REVIEW

National Science Review
5: 577–592, 2018
doi: 10.1093/nsr/nwx082
Advance access publication 25 August 2017

COMPUTER SCIENCE

Memory that never forgets: emerging nonvolatile memory and the implication for architecture design

Guangyu Sun[1,*], Jishen Zhao[2], Matt Poremba[3], Cong Xu[4] and Yuan Xie[5,*]

ABSTRACT

In order to mitigate the problem of the 'memory wall', various emerging nonvolatile memory (NVM) technologies have been proposed to replace the traditional ones. These emerging NVMs include spin torque transfer random access memory, PCRAM, resistive random access memory, racetrack memory, etc. Compared to traditional memory technologies, they have the advantages of near-zero standby power, high storage density and nonvolatility, which make them competitive for future memory hierarchy design. However, it is inefficient to directly apply these NVMs in existing memory architectures. On the one hand, these NVMs have their own limitations, such as long write latency, high write energy, limited write numbers, etc. Thus, proper architecture modification is required to adopt them into the traditional memory hierarchy. On the other hand, the unique features of these NVMs allow new memory architectures in memory subsystems and also introduce new challenges to be solved at the same time. In this paper, we not only review various emerging NVMs but also study typical related work on these topics to investigate their implications for memory architecture design.

Keywords: non-volatile memory, cache, main memory, energy efficiency

[1]Center for Energy-Efficient Computing and Applications, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China; [2]Department of Computer Engineering, University of California at Santa Cruz, Santa Cruz, CA 95064, USA; [3]Advanced Micro Devices, Inc., Sunnyvale, CA 94085, USA; [4]Hewlett Packard Lab, Palo Alto, CA 94304, USA and [5]Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106, USA

***Corresponding authors.** E-mails: gsun@pku.edu.cn; yuanxie@ece.ucsb.edu

Received 26 March 2017; **Accepted** 26 April 2017

INTRODUCTION

Due to the fact that the development of memory technology is slower than computing logic technology, the well-known 'memory wall' problem has become a critical challenge for modern computer system design. The increasing number of cores in a CMP requires more data on-chip to meet their processing throughput. However, the low density of traditional SRAM technology limits the increase of on-chip memory capacity. At the same time, the high leakage power consumption of traditional SRAM/DRAM technologies significantly impedes the progress of memory hierarchy design. This problem becomes even worse as the technologies scale down and more on-chip memory is expected. Moreover, data stored in SRAM/DRAM memory become more and more vulnerable to radiation-based soft errors with technology scaling. The extra overhead of error correction mechanisms further limits the improvement of performance and induces more power consumption.

As a summary, traditional memory technologies, such as SRAM and DRAM, cannot satisfy the memory requirements with technology scaling because of low density, high standby power and poor reliability. To solve these problems, various emerging nonvolatile memory (NVM) technologies have been proposed to replace SRAM/DRAM for future memory hierarchy design because of their advantages of high density, zero standby power, fast access speed, nonvolatility, etc. These memory technologies include spin torque transfer random access memory (STT-RAM), phase change memory (PCM), resistive random access memory (RRAM), racetrack memory (RM), etc. With these emerging NVMs, the memory architecture design needs to be rethought to achieve high performance, low power consumption and high reliability. Since these emerging memory technologies have different features, the following issues should be considered so that these NVMs can be efficiently adopted.

First, for each level of the memory hierarchy, a proper NVM candidate should be selected. In fact, it is difficult to select a universal NVM that can be fitted in all levels of the memory hierarchy. More interestingly, the research has demonstrated that it is

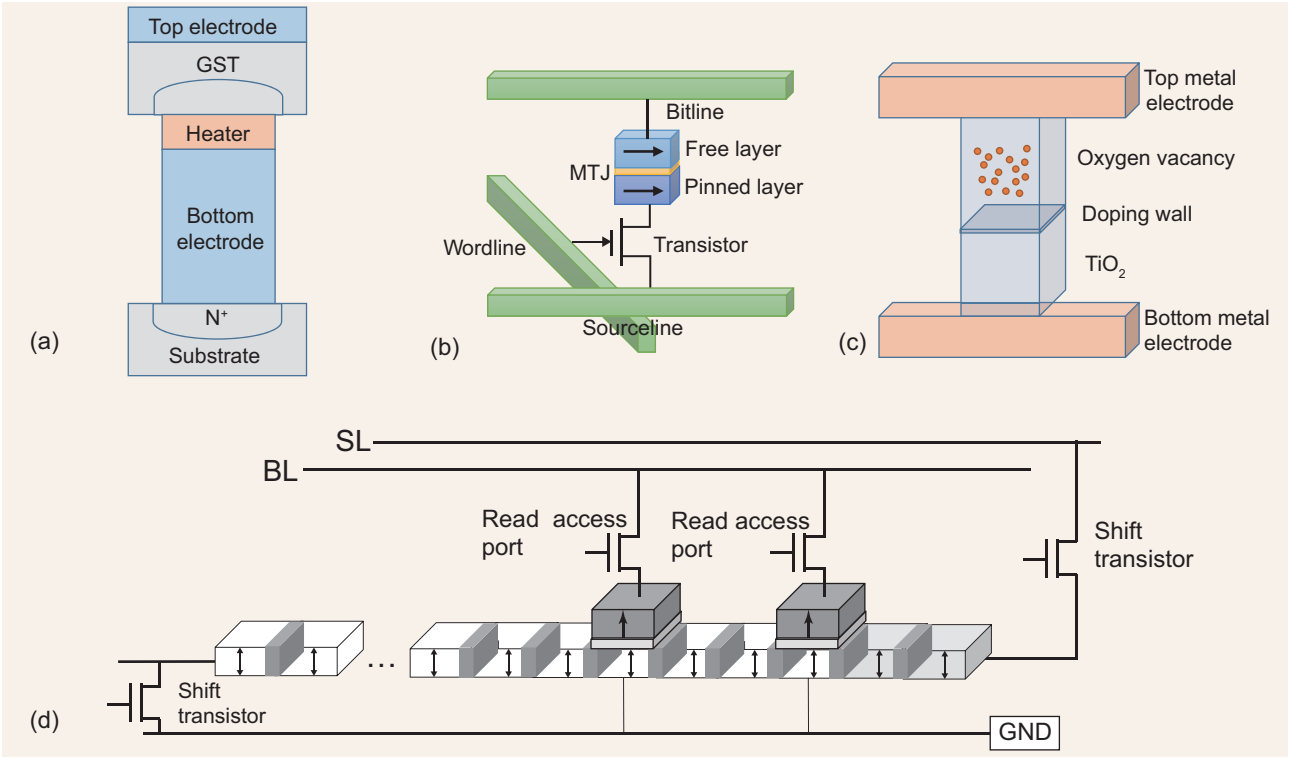

Figure 1. Illustration of various NVM cells.

more beneficial to adopt a hybrid design based on two memory technologies so that their advantages are leveraged and their drawbacks are hidden. Second, conventional memory architecture design for SRAM/DRAM technologies cannot fully exploit the advantages of these emerging NVMs. Thus, proper modification and optimization techniques are required to facilitate the adoption of NVMs in the different memory levels. Third, the unique features of these emerging NVMs not only allow opportunities for potential new memory architecture design, but also raise new challenges, which are not addressed in existing architecture design.

Considering these issues, this paper will review various emerging NVMs and their implications for memory architecture design. The rest of this paper is organized as follows. In the section entitled 'Background of emerging NVMs', popular emerging NVMs are introduced and compared with traditional technologies. The comparison can help researchers understand the advantages and drawbacks of these NVMs. In order to quantitatively evaluate these emerging NVMs, in respect of area, timing, energy, etc., different levels of models are introduced in the section entitled 'Modeling of emerging NVMs'. Then, in the section entitled 'Adopting NVMs in conventional memory hierarchy', research on how to adopt these NVMs in different levels of memory hierarchy with proper architecture modification is discussed. After that, new architectures enabled by these NVMs are introduced in the section entitled 'New architectures based on emerging NVMs'.

BACKGROUND OF EMERGING NVMS

The PCM cell in Fig. 1(a) relies on a chalcogenide alloy (e.g. Ge_2–Sb_2–Te_5, or GST) material [1] to store data. Similar to STT-RAM, data storage is achieved by the resistance transition. However, a PCM cell changes its state between the amorphous (high-resistance) and crystalline (low-resistance) phases of the chalcogenide material. In a SET operation, the phase change material is crystallized by heating the material using an electrical pulse. The state changes when the temperature is above its crystallization temperature. In a RESET operation, a different electrical current pulse is applied and then cut off shortly afterward to change the cell into the amorphous state [2]. PCM has demonstrated integration compatibility with CMOS (complementary metal–oxide–semiconductor) technology [3]. In addition, its storage density and access speed are similar to that of DRAM [4]. However, it suffers from limited endurance numbers [5]. The phase change material also shows good scalability under the current CMOS fabrication process [4,6,7]. Thus, it may be employed for continuous memory

Table 1. Comparison of the different memory technologies.

