Increasing Reconfigurability with Memristive Interconnects

John Demme*, Bipin Rajendran†, Steven M. Nowick*, Simha Sethumadhavan*
*Columbia University, Dept. of Computer Science, NY, USA
†IIT Bombay, Dept. of Electrical Engineering, Mumbai, India
jdd@cs.columbia.edu, bipin@ee.iitb.ac.in, {nowick, simha}@cs.columbia.edu

***Abstract*—The design of on-chip interconnects is largely governed by the size and power of the devices being connected. While large components like memory controllers, video decode accelerators, and cores can afford the overhead of a large packet switching NoC router, smaller components like adders or other ALUs cannot. Instead, they are typically connected via simple wires, limiting their runtime reconfigurability. The notable exception – FPGAs – use an interconnect which allows extreme reconfigurability, but the FPGA pays for it in area, power, and latency costs. Less costly reconfigurable interconnects, therefore, could allow hardware designers to expose more reconfigurability while limiting area and power costs.**

This paper presents the design of a high-radix circuit switching crossbar design using memristors. This design utilizes Phase Change Memory (PCM), overcoming some of its limitations such as leakage power and low voltage operation. The very small size of memristors shrinks the area, power, and latency of crossbars by up to 16x, 4.4x, and 2.4x, respectively, leaving little interconnect overhead but wiring overhead. As a tool for designers, memristive interconnects offer significant potential to increase runtime design flexibility.

I. Introduction

What is the correct granularity for accelerators? Should accelerators be large, super-specialized blocks or small, reconfigurable computational elements which are dynamically composed? Currently, both styles exist though in different contexts. In the mobile space where energy efficiency trumps runtime configurability, large monolithic blocks are common. On the other end of the spectrum, FPGAs are becoming more popular in settings where reconfigurability matters more than energy. Ideally this dichotomy would not exist and accelerators could be both efficient and flexible.

One of the major reasons for fixed function hardware's efficiency and lack of configurability is its use of fixed connectivity wires. Dynamically configurable alternatives to simple wires like multiplexers, crossbars, or networks afford runtime flexibility in dataflow but consume both significant area and power. As a result, designers must trade off reconfigurability for efficiency and cost. If interconnection switches were smaller and more energy efficient, however, accelerators could be designed which have greater breadth while remaining nearly as energy and area efficient.

The impact of interconnection overhead is a phenomenon which is readily observed. Figure 1 shows the area of a number of components. It is interesting to note the position of the NoC router in this figure relative to other components. Components smaller than the router are rarely exposed to the NoC interconnect – at best they are connected via multiplexers, but typically connected via fixed wires. Many of the components larger than the NoC router are generally exposed to the NoC, allowing programmers to use them to accelerate a broader computation.

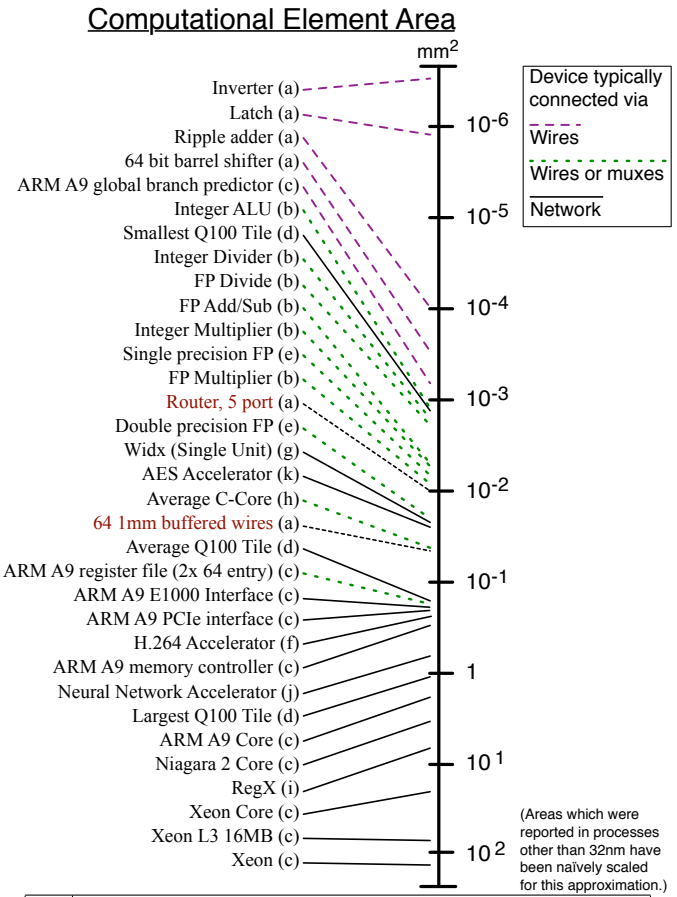

Fig. 1: The approximate area of some devices in 32nm. We note that while the larger devices are often connected together via dynamic networks, small devices are typically connected via wires or simple interconnection devices like multiplexers or crossbars.

