	515	6546	56	8	274	69	228	56	25,9582	476,89%
con1	7	19	139	804	35	5	71	19	62	13	6,58962	524,34%
5xp1	7	140	985	23218	70	10	512	133	438	98	61,8654	418,40%
rd73	7	155	1135	27178	91	13	601	146	502	114	75,499	398,81%
Z5xp1	7	199	1430	38432	84	12	751	191	630	142	96,445	399,95%
misex1	8	72	504	7314	49	7	264	68	226	54	23,0975	511,97%
rd84	8	374	2756	138236	119	17	1459	354	1214	271	268,565	261,77%
ex5	8	1224	8648	1228506	105	15	4544	1188	3776	860	1761,63	141,35%
clip	9	202	1514	42884	77	11	802	196	667	151	108,394	377,48%
9sym	9	239	1773	66596	98	14	936	231	776	170	140,665	342,63%
Z9sym	9	266	1986	66928	112	16	1052	258	869	193	156,091	340,54%
apex4	9	3151	22589	8125002	133	19	11913	3045	9839	2208	11209,4	129,68%
sao2	10	168	1280	32520	98	14	684	157	567	128	84,4688	405,69%
alu4	14	1464	10734	1734366	154	22	5695	1415	4692	1068	2775,75	123,68%
misex3	14	1561	11429	1757416	161	23	6072	1510	4997	1150	2933,87	123,98%
table3	14	2152	15562	3153604	161	23	8294	2094	6777	1603	4743,15	142,25%
b12	15	89	649	12954	56	8	341	85	289	66	37,3648	436,55%
table5	17	1980	14467	2415332	175	25	7774	1941	6296	1544	3939,18	143,80%
duke2	22	673	5015	381946	147	21	2686	651	2199	521	692,15	205,72%
cordic	23	111	865	16618	119	17	459	100	390	84	54,124	417,33%
misex2	25	121	921	19698	77	11	494	118	405	96	49,9148	479,33%
vg2	25	208	1564	43002	140	20	821	198	694	149	113,663	373,55%
apex2	39	445	3627	134188	203	29	1986	439	1597	395	354,094	287,99%
seq	41	2413	17877	3580692	203	29	9599	2357	7790	1869	5833,47	155,45%
apex1	45	2626	19093	4823786	189	27	10151	2566	8315	1939	7179,4	146,71%
apex3	54	2240	16127	4110536	140	20	8510	2178	7023	1584	5738,37	130,22%
e64	65	1437	12134	416608	448	64	6754	1437	5349	1406	1535,65	262,27%
apex5	117	1293	9630	1239620	147	21	5346	1276	4197	1189	2412,64	120,04%
o64	130	130	1162	67872	56	8	645	130	517	130	159,803	180.74%

Table I reports the achieved results for multiple crossbars. For each circuit, tables report the number of inputs (*IN*), minterms, memristors (*Nm*), the area, the estimated power consumption from P_{worst} downto P_{best} including the error estimations (E_{worst} and E_{best}). Moreover, in columns *DSE Time* and *SimTime Overhead* we report respectively the execution time of the proposed formal DSE and the overhead of a single simulation of the synthesized VHDL circuit compared to DSE. As shown in the tables, the cost of performing a full simulation to determine the power consumption is in average very high: about 300% for the multiple crossbar. This clearly prove the benefits of using our approach instead of a full simulation.

It is worth mentioning that, on a given circuit, the simulation is performed on a single workload while the formal DSE execution is actually *independent* from any specific workloads. Thus, the formal DSE takes only an execution in order to perform the estimation of both best and worst cases of power consumption, along with an error estimation. The simulation approach, on the other hand, require a full simulation per each workload to find actual best and worst cases of power consumption.

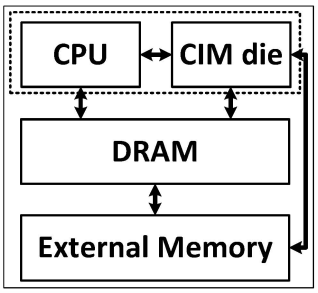


Figure 4. Proposed computing architecture

IV. MEMRISTIVE DEVICE BASED COMPUTING: MYTH OR REALITY?

Memristive devices are under investigation to explore not only their potential for building memories and logic, but also to build radically new computing architectures [36], such as computing-in-memory (CIM).

CIM concept as proposed in [11] is based on the tight integration of computation and storage in a very dense crossbar array where memristors are injected at each junction of the crossbar (top electrode and bottom electrode). The communication and control from/to the crossbar can be realized using CMOS technology; it dictates what kind of operations have to be performed within the crossbar depending on the sequence and the voltages applied to the wordlines and the bitlines of the crossbar. The integration of CIM die with a conventional CPU can enable the realization of accelerators for specific application; these accelerators could outperform the traditional CPU with orders of magnitude [11].

