A Resistive RAM-Based FPGA Architecture Equipped with Efficient Programming Circuitry

Behnam Khaleghi, Hossein Asadi *Senior Member, IEEE*

***Abstract*—Recently, superior characteristics of *Resistive Random Access Memory* (RRAM) have made it more promising than other *Non-Volatile Memories* (NVMs) to be employed in both stand-alone memories and configuration bits of *Field-Programmable Gate Arrays* (FPGAs). However, despite the considerable effort has been put on application of NVMs in FPGAs, previously suggested designs are not mature enough to substitute the state-of-the-art SRAM-based counterparts. In particular, their limitations mainly arise from inefficient building blocks and/or overhead of programming structure of NVM-based configuration memories which can impair the potential benefits of employing NVM-based blocks.**

In this paper, we present an RRAM-based FPGA architecture employing efficient *Switch Box* (SB) and *Look-Up Table* (LUT) designs. In the proposed architecture, we present an efficient programming circuitry and integrate it in both SB and LUT designs. The main aim of this integration is to create area and power efficient programmable components while precluding performance overhead to these blocks. In addition, we present an efficient scheme to load the configuration bitstream into the memory elements, which surpasses the previous work by order of magnitudes in term of configuration time, which makes it comparable to that of SRAM-based FPGAs. Besides, we take into consideration the correct functionality and reliability of the programming structure as well as fluctuations in attributes of RRAM cells in the proposed designs to further investigate applicability of the proposed architecture in industrial FPGAs.

We have examined the efficiency of the proposed architecture in terms of delay and power consumption by carrying out detailed HSPICE simulations. The results show that while the proposed SB and LUT designs occupy 67.2% and 41.1% less area in terms of minimum width transistor area, they improve the static power by 27.1% and 2.2% compared with their SRAM counterparts, respectively. The LUT delay is remained intact and the proposed SB reduces delay by 24.2%. We have also utilized VPR tool with the obtained information to investigate effectiveness of the proposed architecture within FPGA-mapped designs. Experimental results over a set of largest MCNC benchmarks demonstrate that the average area and delay of an FPGA using the proposed architecture are 59.4% and 20.1% less than conventional SRAM-based FPGAs. Compared to a recent RRAM-based architecture, the proposed architecture improves the area and power by 49.7% and 33.8% while keeps the delay intact.

***Index Terms*—Emerging Non-Volatile Memory, Resistive Random Access Memory, Field-Programmable Gate Arrays, Programming Circuitry, Resistive Fluctuation.**

I. Introduction

The use of *Field-Programmable Gate Arrays* (FPGAs) has become popular in digital system design because of their appealing features such as flexibility to modify the whole or part of the implemented circuit and reducing the time-to-market by eliminating the design fabrication process. FPGAs, however, still suffer from significant power consumption and area overhead which obstructs widespread use of these devices. The overhead arises mainly from the configuration memory bits [1]–[4], as previous studies show that configuration memory bits contribute to more than 40% of the total area of FPGAs [5], [6]. In addition to power and area overhead, in which the latter also leads to performance degradation by elongating the wires, the configuration memories of FPGAs also make it inherently susceptible to soft errors [7], [8] and voltage fluctuations [9]. Such threats consequently endanger the design reliability, particularly in the deep nano-CMOS era.

To address the area and power issues of FPGAs, several studies have attempted to employ emerging *Non-Volatile Memories* (NVMs), particularly *Resistive Random Access Memory* (RRAM), as a replacement candidate for SRAM-based configuration memory. RRAM is one of the most promising NVM technologies that has attracted intense attention due to its unique advantages [10]. Structurally, RRAM is a two-terminal device comprised of an oxide layer placed between two metal electrodes, composing a metal-insulator-metal structure. By taking advantage of intensified (i.e., detectable) resistance switching of oxide materials, RRAM establishes resistive switching between a *High Resistance State* (HRS) and a *Low Resistance State* (LRS) through applying an appropriate programming voltage. Switching from HRS to LRS is called as *set* process and switching from LRS to HRS is named as *reset* process. There are two switching modes of RRAMs which can be distinguished as unipolar and bipolar modes. In unipolar switching RRAM, the switching direction (from HRS to LRS or vice versa) depends only on magnitude of the applied voltage. Conversely, bipolar switching indicates that the polarity (direction) of the applied voltage determines the switching direction [11]. Using materials compatible with CMOS *Back-End-Of-Line* (BEOL) fabrication process allows RRAM to be stacked on between top metal layers of device. For instance, ZrO_2 materials are used in CMOS devices as high-k dielectric which can also be employed to devise RRAM devices [12]. Such integration provides excellent CMOS compatibility and high density along with zero-leakage nature of these devices.

Integration of RRAM atop CMOS can help reduce the area taken by configuration cells in FPGAs, resulting in power-efficient, smaller, and thereby faster FPGAs [5], [13]. In addition, RRAM provides superior features such as fast write operation, small cell area, and higher density as compared to other NVM counterparts such as *Phase Change Memory* (PCM) or *Spin-Transfer Torque Magnetic RAM* (STT-MRAM) [10]. On the other hand, compared with *Static RAM* (SRAM) and *Dynamic RAM* (DRAM), the relatively higher write duration and write energy of RRAM is much less problematic for FPGAs as compared to cache memory in microprocessors. This is due to the fact that updating the configuration bits of FPGAs is done quite less frequent than memory bits in the cache of microprocessors.

Beside the aforementioned advantages, in contrary to SRAM cells which directly provide the required logical zero or one, exploiting RRAM within FPGAs needs additional peripheral circuitry due to its resistive nature. The resistance state of RRAM cannot be directly utilized in all parts of the FPGA and requires additional circuitry to convert the resistance state to the corresponding voltage level. Particularly, unlike the memory array that can share a sensing circuitry per each column, all of the FPGA configuration cells are continuously being read, demanding efficient sensing peripheral per each cell. In addition, further circuitry is required to enable the write operation for each individual RRAM cell, resulting in larger and less efficient FPGA blocks.

Previous studies on employing RRAM and similar emerging resistive memory technologies in FPGAs range from detailed circuit level to abstract architecture level designs. Several works have designed functional memory elements using RRAM and other emergin technologies that can directly replace SRAM cells in FPGAs [2], [13]–[16]. There are also other works that have proposed various NVM-embedded blocks to come up with more efficient FPGAs [6], [18]–[24]. These studies suffer from one of the following shortcomings: a) they *only* focus on architectural representation of integrating the non-volatile memories in FPGA and neglect either functionality and/or efficiency of the programming and read circuitries [21], [25], as well as an efficient scheme to load and control the bitstream at the system-level [2], [6], [14], [15], [18]–[20],

Both authors are with the Sharif University of Technology, Tehran, Iran. E-mails: behnam_khaleghi@ce.sharif.edu, asadi@sharif.edu.

or b) they propose area and power inefficient SRAM-like configuration cell which also comprises additional programming transistors [2], [14], [26] or *Look-Up Table* (LUT) and *Switch Box* (SB) designs [6], [20]. Moreover, exploiting conventional 2-NMOS/1-Resistor programming structure in these studies, in addition to imposing considerable area overhead, has several major drawbacks as will be detailed in Section II.

In this paper, we present an efficient FPGA architecture employing SB and LUT designs. In the proposed architecture, we design a power-efficient RRAM-based LUT and SB which conforms with conventional FPGA architectures, providing a promising substitute for SRAM-based equivalents. The proposed architecture is equipped with additional circuitry to allow efficient programmability of RRAM cells with minimum area and performance overhead. In the proposed architecture, we add programming paths to the SBs so that all of its internal configuration RRAM cells can be effectively programmed through shared transistors. We have also designed the programming path for a RRAM-based LUT such that the programming takes place with minimum overhead. Finally, we investigate how these two blocks are integrated into the FPGA so that the state-of-the-art FPGA programming circuitries become compatible with the proposed designs. Similar to state-of-the-art flash-based FPGAs, in the proposed architecture we employ conventional CMOS-based flip-flops.

We have exploited HSPICE [27] with physical RRAM model [28] and *Versatile Place and Route* (VPR) tool [29] to examine the proposed designs individually and within an FPGA built upon these designs. HSPICE circuit-level simulations demonstrate that the proposed SB and LUT consume 27.1% and 2.2% less static power compared with their SRAM counterparts. In addition, replacing the SRAM cells with RRAM equivalent reduces the SB and LUT area by 67.2% and 41.1%, respectively. Experimental results over a set of largest *Microelectronics Center of North Carolina* (MCNC) benchmarks [30] show that the overall area and delay of an FPGA using the proposed components are 59.4% and 20.1% less than conventional SRAM-based FPGAs, respectively.

Our novel contributions in this paper are as follows:
(1) We propose an RRAM-based SB design, by leveraging conventional pass-gate based SB, which is equipped with efficient configuration circuitry by sharing programming transistors.
(2) Programmability of the proposed SB is verified by conducting circuit-level SPICE simulations exploiting physical RRAM model, which is neglected in previous studies.
(3) An efficient LUT design using RRAM-based low-leakage voltage-divider configuration cells is proposed. Unlike previous studies [6], [31], the proposed LUT uses only two small programming transistors and is augmented with introduced bypass transistors which allow using intermediate boosting buffers, making it scalable for large inputs.
(4) We demonstrate the integration of proposed designs and associated programming circuitries within the whole FPGA programming scheme. The controlling circuitry as well as the pulse generation scenarios and corresponding timing control are well elaborated.
(5) Detailed evaluation and comparison with the state-of-the-art techniques [2], [6], [14], [23] considering different figure of merits and using realistic RRAM model and parameters are conducted.
(6) The impact of RRAM parameter variation on efficacy of the proposed architecture has been investigated.

The rest of the paper is organized as follows. Section II details previous studies on NVM-based FPGAs. Section III details the proposed SB and LUT architectures. The experimental setup and simulation results have been reported in Section IV. Section V investigates the reliability challenges of the proposed architecture such as impact of parameter variations on its efficiency of the proposed architecture. Finally, Section VI concludes the paper.

II. Related Work

Several studies have been conducted to employ NVMs within FPGAs. While most of the early works have focused on STT-MRAMs [14], [20], [23], [24], most recent studies have focused on employing PCM [2], [6], [15], [31], [32] and RRAM [13], [18], [21], [22], [26], [33]–[35] in reconfigurable devices. These studies either introduce NVM-based configuration cells as a direct substitute for conventional SRAM cells, or propose customized logic and/or routing blocks composed of NVMs. Here we also review the related work that has employed other NVM technologies such as STT-MRAM or PCM, as well, since in majority of these works different NVM cells could be analogously utilized. In other words, the design structure has the same importance as the employed NVM technology. In the following, we review the related work by categorizing them into (a) NVM-based configuration cell in which a substitute for SRAM-based configuration cell has been proposed and can be directly employed in conventional logic blocks and routing of SRAM-based FPGAs, and (b) NVM-based LUT/SB in which the typical LUT/SB structure has been modified in order to efficiently embed the NVMs, i.e., no specific SRAM equivalent cell is proposed.

A. NVM-Based Configuration Cell

A configuration cell comprising two NVM cells embedded in an SRAM-like sensing structure has been proposed in [2], [14], [26]. While [14] utilizes STT-MRAM cells, [2] and [26] exploit PCM and RRAM cells, respectively. In the proposed cells, NVM cells which are programmed to complementary resistive states, are placed between ground and driver (NMOS) transistors of an SRAM. A so-called sense transistor is also added which connects the Q and $\bar{Q}$ nodes in the case of enabling, i.e., during the device power-on. Releasing the sense transistor causes a race between the nodes and sets them to 0 or 1 based on their relative resistance (NVM) to ground. These cells provide temporal power-down and instantaneous power-on since reconfiguring the same design can be done instantly with enabling the sense signal. However, in addition to an extra (sense) transistor that increases the leakage power of the cell, at least two additional (large) transistors are required to provide programmability, which leads to further area overhead. The overhead becomes worse considering the fact that each RRAM and PCM cell occupies larger area than an SRAM. For instance, the PCM cells used in [2] and [34] occupy $8.75\mu m^2$ footprint which is $30\times$ larger than a typical $140F^2$ SRAM in 45nm technology. Therefore, there is a high chance that the NVM cells to become area bottleneck. Moreover, such NVM-based cells still suffer from the positive-feedback structure of SRAM which is susceptible to soft errors. Functionality of programming structure for the aforementioned cells has not been examined. For instance, [26] uses SRAM internal node to provide the ground voltage. Therefore, the associated transistors should be upsized to tolerate the required current which is neglected in [26].

A non-volatile FPGA employing typical voltage-divider based configuration cells composed of two PCMs as a replacement for SRAM cell has been proposed in [32]. While the cell is reported to improve area and power consumption of LUT and SB, peripheral circuitries and the corresponding scheme required to program the cells are disregarded. Furthermore, a considerable direct current (i.e., $1V$ voltage) is permanently applied on the PCM cells that may lead to a read disturb in PCM-based cells as it has been demonstrated in [36].

A similar PCM based voltage-divider storage element has been proposed in [15] that can replace SRAM cell. The authors report 40% delay improvement achieved by using low resistance PCM (compared with resistance of an NMOS transistor). However, the PCM cells have been employed as configuration cells of the routing multiplexers and produce constant voltages, and hence, they are not in the critical path of the blocks and keep the delay intact. In addition, HRS of such PCM cell with $LRS = 3.7K\Omega$ is estimated to be in the range of Mega-Ohm [6] which consumes a huge leakage power around $1\mu W$.

A general stand-alone memory architecture is suggested to program the cells, but important details (e.g., controlling the word- and bit-lines, loading bitstream, etc.) have been neglected.

