IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 17 May 2016; revised 25 August 2016; accepted 6 November 2016.
Date of publication 17 November 2016; date of current version 7 June 2017.

Digital Object Identifier 10.1109/TETC.2016.2630121

A High-Performance FPGA Architecture Using One-Level RRAM-Based Multiplexers

XIFAN TANG, (Student Member, IEEE), GIOVANNI DE MICHELI, (Fellow, IEEE), AND PIERRE-EMMANUEL GAILLARDON, (Member, IEEE)

X. Tang and G.D. Micheli are with the Integrated Systems Laboratory, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Vaud Lausanne 1015, Switzerland
P.-E. Gaillardon is with the Laboratory for NanoIntegrated Systems, Electrical and Computer Engineering Department, University of Utah, Salt Lake City UT 84112
CORRESPONDING AUTHOR: X. TANG (xifan.tang@epfl.ch)

ABSTRACT *Resistive Random Access Memory* (RRAM)-based routing multiplexers, built using a one-level structure, are significantly more delay efficient than state-of-art SRAM-based implementations thanks to their lower achievable on-state resistance. In addition, the delay of RRAM-based multiplexers scales better with respect to input size than SRAM-based multiplexers. This property allows RRAM-based FPGA architectures to employ larger multiplexers than their SRAM-based counterparts, without generating any delay overhead. In this paper, we first evaluate at the circuit-level the delay improvements of a state-of-art RRAM-based multiplexer. Then, to unlock the potential of RRAM-based multiplexers, we propose three related FPGA architecture optimizations: (a) The routing tracks should be interconnected to *Look-Up Table* (LUT) inputs via a one-level crossbar, instead of through *Connection Blocks* and local routing; (b) The *Switch Block* (SB) should employ larger multiplexers; (c) Length-2 wires should be used instead of length-4 wires. When a classical architecture is considered for both SRAM and RRAM technologies, RRAM-based FPGAs can reduce area by 17 percent and delay by 32 percent with zero power overhead for a 40 nm technology. The proposed architectural enhancements can further improve area by 15 percent, delay by 10 percent and channel width by 13 percent. Combining RRAM technology and architectural enhancements, the proposed RRAM-based FPGA architecture improves *Area-Delay Product* by 57 percent and *Delay-Power Product* by 38 percent, as compared to a SRAM-based FPGA exploiting a classical architecture.

INDEX TERMS Resistive memory (RRAM), FPGA, multiplexer, routing, high-performance, low-power

I. INTRODUCTION

The promises of the *Resistive Random Access Memories* (RRAMs) technology have initiated [1]–[6], in the past few years, intensive research efforts in exploring high-performance RRAM-based FPGA. Previous works [7]–[13] focus on replacing *Static Random Access Memory* (SRAM)-based routing multiplexers of classical FPGA architectures with RRAM-based multiplexers. A RRAM-based routing element uses RRAMs not only to store the configuration of the multiplexer tree, but employs the memory elements to route the data path signals. Since RRAM-based multiplexers can achieve lower on-state resistances than SRAM-based multiplexers, they are naturally more delay-efficient and lead to higher-performance FPGA architectures. It is reported that a 7-15 percent gain in area, a 45-58 percent reduction in delay, and a 20-58 percent reduction in power can be achieved, when compared RRAM-based FPGAs to their SRAM-based counterparts [7]–[11]. However, previous works [7]–[11] typically employ 2T(ransistor)1R(AM)-based circuit designs, and only focus on the architectural repercussions of this technology. Very limited works investigate realistic RRAM-based circuit design constraints, while these have strong impact on the final architectural performances.

Most SRAM-based FPGA architectures typically employ multiple levels of small crossbars, instead of large multiplexers, due to a strong limitation of SRAM-based multiplexer: Whatever multiplexer structure is employed, their area, delay and power increase linearly with the input size [14]. However, we will see in this paper that the delay of RRAM-based multiplexers is independent from the input

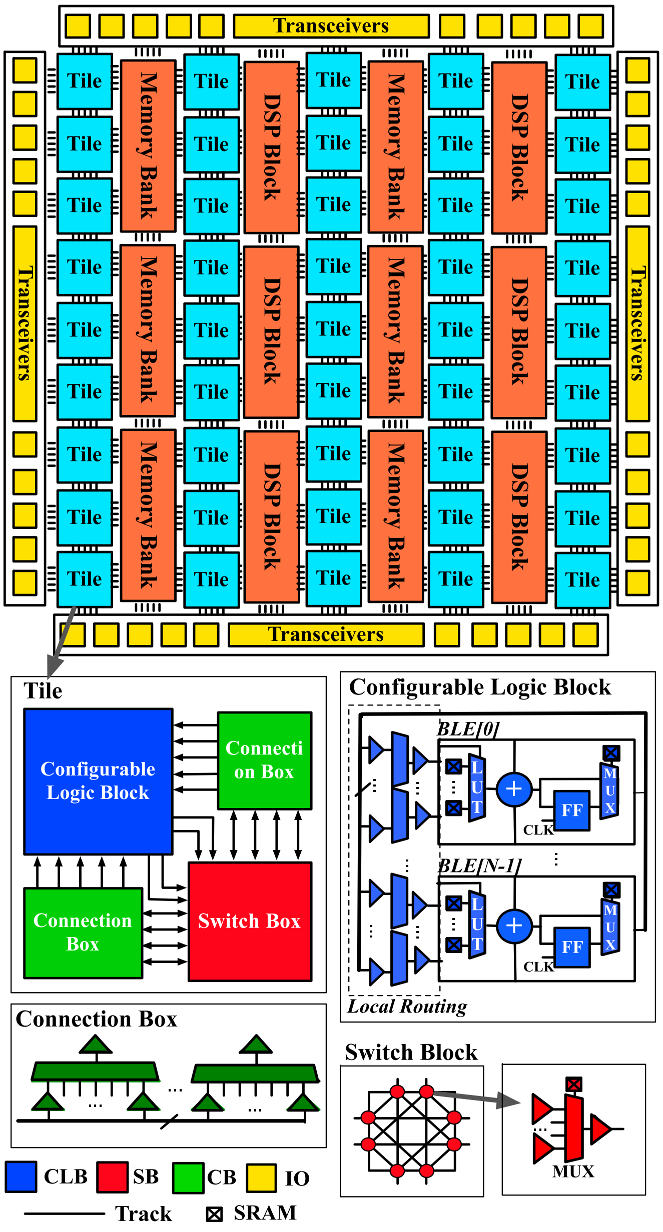


FIGURE 1. **Tile-based FPGA architecture.**

size and therefore the architectural design space can be extended beyond the limitations of SRAM-based multiplexer. Indeed, the properties of RRAM-based multiplexer allow the FPGA architect to size differently its routing multiplexers by: privileging one-level crossbars, made of large multiplexers, as much as possible. This paradigm shift in the interconnection topology also requires to rethink the optimal architectural parameters, which have been well determined for classical SRAM-based architectures. Hence, it is worthwhile to identify properly-sized RRAM-based FPGA architectures which can exploit the full potential of RRAM-based multiplexers, and determine the associated optimal architectural parameters.

Compared to our previous work [11]–[13], the contributions of this paper are:

1) We investigate the circuit design aspects of RRAM-based multiplexers based on the current state-of-art 4T (ransistor)1R(RAM) programming structure [13]. By applying circuit-level optimizations, 4T1R-based multiplexers can achieve up to 2× delay improvements with regards to regular routing multiplexers. Averaged over the twenty biggest MCNC and VTR benchmarks, when a traditional architecture is considered for both SRAM and RRAM technologies, RRAM-based FPGA reduces area by 17 percent and delay by 32 percent with zero power overhead for a 40 nm technology.
2) We identify that the classical FPGA architectures may not fully unlock the potential of 4T1R-based multiplexers. Three architectural optimizations are proposed in order to fully exploit the advantage of 4T1R-based multiplexers: (a) The routing tracks should be interconnected to *Look-Up Table* (LUT) inputs via a one-level crossbar, instead of through *Connection Blocks* (CB) and local routing; (b) As a side effect, the *Connection Block* $F_{c,in}$ and *Switch Block* (SB) F_s connectivity parameters should be increased; (c) The best single wire length L should be smaller. We study the best values of $F_{c,in}$, F_s and L in terms of area, delay, power and channel width. Architecture-level simulations show that a RRAM-based FPGA should employ ($F_{c,in} = 0.33$, $F_s = 6$ and $L = 2$) to achieve the best performance, which are different from those of classical SRAM-based architectures. The proposed architectural enhancements can further improve area by 15 percent, delay by 10 percent and channel width by 13 percent. Combining RRAM technology and architectural enhancements, the proposed RRAM-based FPGA reduces *Area-Delay Product* (ADP) by 57 percent and *Delay-Power Product* (PDP) by 38 percent, when compared to a SRAM-based FPGA with a classical architecture.

The rest of this paper is organized as follows. Section II reviews the background of SRAM-based and previous works on RRAM-based FPGA architectures. Section III introduces the general experimental methodology used in this paper. Section IV presents the RRAM-based multiplexer design based on 4T1R programming structure. Section V proposes three architectural optimizations that exploit the advantages of RRAM-based multiplexers. Section VI shows the overall improvements by combining 4T1R-based multiplexers and architectural optimizations. Section VII concludes the paper.

II. BACKGROUND

In this section, we first review the well optimized SRAM-based FPGA architectures as well as previous works on RRAM-based FPGA architectures.

