Memristors for Neural Branch Prediction: A Case Study in Strict Latency and Write Endurance Challenges

Hebatallah Saadeldeen†, Diana Franklin†, Guoping Long‡, Charlotte Hill†, Aisha Browne†,
Dmitri Strukov§, Timothy Sherwood†, Frederic T. Chong †
† Department of Computer Science, UC Santa Barbara
§ Electrical and Computer Engineering, UC Santa Barbara
‡ Institute of Software, Chinese Academy of Sciences
{heba,franklin,charlottehill,sherwood,chong}@cs.ucsb.edu, aisha_browne@umail.ucsb.edu,
strukov@ece.ucsb.edu, guopinglong@gmail.com

ABSTRACT

Memristors offer many potential advantages over more traditional memory-cell technologies, including the potential for extreme densities, and fast read times. Current devices, however, are plagued by problems of yield, and durability. We present a limit study of an aggressive neural network application that has a high update rate and a strict latency requirement, analog neural branch predictor. Of course, traditional analog neural network (ANN) implementations of branch predictors are not built with the idea that the underlying bits are likely to fail due to both manufacturing and wear-out issues. Without some careful precautions, a direct one-to-one replacement will result in poor behavior.

We propose a hybrid system that uses SRAM front-end cache, and a distributed-sum scheme to overcome memristors' limitations. Our design can leverage devices with even modest durability (surviving only hours of continuous switching) to provide a system lasting 5 or more years of continuous operation. In addition, these schemes allow for a fault-tolerant design as well. We find that, while a neural predictor benefits from larger density, current technology parameters do not allow high dense, energy-efficient design. Thus, we discuss a range of plausible memristor characteristics that would; as the technology advances; make them practical for our application.

Categories and Subject Descriptors

B.3.0 [**Memory Structures**]: Memristors; B.3.4 [**Reliability, Testing, and Fault-Tolerance**]: Miscellaneous

General Terms

Reliability

CF'13, May 14–16, 2013, Ischia, Italy.

Keywords

Neural networks, branch prediction, analog computations, Reliability

1. INTRODUCTION

As concerns mount about the end of DRAM scaling, researchers are beginning to examine alternative memory technologies and the new applications they enable. Resistive devices, based on phase-change transitions (e.g. PCM) and transition metal oxide (e.g. memristors), could potentially replace typical memories like Flash, DRAM and perhaps SRAM. While PCM is the closest to commercialization today (128Mb parts that are currently in production), memristors are moving forward quickly and offer a slightly different set of tradeoffs. In fact, HP announced October last year that memristor commercial devices that replace Flash will be out within 18 months. In addition, a 32Gb memristor crossbar array has been demonstrated [18].

Among the most important advantages of memristors are CMOS-compatibility [40], analog properties [5], and small footprint due to small lateral dimensions and possibility of monolithical 3D stacking [17][31]. These properties enable a wide range of potential applications ranging from memory to computations, from digital circuits to analog circuits, and including nonvolatile memory [13], signal processing circuits [32] and artificial neuromorphic networks [17].

Specifically, memristors and neural applications are always thought to be a perfect match. In this paper, we investigate this assumption given the recent milestones in the technology with the goal of guiding future technology advancements in order to be valuable for real applications.

We choose an analog neural branch predictor for our study. While neural branch predictors have been extensively studied and there exist efficient implementation in terms of power and latency [29], we choose this application to serve as a limit study for memristor technology due to its strict latency requirement and frequent updates. To our knowledge, we are the first to address durability issues assuming SRAM replacement.

The strong affinity between memristors and neural networks is generally based upon the assumption that memristors would be used to create an analog network of neurons. Unfortunately, this purely analog implementation is impractical given the serious issues that currently remain with the technology, such as low write endurance. We find that a hybrid digital-analog approach is needed in order to enable

cache-based write coalescing. Specifically, we choose a table-based perceptron predictor implementation that efficiently permits hybrid system design and exploits memristors' analog properties for prediction computations (expensive dot product). We introduce an SRAM front-end cache to reduce writes to memristor storage. We also propose a distributed sum scheme to extend lifetime of our predictor and make it fault tolerant. Our techniques are general, however, they are specifically suitable for latency sensitive applications. While these techniques succeed to extend predictors' lifetime to more than 5 years, they assume aggressive assumptions about the technology which are not readily available in today's technology. However, these assumptions are nevertheless reasonable and based on the understanding of physics behind memristors' operation, e.g. scaling crossbar wire size and pitch, 3D stacking, and multi-bit storage. Thus, we provide guidance as of how much the memristors' technology should be advanced before it becomes practical for our application.

In summary, this paper makes the following contributions:

1. We propose general techniques to overcome write endurance limits, and provide fault tolerance that are necessary for memristors in order to be valuable for real applications. Our techniques are specifically valuable for latency strict applications.

2. We propose a limit study of Memristor-Neural Analog Predictor (MNAP) that exploits memristors' high density and ability to do analog computations. Our predictor is highly accurate as compared to state of the art neural predictors.

3. We provide guidelines for future technology advancement to be valuable for our application.

The remainder of this paper is organized as follows. In section 2, we provide background on memristor devices. In section 3, we describe ANN-based branch predictors. In section 4, we describe our techniques to mitigate memristors' challenges. In section 5, we provide discussions on memristors' parameters and guideline for future technology advancements. In section 6, we describe our simulation environment. In section 7, we show our accuracy results and evaluate the effectiveness of our proposed techniques to overcome memristors' challenges. In section 8, we present related work. Finally, we conclude the paper in section 9.

2. RESISTIVE MEMORY

In this section we provide brief background on memristors, their key advantages and major challenges.

2.1 Memristors

Figure 1-a shows the simplest two-terminal memristive device structure consisting of two metal electrodes with metal oxide memristive layer sandwiched in between [43]. The devices can adopt either high or low resistance states, which can be considered bits. A positive voltage above a specific threshold ($V > V_{ON}$) will set the memristor to high resistance state. A negative voltage of the same magnitude ($V < V_{OFF}$) toggles it to low-resistance state. Alternatively, by controlling write pulse magnitude and/or duration the memristive device can be switched continuously into intermediate states. Figure 1-a shows typical I-V for bipolar type of the devices.

In general, there are various materials which can implement such functionality [17]. On one hand, the metal-oxide devices and the solid state electrolytes are more attractive because of the combination of high ON/OFF ratio [43], scaling prospects, fast write speed [36], high retention [33], and compatibility with CMOS [40]. On the other hand, ones based on phase-change transitions have intrinsically longer "on" switching time and, larger write energy [2]. In this study, we assume metal oxide devices.