B. NVM-Based LUT and SB

1) STT-MRAM based Designs: A STT-MRAM based LUT has been proposed in [23]. The LUT uses conventional *Pass-Gate* (PG) based multiplexer structure in which an *Magnetic Tunnel Junction* (MTJ) cell is placed between ground and the multiplexer branches (i.e., at the leaves of multiplexer tree). Based on the LUT input signals, the corresponding multiplexer branch and its associated MTJ is selected and compared with a base resistance (which has a value between LRS and HRS values) using a sense amplifier (similar to that in Section II-A) at the output of LUT. Relative resistance of the selected MTJ cell and the base resistance determine the output voltage. The drawback of [23] is the sense amplifier which requires a sensing signal to be triggered after each change in LUT inputs. This is analogous to placing a latch at the output of each conventional LUT that demands a clock at each cycle, which prevents cascading the LUTs and/or designing combinational circuits.

A similar MRAM-based LUT with duplicated multiplexer structure is also proposed in [24]. Two MRAM cells, which correspond to a single bit, are programmed complementary in each multiplexer. Per each LUT input, a pair of MRAMs is selected and compared/sensed at the output of the structure to determine the stored logical value. The proposed sense amplifier has eliminated the aforementioned SRAM-like structure as well as the need for triggering signal (therefore it is hardened to radiations/particles) but the overhead of the duplicated structure and sense amplifier is significant. It should also be noted that since the relative resistance of STT-MRAM to NMOS transistor is low, any fluctuation in the resistance level of the cells can endanger a reliable sense operation in the aforementioned structures.

2) PCM-based Designs: A PCM-based 2×2 SB design employing 24 PCM cells has been proposed in [19]. A major shortcoming of the suggested structure is that some of the PCM cells are not properly placed in between two programming drivers. Therefore, the programming current required for such a cell should pass through another PCM cell which either may be hindered by the cell (if it was previously programmed to HRS) or can cause an unwanted write operation in that cell. In additin, 24 PCM cells occupy huge area with currently matured PCM technology.

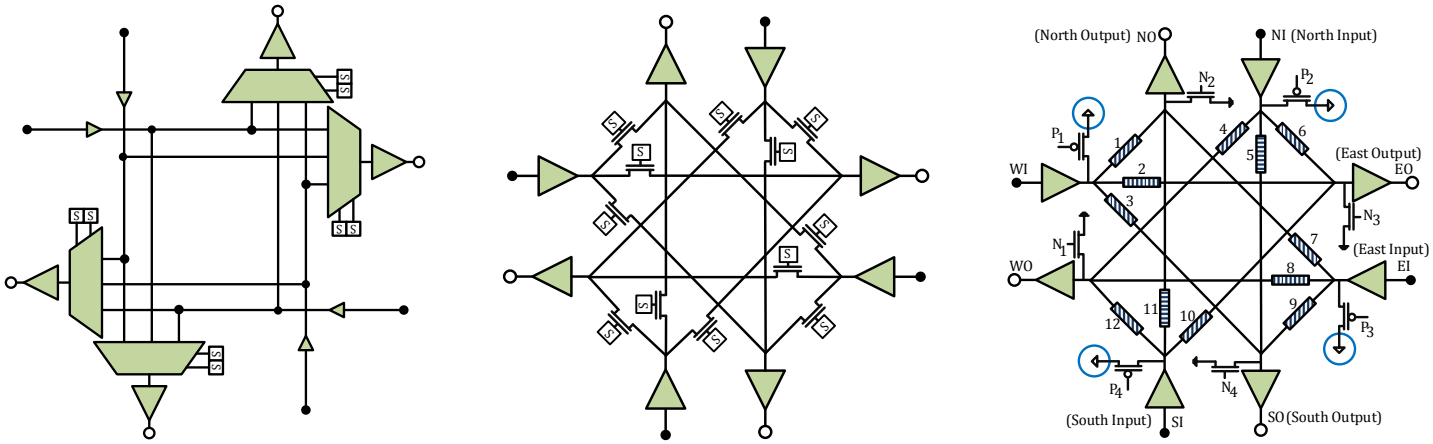
A fabricated PCM-based LUT prototype has been demonstrated in [31]. The LUT structure is based on a duplicated PG-based multiplexer. A configuration cell is represented by two complementary-programmed PCM cells, each placed in a different multiplexer branch (one connected to logical 1, and the complement cell is connected to 0). Based on the relative resistance of the PCM cells, either 0 or 1 is directed to the output. This architecture suffers from area overhead caused by duplicated multiplexer. Two programming transistors per each configuration bit is also used which are in the critical path of the LUT. More importantly, this architecture is not scalable for large-input LUTs (e.g., $K = 6$) since it necessitates inserting buffer in the multiplexer branches, which imposes infinite resistance in the pair of complementary paths and disrupts the voltage-dividing between the selected complementary paths.

PCM-based LUT and SB structures have been proposed in [6]. In the proposed LUT, each SRAM cell is replaced with a single PCM cell and the output is determined by exploiting voltage-dividing between the selected cell and a base resistance, R_{out}, at the LUT output. In such structure, as for [23], only the selected (based on LUT input) PCM cell consumes power. However, the voltage difference between the selected cell and the output node causes a significant power consumption. For example, when the selected cell is in HRS, the LUT output will be zero. This consumes a power equivalent to $\frac{{V_{dd}}^2}{HRS+R_{path}}$, which is higher than the SRAM-based counterpart. This could be alleviated by choosing high resistance cells, however, the delay will be affected adversely since the PCM cells are in the critical path. Similar to [31], this architecture is also not scalable to larger LUTs since inserting any buffer within the multiplexer structure forces its output to zero. Without the buffer, on the other hand, its delay will be increased exponentially. The SB proposed in [6] is based on a conventional 2×2 crossbar structure in which, similar to [18] and [33], each pass-gate is replaced with an RRAM. The LRS state of the RRAM within the path determines that the path is *on*. Conversely, the HRS RRAM turns off the corresponding path. In order to provide high performance, low LRS PCM cells have been utilized, which also have relatively low HRS due to the limited $\frac{HRS}{LRS}$ ratio of PCM. This causes significant leakage power consumption as detailed in Section III. To provide programmability, input buffers (drivers) of the SB are replaced by multiplexers to choose either the input signal or programming current. Therefore, further area is imposed, especially the multiplexer (added) transistors should be large because they reside in the routing paths. In addition, an output-buffered multiplexer will prevent the programming current to pass through. Buffer-less multiplexer, on the other hand, provides weak driving strength and thereby imposes delay overhead. Finally, no system-level programming scheme including the bitstream loading and timing control for neither the LUT nor the SB is suggested.

3) RRAM-based Designs: Generic RRAM-based multiplexer structure has been proposed in [21]. The proposed structure can be employed in either global (i.e., SB) or local (i.e., intra-cluster) routing network as interconnection building blocks, with any input size. In this study, all multiplexer transistors have been replaced with RRAMs that control the *on*/*off* state of routing paths by its LRS/HRS state. Since RRAM footprint is currently larger than a typical NMOS transistor, they can become area bottleneck in such a structure. In addition, all RRAM cells in the same multiplexer stage are programmed simultaneously. Thus, the programming transistors (which are stacked in series) should be upsized accordingly in order to tolerate the huge programming current for all RRAM cells and reduce the stacking effect, which leads to significant area overhead. Furthermore, architecture and controlling scheme of configuring the entire FPGA (i.e., loading the bitstream) has not been provided. Moreover, to implement a 2×2 crossbar SB, four 3-input multiplexer are needed (see Fig. 1(a)) that would require 28 programming transistors and 24 RRAM cells which, in overall, imposes large area footprint.

An FPGA with RRAM-based interconnection networks is suggested in [22] in which only a limited, prefabricated set of buffers are utilized. Therefore, it can improve further CMOS area in conjunction with eliminating the SRAM cells. The proposed SB is similar to [18], [33], and [6], i.e., a 2×2 crossbar in which each pass-gate and SRAM have been replaced by an RRAM. A system-level overview of programming structure has been also discussed in which the RRAMs are arranged as a memory-bank and the cells are selected and programmed consecutively. This individual programming scheme can lead to significant configuration time, especially taking into account the high programming time of RRAM, compared to SRAM. It also uses 2-NMOS/1-Resistor programming structure which has major limitations as explained in [38].

Finally, a one-level RRAM-based multiplexer structure that can be used in routing network of FPGAs has been proposed in [39] and [40]. To resolve the issues with 2-NMOS/1-Resistor programming structure, the authors propose a 4-Transistor/1-Resistor programming structure and use a transistor sharing scheme to reclaim the area. The proposed multiplexer uses a one-by-one programming scheme which excessively increases the FPGA configuration time, particularly, considering the high number of RRAM cells in such an all-RRAM block. Programming a multiplexer is well explained, but the overall configuration architecture has not been discussed; considering the large number of controlling signals, a considerable area and shifting time overhead (necessary for programming each RRAM) for the scan register is expected. Last but not the least, power efficiency of such all-RRAM structure is the key issue; in an N-input one-level multiplexer, $N-1$ of RRAM cells are configured to HRS and



(a) MUX-based SB using SRAM cells (b) Pass-Gate based SB using SRAM cells (c) Proposed RRAM-based SB

Fig. 1. Conventional SRAM-based SB designs (a and b) [37] and our proposed RRAM-based SB with the programming peripherals (c)

consume power due to the voltage difference between their terminals. For example, an SRAM-based 64-input connection block of 2-level multiplexer has 16 SRAMs and consumes $\simeq 16 \times 28.5nW = 456nW$, while in the RRAM-based equivalent, each of the utilized RRAM cells has a HRS of $23M\Omega$, resulting in $63 \times \frac{1V}{23M\Omega} = 2740nW$ which is $6\times$ of the SRAM-based equivalent.

III. Proposed RRAM-based Architecture

In this section, we first detail the structure of the proposed switch box. Next, we discuss its operation and programming modes and then elaborate how it can be programmed using modern FPGA configuration schemes. Afterwards, we detail the structure and the normal operation and programming modes of the proposed LUT. Finally, we demonstrate how the proposed LUT can be used as a part of the FPGA inside the programming scheme of the state-of-the-art FPGAs.

A. Proposed RRAM-based Switch Box

Multiplexer and *Pass-Gate based* (PG-based) designs are two commonly used switch box structures (introduced in [37]), which can be seen in Fig. 1(a) and Fig. 1(b), respectively. PG-based design affords simple architecture and higher performance since only one transistor resides between two input-output terminals. However, it contains higher number of configuration cells, thereby, higher area and power consumption. We borrow the switch box structure of Fig. 1(b) [37] and replace each pass-gate and its associated controlling SRAM cell of the conventional PG-based switch box with a single RRAM cell. Therefore, all SB elements are removed except the buffers, providing an efficient structure. The proposed switch box along with the programming circuitry can be seen in Fig. 1(c). Similar NVM-based SB structure has been also used in [6], [18], [22], [33]. Nevertheless, [18] and [33] use two programming transistors for each RRAM (i.e., total of 24 transistors) and [6] change the input drivers which can lead to area or delay overhead (see Section II-B for more details). Our programming structure, however, simply shares eight large transistors between 12 RRAM cells. Substantial impact of programming transistors in SB area will be detailed in Section IV. It should be noticed that the input buffers shown in Fig. 1(c) are abstract, i.e., they are the output buffers of adjacent SBs. That is, each unidirectional SB has only (four) output drivers, as in the multiplexer-based one shown in Fig. 1(a).

There is a high resistance cell in the paths that are supposed to be open, while there is a low resistance cell in the paths that should conduct. In the proposed SB architecture, all SRAM cells are eliminated and their corresponding high strength pass-gates (or transmission gates) are replaced with smaller number (i.e., eight) of programming transistors.

To illustrate the functionality of the proposed SB, we have depicted the detailed SB architecture in Fig. 2. Input WI drives the output EO through low resistance RRAM2. Since the input terminal of the other two RRAMs connected to EO (i.e., NI through RRAM6 and SI through RRAM10) does not drive any logical value, the strong zero passes properly through RRAM2 from WI. Input EI also drives the output NO by logical 1, through LRS-configured RRAM7. However, NO is also driven by WI, through the HRS-configured RRAM1. Therefore, a voltage dividing takes place in the input of the NO buffer, as $V_{NO} = \frac{HRS}{HRS+LRS} \simeq 1$. Thus, a weak logical 1 (denoted by 1^w) passes to the NO buffer. A similar scenario occurs for the SO buffer but in this case, the LRS RRAM is driven by logical 0, hence, $V_{SO} = \frac{LRS}{HRS+LRS} \simeq 0$. Apparently, in the case that EI and WI drive the same logical value, all of the mentioned nodes will be driven by strong 0 or 1.

We have designed a novel programming circuitry and adapted the FPGA programming scheme to fit the proposed SB. The proposed switch box is programmed by the transistors labelled P1 to P4 and N1 to N4, as shown in Fig. 1(c). The programming voltage is applied to either P1 to P4 transistors while the ground signal is enabled by N1 to N4 transistors depending on the intended cell. The selection of each combination of P1 to P4 and N1 to N4 creates the necessary path for programming of each RRAM cell. For instance, in order to program RRAM7, the transistors P3 and N2 are enabled. Notice that in each write operation, only one P-N pair is enabled; hence the drain node (encircled in Fig. 1(c)) of all P_i transistors is connected to a unique programming node to provide efficient overall configuration loading, as it will be discussed in Section III-D.

