Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

Estimating dynamic power consumption for memristor-based CiM architecture

Marcello Traiola[a,*], Mario Barbareschi[b], Alberto Bosio[a]

[a] LIRMM - University of Montpellier/CNRS - France, 161, rue Ada, Montpellier cedex 5 34095, France
[b] DIETI - University of Naples Federico II - Italy, Via Claudio, 21, Naples 80125, Italy

A R T I C L E I N F O

Keywords:
Computing-in-Memory architecture
Power estimation
Memristor
Design space exploration
Boolean functions
Circuit synthesis

A B S T R A C T

Nowadays, Computing-in-Memory (CiM) represents one of the most relevant solutions to deal with CMOS technological issues and several works have been proposed so far targeting front and back-end synthesis. However, a given CiM architecture can be synthesized depending on different parameters, leading to different implementations w.r.t. area, power consumption and performance. It is thus mandatory to have an evaluation framework to characterize the actual implementation depending on the above terms. This is even more important during the Design Exploration phase, in which many different implementations are explored to identify the best candidate w.r.t. the user requirements. In this work, we focus on the dynamic power consumption estimation of a given CiM implementation. Instead of resorting to a simulation-based power estimation, we propose an analytical approach that will dramatically speed up the estimation since no simulations are required. By comparing the proposed approach against the simulation-based method over a massive experimental campaign, we show that the accuracy of the estimation turns out to be very high.

1. Introduction

Energy-efficient computation is one of the most challenging problem faced today by the research community. Two main walls have to be broken in order to go beyond state-of-the-art solutions: architectural and technology walls. The most advanced computing architectures are still based on the Von Neumann or Harvard paradigm, in which data has to be moved from the storage element (typically a Random Access Memory, RAM) to the computational element (namely the CPU) and then moved back. This is nowadays becoming no more affordable due to the fact that most of the time and energy required to complete any task is spent for data transfer only. The second wall is the CMOS technology that is reaching its physical limits and violating the famous Moore's law. New technology nodes will not double the performances as was in the past. Moreover, the leakage current increasing leads to higher static power consumption and lower reliability [1–4].

One of the emerging solutions to enable energy-efficient computation is the so-called Computation-in-Memory (CiM) paradigm: by merging together computation and storage, it is possible to avoid data transfers from and back to memory. This is possible thanks to the memristor [5], a non-volatile device able to act as both storage and information processing unit. Moreover, the memristor presents many advantages: CMOS process compatibility, lower cost, zero standby power, nanosecond switching speed, great scalability, high density and non-volatile capability [6,7].

Thanks to its nature (i.e., computational as well as storage element), the memristor is exploited in different kinds of applications, such as neuromorphic systems [8], non-volatile memories [9] and computing architectures for data-intensive applications. This paper focuses on the latter ones, and more in particular on the CiM architecture proposed in [10]. For this architecture, some works have been proposed so far targeting both front and back-end synthesis [11,12]. However, a given CiM architecture can be synthesized depending on many parameters [11] leading to different implementations w.r.t. area, power consumption and performances. In order to analyze and compare different solutions varying on these parameters, an evaluation framework needs to be properly defined. This is even more important during the Design Exploration phase during which many different implementations are explored to identify the best candidate(s) w.r.t. the user requirements. In this work, we focus on the dynamic power consumption estimation of a given CiM implementation, proposing a front-end evaluation framework. Instead of resorting to a simulation-based power estimation, we propose an analytical approach that will dramatically speed up the estimation time since no simulations are required. Comparing the proposed approach against the simulation-based method, experimental results show that the dynamic power consumption estimation turns out

[*] Corresponding author.
E-mail addresses: traiola@lirmm.fr (M. Traiola), mario.barbareschi@unina.it (M. Barbareschi), bosio@lirmm.fr (A. Bosio).

https://doi.org/10.1016/j.microrel.2017.12.009
Received 28 July 2017; Received in revised form 27 October 2017; Accepted 4 December 2017

significantly accurate. The proposed methodology can be helpful to designers in the early stages of the circuit design (i.e., front-end) for obtaining a preliminary estimation of the dynamic power consumption. In successive phases, back-end analyses can further refine estimations by including technological information.

The remainder of the paper is structured as follows. Section 2 provides the required background about the memristor-based CiM and the state-of-the-art power estimation approaches. Section 3 presents the synthesis flow and the design space exploration framework, while the Section 4 gives the experimental results. Finally, the Section 5 draws the conclusions.

2. Background and state-of-the-art

In this section we provide basic concepts about the memristor-based CiM architecture as well as about power estimation approaches.

2.1. Memristor-based CiM

The memristor is a two-terminal non-linear passive electrical component characterized by a variable electrical resistance, which value depends on the history of the current flowed through the device itself. Since we exploit memristors to implement digital circuits, we refer to the memristor Voltage-Current relation depicted in Fig. 1, detailed in [13], as the best solution for modeling the device behavior (i.e., taking into account the ideal response to a pulse-wave). Such model considers that the voltage applied to memristor terminals does not change the device resistance until one of the two thresholds $\pm V_{th}$ is crossed. In the adopted ideal model, they are symmetrically defined.

