Emerging NVM: A Survey on Architectural Integration and Research Challenges

JALIL BOUKHOBZA, b<>com Research Institute of Technology, Univ. Bretagne Occidentale, UMR 6285, Lab-STICC, The Hong Kong Polytechnic University
STÉPHANE RUBINI, Univ. Bretagne Occidentale, UMR 6285, Lab-STICC
RENHAI CHEN, Tianjin University
ZILI SHAO, The Hong Kong Polytechnic University

There has been a surge of interest in Non-Volatile Memory (NVM) in recent years. With many advantages, such as density and power consumption, NVM is carving out a place in the memory hierarchy and may eventually change our view of computer architecture. Many NVMs have emerged, such as Magnetoresistive random access memory (MRAM), Phase Change random access memory (PCM), Resistive random access memory (ReRAM), and Ferroelectric random access memory (FeRAM), each with its own peculiar properties and specific challenges. The scientific community has carried out a substantial amount of work on integrating those technologies in the memory hierarchy. As many companies are announcing the imminent mass production of NVMs, we think that it is time to have a step back and discuss the body of literature related to NVM integration. This article surveys state-of-the-art work on integrating NVM into the memory hierarchy. Specially, we introduce the four types of NVM, namely, MRAM, PCM, ReRAM, and FeRAM, and investigate different ways of integrating them into the memory hierarchy from the horizontal or vertical perspectives. Here, horizontal integration means that the new memory is placed at the same level as an existing one, while vertical integration means that the new memory is interleaved between two existing levels. In addition, we describe challenges and opportunities with each NVM technique.

CCS Concepts: • **Information systems → Storage class memory**; **Phase change memory**; **Hierarchical storage management**; • **Hardware → Non-volatile memory**; **Memory and dense storage**; • **Software and its engineering → Memory management**;

Additional Key Words and Phrases: Non-volatile memory, MRAM, PCM, FeRAM, ReRAM, storage, main memory

ACM Reference format:
Jalil Boukhobza, Stéphane Rubini, Renhai Chen, and Zili Shao. 2017. Emerging NVM: A Survey on Architectural Integration and Research Challenges. *ACM Trans. Des. Autom. Electron. Syst.* 23, 2, Article 14 (November 2017), 32 pages.
https://doi.org/10.1145/3131848

Authors' addresses: J. Boukhobza and S. Rubini are with Lab-STICC UMR 6285, Université de Bretagne Occidentale, 20 avenue Le Gorgeu, 29200 Brest, France; emails: {boukhobza; rubini}@univ-brest.fr; Z. Shao is with Embedded Systems and CPS Laboratory, Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong; email: cszlshao@comp.polyu.edu.hk; R. Chen is with Tianjin Key Laboratory of Cognitive Computing and Application, School of Computer Science and Technology, Tianjin University, Peiyang Park Campus, Tianjin, China; email: crh6250790@gmail.com.

https://doi.org/10.1145/3131848

1 INTRODUCTION

Volume, performance, energy efficiency, and scalability are key factors that designers of memory hierarchies have to deal with to satisfy the ever-increasing need for current data intensive applications.

Volume. The amount of digital data produced today, in different forms, is growing exponentially (Ranganathan 2011; Winter 2008). While for transistor integration, Moore's law gives an impressive example of increasing process, the amount of data being produced is growing at a much faster rate. In a recent study, EMC forecasted growth by a factor of 44 in the amount of digital data created per year from 2009 to 2020. This is mainly due to the growth in the number of devices (and sensors within devices) that generates data—around 11 billion devices are connected to the Internet to date. In fact, applications are becoming more and more data-intensive with the increasing popularity of social networking, video playbacks, and transaction processing, and large memory volumes are needed to store and process the data generated from those activities.

Performance. Efficiently processing the data that are generated has become a major economic and social issue. Better analyzing of data means better results, processes, and decisions. It can help in efforts to generate new ideas and solutions, or in attempts to more accurately predict future events (weather forecasts, disease propagation, etc.). Much effort has been expended to provide more processing power for both the online and offline manipulation of these large amounts of data (Mutlu 2015). However, this is putting a great deal of pressure on the memory hierarchy to bridge the historical gap with the processing elements from the performance point of view.

Energy. In addition to performance and capacity, energy efficiency is a critical metric to consider when designing memory systems. Indeed, one of the most challenging technological innovations for the Exascale computing systems planned for 2020 is the management and optimization of power consumption (Kogge et al. 2008). Data centers in the U.S. consume more than 1.5% of the energy generated in the country, and this ratio is projected to increase annually by 18% (Zhang et al. 2010). A large proportion of this energy is due to memory systems. The energy consumption of storage systems is estimated to comprise between 20% and 40% of the total energy consumption of a typical data center (Carter and Rajamani 2010). Other studies gave different numbers, for instance, a dynamic random access memory (DRAM) main memory subsystem consumes between 30% and 50% of an HPC (High Performance Computing) node, even though this ratio depends on the capacity and configuration of the DRAM (Vetter and Mittal 2015).

Scalability. DRAM technology has undergone impressive improvements in terms of size, capacity, and performance; however, many studies predict that DRAM scaling trends will hit a plateau the next 5 to 10 years. DRAM cells need to be large enough for reliable sensing, but this goes against the trend of shrinking feature sizes. As a consequence, high-density DRAMs impose an exponential cost penalty (e.g., in 2008, using 1 DIMM 8GB cost 212$/GB while 2 × 4GB DIMMS cost 50$/GB, and 4 × 2GB DIMMS cost 15$/GB (Mogul et al. 2009)). DRAM cells also need to be periodically refreshed, thus consuming power even when no read/write operations are performed; such power consumption has been proven to be between 19% and 31% of peak power (Mogul et al. 2009). Other problems related to disturbance issues and reliability for features that are very reduced in size pose great difficulties for future DRAM designs (Yoongu et al. 2014; Vetter and Mittal 2015). Therefore, it is becoming more and more difficult to increase the scaling of charge-based technologies such as DRAM, as well as flash memories (Mutlu 2015).

In this context, emerging non-volatile solid-state memories (NVMs) promise to revolutionize the memory hierarchy. They can provide very high density, almost zero static power consumption, and high endurance. Many emerging NVM technologies are being extensively studied, such as Phase Change Memory (called PRAM or PCM), Resistive Memory (ReRAM), Spin Torque Transfer

Memory (STT-RAM), and Ferroelectric memories (FeRAM). Several companies have announced the imminent mass production of such technologies.

The most popular established NVM is NAND flash memory. It took advantage of the surge in the use of smartphones at the beginning of the 2000s and is now ubiquitous in many applicative domains such as embedded systems and high performance computing. It has become an indispensable technology to partly bridge the gap between DRAM and storage performance.

As with flash memory a decade ago, NVMs are attracting a great deal of interest and much work is being conducted on the issue of how different technologies can be integrated in the memory hierarchy. The numerous announcements from different companies seeking to mass produce NVMs justifies the need to take a step back to discuss and classify the options for integration that have been investigated in state-of-the-art work.

The objective of this survey is threefold: (1) To shed some light on the characteristics of NVMs; (2) To classify state-of-the-art work according to architectural integration options—more precisely, horizontal, or vertical integration. Horizontal integration means that the NVM is put at the same level as an existing memory technology (e.g., in cache with SRAM), while vertical integration means that the new memory is interleaved between two existing levels (e.g., between DRAM and traditional storage devices); (3) To discuss the main challenges involved in making those NVMs ready for use.

There are other recent surveys about emerging NVMs. In Yu and Chen (2016), the authors present a mainly technological and low-level description of different types of NVM. Some other very interesting surveys such as that by Mittal et al. (2015b) are more about a specific level of the memory hierarchy (in this case, about cache), or about the software optimizations that were employed (Mittal and Vetter 2016). Other surveys such as that by Xia et al. (2015) are exhaustive and related to one technology (here, PCM). Different from and complementary to previous surveys, ours focuses on architectural integration issues for four NVM technologies: PCM, MRAM, FeRAM, and ReRAM.[1]

The remainder of the article is organized as follows: Section 2 lays the ground for a discussion of NVM integration. Sections 3, 4, 5, and 6 discuss PCM, MRAM, FeRAM, and ReRAM technologies, respectively. In each section, we define one of these technologies and discuss options for its integration. We will highlight the main issues discussed in some representative research work.

2 BACKGROUND

In this section, we first introduce the memory hierarchy and discuss about integration options. Then, we present the example of flash memory integration, which is considered as the last major disruptive memory technology. We end up this section by introducing NVM properties and integration options.

2.1 Overview on Memory Hierarchy

Memories are a key component of computer systems, and both their function and integration into systems have evolved considerably over time. In effect, a large set of devices and technologies have been developed.

The design of modern computers introduced the first division in the management of memory hierarchy management: a fast memory that supports the high-speed movement of data, which is indispensable for running applications and is called primary or main memory; and a memory used for mass storage. The latter supplies the former, thanks to a lower cost data storage property,

[1] Note that in this survey, Domain Wall Memory (DMW) will not be investigated, and we suggest readers to consult the interesting survey in Mittal (2016).