A. MODERN SRAM-BASED FPGA ARCHITECTURE

Modern SRAM-based FPGAs typically employ a *tile*-based architecture, where an array of *logic tiles* are interconnected by high-density routing tracks, as shown in Figure 1 [14]. In each tile, there are a *Configurable Logic Block* (CLB), two

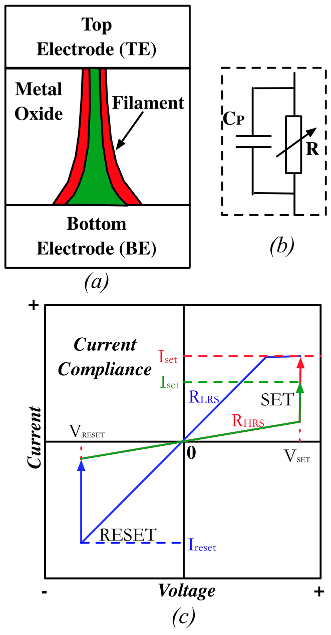


FIGURE 2. **(a) RRAM structure and illustration on conducting filaments; (b) Equivalent RC model of a RRAM; (c) Typical bipolar RRAM I-V characteristics.**

Connection Blocks and a *Switch Block*. Routing tracks inside tiles are connected to CLB inputs by CBs, while SBs interconnects the routing tracks between tiles. A CLB is composed of *N Basic Logic Elements* (BLEs) and a local routing architecture providing inner-block interconnections. A BLE contains a fracturable *Look-Up Table* [15], a *Flip-Flop* (FF) and a 2:1 multiplexer, which selects either a combinational or a sequential output. To improve the efficiency of arithmetic functions, commercial FPGAs [16], [17] widely enhance BLEs by adding more functionalities, such as hardened adders and shift registers. For the sake of accelerating arithmetic-intensive implementations, commercial FPGAs [16], [17] replace columns of tiles by *Digital Signal Processor* (DSP) and memory banks.

B. RRAM TECHNOLOGY

A *Resistive Random Access Memory* (RRAM) typically employ a three-layer (sandwiched) structure, formed by a *Top Electrode* (TE), a *Bottom Electrode* (BE) and a switching metal oxide, as shown in Figure 2(a). RRAMs switching mechanisms can be grouped into *Unipolar Resistive Switching* (URS) and *Bipolar Resistive Switching* (BRS) [18]. Bipolar RRAMs are considered in this paper, by following the choices of most RRAM-based researches [1]–[13]. Figure 2 (b) depicts the equivalent RC model of a RRAM. Besides the configurable resistance R, a parasitic capacitance C_P induced by TE and BE should also be considered. The typical I-V curve of a bipolar RRAM is illustrated in Figure 2(c). By providing the right combination of programming voltage and current, RRAMs can be freely switched between two stable resistance states: a *High Resistance State* (HRS) and a *Low Resistance State* (LRS). A change in conductivity of a RRAM actually results from the growth/dilution of a conductive filament in the switching layer, induced by a positive/negative programming voltage between TE and BE. The width of the filaments, which determines the LRS resistance, is strongly correlated to the programming current flowing through the RRAM. RRAMs are compatible with *Back-End-of-Line* (BEoL) process, and can be fabricated on the top of transistors at low cost[19]. The filamentary conduction property brings to RRAMs not only device-to-device variation but also cycle-to-cycle variability. Both device-to-device and cycle-to-cycle variations are reported to be well controlled between 10-20 percent [20]–[22]. To be more robust in cycle-to-cycle variations, we can introduce program-verify strategy in programming RRAMs, similar to that of Flash memory [23]–[25]. More details about RRAMs can be found in [18].

C. PREVIOUS WORKS ON RRAM-BASED FPGA ARCHITECTURE

Previous RRAM-based FPGA studies [7]–[12] mainly account on one/multi-level RRAM-based multiplexers to achieve area, delay and power reduction. In principle, RRAM multiplexers replace both the routing transmission gates and the configuration SRAMs with single RRAM elements. When a RRAM is programmed to LRS/HRS, it propagates/blocks signals, similar to a transmission gate in *on*/*off* state. However, most RRAM-based researches overlook the challenges coming from physical designs, i.e., consider a ideal RRAM, which may lead to a strong bias in the estimation of any performance metric improvements. Recent work [13] has carefully studied the physical design details of RRAM programming structures, and proposes a 4T(ransistor)1R(RAM) programming structure. Previous works [7]–[11] predict that RRAM-based FPGAs can reduce the area by 7-15 percent, increase the performance by 45-58 percent, and save the power consumption by 20-58 percent, compared to SRAM-based FPGAs. However, these architectural improvements are obtained by simply replacing SRAM-based multiplexers in classical FPGA architectures with RRAM-based multiplexers. Very limited work studies the impact on novel RRAM-based FPGA architectures that exploit the circuit-level features of RRAM-based multiplexers. Therefore, it is worthy to investigate specific architectural optimizations for RRAM-based FPGAs that would derive from realistic RRAM-based multiplexer designs.

III. EXPERIMENTAL METHODOLOGY

In this paper, we will base our analysis using a commercial 40 nm technology, whose nominal working voltage is $V_{DD} = 0.9V$. Area is estimated and expressed by the number of minimum width transistors, based on the area model in [32]. Delay results are extracted from electrical simulations by running HSPICE simulator [34]. Datapath logic gates are built with standard logic transistors ($W_{logic}/L_{logic} = 140nm/40nm$, $W_{PMOS,logic}/W_{NMOS,logic} = 1.9$), while programming structures employ I/O transistors ($W_{prog}/L_{prog} = 320nm/270nm$, $W_{PMOS,IO}/W_{NMOS,IO} = 3$). SRAM-based multiplexers are built with two-level structures and transmission gates for best

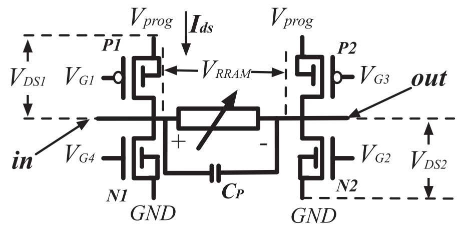


FIGURE 3. Schematic of a 4T1R programming structure.

area-delay product [26]. RRAM-based multiplexers are built with one-level structure and I/O transistors [13]. Electrical simulations use the Stanford RRAM model [33] with following parameters: $R_{LRS} = 2k\Omega$, $R_{HRS} = 27M\Omega$, $I_{set} = 500\mu A$, $V_{set} = V_{reset} = 1.2V$. The parasitic capacitance of a RRAM is considered to be $C_P = 0.02fF$. The considered RRAM parameters are sufficient to guarantee that the RRAM-based circuits are as power efficient as SRAM-based circuits [12]. To determine the size of CB and SB multiplexers, we set channel width to $W = 320$, which is close to the practical number in commercial products [16], [17].

Since each architectural optimization involves different routing architecture parameters, such as $F_{c,in}$, F_s and L, for a fair comparison, we vary a single parameter in each comparison and find a reasonable value for each parameter. Once we find the best value of one parameter, we set it to this value and vary another. All the investigated tile-based FPGA architectures share the Stratix IV-like CLB architecture [35], which contains 10 BLEs, consisting of six-input fracturable LUTs and FFs ($K = 6, N = 10$). We consider a uni-directional routing architecture and the CLB output connection flexibility, $F_{c,out}$, is fixed to 0.1. All the baseline architectures have 40 inputs for each CLB ($I = 40$). Because the local routing is removed in the proposed architecture, we provide 60 inputs for each CLB ($I = K \cdot N = 60$). We will focus on studying the effect of the different architectural modifications on both SRAM-based and RRAM-based FPGAs. Both SRAM-based and RRAM-based implementations of the proposed architecture are then investigated and their benefits are examined by comparing to the baseline SRAM-based and RRAM-based architectures, respectively. We believe that such methodology helps to identify where RRAM FPGAs can be improved beyond SRAM FPGAs. Then, we will discuss the benefits of a properly-optimized RRAM-based FPGA compared to the SRAM counterpart.

We use the VTR flow [37] to evaluate the area, delay, power and channel width of the investigated FPGA architectures. The twenty biggest MCNC [36] and VTR benchmarks [37] suites are logic optimized by ABC [38] and then packed, placed and routed by VPR7. We add a 30 percent slack to the minimum routable channel width W_{min}, in order to simulate a low-stress routing [14]. For a fair comparison, the maximum routing iterations are set to 50 for the classical architecture, while 100 routing iterations are used for the proposed architectures. Indeed, our proposed architecture requires more routing efforts because local routing is removed and more nets have to be routed by the global router.

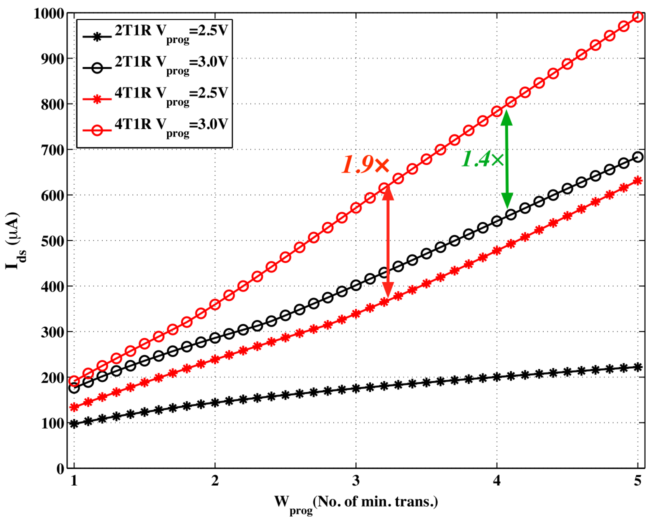


FIGURE 4. A comparison between the programming current driven by 2T1R and 4T1R programming structure. $1W_{prog} = 320$ **nm.**

IV. HIGH-PERFORMANCE RRAM-BASED MULTIPLEXER

In this section, we investigate the circuit design details of RRAM-based multiplexers. We review the benefits of a 4T1R programming structure and discuss the proposed multiplexing structure and possible circuit-level optimizations.