We resort to the Snider Boolean Logic (SBL) [13] convention, whereby a lower resistance (steeper curve denoted as R_{ON}) represents a logic 0 while an higher resistance (lower slope curve denoted as R_{OFF}) represents a logic 1. Two basic operations can be performed, defined as SET and RESET. The former allows to program the memristor to R_{ON}, hence at logic 0, while the latter performs the memristor switching to R_{OFF}, that corresponds to logic 1.

2.1.1. Fast Boolean Logic Circuits

Snider proposed in [13] a design methodology to implement boolean functions on a memristor-based crossbar. The proposed approach was then improved by Xie et al. in [14]. Let us briefly recall their proposition referring to it as Fast Boolean Logic Circuit (***FBLC***). First, the logic circuit requires that the Boolean function is expressed as sum of products as shown in Eq. (1).

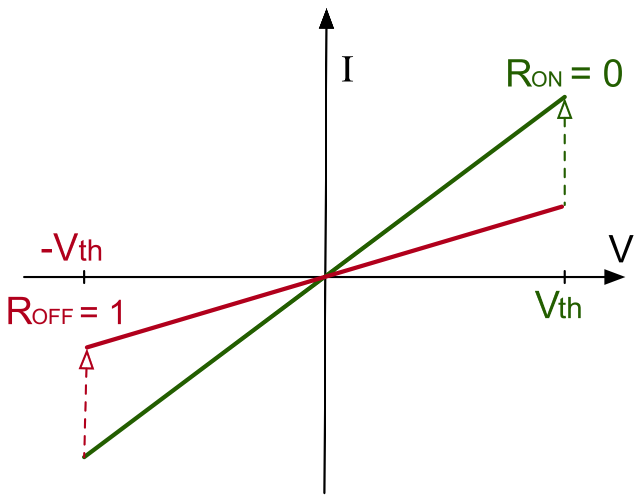

Fig. 1. Memristor ideal model.

$$AB + \overline{A}B + \overline{AB} = \overline{\overline{AB}\cdot\overline{\overline{A}B}\cdot\overline{\overline{AB}}} \tag{1}$$

The left member of the Eq. (1) can be easily manipulated through transformation rules (i.e., De Morgan's laws). The obtained form (right member of the Eq. (1)) can be computed exploiting three boolean operations: NAND, AND and NOT.

Then, as Fig. 2-a shows, FBLC is divided into blocks. The control logic is realized by a Finite State Machine (FSM) through several steps (Fig. 2-b) which are:

INA:	**IN**itialize **A**ll the memristors to R_{OFF};
RI:	the input block **R**eceives the **I**nputs;
CFM:	**C**on**F**igure all **M**interms simultaneously, in parallel;
EVM:	**EV**aluate all **M**interms simultaneously (NAND);
EVR:	**EV**aluate **R**esults: $\overline{F}$ is calculated (AND);
INR:	**IN**vert **R**esults: $\overline{F}$ need to be inverted to achieve f
SO:	**S**end **O**utputs: the result captured in OL is sent out.

Below, the description of the blocks:

- Input box: where inputs are stored during the RI step;
- NAND box: where minterms are configured during CFM and evaluated during EVM;
- AND box: where results of EVM are stored and AND operation is performed during EVR;
- Output box: where results of EVR are stored and inversion operation is performed during INR;

For the purpose of realizing each step of the FSM, the authors of [14] proposed some *primitive operations*. Each of these operations can be performed using as many input and output memristors as desired. By driving the crossbar's nano-wires with proper voltages during each step, it is possible to achieve boolean function calculations in a constant number of steps.

2.2. Power estimation

Power estimation methodologies resort to power models, which are dependent on the target abstraction level. Clearly, the higher estimation precision one desires, the lower abstraction level one has to resort to, at the cost of an increasing estimation time. More in detail, power estimation is based on a macro-modeling approach, in which a power model is created using pre-characterized power values [15]. This power pre-characterization can be computed at different abstraction levels [16,17]. The accuracy and the simulation run-time to obtain the characterization point will depend on the available information of the circuit structure and activity at the considered level (i.e., gate or transistor level). Several works have been proposed at gate-level to assess power consumption per component, i.e. dynamic (switching, short-circuit) and static or leakage power [18,19]. This kind of works models the gate current as a triangular shape taking into account different operational conditions (supply and ground voltages) and input switching conditions (rise, fall, static). Their results show good accuracy with transistor level simulations.

State-of-the-art methodologies need to be adapted for the memristor-based computing architecture. First of all, the static or leakage power does not need to be taken into account anymore, since the memristor is a 0-leakage device [6,7]. Secondly, the fact that CiM architecture is implemented as a memristor-crossbar requires a new approach for computing the dynamic power consumption (i.e., power consumed during memristor switching). In [12], the authors proposed to compute the power consumption of a given CiM architecture as expressed by Eq. (2). It corresponds to the sum of the power consumed by memristor-crossbar (P_{xbar}), by the CMOS voltage drivers ($P_{vd,all}$) and by the crossbar controller (P_{ctrl}) for all steps N_{step} to be executed.