Chapter 10
A Circuit-Architecture Co-optimization Framework for Exploring Nonvolatile Memory Hierarchies

Xiangyu Dong, Norman P. Jouppi and Yuan Xie

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.

10.1 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 [1] and some embedded processor caches can compromise over 40 % of the total

X. Dong
Qualcomm Technology, Inc., Qualcomm, USA
e-mail: xiangyud@qualcomm.com

N. P. Jouppi
Google Inc., CA, USA
e-mail: jouppi@acm.org

Y. Xie (✉)
Pennsylvania State University and AMD Research, Pennsylvania, USA
e-mail: yuanxie@cse.psu.edu

Y. Xie (ed.), *Emerging Memory Technologies*,
DOI: 10.1007/978-1-4419-9551-3_10,

chip power budget [2]. Second, SRAM and DRAM are facing many difficulties in scaling down. For example, it is hard to scale down DRAM below a 20-nm process node due to the difficulty in keeping an adequate amount of cell capacitance [3]. The recent shift of some L3 on-chip caches from SRAM to eDRAM [4] and the research momentum in replacing DRAM main memory with various emerging nonvolatile memories [5–13] reflect the responses to such challenges in designing an energy-efficient and cost-effective memory hierarchy.

Recently, many alternative memory technologies, such as phase-change RAM (PCRAM)[1] [14–16], spin-torque transfer RAM (STTRAM)[2] [17, 18], and resistive RAM (ReRAM)[3] [19–21], 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 their pros and cons and the peripheral circuit design can vary the memory module property greatly, the future memory hierarchy 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.[4] 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 [14–21], where some of the memory prototypes show extremely fast access speed while 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 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 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

[4] Silicon area is the main factor that determines chip cost.

ReRAM in L3 caches can achieve overall improvements on EDP (by 28 %) and EDAP (by 39 %) on an 8-core chip multiprocessor (CMP) system. Finally, we propose a simulated annealing approach that can quickly find the near-optimal solution in designing a energy-efficient or cost-efficient memory hierarchy.

10.2 Related Work

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

10.2.1 Circuit Model for Memory Modules

Many circuit-level models have been provided to enable SRAM or DRAM design explorations. For example, CACTI [22, 23] is to estimate the performance, energy, and area of SRAM and DRAM caches. However, as CACTI is originally designed to model an SRAM-based cache, and some of its fundamental assumptions do not match the actual emerging nonvolatile memory circuit implementations. Besides CACTI, Amrutur and Horowitz [24] introduced an analytical model for estimating the SRAM array speed and power scaling. Another circuit-level model [25] uses a logic synthesis tool to build a circuit library and relies on curve fitting to represent the circuit design trade-offs.

10.2.2 Predictive Performance Model

Statistical models [25–30] 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 simulation, this is a one-time effort, and all 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. [26] used linear regression, Lee and Brooks [27] used cubic splines, whereas Azizi et al. [25] applied polynomial functions to create architecture-level models. Artificial neural network (ANN) [28–30] is another popular approach to build a predictive architecture model, which can efficiently explore exponential-size architectural design spaces with many interacting parameters.

10.3 Circuit Model: An RC Approach

After more than 30 years of application, both SRAM and DRAM memory modules have defined 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 the 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 [14–21], 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 states of these emerging technologies, there is no single design, standard design option can balance the trade-off 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 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.

10.3.1 Modeling Philosophy

We apply the modeling philosophy used in CACTI [31, 32] to establish a library of emerging memory technologies spanning from ultra-fast to ultra-dense 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 components. We obtain the process-related data of transistors and metal layers from the ITRS report [3] and the MASTAR tool [33]. The data cover the process nodes from 22 to 180 nm and support three transistor types, which are *high performance*, *low operating power*, and *low standby power*.

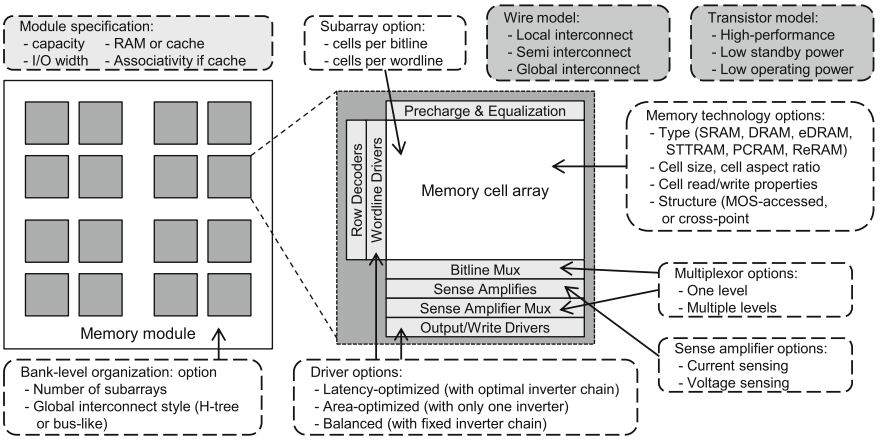

Fig. 10.1 The circuit-level model for memory module timing, power, and area estimations

10.3.2 Circuit Components and Tuning Knobs

Figure 10.1 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 thumbs is that smaller subarrays are faster and larger subarrays are area-efficient.

A subarray is the elementary structure, in which there are a set of peripheral circuit 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 circuit. For example, the output driver design can have the option of following logical effort [34] 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, the voltage-sensing scheme is straightforward but slower, while the current-sensing one incurs two-level sensing but 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 Fig. 10.1.

