A Circuit-Architecture Co-optimization Framework for Exploring Nonvolatile Memory Hierarchies

XIANGYU DONG, Qualcomm Technology, Inc.
NORMAN P. JOUPPI, Hewlett-Packard Labs
YUAN XIE, Pennsylvania State University & AMD Research

Many new memory technologies are available for building future energy-efficient memory hierarchies. It is necessary to have a framework that can quickly find the optimal memory technology at each hierarchy level. In this work, we first build a circuit-architecture joint design space exploration framework by combining RC circuit analysis and Artificial Neural Network (ANN)-based performance modeling. Then, we use this framework to evaluate some emerging nonvolatile memory hierarchies. We demonstrate that a Resistive RAM (ReRAM)-based cache hierarchy on an 8-core Chip-Multiprocessor (CMP) system can achieve a 24% Energy Delay Product (EDP) improvement and a 36% Energy Delay Area Product (EDAP) improvement compared to a conventional hierarchy with SRAM on-chip caches and DRAM main memory.

Categories and Subject Descriptors: B.3.3 [**Memory**]: Performance Analysis and Design Aids

General Terms: Memory, Cache, Nonvolatile, Design exploration

Additional Key Words and Phrases: SRAM, DRAM, ReRAM, STTRAM, PCRAM

ACM Reference Format:
Dong, X., Jouppi, N. P., and Xie, Y. 2013. A circuit-architecture co-optimization framework for exploring nonvolatile memory hierarchies. ACM Trans. Architec. Code Optim. 10, 4, Article 23 (December 2013), 22 pages.
DOI: http://dx.doi.org/10.1145/2541228.2541230

1. INTRODUCTION

The state-of-the-art memory hierarchy design with SRAM on-chip caches and DRAM off-chip main memory is now being challenged from two aspects. First, both SRAM and DRAM technologies are leaky. The SRAM leakage power and the DRAM refresh power will start to dominate if memory capacities keep growing. Some data already show that 25–40% of total power is attributed to the memory system [Udipi et al. 2010] and that some embedded processor caches can consume over 40% of the total chip power budget [Meng et al. 2005]. Second, SRAM and DRAM are facing many difficulties in scaling down. For example, it is hard to scale down DRAM below a 20nm process node due to the difficulty in keeping an adequate amount of cell capacitance [International Technology Roadmap for Semiconductors 2012]. The recent shift of some L3 on-chip caches from

---

Extension of Conference Paper: This submission is extended from "A Circuit-Architecture Co-optimization Framework for Evaluating Emerging Memory Hierarchies" published on ISPASS'13. The additional material provided in the submission includes a detailed explanation of the circuit-level and the architecture-level models, a new case study of using PCRAM, and a sensitivity study on processor core counts.
This work is supported in part by SRC grants, NSF 1218867, 1213052, 0903432 and by DoE under Award Number DE-SC0005026.
Author's addresses: X. Dong, Qualcomm Technology, Inc; email: xydong@cse.psu.edu.

DOI: http://dx.doi.org/10.1145/2541228.2541230

SRAM to eDRAM [Kalla et al. 2010] and the research momentum in replacing DRAM main memory with various emerging nonvolatile memories [Lee et al. 2009; Zhou et al. 2009; Qureshi et al. 2009a, 2009b; Seong et al. 2010; Schechter et al. 2010; Dong et al. 2008; Sun et al. 2009; Smullen et al. 2011] reflect the responses to such challenges in designing an energy-efficent and cost-effective memory hierarchy.

Recently, many alternative memory technologies, such as Phase-Change RAM (PCRAM)[1] [Lee et al. 2008; Sasago et al. 2009; De Sandre et al. 2010], Spin-Torque Transfer RAM (STTRAM)[2] [Kawahara et al. 2007; Tsuchida et al. 2010], and Resistive RAM (ReRAM)[3] [Chen et al. 2003; Kim et al. 2010; Sheu et al. 2011] have been demonstrated. These emerging nonvolatile memory technologies have attractive properties of high density, fast access, good scalability, and nonvolatility, and they have drawn the attention of the computer industry and challenged the role of SRAM and DRAM in the mainstream memory hierarchy for the first time in more than 30 years.

Since each of the emerging memory technologies has its pros and cons and the peripheral circuit design can affect the memory module properties greatly, future memory hierarchies will have a much larger design space. Therefore, it is necessary to have an estimation framework that can quickly find the optimal memory technology choice and the corresponding circuit design style in terms of performance, energy, or area (cost). But there are two challenges before doing that.

First, unlike SRAM, whose cells and macro designs are highly standardized, emerging memory technologies only have prototypes whose performance and energy properties can vary greatly. Such circuit variation can already be observed from the related literature [Lee et al. 2008; Sasago et al. 2009; De Sandre et al. 2010; Kawahara et al. 2007; Tsuchida et al. 2010; Chen et al. 2003; Kim et al. 2010; Sheu et al. 2011], where some of the memory prototypes show extremely fast access speed, whereas others show extremely dense structure. In order to model this variety and the circuit-level trade-offs, we build a circuit-level performance, energy, and area model.

Second, in order to build an optimization loop covering circuit- and architecture-level design options, we require models that reflect how architectural metrics (e.g., IPC and power consumption) change as we tune the underlying memory hierarchy design knobs (i.e., cache capacity, cache associativity, and cache read or write latency). Conventionally, such a model is built through simulations; however, it is impractical to run time-consuming simulations for each possible design input. To surmount this difficulty, we apply statistical analysis and effectively use limited simulation runs to approximate the entire architectural design space.

After modeling the circuit- and architecture-level trade-offs, we combine them into a circuit-architecture joint design space exploration framework and use this framework to optimize different memory hierarchy levels by adopting emerging memory technologies. In this work, we show that combined with SRAM L1 or L2 caches, the versatility of emerging memory technologies can excel in the remaining memory hierarchy levels from L2 or L3 caches to main memories, and that such a hybrid hierarchy has significant benefits in energy and area reduction with insignificant performance degradation overhead. As an example, our analysis shows that using ReRAM in L3 caches can achieve overall improvements in Energy Delay Product (EDP) (by 28%) and Energy Delay Area Product (EDAP) (by 39%) on an 8-core chip-multiprocessor (CMP) system. Finally, we propose a simulated annealing approach that can quickly find a near-optimal solution when designing an energy-efficient or cost-efficient memory hierarchy.

[1]Also called PCM or PRAM.
[2]Also called STT-MRAM or MRAM.
[3]Also called RRAM, CBRAM, or memristor.

2. RELATED WORK

Our work involves both circuit- and architecture-level models; thus, we first describe prior work on circuit-level memory design space exploration and predictive performance models.

2.1. Circuit Model for Memory Modules

Many circuit-level models have been provided to enable SRAM or DRAM design explorations. For example, CACTI [Wilton and Jouppi 1996; Thoziyoor et al. 2008b] is widely used to estimate the performance, energy, and area of SRAM and DRAM caches. However, as CACTI was originally designed to model an SRAM-based cache, some of its fundamental assumptions do not match actual emerging nonvolatile memory circuit implementations. Besides CACTI, Amrutur and Horowitz [2000] introduced an analytical model for estimating SRAM array speed and power scaling. Another circuit-level model [Azizi et al. 2010] uses a logic synthesis tool to build a circuit library and relies on curve fitting to represent circuit design trade-offs.

2.2. Predictive Performance Model

Statistical models [Joseph et al. 2006a, 2006b; Lee and Brooks 2006; Azizi et al. 2010; Ipek et al. 2008; Dubach et al. 2007] can be used to infer the impact of architectural input configurations on overall performance metrics. Although it is time-consuming to collect sufficient sample data from conventional simulations, this is a one-time effort, and all of the later outputs can be generated with the statistical model. Different fitting models have been used in the inference process that fits a predictive model through regression. Joseph et al. [2006a] used linear regression, and Lee and Brooks [2006] used cubic splines; however, Azizi et al. [2010] applied posynomial functions to create architecture-level models. The artificial neural network (ANN) [Ipek et al. 2008; Joseph et al. 2006b; Dubach et al. 2007] is another popular approach to build a predictive architecture model; it can efficiently explore exponential-size architectural design spaces with many interacting parameters.

3. CIRCUIT MODEL: AN RC APPROACH

Both SRAM and DRAM memory modules have their own typical design styles. For example, 6T or 8T SRAM cells are widely adopted in the on-chip cache designs, and 1T DRAM cells are also typical. Moreover, on-chip SRAM designs are mainly supported by standard libraries or even memory compiler tools, and commodity DRAM is highly standardized as well. Due to their technology maturity, both SRAM and DRAM memory modules now have less variety.

However, such design consistency cannot be found in the emerging memory module design. Recently, many STTRAM, PCRAM, and ReRAM prototype chips have been designed and demonstrated [Lee et al. 2008; Sasago et al. 2009; De Sandre et al. 2010; Kawahara et al. 2007; Tsuchida et al. 2010; Chen et al. 2003; Kim et al. 2010; Sheu et al. 2011], but few of them show consistency in reporting the performance, energy, and area data. This is actually common for any new technology. Due to the still evolving state of these emerging technologies, there is no single design standard design option can balance the trade-offs among chip performance, energy consumption, and chip area. Therefore, researchers have made various decisions on design and manufacturing, and thus it causes a large variation among designs.

