A High-Performance Low-Power Near-*Vt* RRAM-based FPGA

Xifan Tang*, Pierre-Emmanuel Gaillardon and Giovanni De Micheli
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
*Email: xifan.tang@epfl.ch

***Abstract*—The routing architecture, heavily using programmable switches, dominates the area, delay and power of *Field Programmable Gate Arrays* (FPGAs). *Resistive Random Access Memories* (RRAMs) enable high-performance routing architectures through the replacement of *Static Random Access Memory* (SRAM)-based programming switches. Exploiting the very low *on*-resistance state achievable by RRAMs, RRAM-based routing multiplexers can be used to significantly reduce the FPGA routing delays. In addition, RRAM-based routing architectures are less sensitive to supply voltage reductions and show promises in low-power FPGA designs. In this paper, we propose a near-*Vt* low-power RRAM-based FPGA where both delay and power reductions are achieved. Experimental results demonstrate that a near-*Vt* RRAM-based FPGA design leads to a 15% area shrink, a 10% delay reduction, and a 65% power improvement, compared to a conventional FPGA design for a given technology node. To achieve low *on*-resistance values, RRAMs typically require high programming currents. In other word, they need relatively large programming transistors, potentially resulting in area, delay and power inefficiencies. We also present a design methodology to properly size the programming transistors of RRAMs in order to further improve the area-efficiency. Experimental results show that a correct programming transistor sizing strategy contributes to further 18% area and 2% delay shrink, compared to the initial near-*Vt* RRAM-based FPGA.**

I. Introduction

Field Programmable Gate Arrays (FPGAs) are more flexibile than *Application-Specific Integrated Circuits* (ASICs) at the cost of 20× bigger area, 4× longer delay, and 12× higher power consumption approximately [1]. The drawbacks of FPGAs lie in the expensive routing architecture, which accounts for about 70% of the area, 80% of the delay and 60% of the power of the whole chip [2]. Power consumption is a serious barrier for the distribution of FPGAs in a large set of consumer applications. Previous works [3]–[5] demonstrate low-power FPGA designs where a low supply voltage is employed to save up to 50% of the power consumption. However, low-power FPGAs generally suffers from large delay degradation (up to 2×).

Resistive Random Access Memories (RRAMs) [6], a member of *Non-Volatile Memory* (NVM) family [7], open opportunities in advancing the FPGA technology with high density, instant power-on and excellent energy efficiency. Overwhelming *Static Random Access Memories* (SRAMs) intrinsically, RRAMs hold storage when powered down and consume less leakage power. Besides, RRAMs can be fabricated between the *Back-End-Of-Line* (BEOL) metal lines, moving the configuration memories onto the top of the transistors, thereby improving the integration density. Using RRAMs as standalone memories, FPGAs can benefit a ∼50% power reduction from instant power-on and normal power-off, compared to SRAM-based counterparts [8]. Furthermore, RRAMs motivate the exploration of novel FPGA architectures whose routing structures are directly employing RRAMs in the data path. In the novel architectures, RRAMs play the role of both configurable memories and programmable switches. Previous works [9]–[13] demonstrate significant improvements in area, delay and power. The BEOL integration leads to area-savings and the *Low-Resistance State* (LRS) of RRAMs (down to 75% lower *on*-resistance than pass transistors) reduces the delay of critical path. Finally, a power efficiency comes from zero leakage power in sleep mode.

In this paper, we study (i) the opportunity of fabricating low-power RRAM-based FPGA. The performances of RRAM-based routing architecture are less sensitive to *Vdd* reduction as compared to pass transistors. Hence, RRAM-based high-performance routing structures are appealing to compensate the traditional delay degradation found in low-power FPGAs, while maintaining a high power efficiency. Therefore, for the first time, we propose a near-*Vt* RRAM-based FPGA design, combining both power-efficiency and performance. Architectural-level simulations show that near-*Vt* RRAM-based FPGA gives a 15% area gain, a 10% delay gain and a 65% power gain, compared to the baseline FPGA architecture. To achieve low *on*-resistance values, RRAMs typically require high programming currents. To drive such high currents, large programming transistors are needed, and they potentially result in area, delay and power inefficiencies. Hence, we investigate (ii) the impact of the size of programming transistors in RRAM-based multiplexers in terms of *Energy-Delay Product* (EDP). Electrical simulations reveal that at near-*Vt* supply voltage, RRAM-based multiplexers with non-uniform programming transistor sizing produce better EDP than those with uniform sizing. Architectural-level simulations show that non-uniform programming transistor sizing further contributes to 18% area gain and 2% delay gain compared to the initial near-*Vt* RRAM-based FPGA.

The rest of paper is organized as follows. Section II introduces the background knowledge of conventional FPGAs and RRAM-based FPGAs. Section III describes the near-*Vt* RRAM-based FPGA architecture. Section IV introduces programming transistor sizing. Section V presents experimental methodology and results. Section VI draws conclusions.

II. Background

In this section, we review the necessary background of conventional FPGA architectures as well as RRAM-based FPGA architectures.

A. Conventional FPGA Architecture

Fig. 1 depicts the conventional FPGA architecture with single-driver routing [16], where *Configurable Logic Blocks* (CLBs) are surrounded by routing resources, such as *Switch Boxes* (SBs) and *Connection Blocks* (CBs). A CLB contains logic resources, called *Basic Logic Elements* (BLEs), as well as routing resources, denoted as local routing. A BLE consists of a *Look-Up Table* (LUT), a *D Flip-Flop* (DFF) and a 2-input multiplexer, which selects either the combinational or sequential version of the LUT output. SBs and CBs consist of groups of multiplexers, that can realize any interconnection as long as there are enough routing tracks. FPGA performance is influenced by the number of LUT inputs, denoted K, the number of BLEs in a CLB, denoted N, and the number of inputs of a CLB, denoted I. Previous works [14] [15] conclude that $I = \frac{K(N+1)}{2}$ ensures over 98% utilization of CLBs. Commercial FPGAs [17]–[19] widely support fracturable LUTs [20] to reduce the critical path. In this paper, we typically consider FPGA consisting of $K = 6$ fracturable LUTs organized in logic blocks described by $N = 10, I = 33$.