2.2 Key Advantages

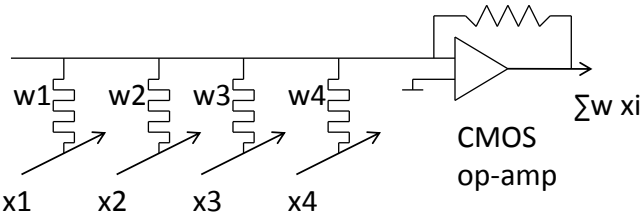


Figure 3: Dot Product circuit

Density One of the advantages of memristive devices is very attractive scaling prospects. The best way to sustain the density of a single memristive device is to integrate them into crossbar structures, based on mutually perpendicular layers of parallel wires (Fig. 1-b). Such structures can be used to implement passive crossbar memories, in which the crosspoint memristors are used as memory cells, while the semiconductor transistor subsystem performs all the peripheral circuitry [30].

Analog properties Dot product computation circuit, which is a bottleneck operation for artificial neural networks in general, is recently demonstrated with hybrid CMOS/ memristor circuits (Fig. 3) with weights of up to 7-bit precision [5]. In such circuit, analog inputs x are multiplied by the weight value w of the corresponding memristive devices and passed to the horizontal wire connected to the virtual ground input of the CMOS-based operational amplifier. As a result, the current injected through all memristive devices on the common wire is equal to the dot product w * x and can be sensed at the output of operational amplifier.

In general, the read time for such circuit is decreased by lowering the ON state resistance for the memristive devices. However, the smallest ON value is mainly limited by the crossbar wire resistance. Therefore the read latency increases for larger crossbar arrays. However, read delay below 1ns is feasible for relatively small ($\leq$ 1000 wide) crossbar arrays [30] with current sensing scheme [41] which is the case in our predictor (for a single perceptron table).

These interesting properties make memristive devices fit a wide range of applications [20][34]. In digital domain, applications could involve non-volatile solid-state memory, and programmable logic. In analog domain, it could be used for neuromorphic circuits, and analog signal processing.

2.3 Challenges

Write Endurance values reported in the literature have been low [17], however it seems that their limits is still far from being reached and depends on choosing the right switching materials [42]. In fact, in 2011, [16] demonstrated TaO_x-based asymmetric passive switching device that results in cycling endurance of $> 10^{12}$. ITRS [2] projected a write cycle endurance of 10^{16} by 2024.

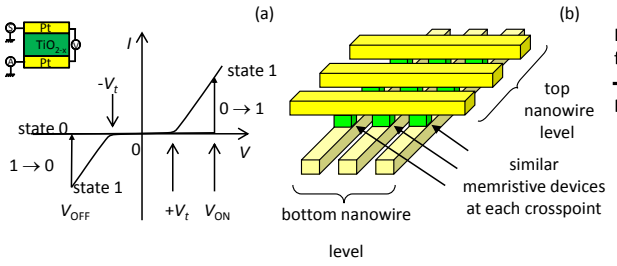


Figure 1: Crossbar memory: (a) I-V curve of a single memristor (schematically), (b) crossbar array structure

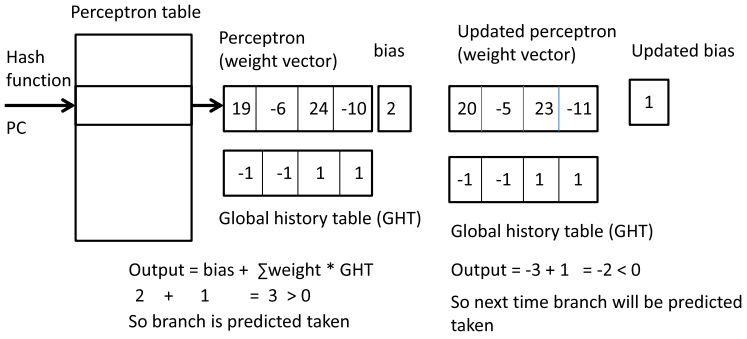


Figure 2: Neural predictor prediction and training

Defect Rates need to be improved before devices could be used in a commercial product. In 2007, [6] showed that about 50% of the bits are defective. In 2009, [14] demonstrated that 8% of the bits are defective (i.e. yield of 92%).

Other limitations for memristors include write latency, and sneak current. Sneak current could be avoided by either integrating diodes with memristors [38], or by special techniques that takes advantages of the non linearity of the devices [42].

3. NEURAL BRANCH PREDICTOR

Most neural branch predictors are derived from the perceptron branch predictor [10][12]. In that model, the perceptron is a vector of h (history length of the predictor) weights. A *perceptron table* of N entries is stored in fast memory. Each perceptron entry represents the correlation among past branches and all the branches that are mapped to that entry. The magnitude of each weight specifies the strength of the correlation with the h most recent branches. There is also a bias weight table which captures the tendency of the branch towards being taken or not taken independent of branch history. A global history table with the h most recent branch outcomes is also kept. Figure 2 shows typical operations of a neural predictor. While latency and power have been a problem for neural predictors implementations, recent research provides efficient implementation overcoming these problems without sacrificing accuracy [29].

Density Advantages Figure 4 demonstrates density advantages on predictor accuracy; MPKI (misprediction per kilo instructions); over two of most accurate up to date table based (L-TAGE) [26], and neural based predictor (SNAP) [29] for SPEC traces. While increasing table size leads to significant improvement in prediction accuracy, there exist a sweet spot (20x) where further increase in density has negligible effect on accuracy. The reason of improved accuracy is reducing the *aliasing* effect where different branches destructively share the same hardware resources. In this work, we choose 20x as our exploration point.

4. MITIGATING CHALLENGES

In this section, we propose architecture solutions that address memristors' challenges while taking advantage of their high density and ability to do analog computations.

4.1 Write Endurance

While wear-leveling has been studied for resistive memories, specifically PCM [22][7], it was in the context of hybrid main memory. However, our application is different in terms of latency requirements (replacement of SRAM), frequency

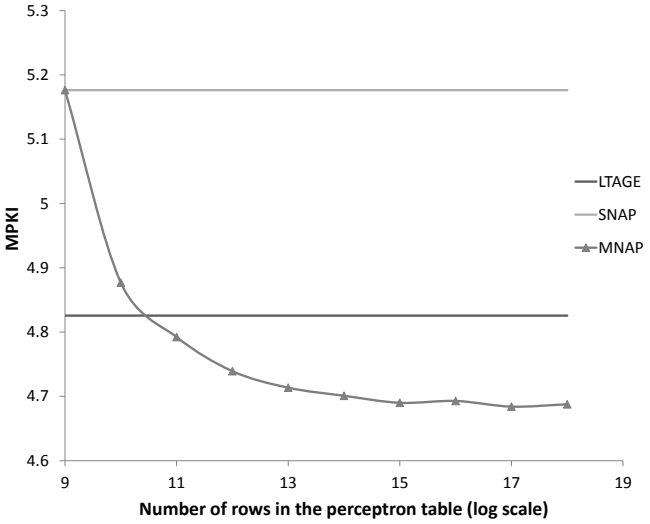


Figure 4: MPKI for SPEC CPU as the size of the perceptron table increase from 2^9 rows to 2^{18} rows. 2^9 rows is the size of table in SNAP predictor

of reads (prediction almost every cycle and thus making any remapping technique that requires a couple of cycles unsuitable), and frequency of updates (< microsec). This makes straight forward application of previously proposed techniques unsuitable. Next, we will discuss our proposed techniques.