Such variation brings challenges as well as opportunities for using emerging memory technologies in future memory hierarchies. The opportunity is that we can still freely bias the optimal design options toward different optimization targets on the different memory hierarchy levels, especially when large prototype chip variations tell us that

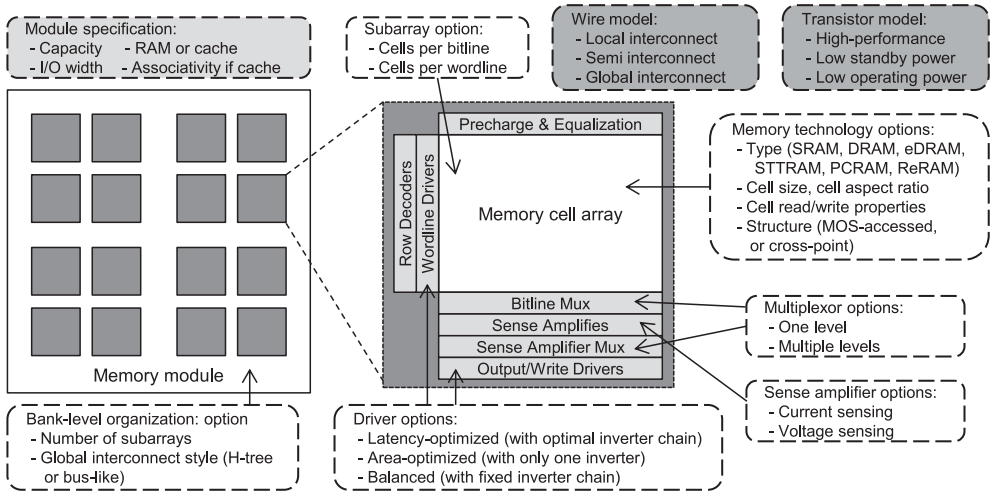

Fig. 1. The circuit-level model for memory module timing, power, and area estimations.

these technologies can cover a wide design spectrum from highly latency-optimized microprocessor caches to highly density-optimized secondary storage. But the challenge is to build a performance, energy, and area model for these emerging memory technologies even before they become mature. Therefore, we first build a circuit-level model for nonvolatile memory technologies.

3.1. Modeling Philosophy

We apply the modeling philosophy used in CACTI [Muralimanohar et al. 2008; Thoziyoor et al. 2008a] to establish a library of emerging memory technologies spanning from ultrafast to ultradense memory designs. Similar to CACTI, we capture the device-level RC property of memory cells and use traditional RC analysis to estimate their performance and energy consumption. We follow standard design rules to predict the silicon area occupied by each circuit component. We obtain the process-related data of transistors and metal layers from the ITRS report [International Technology Roadmap for Semiconductors 2012] and the MASTAR tool [International Technology Roadmap for Semiconductors 2011]. The data covers the process nodes from 22nm to 180nm and supports three transistor types: *High Performance*, *Low Operating Power*, and *Low Stand-by Power*.

3.2. Circuit Components and Tuning Knobs

Figure 1 shows the basic components abstracted in this circuit-level model. Each memory module is modeled as a set of banks, every bank can contain multiple subarrays, and a memory operation is fulfilled by simultaneous accesses to multiple subarrays in a bank. Depending on the design requirement, a bank can be partitioned into subarrays with different granularity. The rule of thumb is that smaller subarrays are faster and larger subarrays are more area efficient.

A subarray is the elementary structure, in which there are a set of peripheral circuits including row decoders, column multiplexers, output drivers, and so on. There are a large amount of design knobs that can be tuned in the subarray design, especially regarding the choice of peripheral circuits. For example, the output driver

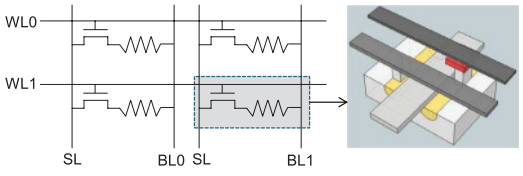

Fig. 2. Schematic view of MOS-accessed ReRAM arrays (WL=wordline; BL=bitline; SL=sourceline).

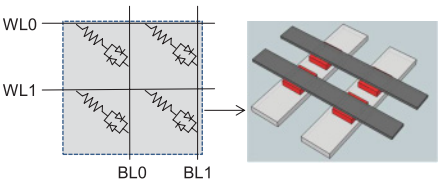

Fig. 3. Schematic view of cross-point ReRAM arrays without access devices (WL=wordline; BL=bitline).

design could follow logical effort [Sutherland et al. 1999] and use optimal levels and sizes of inverters for high performance, or it can be simply designed as a single inverter for area efficiency. For sense amplifiers, voltage sensing is straightforward but slower, whereas current sensing incurs two-level sensing but is much faster and more suitable for sensing the resistance difference of emerging memory cells. Other tuning knobs, such as the multiplexer design, are also shown in Figure 1.

3.3. Memory Array Structure

There are two types of memory arrays modeled in this work: MOS accessed and cross-point.

MOS-accessed cells correspond to the typical 1-transitor-1-resistor (1T1R) structure used by many nonvolatile memory prototype chips [Kawahara et al. 2007; Tsuchida et al. 2010], in which an NMOS access device is connected in series with the nonvolatile storage element (i.e., MTJ in STT-RAM, GST in PCRAM, and metal oxide in ReRAM), as shown in Figure 2. Such an NMOS device turns on/off the access path to the storage element by varying the voltage applied to its gate. The MOS-accessed cell usually has the best isolation between neighboring cells due to the high OFF resistance of the MOSFET. In MOS-accessed cells, the size of an access transistor is bounded by the current needed by the write operation. This NMOS device needs to be sufficiently large so that it can drive enough write current.

Cross-point cells correspond to the 1-diode-1-resistor (1D1R) [Zhang et al. 2007; Lee et al. 2008; Sasago et al. 2009; Lee et al. 2007] or the 0-transistor-1-resistor (0T1R) [Kau et al. 2009; Chen et al. 2003; Kim et al. 2010] structures used by several high-density nonvolatile memory chips. Figure 3 shows a cross-point array without diodes (i.e., 0T1R structure). For a 1D1R structure, a diode is inserted between the word line and the storage element. Such cells rely on the nonlinearity either by the introduction of a unipolar/bipolar diodes (i.e., 1D1R) or the cell's self-built-in characteristic (i.e., 0T1R) to control the memory access path.

Compared to MOS-accessed cells, cross-point cells have much smaller cell sizes. The area-efficiency benefit of the cross-point structure is evident in the comparison between Figure 2 and Figure 3. The removal of MOS access devices leads to a memory cell size of only $4F^2$, where F is the process feature size. Unfortunately, the cross-point structure worsens the isolation among memory cells and thus brings challenges to peripheral circuit designs. Several design issues such as half-select write, two-step sequential write, and external sensing [Xu et al. 2011] are included in our model.

3.4. Model Accuracy

We validate our circuit model against STTRAM [Tsuchida et al. 2010], PCRAM [Lee et al. 2008], and ReRAM [Sheu et al. 2011] prototypes. In general, the performance (i.e., read latency, write latency) estimation error is within 20%, and the area estimation error is below 10%.

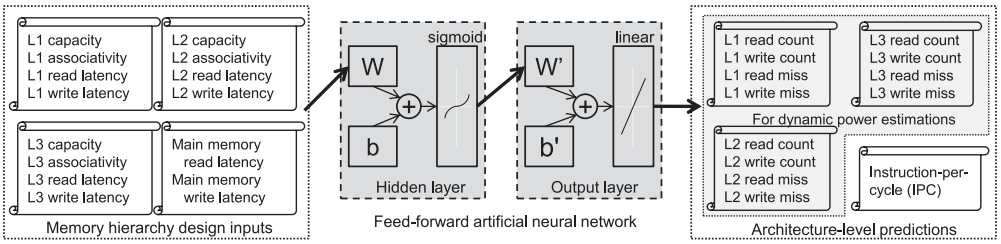

Fig. 4. The basic organization of a two-layer feed-forward ANN.

3.5. Circuit-Level Model Summary

In summary, our circuit-level model takes memory design parameters such as technology node, memory capacity, associativity, block size, and cell type as the inputs, and it gives circuit-level outputs such as read/write latency, read/write dynamic energy per access, leakage power, and silicon area. Our circuit-level model provides eight optimization targets, which are read latency, write latency, read energy, write energy, read EDP, write EDP, silicon area, and leakage power, and each of these optimized designs is evaluated in the later circuit-architecture joint design space exploration. The optimization is achieved by tuning the design knobs, including, but not limited to, subarray size, global interconnect style, driver design, sense amplifier design, multiplexer design, and memory cell structure.

4. ARCHITECTURE MODEL: AN ANN APPROACH

At the architectural level, we need performance models that predict the architectural performance of the overall system, such as IPC, and access counts at all cache levels as we change the underlying memory hierarchy. The input parameters at the architectural level are the parameters such as cache capacity, cache associativity, read latency, and write latency.

In a simulation-based approach, long run times are necessary to simulate each possible input setting, making it intractable to explore a large design space. However, simulation accuracy is not the first priority in such a large scale design space exploration. Instead, a speedy but less accurate architecture-level model is a preferred choice. In this work, since both our input space and output space are high dimensional, we select an ANN to fit the sampled simulation results into a predictive performance model.