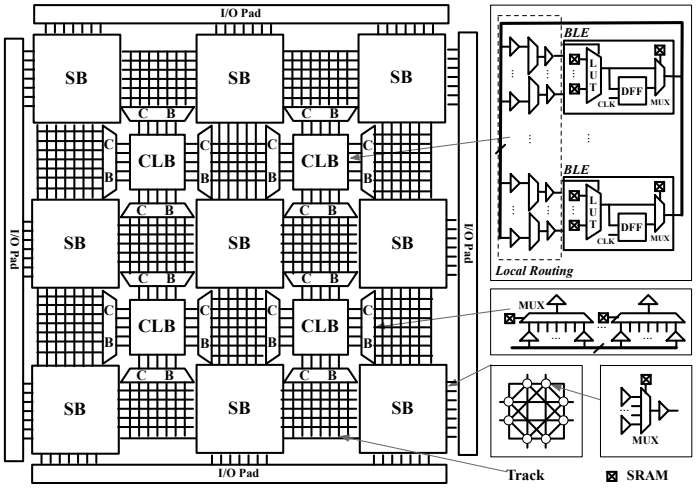

Fig. 1. Conventional FPGA architecture.

B. RRAM Technology

As one of the most promising emerging NVM memories [7], RRAM technologies have been widely investigated [6]. As shown in Fig. 2(a), RRAMs are two-node electronic devices and typically consist of three layers: the top electrode, the metal oxide and the bottom electrode. RRAMs can be programmed into two stable resistance state, a *Low Resistance State* (LRS) and a *High Resistance State* (HRS) respectively by modifying the conductivity of metal oxide. When a programming voltage is applied between the electrodes, the metal oxide sees a conductivity change which leads to the switch of the resistance states. Switching mechanism can be categorized into *Unipolar Resistive Switching* (URS) and *Bipolar Resistive Switching* (BRS) [6]. In this paper, we focus on BRS whose I-V characteristics are illustrated in Fig. 2(c). A positive programming voltage **sets** the RRAM in LRS while a negative one **resets** the RRAM in HRS. The *on*-resistance of RRAM is typically dependent on the programming current passing through the RRAM [21]. The higher programming current we drive, the lower *on*-resistance RRAM we obtain. Note that during the *SET* process, a current compliance is often enforced to avoid permanent breakdown of the device. Fig. 2(b) shows a 1T1R structure, where the programming transistor provides *SET*/*RESET* voltages as well as a current compliance. *Back-End-Of-Line* (BEOL) technology allows RRAM to be fabricated on the top of or between metal layers, saving chip area. Fig. 2(a) illustrates the BEOL integration of RRAMs corresponding to 1T1R programming scheme in Fig. 2(b). For more details about RRAM technology, we refer the interested reader to [6].

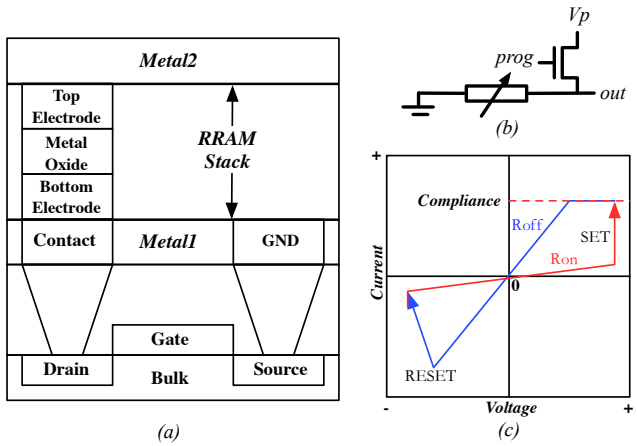

Fig. 2. BEOL integration of RRAM between metal layers (a); implementing a 1T1R structure (b); I-V characteristics of a bipolar RRAM (c).

C. RRAM-based FPGA Architecture

FPGA architecture can benefit from the *non-volatility* as well as the area and performance gains coming from the BEOL integration and the low *on*-resistance values achieved by RRAMs. To improve the LUTs, SRAMs can be simply replaced with voltage divider-like RRAM structures [12]. However, more opportunities lie in the routing architecture where not only SRAMs but also pass-transistors can be improved with RRAMs [9]–[13], thereby increasing the performances significantly. When programmed in LRS, RRAMs introduce about 75% less resistance in the data-path, compared to pass transistors. Works in [9] [10] propose novel routing architecture exploiting RRAM-based programmable switches while [11]–[13] explore the architectural-level potential of RRAM-based multiplexers. To reduce the impact of the programming switches, programming transistor sharing is heavily studied in [9] [10] for area-saving purpose but requires complicated programming operation. In [12], the programming complexity is reduced by exploiting the physical properties of RRAMs. However, previous works [9]–[13] only investigate operations under standard working voltages, leaving near-*Vt* RRAM-based FPGAs an open question.

III. Near-*Vt* RRAM-based FPGA

In this section, we describe our RRAM-based FPGA circuit design and explore its use in near-*Vt* regime.

1) RRAM-based FPGA: The RRAM-based FPGA introduced in this paper has no architectural difference with respect to the conventional SRAM-based FPGA shown in Fig. 1. It

remains an island-style FPGA where the cluster-based CLBs are surrounded by SBs and CBs. The differences lie in the circuit design of those modules heavily relying on LUTs and multiplexers. Fig. 3 compares the circuit designs of LUT and multiplexer between a conventional SRAM-based FPGA and the RRAM-based FPGA introduced in this paper.

In our FPGA, the logic elements exploit *Non-Volatile* (NV) LUTs. Such FPGA does not need to be re-programmed during each power on and can benefit instant-*on* and normally-*off* properties. Typically, a LUT consists of a bank of SRAMs and a multiplexer. The SRAM bank stores a truth table which is decoded by the multiplexer, enabling LUT to realize any logic function. In this paper, we replace the scan-chain SRAMs (Fig. 3(a)) in LUTs with *Non-Volatile* (NV) scan-chain SRAMs borrowed from previous work [22] [23]. The multiplexers in LUTs are still implemented by pass-transistors considering that their decoding results keep changing when the FPGA is operating. If RRAMs are inserted in the data path of LUTs for decoding, their programming speed will drastically limit frequency. Compared to SRAM-based, the NV LUTs have no difference in performance because of the same decoder implementation. Data path DFFs are also *Non-Volatiled* with the same circuit elements. These FFs operate as standard volatile CMOS FF during regular operation but they are also capable to store the data *non-volatily* on demand before a sleep period. Data stored in the NV DFFs can then be restored during wake up. In these flip-flops, RRAMs are written only before the sleep period. These events have very low frequency and are compatible with the endurance capabilities of RRAMs. While supported by the presented architecture, instant-*on* and normally-*off* operation will not be evaluated in this paper. More details about the NV DFF architecture can be found in [23].