10.3.3 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 1T1R (1-transistor-1-resistor) structure used by many nonvolatile memory prototype chips [17, 18], 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 Fig. 10.2. 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 access transistor is bounded by the current needed by the write operation. This NMOS needs to be sufficiently large so that it can drive enough write current.

Cross-point cell corresponds to the 1D1R (1-diode-1-resistor) [14, 15, 35, 36] or the 0T1R (0-transistor-1-resistor) [19, 20, 37] structures used by several high-density nonvolatile memory chips. Figure 10.3 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 either rely on the one-way connectivity of diode (i.e., 1D1R) or the material's nonlinearity (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 Figs. 10.2 and 10.3. 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 circuitry designs. Several design issues such as half-select write, two-step sequential write, and external sensing [38] are included in our model.

10.3.4 Model Accuracy

We validate our circuit model against STTRAM [18], PCRAM [14], and ReRAM [21] prototypes. In general, the performance (i.e., read latency, write latency) estimation error is within 20 %, and the area estimation error is below 10 %.

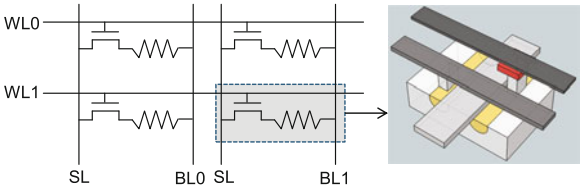

Fig. 10.2 The schematic view of MOS-accessed ReRAM arrays (*WL* wordline, *BL* bitline, *SL* sourceline)

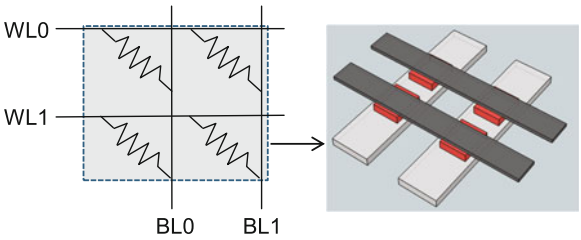

Fig. 10.3 The schematic view of cross-point ReRAM arrays without access devices (*WL* wordline, *BL* it bitline)

10.3.5 Circuit-Level Model Summary

In summary, our circuit-level model takes the 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 8 optimizations 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.

10.4 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.

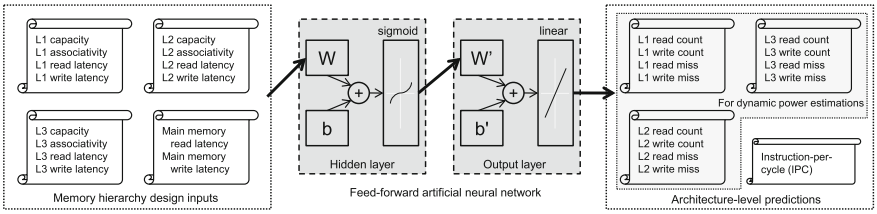

Fig. 10.4 The basic organization of a two-layer feed-forward artificial neural network (ANN)

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 artificial neural network (ANN) to fit the sampled simulation results into a predictive performance model.

10.4.1 Artificial Neural Network

Figure 10.4 shows the simplified diagram of a two-layer artificial neural network with one sigmoid hidden layer (which uses sigmoid functions as the calculation kernel) and one linear output layer (which uses linear functions as the calculation kernel). The input and output design parameters are also shown in Fig. 10.4. 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, and 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 the 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 box that generates predicted outputs as a function of the inputs,

$$\mathrm{L1_{readCount}} = f_1(\mathrm{L1_{capacity}}, \mathrm{L1_{assoc}}, \mathrm{L1_{readLatency}}, ..., \mathrm{L3_{capacity}}, ..., \mathrm{Mem_{writeLatency}}) \tag{10.1}$$

$$... = ...$$

$$\mathrm{L3_{writeMiss}} = f_{n-1}(\mathrm{L1_{capacity}}, \mathrm{L1_{assoc}}, \mathrm{L1_{readLatency}}, ..., \mathrm{L3_{capacity}}, ..., \mathrm{Mem_{writeLatency}}) \tag{10.2}$$

$$\mathrm{IPC} = f_n(\mathrm{L1_{capacity}}, \mathrm{L1_{assoc}}, \mathrm{L1_{readLatency}}, ..., \mathrm{L3_{capacity}}, ..., \mathrm{Mem_{writeLatency}}) \tag{10.3}$$

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 Fig. 10.4, 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:

$$\mathrm{X}_S = \sigma\left(\mathrm{W}_{S\times 14}\mathrm{I}_{14} + \mathrm{b}_S\right) \tag{10.4}$$

$$\mathrm{O}_{13} = \psi\left(\mathrm{W}'_{13\times S}\mathrm{X}_S + \mathrm{b}'_{13}\right) \tag{10.5}$$

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

10.4.2 Training and Validation

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 achieved 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 over-training [39]. 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 30–60 hidden neurons and trained using the Levenberg-Marquardt algorithm [40]. 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.

10.4.3 Sample Collection

In this work, we collect samples to evaluate an 8-core chip multiprocessor (CMP).[5] Each core is configured to be a scaled 32nm in-order SPARC-V9-like processor core with a 3.2 GHz 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. The detailed input design space is listed in Table 10.1. In this work, I-L1 and D-L1 are assumed to have the same design specification.

We use NPB [41] and PARSEC [42] as the experimental workloads. The workload size of the NPB benchmark is CLASS-C (DC has no CLASS-C setting, 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.[6] 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 [43] to collect sample data. Each Simics simulation is fast forwarded to the pre-defined 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.

[5] We also use the same methodology to collect the data for a 4-core CMP performance model, and the result is shown in Sect. 10.6.4.