4.1. Artificial Neural Network

Figure 4 shows a simplified diagram of a two-layer ANN with one sigmoid hidden layer (that uses sigmoid functions as the calculation kernel) and one linear output layer (that uses linear functions as the calculation kernel). The input and output design parameters are also shown in Figure 4. The essential architectural outputs for energy-performance-area evaluation are the read/write access counts and the read/write miss counts of every level of caches, read/write access counts of the main memory, and the number of instructions that each microprocessor core has processed. To feed the architectural model, the inputs of the architectural design space are the capacity, associativity, read/write latency of all cache modules, and the main memory, which can be generated from the aforementioned circuit-level model. The statistical architectural model makes an output estimate from given input sets, and it can be treated as a black

Table I. Input Design Space Parameters

Parameter	Range
Processor frequency	3.2GHz
Processor core	8-core, in-order
I-L1 (D-L1) capacity	8KB to 64KB
I-L1 (D-L1) associativity	4-way to 8-way
I-L1 (D-L1) read latency	2-cc to 40-cc
I-L1 (D-L1) write latency	2-cc to 700-cc
L2 capacity	64KB to 512KB
L2 associativity	8-way or 16-way
L2 read latency	5-cc to 80-cc
L2 write latency	5-cc to 800-cc
L3 capacity	512KB to 128MB
L3 associativity	8-way to 32-way
L3 read latency	20-cc to 100-cc
L3 write latency	20-cc to 900-cc
Memory read latency	30-cc to 300-cc
Memory write latency	30-cc to 1,000-cc

box that generates predicted outputs as a function of the inputs,

$$\mathrm{L1_{readCount}} = f_1(\mathrm{L1_{capacity}}, \mathrm{L1_{assoc}}, \mathrm{L1_{readLatency}}, \ldots, \mathrm{L3_{capacity}}, \ldots, \mathrm{Mem_{writeLatency}}) \quad (1)$$

$$\ldots = \ldots$$

$$\mathrm{L3_{writeMiss}} = f_{n-1}(\mathrm{L1_{capacity}}, \mathrm{L1_{assoc}}, \mathrm{L1_{readLatency}}, \ldots, \mathrm{L3_{capacity}}, \ldots, \mathrm{Mem_{writeLatency}}) \quad (2)$$

$$\mathrm{IPC} = f_n(\mathrm{L1_{capacity}}, \mathrm{L1_{assoc}}, \mathrm{L1_{readLatency}}, \ldots, \mathrm{L3_{capacity}}, \ldots, \mathrm{Mem_{writeLatency}}) \quad (3)$$

In our model, the input dimension is 14 (vector I_{14}), and the output dimension is 13 (vector O_{13}). The number of neurons in the hidden layer (X) is S, which ranges from 30 to 60 depending on different fitting targets. In Figure 4, W and b are the weight matrix and bias vector of the hidden layer; W′ and b′ are those of the output layer. The feed-forward ANN is calculated as follows,

$$\mathrm{X}_S = \sigma(\mathrm{W}_{S\times 14}\mathrm{I}_{14} + \mathrm{b}_S) \quad (4)$$

$$\mathrm{O}_{13} = \psi(\mathrm{W'}_{13\times S}\mathrm{X}_S + \mathrm{b'}_{13}) \quad (5)$$

where $\sigma(\cdot)$ and $\psi(\cdot)$ are sigmoid and linear functions.

4.2. Sample Collection

In this work, we collect samples to evaluate an 8-core CMP.[4] Each core is configured to be a scaled 32nm in-order SPARC-V9–like processor core with a 3.2GHz frequency. A private L1 instruction cache (I-L1), an L1 data cache (D-L1), and a unified L2 cache (L2) are associated with each core. Eight cores together share an on-die L3 cache. We randomly generate architecture inputs from the range listed in Table I and feed each input to a full-system simulator to get the output. Every input and output pair becomes a sample later used in the ANN training.

We use NAS Parallel Benchmarks (NPB) [NASA Advanced Supercomputing (NAS) Division [2012] and PARSEC [Bienia et al. 2008] as the experimental workloads. The workload size of the NPB benchmark is CLASS-C (except DC has no CLASS-C setting,

[4]We also use the same methodology to collect the data for a 16-core CMP performance model, and the result is shown in Section 6.4.

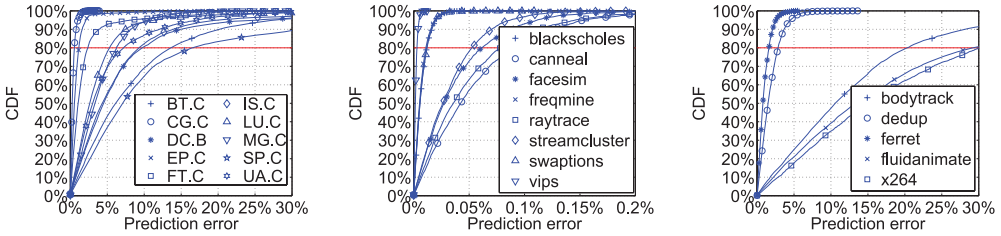

Fig. 5. CDF plots of error on IPC prediction of NPB and PARSEC benchmark applications. The x-axis shows the prediction error; the y-axis shows the percentage of data points that achieve the prediction error less than each x value.

and CLASS-B is used instead), and the native inputs are used for the PARSEC benchmark to generate realistic program behavior. In total, 23 benchmark applications are evaluated, and we build 23 separate ANN models for the 8-core CMP architecture-level model.[5] Later, all the experimental results are based on the average value of these 23 workloads. We randomly pick design configurations per benchmark and use the Simics full-system simulator [Magnusson et al. 2002] to collect sample data. Each Simics simulation is fast forwarded to the predefined breakpoint at the code region of interest, warmed up by 1 billion instructions and then simulated in the detailed timing mode for 10 billion cycles.

4.3. Training and Validation

An ANN is able to fit multidimensional mapping problems given consistent data and enough neurons in the hidden layer. The accuracy of the statistical architectural model depends on the number of training samples provided from actual full-system simulations. In this work, 3,000 cycle-accurate full-system simulation results are collected for each workload. Among each set of 3,000 samples, 2,400 data samples are used for training, 300 are used for testing, and the other 300 are used for validation during the training procedure to prevent overtraining [Sarle 1995]. To reduce variability, multiple rounds of cross-validation, during which data are rotated among the training,testing, and validation sets, are performed using different partitions, and the validation results are averaged over the rounds. Every ANN is configured to have 30 to 60 hidden neurons and trained using the Levenberg-Marquardt algorithm [Marquardt 1963]. The Levenberg-Marquardt algorithm trains the ANN by adjusting the weight matrices and bias vectors based on the data iteratively until the ANN accurately predicts the outputs from the input parameters.

Figure 5 illustrates the IPC prediction errors of the architecture-level performance model after training. The x-axis shows the relative error between the predicted and the actual values, and the y-axis presents the cumulative distribution function. In Figure 5 (middle), we can find that eight benchmarks in PARSEC have a probability of 80% to achieve an IPC prediction error of only 0.1%, and the probability of achieving IPC prediction errors of less than 0.2% is very close to 100%. Section 4.4 shows the detailed data on the ANN model accuracy.

4.4. Model Accuracy

To measure the model accuracy, we use the metric $error = |predicted - actual|/actual$. As we later directly use the ANN model to estimate the system performance, it is very critical to understand the accuracy of the ANN model IPC estimation. Figures 6

[5]Another 23 ANN models are built for the 16-core CMP model.

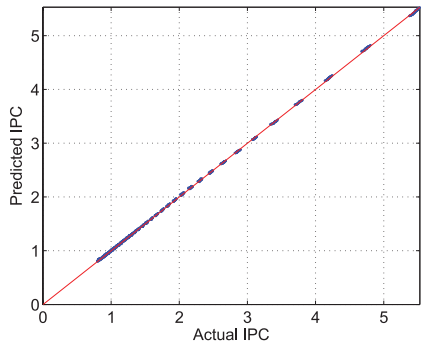

Fig. 6. An accurate IPC estimation example: vips from PARSEC.

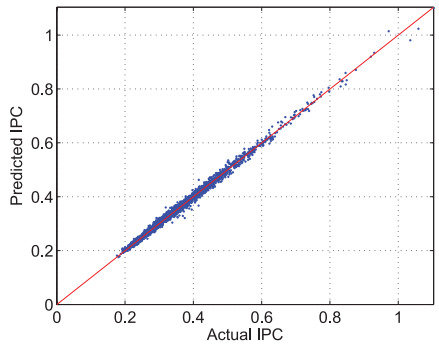

Fig. 7. A typical IPC estimation example: dedup from PARSEC.

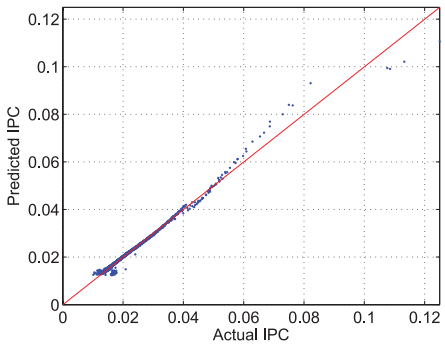

Fig. 8. Another typical IPC estimation example: FT from NPB.