4.1.1 Hybrid Perceptron Table

An SRAM cache is used to keep the most recently *updated* table entries. Since the perceptron table is subdivided into several predictor tables that are indexed differently [25], a separate 4-way set associative cache is used for each predictor table.

This affects both reads and writes to the memristor perceptron table. On a read, the SRAM cache and memristors are read in parallel. In case of a cache hit, cache value is used o.w. memristor value is used. On writes, if the entry is in the cache, it is written there. If not, the entry is first moved to the cache, and then written. This may evict a cache line, necessitating a write to the memristors. This scheme reduces traffic to memristor storage by 75% for our application.

4.1.2 Distributed Sum

Many instances of memristor are used to store a single weight value. When an update occurs for this value, only

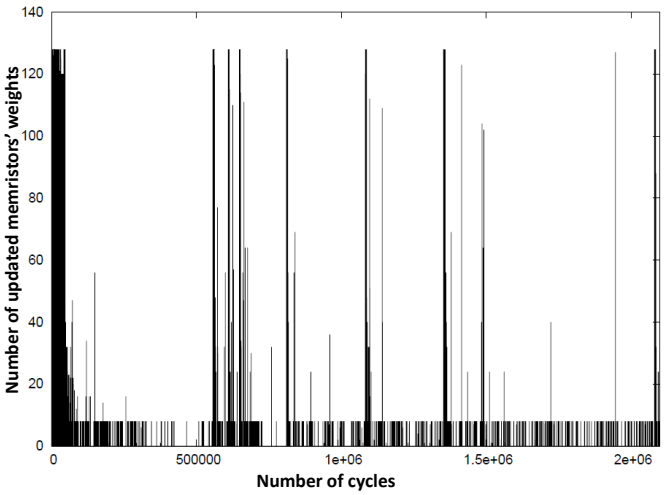


Figure 6: Number of weights evicted from SRAM to memristors. On average only 8 weights need to be updated

one instance is updated, increasing the lifetime of the value. These instances are organized such that the final weight value is the sum of values across all instances. Thus, assuming N writes to a weight, R instances and writes are evenly distributed among all instances, N/R writes is performed to each instance of the weight.

On memristor read, the instances values are added before they are used. In order to store a value, first an instance to modify is chosen. The old value is read; which is the sum of all instance values masking the entry of the instance to be modified. The delta of the old and the new value is calculated, and then written to the candidate instance which is chosen using global round- robin scheme (GRR). GRR uses only one counter for the entire table. Figure 5 shows how this algorithm is applied to our neural predictor assuming three instances.

The main advantage of our scheme is that it exploits memristors' unique analog properties that make predictions within a 1ns feasible. Assuming values of a weight that belongs to different instances are physically stored in the same memristor array, our scheme reads each memory array in parallel (using scheme described in section 2.2), the final current is steered to a positive or negative line based on the XOR of the sign bit for that weight and the appropriate history bit [29]. Finally, the current is sensed at the output of operational amplifier.

However, our scheme uses large storage overhead (requires replication of the whole perceptron table) to avoid remapping (requires couple of cycles). Assuming a near term 100x more dense memristors' storage is achievable; memristors provide more density than required for performance. Thus, sacrificing storage for prediction latency is important. In this paper we do not address the circuit design in detail; however, we discuss some of circuit issues in section 4.4.

4.2 Write Latency

Memristor write times reported in the literature vary significantly from hundreds of pico-seconds [36] to tens of nanoseconds [41]. Since high write latency would affect read throughput, we use a write buffer where data evicted from SRAM cache are kept. In case buffer is full, data will be dropped. In addition, data from the write buffer will only be written to memristor- perceptron table when processor stalls. Figure 6 shows the number of weights that needs to be written to the memristor- perceptron table for hmmer trace. On average only 8 weights (size of a predictor table entry; where perceptron table is divided into 16 predictor tables with 8 weights each [29]) needs to be written suggesting that a write buffer of 8 entries per table is enough.

4.3 High Defect Rates

In order to tolerate memristors' high defect rates, we propose slight changes to our distributed sum scheme. We model defect rates as in [6].

We assume that initially we have a defective weight map that stores a single bit per weight. This bit is set to 1 if the weight has at least one defective bit. This extra bit is stored with every weight in memristor storage. Then we slightly modify the round robin scheme used to choose the instance that needs to be modified as follows. For each weight, we start with the instance chosen by the GRR. If it is defective, we use the next one that is not defective. If all are defective, then we use the one indicated by the GRR. Because this is only on the critical paths of writes, not reads, we are not concerned with the slight additional delay. While this maintains accuracy, it does put pressure on write endurance, since the writes from a defective instance are all placed on the next non-defective instance.