A. 4T1R PROGRAMMING STRUCTURE

The programming structure is one of the most critical considerations to make during the design of RRAM-based multiplexers because it controls the quality of set/reset process, and therefore its electrical parameters. A good programming structure should be able to drive a programming current as large as possible while keeping the introduced parasitics as small as possible. Figure 3 depicts the schematic of a 4T1R programming structure, where two pairs of n-type and p-type transistors are employed to set/reset a RRAM. To enable a set process, transistors **P1** and **N2** are turned *on*, while, in a reset process, transistors **P2** and **N1** are turned *on*. During regular operation, all the programming transistors are turned *off*. Compared to the 2T(ransistor)1R(AM) and 2T(ransmission-gates)1R(AM) programming structures, which are used in previous works [7]–[12], the 4T1R programming structure improves the programming current for a given occupied area by 1.4×, as shown in Figure 4. The efficiency of 4T1R programming structure comes from that it guarantees balanced source-to-drain voltages V_{DS} of both n-type and p-type transistors, maximizing the programming current. Note that the programming voltage V_{prog} should be much larger than V_{DD} in order to ensure sufficient V_{DS} of programming transistors. In order to support such high programming voltage and avoid disturbance with regular logic operations, we have to employ I/O transistors for programming purpose. To maximize the V_{DS}, we can use a V_{prog} close to the breakdown voltage of I/O transistors. As shown in Figure 4, compared to the nominal voltage of I/O transistors ($V_{prog} = 2.5$ V), the higher $V_{prog} = 3.0$ V can further improve programming current by 1.9×. A significant

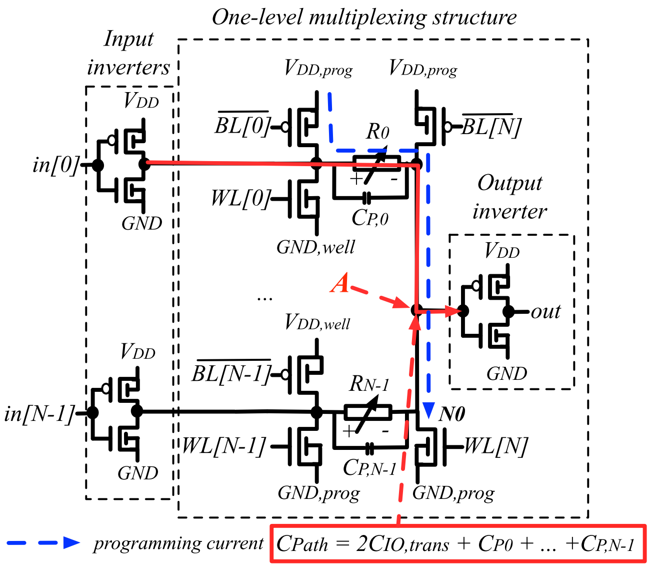


FIGURE 5. Schematic of a $n:1$ one-level 4T1R-based multiplexers.

increase in programming current can lead to lower programmed R_{LRS}, further increasing the performance of RRAM-based multiplexers. More discussion of 4T1R programming structure can be found in [13].

B. MULTIPLEXING STRUCTURE

Figure 5 illustrates the schematic of a one-level $n:1$ 4T1R-based multiplexer. The transistors of the 4T1R programming structures can be effectively shared among RRAMs in the one-level structure at the output node A. All the RRAMs are programmed sequentially. For example, to set a RRAM R_0, control lines $\overline{BL}[0]$ and $WL[N]$ are enabled while the others are disabled, allowing programming current (highlighted in blue dash lines) to flow across the RRAM.

Note that whatever the input size is, there are only one pair of programming transistors at the output node A in Figure 5. The parasitic capacitance of RRAMs ($C_0, \ldots, C_{N-1}$) are typically smaller than that of transistors. This reveals an outstanding feature of RRAM-based multiplexers: the delay of RRAM multiplexer scales better with the input size n than SRAM-based ones. This feature indicates that RRAM-based FPGA architectures can prefer the use of larger multiplexers rather than smaller ones.

C. PROGRAMMING TRANSISTOR SIZING TECHNIQUE

The programming transistor sizing technique is a circuit-level optimization method proposed in [11] to guarantee the best *Power-Delay Product* (PDP) of a RRAM-based multiplexer. Increasing the sizes of programming transistors leads to large programming current meanwhile which in turn leads to low R_{LRS} values, but it also introduces large parasitic capacitances to the datapath. When the increase in parasitic capacitances becomes larger than the reduction of R_{LRS}, the delay and power of a RRAM-based multiplexer would only get worse. Therefore, there exists a best achievable delay for a RRAM-based multiplexer, which comes from a trade-off between the

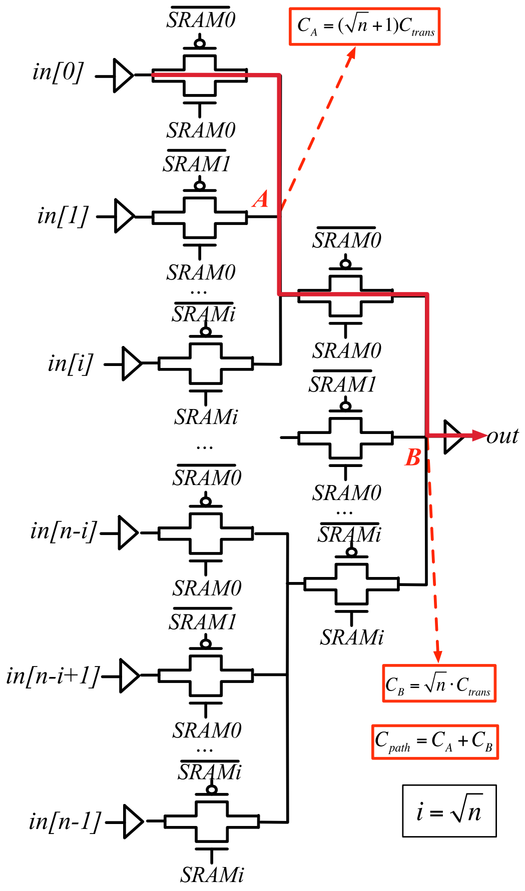


FIGURE 6. Schematic of a $n:1$ two-level SRAM-based multiplexers.

introduced parasitic capacitances and the obtained R_{LRS}. Figure 7 depicts the PDP of a 50-input one-level RRAM-based multiplexers obtained by sweeping the size of programming transistors from 1 W_{prog} to 2 W_{prog}. The minimum sized programming transistors produce the best PDP while larger programming transistors lead to an overhead of up to 15 percent. All the one-level RRAM-based multiplexers considered in this paper, varying with input size, have a similar conclusion as shown in Figure 7. In rest of this paper, if not specified, we consider minimaly sized programming transistors in 4T1R-based multiplexers.

Figure 8 shows the delay of optimally-sized 4T1R-based multiplexers compared to the SRAM-based counterparts. All

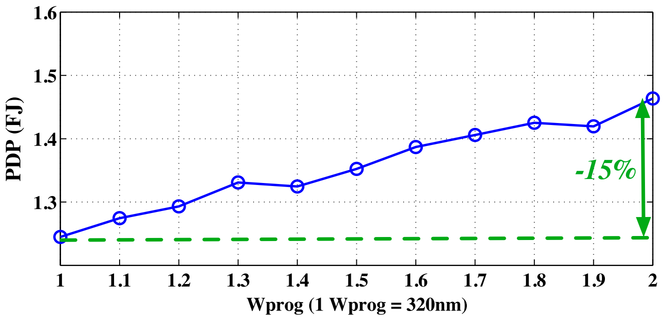


FIGURE 7. Impact of the size of programming transistors on the PDP of a 50:1 one-level 4T1R multiplexer.

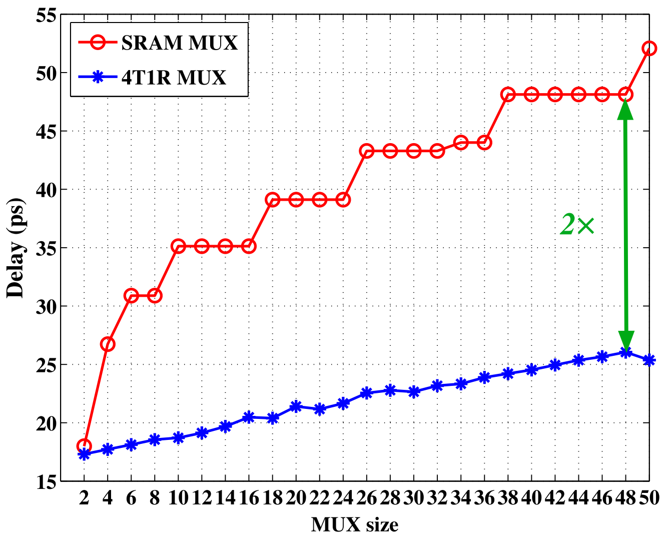


FIGURE 8. Comparison of delay between SRAM-based multiplexers and 4T1R-based multiplexers.