[6] Another 23 ANN models are built for the 4-core CMP model.

10.4.4 Model Accuracy

Figure 10.5 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 Fig. 10.5 (middle), we can find that 8 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 %.

To measure the model accuracy, we use the metric $error = |predicted - actual|/actual$. The average prediction error is 4.29 %. Figures 10.6, 10.7, 10.8 show three examples of the ANN fitting results: a very accurate fit (0.15 % error), a typical fit (3.06 % error), and the worst fit in this work. Even in the worst case, the prediction error is under 18.71 %.

The prediction results of other output parameters (e.g., L1 read count, L1 write count, L2 read count, L2 write count) are similar to the IPC prediction.

10.5 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.

Table 10.1 Input design space parameters

Parameter	Range
Processor frequency	3.2 GHz
Processor core	8-core, in-order
I-L1 (D-L1) capacity	8KB to 64 KB
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	64 KB to 512 KB
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	512 KB to 16 MB
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

10.5.1 Framework Overview

Figure 10.9 shows an overview of this joint circuit-architecture exploration framework. As mentioned, 3,000 randomly generated architecture-level inputs per benchmark 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 [44] 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 Sect. 10.7 in detail.

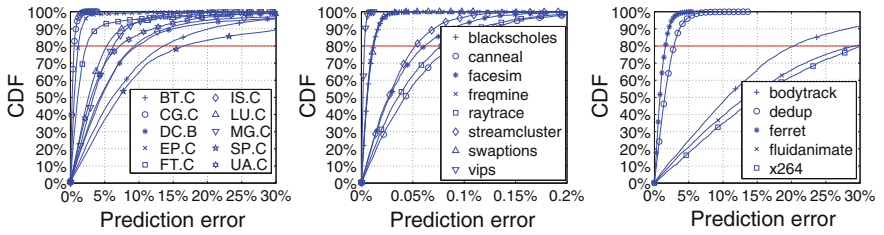

Fig. 10.5 CDF plots of error on IPC prediction of NPB and PARSEC benchmark applications: The x-axis shows the prediction error; the y-axis shows the percentage of data points that achieve the prediction error less than each x value

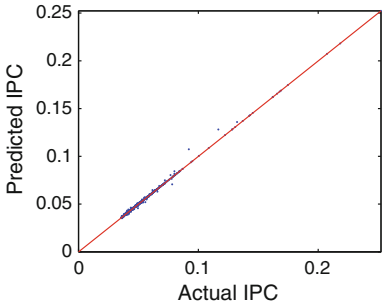

Fig. 10.6 An accurate ANN fitting example: MG from NPB

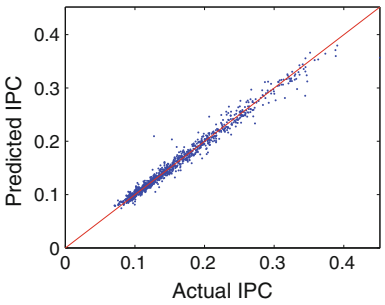

Fig. 10.7 A typical ANN fitting example: dedup from PARSEC

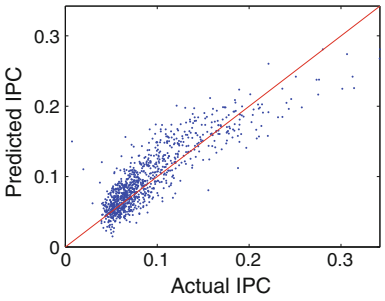

Fig. 10.8 The worst ANN fitting example: x264 from PARSEC

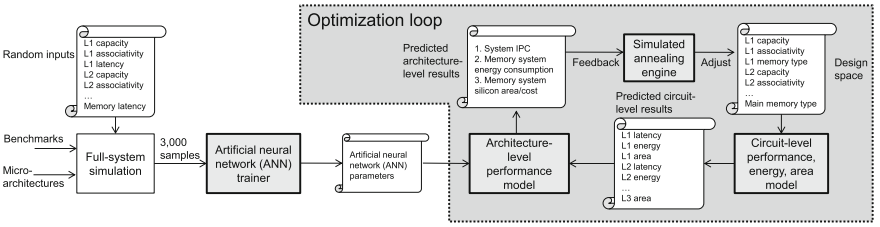

Fig. 10.9 Overview of the optimization framework. Architecture-level models are generated using sampling and an ANN trainer. Circuit-level models are used to estimate the latency, energy, and area of each memory module in the hierarchy. A simulated annealing engine is applied to find the near-optimal solution without exhaustive search

10.5.2 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 memory subsystem power consumption for a fair comparison. The power consumption of logic components including the processor cores, on-chip memory

controller, and inter-core crossbar are estimated by McPAT [45]. We use a 32 nm technology in this work.

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

$$\begin{aligned}\mathrm{E}_{\mathrm{memory,dynamic}} = &\sum_{i=1}^{3}\left[N_{\mathrm{readHit}_i}E_{\mathrm{hit}_i} + N_{\mathrm{readMiss}_i}E_{\mathrm{miss}_i} + \left(N_{\mathrm{writeHit}_i} + N_{\mathrm{writeMiss}_i}\right)E_{\mathrm{write}_i}\right] \\ &+ N_{\mathrm{readMiss}_3}E_{\mathrm{read}_4} + N_{\mathrm{writeMiss}_3}E_{\mathrm{write}_4} \qquad (10.6)\end{aligned}$$

$$\mathrm{P}_{\mathrm{memory,leakage}} = 2N_{\mathrm{core}}P_1 + N_{\mathrm{core}}P_2 + P_3 \qquad (10.7)$$

$$\mathrm{P}_{\mathrm{processor,total}} = \mathrm{E}_{\mathrm{memory,dynamic}}/T + \mathrm{P}_{\mathrm{logic,dynamic}} + \mathrm{P}_{\mathrm{logic,leakage}} \qquad (10.8)$$

In Eq. 10.6, 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. 10.7, 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. Equation 10.8 gives the total power consumption where T is the simulation time ($T = 10\ \mathrm{B}/3.2\ \mathrm{GHz} = 3.125$ s according to our experimental setup).

