Bandwidth-Aware Reconfigurable Memory Design with Hybrid Memory Technologies

***Abstract*—In future chip-multiprocessor (CMP) design, memory bandwidth is a potential bottleneck to system performance. Emerging non-volatile memory technologies, such as spin-torque-transfer memory (STT-RAM), resistive memory (RRAM), and embedded DRAM (eDRAM), are promising solutions as on-chip memories for CMPs. In this paper, we propose a bandwidth-aware reconfigurable on-chip memory (BAROM) design with hybrid memory technologies. We demonstrate a method to dynamically reconfigure the number of shared memory levels and capacity of each level based on statistical prediction. With a set of both multithreaded and multiprogrammed applications, we evaluate system performance obtained with our method. The experimental results show that the proposed reconfigurable hybrid memory design improves the overall system throughput by % compared to pure SRAM memory design. In addition, our design leads to % throughput increase compared to hybrid but fixed memory hierarchy design.**

I. Introduction

One potential bottleneck for chip-multiprocessor (CMP) performance scaling is the widening gap between the bandwidth demand created by processor cores and the limited access bandwidth provided by off-chip main memory [1]–[3]. Such limitation greatly affects the parallelism of memory demanding applications, i.e., applications with a large working set, by consuming additional cycles on off-chip memory access. In addition, even moderate memory demand applications will be affected as the number of cores scales up [4]. The continuing shrink of transistor density not only leads to increasingly powerful processors with a large number of cores, but also puts much bandwidth pressure to off-chip main memory. However, the bandwidth to the off-chip main memory does not improve much compared to the processor core scaling. Potentially, these issues make the applications running on the computing systems be bandwidth-bound at main memory.

While a number techniques can be found in today's systems and research work to alleviate such bandwidth bottleneck, few can improve the system throughput in a power and cost efficient manner. High performance computing machines such as NVIDIA's Tesla [5] rely on very high main memory bandwidths (provided by GDDR memories) to feed the requirement from the processor. However, GDDR memories run at higher clock rates and are more power hungry than conventional DRAM modules. It is undesirable for either general purpose or high performance computing systems to improve their computing performance by sacrificing power efficiency. With the emerging 3D chip technology, caches can be stacked on top of processor cores to provide high memory bandwidth [6]. However, ...

Caching has long been employed as the most effective approach to reduce memory access latency. Proper on-chip memory hierarchy design, however, can also help alleviate the increasing bandwidth pressure of off-chip memory. After an extensive study the problem of limited pin bandwidth to multiprocessor systems, Burger *et al.* conclude that even more complex on-chip cache structures would prove to be costeffective with the limited pin bandwidth severely restricting performance increases [2]. The requirements of on-chip cache structures along with all sorts of optimization techniques that come along with scaling of processor core numbers are carefully explored [3]. The study shows that cache size need to grow much faster than processor core numbers to compensate the limited off-chip bandwidth. The study also shows that effective bandwidth optimization techniques can help reduce cache size requirement and thus help scaling processor cores. Consequently, it is critical to re-design the on-chip memory system, which focuses on bandwidth optimization.

On-chip caches with fast random access, high storage density, and non-volatility become possible due to the emergence of various new non-volatile memory (NVM) technologies, such as spin-torque-transfer memory (STT-RAM), phase-change memory (PCRAM), and resistive memory (RRAM). These emerging memory technologies are believed to be promising solution as on-chip caches [7]. In addition to the benefit of non-volatility, we demonstrate that these memory technologies provide higher bandwidth than SRAM at a large capacity. Consequently, it is beneficial to use NVM as lower level caches when large capacity.

This paper presents a bandwidth-aware memory hierarchy design to enhance system performance of CMPs. Our goal is to enhance the memory system from the bandwidth point of view, by leveraging these emerging memory technologies in devising a hybrid cache hierarchy. The contributions we present in this paper include:

- Bandwidth-aware hybrid on-chip memory hierarchy design...
- Reconfiguration...
- Prediction engine...

II. Related Work

The bandwidth problem of multicore processors has drawn much attention recently. Yu *et al.* proposed a LLC partitioning algorithm to minimize bandwidth requirement to off-chip main memory [**?**]. While most cache partitioning techniques focus on cache miss rates, their work takes a different approach in which tasks memory bandwidth requirements are taken into account when identifying a cache partitioning for multiprogrammed and/or multithreaded workloads. Cache resources are allocated with the objective that the overall system bandwidth requirement is minimized for the target workload. The

key insight is that cache miss-rate information may severely misrepresent the actual bandwidth demand of the task, which ultimately determines the overall system performance and power consumption. However, their work only focused on LLC (L2 cache). Our design targe is the overall on-chip memory system, which provides more design dimensions and flexibility. An analytical performance model was proposed by Sun et al. to explore the design space of memory hierarchies for throughput computing [moguls]. However, their memory hierarchy design was tailored for individual applications, which was undesirable for a real system design.

A large body of recent research focuses on exploring new memory technologies to trade off between latency, bandwidth, and cost. Wu et al. explored various memory technologies - SRAM, eDRAM [8], and STT-RAM [9] to best construct L3 caches in terms of performance and power. The eDRAM and STT-RAM technologies are evaluated as LLCs by stacking each memory on top of the processor die. Both studies mainly focused on reducing the latency gap between L2 cache (LLC) and external memory, but do not examine the design from bandwidth perspective.

