Abstract—Nowadays, Resistive Random Access Memories (RRAMs) are one of the most promising candidates in Non-Volatile Memory (NVM) family. RRAMs are two-node devices and can be configured into two stable resistance states, either Low Resistance State (LRS) or High Resistance State (HRS). Compared to Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) not only suffer larger critical delays due to their routing architecture, but also burn more leakage power due to the volatility of SRAMs. RRAMs open an opportunity in giving FPGA non-volatility and high performance routing architecture. The routing architecture can benefit lower on-resistance (down to 75%) from introducing RRAMs to data paths, compared to Static Random Access Memory (SRAM)-controlled programmable switches. Previous works [1]–[3] investigate three different methodologies in integrating RRAMs with FPGAs. Yi-Chung Chen et al. simply replace SRAMs with RRAMs and propose a non-volatile 3D FPGA. Jason Cong et al. introduce a transistor-less RRAM-based routing architecture, which has significant delay reduction (55% on average). P.-E. Gaillardon et al. propose general RRAM-based multiplexer circuit design, which improves delay by 58% on average. During my PhD study, I plan to (i) optimize the RRAM-based routing architectures with the aid of physical modeling (RRAM-based routing architecture in [3] can be further optimized to achieve area-efficiency, with the aid of physical modeling), (ii) improve EDA techniques to exploit RRAM-based routing architecture, and (iii) design a RRAM-based FPGA chip and test its performance.

Index Terms—FPGA, RRAM, Circuit Design, Computer Aided Design

Proposal submitted to committee: June 16th, 2014; Candidacy exam date: June 23th, 2014; Candidacy exam committee: Prof. Paolo Ienne, Prof. Giovanni De Micheli, Dr. Pierre-Emannuel Gaillardon, Prof. David Atienza.

This research plan has been approved:

Date: 

(name and signature)

Thesis director: 

(name and signature)

Thesis co-director: 

(if applicable) (name and signature)

Doct. prog. director:  

(B. Falsafi) (signature)

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are endowed with programmability at the cost of 20× bigger area, 4× longer delay, and 12× higher power consumption, compared to Application-Specific Integrated Circuits (ASICs) [4]. 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 [5]. Besides, FPGAs have to burn high leakage power even in sleep mode due to the volatility of Static Random Access Memories (SRAMs). Once FPGAs are powered off, they lost configurations. And each time FPGAs are powered on, they have to be re-programmed.

As a member of Non-Volatile Memories (NVMs), Resistive Random Access Memories (RRAMs) are promising in advancing FPGA technologies with instant power-on, high density and excellent energy efficiency. By replacing SRAMs with RRAMs, FPGAs evolve to be non-volatile. Such FPGAs can hold storage when powered off and can be instant powered on without re-programming. RRAMs can be fabricated within Back-End-Of-Line (BEOL) metal layers, moving the configuration memories onto the top of the transistors, thereby improving the integration density. Yi-Chung Chen et al. proposes a 3D stacked RRAM-based FPGA, where RRAMs substitute for SRAMs. They achieve a 62% area reduction and a 34% delay improvement. Furthermore, RRAMs motivate the exploration of novel FPGA routing architectures by employing RRAMs in the data path. In the novel architectures, RRAMs play the role of both configurable memories and programmable switches. Previous works [2] [3] demonstrate significant improvements in area, delay and power. J. Cong et al. [2] study a transistor-less RRAM-based programmable interconnects, which are built with only RRAMs and metal wires. Significant improvements (a 96% area reduction, a 55% delay reduction, a 79% power reduction) can be achieved. P.-E. Gaillardon et al. focus on the circuit design of RRAM-based multiplexers, reducing the area and delay of FPGA by 7% and 58% respectively.