We verify the functionality of the programming structure using the RRAM model proposed in [28]. Consider that all RRAM cells of the SB in Fig. 1(c) have been initially in their HRS state. We validate

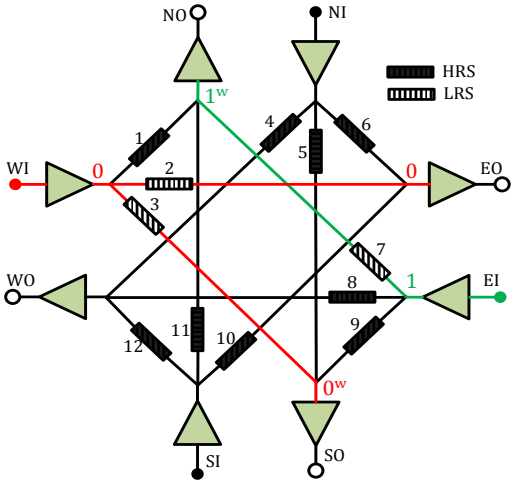


Fig. 2. Examining the functionality of the proposed SB

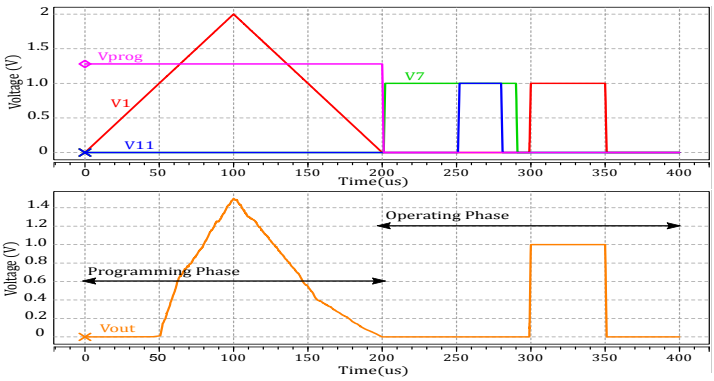


Fig. 3. Verifying the proposed programming structure of SB (top figure shows the applied voltages and the bottom figure shows the output voltage)

the functionality by programming (i.e., *set*) RRAM1 to LRS which connects the WI input to NO output. Since RRAM7 and RRAM11 are also connected to NO output (they connect the EI and SI to NO, respectively) we should also check whether they are still in HRS state. That is, they have not been undesirably programmed while programming R1. As it is demonstrated in Fig. 3, P1 and N2 are enabled by applying $V_{prog} = 1.3V$ to their gate voltage, and at the same time, the programming voltage V1 is applied to the source node of P1. Note that source node of all PMOS transistors are connected to the same node, but since only P1 is enabled, the programming current is expected to flow through P1. After $200\mu s$, the programming phase finishes and V_{prog} is cut-off. Right after, three pulses V1, V7 and V11 are simultaneously applied on, respectively, RRAM1, RRAM7, and RRAM11 through the corresponding input buffers (i.e., WI, EI, and SI). The output node (NO) is initially low and rises to one only when V1 rises (at $T = 300\mu s$). Rising V7 and V11, however, does not affect the output voltage which indicates that their path is correctly off. Here, two reliability challenges may arise. First, the programming current passing from P1 can sink to the ground through NMOS transistor of WI or the applied voltage may interfere with output voltage of WI. Second, an unbearable voltage may occur in the gate of output buffers which can cause them to breakdown. These challenges have been addressed in Section V-C.

It should be noted that during programming RRAM1, while only N2 is enabled, the programming current generated through P1 can also partially flow through other *leaky paths* to sink the ground using N2 (for instance, through the paths RRAM2→RRAM10→RRAM11 or RRAM3→RRAM9→RRAM7). Similar situation can occur in other programming scenarios which may aggressively *set* some of RRAM cells to LRS and create short paths. In addition, the already programmed (LRS) RRAMs may be *reset* back to the HRS state during the programming of subsequent cells. We examined the possibility of such issues by thoroughly investigating all programming scenarios (beginning with all RRAMs in the HRS state) and measuring the voltage across RRAM terminals. For a $V_{set} = 2.0V$ of the employed RRAMs (see Section IV-A for more details), a maximum voltage of $10^{-5}V$ on LRS and $1.2V$ on HRS cells is observed. Compared to $V_{reset} = 1.3V$, the voltage of $10^{-5}V$ is negligible to *reset* the already LRS cells. In addition, the $\Delta V = 1.2V$ on the terminals of HRS cells is smaller than $V_{set} = 2.0V$ (and even less than $V_{reset} = 1.3V$) to unwillingly *set* those cells. Analogously, during the erase operation (i.e., *reset* all RRAMs to HRS) of each RRAM using $V_{reset} = 1.3V$, the voltage across other RRAMs does not exceed $0.78V$. Therefore, no short paths (which need a voltage even higher than applied $1.3V$) can be created unintentionally. Furthermore, we examined the correct functionality of the proposed SB in all configuration scenarios by applying different pulses on the input terminals of *on* RRAMs and observing the corresponding output voltages. According to our experiments, for $1V$ applied pulses, a maximum of $2\times10^{-5}V$ difference between the input and output voltages is observed, which indicates that the proposed SB can properly pass the signals.

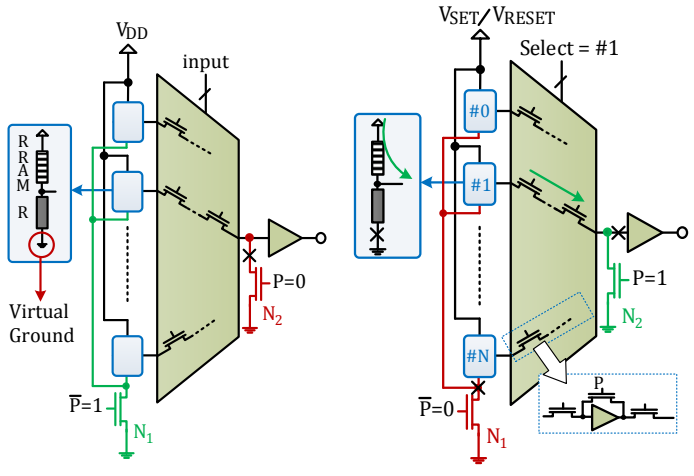
B. Proposed RRAM-based Look-Up Table

The proposed LUT cell along with the programming circuitry can be seen in Fig. 4. In our proposed LUT, the conventional SRAM cell is replaced by RRAM-resistor cells that act as a voltage-divider. In contrary with majority of similar work such as [13], [32], only one programmable RRAM is used in the proposed cell. The passive resistive element (i.e., R) can also be RRAM or any other CMOS compatible technologies, however, only one RRAM in each cell needs to be programmed. The passive resistor has a fixed value between RRAM low and high states, so the RRAM-resistor pair provides either an output voltage of zero or one depending on the RRAM value. Based on the cell structure (i.e., RRAM and passive R locations) shown in Fig. 4, to provide logical 1, RRAM should be programmed to LRS, resulting in $V = \frac{R}{R+LRS} \gg 0.5$. Similarly, a HRS RRAM provides $V = \frac{R}{R+HRS} \ll 0.5$.

While the RRAM and passive resistor locations are interchangeable in the proposed cell, we intentionally connect the RRAM to V_{DD} to provide more power saving opportunity as it will be discussed in Section IV-D. Although a voltage-divider scheme consumes leakage power, it is significantly smaller compared with leakage of an SRAM due to using high resistance RRAM cells. The multiplexer and the input signals are used to select the correct memory value as in SRAM-based LUTs. This value is then propagated to the output of the multiplexer through the transmission or pass-gate tree structure.

In the normal operation mode of the proposed LUT, $\bar{P} = 1$, so N_1 delivers the ground node to all the RRAM-based configuration cells. Due to the high resistance of RRAM and R elements, a minimum size NMOS can tolerate the passing leakage current through all configuration cells. In the programming mode, $\bar{P} = 0$ and N_1 cuts off. Otherwise, the applied programming current/voltage (i.e., V_{set} or V_{reset}) will flow to the ground by passing through all configuration cells. However, $\bar{P} = 0$ provides a theoretically infinite resistance to the cells. On the other hand, the *Select* input provides a ground path to the selected cell (cell #1 in Fig. 4(b)) through N_2. Therefore, the applied programming current only passes through and programs the desired RRAM (due to lower, i.e., non-infinite resistance within its path).

It is noteworthy that to provide voltage boosting and prevent exponential delay increase, the inclusion of intermediate buffers within the transistor tree structure of LUT is inevitable. These buffers, however, will prevent flowing the programming current from RRAM cells to ground through N_2. To resolve this issue, we add a parallel bypass transistor to these buffers, as shown in Fig. 4(b). During the programming phase, $P = 1$ enables this transistor (N_2) and allows programming current to flow through it. The size of this transistor can be smaller than the



(a) Normal operation (b) Programming mode

Fig. 4. Proposed LUT in different operating modes

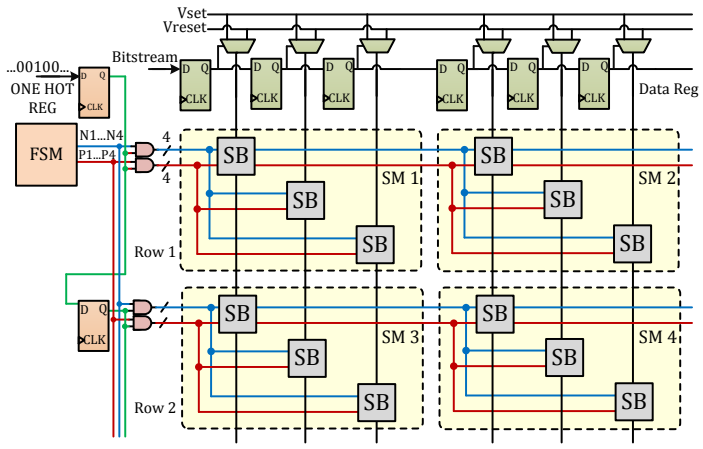


Fig. 5. Integrating the proposed SB in FPGA programming scheme

other transistors of the LUT since it does not reside in the critical path but it should be large enough to pass the programming current. The capacitance impact of these transistors is negligible and does not affect the LUT delay.

C. Discussion

As explained in [38], the 2-NMOS/1-RRAM programming structures suffer from (a) low programming current density due to the voltage drops across transistors, (b) increased transistors threshold, and thereby, reduced driving strength due to the body effect, (c) voltage drops in the driving inverters, and (d) inefficient sizing of transistors for the worst-case current (i.e., *reset* or *set* current). By exploiting unipolar RRAMs, in the proposed 1-PMOS/1-NMOS structure, the NMOS transistors are directly connected to ground (shown in Fig. 1(c)) which eliminates the driving inverters and corresponding voltage drop. In addition, both source and bulk of the NMOS transistors have the same voltage, i.e., $V_s = V_b = GND$; hence, body effect does not occur in NMOS transistors. On the other hand, although in 4-Transistor/1-RRAM structure each pair of PMOS-NMOS transistors is sized according to corresponding *set* and *reset* current, in our 1-PMOS/1-NMOS structure, only one (shared) pair of PMOS-NMOS is used and occupies less area. That is, $P_{max\{set,reset\}} + N_{max\{set,reset\}} < P_{set} + N_{set} + P_{reset} + N_{reset}$. In addition, as detailed in Section IV-A, for our utilized RRAM cell $\bar{V}_{max\{set,reset\}} = 2.0V$ and $I_{max\{set,reset\}} = 100\mu A$ for which a minimum size programming transistor is also adequate for the worst-case scenario. Eventually, as explained in Section IV-A, the shared encircled node of PMOS transistors in an SB should be switched between $V_{reset} = 1.3V$ and $V_{set} = 2.0V$ to *reset* and *set* the RRAM. The required multiplexer is shared between a column of SBs (see Fig. 5), thus it can be upsized by $20\times$ to offset the voltage drop. In addition, since the programming I/O transistors can operate with up to 3.0V and maximum programming voltage is 2.0V, the applied programming voltage can be boosted to conserve the voltage drop.

D. Overall Programming Circuitry

In emerging NVM-based FPGA architectures, loading the configuration bits into NVM cells is one of the major design issues as opposed to the conventional SRAM-based FPGAs, where loading is performed serially [41]. In the proposed architecture, we present a write circuitry to load the configuration bits from registers to the RRAM cells as depicted in Fig. 5. Once the register containing the bitstream data (i.e., *Data Reg*) has been serially loaded, the operation to transfer the data in RRAM configuration cells begins. For switch boxes, a finite state machine generates a sequence to enable the previously explained N1 to N4 and P1 to P4 signals in an appropriate order. At the same time, both *reset* and *set* pulses are sent to their respective lines. Based on the configuration bits, the data register determines which of these pulses are

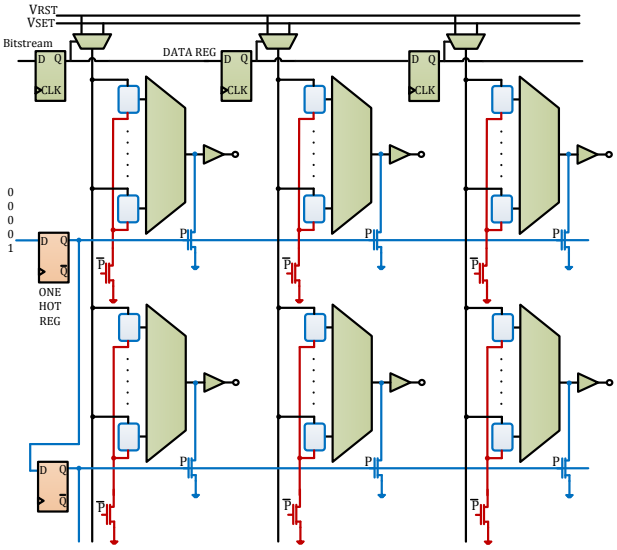


Fig. 6. Integrating the proposed LUT in FPGA programming scheme

fed into the program lines of each selected switch box. For instance, in the first programming cycle, the FSM enables P1 and N2. Thus, RRAM1 is selected in all switch boxes within a *Switch Matrix* (SM) row, i.e., switch boxes in SM1 and SM2 in Fig. 5. Hence, it takes 12 programming periods (due to the total of 12 cells in the SB) to program all switch boxes in a row. Notice that all P_i transistors within a switch box are connected to a unique programming voltage node, however, only the right one is activated based on the FSM output. Analogously, in the second programming cycle, the FSM enables P1 and N3, so RRAM2 will be selected. Simultaneously, the bitstream frame corresponding to all RRAM2 in the first SM row is loaded to Data Reg. After programming all 12 cells in the first SM row, *One-Hot Reg* selects the second SM row and the same programming scenario repeats.