Application behavior prediction is a critical component of reconfigurable architectures. Zhou *et al.* monitored memory access patterns and estimated memory behavior of workloads for energy efficient memory allocation [10]. Duesterwald *et al.* describe different statistical and table based predictors for within- and across-metric predictions of performance monitoring information [11]. They show that the table-based predictor generally outperforms the other predictors they tested. Sarikaya *et al.*describe an optimal prediction technique based on a predictive least squares minimization [12]. Sarikaya *et al.* showed the benefit of statistical metric modeling for tracking varying pattern history lengths and modeling long term patterns [13]. However, ...

III. Background

STT-RAM STT-RAM uses the magnetic property of the material and uses Magnetic Tunnel Junction (MTJ) as its binary storage. As shown in Fig. 1, MTJ contains two ferromagnetic layers and one tunnel barrier layer. The direction of one ferromagnetic layer is fixed, which is called the reference layer, while the direction of the other one can be changed by passing a driving current, which is called the free layer. The relative magnetization direction of two ferromagnetic layers determines the resistance of MTJ. If two ferromagnetic layers have the same directions, the resistance of MTJ is low, indicating a "0" state; if two layers have different directions, the resistance of MTJ is high, indicating a "1" state.

RRAM Memristor, a portmanteau of "memory resistor", is a generalized resistance that maintains a functional relationship between the time integrals of current and voltage. Memristor was first theoretically predicted by Chua in 1971 [14] as the fourth fundamental circuit element from the completeness of relations between the four basic circuit variables, namely, current, voltage, charge, and flux-linkage. The first memristor practical demonstration was presented by Williams *et al.* in 2008 [15]. Fig. 2 shows a conceptual view of the memristor

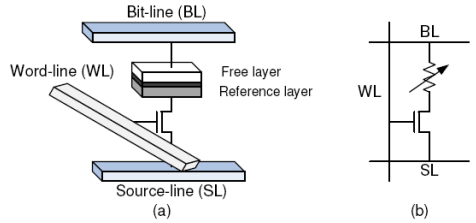


Fig. 1. Demonstration of a MRAM cell. (a) Structural view. (b) Schematic view.

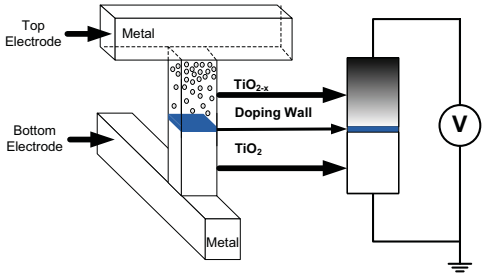


Fig. 2. The conceptual view of the structure of memristor cells.

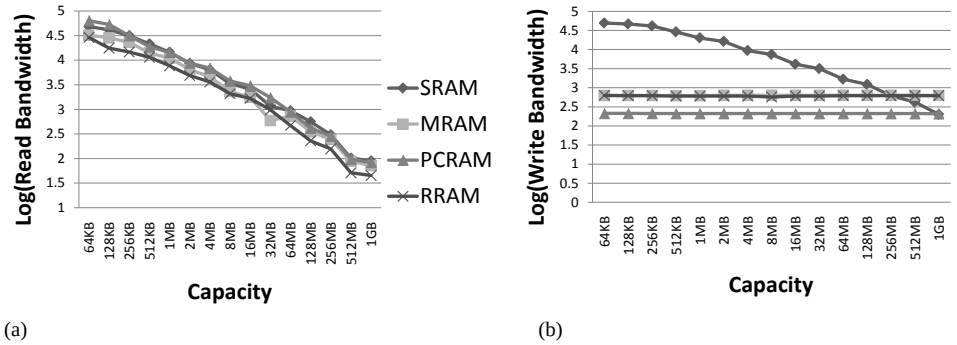
structure [15]. The top electrode and bottom electrode are two metal nanowires on platinum, and the thin titanium dioxide film is sandwiched by the electrodes.

eDRAM??

IV. Memory Technology Exploration

Many modeling tools have been developed during the last decade to enable system-level design exploration for SRAM- or DRAM-based cache and memory design. For example, CACTI [16] is a tool that has been widely used in the computer architecture community to estimate the speed, power, and area of SRAM and DRAM caches. In addition, CACTI has also been extended to evaluate the performance, power, and area for STT-RAM [17], PCRAM [18], [19], and NAND flash [20]. However, as CACTI is originally designed to model SRAM-based cache, some of its fundamental assumptions do not match the actual NVM circuit implementation, and thereby these CACTI-like estimation tools do not model the NVM array organization in the exact way that the chip is fabricated. In this section, we use *NVSim*, a circuit-level model for NVM performance, energy, and area estimation, which supports various NVM technologies including STT-RAM, PCRAM, RRAM, and conventional NAND flash.