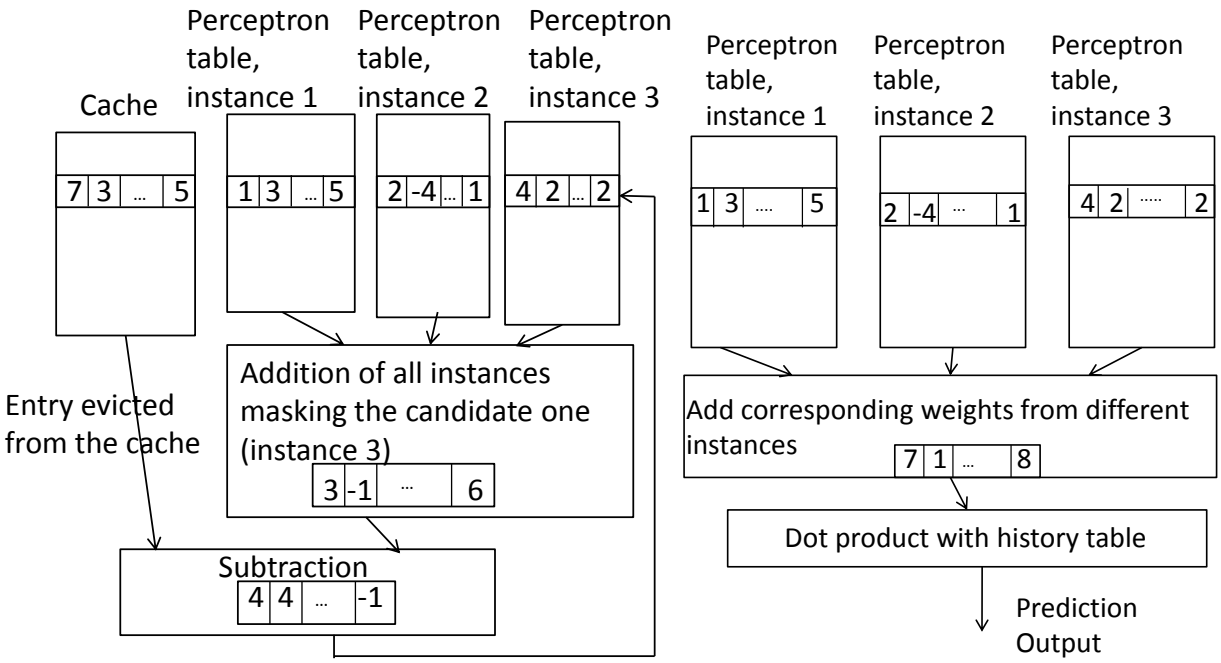
4.4 Circuit Discussion

Since both SRAM and memristors stores data in different forms, additional circuitries are required to move data between them. In order to move data from memristors to SRAM, analog to digital converters (ADCs) are required which represent an important tradeoff in terms of power, accuracy and latency. The power consumed by ADC depends on their resolution and sampling rate. For ADC to be implemented on chip, it has to operate in GSample/sec which consumes in the order of tens of milliwatts. [37] shows that 6bit 2.2 GS/sec ADC consumes 2.6mW. By scaling that to 7 bits ADC, we could conservatively multiply the power by two for every extra bit [19]. Fortunately, for our traces, only 5% of the time data from one *predictor* table needs to be written to SRAM-cache. This suggests that only a maximum of eight (size of an entry in predictor table) ADCs is required.

Both writing to SRAM, and prediction computation (assuming data in SRAM) require converting data in the cache into analog form. As in [29], current steering DACs are chosen because of their high speed and simplicity. The power overhead of such circuit is 7.4 mW and was shown to have negligible effect on the predictor accuracy.

5. MEMRISTOR PARAMETER DISCUSSION

In this paper we have suggested a hybrid approach, based on CMOS and memristor circuits, for branch prediction. Our circuits take advantage of the density of the memristor devices in spite of their intrinsic issues such as endurance and yield. Since memristor technology is still largely immature, it would be beneficial to explore branch predictor performance across wide range of plausible memristor characteristics as opposed to assuming specific parameters for memristor devices and circuits. In this way our study is more general and while it is less detailed it could answer on the question: how much memristor technology should be advanced before it becomes practical for this application?



(a) Updating Memristors-Perceptron table when an entry is evicted from cache (b) Prediction Computation in case of a Cache miss

Figure 5: Weight update algorithm using Distributed-Sum technique and Prediction computation

As of today, memristor device properties are not good enough for the considered application. For example, according to the recent study by Xu et al. [41] performed for $F_{cmos} = F_{nano} = 32$ nm (where F_{cmos} and F_{nano}, are CMOS and crossbar half pitch, correspondingly), the read latency optimized 8 MB memristor crossbar memories would have 1.7ns read latency but only 15% percent area efficiency providing density advantage of $\approx$ 6x over SRAM memory, which falls several magnitude short of the number required to start getting performance benefits. On the other hand, for the area optimized crossbar circuits the area efficiency is close to 40% but at the price of increasing latency by almost 60x and read energy by 10x. Thus, having high dense, energy efficient memristors is not readily achievable. However, these results are directly related to both the devices' assumptions as well as the peripheral logic. Thus, assuming more aggressive assumptions, which are nevertheless reasonable and based on the understanding of the physics behind memristor operation, could significantly improve memory performance.

For example, the area efficiency will be improved for larger crossbar arrays where the peripheral overhead (scales as $N * logN$ where N is the crossbar size) gets smaller compared to the useful memory area. However, this comes at the price of increasing the crossbar (RC) latency and read power. To this end, the latency can be made significantly smaller by having more aggressive type of the electrochemical memristors with lower dielectric constant, such as those based on SiO2 with eps = 3.9 [15].

Alternatively, to minimize peripheral overhead and dynamic power consumption for the multilayer crossbar memory, one can assume CMOL architecture which uses the double decoding scheme which allows creating large address space without adding new circuitry only at the price of increasing wire length [31]. In CMOL, the length of the crossbar segments is fixed and does not scale with the size of the CMOL array thus dynamic power consumption is not traded off for area efficiency and could be negligible for considered parameters [30].

Further improving the density could be achieved through the following. First, the crossbar wire size and pitch can be scaled very aggressively down to 5 nm e.g. using nanoimprint technology [39]. Second, multiple layers can be stacked monolithically - effectively reducing even further the footprint of the devices. Finally, storing multiple bits per memristor cell [5]. Thus, assuming smaller crossbar pitch, a near term 100x density advantage of the crossbar memristor memory can be achieved with either $F_{nano} \approx$ 13nm (still assuming $F_{cmos} =$ 32nm, 60%, 40% area efficiency for SRAM and memristor crossbar, respectively). Or alternatively, with $F_{nano} =$ 30nm and 5 monolithically stacked crossbar layers. Over the long term, a 1000x density could be obtained with $F_{nano} =$ 13nm and 10 crossbar layers [31].

The most important component, static leakage power, is mostly determined by the I-V nonlinearity (selectivity) in memristive devices. Engineering devices with better selectivity and recent progress in integration of diode layers [23] in the device stack are promising solutions to keep leakage power consumption within practical limits.

Finally, we provide a guideline for memristors' parameters that makes the technology valuable for our application.

1. 400x density improvement over SRAM; with reasonable latency; is required to maximize performance benefits(6%), overcome endurance limit, and make predictor fault tolerant.

2. In order to be competitive for analog computations, i.e. those done by SNAP [29], the overhead of memristor peripheral circuitry and, most importantly, that of

additional circuitry to implement conversion between digital and analog signals should be minimized. Similar to SNAP, we expect this overhead to be small especially assuming that hybrid CMOS/memristor circuits would be a natural candidate to implement digital-to-analog (DAC) and analog-to-digital (ADC) conversions. For example, resistor-ladder style DAC [5] and Hopfield network ADCs [35] could be implemented efficiently with hybrid CMOS/memristor circuits and are expected to have much smaller latency, area, and energy as compared to traditional approaches.

6. SIMULATION ENVIRONMENT

We use two simulation environments for our work: trace-driven and cycle-accurate simulations. The trace-driven simulation is used to explore the design space for our predictor, determine the optimal table sizes, cache sizes, and number of instances. Using these parameters, cycle-accurate simulation is used to provide an overall performance evaluation.

For trace-driven simulation, we used the infrastructure provided by the second (CBP-2) [3], and third branch predictor contest (CBP-3) [1]. We evaluated our predictor using SPEC2000 [3], SPEC2006[29], and CBP-3 traces [1]. The SPEC trace set includes 21 traces from both SPEC CPU2000 [3], and SPEC CPU2006 [29]. Each trace has 100 million branch instructions. The CBP-3 trace set includes 40 traces classified into 5 different categories; Client, Integer, Multimedia, Server and Workstation. Traces are approximately 50 million micro-ops long. For cycle accurate simulation, we used PolyScalar, a significantly modified version of SimpleScalar [4]. We used 8-wide processor with 12 cycles misprediction penalty. We run 7 SPEC CPU 2000 benchmarks using simpoints [28].