	SRAM	DRAM	NAND flash	PCM	STT-RAM	RRAM	RM
Data retention	N	N	Y	Y	Y	Y	Y
Cell factor (F^2)	50–120	6–10	2–5	6–12	4–20	<1	1–2
Read latency (ns)	1	30	50	20–50	2–20	<50	2–20
Write latency (ns)	1	50	$>10^6$	50–120	2–20	<100	2–20
Write numbers	10^{16}	10^{16}	10^5	10^{10}	10^{15}	10^{15}	10^{15}
Read/write power	Low	Low	High	High	Low	Low	Low
Other power	Leakage	Refreshing	None	None	None	None	Shifting

density increase [8,9]. Many PCM prototypes have been shown recently by many companies [10–14].

STT-RAM is based on a magnetic tunneling junction (MTJ) for data storage. In an MTJ, a layer of tunneling dielectric (e.g. MgO) is sandwiched between two ferromagnetic layers, as shown in Fig. 1(b). The magnetization of one ferromagnetic layer is pinned while that of the other one can be flipped during a write operation. Thus, these two layers are called the 'reference layer' and 'free layer', respectively. Data value stores in an MTJ depend on its resistance. When the magnetizations of the two ferromagnetic layers are in the same direction, the MTJ is in a low-resistance state, which can be used to represent bit '0'. In contrast, bit '1' is represented by the high-resistance state of the MTJ. Magnetization of the free layer is changed by the electrical current directly. Since the MTJ programming current is proportional to the MTJ area, STT-RAM is considered to have good scalability [15,16]. At the same time, STT-RAM achieves comparable access performance to SRAM technology. STT-RAM prototypes have recently been demonstrated by many companies and research groups [15],17–20]. Commercial products have also been launched by some companies [21].

RRAM is normally referred as those NVM technologies built on the resistance changing mechanisms, other than PCM and STT-RAM. A typical cell structure is shown in Fig. 1(c). The storage mechanisms of RRAM devices can be categorized as (but not limited to) space-charge-limited-current (SCLC) [22], filament [23], programmable-metallization-cell (PMC) [24], Schottky contact and traps (SCT) [25], etc. RRAM devices can also be classified as unipolar switching [26] and bipolar switching [24]. In unipolar switching, the programming operations are executed by using pulses with the same polarity but different durations or magnitudes. In bipolar switching, the operations are completed by applying pulses with opposite voltage polarities. Many companies have been working on RRAM developments, including Fujitsu, Sharp, HP Lab, IMEC, Unity [27–29], etc.

Racetrack memory (also known as domain-wall memory) is also considered as next-generation spintronic-technology-based memory after STT-RAM. A racetrack memory cell is composed of a tape-like stripe and several access transistors. A typical cell structure [30] is illustrated in Fig. 1(d). The racetrack memory stripe is made of magnetic material. It contains a lot of domains isolated by domain walls. The magnetization direction of a domain is programmed to store either bit 1 or bit 0. Several transistors are connected to the stripe to perform read, write and shift operations, respectively. They are called read access ports. Similar to STT-RAM, the bit value in a domain is sensed or programmed according to its magnetization direction. Together with the domain aligned under the access port, the reference domain forms a sandwich-structure magnetic tunneling junction (MTJ). Since the read port can only read the bit in a domain aligned under it, a shift operation is needed before reading other domains. Shift operations are achieved by using two transistors attached to both ends of the racetrack memory stripe. The driving current density needs to reach a threshold to enable movement of these domain walls. Note that all domain walls move in the same direction with the same speed [31]. Obviously, racetrack has major advantages of ultra-high density and fast access speed. Recently, as several prototypes of racetrack have been demonstrated, it has attracted more and more attention from researchers [32–35].

Comparisons between emerging NVMs and traditional memory technologies are shown in Table 1, in respect of different metrics. Previous research has demonstrated that these NVMs should be applied to different levels of memory hierarchy. STT-RAM and racetrack memory have been proposed for on-chip memory design. PCM is usually employed in main memory design or as the cache of NAND flash. Since there exist a lot of types of RRAM that have demonstrated different features, in respect of performance, energy and density, they may be applied to different levels accordingly. In fact, for any type of NVM, there exists a huge design space for trade-offs between performance, energy, and density.

In order to explore the design space for different design goals, low-level models are necessary to

estimate an NVM design, in respect of area, timing and energy. In addition, to quantitatively evaluate whether it is appropriate to adopting these NVMs in the memory hierarchy, an architectural level model is also needed. In the next section, state-of-the-art research on modeling of NVMs is introduced.

MODELING OF EMERGING NVMS

In this section, we take a bottom-up approach to model different NVMs from array through circuit to architecture level. The purpose of this model is to provide a fast circuit-architecture modeling of NVMs with acceptable accuracy for further architecture level exploration. The model has been integrated in an open-source simulation tool for public research. Although the simulation results may not be used for real implementation of these NVMs, they provide a reasonable trend for comparison.

Array-level modeling

Compared to traditional SRAM and DRAM, the array organizations of NVMs can have two forms: MOS-accessed array (1T1R) or diode-accessed array (1D1R).

1T1R structure

Conventionally, memory cells are connected together to form an array and isolated by using MOS access devices, as illustrated. In the MOS-accessed array structure, also known as 1T1R structure, the cell size is dominated by the large MOS device, which is necessary to drive enough write current even though the storage element (MTJ, GST or MIM) itself is much smaller. The driving current of NMOS, I_{DS}, can be first-order estimated as follows (note that Equations 1 and 2 are for long-channel drift/diffusion devices; the equations are subject to change depending on the technology, though the proportional relationship between the current and W/L still hold for very advanced technologies):

$$I_{DS} = K\frac{W}{L}\left[\left(V_{GS} - V_{TH}\right)V_{DS} - \frac{V_{DS}^2}{2}\right] \quad (1)$$

if NMOS is working at the linear region; or calculated by

$$I_{DS} = \frac{K}{2}\frac{W}{L}\left(V_{GS} - V_{TH}\right)^2\left(1 + \lambda V_{DS}\right) \quad (2)$$

if NMOS is working at the saturation region. Hence, no matter in which region NMOS is working, the current driving ability of NMOS is proportional to its width-to-length (W/L) ratio, which determines the NMOS size. (Usually, the transistor length (L) is fixed as the minimal feature size, and the transistor width (W) is adjustable.) Once the transistor size is determined, the cell size can be calculated using the following equation:

$$\text{Area}_{\text{cell,MOS-accessed}} = f(W/L)(F^2), \quad (3)$$

in which the width-to-length ratio (W/L) is determined by Equation 1 or Equation 2, and f is specific to the design rule.

1D1R structure

1D1R (1-diode–1-resistor), also known as 1S1R (1-selector–1-resistor), structures use cross-point architecture to enable high-density NVM design [36]. For 1D1R structure, a diode is inserted between the word line and the storage element. Such cells either rely on the one-way connectivity of the diode (i.e. 1D1R) or the leverage materials' nonlinearity (i.e. 0T1R) to control the memory access path. The width of wordlines and bitlines can be the minimal value of 1F and the spacing in each direction is also 1F; thus, the cell size of each cross-point cell can be as small as $4F^2$. Although the crossbar structure brings substantial density benefits, it also introduces a series of design challenges; two representative examples are the sneak current issue and the IR drop issue. The sneak current appears in the crossbar array as a result of uncontrolled current flow. In order to overcome this issue, an access diode, which has a much smaller overhead than an access transistor, can be employed to restrict the sneak current. To achieve the best area efficiency, the diode is vertically connected to the cell. In this way, the area overhead of the access device is eliminated. A good example of the access device was proposed by Shenoy *et al.* [37]; it provides the potential to build a large RRAM array with a very small sneak current. A detailed modeling methodology has also been proposed [38].

Circuit-level modeling

Generic timing and power estimation

An accurate circuit-level model should consider wire resistance and wire capacitance from interconnects, turn-on resistance, switching resistance, gate, and drain capacitances from transistors. In addition, it should consider equivalent resistance and capacitance from memory storage elements (e.g. MTJ in STT-RAM, GST in PCM, or metal oxide in RRAM). After calculating the resistances and capacitances

of nodes, the delay of each logic component is calculated as follows:

$$\text{Delay} = \tau\sqrt{\left(ln\frac{1}{2}\right)^2 + \alpha\beta}, \quad (4)$$

where α is the slope of the input, $\beta = g_m R$ is the normalized input transconductance by the output resistance, and $\tau = RC$ is the RC time constant.

The dynamic energy and leakage power consumption can be modeled as

$$\text{Energy}_{\text{dynamic}} = C V_{DD}^2 \quad (5)$$

$$\text{Power}_{\text{leakage}} = V_{DD} I_{\text{leak}} \quad (6)$$

where both gate leakage and sub-threshold leakage currents are included in I_{leak}.

The overall memory access latency and energy consumption are estimated by combining all the timing and power values of the circuit components together. The generic models are widely adopted in CACTI [39], McPAT [40] and NANDFlashsim [41].