To enable systems which are both reconfigurable and efficient, this paper discusses the design of crossbar switches built using memristors like Phase Change Memory (PCM). Compared to traditional CMOS pass gate crossbars, memristor crossbars are physically smaller, more power efficient, and efficiently scale to larger radices. To our knowledge, we are the first to propose high-radix memristive crossbars. These memristive crossbars are up to 16x smaller, 4.4x more power

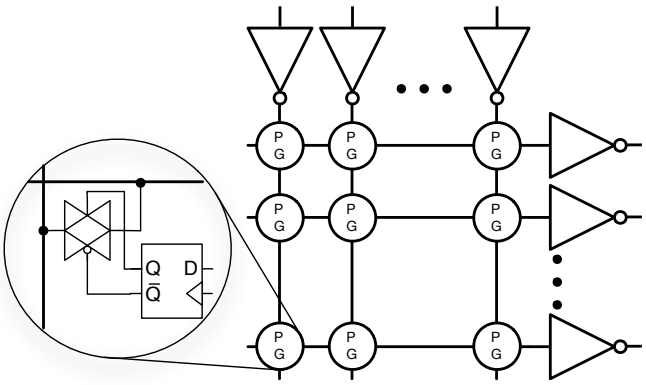

Fig. 2: A CMOS pass gate crossbar

efficient, and 2.4x faster than pass gate crossbars. As a result, designers using memristive crossbars can implement significantly more reconfigurability with far less area, power, and latency overheads than was previously possible.

The remainder of this paper is structured as follows: First, in Section II we detail the design of memristor crossbars circuits as compared to traditional CMOS passgate crossbars. Section III presents a detailed evaluation of their area, power, and latency. We review potential pitfalls in this paper's modeling methodology in Section IV. We discuss related work in Section V and conclude in Section VI.

II. Memristive Crossbars

Crossbars are one of the core components in many interconnects. This section briefly reviews a CMOS crossbar design which serves as a baseline, provides an overview of memristors, and then details the design of a memristor based crossbar.

A. CMOS Crossbars

Figure 2 shows the design of a pass gate crossbar. Each of M inputs drives a column wire which is connected to N output wires through N complimentary pass gates. In order to route signals, the pass gates are selectively enabled. For instance, if output j is to be driven by input i, then the pass gate at row j, column i is enabled and all other pass gates in that row are disabled. Each pass gate is driven by a latch, the set of which store the crossbar's routes.

One significant problem with all crossbars is that the number of pass gates scales quadratically with inputs and outputs so the area required to build high radix crossbars becomes prohibitive. Additionally, each pass gate adds source and drain capacitance to both the row and column to which it is connected. As a result, the energy required to transmit a single bit through the crossbar grows linearly with size, so maximum total power increases quadratically. Generally speaking, these phenomena limit the practical radices of crossbars.

B. Memristors

A new set of emerging technologies may soon fill the role of storage class memories. Chief among them are resistive

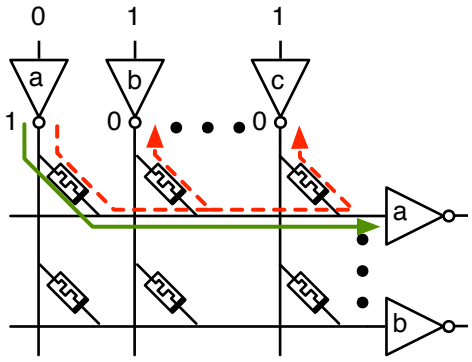

Fig. 3: In a naïve memristive crossbar implementation, sneak paths cause power and correctness issues. This figure shows the correct current path in green and sneak paths in red. These red paths lead to static leakage and voltage drop on the transmission line.

memories.[1] These memory cells work by switching between high or low resistance states (HRS and LRS, respectively) in response to programming. As an ideal model, memristors can connect two wires (with their low resistance state) or disconnect them (with their high resistance state). In this ideal model, memristors can directly replace both the pass gates and controlling latches in a crossbar to create an extremely small, efficient crossbar, as shown in Figure 3.

Unfortunately, actual devices are not ideal. There are two major problems with using them in this context:

a) Read Disturbance: Memristor state is generally programmed by exposing a cell to a "high" programming voltage. This programming voltage, however, is often quite low – for some technologies it can be as low as 0.3V, though it can be as high as 0.7V for others. As a result, if one were to pass a typical logic voltage (32nm V_{DD} is about 0.9V) through them, the programmed state would be lost and the crossbar would not operate correctly.

b) Low HRS/LRS Ratio: Real memristors have finite resistances in their HRS and non-zero resistances in their LRS. In fact, LRS resistances of more than several kΩ are typical and high resistances range from only a single order of magnitude higher to 7 orders of magnitude higher depending on the technology. Realistic HRS and LRS resistances lead to two issues in crossbars, as demonstrated in Figure 3.

The first problem is correctness. Assume the leftmost input is set to 0 and the rest 1, yielding 1 Volt from input inverter 'a' and 0V from the rest (assuming a V_{DD} of 1V). Ideally, output inverter 'a' would also see 1V, however even though the other memristors in its row are set to their high-resistance state, they still conduct at least a little, pulling down the voltage. Worse yet, in this case they all pull down in parallel. In fact, this input pattern (and its inverse) represent worst case scenarios in terms of pulling rows' voltages from their correct value. To quantify the problem, if there are $n+1$ inputs and the HRS/LRS ratio is

[1] The original definition of "memristor" does not include resistive memories. However, it has been more recently argued [12] that the definition can be generalized to include them.

r, then the inverter will see a voltage of approximately $\frac{r}{n+r}$.[2] Thus, if the crossbar size (n) is too large and the HRS/LRS ratio (r) too low, an incorrect bit will be transmitted.

Power is second problem with the sneak paths illustrated in Figure 3. Even if the HRS/LRS ratio is high enough to ensure correct operation, current is constantly leaking through the memristor array through the dashed red path and many others (not shown) in the array. As a result, this memristor crossbar has extremely high static power dissipation.