While the decoded paths of the LUT multiplexer change at runtime, the selected paths in the routing multiplexers (i.e., in BLE output selector, local routing, SBs and CBs) remain unchanged during the runtime. Therefore, RRAMs can be inserted in the data path of routing architecture without challenging the endurance. Fig. 3(d) illustrates the RRAM-based multiplexer [12] which replaces the SRAM-based multiplexer shown in Fig. 3(c). RRAM-based multiplexers take advantage of the *Bipolar Resistive Switching* (BRS) in order to share programming transistors and achieve area-efficiency [12]. As shown in Fig. 3(d), each pair of RRAMs (e.g., $R1$ and $R2$) can be programmed in either $HRS+LRS$ or $LRS+HRS$ in one step. Compared to the SRAM-based multiplexers, the RRAM-based multiplexers exhibit high performances accounted to the low *on*-resistance of the RRAMs introduced in the data path. However, a low *on*-resistance of the RRAMs means high programming currents. In other words, they need large programming transistors which potentially introduce large parasitic capacitance to the data paths and result in area and delay in-efficiencies.

2) Impact of Vdd Reduction on RRAM-based Routing Architecture: In conventional SRAM-based low-power FPGAs, a reduction of the supply voltage down to near/sub-*Vt* regime trades off power reduction with delay degradation. In RRAM-based FPGAs, logic elements such as LUTs and DFFs rely on the same circuit topologies. Therefore, their performances degrade when supply voltage reduces to near/sub-*Vt* regime.

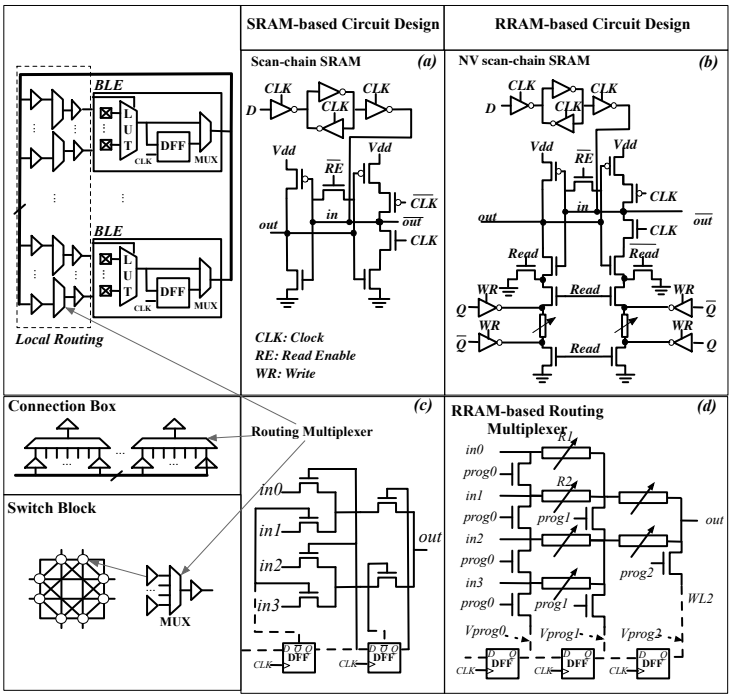

Fig. 3. SRAM-based FPGA and RRAM-based FPGA.

However, routing architectures in the RRAM-based FPGA exploit RRAMs in the data paths and may perform differently compared to SRAM-based when supply voltage changes. Hence, in this part, we study the impact of supply voltage on the performances of RRAM-based routing architecture.

Electrical simulations are performed in a commercial $0.18\mu m$ technology. Nowadays, low-power near/sub-*Vt* designs are implemented with mature technology node for better leakage characteristics and reliability. Nevertheless, the approach introduced in this paper is general and can lead to the same conclusion under advanced technology nodes. We consider $R_{on} = 1k\Omega$ and $R_{off} = 1M\Omega$ for the RRAM device parameters, as per [24] [25].

The FPGA routing architecture consists of multiplexers of different sizes, which appear in local routing, CBs and SBs. Here, we consider a local 32-input multiplexer. For the multiplexers of other sizes, the same conclusions can be reached. Fig. 4 compares the delay and power between a 32-input SRAM-based multiplexer and its RRAM-based counterpart when Vdd ranges from $0.4V$ to $1.8V$. Both RRAM-based and SRAM-based multiplexers reduce power but suffer from delay degradation when Vdd decreases. Generally, RRAM-based multiplexer consumes slightly more power than SRAM-based due to the low *on*-resistance of RRAMs in data paths. However, SRAM-based FPGA routing architecture suffers serious delay degradation when Vdd decreases. In contrast, RRAM-based FPGA routing architecture benefit the same power reduction but with very moderate delay degradation. The different trends in delay degradations are accounted to the low *on*-resistance of RRAMs which is achieved independently from Vdd, while *on*-resistance of pass transistors increase sharply when Vdd decreases. Furthermore, the parasitic capacitances brought by the programming transistors do not vary significantly until Vdd drops to sub-*Vt* regime. Therefore, the delay of RRAM-based multiplexer in near-*Vt* regime remains as they are at $Vdd =$

$1.8V$ since its RC characteristic does not change. When Vdd drops to sub-*Vt* regime, RRAM-based multiplexer has serious delay degradation as well due to parasitic capacitances of programming transistors increase. Fig. 4 shows us to select a proper *Vdd* in the near-*Vt* regime. Hence, the RRAM-based FPGA will achieve both low-power and high-performance. The high-performance RRAM-based routing architectures are expected to compensate the delay degradation in the logic elements, and even reduce the overall critical path delay.