Sensing model

Unlike other peripheral circuitry, the sense amplifier is an analog design instead of a logic design. Different sensing schemes have their impacts on the trade-offs between performance, energy and area. In the current sensing scheme, the state of the memory cell (STT-RAM, PCRAM or RRAM) is read out by measuring the resulting current through the selected memory cell when a read voltage is applied: the current on the bitline is compared to the reference current generated by the reference cells, the current difference is amplified by current-mode sense amplifiers, and they are eventually converted to voltage signals. Voltage sensing is an alternative approach that applies a current source to the selected memory cell and senses the voltage via a voltage-mode sense amplifier. The RC models of different sensing schemes are discussed in prior work [42]. Current sensing is significantly faster than voltage sensing with a larger area overhead.

Charge pump

The write operations of some NVM chips require a voltage higher than the chip supply voltage. Therefore, a charge pump that uses capacitors as energy storage elements to create a higher voltage is necessary in some NVM designs. The area of the charge pump circuits can be calculated as [43]

$$A_{\text{charge-pump}} = k \cdot \frac{N^2}{(N+1) \times V_{DD} - V_{\text{Out}}} \frac{I_L}{f}, \quad (7)$$

where k is a technology-dependent constant, N is the number of stages in the charge pump, V_{Out} is the output voltage, I_L is the write current and f is the working frequency.

Simulation framework

The NVM technologies are still not mature, and only a limited number of prototype chips have been demonstrated; these just cover a small portion of the entire design space. In order to facilitate the architecture-level NVM research by estimating the NVM performance, energy and area values under different design specifications before fabricating a real chip, NVSim [44] was built as a circuit-level modeling framework for NVM performance, energy and area estimations; it supports various NVM technologies including STT-RAM, PCRAM, RRAM and legacy NAND flash.

Architecture-level modeling

In addition to circuit-level modeling requirements, NVM memory designs consist of new concepts that are not applicable to SRAM and DRAM. Major differences include mitigating write problems, endurance improvement techniques, multi-level storage, and potential for hybrid memory designs. In order to model these differences, NVMain was developed as an NVM-specific memory framework [45]. The framework outlines two key features that make it desirable for NVM memory modeling. First, a modular design is used so that different methods of alleviating endurance, writes issues and hybrid memories can be easily enabled or disabled for architectural design space exploration. Second, it is the first main memory simulator to consider data values and does so in an efficient manner. Data values are very useful to explore application-level implications of NVM designs, especially multi-level cell-based memories. Several techniques have been proposed in the literature for all of these difference and are described in the section entitled 'Adopting NVMs in conventional memory hierarchy'. Here we give more details about how and why they may be modeled in an NVM main memory framework.

Modeling lifetime optimizations

Techniques for managing NVM lifetime are modeled by providing the ability for both data encoding to reduce writes and address manipulation to

provide fault correction and a more uniform distribution of writes known as wear-leveling. In the data encoding approach, the model considers both the time and energy required to encode data in order to improve lifetime. Typically, this requires writing as few bits as possible, so we must also be able to model read-before-write techniques depending on implementation. This will impact the total latency of write requests and thus should always be considered in such implementations. For implementations that do not require read-before-write approaches, encoding techniques that trade lifetime for latency can also be simulated. In the case of wear-leveling and errors that cannot be corrected, NVM designs may choose to build 'spare' rows or use 'pointers' to other locations in memory that will also impact access latency and should be modeled.

Multi-level cells and write optimization

Modeling multi-level cells is potentially cumbersome yet important. This low-level modeling of data requires knowing both the previous and next values of individual groups of bits. To accommodate this, highly efficient bit comparison techniques are necessary for modeling. Once the data change values are determined, it allows us to model the required transitions between the two values. Depending on the memory technology, this can result in extremely diverse performance and energy characteristics. NVMs that use an iterative write approach with a wide variance will have a high impact on memory request latency and energy values. As a result of this, modeling techniques to mitigate potentially long writes e.g. via delaying or rescheduling are also desirable. Furthermore, to make effective use of write drivers, we may model 'scheduling' of the write drivers to complete writing all bits as quickly as possible.

Hybrid memory designs

The research has shown that NVM designs may also consist of multiple levels of memory (e.g. slow/fast memory or hybrid NVM and SRAM/DRAM). In order to support simultaneous modeling and interactions between multiple levels of memory in such designs, we need both the basic memory model as well as the 'glue' to connect memories together. It is therefore desirable for a model to be able to consider multiple types of memory protocols (i.e. DDRx and LPDDRx) as well as memory technology (e.g. NVM and DRAM). The 'glue' can be approached in different ways, including software-managed slow/fast memory mapped regions, hardware-managed migration techniques between slow/fast memory, and main memory style caches placed before a final backing main memory. Modular approaches are highly desirable for this type of modeling so that we may explore and compare architectures.

ADOPTING NVMS IN CONVENTIONAL MEMORY HIERARCHY

According to their characteristics, these emerging NVMs are adopted into different levels of a conventional memory hierarchy, which include cache, main memory and storage. More importantly, architectural-level modification is required to facilitate the adoption in different levels, which are introduced in detail in this section.

NVM-based cache architecture

Traditionally, SRAM is employed for on-chip cache design because of its fast access speed. Compared to SRAM technology, two emerging NVMs based on spintronic technologies, STT-RAM and racetrack memory, can achieve similar access performances. At the same time, these NVMs can increase cache capacity several times over compared to traditional SRAM with the same area constraint. Thus, both STT-RAM and racetrack memory have the potential to replace SRAM for future on-chip cache design, especially for multi-core processors. However, NVM caches also have their own issues, including long write latency/energy, retention time, extra shift operations, etc.

STT-RAM-based cache architecture

There exists extensive research on how to adopt STT-RAM as on-chip memory, especially for the last-level cache (LLC). The research has reached several conclusions, which are summarized as follows. First, because STT-RAM has much higher storage density than SRAM, the capacity of a cache can be increased under the same area constraint. Consequently, the cache hit rate is increased so that performance is improved. Second, since STT-RAM has near-zero standby power, the leakage power consumption of a cache is reduced significantly. Third, both write latency and energy of STT-RAM are several times higher than those of SRAM. Thus, for those applications with intensive write operations to the cache, performance may be degraded due to the long write latency of STT-RAM. In addition, the dynamic energy consumption is also increased substantially, which may offset the benefit of leakage power reduction. Therefore, the major challenge of using STT-RAM cache is how to mitigate its write problem. To overcome this problem, various techniques have been proposed. Basically, they can be

categorized into the following groups, targeting different optimization goals.

Write latency reduction techniques As addressed above, cache performance may be degraded due to the long write latency of STT-RAM. Thus, how to reduce the total write latency has become a major research problem. There are two potential solutions: 'reducing total write intensity' and 'hiding write latency'.

It is straightforward that write latency is decreased when the total write intensity is reduced. A typical technique is to optimize the cache replacement policy to reduce the write intensity [46]. For example, if STT-RAM is employed for L3 cache design, the replacement policy of L2 cache can be modified to reduce the number of write-back operations to the L3 cache. A straightforward method is to select 'clean' data rather than 'dirty' data for replacement. However, one drawback is that it may reduce L2 cache hit ratio. Thus, the trade-off should be explored to achieve the best performance. For example, clean data and dirty data can be managed separately with different priorities [46].

Considering the fact that write operations are normally out of the critical path of data access, another solution is to try to hide the write latency by completing write operations without blocking those pending read operations. The preemptive write-buffer design [47] is a typical technique following this method. In this write-buffer design, when a write operation is in process, an incoming read operation can stall the write operation accordingly so that the read operation on the critical path will not be blocked for a long period. Whether the write operation should be preempted depends on its completeness and the difference between write latency and read latency.

Hybrid SRAM and STT-RAM architecture The main purpose of using hybrid architecture is to leverage both the low write latency of SRAM and the high storage density and low leakage power of STT-RAM. The pivot of this method is to properly select the ratio between SRAM and STT-RAM and propose corresponding the data allocation/mitigation policy [47–49]. Considering the features of STT-RAM and SRAM, the hybrid cache architecture is normally dominated by STT-RAM with a small portion of SRAM. For example, the STT-RAM to SRAM ratio is normally set as 32:1 or 16:1 [47]. For the data allocation policy, the basic idea is to allocate data with a high write intensity to SRAM and the rest to STT-RAM. The allocation logic can either be designed in the cache controller or implemented at software level through a compiling technique [47–49]. A state-of-the-art allocation policy is to categorize cache requests into different types, which are assigned different corresponding policies [48]. Obviously, the hardware-based design is simple and transparent to applications. More complicated policies may be implemented in software dedicated to various purposes and applications.

Device-architecture co-optimization The long write latency and high write energy problems of STT-RAM can also be mitigated by leveraging device-level features and modifying the architecture accordingly. One typical architecture is to trade the nonvolatility of STT-RAM for performance and energy improvement. This is based on the fact that some data in caches are updated or replaced frequently. Thus, the requirement for data retention time can be relaxed. Sun *et al.* propose an STT-RAM cache with both volatile and nonvolatile regions [50]. Data that are not frequently accessed are allocated to the nonvolatile region. Data that are frequently accessed are moved to the volatile region. In addition, a data-refreshing technique is introduced to ensure that data in volatile region are not lost. Another optimization is based on the difference between programming bit '0' and bit '1'. When process variations are considered, the latency of writing bit '1' is several times longer than that of writing bit '0'. Thus, a log-based write method is proposed to improve the write performance of STT-RAM. The basic idea is to 'erase' all data of STT-RAM to bit '1' before they are updated by valid data. Consequently, only bit '0' is written in a normal write operation. Since the erase operation can be processed in the background, the write performance is significantly improved [51].