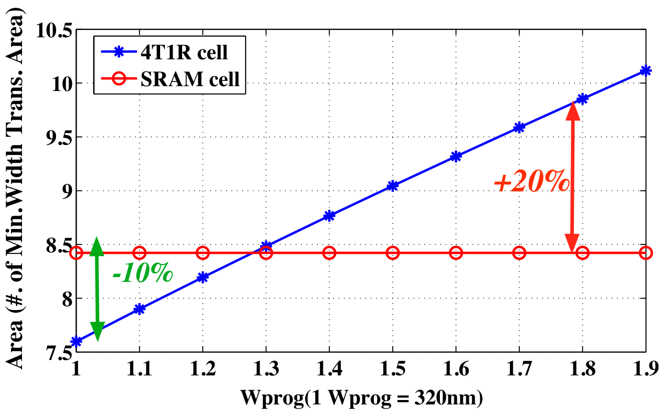


FIGURE 9. Area comparison between a 4T1R cell and a SRAM cell under different W_{prog}.

the multiplexers drive a fan-out equivalent to the smallest inverter. Delay of 4T1R-based multiplexers scales better with the number of inputs thanks to their smaller parasitic capacitances than those of SRAM-based multiplexers. For large multiplexers, i.e., input size is 50, 4T1R-based implementations can reach 2× delay improvement as compared to SRAM-based. Note that even for the smallest multiplexers (input size is 2), 4T1R-based multiplexer is as delay efficient as SRAM-based. However, in modern FPGA architectures, most of the multiplexers are large, i.e., local routing and connection blocks, and very few 2:1 multiplexers are deployed, i.e., BLE output selector. Therefore, the FPGA architectures can benefit from the outstanding performance of 4T1R-based multiplexers. Table 1 compares the delay of SRAM-based and 4T1R-based multiplexer in their architectural context, i.e., by considering realistic sizing and loads. In high fan-in and low fan-out condition, such as local routing, the 4T1R-based multiplexer can achieve 48 percent reduction in delay. In contrast, when fan-in is low and fan-out is high, e.g., the BLE output selector, 4T1R-based multiplexer guarantees a similar performance level as an SRAM-based implementation. The area of 4T1R programming structures is also impacted by the sizes of programming transistors, which also determines the amount of set/reset currents for a RRAM. We assume the transistor area of a 4T1R programming structure composed of two pairs of *p*-type and *n*-type programming transistors as shown in Figure 3 (RRAM is fabricated above transistors and does not occupy transistor area) while a SRAM cell consists of a 6-transistor SRAM and a transmission gate. The area of a programming transistor is considered to be 40 percent larger than a standard logic transistor according to layout rules. Figure 9 compares the area between a 4T1R programming structure and a SRAM cell by sizing programming transistors, expressed in terms of the number of Minimum Width Transistor Area. Since the programming transistors are typically minimum sized for best PDP as illustrated in Figure 7, a 4T1R programming structure leads 10 percent less area than a SRAM cell.

D. LEAKAGE ISSUE

As explained in [12], R_{HRS} of RRAMs significantly influence the leakage power of RRAM-based multiplexers. When a small R_{HRS}, i.e., $= 20M\Omega$, is used, the leakage power of 4T1R-based multiplexer is 5× larger than SRAM-based, similar to the conclusion in [12]. On the one hand, a trivial route to suppress the leakage power overhead is to technologically increase R_{HRS} to $200M\Omega$. On the other hand, the leakage overhead observed at the multiplexer level must be put into perspective. Indeed, the overhead is in fact non-existent in the architecture context, because: (a) in high fan-out condition, the leakage power of a multiplexer is much smaller than its large output buffers; (b) leakage power is negligible as compared to dynamic power and 4T1R-based multiplexers consume less dynamic power than SRAM-based thanks to their lower parasitic capacitances. (c) RRAM enables efficient power saving and instant-on/normally-off behavior [39]. As a

TABLE 1. Delay comparison between SRAM-Based and RRAM-Based routing multiplexers.

Multiplexer Location	Input Size	fan-out	SRAM-based MUX (ps)	4T1R-based MUX (ps)	Improvements
Local Routing	80	1	57.7	30.4	−48%
BLE output selector	2	70	38.8	42.2	+11%
Connection block	48	60	76.0	48.2	−36%
Switch block	4	124[1]	57.8	49.6	−14%

Output buffers are considered and sized according to the fan-outs of routing multiplexers in architecture.
[1]*The fanout includes the parasitics of long metal wires driven by SBs.*

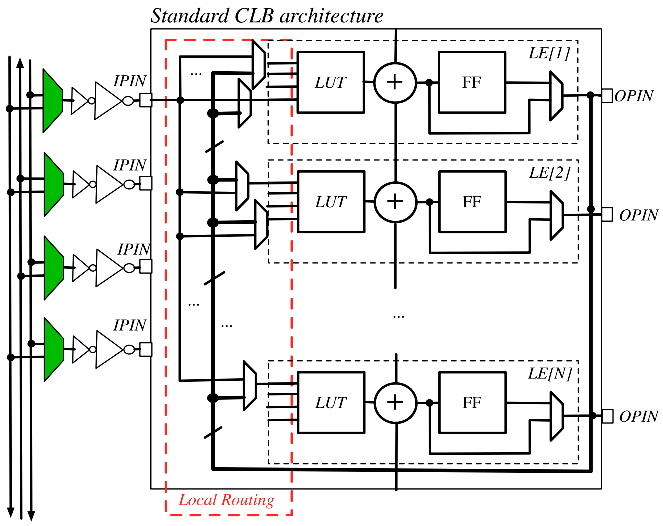


FIGURE 10. Classical interconnection from routing tracks to LUT inputs.

result, according to [12], although $R_{HRS} = 20M\Omega$ introduces a high leakage overhead (5×) to 4T1R-based multiplexers, the total power of the resulting RRAM-based FPGA is similar to its SRAM-based counterpart. Therefore, we believe that the considered $R_{HRS} = 27M\Omega$ in this paper is large enough to prevent any power issues.

V. ARCHITECTURAL OPTIMIZATIONS

As discussed in Section IV, there is a strong need to optimize FPGA architectures, in order to exploit the high-performance of 4T1R-based multiplexers. In this section, we propose three architectural optimizations: (1) The realization of a Unified Connection Block; (2) The increase of Switch Blocks capacity; (3) The decrease of the best length of routing wire; For each architectural optimization, we study its impact on both SRAM-based and RRAM-based FPGAs.

A. UNIFIED CONNECTION BLOCK

In SRAM-based FPGA architectures, a routing track has to pass through a CB multiplexer and a local routing multiplexer before reaching a LUT input, as shown in Figure 10. Such routing architecture efficiently reduces the number of CB multiplexer to be used. Indeed, the number of the inputs of a CLB, typically $I = K(N+1)/2$, is smaller than the total number of LUTs inputs, $K \cdot N$, where K is the input size of a LUT and N is the number of BLEs in a CLB. However, it requires tapered buffers at the outputs of CB multiplexers, in order to drive the high fan-outs. Take the example in Figure 10, each CB multiplexer has to drive $K \cdot N$ local routing multiplexers. The use of large tapered buffers potentially increase the delay from a routing track to a LUT input. This situation is extremely inefficient for RRAM-based FPGAs since the delay of a tapered buffer may be far larger than the delay of the RRAM-based multiplexer itself.

Therefore, we propose that RRAM-based FPGA should use a one-level RRAM-based crossbar to provide

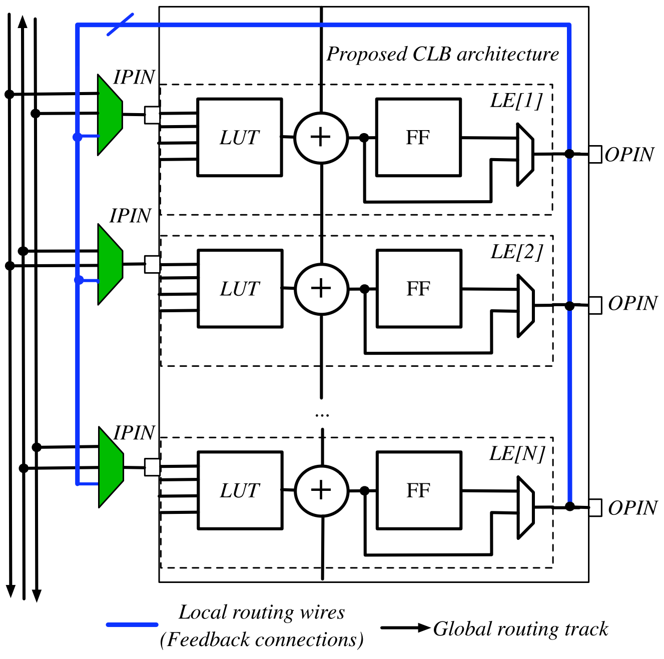


FIGURE 11. Proposed interconnection from routing tracks to LUT inputs.

interconnections between routing tracks and LUT inputs, as illustrated in Figure 11. Note that feedback connections are also resolved by the unified Connection Block. The proposed routing architecture is well suited to RRAM-based multiplexers for three reasons: (a) Each CB multiplexers now has a unique fan-out, and tapered buffers can be avoided; (b) Only one large multiplexer interconnects between a routing track to a LUT input; Both routing delay and feedback delay can be significantly reduced when a RRAM-based multiplexer is used; (c) The number of inputs of a CLB is increased to $I = K \cdot N$, which can potentially lead to a total area reduction even for SRAM-based FPGAs [27]; Since RRAM-based multiplexers require a smaller footprint, the area reduction could be more significant.