C. A Dual Array Memristor Crossbar

The problems with using memristors in crossbars can be mitigated to some extent. Figure 4 shows a design which fixes the read disturbance problem and mitigates power leakage due to low HRS/LRS ratios. This crossbar differs in two important ways: first, it integrates a latch into the output. This allows us to activate the array for only short bursts at the end of each clock cycle, reducing the amount of static leakage power. Second, it uses two arrays instead of one (shown overlaid atop one another in Figure 4). One of the arrays is responsible for conducting V_{DD} and the other responsible for conducting V_{SS}. The advantage of this scheme is twofold: it reduces the voltage across each memristor (helping with the read disturbance problem) and it also reduces power loss through sneak paths.

a) Operation: Memristors' states are set with the same pattern in each array: one in each row is set to a low resistance state (the one corresponding to the column from which that row is to be driven) and the rest are set to high resistances. During each cycle, inputs arrive on each input pin. When a zero is present on a pin, the pMOS in the transmission module is active and the V_{DD} array is pulled up. When a one is present, the opposite happens: the nMOS is activated and the V_{SS} array is pulled down. This operation, however, does not result in any leakage because the V_{DD} array is only pulled to V_{DD}, the V_{SS} array to V_{SS} and the two arrays are not connected.

Towards the end of that clock cycle, the values being transmitted must be read and latched. During this stage, the dual-rail enable signal is pulsed for a short amount of time (between 25 and 200 picoseconds). Essentially, these latches must run on a separate, synchronous clock with a short duty cycle. When enabled, each receiver's pair of bridging transistors connecting the corresponding rows in each array are activated. They allow current to flow between the V_{DD} and V_{SS} arrays, forming a voltage divider in the middle of the bridge. On each side of the bridge, a single memristor in a low resistance state connects the bridge to the transmission module from which it is attempting to read. However, only one of those two memristors is being directly pulled up or down by its transmission module. As a result, if the HRS/LRS ratio is high enough, the bridge will output a voltage on the correct side of $V_{DD}/2$.

While the enable pulse is asserted, the output latch also becomes transparent and captures the value which was transmitted after the enable pulse is de-asserted, allowing subsequent components to continue using the value. While these output latches increase the area and power dissipation of the crossbar, a pipelined interconnect must contain latches regardless of the crossbar technology. Accordingly, this memristor crossbar design is suited only to pipelined interconnects – which have become more and more common as wire delays have become more prominent in smaller processes.

b) Potential Area & Power Advantages: Despite this new design having two arrays instead of just one, we expect it to be smaller and more energy efficient than its CMOS pass gate counterpart. Memristors are extremely small, so the arrays themselves are extremely small. Pass gate arrays, in contrast, contain 8 transistors at each junction (2 for the pass gate, 6 for the latch). Additionally, the memristors add little or no capacitance to the wires to which they are connected. The pass gates, however, add a source and drain capacitance to each of the two wires to which they are connected, significantly increasing the energy required to drive each row and column.

We also expect memristive crossbars to scale to higher radices better than CMOS pass gate crossbars. While the input and output circuitry for the memristive crossbar are more complex than the traditional crossbar, they matter less (in terms of area and power) as the crossbar scales up since they scale linearly with its size. The array, however, scales quadratically with size, so the power and area of its junctions dominate. Memristors, being smaller than even a single transistor, will outperform any silicon device, thus the overall memristive crossbar will be smaller than CMOS when their sizes are sufficiently large.

D. Memristive Technologies

Thus far the memristor crossbar design has remained generic, not tied to any particular memristor device technology. This paper examines one particular device technology: phase change memory (PCM). PCM is in production with large companies like Samsung and Micron announcing products within the last few years [13], [14]. Other memristive memory technologies are in research and development at various phases, though none are as far in development as PCM. The design presented in this section can be applied to other resistive memories (we have also designed crossbars based on Redox memory). This paper does not present those results both for brevity and because other memory technologies showed only marginal improvement over PCM for crossbar usage while being less mature.

a) Phase Change Memory: PCM is a resistive memory element which operates by changing the physical state of a small volume of an easily melt-able material, most commonly chalcogenide glass. In other words, the resistivity of the cell is defined by putting it in either an amorphous or crystalline state. This state is "programmed" into the cell by heating it (often with a small Joule heater just below the cell) then cooling the cell either quickly or slowly, either annealing it into a conductive crystalline structure or creating a highly resistive amorphous material. [15]

There are three important characteristics of PCM: their size, their HRS and LRS resistances, and the way they are programmed. PCM cells have been fabricated as small as $4F^2$ [16] (F as a manufacturing process minimum feature

[2]Disregarding second order sneak paths and assuming ideal transistors in the input inverters. This problem actually gets worse when they have high effective resistance, so in larger memristor crossbars the input inverters' drive strength (transistor width) must be increased.

