R-Accelerator: A Reconfigurable Accelerator with RRAM Based Logic Contraction and Resource Optimization for Application Specific Computing

Zhengyu Chen, Hai Zhou, and Jie Gu
Department of Electrical Engineering and Computer Science, Northwestern University
2145 Sheridan Road, Evanston, IL, 60208, USA
Email: zhengyuchen2015@u.northwestern.edu, haizhou@northwestern.edu, jgu@northwestern.edu

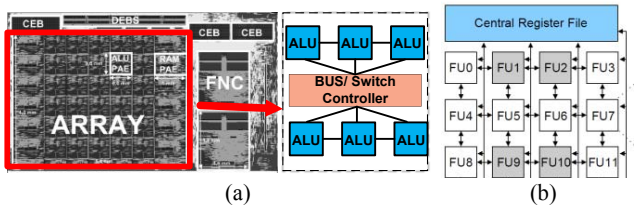
***ABSTRACT*—In this paper, we introduce a novel reconfigurable accelerator (R-accelerator) design which embeds RRAM device into traditional logic circuits for high-performance application specific computing. To facilitate the synthesis of the proposed RRAM based logic cell, a special logic contraction technique is developed to maximize the area saving. In order to optimize the arithmetic unit array for instruction set mapping and interconnect routing, a new resource allocation algorithm is also proposed to achieve further saving in area and power. Using a fully integrated design flow with commercial design tools, our experimental results show that the proposed RRAM based R-accelerator architecture offers 45% area improvement, 33% power reduction and 32% performance enhancement in a 45nm CMOS process compared with conventional CMOS design.**

***Keywords*—memristor, RRAM, logic contraction, reconfigurable architecture**

I. INTRODUCTION

Accelerator enriched microprocessor design has recently drawn tremendous interest from consumer electronic industry due to the rapid growth from applications such as virtual reality, artificial intelligence. Unfortunately, the complex algorithms utilized by such applications, e.g. facial recognition, lead to significant challenges to the existing IC development models. On one hand, the market of electronic devices pushes designs toward more complex and higher performance system-on-chip (SoC) devices. As a result, accelerator based ASIC chips or Application Specific Instruction Processors (ASIP) are developed to provide dedicated processing power for specific computing tasks [1,2]. On the other hand, the increasing IC design costs at advanced CMOS technology, and demand for shorter time-to-market cycle requires lower risk and faster turnaround solutions which pushes toward more flexible, reconfigurable hardware solution, diminishing the benefits of dedicated ASIC design. A typical example is the general-purpose microprocessor which provides flexible support to various applications but suffers from lower performance and higher energy cost for handling the complex algorithm in real-time image processing, such as median filter, and optimal margin classifier, etc. [3, 4]. Meanwhile, even though ASIC or ASIP solution does provide a performance boost of as much as three orders of magnitude to its CPU counterpart, it becomes impractically expensive to implement all potentially used complex algorithm in an ASIC or ASIP design [5]. For this reason, a reconfigurable accelerator-based hardware solution which provide combined advantages of both high performance and hardware flexibility has become increasingly popular recently [6,7].

As an alternative solution, Coarse Grain Reconfigurable Arrays (CGRA) represents a second class of reconfigurable architectures. In CGRA, predefined and optimized reconfigurable Arithmetic Logic Units (ALU) are used as building blocks. Complex algorithms are then allocated into ALU arrays to realize application specific instruction sets removing conventional performance limitation from multi-cycle pipeline operations. CGRA significantly reduces the overhead of interconnects and associated memory components compared with solution such as FPGA while keeping a high level of reconfigurability [6, 8]. An example of such architecture is shown in Fig. 1 (a) where totally 30 ALUs are connected to realize a complex operation such as square-root without intermediate storage or pipelines [1, 6]. Fig. 1 (b) shows another CGRA design which contains 16 nodes arranged in a 4×4 mesh [8]. A similar CGRA work with reconfigurable interconnect but focuses on providing single-cycle communications between functional units (FUs) is proposed in [9]. Although CGRA solution provides an enhanced performance, significant challenges still exist including (1) high cost in the design of reconfigurable ALU units with large amount of function support; (2) Associated memory costs for bookkeeping the configuration; (3) High cost of establishing flexible interconnects between Arithmetic Units (AU). In this paper, we try to address the above issues of CGRA design using novel cross-layer solutions ranging from device, circuits, to architecture levels.



(a) (b)

Fig. 1. (a) Layout and core circuit diagram of previous CGRA work [6]; (b) Block diagram of another CGRA work [8].

This work is funded by NSF grant CCF-1533656.

2576-6996/18/$31.00 ©2018 IEEE
DOI 10.1109/ICCD.2018.00034

Among recent technology development, the emerging non-volatile memory (NVM) technologies, i.e. RRAM or memristor have provided tremendous new opportunities to the development of high performance microprocessors. The use of RRAM/memristor can be classified into two large classes. The first class is in memory application where RRAM is utilized to provide on-chip NVM solution replacing conventional Cache or DRAM. In addition, the very simple thin film topology of RRAM allows monolithic integration of such memory device in vertical stacks on top of the processing units, leading to an extremely high-density storage solution [10]. In the second class, rather than using NVM as a storage unit, RRAM/memristor device has been proposed as processing units in emerging application such as neuromorphic computing [11]. Such an implementation dramatically reduced the cost of convolution operation, leading to tremendous saving of computing power. Furthermore, a mrFPGA technique was also proposed to use memristor as an interconnect solution to replace existing SRAM based FPGA, leading to 5X improvement in area [5]. Other relative work utilized RRAM as programmable interconnect mainly focused on the circuit level implementation [12, 13].