After applying the techniques above to STT-RAM cache architecture, it can outperform the SRAM-based cache substantially, in respect of both performance and power consumption. Note that similar techniques can also be employed for the cache architecture based on other NVMs (e.g. RRAM). Related work is not discussed due to the page limitation.

RM-based cache architecture

There exists some work focusing on improving the performance of shift operations. Several techniques are proposed to reduce the number of shift operations, or remove them from the critical path. The major methods include pre-shifting by prediction, reordering by profiling, and compression by reorganizing the data structure.

Motaman *et al.* [52] predict the shift intensity, and change the shift current to get minimal shift latency for high intensity. Venkatesan *et al.* [32]

pre-shift the racetrack memory head by predicting incoming interleaved warps in the GPGPU (general-purpose graphics processing unit). For regular data patterns, this method can achieve a reasonable performance improvement, but irregular patterns always reduce the prediction accuracy.

Besides the prediction, some research reorders the access sequence or the shift sequence. Zhenyu *et al.* [30] proposed a hardware-based way block reorder mechanism in the cache, to swap frequently accessed blocks to access ports. Mao *et al.* [53] utilized the sequential access pattern by rescheduling the warp issuing order in the GPGPU, to amortize the shift overhead in the GPGPU register file. By manipulating the incoming request sequence, this technique can further improve the performance.

Recently, researchers have demonstrated that the shift penalty can also be reduced by compression by reorganizing the data structure. Ranjan *et al.* [34] lazily shrink and expand the bits used in the racetrack memory strip in the direct mapped cache, dynamically reconfiguring design objects between low shift penalty and low miss penalty. Haifeng *et al.* [54] compress the cache line and shift each racetrack independently to reduce the number of shifted racetracks. By reducing the amount of bits that it is necessary to access, this method can also improve the performance.

Besides the performance improvement, some other work focuses on improving the reliability of the racetrack memory, such as shift errors, and thermal stability. Zhang *et al.* [35] define and correct the position error issue, by the hi-fi playback technique. Iyengar and Ghosh [55] model the joule heating issue of racetrack memory shift operations. These techniques help keep racetrack memory reliable and provide sustainable functionality.

NVM-based main memory architecture

Traditional main memory technology, DRAM, has the issue of high static power dissipation. Therefore, researchers explore PCM as a replacement for DRAM in main memory design. A comparison between DRAM and PCM shows that PCM can significantly reduce the static power of main memory. In addition, PCM has a much higher storage density. However, PCM has a long write latency and a high write energy consumption. More importantly, PCM has much lower endurance than DRAM. Consequently, lifetime is a critical design challenge of PCM-based main memory. Furthermore, the non-volatility property of PCM introduces a potential security problem for the main memory. In the following, we will discuss recent studies seeking to tackle the write latency, write energy and security challenges of PCM-based main memory design.

Optimizing write latency and write energy

In 2009, researchers start to explore using PCM to replace DRAM in main memory design with two efficient optimization mechanisms [56–58]. First, a few studies reduce PCM writes by employing DRAM as a buffer for PCM access. Second, recent studies develop a partial write scheme, which effectively reduces PCM writes by only updating the modified data. Other studies develop DRAM/PCM hybrid main memory designs that exploit the benefits from both DRAM and PCM. The basic idea is similar to the aforementioned hybrid cache designs. Recent works also show that PCM with multi-level cells can further increase the storage density of the main memory, whereas such designs can incur even longer write latency and write energy consumption with PCM access. A large body of recent studies intend to optimize the performance of multi-level cell PCM-based main memory. One solution is to design a hybrid memory with both multi-level cell and single-level cell PCMs. Such a design can achieve a balance between performance/energy and capacity. Another solution is to adopt multi-stage writes to reduce the write energy consumption. Recent studies also employ selective data combination to improve PCM write bandwidth. Write pause mechanisms similar to those used in STT-RAM caches are also explored to improve the write performance of PCM access [59,60].

Optimizing lifetime

Recent studies on optimizing PCM lifetime [56–58] can be categorized into two types: (1) reducing the number of writes and (2) wear-leveling. To reduce the number of PCM writes, one simple yet effective method is to compare the data updates and the existing data values bit by bit and only modify those bits that have a different value. Furthermore, recent studies explore a flip-N-write technique, which only performs bit flips when there is a difference generated by comparison. Such methods can reduce PCM writes by over 80%. Another solution is to employ data encoding techniques to reduce PCM writes. For example, recent studies [61] propose to compress the frequently accessed data before writing them into PCM using a coset-coding encoding scheme. Such method can effectively reduce the number of PCM writes. Wear-leveling is another promising solution to improve PCM lifetime. Typically, some regions

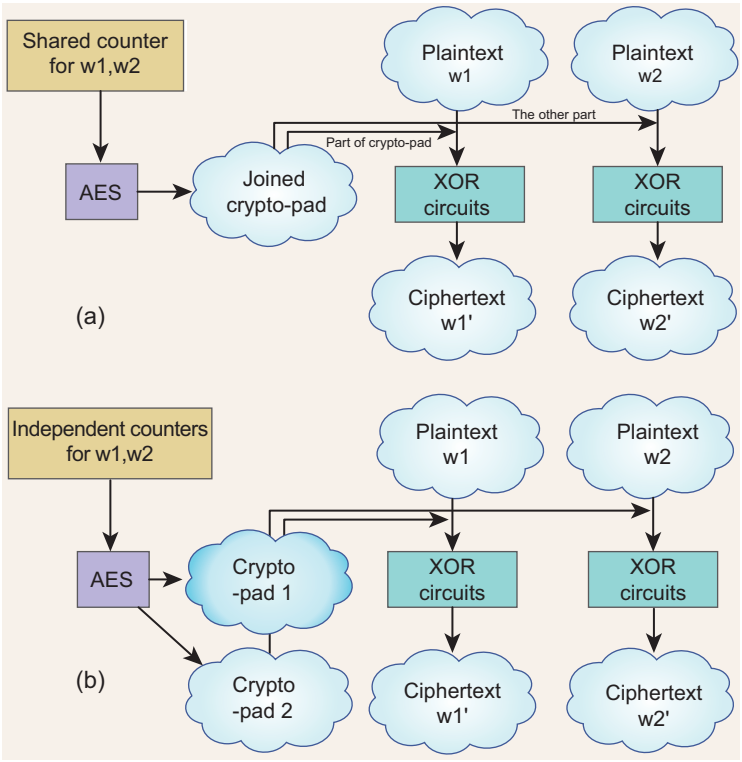

Figure 2. Counter-mode encryption: (a) traditional design and (b) improved design.

of the main memory are frequently updated while values of other regions remain the same most of the time. As a result, the frequently updated memory regions can wear much faster than the rest of the main memory. Wear-leveling is developed to address such issue. Wear-leveling techniques can be categorized into three types based on the granularity: wear-leveling within memory line, wear-leveling between memory lines and wear-leveling among segmentation. All three types adopt the same basic idea: periodically switching the data between various address spaces. Wear-leveling mechanisms can adopt either table-based mapping [57,58] or algebraic mapping [62,63] to keep track of the data switching. With table-based mapping, we need to build a mapping table between logical and physical addresses with a method similar to that used in flash-based SSDs. Algebraic mapping adopts a predefined formula to determine the mapping. Instead of reducing the number of PCM writes, we can also adopt ECC to improve PCM lifetime, similar to the method that has been widely adopted by DRAM-based main memory.

Improving security

Due to its nonvolatility, PCM retains data even after the system loses power. As such, security attacks can obtain critical information by physically obtaining the PCM-based main memory and scanning the memory regions.

To address this issue, researchers first proposed an improved 'counter-mode'-based encryption method [64]. As shown in Fig. 2, the improved 'counter-mode' method is compared to the traditional one. A word-level counter is proposed so that only the counter of words written is updated. Using this method can reduce the write intensity to NVM memory substantially.

Chhabrhabra and Solihin employ AES encryption to protect the data stored in PCM [65]. However, AES introduces two disadvantages: (1) long latency and high energy consumption and (2) the write amplification can significantly increase the number of PCM writes and in turn reduce PCM lifetime. Zhang *et al.* propose a XOR-PAD-based encryption method [66]. This stores part of the encryption information in volatile memory. In addition, it can leverage wear-leveling mechanisms to further improve security. The cost of such a solution is less than 1% on average without extra PCM writes.

Recently, researchers have proposed to leverage the sneak-path of NVM for encryption [67]. The sneak-path is generated randomly during fabrication of the NVM array. Thus, by adjusting the programming voltage and pulse, data bits stored in NVM are randomly generated, which can be leveraged for encryption. Later, an encryption method called DEUCE was proposed to increase the efficiency of counter-mode encryption [68]. In this approach, two counters are employed. The leading counter (LCTR) is used to encrypted updated words in the memory line, while the trailing counter (TCTR) is used for unchanged ones. In addition, horizontal wear-leveling is leveraged to improve the lifetime of the NVM. Similarly, another technique is proposed to integrate encryption and wear-leveling so that both performance and lifetime are improved compared to the counter-mode approach [69].