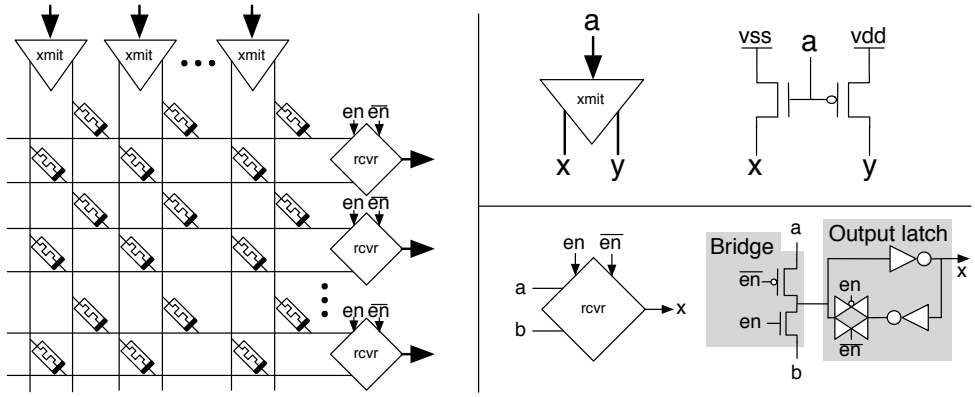

Fig. 4: A memristor-based crossbar. The design is shown on the left, transmission module symbol and circuit on the upper right, and receiver module symbol and circuit on the lower right.

size) and are predicted to stay around that size [17], making them somewhat smaller than the M1 wire pitch. Next is their resistance: this paper models nearly linear I-V curves based on results from Lavizzari *et al.* [18]. They report HRS/LRS ratios of about 350 – high enough that one can build large crossbars, though small enough that leakage is a significant concern. Finally, programming: PCM cells are programmed by heating, which is achieved by raising their voltage above a certain level, which varies a bit from cell to cell. Lavizzari *et al.* report, however, that if voltages are kept below 0.77V, cell failure rates will be about one in 10^9 over 10^4 seconds. In order to reduce this failure rate further, the design ensures that PCM cells are never exposed to voltages above 0.6V (even transient spikes) during normal operation.

b) Programming the Memristors: Thus far this paper has discussed only the normal operation of the proposed crossbar, but have not discussed how its memristors can be programmed. It will not be discussed in depth here as existing work already discusses it in detail [19], [18], [20], including in passive PCM arrays without selection devices (like ours) [20]. In short, programming the cells involves some selection logic and programming pulse circuitry which are external to the array. These signals can then be selectively broadcast into the array. As a result, the programming circuitry can be amortized over the entire array or even over many arrays, like multiple bit-slices of a wide crossbar.

While multiplexing programming signals slows down programming considerably, in reconfigurable applications the routing patterns are changed infrequently – on the granularity of hours or more is typical for FPGAs – so programming speed is not an issue. However, adding programming capability does involve adding a programming transistor to each row and column in the array, which adds capacitance thus affecting normal operation performance. These programming transistors are modeled in the simulations.

III. Evaluation

In this section we quantitatively measure the advantages of memristive crossbars. We evaluate memristive crossbars relative to CMOS pass gate crossbars. In order to compare functionally equivalent devices, we add output latches to the CMOS crossbar shown in Figure 2. While these latches are unnecessary overhead in some situations, in the context of pipelined networks they are likely to occur after the crossbar anyway. We determine the power dissipation and area for each type of crossbar over a range of sizes, 2x2 up to 32x32. As stated earlier, crossbars are generally kept small, so 32x32 crossbars are unusual. However, if memristors successfully mitigate the overheads in crossbars of this size, it is likely they will have interesting applications.

A. Experimental Methodology for Power and Latency

To evaluate the power dissipation of the crossbars, we build Spice models for each crossbar. We use 32nm high performance transistor models from the predictive technology model [21] and Ngspice [22] for simulation. Since PCM does not have a standard Spice model, we model them as simple resistors with nonlinear I-V curves from [18]. To reduce power and area, all nMOS transistors are minimum width (pMOS transistors are sized to match drive strength) except for power gating transistors.

Since the wires in a full crossbar could become relatively large, we model some of the wire parasitics of the wires in the PCM and CMOS wire arrays. Wire parasitics in other crossbar structures (like latches and the clock tree) are neglected as they are assumed to be short wires. In the PCM wire array we use parasitic values given by PTM [21] to model series resistance, ground capacitance, and coupling capacitance with adjacent wires in the same layer. (It should be noted that the values given by PTM match the parasitics modeled by DSENT [1], which we use to model area and other circuits in subsequent sections.) In the CMOS array we model the same parasitics though coupling capacitance is neglected since array wires are spaced much further apart – the PCM array wires have a minimum M1 pitch of 110nm whereas the CMOS array wire pitch is limited by the pass gate/latch combination at each crosspoint. Wire parasitic inductance is not modeled as it hindered Ngspice convergence and we expect its effects to be minor.

All of the crossbars are clocked at 2GHz and inputs are randomly generated. We simulate them for 50 cycles at 2ps timesteps. To ensure that the crossbars operate properly, we monitor voltages across each memristor to ensure they are not read disturbed during operation. We also monitor the outputs to make sure they are correct. Finally, in order to make sure that the memristor designs do not suffer from HRS/LRS ratio

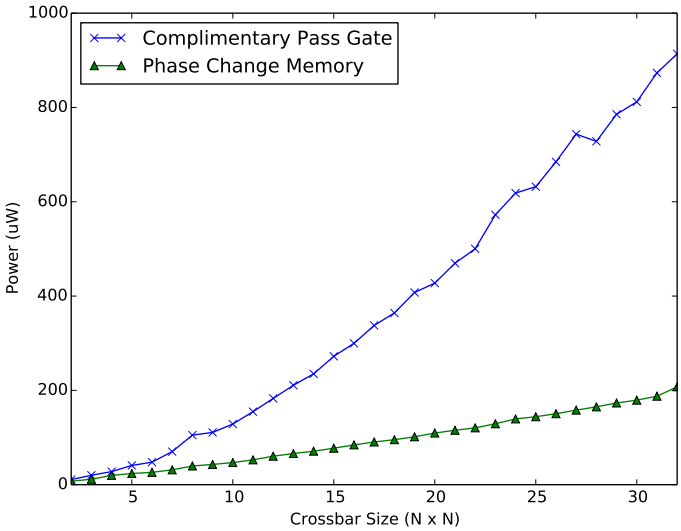

Fig. 5: Comparison CMOS and PCM crossbar power usage as NxN crossbars increase in radix (N).