We compare our predictor to the winner of each contest, as well as the neural predictor we are augmenting. For SPEC CPU traces, we compare MNAP to two other predictors: L-TAGE [26](winner of CBP-2), and SNAP [29] (predictor we are augmenting). We restrict our hardware to the 32KB hardware budget as the predictors we are comparing to. Similarly, for the CBP-3 traces, we compare our predictor to both ISL-TAGE [27] (winner), and OH-SNAP [9] (predictor we are augmenting). We restrict our hardware to the 64KB hardware budget as the predictors we are comparing to. Due to lack of space, we only provide detailed results for SPEC traces. However, since CBP-3 traces have much higher variance, leading to many more updates per instruction to the predictor, we will discuss their results when necessary.

7. EXPERIMENTAL RESULTS

Using memristors to implement branch predictors is a study of trade-offs. We begin by exploring the main advantage of memristors; density and its potential benefit to prediction accuracy. We then evaluate solutions to handle the two main challenges of using memristors: the write endurance and the high defect rate. Our evaluation is area normalized, and not power normalized. At present, memristor interface circuitry would pose a problem for power normalized implementations. Ongoing work in 3D integration [30], however, promises to overcome this challenge and we leave this issue for future work. In addition, we assume a cell lifetime of 10^{12} [16].

7.1 Prediction Accuracy

Figures 7 shows MPKI results for SPEC traces at the sweet spot - 20x density improvement (as shown in section 3). A larger table improved the average MPKI by 9% over SNAP, and 3% over L-TAGE [26]. On the other hand, For CBP-3 traces, a larger table results in 3% improvement in average MPKI over OH-SNAP[9]. It also provides comparable accuracy (within 1%) to ISL-TAGE [27].

7.2 Write Endurance

Using memristors as full SRAM replacement in a branch predictor would shorten the lifetime of the chip to approximately 4 months or 12 hours for SPEC and CBP-3 traces, respectively. In this section we study the effect of our proposed write reduction techniques: caching of the most recently *updated* perceptron entries and spreading the writes among multiple instances.

7.2.1 Trading off cache space for redundancy

Assuming a fixed hardware budget of 32KB, and memristors 10000x more dense, we show the trade-off between cache size, and number of possible instances. Increasing the SRAM area increases our cache space, allowing us to hold more entries and thus potentially avoiding more updates. Increasing memristors area, however, allows us to have more instances and thus being able to absorb more writes (evictions from the SRAM) by spreading them over multiple instances.

Figure 8 illustrates that trade-off. The x-axis represents the amount of increase in number of logical rows in a single perceptron table. If the cache size is held constant and the number of logical rows is increased, then the number of instances is reduced. Within a fixed number of logical rows, as SRAM space is increased, the number of instances is also reduced. The maximum number of writes is the number of updates to a single row in the memristors, divided by the number of instances of that row.

What we expect, then, is that as the number of logical rows grows, the maximum number of writes will increase, since there are fewer instances at the same cache size. In addition, within a certain number of rows, there is some sweet spot, which is the "right" amount of cache in trading off between instances and cache space. Surprisingly, within a trace, the optimal cache size is the same, regardless of how many instances are left in the memristors. The amount of cache space we found to be optimal is 12K of a 32K budget (CBP-2[3]) or 17K of a 64K budget (CBP-3[1]).

7.2.2 Cache Effect on Write Endurance

In order to isolate the contribution of the cache, we counted the number of writes that would have occurred without using instances as shown in figure 9. We chose cache sizes of 12K, which were identified in previous section as the best configurations for SPEC traces when using both caches and instances. Using a cache to store the most recently updated perceptron entries reduces the number of writes by 96% on average for both trace families. Despite the significant reduction in number of writes, and with the assumption of 1GHz processor and 10^{12} write cycles, the predictor would only last for 24 months for SPEC CPU and 4 months for CBP-3 traces, respectively. Caches, alone, are not enough to make memristors feasible for our application.

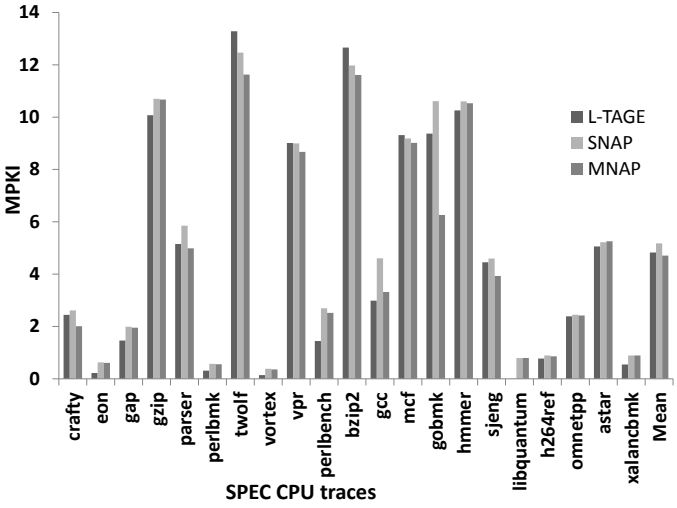


Figure 7: MPKI for SPEC CPU traces at 20x increase in table size (10k rows) over SNAP

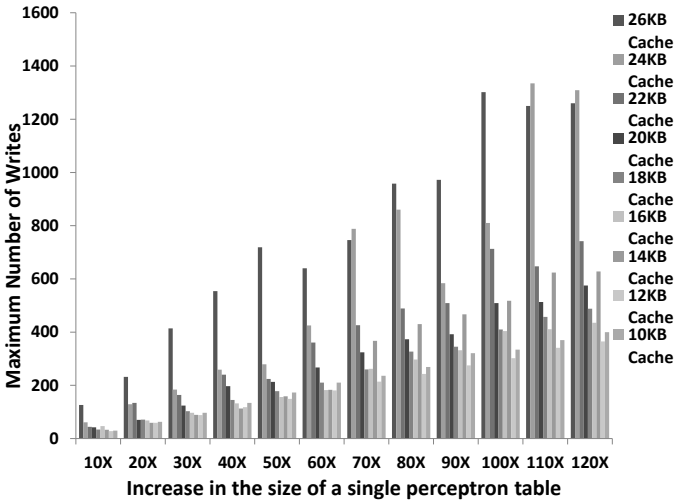


Figure 8: Trade-offs between cache size, instances, and the maximum number of writes across all traces

7.2.3 *Multiple Instances of Perceptron table*

Figure 9 shows the further reduction in writes when using the distributed sum scheme. With 5 instances per value and a realistic GRR (Global round robin) scheme, the maximum number of writes is further reduced by 80%, on average. Assuming a 1GHz processor and 10^{12} write endurance, SPEC CPU and CBP-3 traces require 5 and 40 instances to last 6 years, respectively. For the CBP-3 trace, many more instances are required because of a substantially higher update rate.

7.3 Defective Devices Model

Figure 10 shows the effect of the percentage of defective devices on the accuracy of prediction without assuming any of our techniques. The prediction accuracy is highly sensitive to a high percentage of defective devices. For a 10% defect rate, the prediction accuracy is reduced by 14% and 4% for SPEC and CBP-3 workloads, respectively. Note that a 10% defect rate at the bit level leads to up to a 70% defect rate at the weight level since each weight is made up of 7 bits. However, with our proposed schemes, we found that using 10 instances per weight makes our predictor fault tolerant ($\prec$ 0.2% increase in MPKI) for both traces.