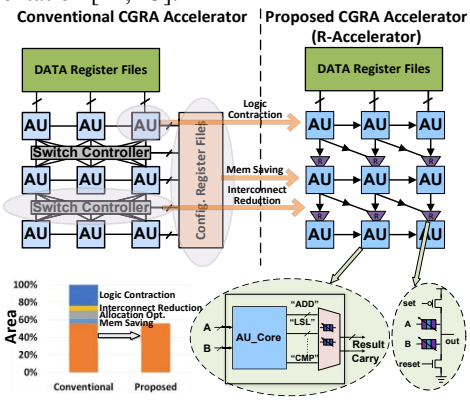


Fig. 2. Left: Conventional CGRA accelerator; Right: Proposed RRAM-based CGRA accelerator (R-Accelerator).

Different from the prior arts, this paper proposes a fundamentally new scheme for implementing CGRA-based accelerator using the emerging RRAM/memristor device, referred as R-Accelerator. Fig. 2 summarizes the proposed R-Accelerator architecture in the following aspects: (1) the conventional AU is replaced by RRAM-based reconfigurable AU with a specially developed logic contraction technique rending significant logic simplification; (2) The switch controller in conventional design is reduced and simplified by the RRAM-based programmable interconnection; (3) Configuration register file is eliminated as the RRAM works as both non-volatile storage and programmable interconnect. Below summarizes the cross-layer contribution of this paper from circuit, architecture to design automation:

- In the circuit level, (1) we propose a RRAM based logic cell design which works as both function units and storage units to realize logic reconfigurability; (2) We propose a special logic synthesis technique, referred as logic contraction, which dramatically reduce area overhead associated with realizing reconfigurability in conventional CMOS design.
- In the architecture level, we proposed the CGRA-based accelerator architecture with RRAM-based programmable interconnect.
- In the design automation level, (1) to leverage the help from memristive device and overcome the challenges of allocating arbitrary algorithm into CGRA AU arrays, we propose special allocation solution based on heuristic algorithm with thousands time of speedup; (2) A full layer implementation including PCell and backend supports have been developed and fully integrated into conventional EDA design flows/tools, leading to an automatic design flow of the proposed R-accelerator design.

III. RRAM-BASED RECONFIGURABLE AU DESIGN

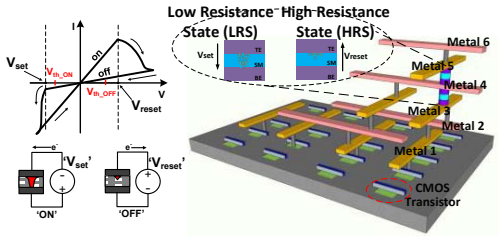
A. Memristor Devices

A memristor is a 2-terminal passive device in which the resistance between its terminals can be changed into a high resistance (OFF) state, or a low resistance (ON) state. The 'ON' and 'OFF' states are reversibly controlled by an external voltage applied across its terminals. The two major classes of application are (1) RRAM memory with binary states; (2) Analog memristor where the resistance states possess continuous tuning and larger range. Normally, higher resistance ratio is observed in the second-class due to the requirement of higher resolutions. Here we briefly survey the selected published RRAM characteristics in Table-1. As our paper explores a novel usage of the resistive memory as a non-volatile reconfigurable logic, we only utilize the RRAM as a binary state device, i.e. low resistive state (LRS) and high resistive state (HRS) with relaxation on the tuning resolution requirement of the device as compared with previous publication for neuromorphic computing [14]. Although matching of exact device characteristics to particular published device is beyond the scope of this paper, we devote our effort to incorporate more realistic behavior and electronic properties of the RRAM device by creating (1) A realistic parameterized cell (PCell) that can be extracted from real device layout and simulated by spice simulator to characterize the performance of the proposed circuits. (2) An integrated library cell that is fully supported by EDA tools, e.g. Cadence encounter, to perform synthesis, place&route of the developed circuit components.

Table 1: Selected Published RRAM/Memristor Characteristics

Material	LRS (ohm)	HRS (ohm)	Write Time(ns)	Set Voltage (V)
NiO [17]	500	10M	10~50	1.5
SiOxNy [20]	100	1M	100	1
TaO [21]	100	100k	2	1.5
HfO2 [22]	1k	100k	100	1.5
HfO2 [23]	5k	10~100M	100	2
This work	10k	1M	10	1.2

B. Electrical Characteristics of Memristors



(a) (b)

Fig. 3. (a) Typical bipolar I-V linear curve of memristor; (b) 3-D diagram of memristor based circuit.

Fig. 3(a) shows the representative bipolar I-V linear characteristics of such a memristor [10, 29]. To set (or program) the device into a memristive 'ON' or 'low resistance' state, a voltage Vset above the threshold Vth_ON is applied across its terminals until a conductive filament is formed. The conductive filament shorts the device and lowers its resistance to RON. Reverse operation is applied to "reset" the device into 'OFF' or 'high resistance' state. Recent development from commercial vendors such as Crossbar, inc has demonstrated a fully integrated monolithic solution where memristive device can be inserted between any metal layers of existing CMOS chips and possess a large on-off ratio, e.g. 1,000 [10, 30]. Fig. 3 (b) shows the 3-D diagram of memristor based circuit design where RRAM is inserted between metal 4 and metal 5 on top of CMOS transistor layers.