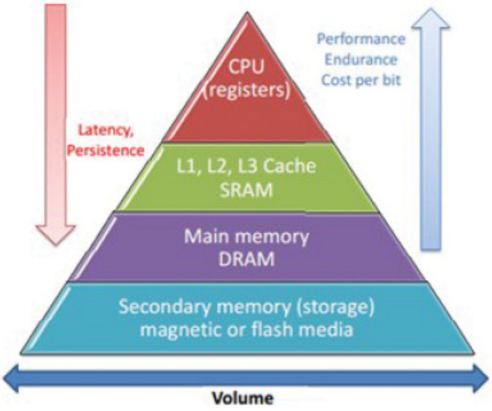


Fig. 1. The memory hierarchy pyramid.

which is called secondary memory or storage memory. This hardware classification of memories has translated into different management policies at the operating system level, as different services manage those memories. Indeed, a specific service manages the primary memory and shares this resource among different application processes, while the storage system is managed as a peripheral, more precisely, as a block device.

The position of a given memory technology in the memory hierarchy pyramid (see Figure 1) reflects the distance to the main processing unit. The better its performance in terms of bandwidth and endurance, the closer it is architected with respect to the processing unit; the poorer its performance, the farther it is architected with respect to the processing unit.

From an architectural point of view, one might think about inserting a given NVM technology in any location within the memory hierarchy pyramid solely according to its characteristics in terms of technology process, bandwidth, and endurance. From a quantitative point of view, the integration would depend on some cost considerations. Generally speaking, the greater the performance of a given memory technology, the higher the cost and the closer to the CPU it is integrated. For instance, an example of integration is under the form of a cache level. SRAM caches are memories that are closer to the CPU.

However, the situation is different from both the applicative and operating system points of view. For instance, from the operating systems perspective, even though many memory components are integrated in a computer, it is mainly the following three that are recognized:

(1) The cache system, which can be activated or deactivated, but for which no explicit management is provided, as the CPU cache is managed by a specific hardware component.
(2) The main memory, which is byte addressable and is managed as a resource with a dedicated service in the operating system. With regard to the latter, the system relies on some hardware components to accelerate its management (for instance, the Translation Lookaside Buffer, TLB).
(3) Finally, the storage system, whether it be a traditional magnetic disk or a flash-based storage drive (Solid State Drive, or SSD). The storage device is a block-addressed device managed as a peripheral from the operating system perspective.

From the architectural point of view, the integration of NVM can be performed **vertically** or **horizontally**.

Vertical integration means that the integrated NVM will shift a given memory technology down in the memory hierarchy and replace it. In this case, the used access method is the same as the one of the replaced memory technology. For instance, in case DRAM is replaced, the NVM is accessed with the same way as DRAM is (from an application level).

Horizontal integration means supplementing a given memory technology in the hierarchy (Vetter and Mittal 2015). In horizontal integration, the newly integrated NVM is interfaced in the same way as the existing memory at the same level. For instance, in case of a horizontal integration within the main memory, some specific application programming interfaces are needed so an application can take benefit of memory heterogeneity. Some standardization organisms such as SNIA already began working on the question throughout the "NVM Programming Model (NPM)" (SNIA 2015). This document claims that this kind of new architecture needs some specific API, but the standard POSIX API is supported as well. This means that a system needs to provide two APIs, one allowing the programmer to be aware of the presence of NVM while the other (standard POSIX) hides the memory heterogeneity.

Whether the integration is horizontal or vertical, hardware (controller) and/or software (operating system and/or applications) components must be aware of the new memory stack for performance optimization reasons.

To fully integrate NVMs, one must consider hardware integration and software integration. The former answers the question about the horizontal or vertical integration and how NVM should interface with the rest of the system, while the latter is related to the system and to application upgrades to make full profit of the additional memory technology.

2.2 Flash Memory, a Pioneer NVM

Flash memory has been a disruptive technology as it has allowed discussions to be raised about the standard storage integration on operating systems, namely the block I/O layer (Bjørling et al. 2013; Ouyang et al. 2011).

Flash memory displays high density, consumes little power when idle, and provides faster access time than HDD, especially for random reads. But to be efficiently integrated in a given storage system, one must take into consideration its peculiarities: read/write performance asymmetry, a limited lifetime, write/erase operations granularity asymmetry, and so on.

2.1.1 Flash Memory Integration. Flash memory can be integrated into storage servers in three ways (Roberts et al. 2009; Boukhobza 2013): (1) as an extension of the memory system (horizontal integration), (2) as a storage accelerator where flash is managed as a cache storing frequently accessed code and data and so reducing the number of I/Os to HDD (vertical integration), or (3) as an alternative storage device where flash SSDs are used as a replacement or to complement HDDs (horizontal integration again). Of the above three options, two are horizontal integrations but are placed at two different levels: main or secondary memory. From the applicative point of view, in the case where flash is integrated with DRAM, the integration is transparent to the applications as the flash memory is managed through dedicated hardware to supplement DRAM (e.g., NVDIMM (Huang and Jiang 2014; Chen et al. 2016)). In the second case of horizontal integration, the flash memory is revealed to the rest of the system through a traditional storage system interface and is thus accessed like an HDD through standard file system interfaces. Flash memory has also been integrated through a PCIe interface, allowing more room for performance enhancement by making use of internal parallelism that can be achieved with multi-channel and other multi-die, and plane parallelism. This integration through PCIe can be storage stack compliant and thus seen as a storage device, or non-compliant and thus needing some specific driver and software tools to be accessed by applications. The objective of the latter is to get rid of traditional storage stack latencies through specialized software.

Because of cost considerations (flash is roughly one order of magnitude more expensive than HDDs), flash-based storage systems will not replace disk systems in the medium term (Gurumurthi 2009). Thus, disk storage systems are not about to disappear from the memory hierarchy.

2.1.2 Flash Memory Properties and Management. As compared to traditional magnetic storage systems, NAND flash memories have some peculiar characteristics that have pushed the scientific community to design specific solutions for their integration (both horizontal and vertical). Here is a set of flash memory characteristics taken from (Boukhobza 2013; Boukhobza and Olivier 2017):

—**Random reads**: Flash memories are entirely electronic-based components with no moving mechanical parts. Contrary to HDDs, read access latency no longer depends on the physical location of data, so random reads perform as well as sequential ones. This makes flash memories more efficient and predictable for read operations as compared to traditional HDDs. The poor performance of HDDs in cases of random access has led to the implementation of many mechanisms at the operating system level to try to minimize this performance gap: I/O scheduler, page cache, and others. At the user layer, many applications also suppose an underlying HDD and thus try to maximize sequential requests while minimizing random ones such as database management systems (e.g., sorting algorithms), and big data applications (e.g., MapReduce). In the case of flash memory integration, those mechanisms were revisited to take full advantage of flash memory performance and the scientific community has made significant contributions toward this end.

—**I/O performance asymmetry**: Another specificity of flash memories is that, unlike HDDs, reads are generally much faster than writes. This is mainly due to the electrical properties of the basic flash memory cell; it takes more time to program the cell to reach a stable state than simply to read it. Many flash-specific buffer system mechanisms have been designed to absorb write operations and so increase the performance.

—**Sequential writes in a block**: To avoid write errors (due to some electrical disturbance properties), writes must be performed sequentially within one block, especially for MLC (Multi Level Cell) technology. Error correction codes (ECC) are implemented in hardware to cope with such errors.

—**Low power consumption**: Even though the power consumption of flash-based storage systems depends on its internal architecture and on the applied workload (Park et al. 2011), it remains more energy efficient than HDDs. As the roadmap of high performance computing integrates more and more constraints on energy efficiency, such a storage system technology will be more sought after than ever.

—**I/O interface**: To ease the adoption of flash memory, the integration of SSD as a replacement technology relies on the use of I/O interfaces similar to HDDs. This characteristic can be considered an advantage from the point of view of the user, as no modifications to the operating system are required (with the exception of drivers). However, this may give the illusion that one must deal with flash memories in the same way as with HDDs, which is far from true.

—**Shock resistance**: This characteristic is yet another consequence of the absence of mechanical parts.

A given set of constraints must be dealt with when designing flash-based solutions. These constraints can be summarized as follows:

—**Erase-before-write limitation:** This is one of the most critical constraints on flash memories. One cannot directly overwrite data. This constraint means that if the data of a given page need to be modified in-place, then the whole block should first be erased (see the previous constraint). This is time consuming; thus, data are generally updated out of place (after invalidating the old data) via a **mapping scheme**. This mapping scheme has a strong impact on the performance of the SSD. In addition to the mapping scheme, a flash memory controller must implement a **garbage collector** to recycle previously invalidated data.

- **Write/Erase granularity asymmetry**: Writes are performed on pages, while erase operations are realized on blocks. Flash memory blocks are composed of a given number of pages of a size between 2 and 8 KB (powers of 2).
- **Limited number of Erase/Write cycles**: The average number of write/erase (W/E) cycles is approximately 10^5 for a single level cell (SLC), 10^4 for a multi-level cell (MLC), and 5000 for a triple level cell (TLC). After the maximum number of erase cycles is reached, a given memory cell may no longer retain data. Due to this constraint on endurance, flash memory controllers implement **wear leveling techniques** to balance the wear out of the flash memory blocks.

Therefore, to be integrated with a traditional storage interface (just like an HDD), a flash memory controller should implement at least the three abovementioned services: a mapping scheme, a garbage collector, and a wear leveler. Those mechanisms have a very strong impact on performance.

2.3 NVM, Why and How?

From the architectural and system points of view, a new NVM can be inserted in the memory hierarchy in one of the three main subclasses of memory: processor caches, main memory, or storage systems. NVMs can be vertically or horizontally integrated.