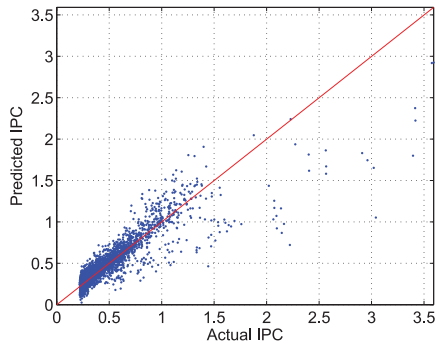

Fig. 9. The worst IPC estimation example: x264 from PARSEC.

through 9 show four examples of the IPC estimation result: a very accurate fit (0.15% error), two typical fits (3.06% error for *dedup* and 3.52% error for *FT*), and the worst fit (18.71% error) in this work. The average IPC estimation error is 4.29%.

Besides the performance metric, the estimation of memory system activity counts (e.g., L1 read count, L1 write count, L2 read count, L2 write count, etc.) are also important to us because we rely on them to get the memory system power estimation. In order to demonstrate that the ANN is also accurate for these metrics, we plot Figures 10 through 13 showing the difference between the simulated L2 read count and the predicted L2 read count. The result shows that the ANN is very accurate for these metrics as well.

5. JOINT DESIGN SPACE EXPLORATION FRAMEWORK

In this section, we describe how the circuit- and the architecture-level models are combined into a joint design space exploration framework.

5.1. Framework Overview

Figure 14 shows an overview of this joint circuit-architecture exploration framework. As mentioned, 3,000 randomly generated architecture-level inputs per benchmark

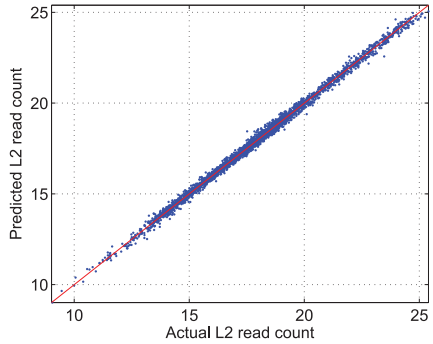

Fig. 10. An L2 read count estimation example: vips from PARSEC.

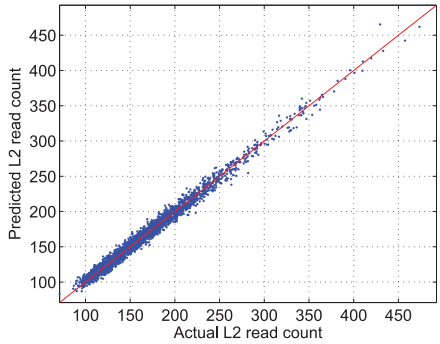

Fig. 11. An L2 read count estimation example: dedup from PARSEC.

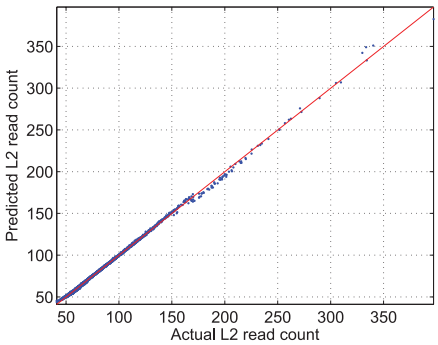

Fig. 12. An L2 read count estimation example: FT from NPB.

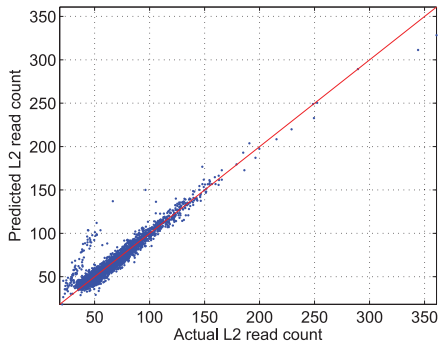

Fig. 13. An L2 read count estimation example: x264 from PARSEC.

workload are used to produce 3,000 corresponding samples in the architectural design space. The samples are then fed into the ANN trainer to establish the architecture-level performance model for each benchmark workload. The trained ANN is used as the architecture-level performance model. The circuit-level inputs are first passed through the memory module performance, energy, and area model, and then fed into the ANN-based architecture-level performance model to generate the predicted architecture-level results, such as IPC and power consumption, together with the silicon area estimates. When the predicted result does not meet the design requirement, feedback information containing the distance between the design optimization target and the current achieved result is sent to a simulated annealing [Kirkpatrick et al. 1983] optimization engine, and a new design trial is generated for the optimization loop. This optimization procedure steps forward iteratively until the design requirement (e.g., best EDP or best EDAP) is achieved or a near-optimal solution is reached. We use a simulated annealing engine to conduct this optimization step, and this is described in Section 7 in detail.

5.2. Circuit-Architecture Combination

After obtaining the access activities of each cache level, the memory subsystem power consumption can be calculated. Because the dynamic energy consumption of main memory is proportional to the last-level cache miss rate, we include it as a part of the

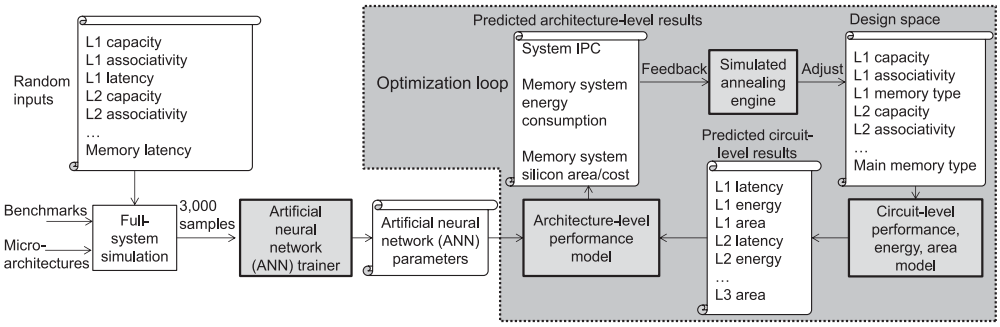

Fig. 14. Overview of the optimization framework. Architecture-level models are generated using sampling and an ANN trainer. Circuit-level models are used to estimate the latency, energy, and area of each memory module in the hierarchy. A simulated annealing engine is applied to find the near-optimal solution without exhaustive search.

memory subsystem power consumption for a fair comparison. The power consumption of logic components, including the processor cores, on-chip memory controller, and intercore crossbar, are estimated by McPAT [Li et al. 2009]. We use a 32nm technology in the McPAT simulation.

From McPAT, the logic components have 7.41W leakage power ($\mathrm{P}_{\mathrm{logic,leakage}}$) and 10.98W peak dynamic power. The runtime dynamic power consumption ($\mathrm{P}_{\mathrm{logic,dynamic}}$) is scaled down from the peak dynamic power according to the actual IPC value.[6] The total power consumption of the processor chip is calculated as follows:

$$\begin{aligned}\mathrm{E}_{\mathrm{memory,dynamic}} &= \sum_{i=1}^{3}\left[N_{\mathrm{readHit}_i}E_{\mathrm{hit}_i} + N_{\mathrm{readMiss}_i}E_{\mathrm{miss}_i} + \left(N_{\mathrm{writeHit}_i} + N_{\mathrm{writeMiss}_i}\right)E_{\mathrm{write}_i}\right] \\ &\quad + N_{\mathrm{readMiss}_3}E_{\mathrm{read}_4} + N_{\mathrm{writeMiss}_3}E_{\mathrm{write}_4} \qquad (6)\end{aligned}$$

$$\mathrm{P}_{\mathrm{memory,leakage}} = 2N_{\mathrm{core}}P_1 + N_{\mathrm{core}}P_2 + P_3 \qquad (7)$$

$$\mathrm{P}_{\mathrm{processor,total}} = \mathrm{E}_{\mathrm{memory,dynamic}}/T + \mathrm{P}_{\mathrm{logic,dynamic}} + \mathrm{P}_{\mathrm{memory,leakage}} + \mathrm{P}_{\mathrm{logic,leakage}} \qquad (8)$$

In Eq. (6), N_{readHit_i}, N_{readMiss_i}, N_{writeHit_i}, and $N_{\mathrm{writeMiss}_i}$ are the read count, read miss count, write count, and write miss count of the level-i cache, which are generated from the ANN-based architecture-level model. E_{hit_i}, E_{miss_i}, and E_{write_i} are the dynamic energy consumption of a hit, miss, and write operation in the level-i cache, and they are obtained from the RC-based circuit-level model. E_{read_4} and E_{read_4} are the dynamic energy consumption of main memory read and write operations, since we label the main memory as the fourth level of the memory hierarchy. In Eq. (7), N_{core} is the number of cores, and P_i represents the leakage power consumption of each cache level. The coefficient 2 is because of the identical data and instruction L1 caches (D-L1 and I-L1) in this work. Eq. (8) gives the total power consumption where T is the simulation time ($T = 10\mathrm{B}/3.2\mathrm{GHz} = 3.125\mathrm{s}$ according to our experimental setup).

6. DESIGN EXPLORATION: A RERAM CASE STUDY

In this section, we demonstrate how to perform a circuit-architecture joint memory hierarchy design space exploration by adopting emerging ReRAM technology.

[6]We use an empirical scaling model, $\mathrm{P}_{\mathrm{actual}} = (A + (1-A)\cdot \mathrm{IPC}_{\mathrm{actual}}/\mathrm{IPC}_{\mathrm{peak}})\cdot \mathrm{P}_{\mathrm{peak}}$.