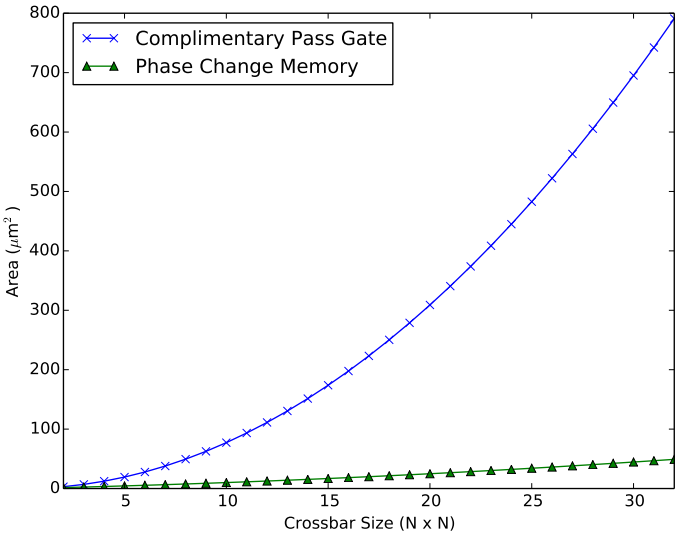

Fig. 6: Comparison of estimated area requirements for different crossbar types. While both crossbars exhibit quadratic scaling, the coefficient for the PCM crossbar is radically lower since the PCM cells are tiny. As a result, they use more than an order of magnitude less area than CMOS crossbars at high radices.

correctness issues, we test the two corner case inputs of only a single input set to 1 and a single input set to 0. All of the designs evaluated presented here passed all tests.

B. Experimental Methodology for Area

To estimate area, we use DSENT [1]. DSENT has energy, static power, and area models for common NoC components and several standard cells (like inverters, latches, and some gates) which can be composed to obtain estimates for larger systems.

To model the area for crossbars, we simply use the various standard cells in DSENT to build the designs and sum the total area. DSENT, however, cannot model the area of the PCM array. These areas, however, are easy to calculate since we know the minimum pitch in these arrays from previous work (88nm for PCM [19]) is smaller than the M1 pitch modeled by DSENT in 32nm (110nm). As a result, each of the two PCM arrays are square with a dimension of the radix multiplied by the minimum M1 pitch.

a) Results: The results of the power simulations can be found in Figure 5 and they more or less match expectations. For every size, memristive crossbars are superior to their CMOS counterpart in area and power. At relatively small sizes, however, they do not offer much benefit. This is because for these small sizes, power is dominated by the input and output circuitry, which is very similar for both CMOS and memristors. However, as the area and number of components in the of the array grows quadratically with size, the CMOS crossbar rapidly grows beyond ideal, linear scaling. As a result of this scaling, the power reduction resulting from PCM ranges from only 1.4x in a small 4x4 crossbar to 4.4x in a 32x32 crossbar.

The PCM crossbar also increases power dissipation very slightly super-linearly. The reason for this is PCM's relatively low HRS/LRS ratio. As the crossbar array increases, so does the sneak path current as more and more high-resistance state memristors leak in parallel.

Estimates of area for the crossbars are shown in Figure 6. All designs scale quadratically, however due to memristors' small size, their quadratic coefficient is far smaller than on the CMOS crossbar. Although these area estimates are based on standard cells instead of custom layout, it appears likely that memristors' area advantage is significant. Here the scaling trends are quantitatively clearer with the area improvement in a 4x4 crossbar only 3.6x better compared to a 16x improvement in a 32x32 crossbar.

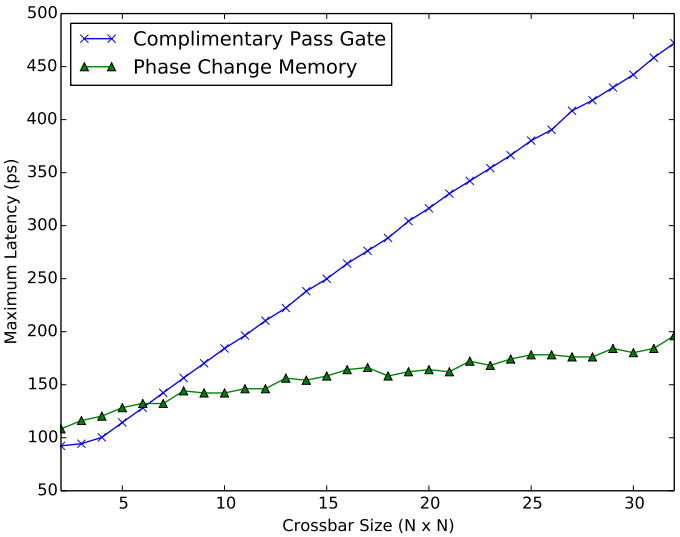

Fig. 7: Comparison of maximum latency through PCM and Pass Gate crossbar types.

Finally, Figure 7 shows the maximum observed latency through each crossbar. As a result of reduced size and thus capacitance in the array, the PCM crossbar scales much more gracefully than the CMOS crossbar, which barely operates at a 2GHz clock rate in large radices compared to less than 200ps latency in the PCM equivalent.