10.6 Design Exploration: An 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.

10.6.1 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]. An ReRAM cell consists of a metal oxide layer sandwiched between two metal electrodes as shown in Fig. 10.10. The electronic behavior of metal/oxide interfaces depends on the oxygen vacancy concentration of the metal oxide layer. Typically,

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

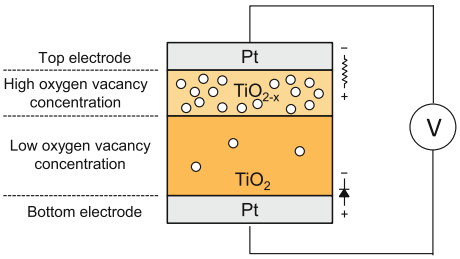

Fig. 10.10 An ReRAM cell example which uses Pt and Ti

Table 10.2 ReRAM technology assumptions

	ReRAM [38]	
	MOS-accessed	Cross-point
Cell size	20 F^2	$4F^2$
Write pulse duration	One pulse 50 ns per pulse	Two pulses 50 ns per pulse
State-0 resistance	10 kΩ	
State-1 resistance	500 kΩ	
Half-select resistance	–	100 kΩ
Write endurance	10^{12}	

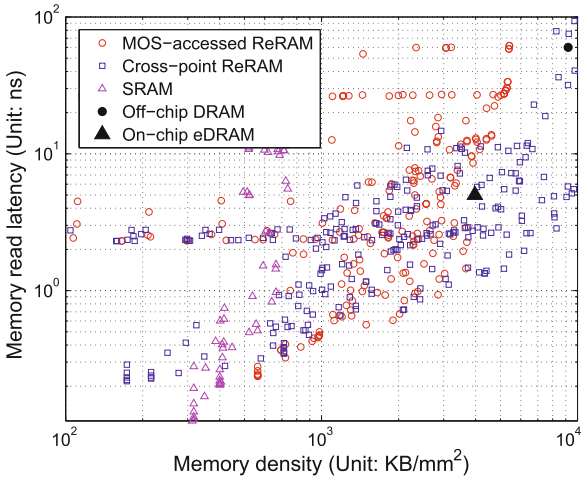

Fig. 10.11 The design spectrum of 32 nm ReRAM: read latency versus density

the metal/oxide interface shows Ohmic behavior in the case of very high doping and rectifying in the case of low doping [46]. In Fig. 10.10, the TiO_x region is semi-insulating indicating lower oxygen vacancy concentration, while the TiO_{2-x} is conductive indicating higher concentration.

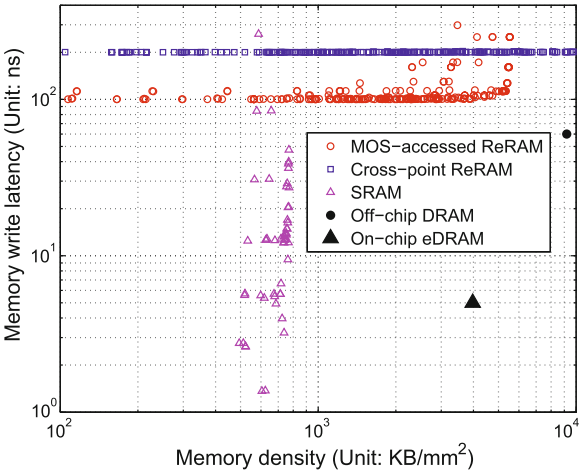

Fig. 10.12 The design spectrum of 32 nm ReRAM: write latency versus density

As an example, we use the ReRAM device parameters shown in Table 10.2 and explore the circuit-level design space at first. Figures 10.11 and 10.12 demonstrate the design spectrum of the emerging ReRAM memory 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 10X 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 50 ns in our experiments, and the latency of cross-point ReRAM is twice this due to two-step writes.

10.6.2 Wear-Leveling Assumption

Similar to NAND flash, ReRAM technology has limited write endurance (i.e., the number of times that an ReRAM cell can be overwritten). Many recent techniques [6, 7, 10] have been developed for extending the endurance of PCRAM-based memories, which can be directly borrowed for ReRAM wear leveling. In this case study, we simply assume that these techniques can give us perfect wear leveling. Given this assumption and the current write endurance level of 10^{10}–10^{12} [21, 47, 48], we can explore the possibility of using ReRAM technology in L2 and L3 caches. In later experiments, we add an additional 2 ns access latency due to the consideration of wear-leveling hardware overhead.