The proposed routing architecture requires to redefine the best fraction of routing tracks can be reached by each CB multiplexer, $F_{c,in}$. Note that in the classical architecture ($F_{c,in} = 0.15$), all the nets mapped to the inputs of a CLB are different because the local routing can connect a net from a CLB input to multiple LUTs. The proposed architecture may have a net mapped to multiple CLB inputs due to the absence of local routing. Therefore, we need to increase $F_{c,in}$ to allow more CLB inputs to be reached by a single routing track, to compensate the potential loss in routability. In an FPGA tile, all the LUT inputs are connected to the right and bottom sides of a CLB. Each LUT has $K/2$ input connected to the right/bottom side of a CLB. To ensure that different LUT inputs can be connected from a common routing track, $F_{c,in}$ should be at least $2/K$. Figure 12 depicts such an example when $K = 6$. Input $in0$ of $LUT0$ and input $in0$ of $LUT1$ can be reached by the same track $Track0$. Note that there is no need to allow two inputs of the same LUT to share a routing track. The case where two inputs of a LUT share the same

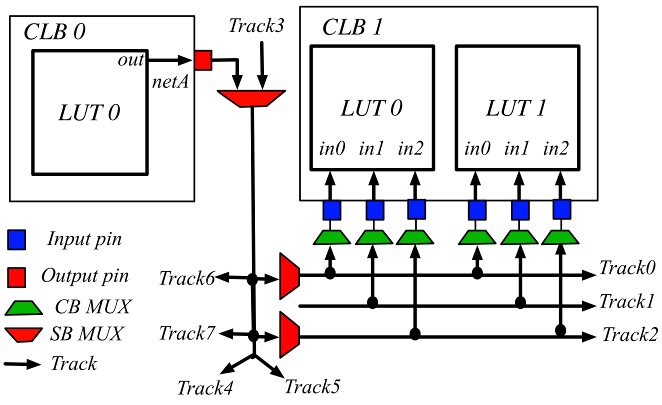
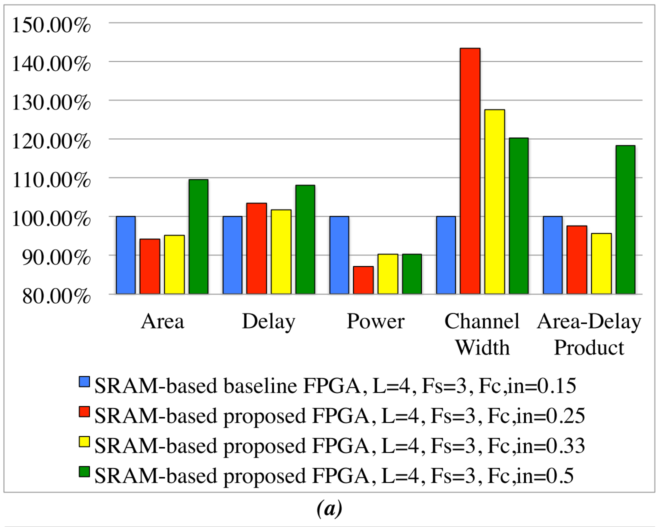


FIGURE 12. An illustrative example of the proposed routing architecture ($K = 6$) with $F_{c,in} = 0.33$ and $F_s = 6$.

net can never happen because the inputs of a LUT are naturally logic equivalent.

By considering architecture parameters $K = 6$, the proposed architecture requires $F_{c,in}$ to be at least 0.33, in order to ensure routability. In this part, we sweep $F_{c,in}$ to examine the best $F_{c,in}$ for the proposed architecture.

Figures 13(a) and 13(b) show normalized area, delay, power and channel width of SRAM-based and RRAM-based proposed architectures with $F_{c,in} = \{0.15, 0.25, 0.33, 0.5\}$, when compared to baseline architectures respectively. The SRAM-based proposed architecture with $F_{c,in} = 0.33$ produces a slightly better area-delay product (−4 percent) than the classical architecture, but performs worse (+2 percent) in delay. In contrast, the RRAM-based proposed architecture with $F_{c,in} = 0.33$ reduces delay by 3 percent and area-delay product by 15 percent, when compared to the classical architecture. In either SRAM-based or RRAM-based FPGAs, the proposed architecture with $F_{c,in} = 0.33$ produces the best area-delay product. Note that we see a 5 percent area reduction in both SRAM-based and RRAM-based proposed architectures when $F_{c,in} = 0.33$, which is close to the conclusion of literature [27]. The proposed architecture with varying $F_{c,in}$ reduces power by 10-13 percent for SRAM-based and RRAM-based FPGAs. In the classical architecture, there are two-stages of multiplexers (local routing and classical connection blocks) that lead to four levels of transmission gates between the routing tracks and the LUTs. However, in the proposed unified connection block, there is only one-stage of multiplexers (two-levels of transmission gates) between the routing tracks and the LUTs, contributing to power efficiency. Besides, the unified connection blocks eliminates the need for intermediate buffers between the local routing and the connection block, which further reduce the power. Channel width overheads are observed in both SRAM-based and RRAM-based proposed architectures, because their routability is lower than their baselines due to the absence of local routing. However, these overheads can be potentially eliminated because the routability can be significantly improved when we increase F_s and decrease L. In terms of the best overall performance, we consider $F_{c,in} = 0.33$ for the proposed FPGA architectures in the rest of this paper.



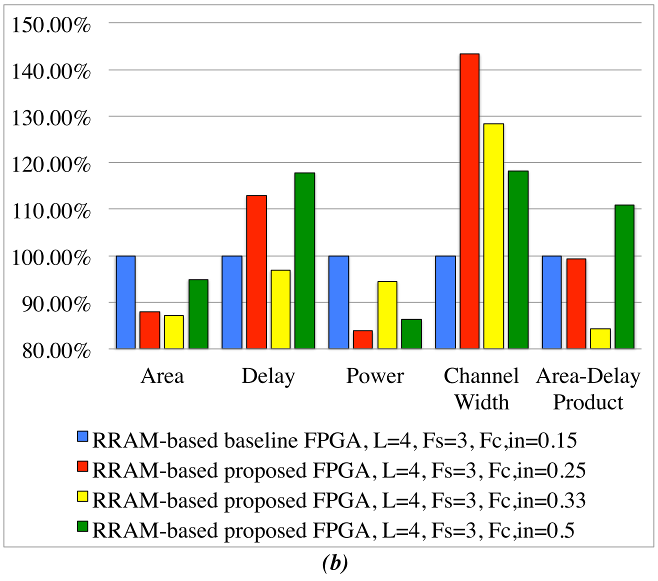


FIGURE 13. Normalized average area, delay, power and channel width of baseline and proposed architecture by sweeping $F_{c,in}$: (a) SRAM-based architectures; (b) RRAM-based architectures.

Figure 14 compares the tile area of a classical FPGA architecture ($I = 40, F_{c,in} = 0.15$) and the proposed RRAM FPGA architecture ($I = W \cdot F_{c,in}, F_{c,in} = 0.33$) for a sweeping channel width W from 100 to 350. Note that the input size of local routing multiplexers in traditional SRAM FPGAs is fixed for every W, while that of proposed RRAM FPGAs is directly related to W. When a small W, e.g. $= 100$, is used, the size of the local routing multiplexers in the proposed RRAM FPGAs is smaller than for a classical FPGA architecture. Therefore, when $W < 300$, the proposed RRAM FPGA architecture benefits up to 36 percent area reduction as compared to classical FPGA architecture. When $W > 300$, the input size of multiplexers in the proposed RRAM FPGAs becomes larger, leading to a 9 percent area overhead when $W = 350$. The considered $W = 320$ in this paper promises that the proposed RRAM FPGAs is as area efficient as classical SRAM FPGAs.

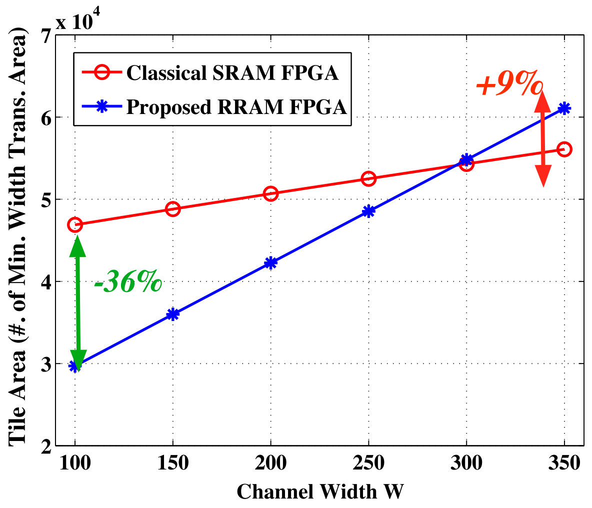


FIGURE 14. Tile area comparison between a traditional FPGA architecture and the proposed RRAM FPGA architecture for different channel width W.