The proposed programming scheme works also when *set* and *reset* duration are not necessarily the same. Generally, after loading each bitstream frame into the data register which takes $T_{frame} = C \times {}^{W}/_{2} \times T_{DFFshift}$ (see Section IV-A), V_{set} and V_{reset} pulses are applied for durations T_{set} and T_{reset}, respectively. Therefore, assuming $T_{set} > T_{reset}$, for the *set* voltage, a simple periodic pulse with the period of $T_{V_{set}} = T_{frame} + T_{set}$ and the duty cycle (i.e., percentage of a cycle in which signal is high) of $D_{set} = \frac{T_{set}}{T_{frame}+T_{set}}$ should be applied. The *reset* pulses are also synchronized with *set* pulses, i.e., $T_{V_{reset}} = T_{frame} + T_{set}$. However, the duty cycle of *reset* voltages is determined by *reset* duration, i.e., $D_{reset} = \frac{T_{reset}}{T_{frame}+T_{set}}$. This means that both *set* and *reset* pulses are disabled (zero) during loading the bitstream and the *set* pulse is active during the remaining programming cycle. Nevertheless, the *reset* pulses become zero after T_{reset}.

Note that the programming current of RRAMs can be controlled by either adjusting the drain voltage or gate bias voltage, i.e., P_i. However, since the P_i signals are shared between all SBs in the same row and one RRAM can be programmed to LRS while the other needs to be programmed to HRS, controlling by the gate bias will require different P_i values in such a scenario. Therefore, we adjust different V_{set} and V_{reset} voltages for the constant P_i for both *set* and *reset* operations.

The configuration operation for the LUT takes place by consecutively writing each of the RRAM cells. The scheme for this purpose is depicted in Fig. 6. The scheme is similar to that of switch boxes except the path to the ground signal plays the critical role in LUTs. The added transistor before the output buffer can provide the ground signal at the end of the multiplexer tree. This allows having only one of such transistors for each LUT. To program a cell, (a) P is set to active, (b) the appropriate pulse is selected using the serially shifted register (data register), and (c) the path to the newly created ground signal is activated using multiplexer inputs. Notice that all original virtual grounds should be cut off using the

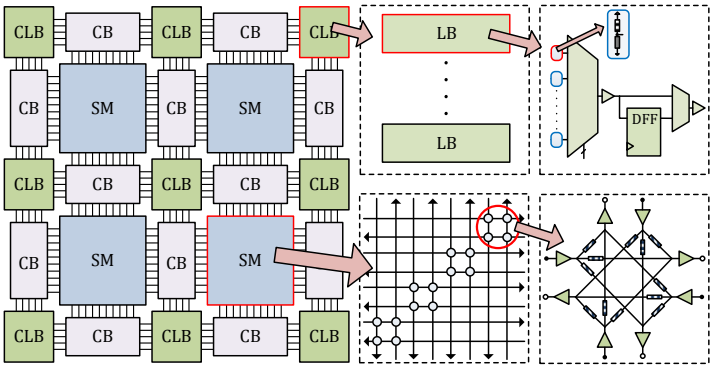


Fig. 7. Proposed designs inside island-style FPGA architecture

$\bar{P}$ transistors in order to avoid undesired write operations. This process has to be repeated for all RRAM resistor pairs in each LUT. Several LUTs can be programmed at the same time as long as they are in the same frame and use the data register at the same time.

IV. Evaluation

In this section, we first compare the configuration time of the proposed RRAM-based FPGA with the SRAM-based equivalent by mapping and routing the MCNC benchmarks. Afterwards, we compare the proposed LUT/SB building blocks with state-of-the-art blocks as well as we compare the RRAM-based FPGA built upon the proposed designs with FPGAs based on (a) a STT-MRAM-based non-volatile configuration cell [14], (b) PCM-based non-volatile cells [2], (c) PCM-based LUT and SB designs [6], (d) a *perpendicular Magnetic Tunnel Junction* (pMTJ)-based LUT [23], and (e) a one-level RRAM-based multiplexer for routing network [39]. Our evaluation also includes finding optimum parameters for the proposed designs. The experimental setup and toolsets to estimate each metric are detailed in the corresponding subsection.

A. Programming Time

The routing and logic blocks can be programmed simultaneously, however, we examine them separately for the sake of simplicity. Let's assume that, as shown in Fig. 7, the target device has an island-style architecture as most of the modern FPGAs such as Xilinx Virtex Series [42] and it consists of R rows and C columns of switch matrices and *Configurable Logic Blocks* (CLBs). We also assume the unidirectional routing channel width is W, thus there are $^{W}/_{2}$ switch boxes in each SM (see Fig. 7).

As detailed in Section III-D, programming each row of SM begins with loading the bitstream frame into Data Reg (see Fig. 5). The total number of flip-flops in Data Reg can be obtained by multiplying the number of SM columns by the number of SBs inside each SM, i.e., $L_{DataReg} = C \times {}^{W}/_{2}$. RRAM1 in all SBs is simultaneously programmed, which takes $max\{T_{set}, T_{reset}\} = T_{set}$. This procedure repeats for the remaining 11 RRAMs in the first SM row, and then, repeats for the whole *R* device rows. Accordingly, the total SM programming can be calculated according to Equation 1 in which $T_{DFFshift}$ is the shift delay of each flip-flop (e.g., 0.3ns).

$$T_{SM,prog} = R \times (12 \times (C \times {}^{W}/_{2} \times T_{DFFshift} + T_{set})) \quad (1)$$

As explained in Section III-A, programming SBs requires to initially erase/*reset* them which requires erase time of $T_{SM,erase}$ as Equation 2.

$$T_{SM,erase} = R \times (12 \times (C \times {}^{W}/_{2} \times T_{DFFshift} + T_{reset})) \quad (2)$$

Similarly, supposing there are N_{LUT} number of K-input LUTs inside every CLB (which are arranged in a $R \times C$ array), the total programming time of logic blocks (T_{LB}) can be computed according to Equation 3.

$$T_{LB} = R \times (2^{K} \times (N \times C \times T_{DFFshift} + T_{set})) \quad (3)$$

We compare the programming time of the proposed architecture with that of conventional SRAM-based FPGAs. In this regard, we place and route the 20 largest MCNC benchmarks [30] on minimum size FPGAs using VPR 7.0 toolset [29] to find the array size (i.e., R and C parameters). MCNC benchmarks include both combinational and sequential circuits and have been widely used in, especially, FPGA academic research. These circuits are also provided in *Berkeley Logic Interchange Format* (BLIF) in the VPR repository. The name and number of LUTs of these benchmarks are provided in Table II. In the experiments, we have targeted 4-input LUTs, i.e., $K = 4$. The cluster size of $N = 10$ (thereby 10 LUTs within each cluster) has been chosen as it is considered in most recent studies [29]. Notice that the FPGA flow (map, place, and route) has nothing to do with the underlying technology or programming scheme. Therefore, no modification in the CAD tool is required for this purpose. The architectural setup used in VPR is reported in Table I. Unidirectional SBs arranged in a *Subset* topology, as shown in Fig. 7, has been used. These parameters along with area and timing parameters (see Section IV-B and Section IV-C) are wrapped an architecture file with $X = 16$ (see Table I) provided in VPR power repository. Table II provides the number of 4-input LUTs in netlist, and size (R and C) and channel width (W) of the mapped (i.e., placed and routed) device for each circuit.

Plenty of RRAM devices with various electrical and resistive attributes have been devised [11]. While some previous studies have used RRAM with $LRS = 1K\Omega$ and $HRS = 1G\Omega$ [22] which pretty fits in our proposed designs due to its high $^{HRS}/_{LRS}$ ratio, such cells are not mature enough at the moment due to their large footprint and programming current (e.g., $3\mu m \times 3\mu m$ [12]). In our experiments, we use RRAM cells with $1G\Omega$ and $10K\Omega$ HRS and LRS resistances, respectively and $1\mu m^2$ footprint which has been properly fabricated in [43]. As it has been demonstrated in [43], the LRS resistance of our employed RRAM can be adjusted by controlling the bias condition. To achieve low LRS values, higher *set* and *reset* currents (voltages) are required. For $10K\Omega$, measured *reset* and *set* currents are $100\mu A$ and $\sim 3\mu A$, respectively. An average 2.0V and 1.3V *set* and *reset* voltages are reported. It should be noticed that RRAM cells necessarily do not show Ohmic behaviour, especially in the HRS state [11], [43]. This means that the programming voltage of RRAM is not simply corresponded to its programming current (and vice versa), i.e., $I \neq \frac{V}{R}$ or $R \neq \frac{V}{I}$. Since the programming speed of the employed RRAM has not been reported, we pessimistically assume $T_{set} = 50ns$ and $T_{reset} = 10ns$ which are large-enough comparing with similar devices [11]. Finally, we set $T_{DFFshift} = 0.24ns$ by synthesizing a shift register using Synopsys Design Compiler [27] and Nangate 45nm Standard Cell Library [44] and – conservatively – choosing minimum size flip-flop cells. Notice that operating frequency of a shift register is independent of its width (i.e., $T_{DFFshift} = T_{clk\_to\_Q} + T_{setup}$).

Fig. 8 compares the programming time (as sum of $T_{SM} = T_{SM,prog} + T_{SM,erase}$ and T_{LB}) of the FPGA based on the proposed RRAM-based LUTs and SBs with the conventional SRAM-based FPGAs. For the SRAM-based FPGA, the total number of configuration bits is calculated by Equation 4 and a maximum programming rate of $371.4^{MB}/_{s}$ (i.e.,

TABLE I
VPR architectural parameters

Parameter	Definition	Value
N	Cluster size	10
K	LUT size	4
I	Routing to cluster input	22
L	Wire segment length	1
Fs	Wire branches in each switch box	3
Fcin	Ratio of channel tracks connected to connection block	0.2
Fcout	Ratio of channel tracks a LB output connects to	0.1
X	Intra-cluster multiplexer size	16

TABLE II
NUMBER OF 4-INPUT LUTS, DEVICE SIZE, AND CHANNEL WIDTH (W) OF MCNC BENCHMARKS

Benchmark	LUT4, Size, W	Benchmark	LUT4, Size, W	Benchmark	LUT4, Size, W
alu4	861, 11, 88	dsip	679, 14, 132	s298	666, 10, 82
apex2	993, 13, 110	elliptic	1905, 15, 130	s38417	2294, 19, 194
apex4	836, 11, 110	ex1010	2897, 22, 180	s385841	2116, 20, 130
bigkey	567, 14, 110	ex5p	614, 10, 100	seq	916, 12, 114
clma	3206, 22, 160	frisc	1752, 16, 138	spla	1872, 17, 138
des	815, 16, 130	misex3	799, 11, 100	tseng	706, 10, 80
diffeq	698, 10, 80	pdc	2443, 20, 154	average	1382, 15, 75

$\sim 0.337ns$ per SRAM) is assumed [45].

$$N_{SRAM} = (R \times C \times {}^{W}/_{2} \times 12) + (R \times C \times N_{LUT} \times 2^{K}) \quad (4)$$

While the programming time of an individual RRAM cell is substantially higher than an SRAM cell (i.e., $\frac{50ns}{0.337ns} = 148\times$), due to the efficient programming circuitry of the proposed designs, the configuration time is increased by only 81%. This is 83× faster compared with individually programming of RRAM cells (i.e., 50ns per cell) as in [22]. These numbers can be obtained by using the device size and channel width of each benchmark presented in Table II. For instance, for *pdc* benchmark with a 20×20 array size and channel width of 106, configuration time of the logic and routing resources will be $31.4\mu s$ and $63.5\mu s + 73.1\mu s$ (erase and program), respectively, with a total of $167.9\mu s$. On the other hand, it contains 318,400 configuration cells which results in $318,400 \times 0.337ns = 107.2\mu s$ and $318,400 \times 50ns = 15,920\mu s$ (i.e., 95× of the proposed method) for, respectively, conventional SRAM-based shifting and RRAM-based one-by-one programming scheme [22]. In larger designs such as *ex1010*, the delay overhead of shifting a frame (into Data Reg) is well compensated by simultaneous programming a large fraction of cells, hence such designs have better relative programming time. For parameters of a moderate-size commercial Virtex-II consisting of 64×64 CLB array, $W = 192$ and $N = 4$ [5], the configuration time in the proposed design is only 39.3% larger than that of the equivalent SRAM-based FPGA.