C. RRAM Model and PCell Design

Fig. 4 (a) shows our developed RRAM PCell including both schematic and layout. The PCell allows variable dimension as well as placement of RRAM at any user specific metal layers. For spice simulation, parasitics due to routing and via connections are first extracted from Calibre. The devices are then simulated with verilog model together with extracted parasitics. The VerilogA model was developed based on various reported resources [19]. In our design, we used one of the high-level metals to avoid causing congestion with local routing. The parasitic impact from the connection between local transistors and RRAM devices has been included in our extracted model and library cell component in our digital design environment. Fig. 4 (b) shows our integration of the developed RRAM device and RRAM-based circuits into digital design flow for large scale integration used in this work.

Practical issues of RRAM/memristor devices have been reported including (1) variability of the cell resistance especially at high resistance state and (2) endurance of resistance value during writing and reading [18,19,31]. Although an exact device level evaluation is out of scope of this paper, we performed variability test by varying HRS and LRS by a factor of 300% and evaluate the performance, i.e. cell delay impact. Our evaluation shows the performance variation is within 10% of nominal value because the logic cell delay is strongly influenced by intrinsic transistors from driving buffers rather than the interconnect variation. In addition, the proposed reconfiguration operation using RRAM device is only performed at beginning of each program and thus can be repeatedly tuned to eliminate potential variation of the device resistance. Compared with previous proposed neuromorphic application where high resolution and sturdy resistance of memristor device is required, the proposed circuits have significantly relaxed requirement on the variability and durability of the device and thus can be easily utilized from commonly used RRAM/memristor devices.

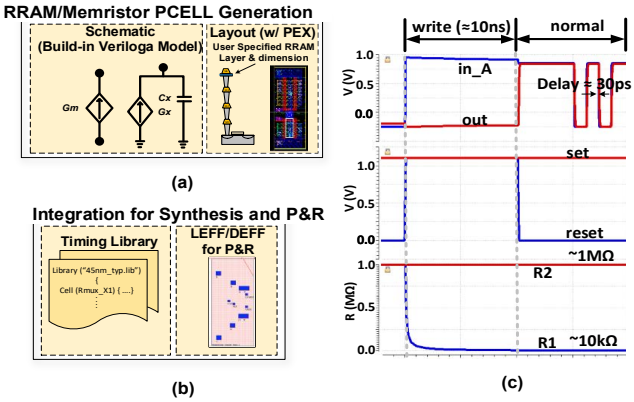


Fig. 4. Model and implementation of RRAM. (a) PCell layout and schematic; (b) Integration into conventional EDA flow; (c) Spice simulation waveforms of RRAM based MUX.

D. RRAM-Based Logic Cell Design

Multiplexer logic cells are heavily used in reconfigurable logics for realizing configurable functionality. Figs.5 (a) and (b) show two conventional implementations of MUX2 cell which contains 16 and 12 transistors with one select bit provided from a memory cell. As an alternative solution, we propose a RRAM-based MUX (R-MUX) cell with significant saving on cell footprint.

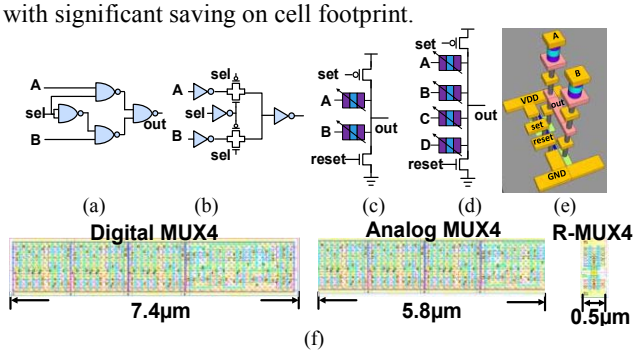


Fig. 5. Multiplexer design. (a) Conv. digital MUX2. (b) Conv. analog MUX2. (c) R-MUX2. (d) R-MUX4. (e) 3-D diagram of R-MUX2; (f) Layout comparison: Conv. digital MUX4 (left), conv. analog MUX4 (middle), R-MUX4 (right).

Fig. 5 (c) and (e) show our proposed RRAM based MUX2 (R-MUX2) cells and its 3-D drawing with only two memristors and two transistors. Since the R-MUX can be

fabricated over the logic blocks, integrating more bit into the R-MUX would not significantly increase the area of the device. Fig. 5 (d) shows the schematic of a 4-to-1 R-MUX with the same footprint as 2-to-1 R-MUX. For comparison, conventional 4-to-1 MUX takes 3 times area as 2-to-1 MUX. The footprint of the proposed R-MUX4 cell is reduced by 9X compared with conventional ones. In addition, there is no extra SRAM needed to store the configuration data since the RRAM works as non-volatile storage. Including memory space saving, an overall saving of 12X to 15X compared to conventional designs can be achieved. At configuration phase of the proposed circuits, by tuning on/off the set/reset transistors with particular input combination from previous stage, we can selectively program the RRAM into LRS and HRS, i.e. only one of RRAM stays in LRS while the rest stay in HRS. In this way, only the signal through the path with the RRAM in LRS will be passed realizing a multiplexer. The proposed programming sequence is (1) presetting primary input values for A, B, C, etc. and set, reset signals to program each RRAM device; (2) Repeat (1) for every stage of RRAM in a sequential order so that the input states are always defined in the logic circuits. Note that to avoid drifting of the resistive values, the configuration phase should be performed at elevated voltage above programming voltage, e.g. 1.2V, rather than the normal operation voltage, e.g. 1.0V. Fig. 4(c) shows the spice simulation of operation waveforms of the R-MUX circuits, exhibiting a write-speed of ~10ns, On-off resistive ratio of ~100 and a logic delay of ~30ps with correct functionality.