All the three papers deliver the concepts of RRAM-based FPGAs without seriously considering the parasitic effects from the RRAMs. These parasitic effects of RRAMs, such as parasitic capacitance, should be taken into account when design and optimize a RRAM-based FPGA. During my PhD study, I will first (i) optimize the RRAM-based routing architecture through physical modeling. Physical modeling can estimate how the parasitic effects of RRAMs impacts the performance of RRAM-based architecture, and analysis if there exists any trade-off or bound on the novel architecture. In the next step, I will (ii) improve EDA techniques to exploit
RRAM-based routing architecture (standard EDA tools do not support RRAM-based architecture well). Then I will (iii) design a RRAM-based FPGA chip and check the performance of optimized architecture on a silicon chip.

The rest of the write-up is organized as follows. Section II reviews the background knowledges, covering FPGA architecture and RRAM technologies. Section III surveys the three selected papers. Section IV introduces the research proposal.

II. BACKGROUND

A. FPGA Architecture

Fig. 1 depicts the conventional FPGA architecture with single-driver routing [6], 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 $J$. Previous works [20]–[8] conclude that $I = \frac{CN+1}{2}$ ensures over 98% utilization of CLBs. Commercial FPGAs [9]–[11] widely support fracturable LUTs [12] to reduce the critical path.

B. RRAM Technologies

As one of the most promising emerging NVM memories [14], RRAM technologies have been widely investigated [15]. As shown in Fig. 2(a), RRAMs are two-node electronic devices and are typically composed 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) [15]. 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 [13]. 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).

III. SURVEY OF SELECTED PAPERS

In this section, I first introduce the idea of FPGA RRAM-based Programmable Interconnects (RPI) [2] about transistor-less programmable interconnect. Then, I review the 3D Stacking RRAM-based FPGA design [1]. Finally, I survey a more general and practical circuit design for RRAM-based FPGA [3].

A. FPGA RPI [2]

The routing architectures in SRAM-based FPGAs are quite flexible but very expensive in terms of their large area, low speed and high power consumption. J. Cong et al. see the opportunities in a high-performance RRAM-based routing architecture by introducing the low on-resistance of RRAMs into data paths, and propose transistor-less programmable
interconnects. The high performance comes from that the on-resistance of a RRAM (∼1kΩ) is typically much lower than a pass transistor (∼4kΩ). In their paper, the RRAM technology in [17] is considered, where $R_{on} = 100\Omega$ and $R_{off} = 1M\Omega$.

Fig. 3. Circuit schematic of transistor-less programmable interconnects. [2] (a) RRAM-based connection block. (b) RRAM-based switch box.

1) Contributions: Fig. 3 illustrates the schematic of transistor-less programmable interconnects. All the pass transistors in the connection blocks and switch boxes are replaced with RRAMs. The programming transistors of RRAMs in the connection blocks inside a tile are heavily shared, as highlighted in red in Fig. 3(a). Each routing track has a programming transistor that is shared by the connected RRAMs. Each pin of logic block has a programming transistor that is shared by the connected RRAMs. The programming transistors of RRAMs in the switch boxes are intensively shared as well, as highlighted in red in Fig. 3(b). Each routing track requires a programming transistor that is shared by the connected RRAMs. Hence, in one tile of FPGA (highlighted in red rectangle in Fig. 1), the number of programming transistors is $M + 3W$, where $M$ denotes the number of pins of a logic block and $W$ represents the number of routing tracks in a channel. Since the transistor-less routing architecture consists of many RRAMs and only a few programming transistors, it can be move onto the top of logic blocks as shown in Fig. 4.

Another contribution of this paper is the on-demand buffering architecture. Instead of adding buffers for all the routing tracks, only a limited number of buffers are pre-fabricated in each channel of the novel routing architecture and they can be shared among the routing tracks in the channel. Buffers are allocated for the critical paths and the long paths in order to reduce their delays. For non-critical or short paths, buffers are not desperately needed because they can be relaxed and cast no impact on the performance. The authors also determine the optimal length of a wire between two adjacent buffers is 3.

2) Results: When compared to the conventional architecture, the RRAM-based FPGA with transistor-less routing architecture reduces the area by 96%, the delay by 55% and power consumption by 79%.