B. Area

Area of the proposed LUT is reduced by eliminating the SRAM cells and placing the RRAM cells atop CMOS. However, a few extra transistors, including N_1 and N_2 and bypass transistors for intermediate buffers (in overall, eight buffers right after the third stage of LUT) are added. Due to the low programming current ($I_{reset} = 100\mu A$), these transistors can be even selected to have minimum width, i.e., 90nm in the 45nm process. Area of the proposed SB is reduced by removing the SRAM cells and associated pass-gates. However, as shown in Fig. 1(c), eight programming transistors has been added in the structure of SB. The P_i transistors are directly connected to the programming voltage source and should provide an average 2.0V (up to maximum 2.5V) *set* voltage for the employed RRAMs; thus, larger transistor with thick gate oxide should be utilized. As reported in [38], I/O transistors of a commercial 45nm technology with $\frac{W}{L} = \frac{320nm}{270nm}$ can tolerate V_{GS} and V_{DS} of 2.5V.

Since the area footprint of a design depends on the geometrical shape of its layout, we use *minimum width transistor* area model which follows the dimensionless formula $Area(x) = 0.447 + 0.128x + 0.391\sqrt{x}$ [46] in which x is the ratio of transistor strength (size) to the minimum

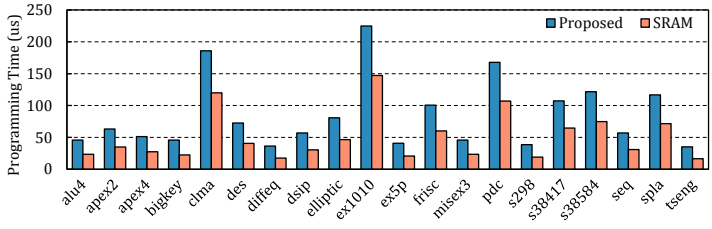


Fig. 8. Comparing the programming time of the proposed RRAM-based FPGA and SRAM-based counterparts

TABLE III
VPR EXPERIMENTAL PARAMETERS (AREA IS DIMENSIONLESS AS DETAILED IN SECTION IV-B)

Parameter	Value	Parameter	Value
Switch type	Buffered	SRAM-based SB area	133.7
Segment length	1	Proposed SB area	43.9
SB buffer stage ratio	5×	SRAM-based LUT area	233.5
SRAM area	6.0	Proposed LUT area	137.5
SB pass-gate size	10×	Prog. transistors size	$^{320nm}/_{270nm}$

width transistor in the same technology. It gives the area as a factor of area of minimum width transistor (which is typically $60L^2$) and has been adjusted for 65nm and smaller technologies. Further details can be found in [46]. Table III summarizes the area parameters used in VPR experiments. Both SRAM- and RRAM-based LUTs employed in the experiments are based on transmission-gates and include isolating buffers (inverter) after the configuration cells to provide higher drive strength and reliability. During programming the proposed LUT, these buffers can be bypassed similar to the scheme explained in Section III-B. Area of programming transistors (assumed as 2× of standard transistors [39]) is included in both the proposed SB and LUT designs.

Fig. 9 compares the area (as well as the delay and power which will be discussed in Sec. IV-C and Sec. IV-D) of the proposed LUT and SB designs with the aforementioned studies. The results reported in this figure will be later used to compare the efficiency of the FPGAs built upon these blocks. All results are normalized to that of the SRAM-based block. Actual quantities of SRAM- and the proposed RRAM-based designs have been represented in Table III. It should be noted that [23] has proposed a NVM-based LUT, but lacks a NVM-based SB. Also, the study in [39] proposes a NVM-based SB and lacks a LUT design. Therefore, for the sake of holistic comparison in the subsequent sections, we assume SRAM-based designs for those lacking circuits. Area of both [2] and [14] has been slightly increased due to larger cell areas (8.9 and 7.0 minimum width transistor area, compared to 6.0 for SRAM). Area of one-level RRAM-based SB [39] is also higher than the area of the proposed SB since [39] employs two programming transistors per each RRAM cell with an additional shared pair. Area of LUT presented in [6] is slightly smaller than our proposed LUT since the proposed LUT comprises a set of bypass transistors for internal buffers, as explained in Section III-B. Eventually, the MTJ-based LUT structure [23] has the minimum area because it eliminates the input buffers.

Fig. 10 compares the area of the proposed RRAM-based FPGA and the related studies over the MCNC benchmarks using the parameters of Table III and Fig. 10. The results reveal that, compared to SRAM-based FPGAs, total area is reduced by 59.4%, on average (a detailed comparison can be found in Table VII). This improvement converges with [5] which indicates that if all configuration cells and SBs are moved to top layers, the area can be reduced by ∼57%. Notice that the area of RRAMs has been set to zero since they can be stacked atop CMOS layers. In addition, the area of the RRAM layer is not larger than the CMOS layer (i.e., RRAMs are not bottleneck). The total area of 12 RRAM cell is $12\mu m^2$ [43]. On the other hand, based on the Nangate Cell Library [44], a 5× buffer occupies $\sim 2.5\mu m^2$, therefore the area corresponding to four buffers of the proposed SB is $10\mu m^2$. In addition, the area of the eight programming transistors would be larger than $2\mu m^2$,

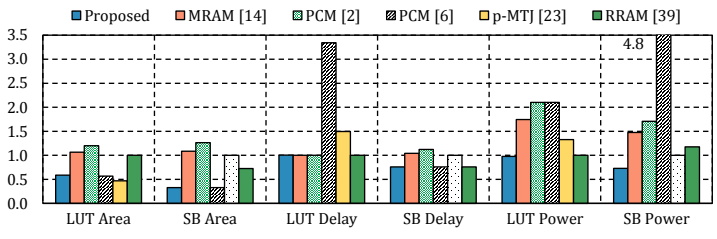


Fig. 9. Comparison of the proposed architecture with the related work in terms of area, delay, and power

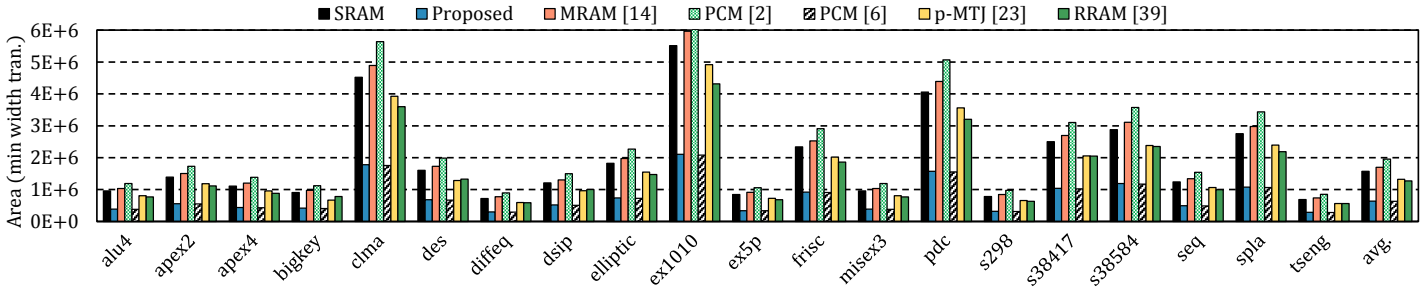


Fig. 10. Comparing the total area between the proposed architecture and related work

making the aggregate area of the CMOS part larger than that of RRAM.

Programming Peripheral: The programming circuitry of the proposed architecture comprises $C \times \frac{W}{2} + R$ and $N \times C + R$ flip-flops (including data and one-hot ones, respectively) for SBs and LUTs configuration, respectively (see Fig. 5 and Fig. 6). Also, $C \times \frac{W}{2} + 8 \times R$ and $N \times C + R$ column and row drivers are required, respectively. For SRAM-based FPGAs, a memory-array-like structure for configuration cells can be considered, as in [41], in which N_{SRAM} number of configuration bits are arranged as a $\sqrt{N_{SRAM}} \times \sqrt{N_{SRAM}}$ array. The number of SRAMs can be obtained from Equation 4. This structure has $\sqrt{N_{SRAM}} + \sqrt{N_{SRAM}}$ (column and row) flip-flops, and $\sqrt{N_{SRAM}} + 2\sqrt{N_{SRAM}}$ (for Wordline and Bitline/$\overline{\text{Bitline}}$) drivers. According to Nangate Open Cell Library, the area of a flip-flop and maximum-size buffer is $5\mu m^2$ and $13\mu m^2$, respectively. Using this information with the C, R, and W values of Table II reveals that the programming peripheral of the proposed architecture, on average, occupies 19.3% less area compared with SRAM-based architecture.

C. Delay

The total delay of an FPGA mapped design is composed of logic and routing delay, in which the latter includes the switch box and its associated routing wire (i.e., wire driven by the switch box) delays. The proposed RRAM-based LUT does not affect the delay since it employs the same multiplexer structure of an SRAM-based LUT. In addition, as for the SRAM-based LUTs, the RRAM-based LUT includes an isolating buffer between the configuration cells and the multiplexer input to provide robustness by preventing rush currents from the multiplexer to the cells (which can cause bit flip in the SRAM cells) and also providing higher drive strength for the multiplexer.

Delay of the LUT and SB designs is measured by using HSPICE circuit level simulation and employing High-Performance 45nm *Predictive Technology Model* (PTM) [47]. To model the RRAM cells, we exploit RRAM model presented in [28] and change the LRS and HRS values based on the parameters of the employed RRAM [43]. Sizing of switch box buffer and transistor have been reported in Table III. The resistive and capacitive characteristics of metal layers for local and global routing of default 22nm process are first extracted using COFFE tool [46] with the architectural parameters of Table I . Afterwards, we scaled the extracted values to 45nm using the scaling parameters obtained from [48], i.e., $\frac{R_{45nm}}{R_{22nm}} = (\frac{22}{45})^2$ and $\frac{C_{45nm}}{C_{22nm}} = (\frac{45}{22})^{0.15}$. In the experiments, we also take into account the role of area reduction in the wire length and its resistance and capacitance [5]. To this end, we correlate the area and wire length by $l = \sqrt{\frac{\bar{s}}{s}}$ in which l is the wire length scaling/reduction factor and $\bar{s}$ and s are the new and original area, respectively. In other words, if the area is scaled by α, then the device dimensions, and hence, wire lengths and the corresponding resistance and capacitance will be scaled by $\sqrt{\alpha}$. As observed in Section IV-B, RRAM-based FPGA reduces the area by 59.4%, hence, $\alpha = 0.41$. It is noteworthy that the exploited RRAM model [28] takes the inherent parasitic capacitance between electrodes of an RRAM cell into account. The parasitic capacitance can be a delay bottleneck in an all-RRAM design such as [39] and [40] wherein all RRAM cells are connected to a single output node. However, the parasitic contact resistance of the electrodes is negligible compared to resistance of the RRAM cell itself [28].

TABLE IV
HSPICE DELAY MEASUREMENTS PARAMETERS AND RESULTS

Parameter	Value	Parameter	Value
SRAM routing res.	50.9Ω	SRAM LUT delay (avg)	$102ps$
RRAM routing res.	32.6Ω	RRAM LUT delay (avg)	$103ps$
SRAM routing cap.	$11.8fF$	SRAM SB + wire delay	$99ps$
RRAM routing cap.	$7.6fF$	RRAM SB + wire delay	$75ps$

HSPICE reports for the RRAM- and SRAM-based LUT and SB delay are summarized in Table IV. As reported in this table, 24.2% improvement in SB delay has been obtained. This improvement arises from scaled area of RRAM-based FPGA, and thereby, reduced wire resistance and capacitance. It is noteworthy that a minimum width NMOS transistor resistance is $\sim 9K\Omega$ [29]. Since $10\times$ width transistors are used for SRAM-based SB, the resistance of pass-gate would be $\sim 10\times$ smaller than that of employed RRAM cell with $R_{LRS} = 10K\Omega$. Pass-gate based switch, however, considerably increases the delay since the NMOS transistor cannot appropriately pass the logical one. On the other hand, the transmission-gate based SB increases the area significantly. Therefore, to boost the baseline SRAM-based SB, we used SRAM cells with $V_{DD} = 1.2V$. For longer wire segments, e.g., $L = 4$, the delay could further be decreased since in larger segment lengths, the wire delay becomes dominant, which is reduced in the proposed designs. The normalized delay of the previous studies has been provided in Fig. 9. The delay of the proposed LUT, [14], and [2] has almost remained intact with regard to SRAM-based LUT because of using the same multiplexer structure and providing the required voltage level at multiplexer inputs (i.e., configuration cells). Delay of the LUT in [6], however, has been increased because of the passive output-resistance which is used for voltage-dividing and also resides within its critical path. Particularly, the required voltage/current passes through the configuration PCM and is divided at the multiplexer output. Large HRS of PCM cells significantly increases the delay. On the other hand, similar to the proposed SB, [6] improves the SB delay by replacing low-resistance PCM cells instead of pass-gates and having smaller wire length due to device shrinking. Delay of the pMTJ-based LUT structure is determined by the interval in which the sense signal should be enabled to have a successful read operation. This delay could be further increased by adding a safety margin for sensing operation. The increase in SB delay of [2] and [14] stems from increased FPGA area, and subsequently, wire delay. Experimental results over the MCNC benchmark are shown in Fig. 11. On average, the proposed design reduces the total delay by 20.1% compared to SRAM-based FPGAs (please refer to Table VII for detailed comparison). It is noteworthy that while [6] could afford a similar area improvement to the proposed architecture, it degrades the average performance by 19.6%.