First of all, we estimate the read and write bandwidths that can be provided by different memory technologies. Figure 3 shows the results, with both x- and y-values in *log* scale. The figure illustrates both the provided read and write bandwidth as a function of memory capacity. Each of the memory technologies actually provide nearly the same read bandwidths, as is shown in figure 3(a). On the other hand, a straight forward observation from figure 3(b) is that the write bandwidth varies among different memory technologies. The shape of the SRAM write bandwidth curve is very similar to the read bandwidth curve. The write bandwidth curves of the other three memory technologies appear to be



(a) (b)

Fig. 3. Read and write bandwidths provided by different memory technologies. (a) Read bandwidth provided by different memory technologes. (b) Write bandwidth provided by different memory technologies.

very different. The reason is that write latencies of the three non-volatile memories are much higher than read latencies. Another observation from figure 3(b) is that the curves cross to each other at different locations.

Being aware that the read and write latencies of NVM are asymmetric, we consider the read latency at first. In NVsim, we divide the entire cache read latency into the following components:

1) H-tree input delay
2) Decoder + word-line delay
3) Bit-line delay + Sense Amplifier delay
4) Comparator Delay (for tag part only)
5) H-tree output delay

H-tree latencies 1) and 5) are mainly determined by the RC delay of global wires, which is positive proportional to the area of memory macro. Sensing delay 4) is related to the read noise margin of memory cell that is affected by off/on resistance ratio. Figure 4 (a) illustrates the read latency of different memories. We can see that sensing delay dominates the read latency of NVM at small capacity so that PCRAM (with the largest resistance window) is faster than RRAM and MRAM (with the smallest resistance window). While H-tree delay unveils at large capacity so that RRAM (with the smallest cell size) becomes faster than PRAM and MRAM (with the biggest cell size). The read latency of SRAM bank will increase rapidly after 128MB due to large area.

Write latency of NVM are almost dominated by the write pulse width. In this work we assume 10ns, 20ns, 100ns for MRAM, RRAM and PCRAM. While write latency of SRAM and eDRAM is a function of capacity, as similar to the read latency. The results in 4 (b) indicates that NVM is suitable for memory with large capacity.

Figure 5 has demonstrated the dynamic energy of different memory technologies when 20% and 50% write access are assumed. eDRAM will be better than SRAM in terms of dynamic energy after 16MB and this is verified by IBM Power7 L3 cache. The cross-point between NVM and SRAM/eDRAM is postponed for 50% write than 20% write.

Based on these observations, we would like to design a bandwidth-aware hybrid memory hierarchy, which always provides the high memory bandwidth with the given capacity. For example,

- eDRAM has better latency and enegry than SRAM when capacity is larger than 16MB.
- MRAM is more competitive to SRAM and eDRAM when capacity is larger than 128MB.
- PCRAM has serious endurance issue and is targeted as main memory replacement.
- RRAM might fit into cache hierarchy as last level cache replacement when there multiple levels of cache for applications with very few writes.

V. Design Method

Based on our memory technology exploration, we now provide an overview of BAROM design. While many design methods involved with new memory technologies endeavor to reduce the off-chip memory access latency, this work focuses on minimizing off-chip bandwidth requirement by employing hybrid on-chip memory and reconfiguration. For each level of memory, we select the memory technology that has the highest provided bandwidth within given capacity. The total capacity of each level is then partitioned to subsets of ways with different provided bandwidths. On top of such hardware hierarchy design, we apply dynamic memory system reconfiguration with the bandwidth demand of a specific set of applications as the major metric. One desirable characteristic of BAROM is that it provides prioritized access to the residual memory capacity available at runtime instead of sharing it among all memory accesses. By having control over the residual capacity, we can better utilize the limited resources to enhance system performance. The hardware reconfiguration is further facilitated by a prediction engine. Rather than conventional last value or history table based predictors, we employ a probability-based statistical predictor which can achieve higher accuracy with small performance overhead. The rest of this section describes the BAROM design, which is composed of on-chip memory hierarchy design, the reconfiguration mechanism, and the prediction engine.

A. Memory Hierarchy Design

Figure 6 depicts an overview of BAROM hardware configuration. The CMP system consists of multiple cores, where L1 cahces are private to each core and lower level caches are shared by the cores. With the provided bandwidth vs. capacity

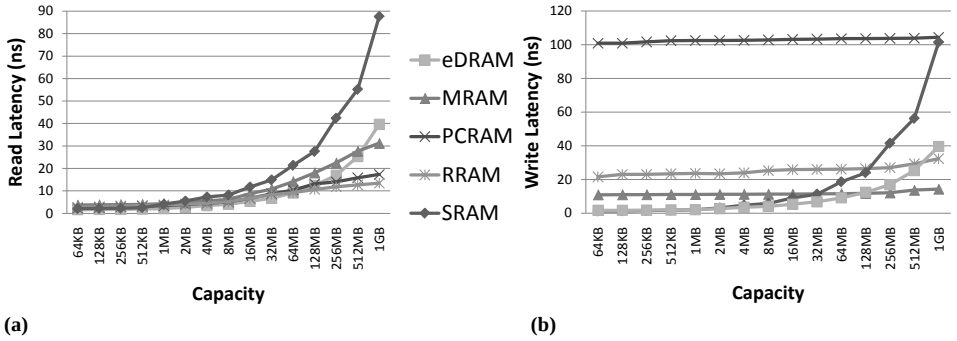


Fig. 4. Latency of different memory technologies. (a) Read Latency. (b) Write Latency.