NVM-based storage architecture

NAND flash has been widely used in SSD-based storage system designs. As an alternative storage technology, PCM has the potential to be used to develop storage components. Compared to NAND flash, PCM has much better energy efficiency and lifetime. In addition, PCM is byte-addressable and therefore allows finer granularity access than NAND flash. However, the storage density of PCM is lower than NAND flash. Thus, a reasonable solution is to use hybrid storage with two technologies.

NAND flash and PCM hybrid storage

Although PCM has much better performance, energy and lifetime properties than NAND flash, the manufacture of PCM is still in the early stages of development. PCM cannot completely replace NAND flash in SSD design. For example, early PCM designs typically have small capacity. Therefore, they are used for storing metadata. Recent studies employ PCM as log storage of flash SSD. In the hybrid storage system, NAND flash is used to store data while PCM is used to store log updates. Every erase unit of the NAND flash has its own PCM log regions. A key design challenge is therefore how to allocate the PCM log regions. The study adopts two mechanisms to perform the allocation. The first mechanism is static allocation, which allocates a given amount of PCM space for each erase unit. Such a mechanism is easy to implement; however, it can lead to unbalanced utilization of PCM. Another mechanism dynamically allocates PCM regions based on the use of the erase units. Such a mechanism can efficiently utilize the PCM regions, but it increases the implementation complexity. Evaluation results show that the proposed methods can effectively improve SSD performance, reduce system power and increase storage lifetime.

NEW ARCHITECTURES BASED ON EMERGING NVMS

In this section, we will investigate new architectures based on these emerging NVMs, which do not exist in traditional memory hierarchy or cannot be designed based on traditional memory technologies. Moreover, we will introduce new challenges introduced by using these emerging NVMs.

Persistent memory

NVM promises a new tier between memory and storage with attributes of both. With the memory attribute, it allows data access through a load/store interface; with the storage attribute, it ensures data persistence that is typically offered by storage systems (such as databases and file systems). We refer to this new tier as 'persistent memory'. Obviously, using NVM can increase the capacity of main memory with low standby power. In addition, the persistent memory can enable new features like instant power-on/off of a system while flushing data in memory back to storage. However, keeping persistent data in this tier introduces new challenges.

A persistent memory system needs to maintain multi-versioning and ordering control to preserve data integrity with atomic, consistent and durable updates, in the face of various types of system crashes or program errors. Multi-versioning ensures that at least one version of the data is valid in the event of a sudden system crash. Ordering control prevents caches and memory controllers from violating application-defined write orders.

Recent work on persistent memory has demonstrated much higher program throughput (up to 32× [71,72]). These studies operate directly on nonvolatile data that are accessible through the processor-memory bus, obviate the overhead of PCIe or SATA accesses and legacy block-oriented file-system interfaces, and update the persistent data structures at a fine granularity and high frequency without the need for batching. Unfortunately, these persistent memory methods still incur significant performance overheads due to multi-versioning with logging or copy-on-write (COW). These mechanisms explicitly allocate system memory to store logs or temporary data copies, used as an alternative version of the real in-memory data. Therefore, logging and COW increase the demands of memory space and memory traffic. Furthermore, most previous designs place rigorous ordering control over data updates, by adopting write-through caching or employing flush (`clflush`) and memory fence (`mfence`) instructions to flush all dirty cache lines following each persistent memory update. The ordering control cancels out the optimization of write-back caching and memory controller scheduling policies.

Two recent studies, Kiln [70] and FIRM [73], address this performance issue by rearchitecting the memory and storage stack. Kiln [70] leverages processor cache and main memory hierarchy to naturally maintain atomicity without performing logging or copy-on-write. As illustrated in Fig. 3, the main idea of the Kiln solution is to leverage hardware to directly perform *in-place updates* to the real data and accelerate cache flushes with *clean-on-commit*. The design consists of a nonvolatile last-level cache (NV cache) and a nonvolatile memory (NV memory). The NV cache contains one version of

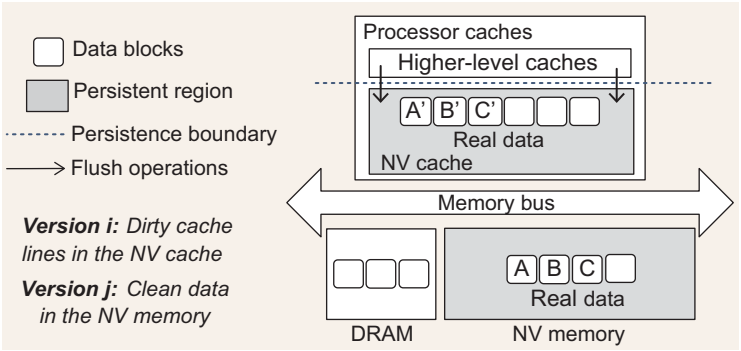

Figure 3. The architecture of Kiln design. Reproduced with permission from Ref [70].

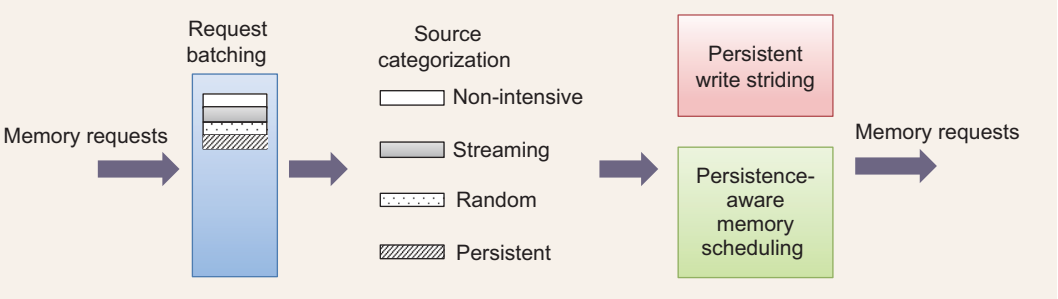

Figure 4. FIRM components. Reproduced with permission from Ref [73].

persistent data in the form of dirty cache lines. The other version is the clean data in the NV memory. Both versions are mapped to the same addresses so this persistent memory hierarchy directly updates real data structures rather than performing logging or COW. To preserve ordering, *clean-on-commit* operations only flush the particular dirty lines of an application-defined data update, from higher-level volatile caches down to the NV cache. As a result, Kiln can improve system throughput by more than 3×. FIRM [73] devised a memory control scheme that achieves both fair memory access and high system throughput in an NVM system shared by persistent and non-persistent applications. The inefficiency of prior memory scheduling schemes is rooted in the fact that they discriminate write requests and asynchronously service reads and writes. Such an asynchronous approach cannot address the disparate needs of persistent and non-persistent applications. As shown in Fig. 4, FIRM adopts two novel design principles to address such issues: First, *persistent write striding* ensures high bank-level parallelism among persistent writes such that memory bandwidth is well utilized; second, *persistence-aware memory scheduling* minimizes the frequency of write queue drains and bus turnarounds by scheduling the queued-up reads and writes in a synchronous manner. FIRM leads to both performance and fairness improvements of more than 20%, compared with the best of prior memory controller designs.

SCM file systems based on NVM

Recently, researchers have developed a few file systems based on storage-class memory (SCM), which are built on top of NVM. They have also demonstrated that the inefficiency in storage system software is becoming a critical performance bottleneck in NVM-based storage systems. In particular, the performance cost of I/O system software is much higher than PCM hardware access. Consequently, it is critical to coordinatively optimize storage system software and hardware. Moneta [74] provides a promising prototype system that emulates future high-performance storage systems built upon NVMs. It modifies the I/O system software that has been optimized for disk-based storage systems, by eliminating the locks that reduces the system parallelism. It also mitigates the performance overhead of resolving interrupts and improves read/write bandwidth by increasing hardware queue length. Evaluation results show that Moneta can improve the write throughput of 4 KB random access and 512 B write latency by 8× and 5.6× compared to state-of-the-art flash-based SSDs, respectively. Prior studies also show that a high-performance NV-heaps system can enable high-performance access to various data structures (e.g. B-tree, hash table, sparse graph and array) on NVM [72].

Wu and Reddy [75] implement a file system, SCMFS, in virtual address space. The file system leverages MMU to map file system address space to the SCM physical address space, leading to a simple address mapping. SCMFS also adopts the native memory manager of the OS to enforce that each file has a sequential virtual address space, reducing the access management of file systems. SCMFS also employs efficient address space allocation and garbage collection mechanisms to reduce the memory consumption. Jung *et al.* [76] have developed a hybrid file system, FRASH, which incorporates NAND flash and SCM to provide a large capacity for file systems. The design adopts SCM as a storage space yet maps it into the main memory address space to exploit its byte-addressability. In addition, FRASH stores critical data structures in the SCM to support fast access to the log structured file systems; it also improves the reliability of the file system by maintaining a snapshot of the file directory in the SCM. Such new file system designs optimize both the software/hardware interface and the file system, fully exploiting the benefits of SCM.