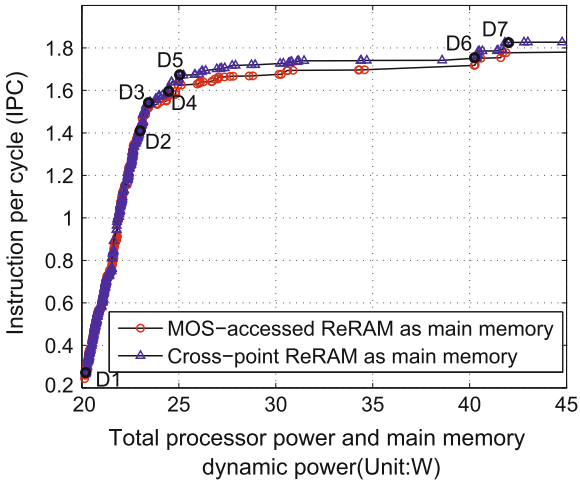

Fig. 10.13 *Pareto curves*: energy and performance trade-off of the memory hierarchy. Main memory dynamic power is included for a fair comparison

10.6.3 Memory Hierarchy Design Exploration

We next explore architecture-level memory hierarchy design space via ANN to show the Pareto-optimal curves of the trade-off range after adopting ReRAM technology. In this step, we separate the cache design space (L1, L2, and L3) and the memory design space. We assume 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 10.3 lists the timing and area parameters of MOS-accessed and cross-point ReRAM main memory solutions.

Focusing on the design space exploration of ReRAM-based memory hierarchies, Fig. 10.13 shows the Pareto-optimal curves of the power-performance trade-off. 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 Fig. 10.13 that a great amount of power consumption can be reduced by only taking a small amount of performance degradation. For instance, as shown in Fig. 10.13, design option D4 (using SRAM L1 and L2 caches but ReRAM L3 cache) reaches 1.610 IPC by consuming 24.50 W total power. Compared to design option D7 (using SRAM-only cache hierarchy) that reaches 1.826 IPC but consumes 42.00 W power, the achieved power reduction is 42 % but the performance degradation is only 12 %. The design option also meets the

Table 10.3 MOS-accessed and cross-point ReRAM main memory parameters (1 Gb, 8-bit, 16-bank)

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

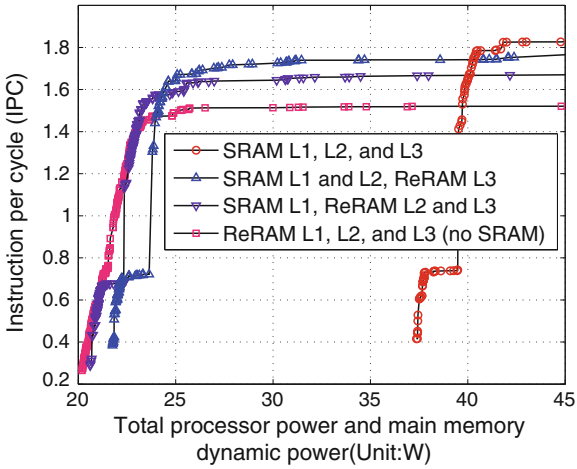

Fig. 10.14 *Pareto curves* (cross-point ReRAM as main memory): energy and performance trade-off under different constraints on ReRAM deployment

constraint set by 10^{10} write endurance as discussed in Sect. 10.6.2. 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 1.545 IPC by consuming only 23.52 W total power. To show how different cache hierarchy designs have been explored, we list the design parameters of 7 design options (D1 to D7) in Table 10.4.

We find that the Pareto-optimal curves are composed of several segments, such as D1-to-D2, D4-to-D6. The joints between every two segments represent the place where SRAM/ReRAM replacement occurs. Such replacements can be found in Fig. 10.14. In general, IPC improvements are achieved by adding more SRAM resources, and greater reductions in power consumption come from replacing SRAM with ReRAM. Figure 10.14 shows an ReRAM-only cache hierarchy is on the global Pareto-front but it achieves less than 0.30 IPC, and that segment has a large slope. Thus, it suggests SRAM should still be deployed at least in L1 caches. But, starting form L2, ReRAM cache deployment can achieve considerable additional power reduction by only sacrificing a small amount of performance. This is especially true for a hybrid on-chip cache hierarchy with SRAM L1/L2 caches and ReRAM L3 caches. Figure 10.14 shows that in this region, the total power consumption can be lowered to 24.50 W but the IPC is only degraded from 1.826 to 1.610 (i.e., design option D4 in Fig. 10.13).

Another benefit obtained from using ReRAM caches is silicon area reduction. Figure 10.15 shows the Pareto-optimal curves of cache area and performance trade-offs, which have similar shapes to the ones in the power-performance trade-off as shown in Fig. 10.15. The processor core area (including memory controller and crossbar) is 45.6 mm^2 as estimated by McPAT [45]. Achieving the highest performance using pure-SRAM caches costs at least another 12 mm^2 of silicon area while

Table 10.4 On-die cache hierarchy design parameters of 7 design options

	D1	D2	D3	D4	D5	D6	D7
L1 capacity	32 KB	8 KB	8 KB	8 KB	32 KB	8 KB	8 KB
L1 associativity	8	8	4	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	512 KB	512 KB	512 KB	64 KB	64 KB	64 KB	64 KB
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	16 MB	16 MB	8 MB	8 MB	64 MB	8 MB	8 MB
L3 associativity	16	16	8	8	8	8	8
L3 memory type	M-ReRAM	M-ReRAM	M-ReRAM	X-ReRAM	M-ReRAM	SRAM	SRAM
L3 optimized for	L	L	L	L	RE	L	WP
L3 sensing scheme	IN	IN	EX	IN	EX	IN	IN
IPC	0.265	1.410	1.545	1.610	1.677	1.751	1.826
Power consumption (W)	20.20	22.98	23.52	24.50	25.08	40.31	42.00
Silicon area (mm^2)	49.45	49.64	48.25	48.16	62.92	57.41	58.23

*Memory type abbreviation: M-ReRAM = MOS-accessed ReRAM, X-ReRAM = Cross-point ReRAM; Optimization abbreviation: *RL* read latency, *WL* write latency, *RE* read energy, *WE* write energy, *RP* read EDP, *WP* write EDP, *L* leakage, *A* area; sensing scheme abbreviation: *IN* internal, *EX* external