D. Power Consumption

1) Cell Structure: In contrary to the SRAM-based LUTs in which power consumption of an SRAM cell does not depend on its holding value, the resistive state of the RRAM cell (i.e., HRS or LRS) and the percentage of time in which a cell contains a specific configuration

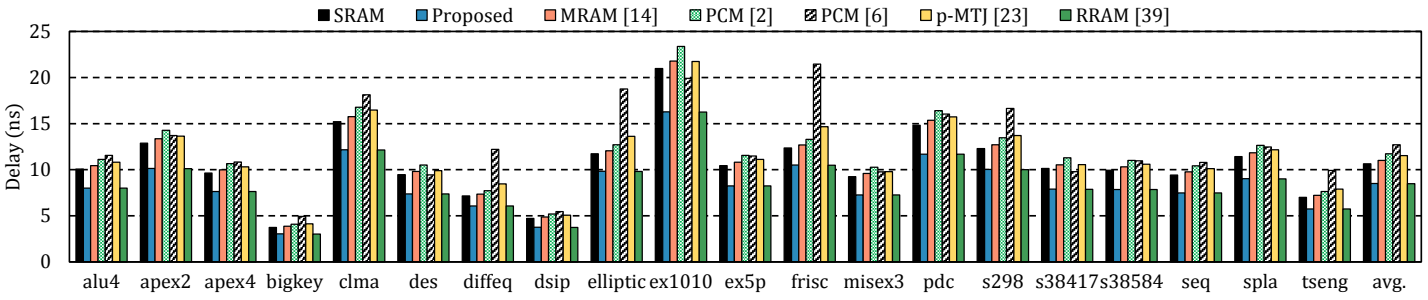


Fig. 11. Comparing the total delay between the proposed architecture and related work

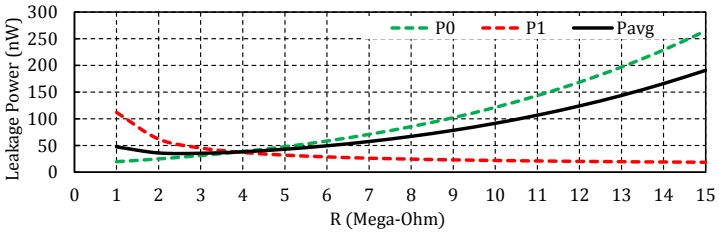


Fig. 12. Calibrating the optimum R value to minimize the cell power

play a substantial role in determining the overall power consumption of the proposed LUTs. Therefore, we first attempt to find the optimum cell structure and the passive resistor value, i.e., R. For this end, analyzing the configuration bits of MCNC benchmarks revealed that about 70% of the LUT configuration cells hold zero and the remaining is configured to one. Table VI illustrates the structures and the corresponding leakage power wherein the first structure consumes less power. Notice that $LRS \ll R \ll HRS$, thus $R + HRS$ and $R + LRS$ are simplified to HRS and R, respectively.

2) Optimum R: While specifying R to its maximum allowable value (i.e., $R = {}^{HRS}/_{2}$ which provides an effective one and a logical zero $\simeq \frac{1}{3}$) makes the voltage-divider power efficient, it causes the associated isolating buffer to be always *on* which consequently will consume significant power. Indeed, the output logical zero of voltage-divider part should be considerably smaller than the threshold voltage ($V_{th} \simeq 0.3V$) of associated buffer to prohibit significant leakage power. Analogously, logical one of the structure should be larger than $V_{DD} - V_{th}$. Moreover, it provides weak drive strength to the multiplexer tree. Therefore, we carry out HSPICE simulations investigating the whole R space such that $P_{avg} = 0.7P_0 + 0.3P_1$ is minimized. As demonstrated in Fig. 12, iterating with $10M\Omega$ steps suggests that $R = 30M\Omega$ provides the best power efficiency. The power includes the power of resistive structure and the associated isolating buffer.

3) Estimation: The power consumption of an RRAM inside the proposed switch box depends on the voltage difference on its terminals. As shown in Fig. 2, for example, RRAM2, RRAM3, and RRAM7 have zero leakage power because there is no voltage difference between their terminals. However, RRAM1 and RRAM9 consume the maximum power due to $\Delta V = 1V$ on their terminal. Hence, in order to accurately estimate the static power of each benchmark, we obtain the average signal *probability* of the wires by exploiting ACE 2.0 activity estimator [49]. Accordingly, the probability of voltage difference between the RRAM cells can be obtained. In order to estimate the dynamic power of each benchmark, we first calculate its average signal *activity* using the ACE tool. Next, we conduct HSPICE simulations to obtain the dynamic

TABLE V
STATIC POWER PARAMETERS OBTAINED BY HSPICE SIMULATIONS

Design	Power (μW)	Design	Power (μW)
RRAM LUT (all 0)	0.82	SRAM LUT (all 0)	0.87
RRAM LUT (all 1)	1.06	SRAM LUT (all 1)	1.01
RRAM SB (avg)	0.78	SRAM SB	1.07

TABLE VI
POWER CONSUMPTION OF DIFFERENT RRAM-BASED CELL STRUCTURES

Cell	Config 0	Config 1	$\sim P_0$	$\sim P_1$	$\sim P_{avg}$
	$\frac{R}{R+HRS}$	$\frac{R}{R+LRS}$	$\frac{1}{HRS}$	$\frac{1}{R}$	$\frac{0.7}{HRS}+\frac{0.3}{R}$
	$\frac{LRS}{LRS+R}$	$\frac{HRS}{HRS+R}$	$\frac{1}{R}$	$\frac{1}{HRS}$	$\frac{0.7}{R}+\frac{0.3}{HRS}$

TABLE VII
OVERALL COMPARISON BETWEEN THE PROPOSED ARCHITECTURE AND RELATED WORK

	Proposed	SRAM	MRAM [14]	PCM [2]	PCM [6]	p-MTJ [23]	RRAM [39]
Area	1.00×	2.46×	2.66×	3.06×	0.98×	2.07×	1.99×
Delay	1.00×	1.25×	1.29×	1.38×	1.50×	1.36×	1.00×
Power	1.00×	1.32×	1.99×	2.31×	5.60×	1.38×	1.51×
Power-Delay	1.00×	1.65×	2.57×	3.19×	8.37×	1.88×	1.51×

power of a single switch box using the wire resistance and capacitance represented in Table IV. Initially, we assume an activity factor $\alpha = 1$ and frequency of 100MHz, and also assume that one output of the switch boxes is active (i.e., are switching). Table V summarizes the HSPICE static power results for individual designs. Dynamic power of SRAM- and RRAM-based SBs are, respectively, $1.66\mu W$ and $0.73\mu W$. Note that the dynamic power of logic elements is equal in both the RRAM- and SRAM-based designs since the proposed LUT does not affect the original multiplexer structure and, additionally, the configuration cells have no impact on dynamic power consumption. The reduction in dynamic power of the proposed SB is due to the smaller wire capacitance of the RRAM cells compared with the large pass-gates of the SRAM-based SB.

The relative power consumption of the designs proposed in the related work can also be obtained from Fig. 9. According to this figure, both [14] and [2] increase the LUT and SB power compared with SRAM-based design because of adding the (large) sensing transistor within their SRAM-like sensing circuitry which adds to the static power. Especially, [2] uses a PMOS as sensing transistor which required higher channel width results in higher area and leakage power. On the other hand, experiments revealed that [6] could improve the LUT power (compared with our proposed architecture) by tuning the resistor in its structure. Nevertheless, it would increase the LUT delay intensively as discussed in Section II. Therefore, optimum parameters that minimize its power-delay product are used in our simulations. An advantage of the LUT proposed in [6] is that only the selected cell consumes power. Notice that our results are in contrast with that in the original paper of [2] as this study has reported $1.2nW$ leakage power for the entire 4-input LUT. This value, however, is even smaller than the power of a minimum-size inverter [44]. As for the LUT, the SB power of [14] and [2] is increased due to their increased cell power. The SB power of [6] is increased due to using low resistance PCM cells which consume significant power when a $\Delta V = 1V$ occurs on their terminal. As discussed in Section II, the NVM cells of the LUT proposed in [23] do not consume leakage

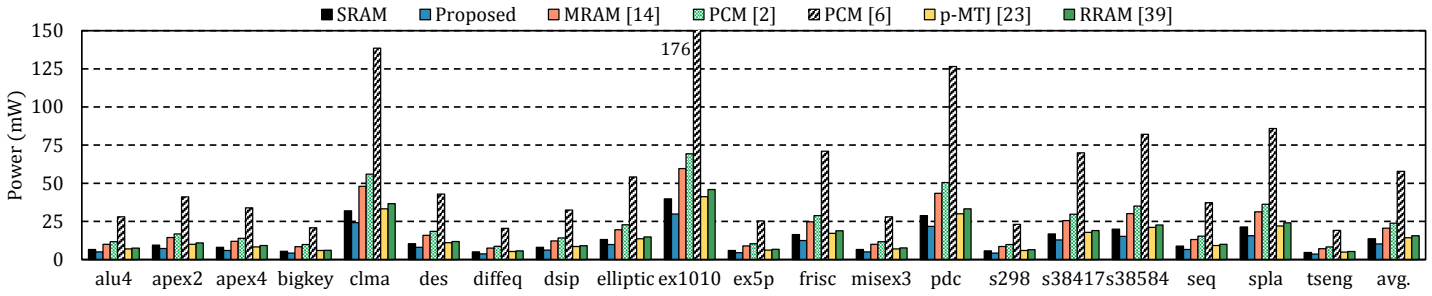


Fig. 13. Comparing the total power between the proposed architecture and related work

power. Nonetheless, it requires to trigger the sense signal which floats the SRAM-like sensing circuitry and consumes significant power on that interval.

Fig. 13 demonstrates the overall power consumption of each circuit which is obtained by post-processing the architectural results for each circuit (e.g., array size and channel width) and applying its actual activity factor and segment utilization. An average saving of 24.3% with respect to SRAM-based designs has been achieved over the MCNC benchmarks which stems from reducing the static and dynamic power of the routing resources. Table VII presents the power and power-delay product results of other designs, as well.

It should be noted that the impact of the architectural parameters on dynamic power (i.e., F_s, F_{cin} and F_{cout} which are the same for all benchmark circuits) had been inherently considered while obtaining the wires capacitance and resistance using COFFE tool [46]. The channel width, however, does not affect the power of each individual SB. It is because $\frac{I}{4} \times 2$ *Connection Block* (CB) multiplexers with $F_{cin} \times W$ are connected to each horizontal or vertical channel from adjacent CLBs. Thus, totally $\frac{I}{4} \times 2 \times F_{cin} \times W$ inputs are connected to W tracks of the channel, resulting in $\frac{I}{4} \times 2 \times F_{cin}$ inputs per routing track, which induce a capacitance independent of channel width.

E. Impact of LUT Input Size

To demonstrate the scalability of the proposed architecture, we have compared the area and average delay and power of the proposed four to six input LUTs in Table VIII. Expectedly, delay and power of SRAM-based and the proposed LUTs are similar in LUTs with different sizes. It is because the original multiplexer-based structure of the LUT has not been modified in the proposed design and the RRAM configuration cells are not in the critical path. In addition, since the average power consumption of the RRAM cell is almost equal to an SRAM cell, both designs consume similar average power. On the other hand, area of a K-input SRAM- and RRAM-based LUT can be estimated as $2^K S_{SRAM} + (2^K - 1)A_{mux2}$ and $(2^K - 1)A_{mux2}$, respectively. Therefore, $\frac{A_{LUT,RRAM}}{A_{LUT,SRAM}} \simeq \frac{2^K A_{mux2}}{2^K A_{SRAM} + 2^K A_{mux2}} = \frac{A_{mux2}}{A_{SRAM} + A_{mux2}}$ which is independent of LUT size. According to experimental results in Table VIII, area improvement of the proposed LUTs are between 41% and 42%, i.e., it is independent of number of the inputs.

V. Reliability

One of the major obstacles that may hinder the industrial success of RRAM devices is their reliability. In the context of NVMs, reliability aspects not only include device inherent parameters such as endurance (i.e., the maximum number of reliable programming cycles), retention time, and HRS/LRS resistance values, but also switching parameters

TABLE VIII
Comparing different SRAM- and RRAM-based LUTs

	SRAM/RRAM LUT4	SRAM/RRAM LUT5	SRAM/RRAM LUT6
Delay (ps)	102/103	126/127	139/140
Power (uW)	0.91/0.89	1.75/1.71	3.45/3.37
Area	233.5/137.5	467.1/270.9	932.1/548.1

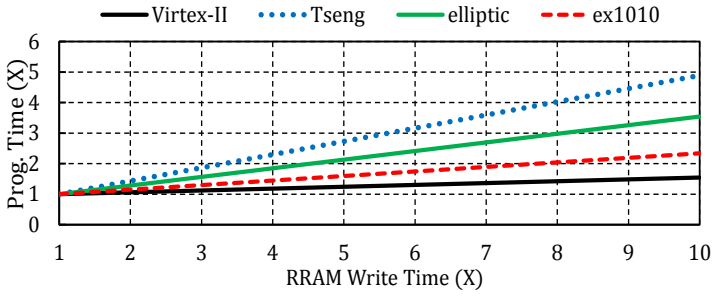


Fig. 14. Programming time of RRAM-based FPGA (with respect to initial time) for T_{set} and T_{reset} from $1\times$ to $10\times$

such as programming duration and voltage. Moreover, the relatively high voltage requirement of programming the RRAM cells can interfere with regular transistors of the device and cause them to breakdown. In this section, we examine the impact of uncertainties in parameters such as programming time and HRS and LRS resistance values on the proposed designs. At the end, reliability aspects of the proposed programming structure is discussed.