Nonvolatile processors

Since NVM technology can be applied to each level of the memory hierarchy, it is possible to design a processor with all memory based on NVM, which

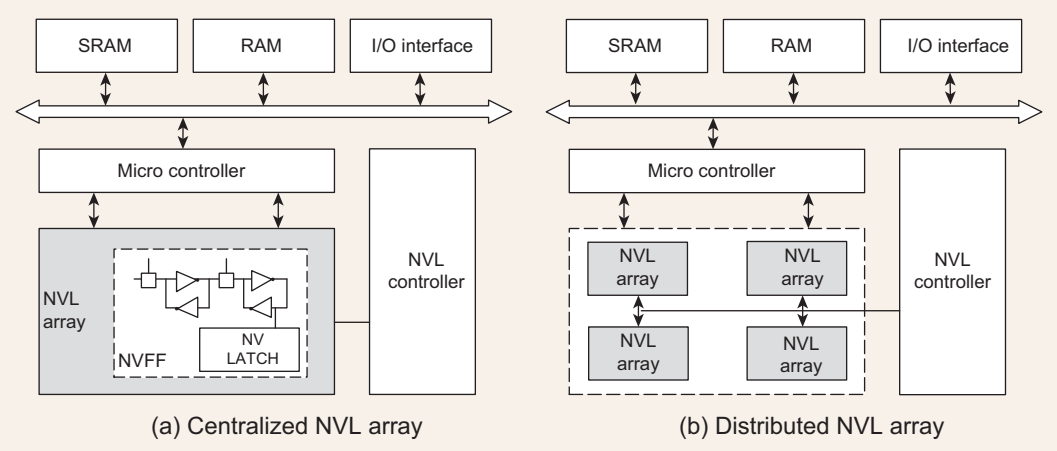

Figure 5. Overall structure of a nonvolatile processor.

is called a nonvolatile processor. With the increasing popularity of energy-harvesting systems, wearable devices and the Internet of things (IoT), nonvolatile processors have been widely investigated as promising candidates in modern self-powered devices. They can provide an ultra-low-power data processing system and work with unstable power supplies. The following works introduced in this subsection address how to reduce the area and explore the trade-off between performance and power.

Nonvolatile processor structure

In 2012, a fabricated nonvolatile processor based on ferroelectric flip-flops was presented [77]. It adopts a fully parallel read/write scheme to achieve a fast data backup/restore process and zero standby power, as shown in Fig. 5(a). Bartling *et al.* [78] demonstrate a distributed nonvolatile logic array design for data retention. Compared with the centralized array design, it saves the wakeup time as well as excessive routing and power costs. Sakimura *et al.* [79] integrate 3T-SpinRAM into a nonvolatile microcontroller consisting of three-terminal MTJ cells. The backup/restore energy is reduced to several nanoseconds per word. Singhal *et al.* [80] present an ultra-low-power microcontroller that addresses both active and standby power reduction. A shadow-latch-based MTCMOS is used in sequential cells like flip-flops to maintain low leakage with state retention during standby modes. Moreover, active power is also reduced by using the combination of standard-V_t and high-V_t transistors.

Area reduction techniques

It is well known that nonvolatile flip-flops introduce large area overheads and high cost. In order to mitigate this problem, plenty of strategies in the architecture level and compiler level have been proposed. A compare-and-compress recovery architecture consisting of a parallel run-length codec (PRLC) and a state table logic to reduce the area of nonvolatile registers has been proposed [81]. Later, a segment-based parallel compression architecture (SPaC) to achieve trade-offs between area and backup speed was presented [82]. At the software level, Zhao *et al.* proposed an NV register file reduction technique by stack utilization analysis and backup position optimizations [83].

Architecture and power system design

The power supply system and the architecture design for nonvolatile processors are also two hot topics. Ma *et al.* [84] analyze the design space of nonvolatile processors to determine optimal configurations for applications running on energy-harvesting platforms. Several selective backup policies that provide a trade-off between performance and the utilization of available energy for different CPU architectures are also discussed. As for the power supply module, several works have been proposed to improve the energy efficiency for nonvolatile processor platforms. Wang *et al.* [85] propose a storageless and converterless energy-harvesting system that performs the maximum power point tracking of PV cells. It integrates the nonvolatile processors to achieve extremely fast context saving and restoration after power interruptions.

Processing-in-memory

An early attempt at processing-in-memory was performed in the 1960s [86]. Specialized logic and modified decoders were augmented to the DRAM arrays to fulfill simple arithmetic operations at that time. In 1995, the term 'processing-in-memory' (PIM) was introduced by Terasys PIM design [87]. Simplified cores are implemented in the memory

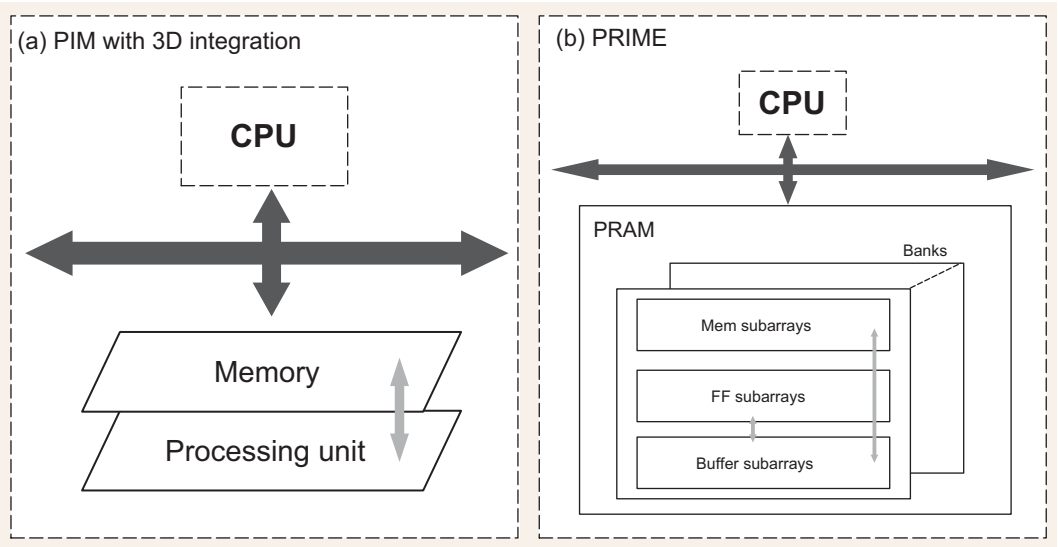

Figure 6. Typical PIM designs: (a) 3D-stacked PIM designs, (b) RRAM-based PIM design; PRIME.

die. However, this leads to reduced memory yield, because the optimization targets of high-speed logic and massive bytes per wafer are in conflict. To solve this problem, PIM designs using 3D-stacked DRAMs, as shown in Fig. 6(a), are now proposed. Those solutions have been proposed for neural networks [88], graph processing [89], Big Data processing [90], real-time analytics [91], sparse matrix multiplication [92] and in-memory databases [93].

However, the 3D-stacked PIM solutions generally suffer from high cost and thermal issues. To solve these problems, NVM-based PIM designs have recently been proposed, offloading logic into memory, leveraging the calculation capability of the memory itself. A famous example of RRAM-based PIM design is PRIME [94], as shown in Fig. 6(b). It designs full-function memory arrays to accommodate the computation while in computing mode, and switches the arrays back to memory mode when they are used only as memory. Due to the fact that the crossbar structure of RRAM is very suitable for the matrix multiplication, PRIME shows significant performance and energy benefits compared with its DRAM counterparts.

CONCLUSION

Compared to traditional memory technologies, emerging NVMs have advantages of low standby power, nonvolatility, high storage density, etc. They have the potential to replace traditional memory for future memory subsystem design. However, it is inefficient to directly adopt them in traditional memory architecture because they also have several limitations, such as high write latency/energy, limited write cycle endurance and vulnerability to security attacks. Thus, memory architecture modification is required to leverage the advantages of these NVMs and mitigate their drawbacks at the same time. In addition, to fully exploit the advantages of emerging NVMs, new architectures, such as persistent memory and SCM, should be introduced to traditional memory hierarchy. As a summary, we need to rethink the holistic memory architecture design to facilitate adoption of these emerging NVMs.

FUNDING

This work was supported in part by the National Natural Science Foundation of China (61572045).

REFERENCES

1. Bedeschi F, Fackenthal R and Resta C *et al.* A bipolar-selected phase change memory featuring multi-level cell storage. *IEEE J Solid State Circ* 2009; **44**: 217–27.
2. Burr GW, Kurdi BN and Scott JC *et al.* Overview of candidate device technologies for storage-class memory. *IBM J Res Dev* 2008; **52**: 449–64.
3. Oh J, Park JH and Lim Y *et al.* Full integration of highly manufacturable 512Mb PRAM based on 90nm technology. In: *Electron Devices Meeting 2006: IEDM'06*. IEEE International 2006, 1–4.
4. Pirovano A, Lacaita A and Benvenuti A *et al.* Scaling analysis of phase-change memory technology. In: *Electron Devices Meeting 2003: IEDM'03 Technical Digest*. IEEE International 2003, 29–6.
5. Lai S. Current status of the phase change memory and its future. In: *Electron Devices Meeting 2003: IEDM'03 Technical Digest*. IEEE International 2003, 10–1.
6. Cho S, Yi J and Ha Y *et al.* Highly scalable on-axis confined cell structure for high density PRAM beyond 256Mb. In: *2005 Symposium on VLSI Technology: Digest of Technical Papers*. IEEE 2005, 96–7.