E. Logic Contraction using RRAM Based Logic Cell for Reconfigurable AU design

1) Reconfigurable Arithmetic Unit (AU) Design

In order to build the element of reconfigurable accelerator, we study the commonly used instructions in applications such as facial recognition and pattern classification. We identify the following most commonly used algorithms including multiplication(MUL), square, square root, division, winner-take-all (WTA), maximum value (MAX), absolute value (ABS), convolution, and multiply-accumulate operation (MAC). Accordingly, we build 8-bit arithmetic unit in our design containing 8 basic configurable operations: addition (ADD), subtraction (SUB), logic shift left (LSL), logic shift right (LSR), comparison (CMP), MUX, XOR and XNOR. A bypass mode is also introduced to pass input signals directly into the next stages. The AU units can then be jointly composed to realize the above complex algorithms.

The conventional AU uses combinational logics to realize configurability of different supported functionality. We experiment the area benefits of using proposed R-MUX by simply replacing some of the conventional MUXs by R-MUXs in the synthesized and P&R AU design Surprisingly, despite of the significant area reduction shown in Section II-C, our experiments show only 5% of area saving can be achieved by replacing conventional MUX by R-MUX. This is because in a highly-optimized gate-level netlist, most of the selection logic of the AU are synthesized into more complex logic, e.g. AOI logic gates which cannot be replaced. The left layout in Fig. 7 shows the conventional AU layout with MUXs marked by red color boxes. The area benefit from simply replacing existing MUXs with R-MUXs is insignificant due to the low occurrence of conventional MUXs because of the merging of selection logic with main functional logics from conventional synthesis methodology. We address this issue in the next section.

2) Proposed Logic Contraction Method and Flow

In this section, a novel control logic contraction technique is proposed to provide substantial area saving using RRAM based logic cells. In conventional design, an 8-bit AU is realized by the digital operation code to configure the AU core into different operation modes, e.g. ADD, SUB, CMP etc. For instance, an 8-bit 2-operation conventional AU with 1-bit operation code (c0) can be expressed as:

$$O_{conv-AU} = \bar{c_0}ADD(A,B) + c_0SUB(A,B) \quad (1)$$

It is important to highlight the overhead of the configurability. Even with a fully optimized adder/subtractor with technology mapping using complex standard cell, a selection logic for realizing eq. (1) requires approximately 16 ANDs, 8 ORs gates and 8 NORs for this 2-operaton AU. The block diagram of the conventional AU design is shown in Fig. 6 (a) with distributed selection logic circuits marked in grey. Compare to the AU core function, i.e. ADD/SUB, the selection logic circuit consumes a total of 20% to 40% area. On the other hand, the selection logic overhead can be dramatically suppressed by utilizing the proposed R-MUX and logic contraction technique for configuration. Its block diagram is shown in Fig. 6 (b).

$$O_{RRAM-AU} = M[ADD(A,B), SUB(A,B)] \quad (2)$$

Eq. (2) shows the expression of the RRAM-based AU design. The M function represents the R-MUX logic which consumes much less area when compared to the conventional digital MUX cells. More importantly, the selection logic operation in eq. (2) is eliminated from conventional expression as in eq. (1). As a result, the digital logic implementation using proposed RMUX becomes much simpler. In fact, the more functionality to be included in AU design, the more area saving can be achieved due to the small area consumption of R-MUX logic. The output expressions of 8-bit 4-operation conventional and R-MUX based AUs are shown in equations (3), (4):

$$O_{conv-AU} = \bar{c_1}\bar{c_0}ADD(A,B) + \bar{c_1}c_0SUB(A,B) + c_1\bar{c_0}LSL(A,Bc_1c_0) + LSR(A,B) \quad (3)$$

$$O_{RRAM-AU} = M[ADD(A,B), SUB(A,B), LSL(A,B), LSR(A,B)] \quad (4)$$

Unfortunately, standard logic synthesis technique does not support insertion of the RMUX logic. To facilitate the logic synthesis of the proposed RRAM based reconfigurable cells, we developed a special logic contraction flow. As illustrated in Fig. 6 (b), at first, we rewrite the RTL for AU by elaborating outputs from each supporting function and modify the high level top module with separation of core

functionality and selection logic. Secondly, we perform R-MUX integration using cells built from RRAM PCell through standard synthesis procedure of logic design. This method allows us to fully utilize the optimization power of modern synthesis tool while still integrating RMUX automatically into the final netlist and layout. The layouts of our proposed 8-bit 8-operation AU and conventional AU are shown in Fig. 7. A maximum area saving of 30% is achieved by introducing the logic contraction technique.

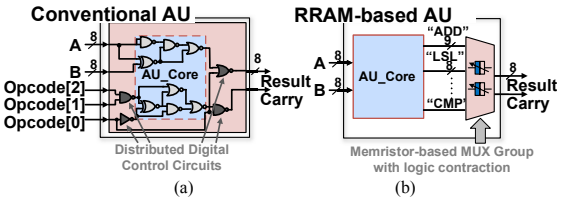


Fig. 6: AU circuit diagram. (a) Conventional; (b) RRAM-based.