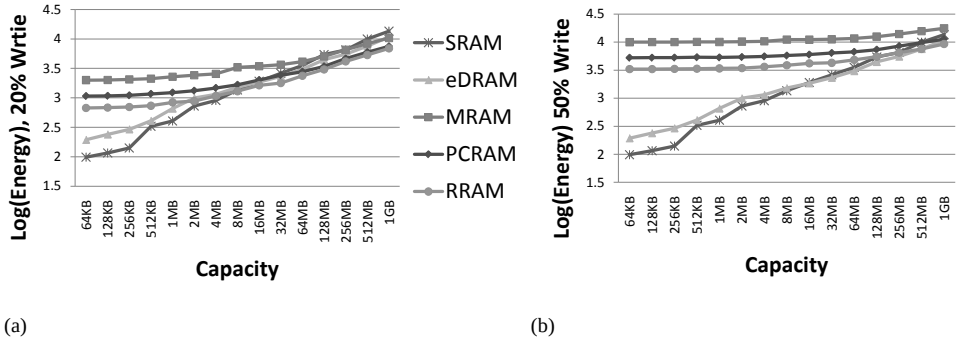


Fig. 5. Dynamic energy consumption with the provided bandwidths of different memory technologies. (a) Dynamic Energy with 20% write. (b) Dynamic Energy with 50% write.

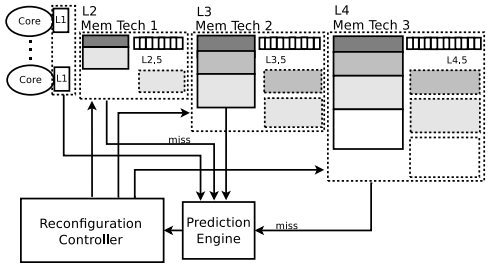


Fig. 6. Overview of BAROM hardware configuration.

curves of various memory technologies that are obtained using the method described in section **??**, we can design a hybrid on-chip memory system to always provide the highest bandwith. In order to achieve this goal, we configure the memory hierarchy in the following manner.

Number of levels. In order to decide the number of memory levels, we examine the provided bandwidth curve. Memory levels are seperated by the cross points of two memory technologies. For example, Figure 3 illustrates that there are two cross points (SRAM and STT-RAM, STT-RAM and RRAM). Therefore, we will have three levels of shared caches.

Memory technology of each level. The memory technology with the highest provided bandwidth within the given range of capacity of each level is selected. In our case, SRAM, STT-RAM, and RRAM are selected as the L2, L3, and L4 caches respectively.

Capacity of each level. The maximum availabe capacity of each level is determined by the cross point of two memory technologies. In our case, the capacities of SRAM/L2, STT-RAM/L2, and RRAM/L4 are 2MB, 512MB, and 1GB respectively. Each level of memory if further configured to be multiple banks the same way as the conventional cache design. At the system initialization, the capacity of each level is defined to be the maximum available capacity of the level. According to the reqirement of different applications, the capacities can be reconfigured at run-time to provide best performance. The memory space of each level can be reconfigured to form an internal level as shown in Figure 6. The hardware and software supports for reconfiguration will be presented in the following sections.

The goal of such architecture design is to make use of different memory technologies to configure an on-chip memory system with optimal provided bandwidth over a range of capacities. As a result, the overall provided bandwidth curve of the on-chip memory system will appear to be an envelope line of all the available memory technologies as illustrated in Figure **??**. It is illustrated in this figure that the hybrid memory system always provides the highest bandwidth over different capacities.

B. Reconfiguration

Although the above hybrid memory configuration maintains the optimal provided bandwidth curve over all the capacities, it does not garantee the best performance for different applications with a variety of bandwidth requirements. The overall provided bandwidth curve shown in Figure **??** appears to be monotonically decreasing. A specific application may have a bandwidth requirement that can be mapped to a single point on

the curve. In order to satisfy the bandwidth demand of different applications, we dynamically reconfigure the memory space of each level corresponding to the bandwidth requirement of each set of applications running on the system. To be more aggressive, the reconfiguration is applied at the end of each given execution time interval to accomodate different program phases of applications.

The major system parameter to be reconfigured is the capacity of each memory level. At each time interval, we determine the upper bound of the capacity of memory level-i (s_i^u) by mapping the demand bandwidth (DBW) of a specific application to the system's provided bandwidth curve, i.e., $s_i^u = f^{-1}(\text{DBW})$ where $f(x)$ represents the provided bandwidth curve of the on-chip memory system stored in the look-up-table in a reserved non-volatile memory space. The demand bandwidth for each memory level is generated using the prediction engine presented in section **??**. In previous sections, we show that the memory with smaller capacity provides higher bandwidth. Theoritically, the higher bandwidth provided by the memory system leads to higher performance (in terms of throughput) and power consumption of throughput computing applications. Consequently, we define a lower bound of memory capacity as $s_i^l = f^{-1}(\text{DBW} * (1 + \sigma))$, where σ is a pre-defined threshold to make sure possible provided bandwidth increase without much power overhead. The capacity of memory level-i (s_i) is therefore selected to satisfy $s_i^l \leq s_i \leq s_i^u$. Furthermore, the rest of the capacity can be reconfigured to become level-i.5 if necessary. We describe the reconfiguration algorithm as follows.