B. INCREASE CAPACITY OF SB MUXES

Since RRAM-based multiplexer is more delay-efficient than SRAM-based multiplexer, the connection flexibility parameter of *Switch Block* F_s can be increased. Classical FPGA architectures typically set $F_s = 3$, where each routing track on one side of a SB can reach three other routing tracks on different sides of a SB. In SRAM-based FPGAs, $F_s = 3$ promises the best area-delay product [29]. Indeed, a larger F_s can improve the routability but it may produce area and delay overhead coming from the larger SB multiplexers to be used. However, considering RRAM-based routing architecture, the delay overhead is no longer a concern thanks to the advantage of RRAM multiplexers. Therefore, a larger F_s, i.e. $= 6$, can considered, where a routing track can drive six different tracks, as shown in Figure 12 with *Track*3. Note that a large F_s significantly improves the routability of the proposed routing architecture. Take the example of Figure 12 where *netA* is routed through *Track*3. If $F_s = 3$, *Track*3 can only drive *Track*0, *Track*4 and *Track*6. If *Track*0 is not available, the output of *LUT*0 has to seek for another routing track by increasing the channel width. If $F_s = 6$, *Track*3 can reach both *Track*0 and *Track*2. When *Track*0 is occupied by another net, *Track*3 can easily use *Track*2 to route *netA*.

We sweep F_s to determine its best value for the proposed architecture. Figures 16(a) and 16(b) show normalized average area, delay, power and channel width of SRAM-based and RRAM-based proposed architectures with $F_s = \{3, 6, 9\}$, when compared to the baseline architectures, respectively. The proposed RRAM-based architectures can benefit larger delay reduction (-7 percent) than SRAM-based (-4 percent), because RRAM-based multiplexers are more delay efficient for the unified connection block. However, $F_s > 3$ introduces larger SB multiplexers, which potentially increases the area of both SRAM-based and RRAM-based proposed architectures. On the other hand, larger SB multiplexers improve the flexibility of the routing architecture and reduce the number of necessary SB multiplexers, as explained in Figure 12. In the end, the proposed architecture can maintain the same power efficiency as baseline SRAM one. Therefore, $F_s = 6$ produces the best area-delay-power product for both SRAM-based and RRAM-based proposed architectures. Note that, even when $F_s = 9$, RRAM-based proposed architecture leads to a 8 percent delay reduction thanks to its RRAM-based multiplexer, while, the SRAM-based proposed architecture has a 5 percent delay overhead. As a large F_s boosts the routability, a 20 percent channel width reduction is achieved in both SRAM-based and RRAM-based proposed architectures, as compared to those with $F_s = 3$. In terms of the best overall performance, we consider $F_s = 6$ for the proposed FPGA architectures in the rest of this paper.

C. SMALLER BEST LENGTH WIRE < 4

In FPGA architectures, a length-L wire is a wire that spans across L CLBs [14]. As illustrated in Figure 15(a), a length-L wire is driven by an output of $CLB[0]$ and ends at $CLB[L-1]$. All the CLBs and SBs along the length-L wire can be directly routed from the driving output of $CLB[0]$. When only one type of wires is allowed to be used in an FPGA, the type of length-L wires that produces best area-delay product is called *best single wire length*. Commercial FPGAs typically provide different types of wires, i.e., length-1 for short connections and length-8 for long connections. However, best single wire length is useful in deciding which type of wires should be predominant within the architecture.

Length-4 wires are the best choice for classical SRAM-based FPGA architectures ($F_{c,in} = 0.15, F_s = 3$) [14]. Betz *et al.* show that a length-4 wire is faster than shorter wires in terms of delay per logic block ($= T_{delay,wire}/Length$). In other words, for a routing path spanning X CLBs, length-4 wires

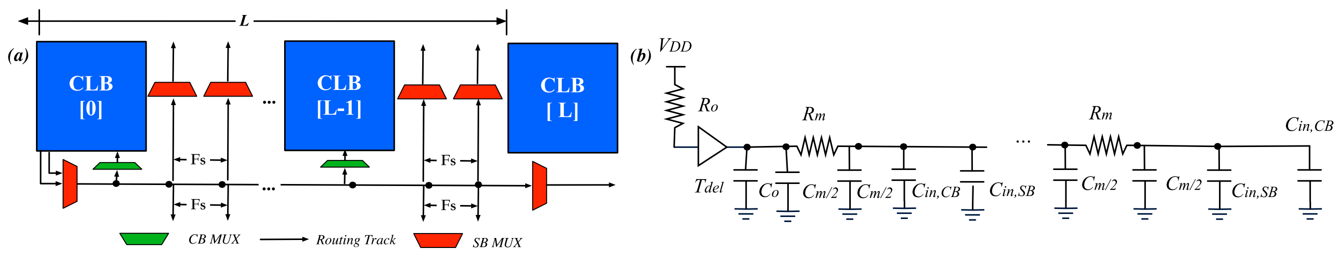
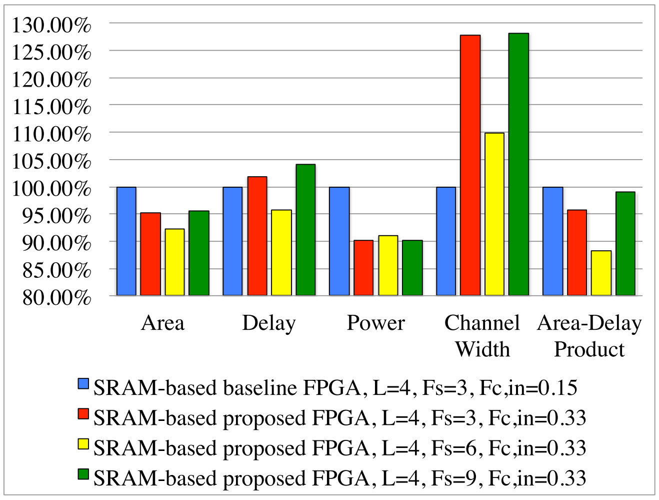
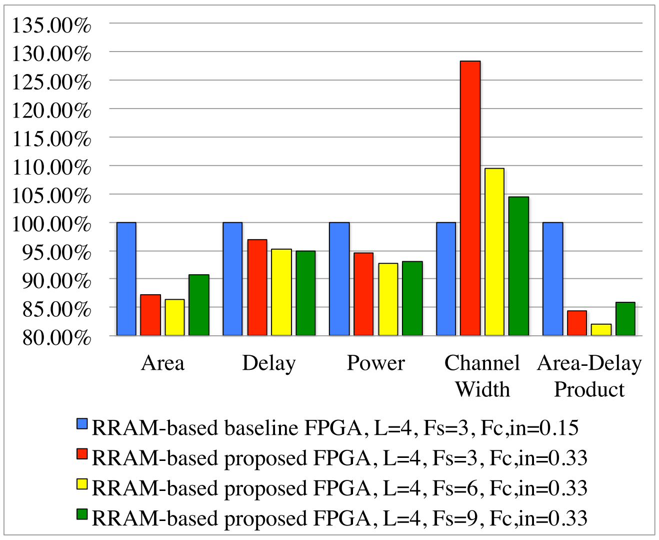


FIGURE 15. (a) Driver multiplexer and fan-outs of a Length-L wire; (b) Equivalent RC model of a Length-L wire.



(a)



(b)

FIGURE 16. Normalized average area, delay, power and channel width of baseline and proposed architectures by sweeping F_s: (a) SRAM-based architectures; (b) RRAM-based architectures.

promise the best average delay. Indeed, when there is a routing path with $X < 4$, shorter wires such as length-1 or length-2 will give better delay. However, for a routing path with $X \geq 4$, multiple cascaded length-4 wires are faster than not only any length-X ($X > 4$) wire but also multiple cascaded length-1 or length-2 wires. Therefore, on average, length-4 wires provide the best trade-off between short and long connections.

In SRAM-based FPGAs, why long length wires, such as length-4 wires, are preferred is established on the fact that the delay of a SB multiplexer is larger than a long metal wire across a logic block. However, RRAM-based multiplexers are more delay efficient and can be even faster than a long metal wire. Therefore, as the cost function between a SB multiplexer and a long metal wire has been twisted, the best single wire length L should be revisited. Figure 15(a) illustrates the different elements composing a length-L wire, while Figure 15(b) shows the extracted RC model. We use Elmore delay [31] to estimate the delay per logic block of a Length-L wire

$$\begin{aligned} T_{delay,wire}/L &= \frac{1}{L}\sum_{i=0}^{L-1} R_i \sum_{j=i}^{L-1} C_j \\ &= L \cdot \frac{R_m C_m}{2} + \frac{1}{L} \cdot (T_{del} + R_o C_o - 2R_m C_{SB} - 2R_m C_{CB}) \\ &\quad + R_m(C_{SB} + C_{CB} - C_m) + R_o(C_m + C_{SB} + C_{CB}), \end{aligned} \tag{1}$$

where R_m and C_m are the resistance and capacitance of a metal wire spanning a logic block, respectively, T_{del} represents the intrinsic delay of a SB multiplexer, R_o and C_o denote the equivalent resistance and capacitance of the tapered buffer that drives the metal wire, respectively, C_{SB} and C_{CB} are the equivalent input capacitance of each SB and CB, respectively. According to (1), there exists a $L_{optimal}$ which guarantees the minimum $T_{delay,wire}/L$

$$L_{optimal} = \frac{(T_{del} + R_o C_o - 2R_m C_{SB} - 2R_m C_{CB})}{2R_m C_m}. \tag{2}$$

Note that C_{SB} and C_{CB} are related to F_s and $F_{c,in}$ respectively

$$\begin{aligned} C_{SB} &= F_s \cdot C_{in} \\ C_{CB} &= W \cdot F_{c,in} \cdot C_{in}. \end{aligned} \tag{3}$$

In the proposed RRAM-based routing architecture, where both F_s and $F_{c,in}$ increased and T_{del} decreased thanks to RRAM-based multiplexer, $L_{optimal}$ will definitely decrease. In addition, the tile area of the proposed architecture may be slightly larger than the classical architecture because of the F_s and $F_{c,in}$ increases, leading to an increased R_m and C_m. This would further decrease the $L_{optimal}$. Therefore, the best single wire length of the proposed routing architecture will be smaller than 4. When a smaller L (< 4) is used, previous work [14] show that the routability is improved significantly. Therefore, the proposed RRAM-based routing architecture can achieve routability improvement without delay overhead.