2.2.1 As a Storage System. From the storage system point of view, NVM can be integrated horizontally. Therefore, similar to the flash memory, some constraint management mechanisms should be implemented and abstracted to higher levels so the new NVM-based storage system can be integrated in the traditional storage software stack. NVM can also be integrated vertically such as PCIe-based flash devices. Another possibility is to have some hybrid devices such as HDDs integrate flash memories, but this time with NVM technologies other than flash.

2.2.2 As a Main Memory. As many NVM technologies have very appealing performance characteristics (see Table 1), they can be integrated into a higher level in the memory hierarchy than the storage system. NVMs have some peculiar characteristics, such as a read performance that is higher than the write performance, an endurance that varies according to the technology and that is more impacted by write operations than by read operations, and the fact that most NVMs use only power when accessed. All of these characteristics must be dealt with and some new services must be included when integrating NVM as DRAM or next to DRAM to have some hybrid main memory.

Again, NVM can be integrated vertically with DRAM or horizontally (Li et al. 2012). In the first case, DRAM can be regarded as a cache for the NVM to reduce the access latency to the primary memory. NVM can also be integrated horizontally on the same bus as the memory. Data placement issues could be managed by hardware and thus abstracted to the operating system, or managed by the operating system and abstracted to the application. Finally, some interface could be made available to the application by the operating system, to facilitate decisions on placements at the applicative layer. In such cases, operating system support is a critical issue when dealing with memory heterogeneity.

Table 1. Characteristics of NVMs According to State-of-the-art Studies[2] (Vetter and Mittal 2015; Mittal et al. 2015b; Xia et al. 2015; Wang et al. 2014a; Suresh et al. 2014; Baek et al. 2013; Meena et al. 2014)

	SRAM	DRAM	HDD	NAND flash	STT-RAM	ReRAM	PCM	FeRAM
Cell size (F^2)	120–200	60–100	N/A	4–6	6–50	4–10	4–12	6–40
Write Endurance	10^{16}	$>10^{15}$	$>10^{15}$ (pb: mechanical parts)	10^4–10^5	10^{12}–10^{15}	10^8–10^{11}	10^8–10^9	10^{14}–10^{15}
Read Latency	~0.2–2ns	~10ns	3–5ms	15–35μs	2–35ns	~10ns	20–60ns	20–80ns
Write Latency	~0.2–2ns	~10ns	3–5ms	200–500μs	3–50ns	~50ns	20–150ns	50–75ns
Leakage Power	High	Medium	(Mechanical parts)	Low	Low	Low	Low	Low
Dynamic Energy (R/W)	Low	Medium	(Mechanical parts)	Low	Low/High	Low/High	Medium/ High	Low/High
Maturity	Mature	Mature	Mature	Mature	Test chips	Test chips	Test chips	Manufactured

2.2.3 As a Processor Cache. From the CPU cache perspective, the first-level cache is generally accessed at a high frequency, so very low latency and high endurance is required. This is hard to achieve with most NVMs, even if some of them can be considered candidates for such integration. On the other hand, last-level caches are designed more to reduce off-chip data movements and thus must have high density (to achieve large capacity). NVMs integrated in CPU caches are intended more for use in the last-level cache than in other levels of cache (Mittal et al. 2015b). However, even in this case, NVMs will absorb very large amounts of write operations, so issues of wear must be mitigated. In addition, the compatibility of the technology with the fabrication process is also a very important issue.

Table 2 summarizes state-of-the-art work on the NVM integration with regard to the memory hierarchy. As previously discussed, NVM can be integrated horizontally with some other technology; in this case some data migration mechanisms need to be provided whether at the hardware, operating system, or applicative level. NVM can be integrated vertically by shifting another memory down, or it can replace an existing memory technology. In Table 2, in the case of the processor cache, the line "replacement" lists studies where all of the levels of cache have been replaced by an NVM.

It should be noted in Table 2 that the most studied options are the vertical integration of MRAM in processor caches and the horizontal integration of PCM in main memory, with those two technologies being the most investigated in state-of-the-art work.

Note that for many studies related to PCM (or ReRAM), the proposed mechanisms can also work for ReRAM (or PCM).

In the rest of the article, we will investigate each of the following NVMs: PCM, MRAM, FeRAM, and ReRAM, and their integration.

3 PCM

As one can observe in Table 1, phase change memory (PCM), also called phase change random access memory (PRAM), has small-sized cells and excellent scalability within the CMOS fabrication process, relatively fast random access, a moderate write throughput, a good retention time, very

[2] The values are not strict values, but give an overall idea of the characteristics.

Table 2. Classification of State-of-the-art Studies about NVM Integration

		MRAM	ReRAM	PCM	FeRAM
Cache (different levels)	Horizontal	(Oboril et al. 2015; Li et al. 2012; Syu et al. 2013; Li et al. 2014; Wu et al. 2009; Jadidi et al. 2011; Li et al. 2011; Wang et al. 2014 ; Komalan et al. 2014; Cheng et al. 2016)	(Wang et al. 2014a; Komalan et al. 2013; Mittal and Vetter 2015)	(Wu et al. 2009; Joo et al. 2010)	
	Vertical	(Oboril et al. 2015; Senni et al. 2014; Sun et al. 2009; Wu et al. 2009; Samavatian et al. 2014; Smullen et al. 2011; Jog et al. 2012; Zhou et al. 2009b; Goswami et al. 2013; Yazdanshenas et al. 2014; Ahn et al. 2012; Rasquinha et al. 2010; Park et al. 2012; Chen et al. 2013; Kwon et al. 2014; Jokar et al. 2016; Senni et al. 2015; Cheng et al. 2016)	(Dong et al. 2013; Wang et al. 2013; Jokar et al. 2016)	(Wu et al. 2009)	
	Replacement	(Oboril et al. 2015; Smullen et al. 2011; Sun et al. 2011; Guo et al. 2010; Goswami et al. 2013; Wang et al. 2015)	(Dong et al. 2013)		
Main Memory	Horizontal	(Yang et al. 2013; Suresh et al. 2014; Wei et al. 2015)	(Hassan et al. 2015; Wei et al. 2015)	(Dhiman et al. 2009; Park et al. 2010; Bock et al. 2011; Suresh et al. 2014; Zhou et al. 2009a; Sun et al. 2015; Wei et al. 2015; Lee et al. 2014; Salkhordeh and Asadi 2016; Wei et al. 2015; Kannan et al. 2016; Dulloor et al. 2016; Wu et al. 2016 ; Li et al. 2012 ; Oikawa 2014; Gao et al. 2015)	(Doh et al. 2007; Suresh et al. 2014)
	Vertical	(Suresh et al. 2014)		(Qureshi and Srinivasan 2009; Suresh et al. 2014; Awad et al. 2016; Wu et al. 2016)	(Suresh et al. 2014; Jung et al. 2010)
	Replacement	(Kultursay et al. 2013; Wang et al. 2014; Jin et al. 2014)	(Xu et al. 2013 ; Xu et al. 2015)	(Lee et al. 2009; Chen et al. 2012; Park et al. 2015)	(Baek et al. 2013)
Storage	Horizontal	(Lee et al. 2014)	(Tanakamaru et al. 2014; Sun et al. 2014; Fujii et al. 2012)	(Sun et al. 2010; Caulfield et al. 2010; Park et al. 2010)	(Yoon et al. 2008)
	Vertical	(Kang et al. 2015)		(Liu et al. 2011 ; Kang et al. 2015)	
	Replacement	(Lee et al. 2014)	(Jung et al. 2013)	(Akel et al. 2011; Kim et al. 2014)	(Baek et al. 2013)

good endurance as compared to NAND flash, and erase-less programming (no need for an erase operation to update data) (Qureshi et al. 2011).

3.1 Basic Concepts

3.1.1 PCM Technology. PCM uses a chalcogenide alloy to implement a memory cell. A PCM cell usually uses a thin layer of chalcogenide such as Ge2Sb2Te5 (GST) (Xue et al. 2011) and two

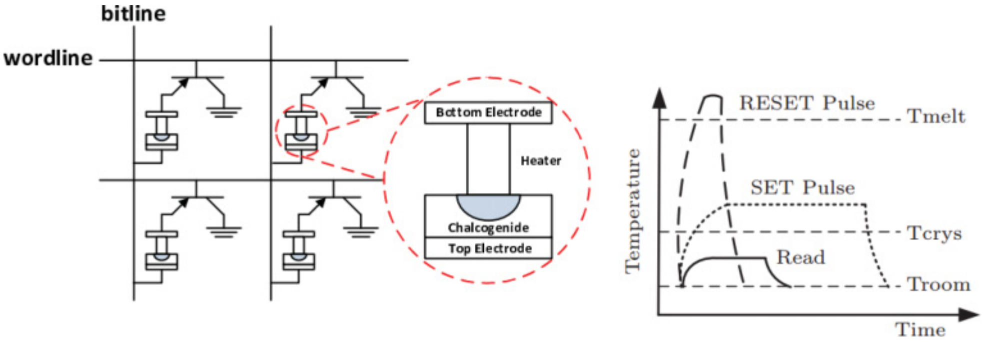


Fig. 2. PCM cell array (Xue et al. 2011) and RESET/SET temperatures (Xia et al. 2015).

electrodes that wrap the chalcogenide on both sides, in addition to a heater (see Figure 2). PCM is a resistive NVM that utilizes different resistances to represent bit information.