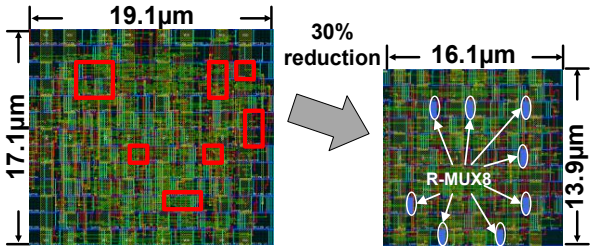


Fig. 7: Layout comparison between conventional (left) and RRAM-based AU (right).

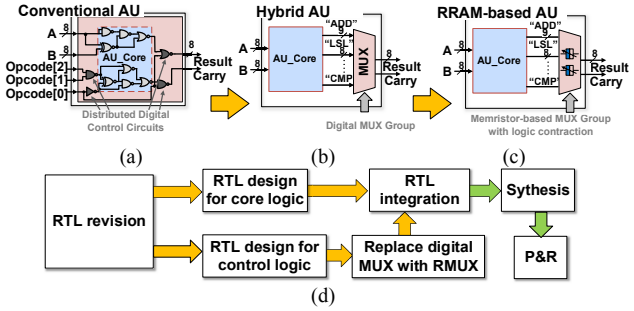


Fig. 8. (a) Conventional AU; (b) Hybrid AU; (c) RRAM-based AU; (d) Flowchart for logic contraction;

The proposed logic contraction method is a special method developed for effective insertion of RRAM based logic cells. Since different applications have different functionality defined, the benefits reported in previous section may not be universal to a general design. We provide a general logic contraction flow as shown in Fig. 8: (1) we first extract the "core function" from the conventional AU. This can be realized by eliminating the control logic in RTL design step. (2) We generate a separate control logic block which can be realized by groups of digital multiplexers, and connect it with the function core to form the "hybrid AU" as in Fig. 8(b). This "hybrid AU" realizes same functionality as conventional AU with an overhead due to logic separation. (3) Replace the digital multiplexers with RMUX to form the proposed RRAM-base AU which is shown Fig. 8 (c). The flowchart of proposed method is shown in Fig. 8 (d).

IV. RECONFIGURABLE AU ARRAY ARCHITECTURE

A. *Interconnect and Reconfigurable AU Arrays*

In order to implement instruction sets, the reconfigurable AU array which consists of a group of RRAM-based AUs should be configured into different logic topology to adapt to the data path of particular instructions. Conventionally, the reconfigurable AU array consists of AUs and switch controller (SC) [8]. The architecture of conventional interconnect network with SC is shown in Fig. 9 (a).

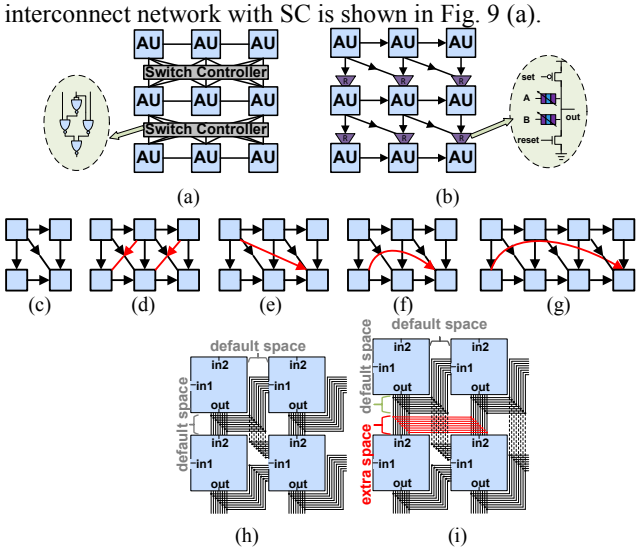


Fig. 9. AU array interconnect. (a) Conventional interconnect; (b) Proposed RRAM-based interconnect; (c) Option 2; (d) Option 3; (e) Option 4; (f) Option 5. (g) Detailed routing of (c); (h) Detailed routing of (d) which shows the overhead of extra routing channel.

To reduce the complexity of interconnect network, in our work, the conventional switch controller is simplified into a basic uni-directional interconnect in which only one signal propagation direction is allowed in one signal channel. Beside the basic interconnect, four more optional routing channels depending on the demand of target instruction sets. As a result, we introduce totally five interconnect options: (1) The basic interconnect which is shown in Fig.9 (c); (2) Adding an extra diagonal interconnect channel which is shown in Fig.9 (d); (3) Adding an extra free interconnect channel between nearby rows which is shown in Fig.9 (e); (4) Adding an extra interconnect channel within same row but between different columns which is shown in Fig.9 (f); (5) Continually adding extra routing channels within the same row which is shown in Fig.9 (g).

The proposed uni-directional network dramatically decreases the congestion on interconnecting wires compared with fully connected crossbar. The interconnect delay and area overhead can be further reduced when introducing proposed R-MUX logic to replace the conventional switch

controller (SC) which consists of a large amount of conventional MUXs.