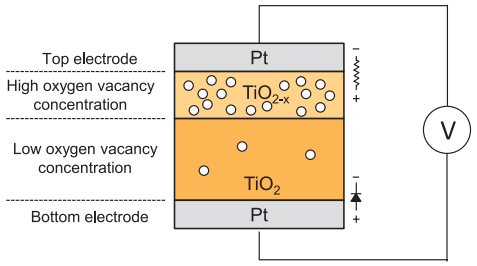

Fig. 15. A ReRAM cell example that uses Pt and Ti.

Table II. ReRAM Technology Assumptions

	ReRAM	
	MOS-accessed	Cross-point
Cell size	$20F^2$	$4F^2$
Write pulse duration	1 pulse 50ns per pulse	2 pulses 50ns per pulse
State-0 resistance	10 kΩ	
State-1 resistance	500kΩ	
Half-select resistance	-	100 kΩ
Write endurance	10^{12}	

6.1. ReRAM Technology

ReRAM is an emerging nonvolatile memory technology that involves electro- and thermochemical effects in the resistance change of a metal-oxide-metal system.[7] A ReRAM cell consists of a metal oxide layer sandwiched between two metal electrodes as shown in Figure 15. The electronic behavior of metal/oxide interfaces depends on the oxygen vacancy concentration of the metal oxide layer. Typically, the metal/oxide interface shows Ohmic behavior in the case of very high doping and rectifying in the case of low doping [Yang et al. 2008]. In Figure 15, the TiO_x region is semi-insulating, indicating lower oxygen vacancy concentration, whereas the TiO_{2-x} is conductive, indicating higher concentration.

As an example, we use the ReRAM device parameters shown in Table II and explore the circuit-level design space at first. Figures 16 and 17 demonstrate the design spectrum of emerging ReRAM technology. For comparison, the design spectrum of SRAM and DRAM is also shown. Note that MOS-accessed ReRAM and cross-point ReRAM are more than 10 times denser than SRAM, and cross-point ReRAM can be as dense as DRAM. In terms of speed, ReRAM has comparable read speed to that of SRAM, but significantly slower write speed. The write latency of MOS-accessed ReRAM is dominated by the switching pulse duration, which is 50ns in our experiments, and the latency of cross-point ReRAM is twice this due to two-step writes.

6.2. Wear-Leveling Assumption

Similar to NAND flash, ReRAM has limited write endurance (i.e., the number of times that a ReRAM cell can be overwritten). Many techniques [Zhou et al. 2009; Qureshi et al. 2009b; Schechter et al. 2010] have been developed to extend the lifetime of PCRAM-based main memories, and they can be borrowed for ReRAM wear leveling. Recently, i^2WAP [Wang et al. 2013], a wear-leveling scheme for nonvolatile caches, was proposed to mitigate both the cache interset and intraset write count variation.

[7]There are other models explaining the ReRAM working mechanism.

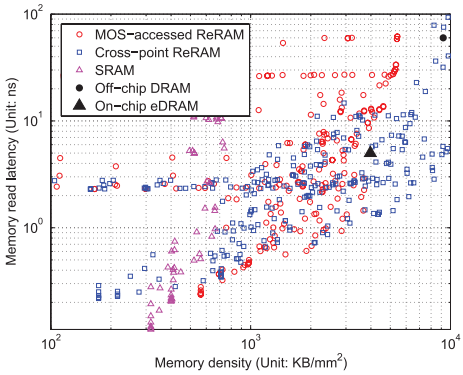

Fig. 16. The design spectrum of 32nm ReRAM: read latency versus density.

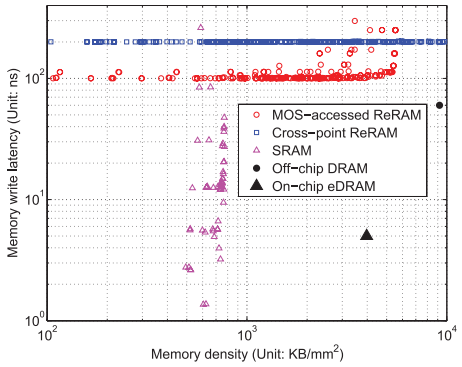

Fig. 17. The design spectrum of 32nm ReRAM: write latency versus density.

Table III. MOS-Accessed and Cross-Point ReRAM Main Memory Parameters (1Gb, 8-bit, 16-bank) Obtained from the Circuit-Level Model

	MOS-accessed	Cross-point
Die area	129mm^2	48mm^2
Read latency	6.2ns	10.0ns
Write latency	54.9ns	107.1ns
Burst read latency	4.3ns	4.3ns
Burst write latency	4.3ns	4.3ns

i^{2}WAP can evenly distribute cache write accesses among cache sets using a one-layer address remapping. The remapping information is stored in two global registers. In addition, i^{2}WAP handles the write unbalance inside a cache set (i.e., among different cache ways) by slightly changing the cache replacement policy without hurting performance. According to i^{2}WAP, we only need to add two global counters and two global registers for the cache wear-leveling hardware. In this case study, we assume that future nonvolatile caches will use i^{2}WAP and can achieve a low-variance wear leveling. As a result, we consider the current ReRAM write endurance (10^{10}–10^{12} [Sheu et al. 2011; Kim et al. 2011; Eshraghian et al. 2010]) high enough for L2 and L3 cache applications. In all the later experiments, we include the wear-leveling performance overhead by conservatively adding 2ns on top of the access latency obtained from our circuit-level model.

6.3. Memory Hierarchy Design Exploration

We next use the circuit-architecture joint-space design space exploration framework to analyze the energy versus performance trade-off of adopting ReRAM-based caches. In this step, we separate the cache design space (L1, L2, and L3) and the memory design space. We assume that the main memory is built by either cross-point ReRAMs that are optimized for density or MOS-accessed ReRAMs that are optimized for latency. Table III lists the timing and area parameters of both MOS-accessed and cross-point ReRAM main memory solutions.

Since we use a trained ANN model for 8-core CMP microprocessors as described in Section 4 and the random training inputs whose range are listed in Table I, we can use the circuit-architecture joint-space design space exploration framework to permute all the possible cache hierarchy configurations, and we list this permutation in Table IV.

Table IV. Cache Hierarchy Design Space

Parameter	Range
Processor frequency	3.2GHz
Processor core	8-core, in-order
I-L1 (D-L1) memory type	SRAM or ReRAM
I-L1 (D-L1) capacity	8KB, 16KB, 32KB, or 64KB
I-L1 (D-L1) associativity	4-way or 8-way
I-L1 (D-L1) read latency I-L1 (D-L1) write latency	Obtained from the circuit-level model (based on memory type, capacity, and associativity)
L2 memory type	SRAM or ReRAM
L2 capacity	64KB, 128KB, 256KB, or 512KB
L2 associativity	8-way or 16-way
L2 read latency L2 write latency	Obtained from the circuit-level model (based on memory type, capacity, and associativity)
L3 memory type	SRAM or ReRAM
L3 capacity	4MB, 8MB, 16MB, 32MB, 64MB, or 128MB
L3 associativity	8-way, 16-way, or 32-way
L3 read latency L3 write latency	Obtained from the circuit-level model (based on memory type, capacity, and associativity)

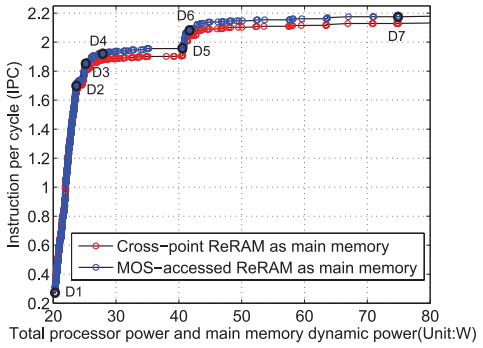

Fig. 18. Pareto curves: energy and performance trade-off of the memory hierarchy. Main memory dynamic power is included for a fair comparison.

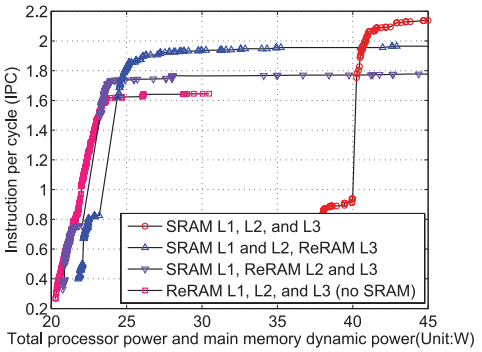

Fig. 19. Pareto curves (MOS-accessed ReRAM as main memory): energy and performance trade-off under different constraints on ReRAM deployment.