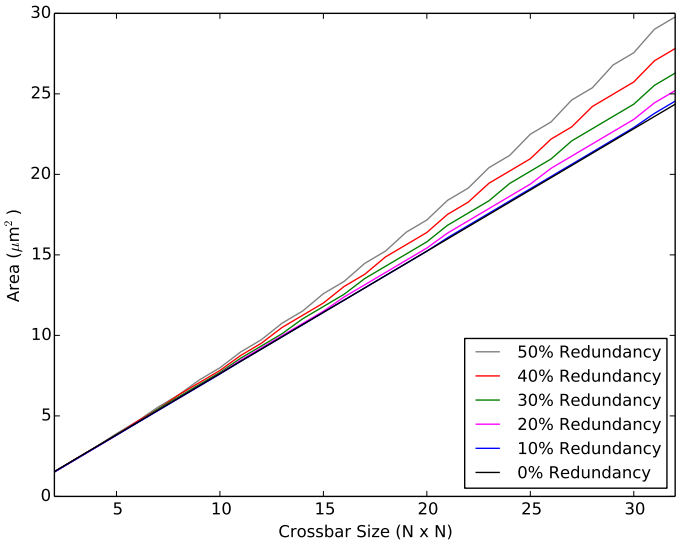

Fig. 8: In order to deal with hard faults and wear out in the memristor arrays, bypass rows and columns can be added. This redundancy adds area overhead as shown in this figure of PCM crossbar area. Even with these overheads, the PCM crossbars remain far smaller then a pass gate crossbar for which a 32 radix occupies nearly 800μm^2.

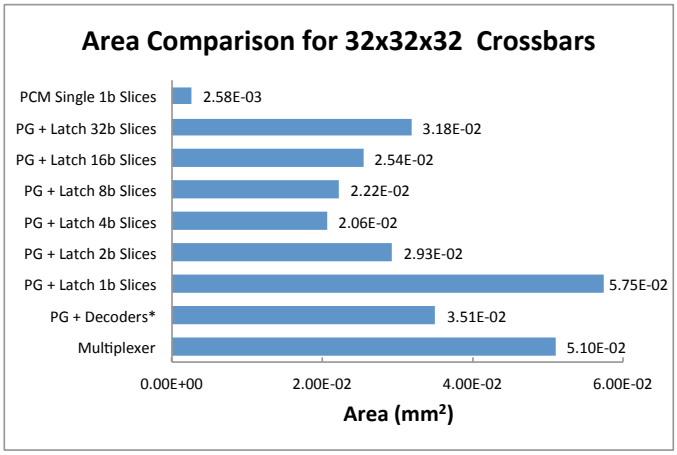

Fig. 9: In a 32-bit, 32-radix crossbar, alternative CMOS designs use less area than our pass gate + latch per crosspoint design unless we route multiple bits in each array, amortizing the latch overhead. Regardless of these optimizations, however, the PCM crossbar uses significantly less area than any CMOS design. *Note: the pass gate + decoder design area estimate does not include the overhead of routing control wires into the pass gate arrays. This overhead may be significant.

C. Wear Out and Hard Fault Tolerance

As in CMOS transistor fabrication, some of the memristors in an array may not operate correctly – some may be stuck in a high resistance state and some in a low resistance state. There are two interesting ways around the problem. First, the application may be able to change signal routing to simply not use output associated with the broken memristor. For instance, in a circuit switched network application, whatever is allocating wires can simply choose not to allocate the wire to which that memristor is connected. Second, shortly after fabrication, broken LRS memristors can be "blown out" by applying an over-current to their junction. Then, bypass connections (several extra rows and columns in the array) can be used instead of the broken memristors. Figure 8 shows the area overhead which they would incur, which is relatively minor with 21% overhead for 50% redundancy in a 32x32 crossbar. The reason for this relatively smaller overhead is that the PCM arrays themselves are very small – area overhead is dominated by input and output circuitry – so adding redundancy to them does not increase the overall size.

Like Flash, memristors have wear out issues. ITRS projects that PCM will be able to withstand 10^9 writes. If the crossbars are reprogrammed at the granularity of a program context swap (every 10ms), PCM crossbars are likely to begin breaking in under a year, though if they are reprogrammed only every second, they should last as long as the rest of the chip. Future memristors like Redox memory are projected to support 10^{16} writes [17], so they will not have wearing issues at any reasonable reconfiguration interval. Additionally, the fault tolerance scheme described above could also be used to extend array lifetimes.

Even without technology or microarchitectural solutions to wear out, many applications could support limited reconfiguration frequencies. FPGAs, for example, are rarely reconfigured more often than once an hour and often times programmed only several times over their entire lifetime. Similarly, accelerators often run a single workload at a time and do not support context swapping. Although context swapping may be desired, the relatively large architectural state accelerators contain often make rapid context switching difficult.

D. Alternative CMOS Designs

The CMOS crossbar design we detail and evaluate in this section is only one of several CMOS designs possible. While it turns out to be a relatively optimized baseline in terms of power for our proposed application, alternatives exist and are sometimes superior. This subsection explores and compares several alternatives.

a) Multi-bit Crossbars: Although simple, the design requires a pass gate and latch at each crosspoint, which is area wasteful in a multi-bit crossbar since the routing information in those latches is redundant. To evaluate this potential waste, we summarize and evaluate several alternative designs: two other popular designs and one variant of our design: (1) When synthesizing from RTL, a multiplexer-based crossbar often results. This crossbar uses an N-to-1 crossbar for each of N outputs, so it too scales quadratically with radix. (2) Instead of keeping routing information at a latch in each cross point, a decoder can be used for each output and control signals routed to each pass gate. This design, however, has significant wire routing overheads. (3) While we show the design of a single bit slice crossbar, multiple bits could be routed in the crossbar. This design amortizes the overhead of the latch in each cross point, but the increased bit width contributes quadratically to the area of the crossbar, presenting a trade-off.