This chalcogenide material is subject to rapid amorphous-to-crystalline phase-change process that is electrically initiated.

The phase is dependent upon the heat produced by an electrical current pulse. A short but high voltage pulse results in an amorphous state (RESET, bit to 0), where the GST is heated above the melting temperature (T_{melt} in Figure 2). On the other hand, a long pulse with a low voltage results in a crystalline state (SET, bit to 1). Here the GST is heated above the crystallization temperature (T_{crys}) but below the melting temperature (see Figure 2). The write performance is thus determined by the longer operation that is the SET. Once a phase has been established, it may be read without disturbing this phase; the ratio between the resistance of the material in a SET and a RESET phase is between 10^2 and 10^4 (Kim 2008).

Thanks to a large difference between the two state resistance values, it is possible to store multiple bits (Multi-Level Cells) by making use of intermediate resistive values, thus increasing the storage density of PCM. Several state-of-the-art studies reported 3 bits per cell density (Wu et al. 2015). However, due to material composition fluctuation, the storage of the intermediate values needs an iterative Program-and-Verify process that is based on few SET operations until the target resistance value is reached (Jiang 2014).

3.1.2 Write Endurance. The main issue with PCM is the write endurance, as a PCM cell can only sustain a limited number of write operations (generally 10^8). This is mainly due to repeated heat stress applied on phase change material. Indeed, thermal expansion and contraction degrades the electrode-storage contact, which leads to a decrease in the reliability of writing currents in the cell. However, the endurance of PCM is better than that of flash memory (around 10^5 cycles), but worse than that of DRAM (10^{15}). This property is crucial when deciding on the integration of such a technology into the memory hierarchy.

Another issue regarding the write endurance of PCM is related to the RESET optimal current. State-of-the-art studies have confirmed that using current values above the optimal value leads to quick degradation of the memory cell, thus decreasing its endurance. As mentioned in Xue et al. (2011) and Kim and Ahn (2005), a twofold increase in the write energy leads to an approximately 50-fold decrease in the endurance.

3.2 Integration Options

The write characteristics of PCM narrow the scope of its use in different levels of the memory hierarchy. The write endurance and high write latency are the main issues, as they do not allow,

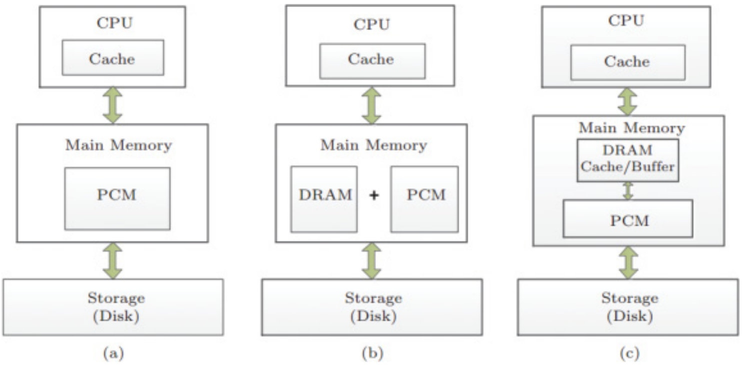


Fig. 3. Three PCM integration options (Xia et al. 2015).

for instance, for an integration in the first levels of the processor cache as the write traffic is very heavy (Mittal et al. 2015b). In state-of-the-art work, the main integration possibilities are as a main memory or as a storage system. In both cases, the PCM can be integrated vertically or horizontally.

3.2.1 PCM in Main Memory. There are three integration possibilities for PCM as a main memory: (1) as a replacement of the DRAM, (2) horizontally at the same level as the DRAM, (3) vertically with the DRAM.

Figure 3 shows the different integration options. When the DRAM and PCM are placed at the same level, the placement of data according to the access pattern is achieved by the application (for instance, through a specific API such as in Kannan et al. (2016)), at the operating system level (Lee et al. 2014; Salkhordeh and Asadi 2016; Wei et al. 2015), or at the memory controller level (Dhiman et al. 2009; Qureshi and Srinivasan 2009; Lee et al. 2009). In fact, the main objectives would be to hide the high latency of the write operations by moving write intensive workloads/data on DRAM and to rely on the non-volatility of PCM to save energy (Dhiman et al. 2009; Sun et al. 2015). In Wei et al. (2015), for instance, the authors use the NVM space to store file system meta data. Their system, called FSMAC, decouples the data and meta data I/O flow from the file system level and up to the hardware, which allows for a high-performance gain. PCM is also used in main memory for checkpointing (Oikawa 2014; Gao et al. 2015) and fault tolerance issues (Li et al. 2012).

In the case where DRAM is integrated as a cache for PCM, its objective is to absorb write operations to reduce the overall write latency and to address the problem of the write endurance of the PCM. Indeed, some experimental results (Qureshi and Srinivasan 2009) have shown that, with regard to memory, given a small DRAM buffer (3% of the PCM) a threefold improvement in speed can be achieved, as well as a three-fold extension in lifetime. Of course, performance depends on the write patterns. (Awad et al. 2016) emphasized the high impact of operating system mechanisms such as page prefetching and page replacement policies, on such PCM integration. Employing both horizontal and vertical integration, (Wu et al. 2016) used a small portion of DRAM to accelerate access to the NVM, while placing another portion horizontally in PCM.

Park et al. (2015) chose option (a) of Figure 3; they replaced the DRAM with PCM while trying to optimize the use of row buffers for an LPDDR2 interface integration.

3.2.2 PCM in Storage Systems. As compared to NAND flash memory, PCM has the ability to perform byte-wise random access and direct in-place updates; this makes it a serious competitor to flash memories.

PCM-based storage systems have already been prototyped. For instance, Onyx (Akel et al. 2011) is a PCM-based storage array interfaced in PCIe that has been proven to have better performance than state-of-the-art SSDs for some specific workload patterns (irregular and read dominated).

Many studies have focused on the vertical integration of PCM within storage systems. For instance, in Sun et al. (2010), PCM was used and integrated in solid state storage with flash memory to absorb log updates, similar to a cache. The objectives were (1) to relieve the flash memory from bursts of updates, which decreases performance and affects endurance; (2) to improve performance, as PCM allows in-place updates; and (3) to save energy, as PCM is more energy efficient than DRAM. Other studies used the same principle of using PCM inside SSDs to supplement or replace DRAM (Liu et al. 2011) with the objective of optimizing performance and avoiding loss of data from the SSD cache due to power failures, with the help of PCM non-volatility.

According to state-of-the-art studies and prototypes, PCM-based storage systems show great promise, as they can drastically reduce latencies and increase the bandwidth of storage systems for high performance I/O intensive computing (Caulfield et al. 2010).

How to architect this type of NVM to take full advantage of its performance is a critical issue. Many of the prototype storage systems are based on PCIe, and the authors claim that this interface can be a bottleneck as it can limit the achievable performance (in terms of latency, for instance).

From an operating system perspective, many layers of the storage system software stack were devised with HDD in mind, so some micro seconds of latency can be tolerated. This is no longer the case, and efforts need to be expended to adapt the storage software stack to the new coming NVMs (Lee et al. 2014). Many questions need to be answered, such as what to abstract to the OS from the underlying hybrid architecture, how to interface the different storage systems, and which protocol/mechanisms should be used. Towards that end, some propositions have been made, for instance, in Park et al. (2010) the authors proposed two possible integrations of PCM in the Linux operating system:

1. Having both DRAM and PCM in main memory and performing data placement according to the I/O type. This would allow large amounts of energy to be saved as compared to full DRAM implementation. Page placement is performed according to the segmentation of the executed processes.
2. As complementary storage with traditional storage systems. Here, data placement is based on putting small-sized random accessed data in the PCM.

This study presents a concrete solution on how to interface hybrid memory systems on Linux.

3.3 Open Research Questions

Xia et al. (2015) observed that many optimization issues have been discussed in the literature, with the aim of making this technology viable in different systems. The main issues deal with optimizing the write latency so PCM can be used in place of DRAM, optimizing the endurance, and saving energy.

3.3.1 Optimizing the Write Latency. As the write latency is due to some physical properties of the phase change material, the optimizations should be performed at other levels.

Indeed, the first optimization technique consists of optimizing the write command latency. Many options are possible such as: (1) Reducing the write bits: this solution consists of writing only modified bits (Zhou et al. 2009a; Yang et al. 2007; Dong et al. 2015); (2) increasing the level of parallelism of the writing bits by making better use of the SET and RESET operation properties (Yue et al. 2013a; Du et al. 2013); (3) hiding the write latency for read operation optimization: this is achieved, for instance, by pausing long write operations when a read operation arrives, and then resuming the write once the read is achieved (Qureshi et al. 2010).

3.3.2 Optimizing the Endurance. One of the main issues regarding PCM as a DRAM replacement is the write endurance, which is due to the continuous heating of the phase change material. Many techniques have been proposed to cope with this problem.