3) Shortcomings: The experimental results are too optimistic. The transistor-less routing architecture requires a very ideal RRAM whose $R_{on}$ is 100Ω. Configuring such a RRAM requires a 10mA programming current. In the 32nm CMOS technology, the driving current of a minimum width $n$-type transistor is ∼250µA. Therefore, the size of a single programming transistor should be ∼40, which will potentially cause large area overhead. In addition, the large programming transistor introduce large parasitic capacitance into the data path which will potentially weaken the performance gain. The proposal on the programming transistor sharing among the routing tracks is very smart and efficient but depends on a complex methodology in configuring the RRAMs. The connection blocks and switch boxes are decoded multiplexers and consume a lot of memory bits. These memory bits require huge memory banks to program them all.

B. Non-volatile 3D Stacking RRAM-based FPGA [1]

Yi-Chung Chen et al. investigate the opportunity in building a non-volatile 3D stacking RRAM-based FPGA. They point out that the transistor-less routing architecture requires an ideal RRAM which is hard to be fabricated. Hence, they do not introduce any RRAM into the data paths, but instead simply replace the SRAMs with RRAMs in a FPGA and utilize some optimized 3D RRAM-based structures.

1) Contributions: In the non-volatile 3D stacking RRAM-based FPGA, the SRAMs are replaced with Complementary Resistive Switches (CRS) [18]. CRS consists of two RRAMs, which share a programming transistor. The two RRAMs have opposite polarities and can be configured in one step with the shared programming transistor. A CRS-friendly structure, called 3D-HIM architecture, is used in LUTs. Compared to SRAM-based LUTs, the RRAM-based LUTs have the same functionality and can work in a second mode: runtime program configuring bits. In the second mode, the LUTs can be used as extra random access memories. In the routing architecture,
the pass transistors remain as they are in the SRAM-based FPGA but controlled by CRS structure, as depicted in Fig. 5. Similar to the work in [2], the connection blocks and switch boxes are implemented with pass-gate switches instead of multiplexers. Using the pass-gate switches can reduce the number of CMOS transistors in routing architecture and increase the area-efficiency of 3D structure.

![Switch Boxes based on pass-gate switches controlled by CRS structure](image)

Fig. 5. Switch Boxes based on pass-gate switches controlled by CRS structure [1].

2) Results: Compared to the conventional architecture, the proposed RRAM-based FPGA improves the area by 62.7%, the delay by 34%. However, compared to the RRAM-based FPGA that introduces RRAMs into the data paths [19], the proposed RRAM-based FPGA has extra 52.9% area, 19.8% delay and 5x power consumption.

3) Shortcomings: This paper introduces such a RRAM-based FPGA that does not require RRAMs with very low on-resistance. Without putting RRAMs into the data paths, the area and delay gains are still attractive. However, the power consumption becomes another serious concern. In every memory bit, the CRS structure creates a pure resistive path between VDD and GND. In such case, the leakage power is \( VDD^2/(R_{on} + R_{off}) \), which may potentially burn a large static power. To suppress the leakage power, the \( R_{on} \) as well as \( R_{off}/R_{on} \) should be high. Therefore, this work requires another ideal RRAM as well. Another drawback is the number of memory bits due to the pass-gate programmable switches. A large number of the memory bits requires a lot of CRS structures, resulting in high leakage power.

C. Generic Memristive Structure for RRAM-based FPGA [3]

1) Contributions: P.E. Gaillardon et al. propose a generic structure for RRAM-based multiplexers. CRS structure can be programmed in LRS+HRS or HRS+LRS, which natively meets the need of the tree-like multiplexer designs. Fig. 6(a) illustrates the circuit design of a 4 to 1 multiplexer implemented with three CRS structures. The first stage is composed of two parallel CRS structures and the third CRS structure constructs the second stage. To configure a single CRS structure in the multiplexers, three programming transistors are needed. For instance, the second stage has three programming transistor tagged \( n5, n6 \) and \( n7 \) in Fig. 6. In a fully-encoded multiplexer design, all the CRS structures in a stage share the same configuration. Hence, programming transistors can be heavily shared in such multiplexer designs. For example, in Fig. 6(a), the programming transistor of CRS 1, \( n2 \) and \( n5 \), are connected to CRS 2 in order to share the same configuration bits, and \( n5 \) is also shared by CRS 3 in programming the second stage. To construct a \( N(N = 2^k, k = 1, 2, 3... \) to 1 RRAM-based multiplexer, we need \( 2N - 2 \) RRAMs (\( N - 1 \) CRS structures) and \( 2N + 1 \) programming transistors. Configuring the RRAM-based multiplexers is similar to the SRAM-based multiplexers, and much easier than the programmable switches in [2]. As shown in Fig. 6(b), each stage of a multiplexer can be configured in one step. A \( N \) to 1 RRAM-based multiplexer consumes \( \log_2 N \) configurable bits, far less than what are proposed in [1], [2] (require \( N \) configurable bits).