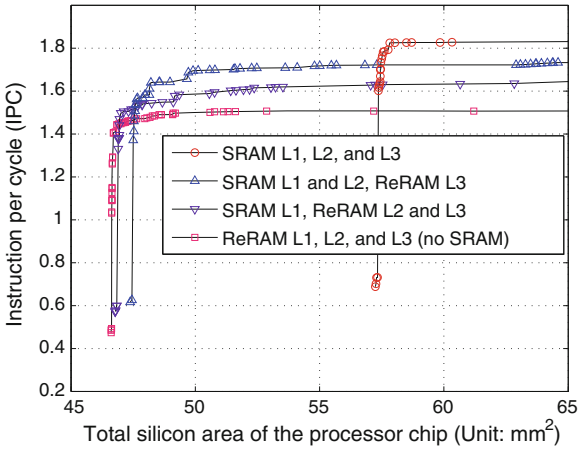

Fig. 10.15 *Pareto curves* (cross-point ReRAM as main memory): cache area and performance trade-off under different ReRAM deployments

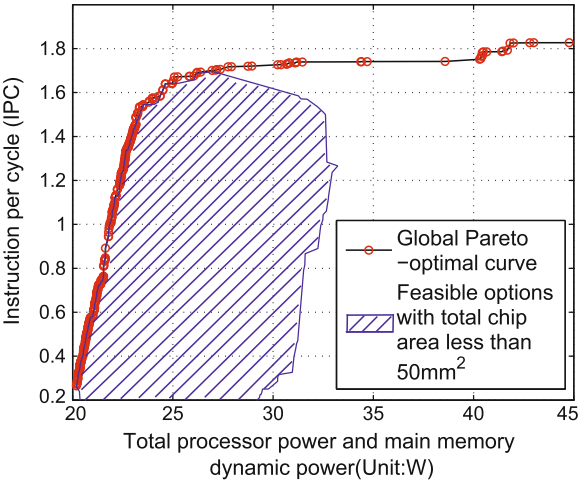

Fig. 10.16 The global Pareto-optimal curve (cross-point ReRAM as main memory) and feasible design options with total chip area less than 50 mm^2

replacing the SRAM L3 cache with ReRAM can save more than 7 mm^2 in chip area by degrading performance from an IPC of 1.82–1.70. We show the feasible region of designs with less than 50 mm^2 total cache area in Fig. 10.16. This result is extremely useful in the low-cost computing segment where the performance requirement is just-in-time but the chip cost is the first priority. Figure 10.16 also indicates using ReRAM caches can reduce power consumption and silicon area at the same time, which further improves EDAP.

10.6.4 Other Case Studies

As a general-purpose tool, our circuit-architecture joint space exploration framework is not limited to the case study of ReRAM-based cache hierarchy on an 8-core CMP system. Using the same circuit-level model, we can build another circuit library for PCRAM-based cache modules (PCRAM parameters are listed in Table 10.5). Furthermore, using the same architecture-level model, a 4-core CMP performance model can also be established.

Figures 10.17, 10.18 demonstrate how the framework can be easily adapted and help the design space exploration for an 8-core PCRAM-based and a 4-core ReRAM-based cache hierarchy in terms of the power-performance trade-off and the area-performance trade-off, respectively. The tool quickly provides the estimation that:

- In a 8-core CMP system, adopting *PCRAM* technology in L3 caches can achieve an EDP improvement of 16 % and an EDAP improvement of 25 %.
- In a *4-core* CMP system, adopting ReRAM technology in L3 caches can achieve an EDP improvement of 25 % and an EDAP improvement of 35 %.
- In a 8-core CMP system, adopting ReRAM technology in L3 caches can achieve an EDP improvement of 28 % and an EDAP improvement of 39 %.

From the 8-core PCRAM/ReRAM comparison, we can quickly find that ReRAM's fast switching property makes ReRAM outperform PCRAM in terms of IPC.

10.7 Design Optimization

Running full design space exploration using an exhaustive search is time-consuming and may not be necessary in most cases. 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 [44] algorithm to find a near globally optimal solution. The simulated annealing heuristic is described in Algorithm 1.

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 neighbor-

Table 10.5 PCRAM technology assumptions [16]

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

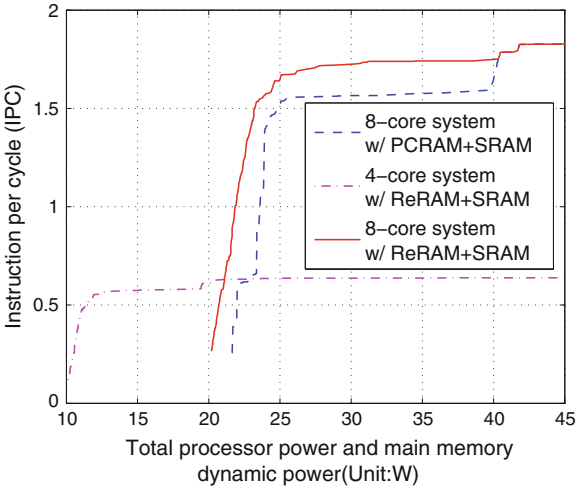

Fig. 10.17 The power-performance *Pareto curves*: (*1*) an 8-core system with PCRAM+SRAM; (*2*) a 4-core system with ReRAM+SRAM; (*3*) an 8-core system with ReRAM+SRAM