A. Programming Duration

The programming speed of the proposed FPGA is determined by both *set* and *reset* operations. As it has been investigated in [50], in the worst case, $10\times$ increase in programming duration of devices with initial T_{set} of $\sim$100ns (TiO_2) can be observed. It is noteworthy that there is a trade-off between programming duration and cell disturb immunity. Thus, providing long *reset* or *set* periods enhances the programming reliability, in addition to taking the worst-case cells duration into consideration. Therefore, we estimate the programming time of the proposed RRAM-based FPGA considering T_{set} and T_{reset} from $1\times$ (i.e., initial 50ns and 10ns) to $10\times$ (i.e., $0.5\mu s$ and 100ns). Fig. 14 demonstrates the programming time of RRAM-based FPGAs as RRAM write time (*reset* and *set*) increases from $1\times$ to $10\times$. For this end, four devices with different sizes are selected, i.e., *tseng* (smallest), *elliptic* (medium), and *ex1010* (largest). In addition, the largest Virtex-II device, i.e., XC2V8000 with 93,184 LUTs which can be assumed as a $\sim 96 \times 96$ device is also considered. As shown in Fig. 14, the programming time of larger devices is less affected since in these devices, the T_{set} and T_{reset} factors are prorated by the frame loading time. *Tseng* exhibits $4.9\times$ increase in programming time when T_{set} and T_{reset} increases to $10\times$, however, this value is only $1.5\times$ for Virtex-II device.

B. HRS and LRS Variation

The device-level phenomenon behind the origin of HRS and LRS resistance variations together with techniques to alleviate them (e.g., by tuning the oxide thickness using additional buffer oxide layer or using write-verify technique) has been discussed in the literature. For more details, the reader can refer to [51]. LRS resistance variation manifests itself as increase in the original value. On the other hand, HRS drift exhibits as decrease in the primary resistance. Such techniques can reduce the distribution of tail bits of HRS to $> 0.5\times$ and the tail bits of LRS to $< 2\times$ of the intended values.

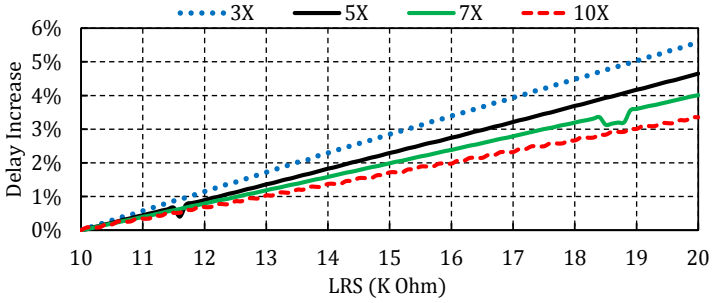


Fig. 15. Delay of SBs with different buffer size due to LRS variation

We examine the impact of resistance variations of the RRAM cells in the delay and power characteristics of the proposed designs. To this end, we suppose no device-level or architectural-level enhancements on the RRAM resistance have been carried out; hence a pessimistic $2\times$ increase and $0.1\times$ decrease (i.e., decrease to $0.1\times$ of the initial value) [51] for the LRS and HRS resistance values is considered, respectively. It should be noticed that the delay of the proposed SB merely depends on the LRS value of the RRAM. However, the leakage power of the proposed LUT and SB depends on both the LRS and HRS cells. Thus, a two-dimensional resistance state should be investigated for the latter cases. In addition, since the configuration cells of the LUT are not in its critical path, variation in RRAM resistance does not affect the LUT delay (unless the associated buffer fails to provide a strong 0/1 which does not occur in the considered range of variations).

Fig. 15 represents the increase in SB delay (with different buffer strengths) as the LRS increases from original $10K\Omega$ to the maximum of $20K\Omega$. This $2\times$ increase results in only 4.7% delay increase in the SB used in our experiments (i.e., SB with $5\times$ buffer size) because the main contributor in the SB delay is the input/output buffer and the corresponding wire segment resistive and capacitive parameters. Therefore, even assuming worst-case LRS value which increases the SB delay to $83ps$, the proposed SB has still smaller delay than SRAM-based counterpart. Extrapolating the results reported in Fig. 15 reveals that up to $6\times$ increase in RRAM LRS value ($60K\Omega$) in the proposed SB can be tolerated, i.e., it still affords a delay smaller than SRAM-based SB delay. In addition, as it is evident from Fig. 15, the impact of RRAM variation diminishes in larger buffers.

It should be noted that our investigation on impact of the RRAM parameter variation on SB delay is different from [52]. We study the impact of undesirable resistance variations on circuit performance. These variations can either have device-level [51] or architecture-level origin such as variations in *set* or *reset* currents. For example, the $6\times$ increase in LRS resistance can be accounted for (unwanted) $3\times$ fluctuation in the *set* current (voltage), as the measured *set* current associated with $LRS = 60K\Omega$ is ~$1\mu A$, compared with $3\mu A$ required to set the RRAM to $10K\Omega$. On the other hand, [52] explores the efficient size for programming transistors which gives the minimum path delay. In our analysis, we assumed a fixed programming transistor size (hence, constant parasitic capacitance) because the RRAM resistance values are supposed to be fixed (except the unwillingly varied ones). In addition, we have used minimum size I/O transistors ($\frac{320nm}{270nm}$) which is sufficient to deliver $100\mu A$ and only the programming voltage was the determinant factor. RRAM cell with higher LRS resistance requires lower *reset* and *set* current, however, it does not affect the size of currently minimum-size transistors.

Fig. 16(a) and Fig. 16(b), respectively, demonstrate the leakage power of the proposed SB and LUT with regard to varying LRS and HRS values. In both figures, LRS variation has negligible impact on the design power. It is because, as mentioned in Section IV-D, in the proposed SB, LRS cells consume almost zero power since there is no voltage difference between terminals (nodes) of such cells. Additionally, in the proposed LUT, the RRAM is in its LRS state when the cell holds one. Accordingly, the cell power will be equal to $V^2/(LRS+R) \simeq V^2/R$ since R is significantly greater than LRS, i.e., $R \gg LRS$.

As illustrated in Fig. 16(a), the leakage power of the proposed SB increases by 20.6% as the HRS values of all cells reduce to $0.1\times$. Therefore, the voltage difference on a HRS cell nodes (see Section IV-D or Fig. 2) results in approximately $10\times$ power consumption of such cells[1]. However, low fraction of cells with such condition (i.e., having voltage difference between nodes) and relatively high power of SB large buffers reclaims its total power increase. On the other hand, as shown in Fig. 16(b), drift in RRAMs HRS down to $0.3\times$ imposes up to 90% power overhead and it exceeds 200% when the RRAMs resistance diminishes to $0.2\times$[2]. The exponential increase of power as the HRS reduces corresponds to leakage power of the associated isolating buffers (see Section. IV-C). Low HRS resistance value produces weak zero (e.g., $HRS = 300M\Omega$ results in $V = R/(R+HRS) = 130mV$) which subsequently applies a near-V_{th} voltage to the configuration buffer and pushes its transistors into high leakage saturation region. Nevertheless, this problem could be mitigated by choosing high-V_{th} buffers – especially as the buffers are not in the critical path.

It is noteworthy that the employed RRAM cell [43] maintains the initial $1G\Omega$ HRS after 10^5 programming cycles, and such pessimistic resistance drift will not occur. A $2\times$ increase in LRS value is observed after 10^5 cycles, however, as shown in Fig. 16, the impact of LRS on power consumption is negligible. The retention time of 10 years has been observed for the employed RRAM cells, as well.

C. Programming Reliability

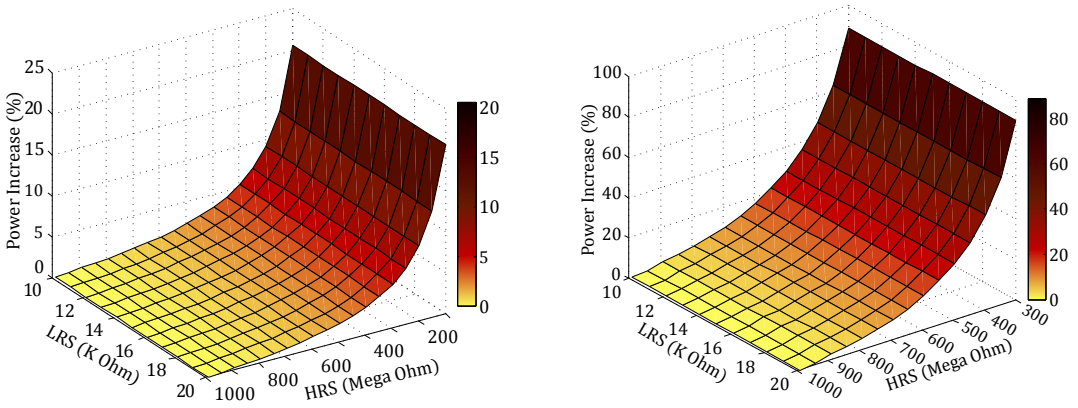
Back to Fig. 1(c), during programming (for instance) RRAM1, $P1$ is enabled and delivers the programming voltage/current to the output node of buffer WI. In this condition, depending to its input, either the NMOS or PMOS transistor of WI buffer is *on*. The former scenario causes the programming current to sink to ground through the WI buffer (in addition to N2) and consequently reduces the current density necessary to program RRAM1. To address this issue, the input buffers of the proposed switch box can be equipped with enabling (i.e., tri-state) transistors to cut off the buffer during the programming and preclude of sinking the programming current.

On the other hand, since the programming current flows through the transistor N2, a non-zero voltage occurs on input node of the NO buffer. If this voltage rises above the operating voltage of the transistors of NO buffer, (which is typically $1V-1.2V$ in 45nm technology) it can cause them to breakdown. Nevertheless, during the reset process, $V_{reset} = 1.3V$ and resistance of the RRAM increases from initial $10K\Omega$ (to $1G\Omega$). Thus, the resistance of P1-RRAM1 pair will be always larger than that of N2 during the reset process. Thus, voltage of input node of NO buffer will be smaller than $\frac{R_{N2}}{R_{P1}+R_{RRAM1}} \times 1.3V = 0.65V$. Analogously, during the set process, a voltage equal to $2.0V$ will be applied to source node of P1 and RRAM1 resistance begins to reduce from $1G\Omega$ downto $10K\Omega$. In the worst case (i.e., $R_{RRAM1} = 10K\Omega$), a maximum voltage of $\frac{2.0}{2} = 1.0V$ will be applied to NO which is far below the breakdown voltage.

VI. Conclusion

In this paper, we presented switch box and lookup table designs for the RRAM-based FPGAs to integrate the programming circuitry in these blocks and create efficient programmable components. By taking advantage of RRAM appealing features and eliminating the CMOS-compatibility and scalability of flash memory, the proposed RRAM-based FPGA offers the features of SRAM-based FPGAs (e.g., design reconfiguration and short time-to-market) with non-volatility of flash-memory-based FPGAs. Experimental results demonstrated that despite using one programmable RRAM per configuration bit, the proposed architecture increases the configuration time by only 39.3% compared with commercial-size SRAM-based FPGAs while it enhances it by

[1]The power of such HRS cells increases from $1/1G\Omega = 1nW$ to $10nW$.
[2]HRS below $300M\Omega$ is not shown in Fig. 16(a) as it disfigures the scales.



(a) SB power distribution

(b) LUT power distribution

Fig. 16. Static power distribution of the proposed switch box and LUT with respect to HRS and LRS variation

$83\times$ compared to previous studies that use one-by-one programming scheme. Simulation results over the MCNC benchmarks demonstrated that the proposed RRAM-based FPGA reduces the total area and delay by 59.4% and 20.1%, respectively. By eliminating the always-on SRAM-based configuration cells, the static power reduced by 24.3%, on average. Finally, keeping the logic dynamic power intact, the dynamic power of our proposed architecture decreased by 56.0%, mainly due to shrinking the routing wire length. We also investigated the role of RRAM parameters variation, e.g., *set* and *reset* write time and resistance drift, in the efficiency of the proposed architecture. Experimental results showed that with $10\times$ increase of the write time, the configuration time of commercials-size device increases by $1.5\times$ while for smallest benchmark circuits, this value can raise up to $4.9\times$. Nonetheless, this programming time is still admissible for FPGA-based designs since the device reconfiguration is usually performed once a while. In addition, in the worst case when HRS resistance value of all switch box RRAM cells shrinks to $0.1\times$ of the original value, its static power increases by 20.6% which is still power-efficient than conventional SRAM-based switch box. On the other hand, the power overhead due to RRAM resistance drift of the proposed LUT can reach to more than $2\times$ in the worst scenario. However, RRAM cell tuning and high-threshold buffers can suppress this overhead without any performance loss.

Behnam Khaleghi has received his B.Sc. and M.Sc. degrees in computer engineering from *Sharif University of Technology* (SUT), Tehran, Iran, in 2013 and 2016, respectively. He is currently working as a research assistant in the *Data Storage, Networks, & Processing* (DSN) Laboratory at the Department of Computer Engineering, SUT. He spent the summer 2014 and 2015 as a research assistant at the Chair for Embedded Systems in the Karlsruhe Institute of Technology. His research interests include reconfigurable architectures and computer-aided design. He has two Best Paper Nominations at the DAC'17 and DATE'17.

Hossein Asadi (M'08, SM'14) received the B.Sc. and M.Sc. degrees in computer engineering from the SUT, Tehran, Iran, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from Northeastern University, Boston, MA, USA, in 2007.