![4 to 1 multiplexer with programming circuits and associated programming diagram](image)

Fig. 6. (a) 4 to 1 multiplexer with programming circuits and (b) associated programming diagram to configure output to input D1 [3].

IV. RESEARCH PROPOSAL

In this section, I first introduce the idea in modeling RRAM-based FPGA architecture. Then I describe the concept of low-power high performance RRAM-based FPGA. In the last part, I discuss the future work in my PhD.

A. Modeling the RRAM-based FPGA Architecture

All the selected papers deliver nice concepts that RRAMs can revolute the FPGA architectures, but lack a detailed modeling. A detailed modeling can help us in further optimizing...
the RRAM-based FPGA architecture. Here, I focus on the modeling the delay of a RRAM-based multiplexer design in [3].

Fig. 7. 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. 7(a) is the critical path of a 4-input RRAM-based multiplexer. Fig. 7(b) extends this to the general case of a n-stage RRAM-based multiplexer, while its equivalent RC model is given in Fig. 7(c).

The resistance and capacitance in Fig. 7(c) can be extracted from Fig. 7(b) and expressed as follows:

\[
\begin{align*}
R_0 &= \frac{R_{\text{inv}}}{W_{\text{inv}}} \\
R_i | 1 \leq i \leq n &= R_{\text{on}} \\
C_0 &= W_{\text{inv}}C_{\text{inv}} + 2W_{\text{prog}}C_{\text{off}} \\
C_i | 1 \leq i \leq n &= 2W_{\text{prog}}C_{\text{off}} \\
C_n &= C_{\text{L}} + W_{\text{prog}}C_{\text{off}}
\end{align*}
\]

(1)

where \( R_{\text{min}} \) denotes the equivalent resistance of a minimum size inverter, \( C_{\text{inv}} \) represents the parasitic capacitance at the output of a minimum size inverter, \( W_{\text{inv}} \) is the size of driving inverter in terms of the minimum width transistor [20]. \( R_{\text{on}} \) denotes the equivalent resistance of a RRAM in on-resistance state. \( W_{\text{prog}} \) represents the width of programming transistor in the unit of the minimum width transistor, and \( C_{\text{off}} \) is the parasitic capacitance of a minimum width programming transistor in off state.

Considering the Elmore delay [21] of the critical path of a general n-stage RRAM-based multiplexer (Fig. 7(b)), we obtain:

\[
\begin{align*}
\tau &= \sum_{i=0}^{n} R_i \sum_{j=1}^{n} C_j \\
&= R_{\text{min}}C_{\text{inv}} + R_{\text{min}} \frac{W_{\text{inv}}}{W_{\text{prog}}} C_{\text{L}} \\
&+ (2n + 1) \frac{R_{\text{min}}}{W_{\text{inv}}} W_{\text{prog}}C_{\text{off}} + n \cdot R_{\text{on}}C_{\text{L}} \\
&+ n^2 R_{\text{on}} W_{\text{prog}} C_{\text{off}}
\end{align*}
\]

(2)

As introduced previously, the on-resistance \( R_{\text{on}} \) of RRAM is dependent on the programming voltage \( V_{\text{prog}} \) and on the programming current \( I_{\text{prog}} \) [15], as follows:

\[
R_{\text{on}} = \frac{V_{\text{prog}}}{I_{\text{prog}}} = \frac{V_{\text{prog}}}{W_{\text{prog}} \cdot I_d}
\]