In the rest of this section, we will briefly discuss an example of such an accelerator, including the architecture and principal of working, the potential applications that could benefit from such architecture, and some preliminary performance results.

A. Principal of working

Figure 4 shows the CIM-based computing architecture concept. It includes a conventional CPU, CIM die, main memory DRAM and external memory. In a conventional architecture, the CPU fetches, decodes and executes a big data program in which intensive memory accesses costs enormous energy consumption and significantly degrades the overall performance. In CIM-based architecture, the aim is to have as much as possible parts of the program (requiring big data) executed locally within the CIM-die; i.e., instead of moving data back and forth to the cache & register file, the (big) data which is initially stored in the CIM die, will be kept in the same location and (ideally) all parts of the program that should be executed on this data should take place locally within the CIM-die. Only the final result is moved from the CIM-die to the master CPU. Note that the operations within CIM-die are performed within the (non-volatile) memory; moreover, the CIM die can perform multiple operations simultaneously without the overhead of fetching data from memory [11, 38]. Therefore, significant reduction in memory access and the use of non-volatile technology for the CIM-die will not only improve the overall performance, but also dramatically reduce energy consumption. Note that idea is to have the (big) data (on which the CIM die needs to perform the operations) loaded to the die itself, and that the capacity of the memory within the die is large enough to store this data. In addition, the optimized performance is obtained in case the we have different operation applied on the same data, which is not changing frequently; this benefit also the endurance of the non-volatile memory of the CIM die.

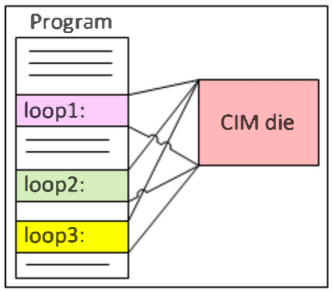


Figure 5. Example program

B. *Potential applications*

It is well recognized that the communication between the CPU and the memory is the killer for both performance and energy, especially for data-intensive applications; loading data from on chip SRAM and from off-chip DRAM cost about 50X, respectively, 6400X more energy as compared with an ALU operation [37]. Clearly reducing the communication will have a significant impact not only on the energy consumption but also on the overall performance. Data-intensive applications which requires the execution of algorithms containing also operations on big data could benefit from the architecture discussed above. The big data can be stored on the CIM die and all the operations that have to used such data can be execute locally with the CIM die, by have the CPU provide appropriate instruction to the local CIM controller; hence, no need for data movement. Moreover, the CIM dies can perform its operation while the CPU is also executing other operation in parallel, resulting in overall performance improvement.

To illustrate the above, assume the program of Figure 5. The program has three loops that need to make use of the big data, and are supposed to run on the CIM die; obviously the data should be stored on the same die. Each time the loop is invoked, the CPU sends a request to the CIM die; the latter, performs the requested operations and returns the results to the CPU. It is possible to maximize the overall performance by having the CPU send the instruction to the CIM die ahead of time and such that the CIM-die will make sure that the results are ready when needed by the CPU. Examples of applications that have similar characteristics as the program of Figure 5 are database applications, where multiple queries (each consisting of large loops) are applied to a fixed database. These queries are used to look for specific data patterns in the database.

C. *Some preliminary results*

To illustrate how CIM-based computing architecture advances the state-of-the-art, the performance of CIM-based and multicore-based architectures are estimated. We assume a program with a number of parallel instructions executed on the two architectures. Assumptions on multicore architecture and CIM-based architecture are similar to those in [11]; the multicore architecture consists of 4 cores (ALU only), 32KB cache and 1GB DRAM; CIM-based architecture consists of one core (ALU only), CIM die with a special computing unit and the data capacity equal to 32KB cache, and 1GB DRAM. The computations are performed by ALU and the special computing unit. The memory operations are modeled by a memory model based on cache miss rate and DRAM access time similarly as in [11]. Two metrics are used for the evaluation: (1) Energy Delay Product Efficiency (EDP) defined as the number of executed instruction divided by the energy delay product, and (2) Energy Efficiency (EE) defined as the number of executed instruction divided by consumed energy.

Fig. 6 shows that CIM-based architecture outperforms multicore architecture; the Energy-Delay product efficiency is two orders of magnitude better while Energy efficiency is ~ 5X better. The improvements are results of significant memory access reduction and the usage of non-volatile memory. The reduction of CPU memory accesses leads to a lower latency and lower energy consumption, while the non-volatile memory reduces the static power to practically zero.