Focusing on the design space exploration of ReRAM-based memory hierarchies, Figure 18 shows the Pareto-optimal curves of the energy-performance trade-off in an 8-core CMP setting. The x-axis is the total power consumption of the processor chip, and the y-axis is the IPC performance. It can be observed from Figure 18 that a great amount of power consumption can be reduced by only incurring a small amount of performance degradation. For instance, as shown in Figure 18, design option D4 (using SRAM L1 and L2 caches but an ReRAM L3 cache) reaches 1.92 IPC by consuming 27.26W total power. Compared to design option D7 (using an SRAM-only cache hierarchy) that reaches 2.18 IPC but consumes 74.92W power, the achieved power reduction is 64% but the performance degradation is only 12%. This design option also meets the constraint set by 10^{10} write endurance as discussed in Section 6.2. If the ReRAM write endurance is assumed to be 10^{12}, more aggressive options (e.g., using L2 ReRAM caches) can further reduce the power consumption. For example, design option D3 (using ReRAM L2 and L3 caches) reaches 1.85 IPC by consuming only 25.34W

Table V. On-Die Cache Hierarchy Design Parameters of Seven Design Options

	D1	D2	D3	D4	D5	D6	D7
L1 capacity	64KB	8KB	8KB	8KB	32KB	8KB	32KB
L1 associativity	4	8	8	4	4	4	4
L1 memory type	M-ReRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
L1 optimized for	L	WP	RL	WP	RL	RP	RL
L1 sensing scheme	EX	IN	IN	IN	IN	IN	IN
L2 capacity	128KB	64KB	512KB	64KB	256KB	64KB	1024KB
L2 associativity	8	16	8	16	8	8	8
L2 memory type	M-ReRAM	M-ReRAM	M-ReRAM	SRAM	SRAM	SRAM	SRAM
L2 optimized for	L	L	L	L	WE	WE	RE
L2 sensing scheme	IN	IN	IN	IN	IN	IN	IN
L3 capacity	8MB	16MB	8MB	8MB	128MB	8MB	8MB
L3 associativity	16	16	8	8	8	8	8
L3 memory type	M-ReRAM	M-ReRAM	M-ReRAM	M-ReRAM	X-ReRAM	SRAM	SRAM
L3 optimized for	L	L	L	L	RE	L	WP
L3 sensing scheme	IN	IN	EX	IN	EX	IN	IN
IPC	0.26	1.70	1.85	1.92	1.96	2.09	2.18
Power consumption (W)	20.28	23.74	25.34	27.26	40.51	41.59	74.92
Silicon area (mm^2)	47.48	49.07	48.20	54.19	83.33	58.09	86.83

*Memory type abbreviations: M-ReRAM = MOS-accessed ReRAM; X-ReRAM = cross-point ReRAM. Optimization abbreviations: RL = Read Latency; WL = Write Latency; RE = Read Energy; WE = Write Energy; RP = Read EDP; WP = Write EDP; L = Leakage; A = Area. Sensing scheme abbreviations: IN = Internal; EX = External.

total power. To show how different cache hierarchy designs have been explored, we list the design parameters of seven example design options (D1 to D7) in Table V.

We find that the Pareto-optimal curves are composed of several segments, such as D1-to-D2, D4-to-D5, etc. The joints between every two segments represent the place where SRAM/ReRAM replacement occurs. Such replacements can be found in Figure 19. In general, IPC improvements are achieved by adding more SRAM resources, and greater reductions in power consumption come from replacing SRAM with ReRAM. Figure 19 shows that a ReRAM-only cache hierarchy is on the global Pareto-front, but the corresponding IPC is less than 1.6, and that segment has a large slope. Thus, it suggests that we should still deploy SRAM L1 caches for performance. However, starting from L2, ReRAM cache deployment can save considerable amounts of power and only sacrifice a small amount of performance. This is especially true for a hybrid on-chip cache hierarchy with SRAM L1/L2 caches and an ReRAM L3 cache. Figure 19 shows that in this region the total power consumption can be lowered to 27.26W but the IPC is only degraded from 2.18 to 1.92 (i.e., design option D4 in Figure 18).

Another benefit that we can get from introducing ReRAM caches is in silicon area reduction. Figure 20 shows the Pareto-optimal curves of area-performance trade-offs, which have similar shapes to the ones in the power-performance trade-off as shown in Figure 20. The processor core area (including memory controller and crossbar) is $45.6mm^2$ from an McPAT [Li et al. 2009] estimation. Achieving the highest performance using pure-SRAM caches costs at least another $12mm^2$ of silicon area, whereas replacing the SRAM L3 cache with ReRAM can save more than than $7mm^2$ in chip area by degrading performance from an IPC of 2.18 to 1.90. We show the feasible region of designs with less than $50mm^2$ total cache area in Figure 21. This result can be extremely useful in some low-cost computing segments where the performance requirement is just-in-time but the chip cost has the first priority. Figure 21 also indicates that using ReRAM caches can reduce power consumption and silicon area at the same time, further improving EDAP.

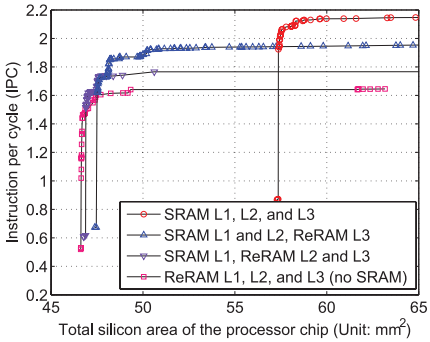

Fig. 20. Pareto curves (MOS-accessed ReRAM as main memory): cache area and performance trade-off under different ReRAM deployments.

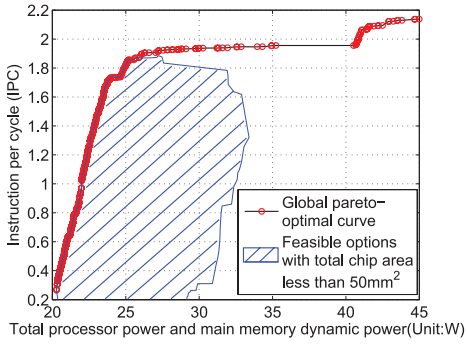

Fig. 21. The global Pareto-optimal curve (MOS-accessed ReRAM as main memory) and feasible design options with total chip area less than 50mm^2.

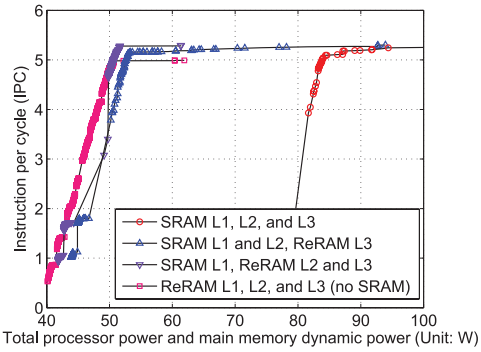

Fig. 22. Pareto curves after scaling up to 16-core: energy and performance trade-off under different constraints on ReRAM deployment.

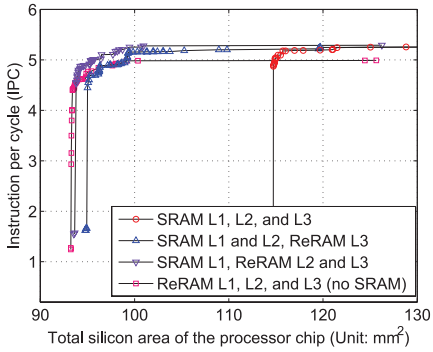

Fig. 23. Pareto curves after scaling up to 16-core: area and performance trade-off under different constraints on ReRAM deployment.

6.4. Architecture Model Change: Scaling to 16-Core

Although all of the previously demonstrated results are based on an 8-core CMP design, it is straightforward to use the same methodology on other analysis targets. For example, if we want to scale the number of cores from 8 to 16, we just need to retrain the ANN-based architecture model to fit the 16-core simulation results.

To prove this point, we collect another set of Simics full-system simulation results on PARSEC and NPB benchmarks, retrain the ANN models, and replot the Pareto curves. Figure 22 shows the new energy-performance trade-off, and Figure 23 shows the new area-performance trade-off. These new simulation results also give us some new observations:

—The SRAM-only options no longer provide the highest performance. This is because doubling the number of cores implies a larger L3 cache capacity. However, as the L3 cache capacity reaches a certain threshold, the interconnect latency starts to dominate the SRAM-based cache access latency. As a result, switching to ReRAM L3 cache becomes beneficial because it provides significant area savings and hence improves the cache access latency.

Table VI. PCRAM Technology Assumptions

Cell size	$36F^2$
Reset pulse duration	100ns
Set pulse duration	300ns
State-0 resistance	$5k\Omega$
State-1 resistance	$500k\Omega$

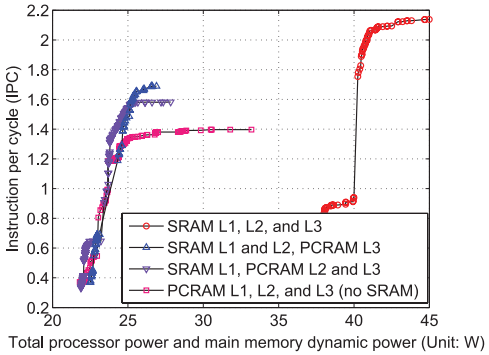

Fig. 24. Pareto curves after switching to PCRAM: energy and performance trade-offs under different constraints on PCRAM deployment.

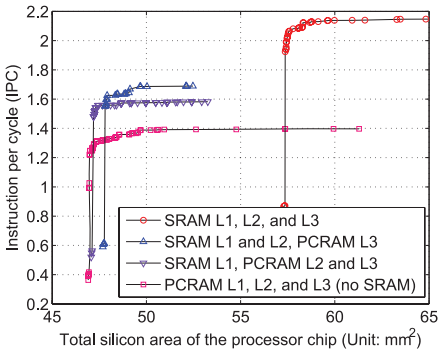

Fig. 25. Pareto curves after switching to PCRAM: area and performance trade-offs under different constraints on PCRAM deployment.

—The performance gap among the four Pareto curves is much smaller compared to the previous 8-core results. This is because the performance of a many-core system depends more on application-level parallelism instead of the horsepower from each core. This observation supports the current trend of building many-core in-order microprocessors to achieve a higher energy efficiency.