Area estimates for the various crossbar designs are shown in Figure 9. As one would intuitively expect, putting a latch at each crosspoint is indeed wasteful. If multiple bits are routed through each array, however, this area is quickly amortized.

We find that 4-bit slices best amortize the latch overhead. Beyond 4-bit slices, the array's quadratic scaling overtakes latch amortization benefits and the overall array size increases. The 4-bit slice configuration, in fact, is the smallest CMOS design we have examined. Even with this optimization for multi-bit crossbars, however, the PCM crossbar remains over 12x smaller.

b) Functionality Comparison: While the multiple bit pass gate + latch crossbar is smaller than other CMOS designs, it achieves this area at the expense of functionality. While the other two CMOS crossbars can be reconfigured (change their routing) every cycle, the crossbars in this paper cannot. The reasons are that it lacks a decoder and its latches must be set via a scan chain. As a result, the routing bit pattern needs to be computed by software and shifted into the latches, so reconfiguration takes too long to support dynamic routing or even time-domain multiplexing. Removing this functionality, however, is so effective in area reduction that this crossbar is even smaller than a naïvely scaled Swizzle-Switch [23], a state-of-the-art crossbar. Additionally, for the applications we are discussing in this paper, dynamic routing is not generally used so this functionality trade off is appropriate and thus an aggressive baseline.

E. Conclusions

As we have shown in this circuit study, memristive crossbars have significant potential to help build efficient, physically small routing devices, especially ones with high radices. The resulting small size and power efficiency yields routing devices which, compared to the size of typical NoC interconnect wires, are practically free, enabling larger amounts of reconfigurability in future accelerator systems.

IV. Threats to Validity

Whenever any new technology is studied it is difficult or impossible to fully and accurately model all aspects of the technology. Even ITRS roadmap projections for well-known transistor devices have been consistently off. Despite these inevitable inaccuracies in modeling, examining emerging (or future) technologies for interesting applications is important to guide implementation of the technology. Since memristor technology is at an early stage of development, device researchers have only considered traditional memory based applications for memristors. This paper motivates another use case (interconnects) which may lead to further work from device experts. For example, PCM programming current and HRS/LRS can be traded-off. Since memristive crossbars would not reconfigured often and a higher HRS/LRS ratio is beneficial, interconnect PCM devices could be built at a different point in the trade-off space.

This study estimates technology parameters based on published devices designed for storage, and build first and second order models. We list some potential caveats to this paper's models:

• This papers does not completely model programming circuitry – only the programming transistors directly wired to the arrays since their capacitance affects the energy for each bit transmission. We make the assumption that previous work regarding programming of the memristors [20] can be applied to the memristor models being used.

• Memristors are modeled as non-linear I-V curves in SPICE. If the memristors have other non-ideal effects on the circuit (like parasitic capacitance), they are not modeled here. Unfortunately, published work does not include numbers about any parasitic capacitance which memristors may add. However, if they do add capacitance, it is likely to be exceedingly small amounts since the memristors are themselves exceedingly small.

• This study assumes that it is possible to add an enable signal (a low duty-cycle clock) which can be generated at sub-cycle times. Although it models the distribution of the enable signal (with several clock tree structures, which consume a large amount of the power in both crossbar designs), this paper assumes that the overhead in generating the enable signals is relatively small and amortized over a large number of units.

• The area estimates are primarily summations of standard cell sizes from DSENT plus wiring area in cases where the paper describes the wire layout. Place and route and/or custom layout will produce different results.

In general, this papers attempts to err on the side of an overly optimistic baseline and create conservative models for our proposed technology, memristive networks. We have not laid out and optimized for all of the technologies involved (as the amount of labor involved would make this study infeasible). As such, not all of the comparisons this papers makes are completely fair. Given that our results indicate order-of-magnitude differences, however, we think this study's modeling inaccuracies are sufficiently small so as not to erode its conclusions.

V. Related Work

Section II discussed background on memristors, in particular PCM. In addition to previous work on the devices, there is also some work related to the non-memory use of memristors.

a) Memristors in non-memory applications: As a new device technology with many interesting potential application, memristors are becoming a popular subject of study for non-storage applications. A range of papers examine memristors to build LUTs, multiplexers, and switchboxes [24], [25], [26], [27]. This paper confirms previous results for multiplexers and switchboxes and applies also to higher radix topologies. Guo *et al.* [28] have also proposed a memristor based TCAM to accelerate associative search applications. The same group also also proposed "resistive computing" to use resistive memory for general computation, specifically using STT-RAM [29]. Finally, memristors are highly anticipated for use in neuromorphic computing [30], [31], [32].

VI. Conclusion

This paper examined communication using a new class of devices: memristors. We found that they have the potential to significantly reduce the area and power overheads in interconnects and we propose a crossbar design which yields those benefits. In particular, the interconnect proposed in this

paper opens the possibility of designing networks of fine-grained microaccelerators instead of coarser designs, allowing a broader range of applications to take advantage of hardware acceleration. Our analysis shows that memristive interconnects can yield over an order of magnitude improvement in area, several factors in power, and maintains reasonable latency during radix scaling. When a technology delivers improvements on that scale, extensive shifts may be imminent. The use of memristors for communication has the possibility to shift the way systems are designed.