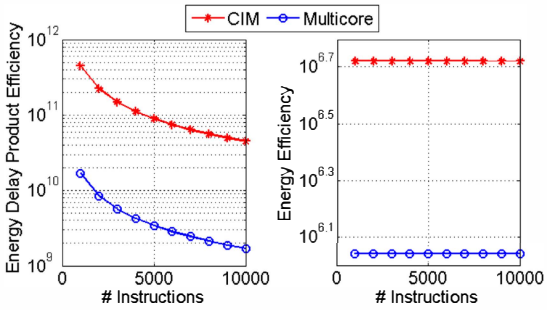


Figure 6. Performance comparison between CIM and multicore

REFERENCES

[1] ITRS 2015 report. [Online]. Available: http://www.itrs.net/

[2] B.Hoefflinger,"Chips2020:AGuidetotheFutureofNanoelectronics", The Frontiers Collection, Springer Berlin Heidelberg, 2012, pp. 421– 427

[3] J. McPherson, "Reliability trends with advanced CMOS scaling and the implications for design", in IEEE Custom Integrated Circuits Conference, 2007, pp. 405–412

[4] S. Borkar, "Design perspectives on 22Nm CMOS and beyond", in Proceedings of the 46th Annual Design Automation Conference, 2009, pp. 93-94

[5] G.Gielen,etal.,"Emergingyieldandreliabilitychallengesinnanometer CMOS technologies", in Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1322-1327, 2008

[6] L.Chua,'Memristor-themissingcircuitelement',IEEETransactionson Circuit Theory, vol. 18, no. 5, pp. 507–519, 1971

[7] R. Waser et al., "Redox-based resistive switching memories–nanoionic mechanisms, prospects, and challenges," Advanced Materials, vol. 21, pp. 2632–2663, 2009

[8] J. J. Yang et al., "Memristive devices for computing," Nature nanotechnology, vol. 8, pp. 13–24, 2013

[9] J. R. Burger et al., "Variation-tolerant computing with memristive reservoirs," IEEE/ACM International Symposium in Nanoscale Architectures (NANOARCH), 2013, pp. 1–6.

[10] K.-H. Kim et al., "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," Nano letters, vol. 12, pp. 389–395, 2011.

[11] S. Hamdioui, et al., "Memristor based computation-in-memory architecture for data-intensive applications," in Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1718-1725, 2015

[12] J.-K. Lee, Sunghun Jung, Jinwon Park and Sung-Woong Chung, "Accurate analysis of conduction and resistive-switching mechanisms in double-layered resistive-switching memory devices," *Appl. Phys. Lett.*, vol. 101, 2012.

[13] L. Zhu, Z. Jian , Z. Guo and Z. Sun, "An overview of materials issues in resistive random access memory," Journal of Materiomics, vol. 1, no. 4, pp. 285-295, 2015.

[14] S. Yu, "Overview of resistive switching memory (RRAM) switching mechanism and device modeling," in IEEE International Symposium on Circuits and Systems (ISCAS), 2014.

[15] E. I. Vatajelu, H. Aziza and C. Zambelli, "Nonvolatile memories: Present and future challenges," in 9th International Design and Test Symposium (IDT), 2014.

[16] E. W. Lim and I. Razali, "Conduction Mechanism of Valence Change Resistive Switching Memory: A Survey," Electronics, vol. 4, no. 3, pp. 586-613, 2015.

[17] A. Wedig, M. Luebben, D.-Y. Cho, M. Moors, K. Skaja, V. Rana, T. Hasegawa, K. K. Adepalli, B. Yildiz, R. Waser and I. Valov, "Nanoscale cation motion in TaOx, HfOx and TiOx memristive systems," Nat Nano, vol. 11, no. 1, pp. 67-74, 2016.

[18] W. R, "Electrochemical and thermochemical memories," in IEEE International Electron Devices Meeting, 2008.

[19] I. Daniele, B. Rainer and W. Rainer, "Thermochemical resistive switching: materials, mechanisms, and scaling projections," Phase Transitions, vol. 84, no. 7, pp. 570-602, 2011.

[20] I. Marco, P. Paolo and V. Elena Ioana, "On the impact of process variability and aging on the reliability of emerging memories (Embedded tutorial)," in 19th IEEE European Test Symposium (ETS), 2014.

[21] G. S, "Embedded memory design for future technologies: Challenges and solutions," VLSI-Design, 2014.

[22] I. D and W. R, Eds., Resistive Switching: From fundamentals of nanoionic redox process to memristive device applications, Wiley-VCH Verlag GmbH&Co, 2016.