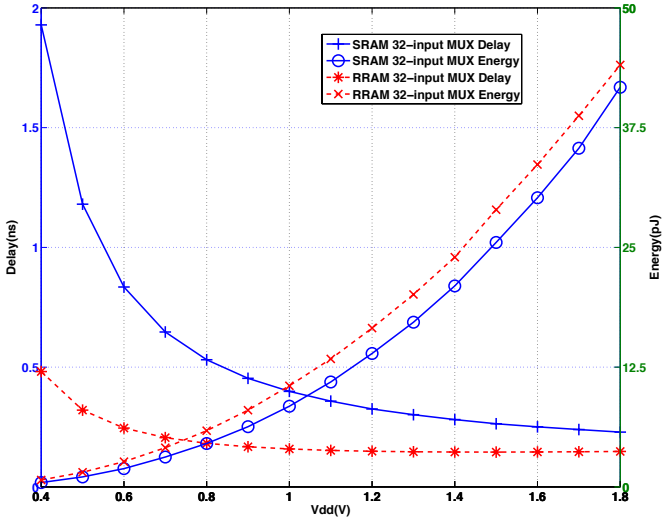

Fig. 4. Energy and delay evaluation of 32-input SRAM-based and RRAM-based multiplexers under *Vdd* reductions.

3) Configuring RRAMs in FPGAs: In SRAM-based FPGAs, SRAMs bits are configured by scan-chain SRAMs, as shown in Fig. 3(a). All scan-chain SRAMs are connected in series (i.e., dash lined in Fig. 3(c)) and the program bitstream is serially loaded to the scan-chain SRAMs until all SRAM bits are configured. In RRAM-based FPGAs, scan-chain SRAMs in logic elements are adapted to the NV scan-chain SRAMs as shown in Fig. 3(b). Each stage of the RRAM-based multiplexer (Fig. 3(d)) is configured sequentially. Full details about the programming strategy are available in [13]. When the program bit is loaded in the scan-chain SRAMs for a certain stage, the corresponding programming transistors are turned *on*. After programming, these programming transistors are turned *off*. In the RRAM-based FPGA, RRAMs in the data paths should not be mistakenly programmed when transmitting signals. This critical concern is avoided by ensuring that the programming voltage V_{prog} for RRAMs is larger than supply voltage Vdd, as shown in equation (1):

$$V_{prog} = \lambda \cdot Vdd (\lambda > 1) \tag{1}$$

In this paper, we set λ to 1.2, to provide enough margin between Vdd and V_{prog} and limit the risk of parasitic programming. The V_{prog} parameter can be easily adjusted by tuning the RRAM stack geometries [6]. Note that Vdd is expected to be near-*Vt*. Therefore V_{prog} will stay in a regular range, i.e., super threshold, of the MOS transistors, that can be used as is.

IV. Programming Transistor Sizing

As compared to standard FPGAs, RRAM-based FPGAs have the unique property of merging memory with the data-paths. In this section, we study the impact of programming transistor size on the performance of RRAM-based routing architecture, estimate their optimal size and verify it by electrical simulations.

A. Impact of Programming Transistor Size

In previous works [9]–[13], the sizes of programming transistors are considered uniform to achieve the lowest *on*-resistance of RRAM, which is assumed to produce the best performance of RRAM-based interconnects. Actually, the delay of RRAM-based programmable interconnects is determined by various factors, such as the size of the driving inverter, the parasitic capacitance of programming transistors, and the resistance of the RRAMs. Hence, as the *on*-resistance value is strongly correlated with the size of the programming transistors [6], there is no guarantee that using the lowest possible *on*-resistance will give the lowest delay. In this section, we focus on the impact of programming transistor size on the delay of RRAM-based multiplexers. Note that the methodology developed here is not dependent on the considered RRAM technology or on the transistor technology nodes, but is rather general.

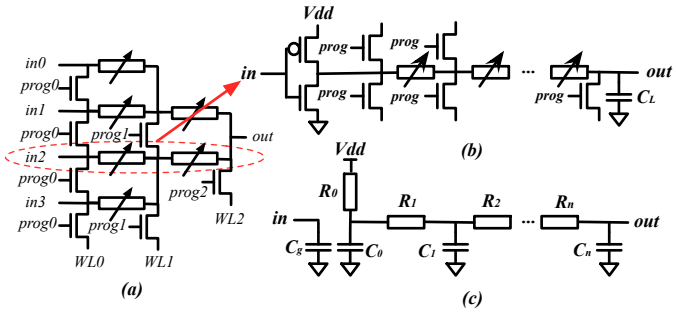

Fig. 5. Critical path of 4-input RRAM-based multiplexer (a); General critical path of RRAM-based multiplexer (b); Equivalent RC model (c).

The critical path of a RRAM-based multiplexer is the path from an input to the output which contains the largest number of RRAMs in the *on*-resistance state and the largest number of programming transistors. For instance, the highlighted path in Fig. 5(a) is the critical path of a 4-input RRAM-based multiplexer. Fig. 5(b) extends this to the general case of a *n*-stage RRAM-based multiplexer, while its equivalent *RC* model is given in Fig. 5(c).

The resistance and capacitance in Fig. 5(c) can be extracted from Fig. 5(b) and expressed as follows:

$$\begin{aligned} &R_0 = R_{inv} = \frac{R_{min}}{W_{inv}}, \\ &R_i|_{1 \le i \le n} = R_{on} \\ &C_0 = W_{inv} C_{inv} + 2 W_{prog} C_{off} \\ &C_i|_{1 \le i < n} = 2 W_{prog} C_{off} \\ &C_n = C_L + W_{prog} C_{off} \end{aligned} \tag{2}$$

where R_{min} denotes the equivalent resistance of a minimum size inverter, C_{inv} represents the parasitic capacitance at the output of a minimum size inverter, W_{inv} is the size of driving inverter in terms of the minimum width transistor [14]. R_{on} denotes the equivalent resistance of a RRAM in *on*-resistance

state. W_{prog} represents the width of programming transistor in the unit of the minimum width transistor, and C_{off} is the parasitic capacitance of a minimum width programming transistor in *off* state.