Solutions for optimizing the endurance can be grouped into three subclasses: (1) Improving the endurance by acting on the electrical characteristics, such as by targeting an optimal RESET current because over-programming, has a very bad impact on endurance. For instance, in Jiang et al. (2012) the authors proposed an elastic RESET, which reduces the RESET current, thus positively impacting the memory lifetime. (2) The second option consists of minimizing the number of write operations; this can be achieved by inserting a DRAM buffer that absorbs frequent writes. Other solutions were proposed, such as avoiding the writing of unmodified memory bits (Lee et al. 2009; Zhou et al. 2009a; Yang et al. 2007) or even avoiding the writing of useless data from the cache (Bock et al. 2011). Other techniques were also explored, such as data compression (Sun et al. 2011). (3) Finally, global PCM endurance can be improved through wear leveling techniques. There are two main classes of wear leveler: wear levelers that rely on memory block states, for example, switching hot (frequently updated) and cold data (Zhou et al. 2009a; Yun et al. 2012); and wear levelers that randomly swap data independently from their access pattern, for example, those that do so periodically, such as in Zhou et al. 2009a; Qureshi and Srinivasan (2009).

3.3.3 Write Disturbance. PCM is subject to write disturbance when the inter-cell distance is too small. The high temperature produced by a RESET operation can lead to a phase change in adjacent cells by thermal conduction. This may induce parasitic SET operations (Russo et al. 2008). This effect could limit the technology scaling under 20 nm. The state-of-the-art solutions include imposing a minimal inter-cell space, and designing write strategies and data coding techniques to avoid write disturbance (Jiang et al. 2014; Wang et al. 2015b; Wang et al. 2015c).

3.3.4 Energy Saving. One of the key arguments in favor of NVMs as a replacement or complement to DRAM concerns their energy behavior. Indeed, the static energy is negligible. Unfortunately, PCM demonstrates very high write energy (higher than DRAM). Many techniques have been proposed to address this problem.

There are two main ways of reducing the energy consumption of write operations: (1) reducing the number of writes and (2) reducing the energy consumption of write operations. The techniques cited previously and which consist of avoiding the updating of all bits, avoiding unnecessary writes, and so on, not only reduce write latency, but also the energy consumption of write operations. As for the second class of solutions, they mainly consist of exploiting the write asymmetry (SET and RESET) (Xu et al. 2009; Mirhoseini et al. 2012; Chen et al. 2012; Yue et al. Zhu 2013b) to reduce energy, as the RESET operation consumes more energy than the SET operation (Xia et al. 2015).

In MLC PCM, writing of a cell line begins with a RESET operation followed by a variable number of SET operations. This number depends on the values to store (P&V process). In Jiang (2014), the authors propose to schedule RESET operations among different cells at different times to reduce the size of charge pumps that supply the RESET operation energy. By reducing their sizes, the wasted power can be reduced. This optimization relies on the fact that the number of SET operations to perform might be different according to the target value. As a consequence, some RESET operations can be easily shifted in time with no impact on the cell line write operation duration.

3.4 PCM Maturity and the State of the Market

Many companies have put a great deal of effort into making PCM available in the market. In August 2004, Nanochip licensed PCM technology for use in MEMS (Micro-Electric-Mechanical-Systems) probe storage devices. In September 2006, Samsung announced a prototype device with 512Mb size using diode switches. The fairly high density is notable. The prototype featured a cell size of only 46.7nm, smaller than commercial flash devices available at that time. In comparison, the only MRAM and FeRAM devices being produced at the time were about 4Mb in size. The high density of Samsung's prototype PCM device suggested that PCM could be a viable competitor to flash, not limited to niche roles as other devices have been (Athmanathan et al. 2016).

Samsung's announcement was followed by announcements from Intel and STMicroelectronics, who demonstrated their own PCM devices in 2006 (at the Intel Developer Forum). They came up with a 128Mb device that STMicroelectronics began manufacturing as a proof-of-concept. In February 2008, Intel and STMicroelectronics revealed the first multi-level (MLC) PCM array prototype. The prototype had two logical bits stored in each physical cell; in effect, 256Mb of memory were stored in a 128Mb physical array. This means that instead of the normal two states, fully amorphous and fully crystalline, an additional two distinct intermediate states represent different degrees of partial crystallization, allowing for twice as many bits to be stored in the same physical area.

In June 2011, IBM announced that they had created a stable, reliable, multi-bit PCM with high performance and stability. Also in February 2008, Intel and STMicroelectronics shipped prototype samples of their first PCM product—a 90nm, 128Mb product that they called Alverstone.

In April 2010, Numonyx announced the Omneo line of 128Mbit NOR-compatible phase-change memories, while Samsung announced that a 512Mb PCM in a multi-chip package (MCP) would be used in mobile handsets shipped by the fall of 2010. In July 2012, Micron announced the availability of PCM for mobile devices, the first PCM solution in production. However, Micron withdrew all PCM parts from the market in January 2014. In 2016, IBM demonstrated that they could store two bits per cell, and were on their way to achieving three bits per cell (Athmanathan et al. 2016).

4 MRAM

One of the more mature and promising NVMs is Magnetic or Magnetoresistive RAM (MRAM), which has the following very attractive properties: high density, fast read, low leakage, and high endurance (see Table 1 for the STT-RAM variant of MRAM).

4.1 Basic Concepts

4.1.1 MRAM Technology. MRAM are based on the magnetic properties of a special material. Contrary to standard DRAM technologies, the information carrier is the Magnetic Tunnel Junction (MTJ) instead of electrical charges. Magnetic orientation can be controlled and sensed using electrical signals (Kultursay et al. 2013).

The MTJ implements the storage function. It consists of two ferromagnetic layers separated by an oxide (tunnel) barrier layer. One of the ferromagnetic layers has a fixed direction of magnetization and is called the *reference (or pinned) layer*, while the other has a variable direction and is called the *free layer*.

The MTJ shows a low resistance when the two layers have the same direction of magnetization; this is called the parallel state and represents the logical 0 (see Figure 4). When magnetization directions are opposite to one another, we call this an anti-parallel state in which the resistance is higher than the parallel state. This represents a logical 1.

In Spin Torque Transfer RAM (STT-RAM), that is widely studied and our focus in this article, the magnetization direction of the free layer changes according to the voltage that is applied

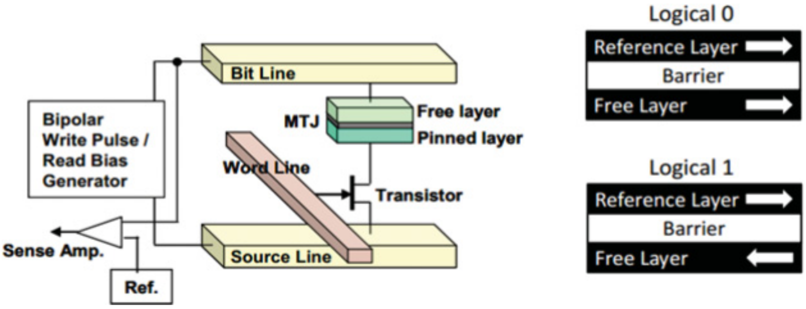


Fig. 4. An STT-RAM memory cell (Xue et al. 2011) and operations (Kultursay et al. 2013).

between the source line (SL) and the bit line (BL). Therefore, when a high positive voltage difference is applied between SL and BL, a logical 0 is written, while if a high negative voltage difference is applied, then a logical 1 is written. To ensure a status reversal, the issued current must be maintained for a given duration of time called the *write pulse width* (between 2ns and 12ns) and its amplitude should be equal to a *threshold current* (between 100μa and 1000μa).

STT-RAM differs from the original MRAM cell in how it sets the orientation of the MTJ: in previous MRAM technology the orientation is set using an external magnetic field, while in STT-RAM a current of polarized electrons is used.

For a read operation to occur, a very small voltage is applied between SL and BL, which causes a current to flow through the MTJ. The value of this current is relative to the resistance of the MTJ, which depends on the magnetization. This current is compared with a reference current, and then the value 0 or 1 is read from the MRAM cell.

Another promising type of MRAM technology is the SOT-RAM (Spin Orbit Torque RAM) (Oboril et al. 2015). The junction here uses perpendicular magnetization and an additional terminal is added to separate the path of read and write operations. This implementation allows for a better optimization of the two (read and write) paths, as they have different physical requirements.

4.1.2 MRAM Endurance. For STT-RAM, a 10^{15} endurance value, in write cycles, has been estimated. However, in some other studies the best endurance test yielded a value of less than 4×10^{12} (Mittal et al. 2015b; Huai 2008).

In fact, MRAM suffers mainly from two sources of unreliability: (1) MTJ cells are subject to thermal instability, which could lead to data loss. The probability of failure depends on the implementation of the junction, the temperature, the vulnerability time (which is the time between the write and the last reading of data), and the number of memory cells. The thermal instability increases as the scale of technology decreases. (2) The high write current stresses the memory cells and degrades the junction, and then limits their integrity over the time. This problem impacts STT-RAMs, where the write current is high.

In addition, as the read operation needs a flow of current, this can also induce a change in the magnetization, called read disturb. As the write current is proportional to the MJT area, the shrinking of the feature size causes the reduction of the write current, whereas the read current must remain sufficient to drive the sense amplifiers (Wang et al. 2015a). As a consequence, the read disturbance rate increases with the memory density. STT-RAMs are also more sensible to such an effect than SOT-RAMs, as the write and read paths are the same.

4.2 Integration Options

As one can see in Table 1, the write endurance of STT-MRAM is approximately the same as that of DRAM and SRAM. This makes it a good candidate for integration at those levels of the memory hierarchy. However, the endurance is far from the ideal value of 10^{15}. Nevertheless, as we will see in the next sections, some efforts have been made to increase the endurance by, for instance, compromising on the retention time.

From a performance point of view, even though the read performance of STT-RAM is comparable to that of SRAM and in some cases better than that of DRAM, it is lagging in terms of write latency and energy.