7. Raoux S, Burr GW and Breitwisch MJ *et al.* Phase-change random access memory: a scalable technology. *IBM J Res Dev* 2008; **52**: 465–79.
8. Nirschl T, Phipp J and Happ T *et al.* Write strategies for 2 and 4-bit multi-level phase-change memory. In: *2007 IEEE International Electron Devices Meeting*.
9. Im D, Lee J and Cho S *et al.* A unified 7.5 nm dash-type confined cell for high performance PRAM device. In: *Electron Devices Meeting 2008: IEDM 2008*. IEEE International 2008, 1–4.
10. Osada K, Kotabe A and Matsui Y *et al.* A 512 kb embedded pram with 416 kbs write throughput at 100\(\ mu\) a cell write current. In: *IEEE International Solid-State Circuits Conference (ISSCC)*. 2007, 26–2.
11. Lee KJ, Cho BH and Cho WY *et al.* A 90 nm 1.8 V 512 Mb diode-switch PRAM with 266 MB/s read throughput. *IEEE J Solid State Circ* 2008; **43**: 150–62.
12. Bedeschi F, Fackenthal R and Resta C *et al.* A multi-level-cell bipolar-selected phase-change memory. In: *Solid-State Circuits Conference (ISSCC) 2008: Digest of Technical Papers*. IEEE 2008, 428–625.
13. De Sandre G, Bettini L and Pirola A *et al.* A 90nm 4Mb embedded phase-change memory with 1.2 V 12ns read access time and 1MB/s write throughput. In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC) 2010*. IEEE 2010, 268–9.
14. Villa C, Mills D and Barkley G *et al.* A 45nm 1Gb 1.8 V phase-change memory. In: *2010 IEEE International Solid-State Circuits Conference (ISSCC)* 2010. 2010, 270–1.
15. Hosomi M, Yamagishi H and Yamamoto T *et al.* A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM. In: *Electron Devices Meeting 2005: IEDM Technical Digest*. IEEE International 2005, 459–62.
16. Kishi T, Yoda H and Kai T *et al.* Lower-current and fast switching of a perpendicular TMR for high speed and high density spin-transfer-torque MRAM. In: *Electron Devices Meeting 2008: IEDM 2008*. IEEE International 2008, 1–4.
17. Motoyoshi M, Yamamura I and Ohtsuka W *et al.* A study for 0.18 μm high-density MRAM. In: *2004 Symposium on VLSI Technology: Digest of Technical Papers*. IEEE 2004, 22–3.
18. Kawahara T, Takemura R and Miura K *et al.* 2mb spin-transfer torque ram (spram) with bit-by-bit bidirectional current write and parallelizing-direction current read. In: *Solid-State Circuits Conference (ISSCC) 2007: Digest of Technical Papers*. IEEE 2007, 480–617.
19. Nebashi R, Sakimura N and Honjo H *et al.* A 90nm 12ns 32Mb 2T1MTJ MRAM. In: *Solid-State Circuits Conference (ISSCC) 2009: Digest of Technical Papers*. IEEE 2009, 462–3.
20. Kawahara T, Takemura R and Miura K *et al.* 2 Mb SPRAM (spin-transfer torque RAM) with bit-by-bit bi-directional current write and parallelizing-direction current read. *IEEE J Solid State Circ* 2008; **43**: 109–20.
21. NEC. http://www.mram-info.com/tags/companies/nec. 2011.
22. Yu LE, Kim S and Ryu MK *et al.* Structure effects on resistive switching of devices for RRAM applications. *IEEE Electron Device Lett* 2008; **29**: 331–3.
23. Pan F, Yin S and Subramanian V. A comprehensive simulation study on metal conducting filament formation in resistive switching memories. In: *3rd IEEE International Memory Workshop (IMW) 2011*. IEEE 2011, 1–4.
24. Kozicki MN, Balakrishnan M and Gopalan C *et al.* Programmable metallization cell memory based on Ag-Ge-S and Cu-Ge-S solid electrolytes. In: *Non-Volatile Memory Technology Symposium 2005*. IEEE 2005, 83–9.
25. Yang WY and Rhee SW. Effect of electrode material on the resistance switching of Cu2O film. *Appl Phys Lett* 2007; **91**: 232907.
26. Inoue IH, Yasuda S and Akinaga H *et al.* Nonpolar resistance switching of metal/binary-transition-metal oxides/metal sandwiches: homogeneous/inhomogeneous transition of current distribution. *Phys Rev B* 2008; **77**: 035105.
27. EE Times. https://www.eetimes.com/document.asp?doc_id=1172802&print=yes. 2010.
28. EE Times. https://www.eetimes.com/document.asp?doc_id=1171010. 2009.
29. EE Times. https://www.eetimes.com/document.asp?doc_id=1168871&print=yes. 2008.
30. Zhenyu S, Wenqing W and Li H. Cross-layer racetrack memory design for ultra high density and low power consumption. In: *Proceedings of the Design Automation Conference (DAC)*. Austin, TX: ACM 2013, 1–6.
31. Parkin SSP, Hayashi M and Thomas L. Magnetic domain-wall racetrack memory. *Science* 2008; **320**: 190–4.
32. Venkatesan R, Ramasubramanian SG and Venkataramani S *et al.* STAG. *ACM SIGARCH Computer Architecture News* 2014; **42**: 253–64. Available from: http://dl.acm.org/citation.cfm?id=2678373.2665710.
33. Iyengar AS, Ghosh S and Ramclam K. Domain wall magnets for embedded memory and hardware security. *IEEE J Emerg Sel Topics Circuits Syst* 2015; **5**: 40–50. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7036142.
34. Ranjan A, Ramasubramanian SG and Venkatesan R *et al.* DyReCTape: a dynamically reconfigurable cache using domain wall memory tapes. In: *Design, Automation Test in Europe Conference Exhibition (DATE) 2015*. 2015, 181–6.
35. Zhang C, Sun G and Zhang W *et al.* Quantitative modeling of racetrack memory, a tradeoff among area, performance, and power. In: *The 20th Asia and South Pacific Design Automation Conference*. IEEE 2015, 100–5. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7058988.
36. Kawahara A, Azuma R and Ikeda Y *et al.* An 8 Mb multi-layered cross-point ReRAM macro with 443 MB/s write throughput. *IEEE J Solid State Circ* 2013; **48**: 178–85.
37. Shenoy RS, Burr GW and Virwani K *et al.* MIEC (mixed-ionic-electronic-conduction)-based access devices for non-volatile crossbar memory arrays. *Semicond Sci Tech* 2014; **29**: 104005.
38. Niu D, Xu C and Muralimanohar N *et al.* Design of cross-point metal-oxide ReRAM emphasizing reliability and cost. In: *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2013, 17–23.
39. Wilton SJE and Jouppi NP. CACTI: an enhanced cache access and cycle time model. *IEEE J Solid State Circ* 1996; **31**: 677–88.
40. Li S, Ahn JH and Strong RD *et al.* McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In: *MICRO-42: 42nd Annual IEEE/ACM International Symposium on Microarchitecture 2009*. 2009, 469–80.
41. Mohan V, Bunker T and Grupp L *et al.* Modeling power consumption of NAND flash memories Using FlashPower. *IEEE Trans Comput-Aided Design Integr Circuits Syst* 2013; **32**: 1031–44.
42. Xu C, Dong X and Jouppi NP *et al.* Design implications of memristor-based RRAM cross-point structures. In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*; 2011, 1–6.
43. Jiang L, Zhao B and Yang J *et al.* A low power and reliable charge pump design for phase change memories. In: *ACM/IEEE 41st International Symposium on Computer Architecture (ISCA) 2014*. 2014, 397–408.
44. Dong X, Xu C and Xie Y *et al.* NVSim: a circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Trans Comput-Aided Design Integr Circuits Syst* 2012; **31**: 994–1007.
45. Poremba M, Zhang T and Xie Y. NVMain 2. 0: architectural simulator to model (non-)volatile memory systems. *IEEE Comput Architect Lett* 2015; **14**: 1.
46. Zhou P, Zhao B and Yang J *et al.* Energy reduction for STT-RAM using early write termination. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2009: Digest of Technical Papers*. IEEE 2009, 264–8.