B. Interconnect Options and Modeling

The more interconnect routing options included, the more flexible the AU array can be to implement complex instructions. In our work, the default interconnect of the AU array is uni-directional. Interestingly, no matter where the input port 1, input port 2 and output port are located, there occurs a free channel for the diagonal interconnects (from top left to bottom right) without introducing extra routing space. Fig. 9 (h) show detailed routing diagram according to default routing in Fig. 9 (c). If more options are offered in the configurable AU array, extra routing space will be needed. Fig.9 (i) shows the detailed routing for routing option in Fig. 9 (d) and the extra routing channel is marked with red. Despite of the extra routing overhead, more interconnect options can lead to less number of AUs to be implemented for the target instruction sets. Thus, there exists an optimal interconnect solution which leads to the minimum area of AU array for supporting target instruction sets which will be discussed in Section IV. The cost function of interconnect is shown as:

$$S_{ALUarray} = N_{row} \times (H_{AU+}H_{intercnct}) \times N_{column} \times (L_{AU+}L_{intercnct}) \quad (5)$$

where Nrow, Ncolumn are the numbers of row and column, HAU is the height and LAU is the length of an AU, Lintercnct, Hintercnct are the extra routing cost of the extra interconnects. The final area of the configurable AU array is determined by both the number of AU (array topology) in the array and the extra routing space introduced by the extra interconnects.

V. INSTRUCTION-TO-AU ARRAY SYNTHESIS AND ALLOCATION ALGORITHM

An important challenge for reconfigurable AU based accelerator design is the allocation of instruction set into existing AU arrays. Several resource-constrained allocation algorithms were proposed in [8, 9, 15]. However, the prior work were focused on instruction scheduling based on existing fixed numbers of processing units. Different from prior work, in this paper, we explore optimal design choices where the number and interconnects of AU arrays are not predefined but a minimum number of target instruction sets are provided. Hence, our work is orthogonal to prior scheduling focused study [8, 9, 15] which cannot be used to provide us the optimal design choices. Generally speaking, this kind of graph allocation work is NP-Hard [16]. We proposed an allocation algorithm based on heuristic algorithm to find the optimal numbers of AU and optimal interconnect options as given in Fig. 9 to achieve minimum area cost.

A. Instruction Decomposition

In this work, all instructions are represented by Dataflow Graph (DFGs). In the DFGs, each vertex represents a basic function operation such as addition, subtraction, logic shift, etc., and each edge represents the data dependency between the connected operations. In order to allocate a particular instruction into the AU arrays, this instruction needs to be decomposed into several operations. The flow of the function decomposition can be comprehended as follows: (1) decompose the complex instruction into several simple operations which can be realized by a single AU; (2) Generate the Dataflow Graph (DFG) of the decomposed instruction. For example, a four-input maximum can be decomposed into two stages two-input maximum: MAX(a,b,c,d) = MAX(MAX(a,b),MAX(c,d)); MAX(a,b) = MUX(a,b,CMP(a,b)).

B. Simultaneous Allocation Algorithm (SAA)

The key idea of simultaneous algorithm is considering all instructions at same time with all the operations allocated sequentially in the given topological order. Fig. 10 shows the allocation growing path of the instruction set consists of 7 instructions based on simultaneous algorithm. First, all the operations whose input are from previous pipeline stage are allocated into the first row of the AU array. Then the rest operations whose input are not from previous pipeline stage are allocated simultaneously and try to share as many common AUs as possible. The allocation result of utilized AU for allocating of first 5 operations is shown as the very left graph of Fig. 10. After the 1st operations of all instruction sets are allocated, the 2nd, 3rd …nth operations will be allocated into the AU array continuously.

Simultaneous Allocation Algorithm

Input: DFGs of Instruction set and data path *P* of the 5*5 AU array
Output: Minimum Area of configurable AU array
1: Allocate the 1st group operations
2: Initialize the complexity of each instruction
3: **for** j = (j_{MAX_1st} +1) .. (j_{MAX_1st} + j_{MAX_2nd}), **do**
4: **for** *i* based on the complexity order, from high to low **do**
5: **repeat**
6: Allocate the operation O_{ij}
7: **until** find the location with minimum cost in that row
8: **end for**
9: Recalculate the complexity of each instruction
10: **end for**
11: Calculate the total rectangular area of the allocated ALUs *S*
12: **return** *S* and the floor plan of AU array

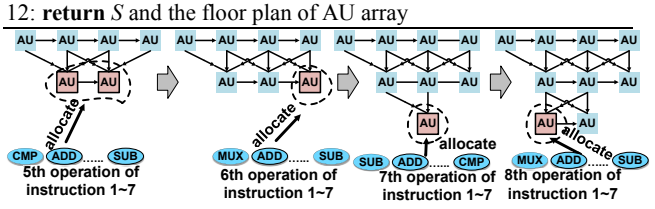


Fig. 10. Allocation algorithm growing paths.

Our tests on instruction groups show that the simultaneous allocation consumes much less runtime but with similar optimization results compared to exhaustive search because its searching space is much smaller than the exhaustively search.

VI. RESULTS AND ANALYSIS

To estimate the performance (delay) improvement, we performed "case-based" static timing analysis (STA) where delay is reported under constraint of configuration of the design rather than general worst-case critical path. The "case-based" STA reported by Cadence encounter tool provides more accurate estimation of specific operation of the accelerator. Table 2 shows the delay improvement for each configuration of the RRAM based AU compared with conventional AU. Table 3 shows the power saving of under different configuration using similar "case based" power analysis in Encounter tool.

Table 2: Delay reduction of different operations in AU.

Function	ADD	SUB	LSL	LSR	CMP	XOR	MUX
Reduction	34%	32%	23%	18%	41%	21%	36%

Table 3: Power saving of different operations in AU.