In conclusion, assuming SPEC CPU traces and a hardware budget of 32KB (used in CBP-2[3]), memristors that are 400x denser than SRAM are enough. On the other hand, assuming CBP-3 traces and a hardware budget of 64KB (used in CBP-3[1]) requires much denser memristors (1000x). As discussed in section 5, these densities could be achieved, on long term, assuming more aggressive assumptions about the technology, which are nevertheless reasonable and based on the understanding of the physics behind memristor operation.

7.4 Results from Microarchitectural Simulation

In this section, we provide overall performance evaluation of our predictor as compared to SNAP (the predictor we are augmenting). In addition, we show the effect of write latency on overall performance.

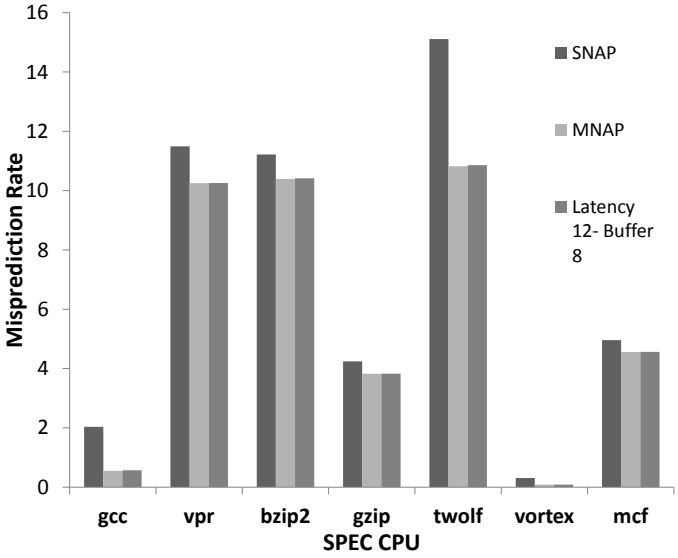


Figure 15: Misprediction rate assuming 12ns write latency

7.4.1 *Performance Evaluation*

Figure 11 shows the misprediction rate for SPEC benchmarks. On average, our predictor provides 7.9% improvement over SNAP. Figure 12 shows the improvement in IPC over SNAP. On average, our predictor results in performance improvement of 2.17%. For twolf, the improvement in IPC is particularly good, 6% due to 28% improvement in misprediction rate. The results in this section assume that the write latency for memristors is 1 cycle. In the next section, we will study the effect of write latency on the performance of MNAP predictor.

7.4.2 *Write Latency*

Figure 13 provides a sensitivity analysis of overall IPC improvement over SNAP as write latency increases. Here we assume a fixed size buffer of 1 cache entry per predictor table, i.e. a buffer size of 8 weights per table (as suggested by figure 6) and that data is written to memristor perceptron table when the processor stalls. For write latencies

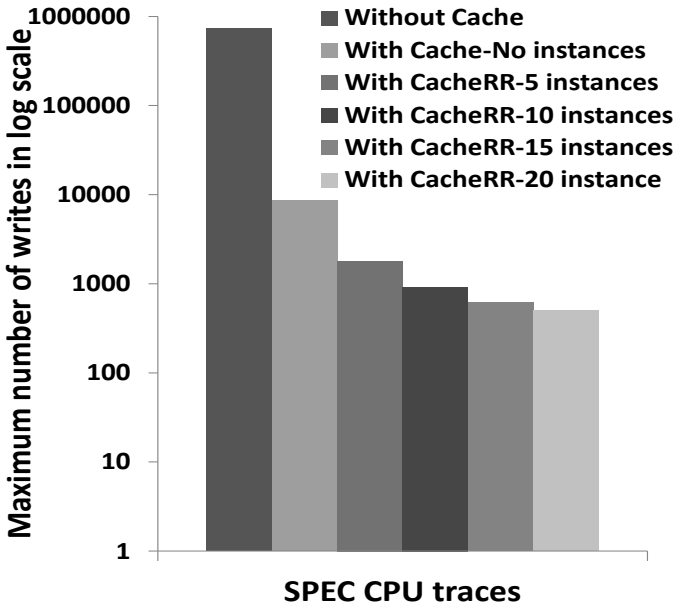


Figure 9: Effect of caching and distributed-sum scheme on the Maximum number of writes for SPEC traces. Cache size is 12KB

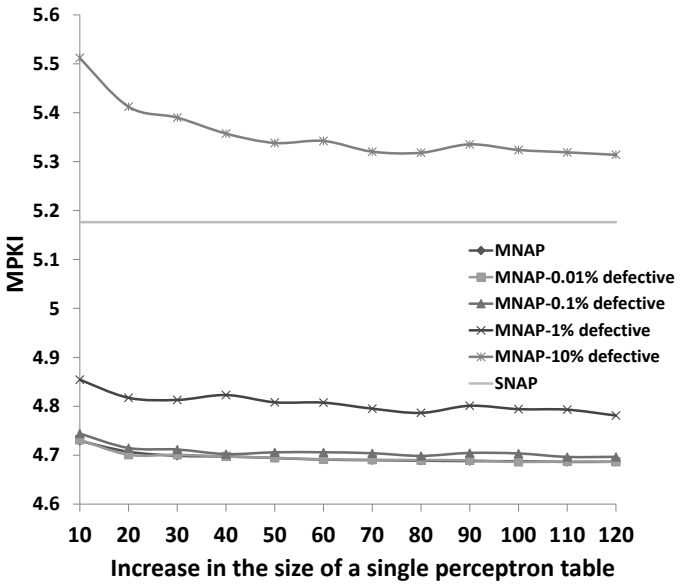


Figure 10: Effect of percentage of defective devices on MPKI for SPEC traces for a wide range of densities

$\leq$ 12ns, the loss in performance as compared to the ideal case (MNAP with 1 cycle write latency shown in figure 12) is negligible. On the other hand, for write latencies $>$ 12 ns some of the benchmarks still performs well, e.g. gcc. However, other benchmarks like twolf show large performance degradation.

The reason is that though the percentage of times data needs to be written to memristors is $<$ 2%, the average size of data dropped due the write buffer being full is only 8 weights (i.e. single cache entry) for small latencies ($\leq$ 12ns). However, the average data dropped increases to be 48 weights (i.e. 6 cache entries) as latency increases. Specifically for twolf, it drops 64 weights on average. We further explore the potential of adding more write buffer space; however, this does not lead to significant improvement over the base case (SNAP) and will come with the cost of additional power and area.