[23] A. Chen and L. M. R., "Variability of resistive switching memories and its impact on crossbar array performance," in International Reliability Physics Symposium, 2011.

[24] X. Xue, W. Jian, J. Yang, F. Xiao, G. Chen, S. Xu, Y. Xie, Y. Lin, R. Huang, Q. Zou and J. Wu, "A 0.13 um 8 Mb Logic-Based CuxSiyO ReRAM With Self-Adaptive Operation for Yield Enhancement and Power Reduction," IEEE Journal of Solid-State Circuits, vol. 48, no. 5, pp. 1315-1322, 2013.

[25] J. K. Park, S. Y. Kim, J. M. Baek, D. J. Seo, J. H. Chun and K. W. Kwon, "Analysis of resistance variations and variance-aware read circuit for cross-point ReRAM," in 2013 5th IEEE International Memory Workshop, 2013.

[26] B. Govoreanu, D. Crotti, S. Subhechha, L. Zhang, Y. Y. Chen, S. Clima, V. Paraschiv, H. Hody, C. Adelmann, M. Popovici, O. Richard and M. Jurczak, "A-VMCO: A novel forming-free, self-rectifying, analog memory cell with low-current operation, nonfilamentary switching and excellent variability," in Symposium on VLSI Technology (VLSI Technology), 2015.

[27] Y. Y. Chen, M. Komura, R. Degraeve, B. Govoreanu, L. Goux, A. Fantini, N. Raghavan, S. Clima, L. Zhang, A. Belmonte, A. Redolfi, G. S. Kar, G. Groeseneken, D. J. Wouters and M. Jurczak, "Improvement of data retention in HfO2/Hf 1T1R RRAM cell under low operating current," in IEEE International Electron Devices Meeting, 2013.

[28] D. J. Wouters, R. Waser and M. Wuttig, "Phase-Change and Redox-Based Resistive Switching Memories," Proceedings of the IEEE, vol. 103, no. 8, pp. 1274-1288, 2015.

[29] H. S. P. Wong, H. Y. Lee, S. Yu, Y. S. Chen, Y. Wu, P. S. Chen, B. Lee, F. T. Chen and M. J. Tsai, "Metal–Oxide RRAM," Proceedings of the IEEE, vol. 100, no. 6, pp. 1951-1970, 2010.

[30] A. Kawahara, R. Azuma, Y. Ikeda, K. Kawai, Y. Katoh, Y. Hayakawa, K. Tsuji, S. Yoneda, A. Himeno, K. Shimakawa, T. Takagi, T. Mikawa and K. Aono, "An 8 Mb Multi-Layered Cross-Point ReRAM Macro With 443 MB/s Write Throughput," IEEE Journal of Solid-State Circuits, vol. 48, no. 1, pp. 178-185, 2013.

[31] Lei Xie, Hoang Anh Du Nguyen, Mottaqiallah Taouil, Said Hamdioui, Koen Bertels, “Fast Boolean Logic Mapped on Memristor Crossbar”, IEEE International Conference on Computer Design (ICCD), pp. 335-342, 2015.

[32] M. Traiola, M. Barbareschi, A. Mazzeo and A. Bosio, “XbarGen: A memristor based boolean logic synthesis tool,” 2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Tallinn, Estonia, 2016, pp. 1-6.

[33] ABC User Guide. [Online]. Available: http://www.eecs.berkeley.edu/~alanmi/abc/

[34] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, Robert K. Brayton and Alberto L. Sangiovanni-Vincentelli, “SIS: A System for Sequential Circuit Synthesis”, EECS Department University of California, Berkeley Technical Report No. UCB/ERL M92/41 1992

[35] Saeyang Yang, “Logic Synthesis and Optimization Benchmarks User Guide”. [Online]. http://ddd.fit.cvut.cz/prj/Benchmarks/LGSynth91.pdf:

[36] S. Hamdioui, et.al, “Memristor For Computing: Myth or Reality?”, in Proceedings of the Conference on Design, Automation and Test in Europe, 2017.

[37] Data from J. Brunhave, W. Dally and M. Horowitz, Stanford University.

[38] H.A. Du Nguyen, et.al, Computation-In-Memory Based Parallel Adder, IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2015.

[39] H. A Du Nguyen, et al., "Synthesizing HDL to memristor technology: A generic framework." In IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2016.

[40] Yu Jintao, et al., "Skeleton-based design and simulation flow for Computation-in-Memory architectures.", IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2016.

[41] S. Yu, B. Lee, H.-S. P. Wong, “Metal Oxide Resistive Switching Memory,” book chapter: “Functional metal-oxide nanostrudtures,” Published by Springer New York, ISSN 0933-033X, 2011.