Function	ADD	SUB	LSL	LSR	CMP	XOR	MUX
Saving	38%	36%	27%	27%	27%	25%	36%

A. Case Study on General Benchmarks

1) Benchmarks

To evaluate the performance in general use cases, we selected a set of six benchmark programs from Mibench [27]. The selected benchmarks are fft, adpcm, basicmatch, bitcount, qsort and rigndael, all of which have repetitive computing loops suitable for mapping into reconfigurable accelerators. For each benchmark, we first used LLVM [28, 30] to obtain the DFGs and representative computing loops. We then identify custom instructions from the DFGs of each benchmark using instruction decomposition described in Section IV-A. The characteristics of representative loops are shown in Table 4.

Table 4: Benchmark Characteristic

Bench mark	Nodes	Edges	Domain
fft	28	41	Telecom
adpcm	27	47	Telecom
basicmatch	18	25	Automotive
bitcount	16	28	Automotive
qsort	27	42	Automotive
rigndael	16	28	Security

2) Performance Results

Fig. 11 shows the detailed implementation of the example benchmark "qsort". The DFG of representative loop in "qsort" contains three multiplications: two additions and a square root. Fig. 13 (a) shows the final configuration floorplan of the R-accelerator which contains a 5×6 AU array. Fig. 12 (a), (b) show the power and delay comparison between proposed R-accelerator and conventional reconfigurable architectures through different benchmarks. The overall geometric mean of power saving and delay improvement are 33% and 32% respectively. For power saving, "fft" has the highest energy saving of 37%. This can be explained by the fact that "fft" has largest number of multiplication (addition) operations which have higher saving compare to other operations. The highest delay improvement is 38% achieved by "adpcm" due to dominant MUX operations which have higher delay reduction compare to other operations. Fig. 12 (c) shows the total area saving of 45% achieved by the proposed R-accelerator design. Among the 45% area reduction, 25% comes from proposed logic contraction, 9% comes from allocation algorithm, 4% comes from interconnect and 7% comes from associated memory.

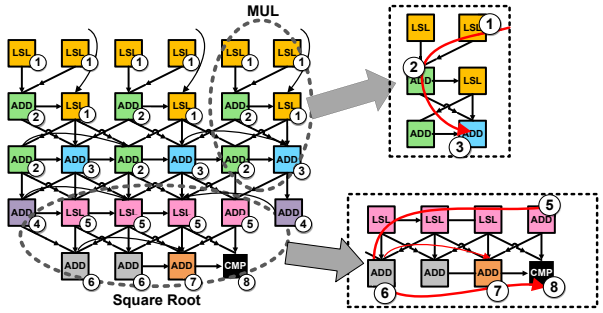


Fig. 11. (a) AU array and interconnection configuration diagram for "qsort"; (b) Detailed configuration road map.

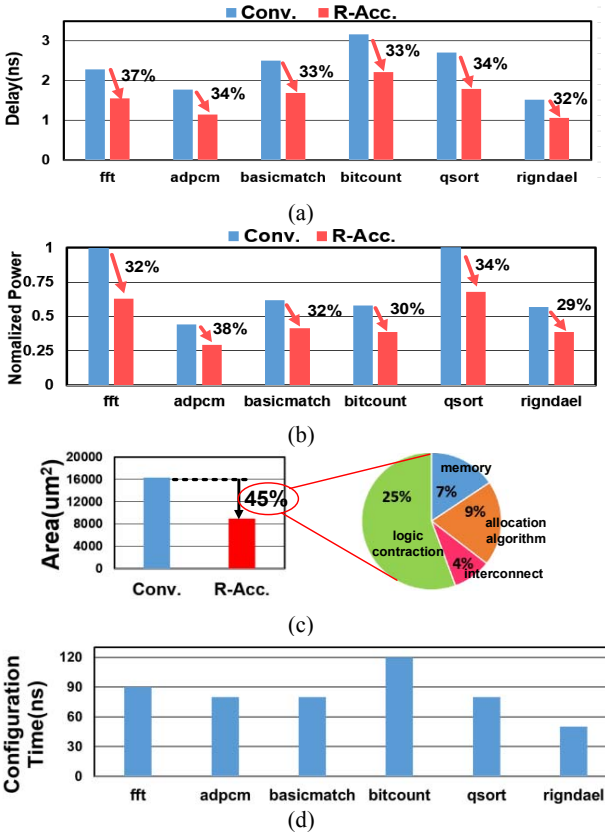


Fig. 12. (a) Power comparison; (b) Delay comparison; (c) Area comparison; (d) Configuration time comparison.

3) Configuration Time Analysis

One major difference of the proposed reconfigurable R-accelerator is that the RRAM needs to be written for each configuration. Compared with standard CMOS logic, the written speed of RRAM is slower and thus leads to drawback

of the proposed design. In addition, a special sequence needs to be enforced for configuration to ensure deterministic writing to each RRAM device. The write time is assumed to be a moderate speed of 10ns as listed in Table 1. Figs. 11 (a) and (b) show the configuration flow of the accelerator for "qsort" program. The different color indicates the order of reconfiguration: (1) the AUs with same color /number are reconfigured at same time; (2) The configuration order of the RRAM-based AU is marked by the number at the right-bottom corner of the AU. The detailed reconfiguration of multiplication and square root are shown in Fig. 11 (b). Fig. 12 (d) shows the configuration time for different benchmarks. The configuration time varies from 50ns to 120ns and among them "bitcount" has the longest configuration time due to a more complex and longer DFG it has. Although the longer configuration time of the proposed R-Accelerator is a drawback compared with conventional design, the overhead of reconfiguration only happens once at the beginning of the program and can often be hidden through careful scheduling with CPU's operation.