6.5. Circuit Model Change: Switching to Other Memory Technologies

As a general purpose tool, our circuit-architecture joint-space exploration framework is not limited to only the case studies for ReRAM technology. Using the same circuit-level model but with a new technology input, we can build another circuit library for PCRAM-based cache modules. Table VI lists the PCM technology assumptions that we use as a new input to the circuit model.

By replacing the previous ReRAM circuit library with a newly generated PCRAM circuit library, we can quickly have an overview that reveals the performance-power-area trade-offs of using PCRAM in different memory hierarchies. Figures 24 and 25 demonstrate how the framework can be easily adapted for the design space exploration of an 8-core PCRAM-based cache hierarchy in terms of the energy-performance trade-off and area-performance trade-off, respectively. Comparing to the previous ReRAM-based results, we find that:

—The Pareto-optimal curve of PCRAM-based cache hierarchies is much shorter. This is because our PCRAM technology input has a very poor RESET/SET setting (100ns and 300ns, respectively), and this causes most of the design points to fall in non-Pareto-optimal regions.
—The PCRAM-based cache hierarchies cause a much larger performance degradation, and IPC drops from 2.2 to 1.4 (ReRAM-based only causes an IPC drop from 2.2 to 1.6). This is again because of relatively worse PCRAM technology parameters.

7. DESIGN OPTIMIZATION

Running full design space exploration using an exhaustive search is time-consuming and may not be necessary in most cases. Although it is possible for designers to use educated guesses to refine the design space before an exhaustive search, the remaining design space might still be too gigantic. Using Figure 21 as an example, even if we add a 50mm^2 area constraint and limit the L3 cache capacity, there are still 1,662,601 feasible configurations in the shaded region of Figure 21 to explore. Therefore, to use this joint circuit-architecture model as a practical memory hierarchy design assistant, an efficient optimization method is required. In this work, we use a simplified simulated annealing [Kirkpatrick et al. 1983] algorithm to find a locally optimal solution. The simulated annealing heuristic is described in Algorithm 1.

ALGORITHM 1: Design Space Optimization Algorithm

```
state = s0, energy = E(state)
repeat
  new_state = neighbour(state), new_energy = E(new_state)
  if new_energy < energy then
    state = new_state, energy = new_energy {Accept unconditionally}
  else if T(energy, new_energy) > random() then
    state = new_state, energy = new_energy {Accept with probability}
  end if
until energy stops improving in the last K rounds
return state
```

In this optimization methodology, we first randomly choose an initial design option, $s0$, and calculate its annealing energy function from the joint circuit-architecture model. The annealing energy function can be EDP, EDAP, or any other energy-performance-area combination. The optimization loop continuously tries neighboring options[8] of the current one. If the new design option is better than the previous one, it is adopted unconditionally; if not, it is adopted with probability depending on an *acceptance* function. The *acceptance* probability, P_{accept}, is defined as

$$P_{\text{accept}}(E, E') = \begin{cases} 1 & \text{if } E' < E \\ E/E' & \text{if } E \le E' < 1.3E, \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where E is the old energy and E' is the new energy. We design this function so that the probability of accepting a move toward nonoptimal directions decreases as the difference between E and E' increases, and the probability goes down to 0 when a move is 1.3 times worse than the current solution. This feature prevents the optimization path from becoming stuck at a local optimum one that is worse than the global one. In theory, the probability that the simulated annealing algorithm terminates with a global optimal solution approaches 100% as we keep iterating the annealing process. However, in practice, we have to stop the optimization upon a given condition, and it becomes possible that the simulated annealing algorithm stops at a solution that is not globally optimal. We choose to end the iteration when the optimization path stops improving in the last K rounds. We use K = 20 in our experiments and find that it is a good trade-off between algorithm accuracy and speed.

[8]In this work, a neighboring option is generated by changing two parameters from the parameter set of L1 capacity, L1 associativity, L1 memory type, L2 capacity, L2 associativity, L2 memory type, L3 capacity, L3 associativity, and L3 memory type.

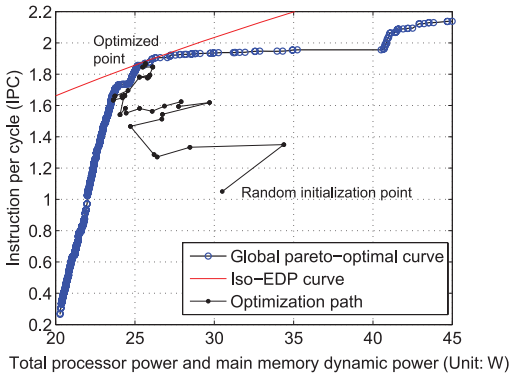

Fig. 26. The path of EDP optimization.

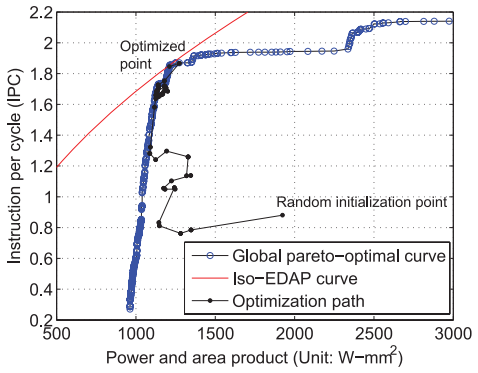

Fig. 27. The path of EDAP optimization.

Figure 26 shows how the simulated annealing algorithm eventually evolves an initial random design option to the global optimal solution in terms of EDP. In addition, Figure 27 shows the EDAP optimization path. By observing the optimization paths in Figures 26 and 27, we can see that the path initially tends to converge to a local optimal solution (e.g., point [X = 1200, Y = 1.7] in Figure 27), but the acceptance probability feature in Algorithm 1 allows the paths to roll back to a suboptimal solution and then keep evolving toward the global optima. Our K = 20 setting enables all of our optimization experiments to find the globally optimal solution (because we also run exhaustive searches and know the ground truth). We can further increase the K value if necessary, but that will lead to a longer optimization time.

Compared to exhaustive search of the same design space that takes more than 8 hours on an 8-core Xeon X5570 microprocessor, the proposed optimization methodology usually finds near-optimal values in less than 30 seconds. This optimization scheme provides an almost instant design decision given specified performance, energy, or area requirements. Furthermore, it becomes feasible to integrate this model into higher-level tools that consider not only memory system design trade-offs but also design trade-offs within microprocessor cores [Azizi et al. 2010].

8. DISCUSSION

Power consumption has been an issue for many years. Our exploration and optimization results demonstrate that both the EDP and the EDAP optimal points are close to the *y*-axis on the IPC-versus-power plot. In this design space region, ReRAM resources are adopted in the memory hierarchy (e.g., using ReRAM L3 caches, or more aggressively using ReRAM L2 and L3 caches). Even if performance constraints are applied, using ReRAM starting with the L3 cache always brings energy efficiency. Moreover, these energy-optimal points on the IPC-versus-area plot also show significant silicon area savings achieved from ReRAM without incurring much performance degradation. Compared to the best values for pure-SRAM designs, the introduction of ReRAM in L3 caches improves EDP and EDAP by 24% and 36% on a scaled 32nm 8-core SPARC-V9–like processor chip, respectively. The memory technology shift from SRAM to ReRAM achieves these improvements for the following reasons:

—The compact ReRAM module size greatly reduces the silicon area used for on-chip memories (EDAP improvement), or allows more on-chip memory to improve the performance (EDP and EDAP improvement);

Table VII. Overview of the Proposed Universal Memory Hierarchy

Level	L1 cache	L2 cache
Memory type	SRAM	SRAM or MOS-accessed ReRAM
Endurance requirement	10^{13} [Section 6.2]	10^{11} [Section 6.2]
Level	L3 cache	Main memory
Memory type	MOS-accessed or cross-point ReRAM	MOS-accessed or cross-point ReRAM
Endurance requirement	10^{10} [Section 6.2]	10^{8} [Qureshi et al. 2009b]

—The relatively smaller ReRAM size implies shorter wordlines and bitlines in the ReRAM cell array, and thus reduces the dynamic energy consumption per memory access (EDP and EDAP improvement); and
—The nonvolatility property of ReRAM eliminates the leakage energy consumption of memory cells (EDP and EDAP improvement).

Therefore, we envision a heterogeneous memory hierarchy as summarized in Table VII. In such a hierarchy, SRAM is used in L1 and L2 caches, MOS-accessed ReRAM may be used in L3 or even in L2 caches if ReRAM technology keeps improving (e.g., improvement on write speed and write endurance), and low-cost cross-point ReRAM may be used in L3 caches and main memory.

9. CONCLUSION

In the next era of computing, we need more energy-efficient and cost-effective computing. However, conventional SRAM and eDRAM technologies used in memory hierarchy designs have problems in reducing power consumption and silicon area with scaling. On the other hand, many emerging nonvolatile memory technologies such as STTRAM, PCRAM, and ReRAM have been researched and corresponding prototypes demonstrated. These new memory technologies bring desired features such as high density, fast access, good scalability, and nonvolatility, and they are potentially useful in many levels of future energy-efficient and cost-effective memory hierarchies. However, such emerging memory technologies are still new, and there are too many uncertainties in evaluating their actual impact on future memory hierarchy design. Therefore, it is necessary to have a framework that can model the circuit-level trade-offs among performance, energy, and area and can leverage such design variety into providing the best architecture-level memory hierarchy.