Considering the Elmore delay [26] of the critical path of a general *n*-stage RRAM-based multiplexer (Fig. 5(b)), we obtain:

$$\begin{aligned}\tau &= \sum_{i=0}^{n} R_i \sum_{j=i}^{n} C_j \\ &= R_{min}C_{inv} + \frac{R_{min}}{W_{inv}}C_L \\ &+ (2n+1)\frac{R_{min}}{W_{inv}}W_{prog}C_{off} + n \cdot R_{on}C_L \\ &+ n^2 R_{on}W_{prog}C_{off}\end{aligned} \tag{3}$$

As introduced previously, the *on*-resistance R_{on} of RRAM is dependent on the programming voltage V_{prog} and on the programming current I_{prog} [6], as follows:

$$R_{on} = \frac{V_{prog}}{I_{prog}} = \frac{V_{prog}}{W_{prog} \cdot I_d} \tag{4}$$

where I_d is the driving current of a minimum width transistor. With equation (4), equation (3) is converted to:

$$\begin{aligned}\tau &= R_{min}C_{inv} + \frac{R_{min}}{W_{inv}}C_L \\ &+ (2n+1)\frac{R_{min}}{W_{inv}}W_{prog}C_{off} + n \cdot \frac{V_{prog}}{I_d W_{prog}}C_L \\ &+ n^2 \frac{V_{prog}}{I_d}C_{off}\end{aligned} \tag{5}$$

The relation between the *n*-stage multiplexer delay and the width of the programming transistor is depicted in Fig. 6.

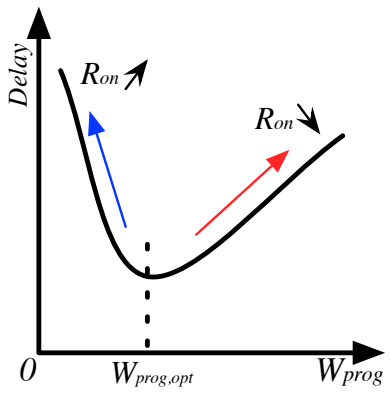

Fig. 6. Relation between W_{prog} and delay of a RRAM-based multiplexer.

When W_{prog} is small, the delay increases due to the large *on*-resistance of RRAM. When W_{prog} is large, the delay increases as well. Indeed, while the *on*-resistance is reduced, large parasitic capacitances are introduced by the programming transistors and limit the performances. Therefore, as shown in Fig. 6, there exists an optimal $W_{prog,opt}$ giving the best performances by trading off the *on*-resistance with the parasitic capacitances from the programming transistors. Equation (5) reaches minimum value (best delay) when:

$$W_{prog,opt} = \sqrt{\frac{nV_{prog}C_L W_{inv}}{(2n+1)I_d R_{min} C_{off}}} \tag{6}$$

In FPGA routing architecture, the number of the stages of multiplexers are diverse. As Equation 6 depends on the size *n* of the multiplexer, using a uniform size of programming transistors [9], [10], [12], [13] does not ensure the best performance. To achieve the best performances, the multiplexers in FPGA should have different $W_{prog,opt}$.

B. Electrical Simulations

In this section, we show some electrical simulations to verify the analysis developed above.

1) Methodology: Equation 6 reveals that $W_{prog,opt}$ is related to many process parameters: V_{prog}, I_d, R_{min} and C_{off}, and some design-dependent parameters, W_{inv}, C_L and n. Process parameters, I_d, R_{min} and C_{off}, are extracted from a commercial $0.18\mu m$ technology. As for design parameters, we refer to [14] [16] and study multiplexers for SB, CB, BLE and local routing assuming a baseline FPGA architecture. Table I

TABLE I. RRAM-based multiplexers in baseline FPGA architecture.

Location	No. of input	Drive *inv.* size	Load *inv.* size
Switch Block	4	1	10
Connection Box	32	1	1
Local routing	53	2	1
BLE	2	1	1

presents the setup for the different RRAM-based multiplexers considered in the FPGA architecture. W_{inv} of multiplexers in SBs, CBs and BLEs are set as 1. W_{inv} of multiplexers in local routing is set as 2 to drive the signal from routing tracks. The load of multiplexers in BLE, CB and local routing are set as a inverter $\times 1$. In SBs, load of multiplexer is set as a inverter $\times 10$ in order to drive the large parasitic capacitance of a routing track.

2) Experimental Results: With all the defined parameters above, we sweep Vdd from 0.4V to 1.8V and W_{prog} from 1 to 3 to explore their impact on delay and EDP. The lower bound is set to 1 for the minimum width transistor. The upper bound of W_{prog} is set to 3, which is the size of a pair of complementary pass transistors, to limit the area overhead.

Equation 6 predicts that when V_{prog} decreases, $W_{prog,opt}$ decreases. Experimental results in Fig. 7 verify this prediction. Fig. 7 depicts the delay of a 32-input multiplexer extracted while sweeping Vdd and W_{prog}. The curves, obtained for $Vdd = 1.8V$ and $Vdd = 1.4V$, are similar to the region highlighted in red in Fig. 6. In these two cases, the best performance is achieved when $W_{prog} = 3$ and $W_{prog} = 2$, respectively. The curve obtained for $Vdd = 0.8V$ corresponds to the blue region highlighted in Fig. 6. In this case, the best performance is achieved when $W_{prog} = 1$. When comparing the three curves, we observe that the best performance

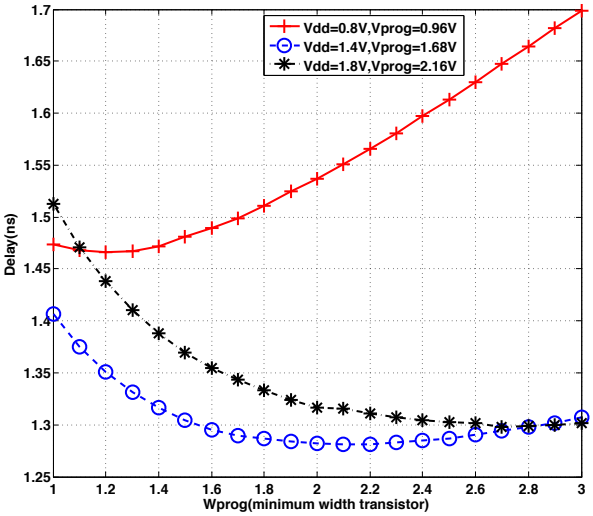

Fig. 7. Delay variations for increasing W_{prog} of 32-input multiplexer.