Most state-of-the-art work integrated STT-MRAM at the primary memory or cache memory level, whether when complementing a given level of the hierarchy or replacing it.

4.2.1 MRAM in On-Chip Cache. STT-MRAM has been integrated in mainly two ways in on-chip caches in state-of-the-art work: (1) in all levels of the cache hierarchy, as in Li et al. (2012), Komalan et al. (2014), and Senni et al. (2015), or (2) in the last levels of cache (LLC), as in Syu et al. (2013), Cheng et al. (2016). The idea behind the second set of solutions is that the large write latencies of STT-RAM cannot be tolerated in first-level cache, because this level is accessed very frequently.

MRAM in all levels of cache. Many studies have proposed using STT-MRAM in all levels of cache. In Senni et al. (2014), the authors highlighted the gain that one can expect in terms of static energy from replacing SRAM caches on L1 and L2 with STT-RAM. The study shows that while the performance is comparable for the tested workloads, STT-RAM drastically decreases the static energy while consuming more dynamic energy. The same conclusions were drawn in Komalan et al. (2014), in which an L1 instruction cache was replaced by STT-RAM by extending the miss status holding registers to cope with the write latency of the NVM.

Multi-level caches offer abilities for mitigating read disturbance issues. For an STT-RAM integration as L2 cache, in Wang et al. (2015), a Read-And-Restore strategy is proposed, by which after the processing of a L1 miss, a restore buffer refreshes the source L2 block in background. Based on the observation that read disturbance happens when the read current is in an opposite direction than the write current, they also establish a two-read sequence, which makes it possible to detect disturbed memory cells, thereby only refreshing these ones. These two techniques optimize the performance and reduce energy overhead due to the read disturbance management.

The authors of Li et al. (2014) envisioned a horizontal integration of STT-RAM with SRAM on a one-level cache system. In such hybrid caches (Sun et al. 2009; Wu et al. 2009; Jadidi et al. 2011; Li et al. 2011), the system migrates data from the two technologies according to the access pattern mainly to mitigate the high write operation latency. The authors of Li et al. (2014) presented a compiler-assisted approach to minimizing the migration overhead. In Wang et al. (2015), the authors proposed a full design of STT-RAM integration as a scratchpad memory for real-time applications.

As previously stated, the very promising SOT-RAM technology drastically reduces the access time and energy consumption. It has even been used in L1 hybrid caches. In Oboril et al. (2015), the authors studied many configurations with SRAM, SOT-RAM, and STT-RAM. The main metrics for evaluation were the area, the execution time, and the energy. The study ended with some interesting conclusions, such as that the SOT-RAM can be the best choice for the first level of cache for low power devices, but that for small memory blocks, SRAM is still the best solution in terms of area and performance. They were even many cases for which hybrid configurations such as SOT-RAM for L1 instruction caches and SRAM for L1 data caches, with SOT-RAM for L2 caches, was the optimal solution.

MRAM in Last-Level Cache (LLC). The main technology used in studies introducing MRAM in LLC is the STT-RAM.

In Wu et al. (2009), the authors presented a very interesting study on exploring the architecture of cache when integrating some NVMs, such as MRAM and PCM, with some new technologies with volatile memories such as eDRAM (embedded DRAM). In this study, the authors evaluated many different architectures split into two types of hybrid caches namely: inter-cache level hybrid cache architectures (LHCA) and intra-cache or region-based hybrid cache architecture (RHCA). In the former, each level consists strictly of one memory technology (vertical integration), while in the latter each level may consist of regions with different technologies (horizontal integration). The study showed that LHCA can provide a 7% improvement in IPC (instructions per cycle) over a three-level SRAM cache, while RHCA can provide a 12% IPC improvement with the same area constraints.

Concerning horizontal integration, in which two memory technologies can be integrated for a given level of cache, the system should apply some adaptive data placement mechanism to take full advantage of each integrated technology. For instance, in Wang et al. (2014), the system tries to reduce writes on STT-RAM by detecting the access pattern on the LLC.

Following the idea of partitioning a given level of cache, in Samavatian et al. (2014) the authors partitioned the L2 cache into two regions of STT-RAM by tuning the retention. They used a low retention partition to absorb frequently written data and a high retention region for less frequently written and very frequently read data. Decreasing the retention allows the write energy and the area of the STT-RAM cell to decrease.

4.2.2 MRAM in Main Memory. Even though most state-of-the-art work on integrating MRAM has focused on on-chip cache subsystems, some studies have explored integrating MRAM as a DRAM replacement or complement.

In Kultursay et al. (2013), the authors studied the complete replacement of the main memory by STT-RAM. In fact, as previously underlined, DRAM is limited by two main factors: scalability and power consumption due to power leakage. The authors first compared DRAM with non-optimized STT-RAM technology. They showed that a simple replacement of DRAM by STT-RAM does not lead to good performance; rather, the STT-RAM main memory behaved very badly because of its write operation latency and energy consumption. The authors then performed some optimizations on STT-RAM. One optimization involved using its property of a decoupled structure of sense amplifiers and row buffers. This allows it to perform selective and partial writes on the memory cell, thereby reducing the number of write operations, thus the latency and energy. Another optimization consisted of bypassing the row buffer for the write operations, as it was observed that the number of hits on write operations is much lower than on read operations. The results of such optimizations on a given number of workloads and a multicore processor showed that an STT-RAM memory, while achieving a performance comparable to that of DRAM, drastically reduces the amount of energy that is consumed (by about 60%).

Another approach studied in Yang et al. (2013) consists of integrating STT-RAM horizontally with DRAM while implementing a scheme to direct write-intensive data to DRAM. The main objective in integrating both memory technologies in a main memory is to reduce energy consumption levels.

Even though previous studies showed good results from integrating STT-RAM in main memory, as noted in Wang et al. (2014), the internal structure of the MRAM chip is not compatible with current DRAM interfaces. Therefore, to address this issue, the authors proposed some techniques to make the design of LPDDR3 compatible with MRAM architecture.

Other concerns were noted in Jin et al. (2014), relating to the area, power, and latency of STT-RAM memories. The authors draw the attention of the reader to the size of an STT-RAM cell, which might present a serious obstacle to attaining a DRAM-like density. In addition, the write power of STT-MRAM is reportedly too large for many application domains. The authors propose modifications to some critical parameters, such as the thermal stability factor, critical current, and retention time to mitigate the STT-RAM's cell energy and latency properties to make it more suitable as a replacement for DRAM.

4.2.3 MRAM in Storage Systems. There are several studies in integrating MRAM in storage system devices. Lee et al. investigated both PCM and STT-RAM integration in storage systems (Lee et al. 2014). This interesting study emphasizes the effect of the kernel I/O software stack by integrating these NVMs. In addition, they compared and analyzed the efficiency of kernel mechanisms (such as read-ahead) and access methods (such as direct and synchronous I/Os) with NVM based storage systems. In Kang et al. (2015), the authors propose a vertical integration of NVM in the storage system. The study focused on PCM but their techniques can work on STT-RAM as well. The idea is to use retention relaxation when NVM is used as a cache in storage systems to decrease the latency, since most data in a cache are evicted within a short time period after they have been entered into the cache.

4.3 Open Research Questions

In many studies on using STT-RAM in a given cache level, the authors tried to improve its performance or to adapt to the constraints of the MRAM through a given set of optimizations. These are summarized in Mittal et al. (2015b):

4.3.1 Relaxing Non-volatility. Relaxing the non-volatility of STT-RAM (Smullen et al. 2011; Jog et al. 2012; Sun et al. 2011; Guo et al. 2010): the idea here is to use different regions (horizontal integration) or levels (vertical integration) of STT-RAM with different retention periods, thus resulting in different write latencies and energy consumption properties.

4.3.2 Minimizing or Avoiding Writes. Minimizing or avoiding writes (Joo et al. 2010; Bishnoi et al. 2014; Zhou et al. 2009b; Goswami et al. 2013; Yazdanshenas et al. 2014; Ahn et al. 2012; Sun et al. 2009; Rasquinha et al. 2010; Jung et al. 2013; Park et al. 2012; Cheng et al. 2016): the main objective of this optimization is to reduce the number of bits updated on the STT-RAM. This can be achieved with different techniques: for instance reading the data before writing it and writing only the different bits, using caches to perform comparisons before updating, coding data to decrease bit modifications, and so on.

4.3.3 Improving the Memory Lifetime. Improving the memory lifetime by using wear-leveling: for flash memories, wear leveling techniques could be used to increase the memory lifetime while coping with the temporal and spatial locality of workloads (Mittal 2013; Chen et al. 2013; Wang et al. 2014a; Jokar et al. 2016).

4.3.4 Addressing 0/1 Asymmetry. As the time required to perform a 1 to 0 transition is larger than that required to perform a 0 to1 transition due to the asymmetry of the switching time of the MTJ, presetting all cells to 0 before an operation allows the effective write latency to be reduced (Kwon et al. 2014).