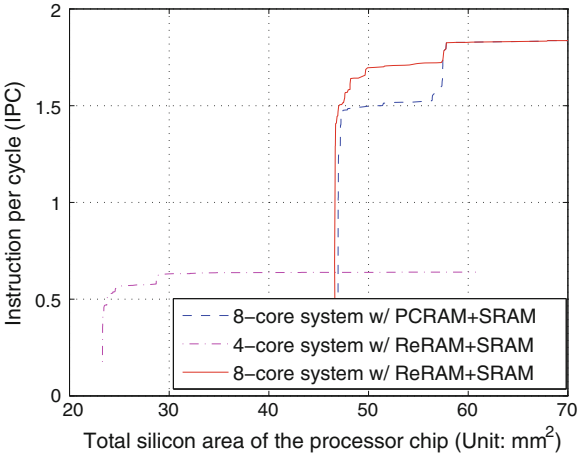

Fig. 10.18 The area-performance *Pareto curves*: (*1*) an 8-core system with PCRAM+SRAM; (*2*) a 4-core system with ReRAM+SRAM; (*3*) an 8-core system with ReRAM+SRAM

ing 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,

[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.

Algorithm 1 Design space optimization algorithm

```
state=s0, energy=E(state)
repeat
  new_state = neighbor(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
```

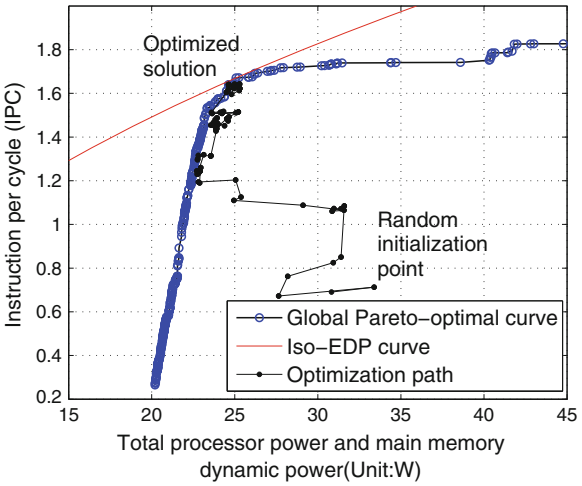

Fig. 10.19 The path of EDP optimization

$$P_{\text{accept}}(E, E') = \begin{cases} 1 & \text{if} E' < E \\ E/E' & \text{if } E \leq E' < 1.3E \\ 0 & \text{otherwise} \end{cases} \tag{10.9}$$

where E is the old energy, and E' is the new energy.

Figure 10.19 shows how the simulated annealing algorithm eventually evolves an initial random design option to a near globally optimal one in terms of EDP. In addition, Fig. 10.20 shows the EDAP optimization path.

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 the near-optimal values in less than 30 seconds. This optimization scheme provides an almost instant design decision given any specified performance, energy, or area requirements. Furthermore, it becomes feasible to integrate this model into higher level tools that consider not only the memory system design trade-offs but also design trade-offs within microprocessor cores [25].

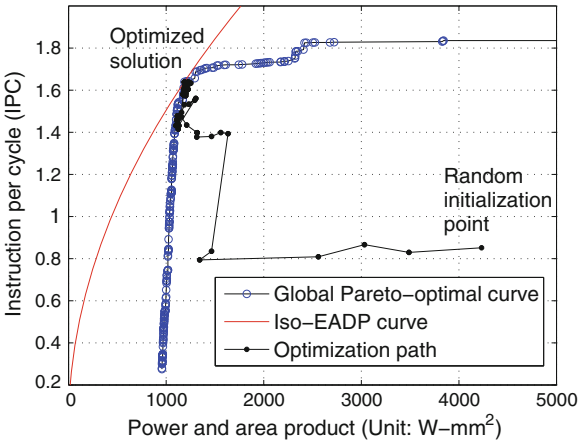

Fig. 10.20 The path of EDAP optimization

10.8 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 design (for example, 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, locating these energy-optimal points on the IPC versus area plot also shows significant silicon area savings achieved from ReRAM without incurring much performance degradation. Compared to the best values of pure-SRAM designs, the introduction of ReRAM in L3 caches improves EDP and EDAP by 28 and 39 % on a scaled 32 nm 8-core SPARC-V9-like processor chip, respectively. The memory technology shift from SRAM to ReRAM achieves these improvements for the following reasons:

Table 10.6 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 10.6.2]	10^{11} [Section 10.6.2]
Level	L3 cache	Main memory
Memory type	MOS-accessed or cross-point ReRAM	MOS-accessed or cross-point ReRAM
Endurance requirement	10^{10} [Section 10.6.2]	10^{8} [7]

- The compact ReRAM module size greatly reduces the silicon area used for on-chip memories (EDAP improvement) or allows more on-chip memory resource deployment to improve the performance (EDP and EDAP improvement);
- 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);
- 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 10.6. 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.

10.9 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 the corresponding prototypes have been 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 the 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 the 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 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 preference 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 28 % EDP improvement and a 39 % EDAP improvement, which means the 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 the initial effort to study the feasibility of building an energy-efficient or cost-efficient memory hierarchies by adopting emerging memory technologies. We believe this work is only the first step toward a new generation of energy-efficient and cost-efficient heterogeneous computer memory hierarchies.