¡pseudocode¿ if (si ¡ stotali/2): if (si+1 ¡= stotali-si): si.5 = stotali-si;

else: si+1 = ...;

Our primary reconfigurable design exploits set associativity in conventional cache organizations. The desirability of such configuration is to exploit the division of ways already present in a conventional cache organization. Figure 6 shows the block diagram for the organization of our design. An n-way set associative cache consists of n data and tag arrays. We divide the each level of on-chip memory into partitions at the granularity of the k-ways, where k is determined by the available capacity range of a specific memory technology on the provided bandwidth curve. ¡100+ ways? experiment section¿ Reconfiguration will not affect the bits of the address fields that are used as tag, index, and block offset bits. Modifications to the conventional cache architecture include:

- **Memory status vector** At each memory level, the corrent configuration is stored in a memory status vector.
- **Input and output paths** The input and output data paths are duplicated.
- **Additional multiplexors** Futhermore, additional wiring and multiplexors at address decoders and tag comparators are also required to support the reconfiguration.

C. Prediction Engine

Memory reconfiguration relies on accurate predictions of demand bandwidth of a workload in response to the dynamically varying application characteristics. Conventional last value or table based predictors can model neither long range patterns of an application nor patterns with variable lengths. In this work, we employ a statistical predictor to support for the BAROM design. The basic idea of the predictor is borrowed from the n-Gram models, which are typically used by speech and natural language processing [21]. The n-gram models are usually formulated as a probability distribution $p(s)$ over a strings s, and attemp to reflect how frequently the string occurs as a sentense. Prediction on a reasonable combination of strings is then generated based on the probability distribution. Our predictor employs the same basic idea, but with very different implementation considerations. While language modeling is built from a set of previous collected training sentenses with finite lengths, our statistical model needs to be able to dynamically generate prediction based on a continuous sequence of metrics. In addition, we can only implement limited resolution for the metrics. Therefore, we need to normalize the demand bandwidth values with a limited number of quantization bins. In our model, each demand bandwidth (DBW) sample abtained in a time interval is analogous to a string in language, and a given length (the "order") of samples is stored in a table as a pattern. At each time interval, the prediction engine will update the pattern table with the new DBW sample, and calculate probability of the updated pattern. The probability of a pattern s of length l can be calculated without loss as the product of a sequence of conditional probabilities using the equation shown below.

$$p(s) = p(w_1)p(w_2|w_1)...p(w_l|w_1...w_{l-1}) = \prod_{i=1}^{l} p(w_i|w_1...w_{i-1}) \tag{1}$$

where w_l is the DBW sample obtained in current time interval, and w_1 through w_{l-1} are the preceding $l-1$ DBW samples. In n-gram models, we make the approximation that each conditional probability only depends on the preceding $n-1$ samples and obtain the following equation.

$$p(s) = \prod_{i=1}^{l} p(w_i|w_{i-n+1}^{i-1}) \tag{2}$$

in which w_{i-n+1}^{i-1} denotes the sequence of $w_{i-n+1}...w_{i-1}$. In order to compute the result of Equation 2, an estimation of of $p(w_i|w_{i-n+1}^{i-1})$ can be generated using maximum likelihood (ML) estimation method. In n-gram models, the largest n in wide use is n=3 which is called a trigram model. Therefore, we adopt n=3 in our predictor. Each conditional probability is calculated using ML estimation in Equation 3.

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-2}^{i})}{c(w_{i-2}^{i-1})} \tag{3}$$

where $c(w_a^b)$ is the number of times that the sequence $w_a..w_b$ appears in preceding samples. The demand bandwidth in the next time interval is predicted to be the last value in the pattern with the highest probability.

Figure 7 shows the hardware components of our prediction engine, which includes a pattern look-up-table, a probability

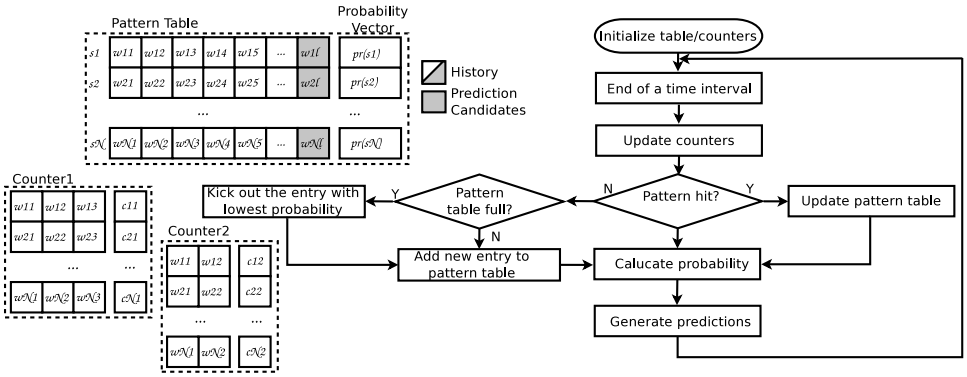


Fig. 7. Components of prediction engine.

vector, and a set of counters. The storage overhead...

The algorithm of the predictor is described in Figure **??**.

¡pseudocode¿

The computational complexity can be estimated as $O(ql)$, where q is the number of quantization bins.