4.4 MRAM Maturity and the State of the Market

Companies have been developing MRAM for many years. In 2003, a 128Kbit MRAM chip manufactured with a 180nm lithographic process was introduced. The next year, a 16Mbit prototype was unveiled by Infinenon. In 2005, Honeywell posted a data sheet for a 1Mbit Rad-hard MRAM using a 150nm lithographic process, followed by a MRAM memory cell running at 2GHz. Sony announced the first lab-produced STT- MRAM that utilizes a spin-polarized current through the tunneling magneto resistance layer to write data. This method consumes less power and is more scalable than a conventional MRAM. That same year, Freescale Semiconductor Inc. demonstrated

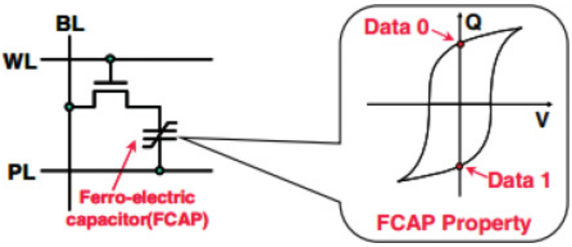


Fig. 5. A FeRAM memory cell.

an MRAM that uses a magnesium oxide barrier and improved bit resistance during the write cycle, thereby reducing the required write current.

Toshiba and NEC announced a 16Mbit MRAM chip with a new *power-forking* design in 2006. It achieved a transfer rate of 200MB/s, with a 34ns cycle time, the best performance of any MRAM chip. Hitachi and Tohoku University demonstrated a 32Mbit STT-RAM in 2009, and then announced a multi-level MRAM in 2010. In November 2012, Everspin debuted a 64Mb STT-MRAM (Everspin 2015).

5 FERAM

Ferroelectric RAM (FeRAM or FRAM) is an NVM that has made it to the stage of mass production earlier than other NVMs (apart from flash memory). Indeed, FeRAM is used in many products today, mainly in the embedded domain (Fujisaki 2013), for example, in automobile equipment such as data recorders and wireless cards, and promises to be a good candidate for the non-volatile storage for wearable electronics (Venkatesh and Singh 2015).

FeRAM exhibits some very attractive properties that made it one of the strongest early candidates (Burr et al. 2008) for the NVM of the future, namely, low latency (as low as that of DRAM), reported to be 20ns (Burr et al. 2008), low power consumption, an erase-less write operation, and the possibility of a straightforward CMOS integration (Kato et al. 2005).

5.1 Basic Concepts

The unit cell of FeRAM resembles that of DRAM in that it consists of one transistor and one Ferro-electric capacitor (FCAP), also known as a 1T1C structure (Kryder et al. 2009; Kanaya et al. 2004). Ferro-electric material is formed of $Pb(Zr_x, Ti_{1-x})O_3$ (PZT or lead zirconate titanate), between two metallic electrodes. FCAP is characterized by two remnant reversible polarization states. The charge of the FCAP retains its polarity without the need for power, thanks to the property of hysteresis that the ferroelectric material exhibits in comparison to the dielectric material in DRAM. This explains its non-volatility. In addition, FeRAM does not require refreshing operations; consequently, it consumes less power than DRAM.

The write operation is achieved by forcing a pulse through the plate line (PL) of the FCAP for a 0, and through the bitline (BL) for a 1 (see Figure 5). Contrary to NAND flash memory, FeRAM does not need asymmetric voltage and/or special high voltage for write operations. For FeRAM, the voltage used is the same as that for Vcc, and can be very small.

As stated earlier, FeRAM was seen as one of the first mature NVMs to replace and/or complement DRAM and storage system technologies. However, after many years of studies on the technology, FeRAM cells still seem difficult to make scalable. In fact, their size remains significantly larger than DRAM cells for gigabit-scale integration. As a consequence, as will be seen in the next section, most recent work on FRAM integration address the issue of whether FeRAM can be used in embedded systems (they are already deployed in MCUs) or in a limited quantity in hybrid systems, or as small volume caches.

5.2 Integration Options

Many different figures on performance have been given in state-of-the-art studies concerning the performance of FeRAM. However, many agree that its read and write operations are more or less symmetric. Unlike MRAM and PCM, FeRAM does not show a strong asymmetry between read and write operations. The performance of FeRAM approaches that of DRAM, but still seems to lag behind. In contrast, it is better than DRAM in terms of power consumption. Concerning endurance, values between 10^{12} and 10^{15} have been reported.

The integration of FeRAM can be discussed from different points of view:

- From an endurance point of view, FeRAM could replace DRAM or storage system technologies such as flash memory.
- From a performance point of view, FeRAM could complement DRAM and replace storage system technologies.
- From a density point of view, FeRAM cannot yet replace either DRAM or storage systems technologies, because its scalability to the gigabit remains an open question. As high density cannot be proven for FeRAM, its price is still very high compared to flash memories, even though some applications in the embedded domain are driving the technology.

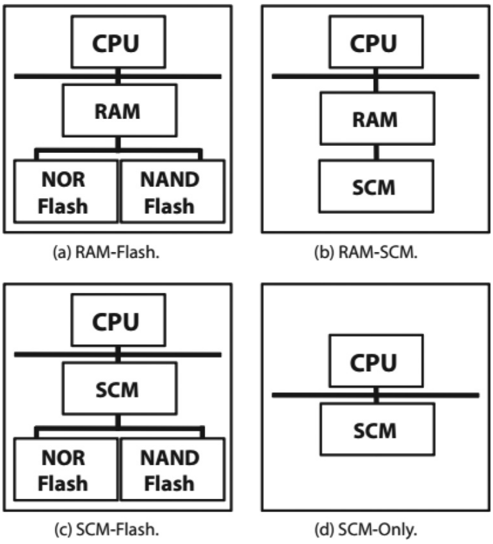
Nevertheless, FeRAM has been studied in state-of-the-art work, indeed, as the physics behind FeRAM have been well mastered, its performance can easily be improved if some new materials with better scalability are discovered.

Some FeRAM integration propositions exist in state-of-the-art studies. To the best of our knowledge, in all studies it was integrated between DRAM and the storage system. The integration was either horizontal, for schemes replacing or complementing DRAM and the storage system with FeRAM; or vertical, involving the insertion of FeRAM between the DRAM and the storage system, generally as a metadata cache (Doh et al. 2007; Yoon et al. 2008).

5.2.1 FeRAM in Main Memory. We did not find an exhaustive set of state-of-the-art work on the use of FeRAM as a primary memory replacement or complement. This is probably due to the drop in performance that can result when integrated in such a way.

Two main studies have been identified. In Suresh et al. (2014), the authors evaluated many different NVM technologies, among which is FeRAM. It has been investigated as a replacement for DRAM and also as a complementary memory to form a hybrid main memory. The performance of FeRAM was tested and compared with that of other NVMs, namely STT-RAM and PCM. Simulations showed that as FeRAM alone can degrade the performance as compared to DRAM, but in some cases, it can lead to small improvements in energy consumption. However, a hybrid memory with DRAM and FeRAM showed better energy efficiency for workloads showing large static energy, even as it degraded performance. As compared to other NVMs, FeRAM was the poorer performer in this study.

Another interesting study on the integration of NVMs (called Storage Class Memory – SCM in this study) was conducted (Baek et al. 2013). In this study, several architectures were tested against different workloads (see Figure 6). This included classical DRAM + flash memory as a storage system, DRAM + SCM, only SCM, and SCM + flash. A FeRAM prototype of 32MB was used as the SCM. The first observation of the study was that SCM + flash gave the worst performance in most cases. This simply means that a straightforward replacement of DRAM by FeRAM is probably not a good idea because of the high latencies of FeRAM (as compared to DRAM). On the other



(a) RAM-Flash. (b) RAM-SCM. (c) SCM-Flash. (d) SCM-Only.

Fig. 6. Tested memory hierarchies in Baek et al. (2013).

hand, the SCM only configuration produced some good results depending on the workload. The authors went further by proposing some optimizations of the SCM only configuration. The idea was to perform in-place updates of files and meta-data rather than having multiple copies as is the case in memory/storage traditional systems. The authors proposed a unified file/data management system referred to as object management. Bridges were built between traditional file and memory API, with the SCM manager implementing object management. With this new architecture, the authors demonstrated some significant performance enhancements over an SCM that simply reuses traditional operating system memory and a storage stack.

The authors of previous studies clearly mentioned that FeRAM performance lags behind that of DRAM, even if the energy consumption properties of FeRAM are superior. However, if it is to be used in high performance computing, then FeRAM needs to demonstrate scalability, which it is far from being able to do.

5.2.2 FeRAM in Storage Systems. From a storage system perspective, FeRAM was mainly integrated horizontally in state-of-the-art studies, except in Chameleon (Yoon et al. 2008), where it was integrated both horizontally and vertically, and in Baek et al. (2013), where the authors mainly focused on replacing the primary storage system with FeRAM, with or without DRAM.

As noted above, FeRAM has been used sparingly because of integration and scalability issues. With the exception of Baek et al. (2013), most state-of-the-art studies used FeRAM mainly to optimize data access to file systems (Jung et al. 2010; Doh et al. 2007) or FTL (Yoon et al. 2008) metadata.

In Chameleon (Yoon et al. 2008), the idea is to store (1) FTL metadata such as mapping tables in FeRAM, and (2) to use a FeRAM block and page buffer to accelerate access to the underlying flash memory. In Doh et al. (2007), the main idea is to store flash file system (MiNV) metadata into FeRAM rather than storing the data into flash and copying them to DRAM at mount time. A very good performance has been observed on the tested workloads.