VII. CONCLUSION

This paper proposes a novel design methodology for creating reconfigurable application specific accelerator using emerging RRAM device. A novel RRAM based logic circuit and a logic contraction technique were proposed to significantly reduce the area cost of reconfigurable arithmetic units. To construct CGRA- based accelerators from the proposed RRAM based AU units, a new heuristic allocation algorithm is developed to achieve the optimal solution for AU placement and interconnect choices. The proposed techniques have been fully integrated into commercial EDA tools in 45nm technology. Case study general benchmark programs are used to highlight the significant improvement of the proposed R-Accelerator technique. Experiment results show that compared with conventional design methodology, a 45% area reduction, 32% of average delay improvement, and 33% of average dynamic power saving can be achieved with the proposed R-accelerator technique.

ACKNOWLEDGMENT

This work is funded by NSF grant CCF-1533656.

REFERENCES

[1] Michele Borgatti, et al. A reconfigurable system featuring dynamically extensible embedded microprocessor, FPGA, and customizable I/O. *JSSC*, 2003.

[2] Tung Thanh-Hoang, et al. A Data Layout Transformation Accelerator: Architectural support for data movement optimization in accelerated-centric heterogeneous systems. *EDAA*, 2016.

[3] Takuki Nakagawa, Tadashi SHIBATA. A real-time image feature vector generator employing functional cache memory for edge flags. *ISCS*, 2009.

[4] B. Boser, L Guyon, Isablele M. Guyon. A training algorithm for optimal margin classifier. ACM Workshop Computational Learning Theory, 1992.

[5] Jason Cong, et al. mrFPGA: a novel FPGA architecture with memristor-based reconfiguration. IEEE International Symposium on Nanoscale Architectures, 2011.

[6] Davide Rossi, et al. A heterogeneous digital signal processor for dynamically reconfigurable computing. *JSSC*, 2010.

[7] Fang-Li Yuan, et al. A multi-granularity FPGA with hierarchical interconnects for efficient and flexible mobile computing. *JSSC*, 2015.

[8] Hyunchul Park, et al. Edge-centric modulo scheduling for coarse-grained reconfigurable architectures. *PAC*, 2008.

[9] Manupa Karunaratne, et al. HyCUBE: A CGRA with reconfigurable single-cycle multi-hop interconnect. *DAC*, 2017.

[10] Crossbar, Inc online white paper. http://www.crossbar-inc.com/assets/resource/whitepaper/Crossbar-RRAM-Technology-Whitepaper.pdf

[11] Yong Shim, et al. Low-power approximate convolution computing unit with domain-wall motion based "spin-memristor" for image processing. *DAC*, 2016.

[12] Anne Siemon, et al. A complementary resistive switch-based crossbar array adder. *JETCAS*, 15

[13] Debjyoti Bhattacharjee, et al. Fast comparator implementation using 1S1R ReRAM crossbar arrays, *APCCAS*,16.

[14] Ping Chi, et al. PRIME: a novel processing-in-memory architecture for neural network computation in reram-based main memory. *AISCA*, 2016.

[15] Yuankai Chen, Hai Zhou. Resource-constrained high-level datapath optimization in ASIP design. DATE, 2013.

[16] Michael R. Garey, *Computers and Intractability*, Bell Laboratories, Murray Hill, New Jersey.

[17] K. Tsunoda, et al. Low power and high speed switching of ti-doped nio reram under the unipolar voltage source of less than 3 V. *IEDM*, 2007.

[18] Akifahara, et al. An 8mb multi-layered cross-point reram macro with 443mb/s write throughput. *ISSCC*, 2012.

[19] Chris Yakopcic, et al. Memristor spice model and crossbar simulation based on devices with nanosecond switching time. *JCNN*,2013.

[20] Ru Huang, et al. Resistive switching of silicon-rich-oxide featuring high compatibility with CMOS technology for 3D stackable and embedded applications. Appl. Phys A, 2011.

[21] Feng Miao, et al. Anatomy of a nanoscale conduction channel reveals the mechanism of a high-performance memristor. Adavance d matirials, 2011.

[22] T. Diokh, et al. Investigation of the impact of the oxide thickness and reset conditions on Disturb in HfO2-RRAM integrated in a 65nm CMOS. *IRPS*, 2013.

[23] Jiun-Jia Huang, et al. One selector-one resistor (1S1R) crossbar array for high-density flexible memory applications. *IEMD*, 2011.

[24] Xifan Tang, et al. Circuit designs of high-performance and low-power RRAM-based multiplexers based on 4T(ransistor)1R(ram) programming structure. *TCS*, 2016.

[25] Xifan Tang, et al. A high-performance low-power near-V_t RRAM-based FPGA. *ICFPF*, 2014.

[26] Tomasz Tala´ska, et al. Analog programmable distance calculation circuit for winner takes all neural network realized in the CMOS technology. *TNNLS*, 2016.

[27] Mibench, www.eecs.umich.edu/mibench/.

[28] LLVM, www.llvm.org.

[29] Miguel Angel Lastras-Montaño, et al. A low-power hybrid reconfigurable architecture for resistive random-access memories. *HPCA*, 2016.

[30] P. Biswas, et al. ISEGEN: generation of high quality instruction set extensions by iterative improvement. DATE, 2005.

[31] Sylvain DUBOSI. Crossbar Resistive RAM (RRAM): The future technology for data storage. http://www.snia.org/sites/default/orig/-DSI2014/presentations/HotTopics/SylvainDuBoise_Future_Technology_final.pdf