In this work, we first build a circuit-level performance, energy, and area estimation model for emerging memory technologies, then use this model to explore a wide range of memory module implementations, and generate a memory module library with various optimized designs. After that, we integrate this circuit-level model into an ANN-based architecture-level model and create a general performance-energy-area optimization framework for the memory hierarchy design in a joint circuit-architecture design space. Our validation results show that the proposed framework is sufficiently accurate for the purpose of design space exploration, and that by using this framework, we are able to rapidly explore a very large space of memory hierarchy designs and find good solutions in terms of energy-performance-area trade-offs. Moreover, we use this framework to evaluate new memory technologies such as ReRAM. Our experimental results reveal the memory design preferences for ReRAM in an 8-core CMP setting when the design targets EDP or EDAP goals. Our results show using ReRAM starting from L3 caches can achieve a 24% EDP improvement and a 36% EDAP improvement, which means that the best trade-offs in designing ReRAM memory hierarchy can greatly boost the energy efficiency or cost efficiency with only a slight impact on the IPC.

In general, this work is an initial effort to study the feasibility of building an energy-efficient or cost-efficient memory hierarchies by adopting emerging memory

technologies. We believe that this work is only the first step toward a new generation of energy-efficient and cost-efficient heterogeneous computer memory hierarchies.

REFERENCES

AMRUTUR, B. S. AND HOROWITZ, M. A. 2000. Speed and power scaling of SRAM's. *IEEE J. Solid-State Circuits 35*, 2, 175–185.

AZIZI, O., ET AL. 2010. Energy-performance tradeoffs in processor architecture and circuit design: A marginal cost analysis. In *Proceedings of the International Symposium on Computer Architecture*. 26–36.

BIENIA, C., ET AL. 2008. The PARSEC benchmark suite: characterization and architectural implications. In *Proceedings of the International Conference on Parallel architectures and Compilation Techniques*. 72–81.

CHEN, Y.-C., ET AL. 2003. An access-transistor-free (0T/1R) non-volatile resistance random access memory (RRAM) using a novel threshold switching, self-rectifying chalcogenide device. In *Proceedings of the International Electron Devices Meeting*. 750–753.

DE SANDRE, G., ET AL. 2010. A 90nm 4Mb embedded phase-change memory with 1.2V 12ns read access time and 1MB/s write throughput. In *Proceedings of the International Solid-State Circuits Conference*. 268–269.

DONG, X., ET AL. 2008. Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement. In *Proceedings of the Design Automation Conference*. 554–559.

DUBACH, C., ET AL. 2007. Microarchitectural design space exploration using an architecture-centric approach. In *Proceedings of the International Symposium on Microarchitecture*. 262–271.

ESHRAGHIAN, K., ET AL. 2010. Memristor MOS content addressable memory (MCAM): Hybrid architecture for future high performance search engines. *IEEE Trans. Very Large Scale Integrat. Syst. 99,*1–11.

INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS. 2011. The Model for Assessment of cmoS Technologies And Roadmaps (MASTAR). http://www.itrs.net/models.html.

INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS. 2012. Process Integration, Devices, and Structures 2012 Update. http://www.itrs.net/.

IPEK, E., ET AL. 2008. Efficient architectural design space exploration via predictive modeling. *ACM Trans. Archit. Code Optim.4*, 4, 1:1–1:34.

JOSEPH, P. J., ET AL. 2006a. Construction and use of linear regression models for processor performance analysis. In *Proceedings of the International Symposium on High-Performance Computer Architecture*. 99–108.

JOSEPH, P. J., ET AL. 2006b. A predictive performance model for superscalar processors. In *Proceedings of the International Symposium on Microarchitecture*. 161–170.

KALLA, R., ET AL. 2010. POWER7: IBM's Next-Generation Server Processor. *IEEE Micro 30*, 2, 7–15.

KAU, D. C., ET AL. 2009. A stackable cross point phase change memory. In *Proceedings of the IEEE International Electron Devices Meeting.* 27.1.1–27.1.4.

KAWAHARA, T., ET AL. 2007. 2Mb spin-transfer torque RAM (SPRAM) with bit-by-bit bidirectional xurrent qrite and parallelizing-direction current read. In *Proceedings of the International Solid-State Circuits Conference*. 480–617.

KIM, K.-H., ET AL. 2010. Nanoscale resistive memory with intrinsic diode characteristics and long endurance. *Appl. Physics Lett. 96*, 5, 053106.1–053106.3.

KIM, Y.-B., ET AL. 2011. Bi-layered RRAM with unlimited endurance and extremely uniform switching. In *Proceedings of the Symposium on VLSI Technology*. 52–53.

KIRKPATRICK, S., ET AL. 1983. Optimization by simulated annealing. *Science 220*, 4598, 671–680.

LEE, B. C., ET AL. 2009. Architecting phase change memory as a scalable DRAM alternative. In *Proceedings of the International Symposium on Computer Architecture*. 2–13.

LEE, B. C. AND BROOKS, D. M. 2006. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. 185–194.

LEE, K.-J., ET AL. 2008. A 90nm 1.8V 512Mb diode-switch PRAM with 266MB/s read throughput. *IEEE J. Solid-State Circuits 43*, 1, 150–162.

LEE, M.-J., ET AL. 2007. 2-stack 1D-1R cross-point structure with oxide diodes as switch elements for high density resistance RAM applications. In *Proceedings of the IEEE International Electron Devices Meeting*. 771–774.

LI, S., ET AL. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the International Symposium on Microarchitecture*. 469–480.

MAGNUSSON, P. S., ET AL. 2002. Simics: A full system simulation platform. *Computer 35*, 2, 50–58.

MARQUARDT, D. W. 1963. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.11*, 2, 431–441.

MENG, Y., ET AL. 2005. On the limits of leakage power reduction in caches. In *Proceedings of the International Symposium on High-Performance Computer Architecture*. 154–165.

MURALIMANOHAR, N., ET AL. 2008. Architecting efficient interconnects for large caches with CACTI 6.0. *IEEE Micro 28*, 1, 69–79.

NASA ADVANCED SUPERCOMPUTING (NAS) DIVISION. 2012. The NAS Parallel Benchmarks (NPB) 3.3. http://www.nas.nasa.gov/Resources/Software/npb.html.

QURESHI, M. K., ET AL. 2009a. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proceedings of the International Symposium on Microarchitecture*. 14–23.

QURESHI, M. K., ET AL. 2009b. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the International Symposium on Computer Architecture*. 24–33.

SARLE, W. S. 1995. Stopped training and other remedies for overfitting. In *Proceedings of the Symposium on the Interface of Computing Science and Statistics*. 55–69.

SASAGO, Y., ET AL. 2009. Cross-point phase change memory with $4F^2$ cell size driven by low-contact-resistivity poly-Si diode. In *Proceedings of the Symposium on VLSI Technology*. 24–25.

SCHECHTER, S., ET AL. 2010. Use ECP, not ECC, for hard failures in resistive memories. In *Proceedings of the International Symposium on Computer Architecture*. 141–152.

SEONG, N. H., ET AL. 2010. Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In *Proceedings of the International Symposium on Computer Architecture*. 383–394.

SHEU, S.-S., ET AL. 2011. A 4Mb embedded SLC resistive-RAM macro with 7.2ns read-write random-access time and 160ns MLC-access capability. In *Proceedings of the IEEE International Solid-State Circuits Conference*. 200–201.

SMULLEN, C. W., ET AL. 2011. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 50–61.

SUN, G., ET AL. 2009. A novel 3D stacked MRAM cache architecture for CMPs. In *Proceedings of the International Symposium on High-Performance Computer Architecture*. 239–249.

SUTHERLAND, I. E., ET AL. 1999. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann.

THOZIYOOR, S., ET AL. 2008a. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. In *Proceedings of the International Symposium on Computer Architecture*. 51–62.

THOZIYOOR, S., ET AL. 2008b. CACTI 5.1 technical report. Tech HPL-2008-20. HP Labs.

TSUCHIDA, K., ET AL. 2010. A 64Mb MRAM with clamped-reference and adequate-reference schemes. In *Proceedings of the International Solid-State Circuits Conference*. 268–269.

UDIPI, A. N., ET AL. 2010. Rethinking DRAM design and organization for energy-constrained multi-cores. In *Proceedings of the International Symposium on Computer Architecture*. 175–186.

WANG, J., ET AL. 2013. i2WAP: Improving non-volatile cache lifetime by reducing inter- and intra-set write variations. In *Proceedings of the International Symposium on High-Performance Computer Architecture*. 234–245.

WILTON, S. J. E. AND JOUPPI, N. P. 1996. CACTI: An enhanced cache access and cycle time model. *IEEE J. Solid-State Circuits 31,* 677–688.

XU, C., ET AL. 2011. Design implications of memristor-based RRAM cross-point structures. In *Proceedings of the Design, Automation & Test in Europe*. 1–6.

YANG, J. J., ET AL. 2008. Memristive switching mechanism for metal/oxide/metal nanodevices. *Nature Nanotechnology 3*, 7, 429–433.

ZHANG, Y., ET AL. 2007. An integrated phase change memory cell with GE nanowire diode for cross-point memory. In *Proceedings of the IEEE Symposium on VLSI Technology*. 98–99.

ZHOU, P., ET AL. 2009. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of the International Symposium on Computer Architecture*. 14–23.

Received December 2012; revised May 2013, August 2013; accepted August 2013