Figure 14 shows detailed performance results assuming write latency of 12ns, and write Buffer of size one entry (8 weights) per table. The loss in performance compared to the ideal case (fig. 12) is negligible. We also evaluate the scheme where fetch stalls when data from memristors needs to be written. This approach still provides performance improvement over SNAP for some of the benchmarks mainly because the cache absorbs most of the updates and thus leading to memristors being written infrequently. For other benchmarks like bzip2, performance degrades significantly as data needs to be written to memristors 2% of the time. Figure 15 shows the misprediction rate assuming 12ns latency and a buffer size of 1 cache entry per predictor table. The increase in MPKI compared to ideal case (MNAP with 1 cycle write latency) is negligible.

8. RELATED WORK

Wear Leveling: Wear Leveling has been studied extensively for Flash, and resistive memories like PCM. [22] proposes a hybrid main memory system consisting of a PCM device and a tightly coupled DRAM buffer that reduces page faults by 5x. [21] proposes start-gap wear-leveling, a low-overhead hardware wear-leveling technique that boosts the lifetime of PCM-based system to 97% of the an ideal wear-leveling scheme. Unlike our work, these techniques target main memory and thus are not latency constrained. Dynamically replicating memory [7] uses hardware and operating systems' support to provide continued operation through graceful degradation. They replicate a physical page that contains hard fault into two faulty, complementary pages, and future reads access both pages to retrieve the correct data. This work is orthogonal to previously proposed wear-leveling techniques.

Neural Branch Predictors: Many enhancements were proposed to improve the accuracy of neural predictors. [25] proposes to partition the weight table into separate tables, each indexed differently, to decrease the aliasing between branches. [24] proposes using a redundant representation of the branch history to improve the prediction accuracy. While these designs lead to an accurate prediction, latency and power consumption for the branch output computation have prevented these neural predictors from being competitive. [8] [11] propose to reduce the latency of neural predictors but sacrifice accuracy. They use ahead pipelining that gradually computes a prediction by adding weights ahead of time so that only the bias weight must be added to the total for a prediction to be made.

SNAP [29] proposes a low power neural predictor by using analog circuitry to do prediction (avoiding the expensive dot product operation). In our work, we propose an alternative method for doing computations by using the analog properties of memristive devices to do prediction. In addition, we exploit the memristive devices high density to provide a highly-accurate predictor.

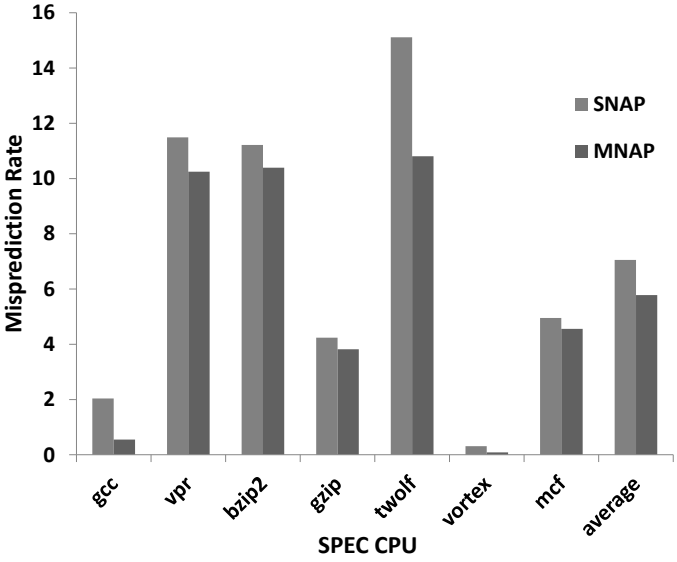


Figure 11: Misprediction rate of SPEC CPU

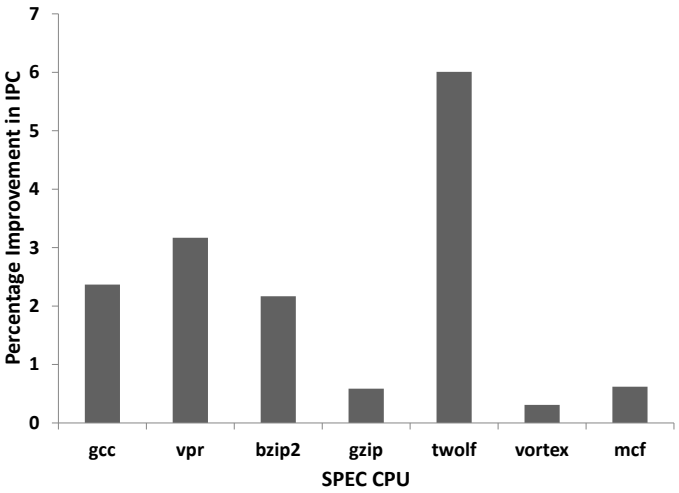


Figure 12: Performance Improvement over SNAP

9. CONCLUSIONS

This paper proposes general techniques for wear-leveling and fault tolerance that are specifically good for delay sensitive applications. We propose hybrid SRAM-memristor storage to overcome the write endurance limit of memristors. In addition, we use a distributed sum scheme that both increases memristor durability, and makes our design fault tolerant to a high percentage of defective devices.

We present a limit study of an analog neural branch predictor. We chose this application due to its challenging requirements in terms of latency and frequency of updates. Our techniques were able to extend the lifetime of the predictor for more than 5 years and is fault tolerant as well. While current technology parameters does not allow high dense power efficient memristors' implementation, we discussed a range of plausible characteristics that as the technology advances would make it suitable for our application. In future, we want to explore various neural network applications (e.g. face recognition) and study the role of memristors for such applications.

10. ACKNOWLEDGMENTS

This work was supported in part by NSF grant CCF-1017578.

11. REFERENCES

[1] http://www.jilp.org/jwac-2/.

[2] ITRS 2009. http://www.itrs.net.

[3] *JILP Special Issue: The Second Championship Branch Prediction Competition (CBP-2)*, 2007. http://cava.cs.utsa.edu/camino/cbp2.

[4] D. Burger and T. M. Austin. The simplescalar tool set version 2.0. *Tech Report, CS Department, University of Wisconsin.*

[5] B. H. F. Alibart, L. Gao and D. Strukov. High-precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. In *Nanotechnology*, 2012.

[6] J. E. Green et al. A 160-kilobit molecular electronic memory patterned at 1011 bits per square centimetre. *Nature*, 445(7126), 2007.

[7] E. Ipek et al. Dynamically replicated memory: building reliable systems from nanoscale resistive memories. *SIGPLAN Not.*, 2010.

[8] D. A. Jimenez. Fast path-based neural branch prediction. In *MICRO*, 2003.

[9] D. A. Jimenez. Oh-snap: Optimized hybrid scaled neural analog predictor. *JWAC-2: Championship Branch Prediction*, 2011.

[10] D. A. Jimenez and C. Lin. Dynamic branch prediction with perceptrons. *HPCA*, 0, 2001.

[11] D. A. Jiminez. Piecewise linear branch prediction. In *ISCA*, 2005.

[12] D. A. Jiminez and L. Calvin. Neural methods for dynamic branch prediction. In *ACM Trans. Comput. Syst.*, 2002.