VI. Experiments

1. nvsim memory bandwidth vs. way-configuration -¿ reasonable fast ways and slow ways for reconfiguration

2. predictor accuracy vs. benchmarks

3. performance of instruction throughput vs. benchmarks

Based on the parameters of different cache configurations collected from our modified version of CACTI [16], we evaluate both pure SRAM-based and hybrid cache hierarchy designs. In this section, we show how bandwidth-aware memory hierarchy design method help with improving the system performance.

A. Experimental Setup

We use Simics [22] as the simulator in our experiments. It is configured to model an eight-core CMP. Each core is in-order, and is similar to UltraSPARC III architecture. The frequency of each core is set to be 1GHz. Table I lists the detailed parameters of the baseline. Our baseline contains 8 private L1 instruction and data caches respectively. Since we only evaluate shared cache hierarchies, each of L1 cache is fixed to be SRAM-based, and of 64KB capacity. As regard to lower level caches, we evaluate two cases, pure SRAM-based and hybrid caches with various memory technologies, including SRAM, MRAM, RRAM, and eDRAM. By evaluating both cases, we would like to find out optimal cache design that leads to the peak performance, i.e., the number of cache levels that is reqired, memory technology used for each level, and capacity of each level.

The benchmarks are selected from PARSEC benchmark suite [23] with multithreaded programs, which focus on emerging workloads that are designed to be representative of next-generation shared-memory programs for CMPs. Since the performance of different memory technologies are closely related to read and write intensities, we selected some workloads that vary in the average numbers of L2 cache read per thousand instructions (RPKI) and write per thousand instructions (WPKI), which are listed in Table II.

TABLE I
Baseline CMP configuration.

Core	
No. of cores	8
Frequency	1GHz
Core architecture	in-order, 14-stage pipeline
Memory	
Private caches	L1-I/D caches: SRAM, 8 x 64KB, 64B line, 2-way, write-through
Shared caches	Case 1: Pure SRAM, 64B line, 8-way, write-back Case 2: Hybrid (SRAM, MRAM, RRAM, eDRAM), 64B line, 8-way, write-back
Main memory	4GB

TABLE II
Characteristics of selected benchmarks.

Benchmarks	RPKI	WPKI
blackscholes	22.8	61.7
bodytrack	5.4	139.5
canneal	5.4	29.3
facesim	6.0	102.4
ferret	5.7	173.4
fluidanimate	2.6	70.1
streamcluster	17.3	16.9
swaptions	2.6	121.4

B. Results

We evaluate various possible configurations with shared caches by simulation, i.e., with possible numbers of levels, memory technologies, and cache capacities for each cache level. We consider SRAM and eDRAM as the possible memory technologies to implement L2 and L3 caches. Both MRAM and RRAM have higher write latency than SRAM. Furthermore, the endurance of RRAM is too low to be

used as lower level caches. Consequently, these two memory technologies are only considered to be used as the last level cache.

In the first set of experiments, we evaluate the system performance with two-level shared caches. Figure 8 shows the system throughput with all the benchmarks. It is illustrated that implementing hybrid cache with eDRAM as the L3 cache helps with performance among most of the benchmarks. Such performance improvement is more than 10% in Figure 8(a). It is indicated by our memory technology exploration that eDRAM shows more latency benefits with larger capacities. As a result, a larger eDRAM-based L3 cache leads to more performance improvement to the pure SRAM implementation, as illustrated by Figure 8(b). However, hybrid cache does not always improve the performance, as shown in Figure 9. With the benchmark *canneal*, hybrid cache configurations outperform the pure SRAM implementation among overall range of various capacities. With the benchmark *ferret*, however, pure SRAM implementation results in higher performance than hybrid cache implementation with the same capacity configuration.

In the second set of experiments, we evaluate the system performance with three-level shared caches. When evaluating hybrid cache configurations, we consider implementing the last level cache by MRAM and RRAM memory technologies, since the data transaction intensity is relatively low in the last level cache. Figure 10 shows the results. Figure 10(a) illustrated that the performance more than 15% higher on average with a last level cache implemented by MRAM than pure SRAM implementations. More performance improvement is obtained by increasing the last level cache capacity, as shown in Figure 10(b). Figure 11 compares performance with pure SRAM and hybrid cache implementations with different capacities. In this case, hybrid cache implementation leads to higher performance in both benchmarks. The results of the two sets of experiments are illustrated together in Figure 12. An interesting observation is that hybrid cache implementation shows high performance improvement with one of the applications *canneal*, whereas as little improvement with the other *ferret*. The reason is that write latency of both MRAM and RRAM is higher than SRAM when capacity is less than 1GB. Performance of applications with large number of writes, such as *ferret* is therefore not benefit from hybrid cache design.

Finally, we evaluate all the possible cache hierarchy configurations with exhaustive simulations. The optimal cache configurations of each benchmark is shown in Table III. Since the characteristics vary among the benchmarks, the optimal cache configurations are different among each of benchmarks. Applications with high write intensities, such as *bodytrack*, *ferret*, and *swaptions* tend to favor SRAM-based caches. Other applications benefit more from hybrid cache implementation in terms of performance.

VII. Conclusion and Future Work

In this project, we explore performance and energy characteristics of various emerging memory technolgies. Based on our exploration, we evaluate the shared cache hierarchy design of CMPs optimized for performance by filling the off-chip memory bandwidth gap. According to our evaluation, SRAM-based caches leads to reasonable performance with write-intensive applications. Hybrid cache design help with performance with other types of benchmarks.