We sweep L to determine its best value for the proposed architecture. Figures 17(a) and 17(b) show normalized average area, delay, power and channel width of SRAM-based and RRAM-based proposed architectures with $L = \{1, 2, 4\}$, when compared to the baseline architectures, respectively. In SRAM-based architectures, whatever F_s is, length-4 wires achieve the best delays and area-delay-power products. However, the proposed RRAM-based architecture with length-2 wires promises the best delay (-11 percent) and also the best area-delay-power product (-24 percent), thanks to its better routability and lower routing congestion. As L is reduced from 4 to 2, we see a 26 percent channel width reduction because short wires are more flexible. Conversely, length-1 wires have the smallest channel width but more SB multiplexers have to be used in long routing paths. Therefore, we see significant area and power overhead. Length-4 wires

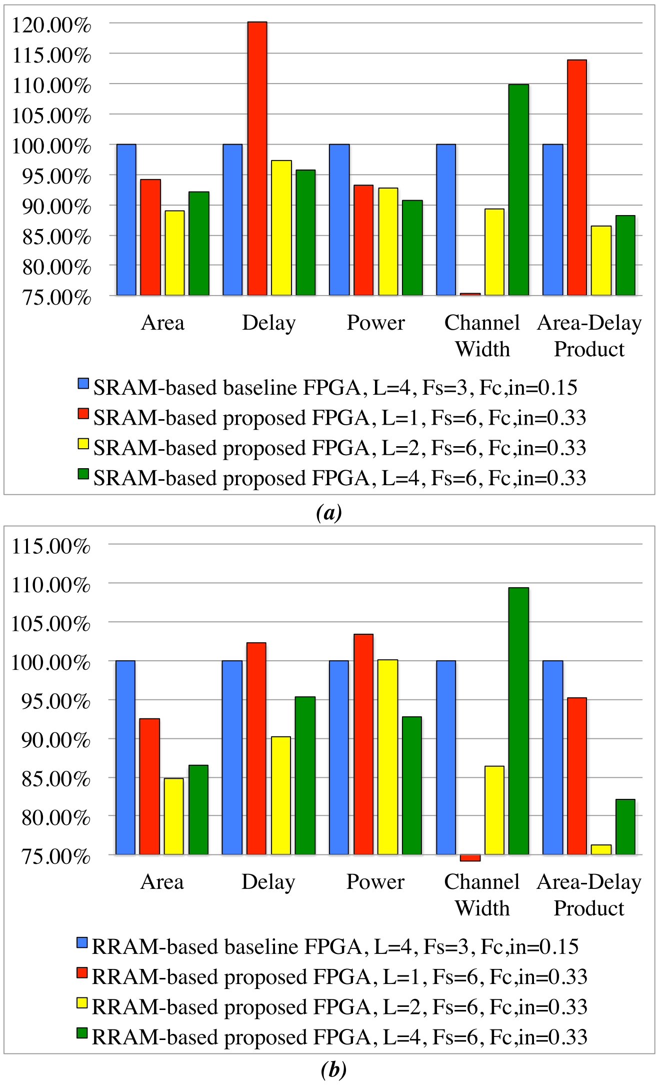


FIGURE 17. Normalized average area, delay, power and channel width of baseline and proposed architectures by sweeping L: (a) SRAM-based architectures; (b) RRAM-based architectures.

guarantee the best power results since less multiplexers are required in a SB compared to the case where length-2 and length-1 wires are used. In terms of the best overall performance, $L = 2$ is the best single wire length for the proposed FPGA architecture.

VI. OVERALL IMPROVEMENTS

In Section V, we determined that $F_{c,in} = 0.33, F_s = 6$ and $L = 2$ produce the best performances for the proposed FPGA architecture. In this section, we make a general comparison between SRAM-based and RRAM-based FPGAs architectures. Figure 18 shows the area, delay, power and channel width of three FPGA architectures: (1) SRAM-based FPGA with classical architecture; (2) RRAM-based FPGA with classical architecture; (3) RRAM-based FPGA with architectural optimizations. When implemented with classical architecture, RRAM-based FPGAs improve the delay by 32 percent and the area by 17 percent, as compared to SRAM-based FPGAs, thanks to the delay efficiency of the RRAM-based routing elements. By properly optimizing the architecture, RRAM-based FPGAs can further reduce the area by 15 percent, the delay by 10 percent and the channel width by 13 percent, leading to a total improvement of 38 percent in delay and 26 percent in area compared to an SRAM-based FPGA architecture. In terms of *Area-Delay Product* and *Delay-Power Product*, the proposed RRAM-based FPGA architecture brings a reduction of 57 and 38 percent respectively. Note that the proposed LB architecture eliminates the complex routing efforts during packing stage, which is required for the local routing in a classical architecture. Although the proposed architecture increase the overall routing (local and global) efforts by 2.4× on average, we believe that the runtime of EDA flow can be significantly reduced by replacing the default packer in VPR with a lighter packer.

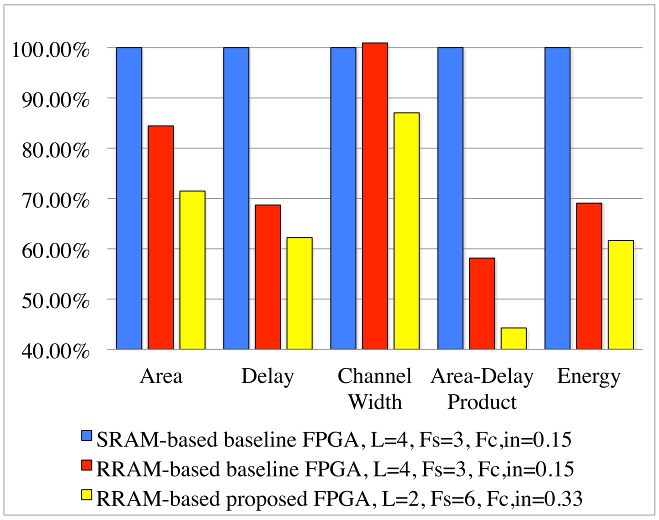


FIGURE 18. Normalized average area, delay, energy and channel width of baseline and proposed architectures: (a) baseline SRAM-based architectures; (b) baseline RRAM-based architectures; (c) proposed RRAM-based architectures.

VII. CONCLUSION

In this paper, we first investigated the circuit design aspects of RRAM-based multiplexers and identified that their delay scales better with input size than SRAM-based implementations. In other words, large RRAM-based multiplexer can be as delay efficient as the smallest ones. To exploit this advantage, we propose three architectural optimizations for RRAM-based FPGAs: (a) The traditional CB and local routing are replaced with a unified CB, leading to ultra-fast interconnection from routing tracks to LUT inputs; (b) The CB connectivity parameter $F_{c,in}$ should be at least 0.33 to ensure routability, while the SB connectivity parameter F_s can be increased to achieve routability improvements without delay overhead; (c) The best single wire length L is reduced, leading to better routability. We study the best values of $F_{c,in}$, F_s and L in terms of area, delay, power and channel width. Experimental results show that a RRAM-based FPGA properly optimized should employ ($F_{c,in} = 0.33$, $F_s = 6$ and

$L = 2$) to achieve optimal performances. When considering a classical architecture for both SRAM-based and RRAM-based FPGAs, an improvement of 32 percent in delay and 17 percent in area can be achieved. The proposed architectural optimizations can further reduce area by 15 percent, delay by 10 percent and channel width by 13 percent, leading to a total improvement of *Area-Delay Product* by 57 percent and *Delay-Power Product* by 38 percent.

ACKNOWLEDGMENTS

This work was supported by the Swiss National Science Foundation under the project number 200021-146600.

REFERENCES

[1] S. S. Sheu, *et al.,* "A 4Mb embedded SLC resistive-RAM macro with 7.2ns read-write random-access time and 160ns MLC-access capability," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest Tech. Papers*, 2011, pp. 200–202.

[2] A. Kawahara, *et al.*, "An 8Mb multi-layered cross-point ReRAM macro with 443MB/s write throughput," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest Tech. Papers*, 2012, pp. 432–434.

[3] T. Y. Liu, *et al.*, "A 130.7mm^2 2-layer 32Gb ReRAM memory device in 24nm technology," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest Tech. Papers*, 2013, pp. 210–211.

[4] R. Fackenthal, *et al.*, "19.7 A 16Gb ReRAM with 200MB/s write and 1GB/s read in 27nm technology," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest Tech. Papers*, 2014, pp. 338–339.

[5] M. F. Chang, *et al.*, "17.5 A 3T1R nonvolatile TCAM using MLC ReRAM with sub-1ns search time," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest Tech. Papers*, 2015, pp. 1–3.

[6] Y. Liu, *et al.*, "4.7 A 65nm ReRAM-enabled nonvolatile processor with 6× reduction in restore time and 4× higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest Tech. Papers*, 2016, pp. 84–86.