REFERENCES

[1] C. Sun, C.-H. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*. IEEE, 2012, pp. 201–210.

[2] V. Govindaraju, C.-H. Ho, and K. Sankaralingam, "Dynamically Specialized Datapaths for energy efficient computing," *HPCA*, pp. 503–514, Feb. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5749755

[3] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*. IEEE, 2009, pp. 469–480.

[4] L. Wu, A. Lottarini, T. K. Paine, M. A. Kim, and K. A. Ross, "Q100: the architecture and design of a database processing unit," in *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*. ACM, 2014, pp. 255–268.

[5] S. Galal and M. Horowitz, "Energy-efficient floating-point unit design," *Computers, IEEE Transactions on*, vol. 60, no. 7, pp. 913–922, 2011.

[6] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz, "Understanding sources of inefficiency in general-purpose chips," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3. ACM, 2010, pp. 37–47.

[7] K. Lim, B. Falsafi, J. Picorel, B. Grot, P. Ranganathan, O. Kocberber *et al.*, "Meet the walkers: Accelerating index traversals for in-memory databases," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, no. EPFL-CONF-190306, 2013.

[8] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor, "Conservation cores: reducing the energy of mature computations," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 1. ACM, 2010, pp. 205–218.

[9] J. V. Lunteren, C. Hagleitner, T. Heil, G. Biran, U. Shvadron, and K. Atasu, "Designing a programmable wire-speed regular-expression matching accelerator," in *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*. IEEE, 2012, pp. 461–472.

[10] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '14. New York, NY, USA: ACM, 2014, pp. 269–284. [Online]. Available: http://doi.acm.org/10.1145/2541940.2541967

[11] S. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S. Hsu, H. Kaul, M. Anders, and R. Krishnamurthy, "53 gbps native $\mathrm{gf}(2^4)^2$ composite-field aes-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 4, pp. 767–776, April 2011.

[12] L. Chua, "Resistance switching memories are memristors," *Applied Physics A*, vol. 102, no. 4, pp. 765–783, 2011.

[13] Micron. Micron announces availability of phase change memory for mobile devices. [Online]. Available: http://investors.micron.com/releasedetail.cfm?ReleaseID=692563

[14] Samsung. Samsung ships industrys first multi-chip package with a pram chip for handsets.

[15] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase Change Memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5609179

[16] Y. Sasago, M. Kinoshita, T. Morikawa, K. Kurotsuchi, S. Hanzawa, T. Mine, A. Shima, Y. Fujisaki, H. Kume, H. Moriya *et al.*, "Cross-point phase change memory with 4f2 cell size driven by low-contact-resistivity poly-si diode," in *VLSI Technology, 2009 Symposium on*. IEEE, 2009, pp. 24–25.

[17] "International technology roadmap for semiconductors," 2011.

[18] S. Lavizzari, D. Sharma, and D. Ielmini, "Threshold-switching delay controlled by 1/f current fluctuations in phase-change memory devices," *Electron Devices, IEEE Transactions on*, vol. 57, no. 5, pp. 1047–1054, 2010.

[19] G. De Sandre, L. Bettini, E. Calvetti, G. Giacomi, M. Pasotti, M. Borghi, P. Zuliani, R. Annunziata, I. Tortorelli, F. Pellizzer, and R. Bez, "Program circuit for a phase change memory array with 2 mb/s write throughput for embedded applications," in *Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European*, 2008, pp. 198–201.

[20] J. Liang and H.-S. Wong, "Cross-point memory array without cell selectors: Device characteristics and data storage pattern dependencies," *Electron Devices, IEEE Transactions on*, vol. 57, no. 10, pp. 2531–2538, 2010.

[21] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," in *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, 2006, pp. 6–590. [Online]. Available: http://ptm.asu.edu

[22] [Online]. Available: http://ngspice.sourceforge.net/

[23] K. Sewell, R. G. Dreslinski, T. Manville, S. Satpathy, N. Pinckney, G. Blake, M. Cieslak, R. Das, T. F. Wenisch, D. Sylvester *et al.*, "Swizzle-switch networks for many-core systems," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 2, no. 2, pp. 278–294, 2012.

[24] C. Wen, J. Li, S. Kim, M. Brietwisch, C. Lam, J. Paramesh, and L. T. Pileggi, "A non-volatile look-up table design using PCM (phase-change memory) cells," in *VLSI Circuits*, 2011, pp. 302–303. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5986431

[25] J. Cong and B. Xiao, "mrfpga: A novel fpga architecture with memristor-based reconfiguration," in *Nanoscale Architectures (NANOARCH), 2011 IEEE/ACM International Symposium on*. IEEE, 2011, pp. 1–8.

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

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

[28] Q. Guo, X. Guo, Y. Bai, and E. Ipek, "A resistive tcam accelerator for data-intensive computing," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2011, pp. 339–350.

[29] X. Guo, E. Ipek, and T. Soyata, "Resistive computation: avoiding the power wall with low-leakage, stt-mram based computing," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3. ACM, 2010, pp. 371–382.

[30] D. Kudithipudi and C. E. Merkel, "Reconfigurable memristor fabrics for heterogeneous computing," in *Advances in Neuromorphic Memristor Science and Applications*. Springer, 2012, pp. 89–106.

[31] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, and T. Prodromakis, "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *arXiv preprint arXiv:1302.7007*, 2013.

[32] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.