(3)

where \( I_d \) is the driving current of a minimum width transistor. With equation (3), equation (2) is converted to:

\[
\begin{align*}
\tau &= R_{\text{min}}C_{\text{inv}} + R_{\text{min}} \frac{W_{\text{inv}}}{W_{\text{prog}}} C_{\text{L}} \\
&+ (2n + 1) \frac{R_{\text{min}}}{W_{\text{inv}}} W_{\text{prog}}C_{\text{off}} + n \cdot \frac{V_{\text{prog}}}{I_d W_{\text{prog}}} C_{\text{L}} \\
&+ n^2 \frac{V_{\text{prog}}}{I_d} C_{\text{off}}
\end{align*}
\]

(4)

The relation between the n-stage multiplexer delay and the width of the programming transistor is depicted in Fig. 8.

Fig. 8. Relation between \( W_{\text{prog}} \) and delay of a RRAM-based multiplexer.

When \( W_{\text{prog}} \) is small, the delay increases due to the large on-resistance of RRAM. When \( W_{\text{prog}} \) 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. 8, there exists an optimal \( W_{\text{prog,opt}} \) giving the best performances by trading off the on-resistance with the parasitic capacitances from the programming transistors. Equation (4) reaches minimum value (best delay) when:

\[
W_{\text{prog,opt}} = \sqrt{\frac{nV_{\text{prog}}C_{\text{L}}W_{\text{inv}}}{(2n + 1)I_dR_{\text{min}}C_{\text{off}}}}
\]

(5)

In FPGA routing architecture, the number of the stages of multiplexers are diverse. As Equation 5 depends on the size \( n \) of the multiplexer, using a uniform size of programming transistors [1]–[3] should not ensure the best performance. To achieve the best performances, the multiplexers in FPGA should have different \( W_{\text{prog,opt}} \).

I also run SPICE simulations to verify the Equation 5. Equation 5 predicts that when \( V_{\text{prog}} \) decreases, \( W_{\text{prog,opt}} \) decreases. Experimental results in Fig. 9 verifies this prediction. Fig. 9 depicts the delay of a 32-input multiplexer extracted while sweeping \( V_{dd} \) and \( W_{\text{prog}} \). The curves, obtained for \( V_{dd} = 1.8V \) and \( V_{dd} = 1.4V \), are similar to the region...
highlighted in red in Fig. 8. In these two cases, the best performance is achieved when \( W_{\text{prog}} = 3 \) and \( W_{\text{prog}} = 2 \), respectively. The curve obtained for \( Vdd = 0.8V \) corresponds to the blue region highlighted in Fig. 8. In this case, the best performance is achieved when \( W_{\text{prog}} = 1 \). When comparing the three curves, we observe that the best performance shifts from \( W_{\text{prog}} = 3 \) when \( Vdd = 1.8V \) to \( W_{\text{prog}} = 1 \) when \( Vdd = 0.8V \) for a 32-input RRAM-based multiplexer.

B. Low-power High Performance RRAM-based FPGA

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. 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.

Here, we consider a local 32-input multiplexer. For the multiplexers of other sizes, the same conclusions can be reached. Fig. 10 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 does 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. 10 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.

C. Future Work

1) Modeling RRAM-based FPGA: The RRAM-based FPGA routing architecture is quite different from the SRAM-based one in terms of the delay modeling. In future, I intend to study the power modeling for RRAM-based FPGA and optimize the current RRAM-based architecture [2] [3]. And through modeling, I may propose a novel routing architecture exploiting RRAMs.

2) CAD for RRAM-based FPGA: With the idea about optimizing the RRAM-based architecture, I will develop CAD tools based on VTR flow [22] to exploit RRAM-based routing architecture. Current power estimation techniques do not well support the RRAM-based FPGA architectures. I plan to develop power estimation techniques that support RRAM-based FPGAs.

3) Circuit Design for RRAM-based FPGA: I plan to design a RRAM-based FPGA that combines RRAM-based architecture in [3] and my optimizations. I wish to check if these optimizations work well on a silicon chip.

REFERENCES