Furthermore, a variety of continuing studies is remained to be explored related to bandwidth-aware memory hierarchy design. First of all, we only examine cache hierarchy design currently. It is necessary to extend our exploration to the overall memory hierarchy design. On the other hand, we do not consider energy efficiency in this project. We would like to put some power constraints to the memory hierarchy, and examine the energy efficiency of various memory hierarchy configurations.

References

[1] S. A. McKee, "Reflections on the memory wall," in *Proceedings of the Conference on Computing Frontiers*, 2004, p. 162.

[2] D. Burger, J. R. Goodman, and A. Kägi, "Memory bandwidth limitations of future microprocessors," in *Proceedings of the International Symposium on Computer Architecture*, 1996, pp. 78–89.

[3] B. M. R. et al, "Scaling the bandwidth wall: challenges in and avenues for cmp scaling," in *Proceedings of the International Symposium on Computer Architecture*, 2009.

[4] J. Huh, D. Burger, and S. W. Keckler, "Exploring the design space of future CMPs," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, 2001, pp. 199–210.

[5] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "Nvidia tesla: a unified graphics and computing architecture," *IEEE Micro*, vol. 28, pp. 39–55, 2008.

[6] G. Sun, X. Wu, and Y. Xie, "Exploration of 3D stacked L2 cache design for high performance and efficient thermal control," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2009, pp. 295–298.

[7] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proceedings of the International Conference on High-Performance Computer Architecture*, 2009, pp. 239–249.

[8] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid cache architecture with disparate memory technologies," in *Proceedings of the International Symposium on Computer Architecture*, 2009, pp. 34–45.

[9] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2009, pp. 239–249.

[10] P. Zhou, V. Pandey, J. Sundaresan, A. Raghuraman, Y. Zhou, and S. Kumar, "Dynamic tracking of page miss ratio curve for memory management," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, 2004, pp. 177–188.

[11] E. Duesterwald, C. Cascaval, and S. Dwarkadas, "Characterizing and predicting program behavior and its variability," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, 2003, p. 220.

[12] R. Sarikaya and A. Buyuktosunoglu, "Predicting program behavior based on objective function minimization," in *Proceedings of the International Symposium on Workload Characterization*, 2007, pp. 25–34.

[13] R. Sarikaya, C. Isci, and A. Buyuktosunoglu, "Runtime workload behavior prediction using statistical metric modeling with application to dynamic power management," in *Proceedings of the International Symposium on Workload Characterization*, 2010, pp. 1–10.

[14] L. Chua, "Memristor: The missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[15] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, 2008.

[16] HP Labs, "CACTI, http://www.hpl.hp.com/research/cacti/," 2010.

[17] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li *et al.*, "Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement," in *Proceedings of the Design Automation Conference*, 2008, pp. 554–559.

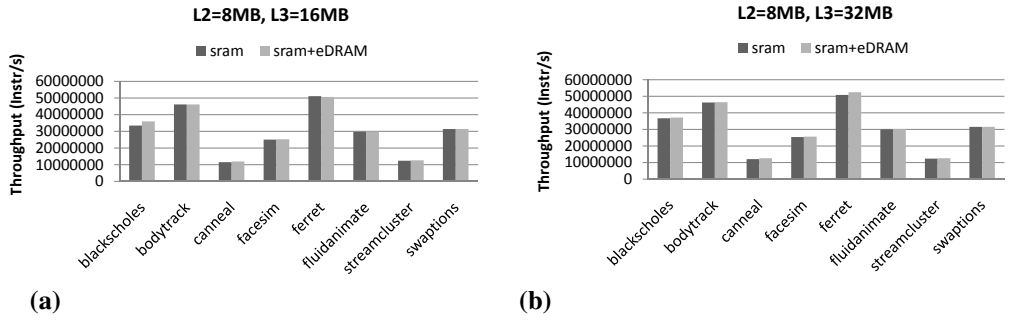


Fig. 8. Performance comparison with two-level caches among various benchmarks. (a) The L3 cache capacity is 16MB. (b) The L3 cache capacity is 32MB.

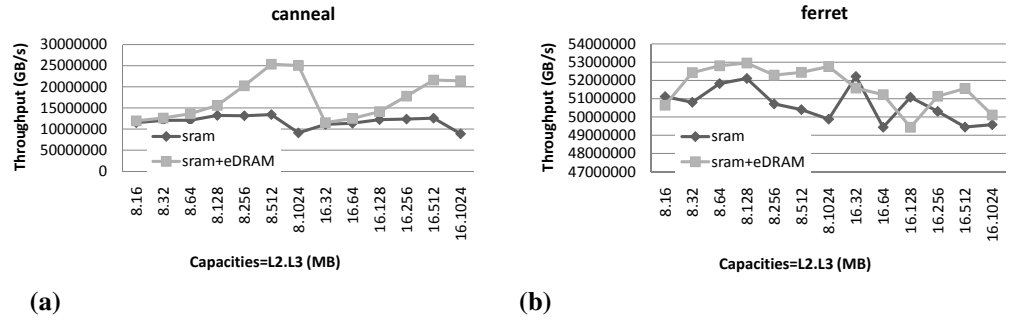


Fig. 9. Performance comparison with two-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark.