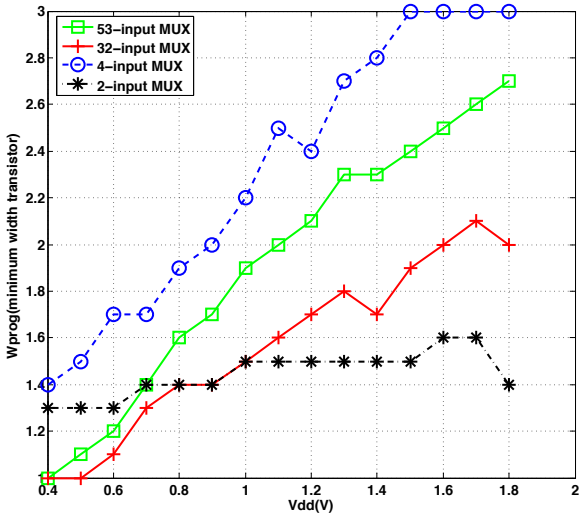

Fig. 8. Optimal W_{prog} (Best EDP) as a function of Vdd for RRAM-based routing multiplexers.

shifts from $W_{prog} = 3$ when $Vdd = 1.8V$ to $W_{prog} = 1$ when $Vdd = 0.8V$ for a 32-input RRAM-based multiplexer.

For low-power FPGAs, designers often consider the best *Energy-Delay Product* (EDP) as a good trade-off metrics. Fig. 8 presents the optimal W_{prog}, i.e., leading to the best EDP, of the different multiplexers listed in Table I by sweeping Vdd ranging from 0.4V to 1.8V. Equation 6 predicts that a large capacitive load leads to a large $W_{prog,opt}$. The curve of the 4-input multiplexers in SBs, whose loads are inverters $10\times$ verifies this prediction, where $W_{prog,opt}$ is significantly larger than the other multiplexers. Equation 6 also predicts that $W_{prog,opt}$ increases when the number of multiplexer stage increases. By comparing the curves of 32-input, 4-input and 2-input multiplexers, we remark that the $W_{prog,opt}$ of a 32-input multiplexer is the largest while a 2-input multiplexer requires the smallest $W_{prog,opt}$. Note that we determine $W_{prog,opt}$ in terms of the best EDP rather than delay. Hence, the results in the sub-*Vt* regime do not strictly perform as predicted by Equation 6 because the energy values dominate the EDP in the sub-*Vt* regime. The experimental results show that non-uniform sizes of programming transistors produce best delay and EDP. Optimal sizes of programming transistors in the multiplexers differ from their design contexts in FPGAs. For instance, the multiplexers in SBs require large programming transistors while the multiplexers in BLEs and local routing require small programming transistors. These experimental results are particularly appealing in the context of sub/near-*Vt* FPGAs, where the sizes of the programming transistors can be reduced, contributing to not only area-saving but also to further delay and power efficiencies. Take the example of the 4-input multiplexers in SBs. When $Vdd = 1.2V$ is applied, compared to uniform size ($W_{prog} = 3$), $W_{prog,opt} = 2.4$ leads to a 20% area reduction, yet ensuring the best EDP.

V. Architectural-Level Simulations

In this section, architectural-level simulations are carried out to evaluate near-*Vt* RRAM-based FPGAs. First, we introduce the experimental methodology and, then, we present the experimental results.

A. Methodology

We compare the area, delay and power of four different FPGAs: (1) the standard CMOS FPGA architecture when $Vdd = 1.8V$, (2) the standard CMOS FPGA architecture when $Vdd = 1.2V$, (3) the RRAM-based FPGA architecture using uniform programming transistor sizing at $Vdd = 1.2V$ and (4) the RRAM-based FPGA architecture using non-uniform optimized programming transistors sizing at $Vdd = 1.2V$. Comparison with previous RRAM-based FPGAs [9]–[13] is out of the scope of this paper, as none of these solutions are operated at low power supply. At near-*Vt* regime, we select $1.2V$ as *Vdd*, because it provides a reasonable trade-off between performance gain in RRAM-based routing architecture and performance degradation in logic elements. Architecture-level results are generated by VTR flow [28]. The twenty largest MCNC benchmarks [27] pass through logic synthesis by ABC [29]. VPR 7 [28] conducts the physical synthesis including packing, placement and routing. We use the *Configurable Logic Block* (CLB) architecture described in Section II and single-driver routing architecture. For the *Connection Blocks* (CBs), we set $F_{c,in} = 0.15$ and $F_{c,out} = 0.10$. For the *Switch Boxes* (SBs), we use a *Wilton* pattern and set $F_s = 3$. Technology parameters (area, delay and power) are extracted from commercial $0.18\mu m$ technology.

B. Experimental Results

Fig. 9, Fig. 10 and Fig. 11 show the experimental results for area, delay and power, respectively. Fig. 9 illustrates the area comparison between the four FPGA architectures. Compared to the standard FPGA architectures, the uniform programming transistor sized RRAM-based FPGA working at $Vdd = 1.2V$ saves 15% area on average thanks to the BEOL technology which moves memories to the top of the chip. Compared to the uniformly sized RRAM-based FPGA, RRAM-based FPGA using the non-uniformly sized programming transistors saves further 18% area on average thanks to the reduced impact of the programming transistors in the routing structure. Fig. 10 illustrates the delay comparison between the four FPGA architectures. When Vdd drops from $1.8V$ to $1.2V$,