47. Sun G, Dong X and Xie Y *et al.* A novel architecture of the 3D stacked MRAM L2 cache for CMPs. In: *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA) 2009*. IEEE 2009, 239–49.
48. Li Q, Li J and Shi L *et al.* Compiler-assisted STT-RAM-based hybrid cache for energy efficient embedded systems. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 2014; **22**: 1829–40.
49. Wang Z, Jiménez DA and Xu C *et al.* Adaptive placement and migration policy for an STT-RAM-based hybrid cache. In: *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA) 2014*. IEEE 2014, 13–24.
50. Sun Z, Bi X and Li HH *et al.* Multi retention level STT-RAM cache designs with a dynamic refresh scheme. In: *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM 2011, 329–38.
51. Sun G, Zhang Y and Wang Y *et al.* Improving energy efficiency of write-asymmetric memories by log style write. In: *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*. ACM 2012, 173–8.
52. Motaman S, Iyengar AS and Ghosh S. Domain wall memory-layout, circuit and synergistic systems. *IEEE Trans Nanotechnol* 2015; **14**: 282–91. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7006702.
53. Mao M, Wen W and Zhang Y *et al.* Exploration of GPGPU register file architecture using domain-wall-shift-write based racetrack memory. In: *Proceedings of the 51st Annual Design Automation Conference on Design Automation Conference (DAC '14)*. New York: ACM Press 2014, 1–6. Available from: http://dl.acm.org/citation.cfm?doid=2593069.2593137.
54. Haifeng X, Yong L and Melhem R *et al.* Multilane racetrack caches: improving efficiency through compression and independent shifting. In: *The 20th Asia and South Pacific Design Automation Conference 2015*, 1–6. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7059042.
55. Iyengar A and Ghosh S. Modeling and analysis of domain wall dynamics for robust and low-power embedded memory. In: *Proceedings of the 51st Annual Design Automation Conference on Design Automation Conference (DAC '14)*. Vol. 1. New York: ACM Press 2014, 1–6. Available from: http://dl.acm.org/citation.cfm?doid=2593069.2593161.
56. Lee BC, Ipek E and Mutlu O *et al.* Architecting phase change memory as a scalable dram alternative. In: *ISCA*. 2009, 2–13.
57. Zhou P, Zhao B and Yang J *et al.* A durable and energy efficient main memory using phase change memory technology. In: *ISCA*. 2009, 14–23.
58. Qureshi MK, Srinivasan V and Rivers JA. Scalable high performance main memory system using phase-change memory technology. In: *ISCA*. 2009, 24–33.
59. Joshi M, Zhang W and Li T. Mercury: a fast and energy-efficient multi-level cell based phase change memory system. In: *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA) 2011*. 2011, 345–56.
60. Qureshi MK, Franceschini MM and Lastras-Montano LA. Improving read performance of phase change memories via write cancellation and write pausing. In: *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA) 2010*. 2010, 1–11.
61. Jacobvitz AN, Calderbank R and Sorin DJ. Coset coding to extend the lifetime of memory. In: *IEEE 19th International Symposium on High Performance Computer Architecture 2013 (HPCA2013)*. 2013, 222–33.
62. Qureshi MK, Karidis J and Franceschini M *et al.* Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In: *MICRO*. 2009, 14–23.
63. Seong NH, Woo DH and Lee HHS. Security refresh: protecting phase-change memory against malicious wear out. *IEEE Micro* 2011; **31**: 119–27.
64. Kong J and Zhou H. Improving privacy and lifetime of PCM-based main memory. In: *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) 2010*. 2010, 333–42.
65. Chhabra S and Solihin Y. i-NVMM: a secure non-volatile main memory system with incremental encryption. In: *Proceedings of the 38th Annual International Symposium on Computer Architecture: ISCA '11*. New York: ACM 2011, 177–88. Available from: http://doi.acm.org/10.1145/2000064.2000086.
66. Zhang X, Zhang C and Sun G *et al.* An efficient run-time encryption scheme for non-volatile main memory. In: *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*. IEEE 2013, 1–10.
67. Kannan S, Karimi N and Sinanoglu O. Secure memristor-based main memory. In: *51st ACM/EDAC/IEEE Design Automation Conference (DAC) 2014*. IEEE 2014, 1–6.
68. Young V, Nair PJ and Qureshi MK. DEUCE: write-efficient encryption for non-volatile memories. In: *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM 2015, 33–44.
69. Liu C and Yang C. Secure and durable (SEDURA): an integrated encryption and wear-leveling framework for PCM-based main memory. In: *Proceedings of the 16th ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2015 CD-ROM*. ACM 2015, 12.
70. Zhao J, Li S and Yoon DH *et al.* KiIn: closing the performance gap between systems with and without persistence support. In: *MICRO*. 2013, 421–32.
71. Volos H, Tack AJ and Swift MM. Mnemosyne: lightweight persistent memory. In: *ASPLOS*. 2011, 91–104.
72. Coburn J, Caulfield AM and Akel A *et al.* NV-heaps: making persistent objects fast and safe with next-generation, non-volatile memories. In: *ASPLOS*. 2011, 105–18.
73. Zhao J, Mutlu O and Xie Y. FIRM: fair and high-performance memory control for persistent memory systems. In: *International Symposium on Microarchitecture*. 2014, 153–65.
74. Caulfield AM, De A and Coburn J *et al.* Moneta: a high-performance storage array architecture for next-generation, non-volatile memories. In: *43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) 2010*. 2010, 385–95.
75. Wu X and Reddy ALN. SCMFS: a file system for storage class memory. In: *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 2011, 1–11.
76. Jung J, Won Y and Kim E *et al.* FRASH: exploiting storage class memory in hybrid file system for hierarchical storage. *ACM Trans Storage* 2010; 1–25.
77. Wang Y, Liu Y and Li S *et al.* A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In: *Proceedings of the ESSCIRC 2012*. IEEE 2012, 149–52.
78. Bartling SC, Khanna S and Clinton MP *et al.* An 8MHz 75μA/MHz zero-leakage non-volatile logic-based Cortex-M0 MCU SoC exhibiting 100% digital state retention at V DD= 0V with $<$ 400ns wakeup and sleep transitions. In: *IEEE International Solid-State Circuits Conference (ISSCC) 2013: Digest of Technical Papers*. IEEE 2013, 432–3.
79. Sakimura N, Tsuji Y and Nebashi R *et al.* 10.5 a 90nm 20mhz fully nonvolatile microcontroller for standby-power-critical applications. In: *IEEE International Solid-State Circuits Conference (ISSCC) 2014: Digest of Technical Papers*. IEEE 2014, 184–5.
80. Singhal VK, Menezes V and Chakravarthy S *et al.* 8.3 A 10.5μA/MHz at 16MHz single-cycle non-volatile memory access microcontroller with full state retention at 108nA in a 90nm process. In: *IEEE International Solid-State Circuits Conference (ISSCC) 2015*. IEEE 2015, 1–3.
81. Wang Y, Liu Y and Liu Y *et al.* A compression-based area-efficient recovery architecture for nonvolatile processors. In: *EDA Consortium Proceedings of the Conference on Design, Automation and Test in Europe*. 2012, 1519–24.

82. Sheng X, Wang Y and Liu Y *et al*. Spac: a segment-based parallel compression for backup acceleration in nonvolatile processors. In: *Design, Automation & Test in Europe Conference & Exhibition (DATE) 2013*. IEEE 2013, 865–8.
83. Zhao M, Li Q and Xie M *et al*. Software assisted non-volatile register reduction for energy harvesting based cyber-physical system. In: *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium 2015, 567–72.
84. Ma K, Zheng Y and Li S *et al*. Architecture exploration for ambient energy harvesting nonvolatile processors. In: *IEEE 21st International Symposium on High Performance Computer Architecture (HPCA) 2015*. IEEE 2015, 526–37.
85. Wang Y, Liu Y and Wang C *et al*. Storage-less and converter-less photovoltaic energy harvesting with maximum power point tracking for internet of things. *IEEE Trans Comput-Aided Design Integr Circuits Syst* 2016; **35**: 173–86.
86. Siegl P, Buchty R and Berekovic M. Data-centric computing frontiers : a survey on processing-in-memory. In: *Memsys*. 2016, 295–308.
87. Gokhale M, Holmes B and Iobst K. Processing in memory: the terasys massively parallel PIM array. *Computer* 1995; **28**: 23–31.
88. Shafiee A, Nag A and Muralimanohar N *et al*. ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In: *ISCA*. 2016, 14–26.
89. Ahn J, Hong S and Yoo S *et al. A scalable processing-in-memory accelerator for parallel graph processing*. 2015; **43**: 105–17.
90. Pugsley SH, Jestes J and Zhang H *et al.* NDC: analyzing the impact of 3D-stacked memory+logic devices on MapReduce workloads. In: *ISPASS 2014: IEEE International Symposium on Performance Analysis of Systems and Software*. 2014, 190–200.
91. Guz Z, Awasthi M and Balakrishnan V *et al*. Real-time analytics as the killer application for processing-in-memory. In: *Near Data Processing (WoNDP)*. 2014, 10–2.
92. Zhu Q, Graf T and Sumbul HE *et al*. Accelerating sparse matrix-matrix multiplication with 3D-stacked logic-in-memory hardware. In: *2013 IEEE High Performance Extreme Computing Conference: HPEC 2013*. 2013, 1–6.
93. Lim K, Meisner D and Saidi AG *et al*. Thin servers with smart pipes: designing SoC accelerators for memcached. In: *ISCA'13*. 2013, 36–47.
94. Chi P, Li S and Xu C *et al. PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory*. 2016; 27–39.