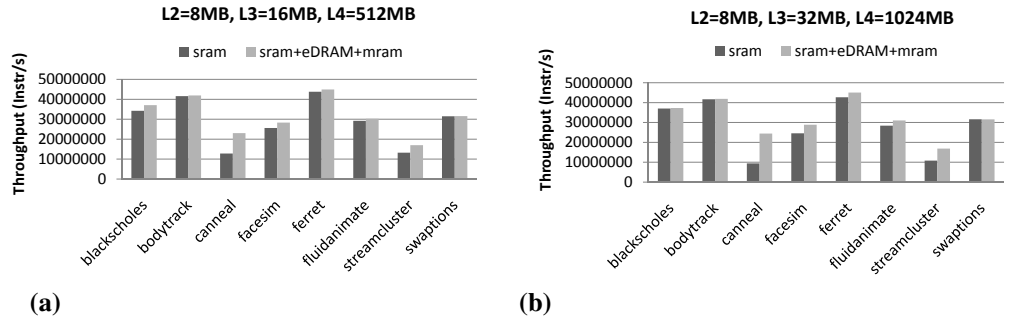


Fig. 10. Performance comparison with three-level caches among various benchmarks. (a) The L3 cache capacity is 16MB, and the L4 cache capacity is 512MB. (b) The L3 cache capacity is 32MB, and the L4 cache capacity is 1GB.

[18] P. Mangalagiri, K. Sarpatwari, A. Yanamandra, V. Narayanan, Y. Xie *et al.*, "A low-power phase change memory based hybrid cache architecture," in *Proceedings of the Great Lakes Symposium on VLSI*, 2008, pp. 395–3–98.

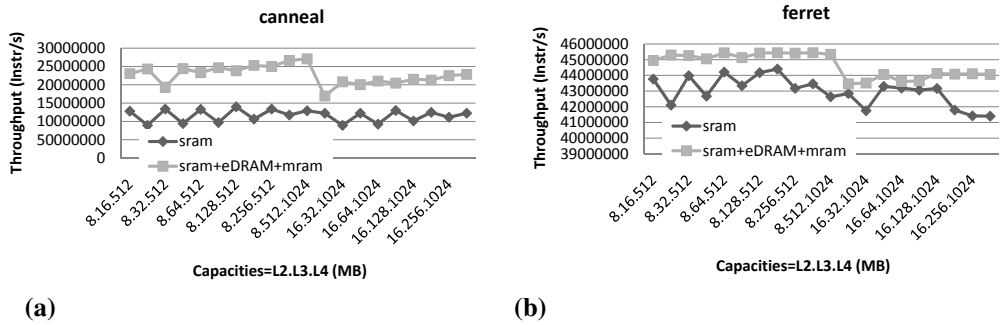
[19] X. Dong, N. P. Jouppi, and Y. Xie, "PCRAMsim: System-level performance, energy, and area modeling for phase-change RAM," in *Proceedings of the International Conference on Computer-Aided Design*, 2009, pp. 269–275.

[20] V. Mohan, S. Gurumurthi, and M. R. Stan, "FlashPower: A detailed power model for NAND flash memory," in *DATE*, 2010, pp. 502–507.

[21] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the Annual Meeting on Association for Computational Linguistics*, 1996, pp. 310–318.

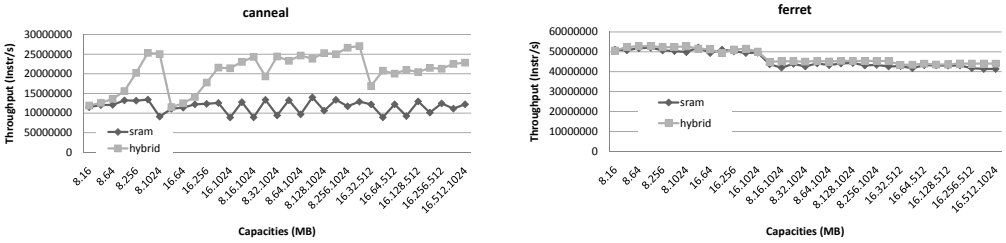
[22] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: a full system simulation platform," *IEEE Transactions on Computer*, vol. 35, no. 2, pp. 50–58, 2002.

[23] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, 2008, pp. 239–249.



(a) **(b)**

Fig. 11. Performance comparison with three-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark..



(a) **(b)**

Fig. 12. Performance comparison with both two-level and three-level caches among various cache capacity. (a) The canneal benchmark. (b) The ferret benchmark.

TABLE III
OPTIMAL CACHE CONFIGURATIONS FOR EACH BENCHMARK.

Benchmarks	**L2**	**L3**	**L4**
blackscholes	SRAM, 8MB	eDRAM, 32MB	RRAM, 1GB
bodytrack	SRAM, 8MB	eDRAM, 128MB	-
canneal	SRAM, 8MB	eDRAM, 512MB	MRAM, 1GB
facesim	SRAM, 8MB	eDRAM, 32MB	MRAM, 1GB
ferret	SRAM, 8MB	eDRAM, 32MB	-
fluidanimate	SRAM, 8MB	eDRAM, 1GB	-
streamcluster	SRAM, 16MB	eDRAM, 128MB	MRAM, 512MB
swaptions	SRAM, 8MB	SRAM, 32MB	SRAM, 1GB