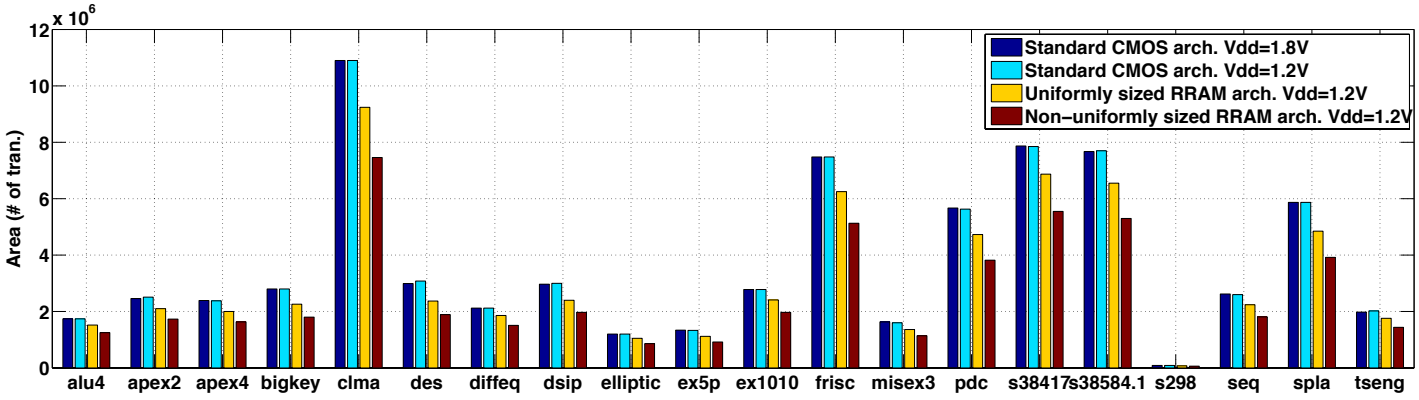

Fig. 9. Area comparison for 20 biggest MCNC benchmarks implemented in standard CMOS architecture at $Vdd = 1.8V$, standard CMOS architecture at $Vdd = 1.2V$, uniformly sized RRAM architecture at $Vdd = 1.2V$, and non-uniformly sized RRAM architecture at $Vdd = 1.2V$.

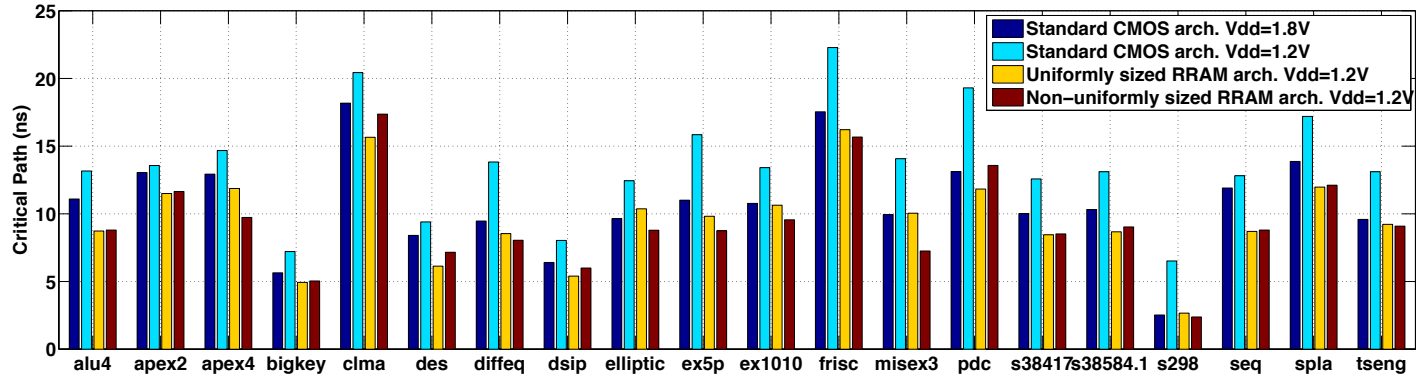

Fig. 10. Critical path comparison for 20 biggest MCNC benchmarks implemented in standard CMOS architecture at $Vdd = 1.8V$, standard CMOS architecture at $Vdd = 1.2V$, uniformly sized RRAM architecture at $Vdd = 1.2V$, and non-uniformly sized RRAM architecture at $Vdd = 1.2V$.

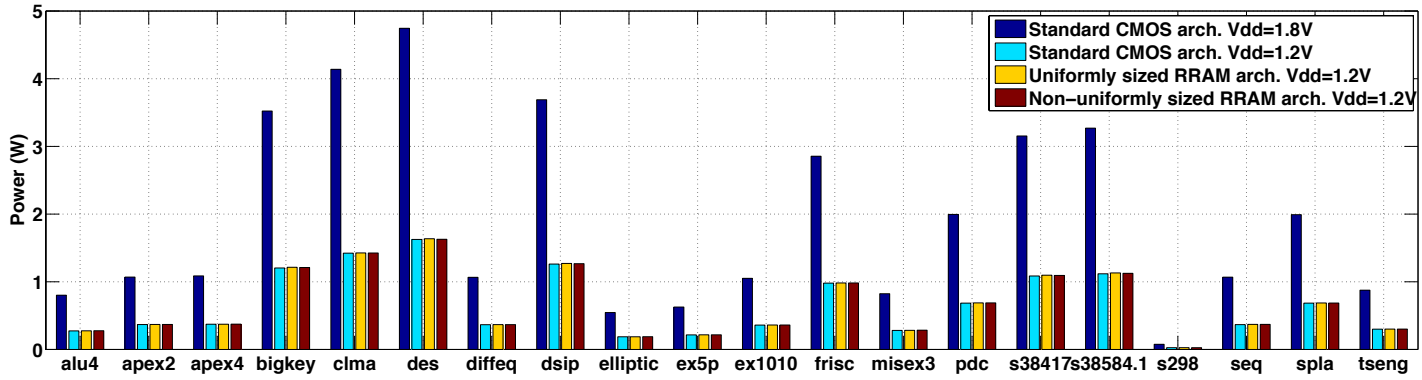

Fig. 11. Power comparison for 20 biggest MCNC benchmarks implemented in standard CMOS architecture at $Vdd = 1.8V$, standard CMOS architecture at $Vdd = 1.2V$, uniformly sized RRAM architecture at $Vdd = 1.2V$, and non-uniformly sized RRAM architecture at $Vdd = 1.2V$.

the standard FPGA architecture sees a 30% increase in its critical path delay, resulting from the degradation of driving current that transistors can provide. Compared to the standard FPGA architecture at $Vdd = 1.8V$, the RRAM-based FPGA using uniformly sized programming transistors reduces by 10% on average the delay even at $Vdd = 1.2V$. This comes from the high performance of the RRAM-based routing architecture. The RRAM-based routing architectures can still produce high performance at $Vdd = 1.2V$ and even compensate the delay degradation in logic elements, leading to overall performance gain. Such a result is extremely interesting as it shows that a near-*Vt* RRAM-based FPGA is able to overperform a regular CMOS architecture working at nominal voltage. Compared to the uniformly sized RRAM-based FPGA, the non-uniformly sized RRAM-based FPGA can further improve 2% delay on average. The delay gain comes from the programming transistor sizing methodology that controls the impact of the parasitic capacitances introduced by the programming transistors and lead to the best EDP figures. Fig. 11 illustrates the power comparison between the four FPGA architectures. Both the standard and RRAM-based near-*Vt* FPGA architectures reduce on average by 65% the power consumption. This is accounted directly to the reduction of Vdd. At the same Vdd, RRAM-based and standard FPGAs have almost no difference in power consumption because of the similar switching capacitances in the data paths. In the logic elements, RRAM-based and standard FPGAs have similar switching capacitances because they share similar circuit topologies. In the RRAM routing architectures, the switch capacitances come from the programming transistors, while in the standard routing architecture, they come from the pass transistors. The number of programming transistors in a RRAM-based multiplexer roughly equals to the number of pass transistors in a standard one. Therefore, the switch capacitances in routing architectures are similar.