References

1. 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* (pp. 175–186).
2. Meng, Y., et al. (2005). On the limits of leakage power reduction in caches. *Architecture: In Proceedings of the International Symposium on High-Performance Computer* (pp. 154–165).
3. International technology roadmap for semiconductors. Process integration, devices, and structures 2010 update. (2010). http://www.itrs.net/.
4. Kalla, R., et al. (2010). POWER7: IBM's next-generation server processor. *IEEE Micro*, *30*(2), 7–15.
5. Lee, B. C., et al. (2009). Architecting phase change memory as a scalable DRAM alternative. *Architecture: In Proceedings of the International Symposium on Computer* (pp. 2–13).
6. Zhou, P., et al. (2009). A durable and energy efficient main memory using phase change memory technology. *Architecture: In Proceedings of the International Symposium on Computer* (14–23).
7. Qureshi, M. K., et al. (2009b). Scalable high performance main memory system using phase-change memory technology. *Architecture: In Proceedings of the International Symposium on Computer* (pp. 24–33).
8. Qureshi, M. K. (2009a). Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. *In Proceedings of the International Symposium on Microarchitecture* (pp. 14–23).
9. Seong, N. S., 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* (pp. 383–394).
10. Schechter, S. (2010). Use ECP, not ECC, for hard failures in resistive memories. *Architecture: In Proceedings of the International Symposium on Computer* (pp. 141–152).
11. 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* (pp. 554–559).
12. Sun, G., et al. (2009). A novel 3D stacked MRAM cache architecture for CMPs. *Architecture: In Proceedings of the International Symposium on High-Performance Computer* (pp. 239–249).
13. 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* (pp. 50–61).
14. Lee, K.-J., et al. (2008). A 90 nm 1.8 V 512 Mb diode-switch PRAM with 266MB/s read throughput. *IEEE Journal of Solid-State Circuits*, *43*(1), 150–162.
15. Yoshitaka, S., 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* (pp. 24–25).

16. 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* (pp. 268–269).
17. Kawahara, T., et al. (2007). 2 Mb spin-transfer torque RAM (SPRAM) with bit-by-bit bidirectional current write and parallelizing-direction current read. *In Proceedings of the International Solid-State Circuits Conference* (pp. 480–617).
18. Tsuchida, K., et al. (2010). A 64 Mb MRAM with clamped-reference and adequate-reference schemes. *In Proceedings of the International Solid-State Circuits Conference* (pp. 268–269).
19. 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* (pp. 750–753).
20. Kim, K.-H., et al. (2010). Nanoscale resistive memory with intrinsic diode characteristics and long endurance. *Applied Physics Letters*, *96*(5), 053106.1-053106.3.
21. Sheu, S.-S., et al. (2011). A 4 Mb embedded SLC resistive-RAM macro with 7.2 ns read-write random-access time and 160 ns MLC-access capability. *In Proceedings of the IEEE International Solid-State Circuits Conference* (pp. 200–201).
22. Wilton, S. J. E., & Jouppi, N. P. (1996). CACTI: An enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, *31*, 677–688.
23. Thoziyoor, S., et al. (2008b). CACTI 5.1 technical report. Technical report HPL-2008-20. HP labs.
24. Amrutur, B. S., & Horowitz, M. A. (2000). Speed and power scaling of SRAM's. *IEEE Journal of Solid-State Circuits*, *35*(2), 175–185.
25. 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* (pp. 26–36).
26. 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* (pp. 99–108).
27. Lee, B. C., & 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* (pp. 185–194).
28. Ipek, E., et al. (2008). Efficient architectural design space exploration via predictive modeling. *ACM Transactions on Architecture and Code Optimization*, *4*(4), 1:1–1:34.
29. Joseph, P. J., et al. (2006b). A predictive performance model for superscalar processors. *In Proceedings of the International Symposium on Microarchitecture* (pp. 161–170).
30. Dubach, C., et al. (2007). Microarchitectural design space exploration using an architecture-centric approach. *In Proceedings of the International Symposium on Microarchitecture* (pp. 262–271).
31. Muralimanohar, N. (2008). Architecting efficient interconnects for large caches with CACTI 6.0. *IEEE Micro*, *28*(1), 69–79.
32. 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* (pp. 51–62).
33. International technology roadmap for semiconductors. The model for assessment of cmoS technologies and roadmaps (MASTAR). (2010). http://www.itrs.net/models.html.
34. Sutherland, I. E., et al. (1999). Logical effort: Designing fast CMOS circuits. Morgan Kaufmann.
35. 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* (pp. 98–99).
36. 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* (pp. 771–774).
37. 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.

38. Xu, C., et al. (2011). Design implications of memristor-based RRAM cross-point structures. *In Proceedings of the Design, Automation and Test in, Europe* (pp. 1–6).
39. Sarle, W. S. (1995). Stopped training and other remedies for overfitting. *In Proceedings of the Symposium on the Interface of Computing Science and, Statistics* (pp. 55–69).
40. Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, *11*(2), 431–441.
41. NASA advanced supercomputing (NAS) division. The NAS parallel benchmarks (NPB) 3.3. http://www.nas.nasa.gov/Resources/Software/npb.html.
42. 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* (pp. 72–81).
43. Magnusson, P. S., et al. (2002). Simics: A full system simulation platform. *Computer*, *35*(2), 50–58.
44. Kirkpatrick, S., et al. (1983). Optimization by simulated annealing. *Science Magazine*, *220*(4598), 671–680.
45. 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* (pp. 469–480).
46. Yang, J. J., et al. (2008). Memristive switching mechanism for metal/oxide/metal nanodevices. *Nature Nanotechnology*, *3*(7), 429–433.
47. Kim, Y.-B., et al. (2011). Bi-Layered RRAM with unlimited endurance and extremely uniform switching. *In Proceedings of the Symposium on VLSI Technology* (pp. 52–53).
48. Eshraghian, K., et al. (2010). Memristor MOS content addressable memory (MCAM): Hybrid architecture for future high performance search engines. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *99*, 1–11.