He was with EMC Corporation, Hopkinton, MA, USA, as a Research Scientist and Senior Hardware Engineer, from 2006 to 2009. From 2002 to 2003, he was a member of the Dependable Systems Laboratory, SUT, where he researched hardware verification techniques. From 2001 to 2002, he was a member of the Sharif Rescue Robots Group. He has been with the Department of Computer Engineering, SUT, since 2009, where he is currently a tenured Associate Professor. He is the Founder and Director of the Data Storage, Networks, and Processing (DSN) Laboratory, Director of Sharif High-Performance Computing Center, the Director of Sharif Information Technology Service Center (ITC), and the President of Sharif ICT Innovation Center. He spent three months in the summer 2015 as a Visiting Professor at the School of Computer and Communication Sciences at the Ecole Poly-technique Federele de Lausanne (EPFL). He has also co-founded the first startup company in the Middle East, called HPDS, designing and fabricating midrange and high-end data storage systems. He has authored and co-authored more than seventy technical papers in reputed journals and conference proceedings. His current research interests include data storage systems and networks, solid-state drives, operating system support for I/O and memory management, and reconfigurable and dependable computing.

Dr. Asadi was a recipient of the Technical Award for the Best Robot Design from the International RoboCup Rescue Competition, organized by AAAI and RoboCup, a recipient of Best Paper Award at the 15th CSI Internation Symposium on Computer Architecture and Digital Systems (CADS), the Distinguished Lecturer Award from SUT in 2010, the Distinguished Researcher Award and the Distinguished Research Institute Award from SUT in 2016. He is also recipient of Extraordinary Ability in Science visa from US Citizenship and Immigration Services in 2008. He has also served as the publication chair of several national and international conferences including CNDS2013, AISP2013, and CSSE2013 during the past four years. Most recently, he has served as a Guest Editor of IEEE Transactions on Computers, a Program Co-Chair of the 18th International Symposium on Computer Architecture & Digital Systems (CADS2015), and the Program Chair of CSI National Computer Conference (CSICC2017).

REFERENCES

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 203–215, 2007.

[2] K. Huang, Y. Ha, R. Zhao, A. Kumar, and Y. Lian, "A Low Active Leakage and High Reliability Phase Change Memory (PCM) Based Non-Volatile FPGA Storage Element," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 9, pp. 2605–2613, Sept 2014.

[3] S. Yazdanshenas and H. Asadi, "Fine-Grained Architecture in Dark Silicon Era for SRAM-Based Reconfigurable Devices," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 10, pp. 798–802, 2014.

[4] Z. Ebrahimi, B. Khaleghi, and H. Asadi, "PEAF: A Power-Efficient Architecture for SRAM-Based FPGAs Using Reconfigurable Hard Logic Design in Dark Silicon Era," *IEEE Transactions on Computers*, vol. 66, no. 6, pp. 982–995, 2017.

[5] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-D FPGA," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 216–229, 2007.

[6] P. Gaillardon, D. Sacchetto, G. B. Beneventi, M. Ben Jamaa, L. Perniola, F. Clermidy, I. O'Connor, and G. De Micheli, "Design and architectural assessment of 3-d resistive memory technologies in FPGAs," *IEEE Transactions on Nanotechnology*, vol. 12, no. 1, pp. 40–50, 2013.

[7] G. Asadi and M. B. Tahoori, "Soft error rate estimation and mitigation for SRAM-based FPGAs," in *Proceedings of the ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays*. 2005, pp. 149–160.

[8] H. Asadi, M. B. Tahoori, B. Mullins, D. Kaeli, and K. Granlund, "Soft error susceptibility analysis of SRAM-based FPGAs in high-performance information systems," *Nuclear Science, IEEE Transactions on*, vol. 54, no. 6, pp. 2714–2726, 2007.

[9] E. Seevinck, F. J. List, and J. Lohstroh, "Static-noise margin analysis of MOS SRAM cells," *Solid-State Circuits, IEEE Journal of*, vol. 22, no. 5, pp. 748–754, 1987.

[10] J. S. Meena, S. M. Sze, U. Chand, and T.-Y. Tseng, "Overview of emerging nonvolatile memory technologies," *Nanoscale Research Letters*, vol. 9, no. 1, pp. 1–33, 2014.

[11] 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, 2012.

[12] W. Guan, S. Long, Q. Liu, M. Liu, and W. Wang, "Nonpolar Nonvolatile Resistive Switching in Cu Doped," *Electron Device Letters, IEEE*, vol. 29, no. 5, pp. 434–437, 2008.

[13] Y. Y. Liauw, Z. Zhang, W. Kim, A. Gamal, and S. S. Wong, "Nonvolatile 3D-FPGA with monolithically stacked RRAM-based configuration memory," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*. IEEE, 2012, pp. 406–408.

[14] W. Zhao, E. Belhaire, V. Javerliac, C. Chappert, and B. Dieny, "Evaluation of a Non-volatile FPGA based on MRAM technology," in *IEEE International Conference on Integrated Circuit Design and Technology*. 2006, pp. 1–4.

[15] P.-E. Gaillardon, M. Ben-Jamaa, M. Reyboz, G. B. Beneventi, F. Clermidy, L. Perniola, and I. O'Connor, "Phase-change-memory-based storage elements for configurable logic," in *International Conference on Field-Programmable Technology (FPT)*. IEEE, 2010, pp. 17–20.

[16] S. Yazdanshenas, B. Khaleghi, P. Ienne, and H. Asadi, "Designing Low Power and Durable Digital Blocks Using Shadow Nanoelectromechanical Relays," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 24, no. 12, pp. 3489–3498, 2016.

[17] J. Cong and B. Xiao, "FPGA-RPI: A novel FPGA architecture with rram-based programmable interconnects," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 864–877, 2014.

[18] M. Liu and W. Wang, "rFPGA: CMOS-nano hybrid FPGA using RRAM components," in *Nanoscale Architectures, IEEE International Symposium on*. IEEE, 2008, pp. 93–98.

[19] P.-E. Gaillardon, M. H. Ben-Jamaa, G. B. Beneventi, F. Clermidy, and L. Perniola, "Emerging memory technologies for reconfigurable routing in FPGA architecture," in *Electronics, Circuits, and Systems (ICECS), 17th IEEE International Conference on*. IEEE, 2010, pp. 62–65.

[20] S. Paul, S. Mukhopadhyay, and S. Bhunia, "A circuit and architecture codesign approach for a hybrid cmos–sttram nonvolatile FPGA," *IEEE Transactions on Nanotechnology*, vol. 10, no. 3, pp. 385–394, 2011.

[21] P.-E. Gaillardon, D. Sacchetto, S. Bobba, Y. Leblebici, and G. De Micheli, "GMS: Generic memristive structure for non-volatile FPGAs," in *VLSI and System-on-Chip, (VLSI-SoC), IEEE/IFIP 20th International Conference on*. IEEE, 2012, pp. 94–98.

[22] J. Cong and B. Xiao, "FPGA-RPI: A novel FPGA architecture with rram-based programmable interconnects," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 864–877, 2014.

[23] D. Suzuki, M. Natsui, A. Mochizuki, S. Miura, H. Honjo, H. Sato, S. Fukami, S. Ikeda, T. Endoh, H. Ohno *et al.*, "Fabrication of a 3000-6-input-LUTs embedded and block-level power-gated nonvolatile FPGA chip using p-MTJ-based logic-in-memory structure," in *Symposium on VLSI Circuits (VLSI Circuits)*. IEEE, 2015, pp. C172–C173.

[24] R. Rajaei, "Radiation-hardened design of nonvolatile mram-based fpga," *IEEE Transactions on Magnetics*, vol. 52, no. 10, pp. 1–10, 2016.

[25] Y. Chen, J. Zhao, and Y. Xie, "3D-nonFAR: three-dimensional non-volatile FPGA architecture using phase change memory," in *ACM/IEEE International Symposium on Low Power Electronics and Design*. 2010, pp. 55–60.

[26] O. Turkyilmaz, S. Onkaraiah, M. Reyboz, F. Clermidy, C. A. Hraziia, J. Portal, and M. Bocquet, "RRAM-based FPGA for normally off, instantly on applications," in *Nanoscale Architectures (NANOARCH), IEEE/ACM International Symposium on*. IEEE, 2012, pp. 101–108.

[27] (2016) Synopsys EDA Tools Flow. [Online]. Available: http://www.synopsys.com/

[28] H. Li, P. Huang, B. Gao, B. Chen, X. Liu, and J. Kang, "A SPICE model of resistive random access memory for large-scale memory array simulation," *IEEE Electron Device Letters*, vol. 35, no. 2, pp. 211–213, 2014.

[29] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed *et al.*, "VTR 7.0: next generation architecture and CAD system for FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 2, p. 6, 2014.

[30] S. Yang, *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC), 1991.

[31] C.-Y. Wen, J. Li, S. Kim, M. Breitwisch, C. Lam, J. Paramesh, and L. Pileggi, "A non-volatile look-up table design using PCM (phase-change memory) cells," in *IEEE Symposium on VLSI Circuits (VLSIC)*. IEEE, 2011, pp. 302–303.

[32] A. Ahari, H. Asadi, B. Khaleghi, and M. Tahoori, "A power-efficient reconfigurable architecture using PCM configuration technology," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2014, pp. 1–6.

[33] S. Tanachutiwat, M. Liu, and W. Wang, "FPGA Based on Integration of CMOS and RRAM," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 11, pp. 2023–2032, 2011.

[34] Y.-C. Chen, W. Wang, H. Li, and W. Zhang, "Non-volatile 3D stacking RRAM-based FPGA," in *Field Programmable Logic and Applications (FPL), 22nd International Conference on*. IEEE, 2012, pp. 367–372.

[35] P.-E. Gaillardon, X. Tang, J. Sandrini, M. Thammasack, S. R. Omam, D. Sacchetto, Y. Leblebici, and G. De Micheli, "A ultra-low-power FPGA based on monolithically integrated RRAMs," in *Design, Automation & Test in Europe Conference & Exhibition*, 2015, pp. 1203–1208.

[36] A. Pirovano, A. Redaelli, F. Pellizzer, F. Ottogalli, M. Tosi, D. Ielmini, A. L. Lacaita, and R. Bez, "Reliability study of phase-change nonvolatile memories," *Device and Materials Reliability, IEEE Transactions on*, vol. 4, no. 3, pp. 422–427, 2004.

[37] G. Lemieux and D. Lewis, *Design of interconnection networks for programmable logic*. Springer, 2004, vol. 22.

[38] X. Tang, G. Kim, P.-E. Gaillardon, and G. De Micheli, "A Study on the Programming Structures for RRAM-Based FPGA Architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 4, pp. 503–516, 2016.

[39] X. Tang, G. De Micheli, and P.-E. Gaillardon, "A high-performance fpga architecture using one-level rram-based multiplexers," *IEEE Transactions on Emerging Topics in Computing*, 2016.

[40] X. Tang, E. Giacomin, G. De Micheli, and P.-E. Gaillardon, "Circuit designs of high-performance and low-power rram-based multiplexers based on 4t (ransistor) 1r (ram) programming structure," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2016.

[41] H. Michinishi, T. Yokohira, T. Okamoto, T. Inoue, and H. Fujiwara, "Testing for the programming circuit of LUT-based FPGAs," in *Asian Test Symposium*, 1997, pp. 242–247.

[42] "Virtex-6 FPGA Configurable Logic Block," User Guide, Xilinx, Feb. 2012.

[43] W. Kim, S. I. Park, Z. Zhang, Y. Yang-Liauw, D. Sekar, H.-S. Wong, and S. S. Wong, "Forming-Free Nitrogen-Doped AlO_X RRAM with Sub-μA Programming Current," in *VLSI Technology (VLSIT), Symposium on*. IEEE, 2011, pp. 22–23.

[44] (2016) Nangate Open Cell Library. Available: http://www.nangate.com/

[45] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-time partial reconfiguration speed investigation and architectural design space exploration," in *Field Programmable Logic and Applications (FPL), International Conference on*. IEEE, 2009, pp. 498–502.

[46] C. Chiasson and V. Betz, "COFFE: Fully-automated transistor sizing for FPGAs," in *Field-Programmable Technology (FPT), International Conference on*. IEEE, 2013, pp. 34–41.

[47] (2013) Predictive Technology Model (PTM). [Online]. Available: http://ptm.asu.edu/

[48] R. Ho, "On-chip wires: scaling and efficiency," Ph.D. dissertation, Citeseer, 2003.

[49] J. Lamoureux and S. J. Wilton, "Activity estimation for field-programmable gate arrays," in *Field Programmable Logic and Applications (FPL), International Conference on*. IEEE, 2006, pp. 1–8.

[50] W.-C. Luo, J.-C. Liu, Y.-C. Lin, C.-L. Lo, J.-J. Huang, K.-L. Lin, and T.-H. Hou, "Statistical model and rapid prediction of RRAM SET speed–disturb dilemma," *Electron Devices, IEEE Transactions on*, vol. 60, no. 11, pp. 3760–3766, 2013.

[51] S. Yu, X. Guan, and H.-S. P. Wong, "On the stochastic nature of resistive switching in metal oxide RRAM: Physical modeling, Monte Carlo simulation, and experimental characterization," in *Electron Devices Meeting (IEDM), IEEE International*. IEEE, 2011, pp. 17–3.

[52] X. Tang, P.-E. Gaillardon, and G. De Micheli, "A high-performance low-power near-Vt RRAM-based FPGA," in *Field-Programmable Technology (FPT), International Conference on*. IEEE, 2014, pp. 207–214.