VI. Conclusion

This paper introduces a near-*Vt* RRAM-based FPGA, where low-power can be achieved along with area reduction and performance improvement thanks to the high performance of RRAM-based routing architecture. Experimental results show that it improves area by 15%, delay by 10% and power by 65% as compared to the standard architecture working at nominal voltage. To push forward the area efficiency of RRAM-based routing architecture, we also propose a design methodology to size the programming transistors of the RRAMs. Both theoretical analysis and electrical simulations show that non-uniform sizing gives not only area savings but also better performance and EDP than using uniformly sized programming transistors. Architectural-level simulations demonstrate optimal sized programming transistors further optimizes the near-*Vt* RRAM-based FPGA by 18% in area and 2% in delay.

Acknowledgment

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

References

[1] I. Kuon *et al.*, *Quantifying and Exploring the Gap Between FPGAs and ASICs*, Springer, 2009.

[2] M. Lin *et al.*, *Performanece Benefits of Monolithically Stacked 3-D FPGA*, IEEE TCAD, Vol. 26, No. 2, 2007, pp. 216-229.

[3] L. Cheng *et al.*, *Device and Architecture Cooptimization for FPGA Power Reduction*, IEEE TCAD, Vol. 26, No. 7, pp. 1211-1221.

[4] T. Tuan *et al.*, *A 90-nm Low-Power FPGA for Battery-Powered Applications*, IEEE TCAD, Vol. 25, No. 2, pp. 296-300.

[5] B. H. Calhoun *et al.*, *Flexible Circuits and Architectures for Ultralow Power*, Proceedings of the IEEE, Vol. 98, No. 2, pp. 267-282.

[6] H.-S. P. Wong *et al.*, *Metal-Oxide RRAM*, Proceedings of the IEEE, Vol. 100, No. 6, 2012, pp. 1951-1970.

[7] G.W. Burr *et al.*, *Overview of Candidate Device Technologies for Storage-Class-Memory*, IBM J. R&D, Vol. 52, No. 4/5, July/Sept. 2008.

[8] O. Turkyilmaz *et al.*, *RRAM-based FPGA for "Normally Off, Instantly On" Applications*, NANOARCH, 2012, pp. 101-108.

[9] S. Tanachutiwat *et al.*, *FPGA Based on Integration of CMOS and RRAM*, IEEE TVLSI, Vol. 19, No. 11, 2010, pp. 2023-2032.

[10] J. Cong and B. Xiao, *FPGA-RPI: A Novel FPGA Architecture With RRAM-Based Programmable Interconnects*, IEEE TVLSI, Vol. 22, No. 4, 2014, pp. 864-877.

[11] P.-E. Gaillardon *et al.*, *Emerging Memory Technologies for Reconfigurable Routing in FPGA Architecture*, ICECS, 2010, pp. 62-65.

[12] P.-E. Gaillardon *et al.*, *GMS: Generic Memristive Structure for Non-Volatile FPGAs*, IEEE/IFIP VLSI-SoC, 2012, pp. 94-98.

[13] P.-E. Gaillardon *et al.*, *Design and Architectural Assessment of 3-D Resistive Memory Technologies in FPGAs*, IEEE TNANO, Vol. 12, No. 1, 2013, pp. 40-50.

[14] V. Betz *et al.*, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, 1998.

[15] E. Ahmed *et al.*, *The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density*, IEEE TVLSI, Vol. 12, No. 3, 2004, pp. 288-298.

[16] G. Lemieux *et al.*, *Directional and Single-Driver Wires in FPGA interconnect*, FPT, 2004, pp. 41-48.

[17] D. Lewis *et al.*, *The Stratix II Logic and Routing Architecture*, FPGA, 2005, pp.14-20.

[18] Altera Corporation, *Stratix IV device handbook version SIV5V1-1.1*, July 2008. http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf

[19] Xilinx, *Virtex-5 User Guide UG190 (v4.0)*, March 2008. http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

[20] M. Hutton *et al.*, *Improving FPGA Performance and Area Using an Adaptive Logic Module*, FPL, 2004, pp. 135-144.

[21] W. Kim *et al.*, *Forming-free Nitrogen-doped AlO_x RRAM with Sub-μA Programming Current*, Symposia on VLSI, 2011, pp. 22-23.

[22] K. Huang *et al.*, *A Low Active Leakage and High Reliability Phase Change Memory (PCM) based Non-Volatile FPGA Storage Element*, IEEE TCAS I, Vol. 61, No. 9, 2014, pp. 2605 - 2613.

[23] I. Kazi *et al.*, *Energy/Reliability Trade-Offs in Low-Voltage ReRAM-Based Non-Volatile Flip-Flop Design*, accepted to IEEE TCAS I.

[24] Z. Fang *et al.*, *$HfO_x/TiO_x/HfO_x/TiO_x$ Multilayer-Based Forming-Free RRAM Devices With Excellent Uniformity*, IEEE EDL, Vol. 32, No. 4, 2011, pp. 566 - 568.

[25] B. Gao *et al.*, *Oxide-Based RRAM: Unified Microscopic Principle for both Unipolar and Bipolar Switching*, IEDM, 2011, pp.417-420.

[26] W.C. Elmore, *The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers*, Journal of Applied Physics, Vol. 19, No. 1, 1948, pp. 55-63.

[27] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0*, MCNC, 1991.

[28] J. Rose *et al.*, *The VTR Project: Architecture and CAD for FPGAs from Verilog to Routing*, FPGA, 2012, pp. 77-86.

[29] University of California in Berkeley, *ABC: A System for Sequential Synthesis and Verification*, Available online. http://www.eecs.berkeley.edu/~alanmi/abc/