[13] S. H. Jo, K.-H. Kim, , and W. Lu. High-density crossbar arrays based on a si memristive system. *Nano Lett*, 9(2):870.874, 2009.

[14] S. H. Jo, K.-H. Kim, and W. Lu. High-density crossbar arrays based on a si memristive system. *Nano Lett.*, 9(2):870–874, 2009.

[15] S. H. Jo and W. Lu. Cmos compatible nanoscale nonvolatile resistance switching memory. *Nano Letters*, 2008.

[16] M.-J. Lee et al. A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta205-x/TaO2-x bilayer structures. *Nature Materials*, 2011.

[17] K. K. Likharev. Hybrid cmos/nanoelectronic circuits: Opportunities and challenges. *J of Nanoelectronics*, 2008.

[18] T.-Y. Liu, T. H. Yan, et al. A 130.7 mm2 two-layer 32-gbit reram memory device in 24-nm technology. In *ISSCC*, 2013.

[19] B. Murmann. A/d converter trends: Power

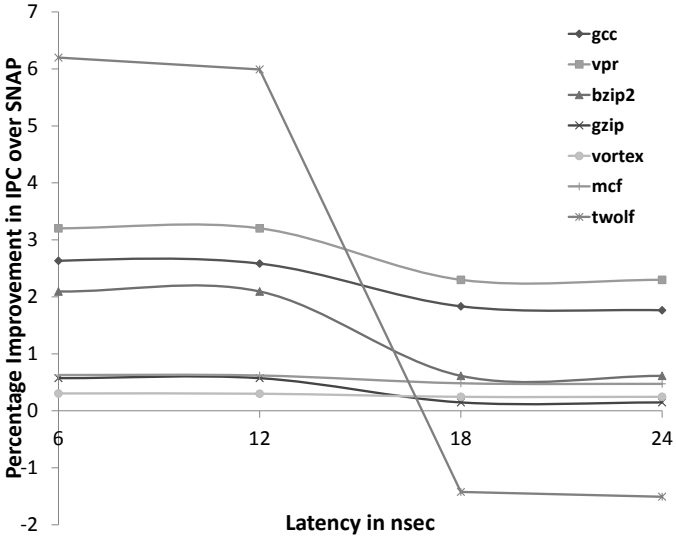


Figure 13: The effect of write latency on Performance

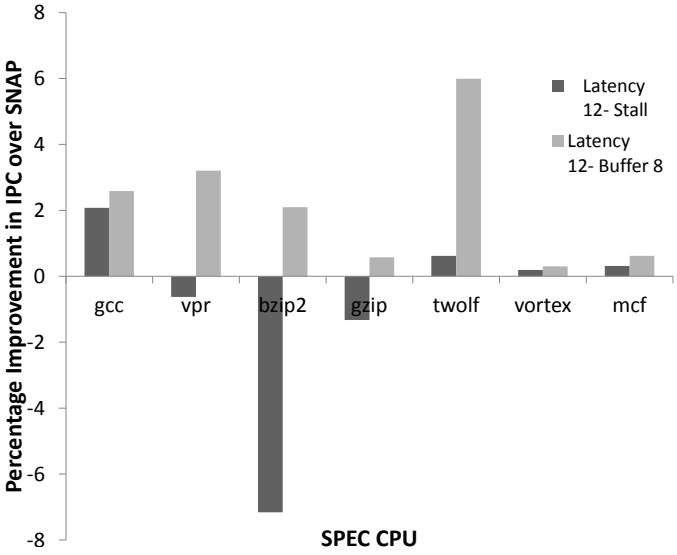


Figure 14: Performance Improvement assuming 12ns write latency

dissipation, scaling and digitally assisted architectures. In *CICC IEEE*, 2008.

[20] Y. V. Pershin and M. D. Ventra. Memory effects in complex materials and nanoscale systems. *Advances in Physics*, 60, 2010.

[21] M. K. Qureshi et al. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In *MICRO*, 2009.

[22] M. K. Qureshi et al. Scalable high performance main memory system using phase-change memory technology. In *ISCA*, 2009.

[23] K. B. S et al. High-current-density cuo x/inznox thin-film diodes for cross-point memory applications. *Adv. Mater*, 2008.

[24] A. Seznec. Redundant history skewed perceptron predictors: Pushing limits on global history branch predictors. 2003.

[25] A. Seznec. Analysis of the o-geometric history length branch predictor. *ISCA*, 0:394–405, 2005.

[26] A. Seznec. A 256 kbits l-tage branch predictor. *JILP TheSecond Championship Branch Prediction Competition (CBP-2)*, 2007.

[27] A. Seznec. A 64 kbytes isl-tage branch predictor, 2011.

[28] T. Sherwood et al. Automatically characterizing large scale program behavior. In *ASPLOS*, 2002.

[29] R. St. Amant, D. A. Jimenez, and D. Burger. Low-power, high-performance analog neural branch prediction. In *MICRO*, 2008.

[30] D. Strukov et al. Defect-tolerant architectures for nanoelectronic crossbar memories. *J. of Nanoscience and Nanotechnology*, 2007.

[31] D. Strukov et al. Four-dimensional address topology for circuits with stacked multilayer crossbar arrays. *Proc. of NAS.*, 2009.

[32] D. Strukov and K. Likharev. Hybrid cmos/nanodevice circuits for digital signal processing. *IEEE Trans. Nanotechnology*, 2007.

[33] D. Strukov and R. Williams. Exponential ionic drift: Fast switching and low volatility of thin film memristors. *Applied Physics*, 2009.

[34] D. B. Strukov and H. Kohlstedt. Resistive switching phenomena in thin films: Materials, devices, and applications. *MRS Bulletin*, 2012.

[35] D. Tank and J. Hopfield. Simple neural optimization networks: An a/d converter, single decision circuit, and a linear programming circuit. *IEEE transactions on Circuits and Systems*, 1986.

[36] A. Torrezan et al. Sub-nanosecond switching of a tantalum oxide memristor. *Nanotechnology*, 2011.

[37] B. Verbruggen et al. A 2.6mw 6b 2.2gs/s 4-times interleaved fully dynamic pipelined adc in 40nm digital cmos. In *(ISSCC)*, 2010.

[38] P. O. Vontobel et al. Writing to and reading from a nano-scale crossbar memory based on memristors. *Nanotechnology*, 2009.

[39] W. Wu et al. Sub-10 nm nanoimprint lithography by wafer bowing. *Nano Letters*, 2008.

[40] Q. F. Xia et al. Memristor-cmos hybrid integrated circuits for reconfigurable logic. *Nano Lett*, 9:3640–3645, Oct 2009.

[41] C. Xu, X. Dong, N. Jouppi, and Y. Xie. Design implications of memristor-based rram cross-point structures. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, march 2011.

[42] C. J. Xue et al. Emerging non-volatile memories: opportunities and challenges. In *Procedings of CODES+ISSS*, 2011.

[43] J. Yang et al. Memristive switching mechanism for metal/oxide/metal nanodevices. *Nature Nanotechnology*, 2009.