[7] S. Tanachutiwat, M. Liu, and W. Wang, "FPGA based on integration of CMOS and RRAM," *IEEE Trans. Very Large Scale Integr.*, vol. 19, no. 11, pp. 2023–2032, Nov. 2010.

[8] J. Cong and B. Xiao, "mrFPGA: A novel FPGA architecture with memristor-based reconfiguration," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, 2011, pp. 1–8.

[9] J. Cong and B. Xiao, "FPGA-RPI: A novel FPGA architecture with RRAM-based programmable interconnects," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 4, pp. 864–877, Apr. 2014.

[10] P.-E. Gaillardon, D. Sacchetto, S. Bobba, Y. Leblebici, and G. D. Micheli, "GMS: Generic memristive structure for non-volatile FPGAs," in *Proc. IEEE/IFIP 20th Int. Conf. VLSI Syst.-on-Chip*, 2012, pp. 94–98.

[11] X. Tang, P.-E. Gaillardon, and G. D. Micheli, "A high-performance low-power near-Vt RRAM-based FPGA," in *Proc. IEEE Int. Conf. Field-Program. Technol.*, 2014, pp. 207–215.

[12] X. Tang, P.-E. Gaillardon, and G. D. Micheli, "Accurate power analysis for near-V_t RRAM-based FPGA," in *Proc. IEEE 25th Int. Conf. Field Program. Logic Appl.*, 2015, pp. 174–177.

[13] X. Tang, G. Kim, P.-E. Gaillardon, and G. D. Micheli, "A study on the programming structures for RRAM-based FPGA architectures," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 63, no. 4, pp. 503–516, Apr. 2016.

[14] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Berlin, Germany: Springer, 1998.

[15] M. Hutton, *et al.*, "Improving FPGA performance and area using an adaptive logic module," in *Proc. 14th Int. Conf. Field Program. Logic Appl.*, 2004, pp. 135–144.

[16] Altera Corporation, "Stratix 10 Advance Information Brief," Jul. 2015, [Online]. Available: https://www.altera.com/en_US/pdfs/literature/hb/stratix-10/S10_TX_FPGA_AIB.pdf

[17] Xilinx Corporation, "Virtex-7 User Guide DS180 (v1.17)," May 2015. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf

[18] H.-S. P. Wong, *et al.*, "Metal-oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.

[19] J. Sandrini, *et al.*, "Co-design of ReRAM passive crossbar arrays integrated in 180nm CMOS technology," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 3, pp. 339–351, Sep. 2016.

[20] Y. Wu, "Resistive switching random access memory (RRAM)—Scaling, materials, and new application," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, 2013.

[21] S. Yu, B. Gao, Z Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using RRAM synaptic devices with sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling," in *Proc. IEEE Int. Electron Devices Meeting*, 2012, pp. 239–242.

[22] F. M. Puglisi, P. Pavan, L. Larcher, and A. Padovani, "Analysis of RTN and cycling variability in H_fO_2 RRAM devices in LRS," in *Proc. 44th Eur. Solid State Device Res. Conf.*, 2014, pp. 246–249.

[23] A. Levisse, B. Giraud, J. P. Noël, M. Moreau, and J. M. Portal, "SneakPath compensation circuit for programming and read operations in RRAM-based CrossPoint architectures," in *Proc. IEEE Non-Volatile Memory Technol. Symp.*, 2015, pp. 1–4.

[24] H. Aziza, M. Bocquet, M. Moreau, and J.-M. Portal, "A built-in self-test structure (BIST) for resistive RAMs characterization: Application to bipolar OxRRAM," *J. Solid-State Electron.*, vol. 103, pp. 73–78, 2015.

[25] F. M. Puglisi, C. Wenger, and P. Pavan, "A novel program-verify algorithm for multi-bit operation in H_fO_2 RRAM," *IEEE Electron Device Lett.*, vol. 36, no. 10, pp. 1030–1032, Oct. 2015.

[26] E. Lee, G. Lemieux, and S. Mirabbasi, "Interconnect driver design for long wires in field-programmable gate arrays," *J. Signal Process. Syst.*, vol. 51, no. 1, pp. 57–76, Apr. 2008.

[27] W. Feng and S. Kaptanoglu, "Designing efficient input interconnect blocks for LUT clusters using counting and entropy," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 1, no. 1, Mar. 2008, Art. no. 6.

[28] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.

[29] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and single-driver wires in FPGA interconnect," in *Proc. IEEE Int. Conf. Field-Program. Technol.*, 2004, pp. 41–48.

[30] D. Lewis, *et al.*, "Architectural enhancements in Stratix V," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2013, pp. 147–156.

[31] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, 1948.

[32] C. Chiasson and V. Betz, "Should FPGAs abandon the pass-gate?" in *Proc. 23rd Int. Conf. Field Program. Logic Appl.*, 2013, pp. 1–8.

[33] Z. Jiang, S. Yu, Y. Wu, J. H. Engel, X. Guan, and H.-S. P. Wong, "Verilog—A compact model for oxide-based resistive random access memory," in *Proc. IEEE Int. Conf. Simul. Semicond. Processes Devices*, 2014, pp. 41–44.

[34] *HSPICE User Guide: Simulation and Analysis, Version I-2013.12*, Synopsys, Mountain View, CA, USA, Dec. 2013.

[35] Altera Corporation, "Stratix IV device handbook version SIV5V1–1.1," Jul. 2008. [Online]. Available: http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf

[36] S. Yang, "Logic synthesis and optimization benchmarks user guide version 3.0," MCNC, Durham, NC, USA, Tech. Rep., Jan. 1991.

[37] J. Rose, *et al.*, "The VTR project: Architecture and CAD for FPGAs from verilog to routing," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2012, pp. 77–86.

[38] University of California in Berkeley, "ABC: A System for Squential Synthesis and Verification," (2007, Sept.). [Online]. Available: http://www.eecs.berkeley.edu/alanmi/abc/

[39] O. Turkyilmaz, *et al.*, "RRAM-based FPGA for "normally off, instantly on" applications," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, 2012, pp. 101–108.

XIFAN TANG (S'13) received the BSc degree in microelectronics from Fudan University, Shanghai, China, in 2011, and the MSc degree in electrical engineering from the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 2013. He is currently working toward the PhD degree at EPFL. His current research interests include computer-aided design for programmable architecture and emerging technologies. He received the 2015 Chinese Government Award for Outstanding Self-Financed Students Abroad. He is a student member of the IEEE.

GIOVANNI DE MICHELI is a professor and director of the Institute of Electrical Engineering and of the Integrated Systems Centre, EPF Lausanne, Switzerland. He is a program leader of the Nano-Tera.ch program. Previously, he was a professor of electrical engineering with Stanford University. His research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis for emerging technologies, networks on chips, and 3D integration. He is also interested in heterogeneous platform design including electrical components and biosensors, as well as in data processing of biomedical information. He is author of: *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994, co-author and/or co-editor of eight other books and of more than 700 technical articles. His citation h-index is 88 according to Google Scholar. He is a member of the Scientific Advisory Board of IMEC (Leuven, B), CfAED (Dresden, D), and STMicroelectronics. He received the 2016 IEEE/CS Harry Goode award for seminal contributions to design and design tools of Networks on Chips, the 2016 EDAA Lifetime Achievement Award, the 2012 IEEE/CAS Mac Van Valkenburg award for contributions to theory, practice and experimentation in design methods and tools, and the 2003 IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems. He also received the Golden Jubilee Medal for outstanding contributions to the IEEE CAS Society in 2000, the D. Pederson Award for the best paper on the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* in 1987, and several Best Paper Awards, including DAC (1983 and 1993), DATE (2005), and Nanoarch (2010 and 2012). He has been serving IEEE in several capacities, namely: Division 1 Director (2008-9), co-founder and president elect of the IEEE Council on EDA (2005-7), president of the IEEE CAS Society (2003), editor in chief of the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (1997-2001). He has been a chair of several conferences, including Memocode (2014) DATE (2010), pHealth (2006), VLSI SOC (2006), DAC (2000), and ICCD (1989). He is a fellow of the ACM and the IEEE and a member of the Academia Europaea.

PIERRE-EMMANUEL GAILLARDON (S'10-M'11) received the electrical engineer degree from CPE-Lyon, France, in 2008, the MSc degree in electrical engineering from INSA Lyon, France, in 2008, and the PhD degree in electrical engineering from CEA-LETI, Grenoble, France, and the University of Lyon, France, in 2011. He is an assistant professor in the Electrical and Computer Engineering (ECE) Department, The University of Utah, Salt Lake City, Utah, and he leads the Laboratory for NanoIntegrated Systems (LNIS). Prior to joining the University of Utah, he was a research associate with the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, within the Laboratory of Integrated Systems (Prof. De Micheli) and a visiting research associate with Stanford University, Palo Alto, California. Previously, he was a research assistant with CEA-LETI, Grenoble, France. He received the C-Innov 2011 best thesis award and the Nanoarch 2012 best paper award. He is an associate editor of the *IEEE Transactions on Nanotechnology*. He has been serving as TPC member of many conferences, including DATE'15-16, DAC'16, Nanoarch'12-16, and is reviewer of several journals and funding agencies. He will serve as Topic co-chair "Emerging Technologies for Future Memories" for DATE'17. His research activities and interests are currently focused on the development of reconfigurable logic architectures and digital circuits exploiting emerging device technologies and novel EDA techniques. He is a member of the IEEE.