In FRASH (Jung et al. 2010), the authors tried to bring persistence to memory data structures, so it would be possible to accelerate some operations, for instance, mount time, and to offer faster access to file system data structures by storing them in NVM (FeRAM). For the first objective, the goal was to overcome the volatility of DRAM and to avoid costly mount time data structures by storing those data structures in FeRAM. For the second objective, FRASH stores some storage system structures, such as page metadata, in the FeRAM. FRASH proved to have good performance in comparison to classical flash file systems such as YAFFS or other SCM integration solutions.

5.3 FeRAM Maturity and the State of the Market

Many companies are offering FeRAM-based products. Among these are Ramtron, which has a large spectrum of products; Texas instruments with FeRAM-based MCUs; and Toshiba, which introduced a 128MB FeRAM in a 130nm technology in 2009 (Shiga et al. 2009).

Much of the current FeRAM technology was developed by Ramtron, a fabless semiconductor company. One major licensee is Fujitsu, which operates what is probably the largest semiconductor foundry production line with FeRAM capability. Since 1999 Fujitsu has been using this line to produce standalone FeRAMs, as well as specialized chips (e.g., chips for smart cards) with embedded FeRAMs. Fujitsu produced devices for Ramtron until 2010. Since 2010 Ramtron's fabricators have been Texas Instruments (TI) and IBM. Since at least 2001 TI has collaborated with Ramtron to develop FeRAM test chips in a modified 130nm process. In the fall of 2005, Ramtron reported that they were evaluating prototype samples of an 8Mbit FeRAM manufactured using TI's FeRAM process. Fujitsu and Seiko-Epson were collaborating in 2005 to develop a 180nm FeRAM process. In 2012, Ramtron was acquired by Cypress Semiconductor.

FeRAM research projects have also been reported at Samsung, Matsushita, Oki, Toshiba, Infineon, Hynix, Symetrix, Cambridge University, the University of Toronto, and the Interuniversity Microelectronics Centre (IMEC, Belgium).

Since 2016, TI's latest MSP430 microcontrollers have been utilizing FeRAM for code and data with a unified memory approach (Wong 2016).

6 RERAM

The basic idea behind Resistive RAM (ReRAM) is that they are memories that store their logical state under the form of a resistive state (Sandhu 2013). Thanks to its characteristics, ReRAM is supposed to integrate many application domains such as consumer electronics, personal computers, medical applications, space, and automotive applications (Meena et al. 2014). Indeed, ReRAM is compatible with conventional semi-conductor fabrication processes. ReRAM is a very promising technology as it has a lower power consumption than PCM and a higher density than MRAM (see Table 1). ReRAM has demonstrated good stability at fabrication technologies of 10nm (Akinaga and Shima 2010). From an endurance point of view, it is between one and five orders of magnitude better than flash memory (see Table 1). From the scalability and fabrication points of view, ReRAM could be considered a serious candidate for universal memory; however, its endurance remains limited and it cannot be a replacement for SRAM or DRAM. Nevertheless, it is currently very extensively studied and many scientific obstacles could be resolved in the coming years.

6.1 Basic Concepts

ReRAM belongs to the class of memristor devices. A *memristor* is a two-terminal resistor device with varying resistance. The resistance value changes according to the magnitude and polarity of the voltage that is applied, and the duration for which the voltage is applied (William 2008). The main characteristic of a memristor is that the resistance value does not change when the power is turned off; this provides it with its property of non-volatility.

The concept of memristor was invented by Leon Chua in 1971 (Chua 1971). It has been called the fourth fundamental building block of electronics (along with resistors, capacitors, and inductors). In 2008, it was demonstrated by a team of researchers from HP labs (Sturkov et al. 2008): "Memristor

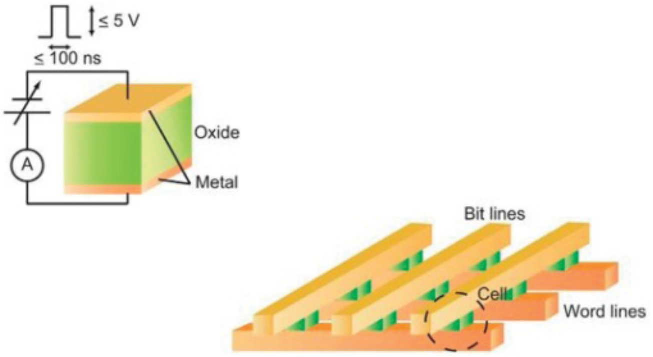


Fig. 7. ReRAM cells[3]

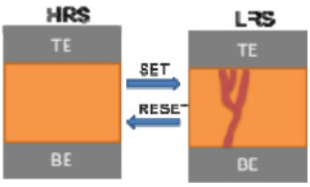


Fig. 8. ReRAM cells and filament formation in Oxide-based RAM (Clermidy et al. 2014).

is a contraction of memory resistor, because that is exactly its function. . . the ability to indefinitely store resistance values means that a memristor can be used as a nonvolatile memory" (William 2008).

In a memristor, a low resistance value is considered as a logical 1 while a high resistance value is considered as a logical 0. To read a memory cell value, the sensing logic compares the current flowing through the selected memristor cell with the one flowing through a reference resistor with the same voltage applied on both.

A ReRAM cell is a two-terminal Metal-Insulator-Metal (MIM) device composed of an insulating or resistive material, which is an oxide, sandwiched between two electrode metal conductors (see Figure 7).

The basic principle behind ReRAM is the formation (Low Resistive State) and disruption (High Resistive State) of a conductive filament, the objective of which is to shunt the top and bottom electrodes (M) through the resistive oxide layer (see Figure 8).

ReRAMs can be classified into the following two types according to their switching behavior: (1) unipolar devices, where the switching direction depends on the amplitude of the applied voltage (Kryder et al. 2009; Sandhu 2013; William 2008), and (2) bipolar devices, where the switching direction depends on the voltage polarity. In bipolar devices, the switching mechanism is based on the creation of oxygen vacancies, which form an electric path between the top and bottom electrodes. TiO_x material has been widely used as top and bottom electrode material (Choi et al. 2005; Fujimoto et al. 2006; Hosoi et al. 2006). Many other materials have been used in state-of-the-art-work, such as ZrOx (Lee et al. 2006) and NiO (Gibbons and Beadle 1964).

[3] Source: http://www.dobreprogramy.pl/Crossbar-pokazal-terabajtowy-czip-pamieci-ReRAM-niebawemzapomnimy-o-flashu,News,45456.html.

Conductive Bridge RAM (CBRAM), another promising technology, is a solid-electrolyte (SE) memory in which the SE material is sandwiched between an inert electrode (cathode) and an oxidizable electrode (anode).

6.2 Integration Options

Due to its integration capabilities and performance, ReRAM has been considered in many studies as a potential replacement and/or as complementary memory in the memory hierarchy. The endurance of ReRAM was reported to be between 10^{10} and 10^{12} (Sheu et al. 2011; Kim et al. 2011; Eshraghian et al. 2011), meaning that it would be difficult to integrate in cache levels without specific optimizations. ReRAM has been investigated for both horizontal and vertical integrations.

6.2.1 ReRAM in Cache Memory. The endurance of ReRAM could pose an obstacle to its integration in low memory levels such as caches, and this possibility has been investigated in many studies.

In Dong et al. (2013), the authors studied many NVMs with a focus on a ReRAM case study. Different models were proposed, allowing for a design space exploration for the integration of ReRAM and other NVMs. Those models were implemented through a simulator called NVSim (Dong et al. 2012). The authors of Dong et al. (2012) compared differences in performance when ReRAM was integrated vertically in the cache (traditional SRAM) stack. Many scenarios were tested: putting ReRAM in all traditional cache levels, then in the two higher levels (L2 and L3), and then only in the higher level (L3). In each case the authors noted an energy gain and a degradation in performance (ReRAM is comparable to SRAM in read performance but significantly slower in write performance). One would need to compromise on the chosen architecture in accordance with the energy/performance tradeoff needed for the application. In a previous study, the authors relied on wear leveling techniques to partly cope with endurance issues (Schechter et al. 2010; Wang et al. 2013). In Mittal and Vetter (2015a), the authors presented a horizontal integration of ReRAM in the L2 cache together with SRAM in a system called AYUSH. To cope with the ReRAM endurance issue, frequently reused data are stored in the SRAM.

ReRAM was also introduced in the memory hierarchy for more specific application domains. In Komalan et al. (2013), it was integrated in the L0 (loop buffer (Uh et al. 1999)) and/or L1 cache level for ultra-low power embedded instruction memories dedicated to low-power wireless/multimedia applications with specific read-dominant workloads (loop dominated code). ReRAM has also shown good performance when integrated into FPGAs as a replacement for SRAM to improve problems such as leakages of power and rapid power on/off operations (Clermidy et al. 2014).

6.2.2 ReRAM in Main Memory. As for PCM, ReRAM has been integrated in main memory to complement DRAM. For instance, in Hassan et al. (2015), the authors considered ReRAM in a horizontal scheme with DRAM. They proposed a data placement and migration strategy at the applicative level and validated it on key value data stores.

In Xu et al. (2015), an 8Gb ReRAM chip is modelled based on cell array structures without selection transistor (0T1R). The chip is compatible with the DDR3 interface and constrained by the same power and area budget as a DRAM of the same capacity. The authors have optimized the write latency by means of microarchitectural level enhancements and data encoding. Their simulations show less than 10% of performance degradations with respect to a main memory system using DRAM chips.

6.2.3 ReRAM in Storage Systems. Many researchers found that the ReRAM endurance cannot fit in a low-level memory hierarchy; consequently, ReRAM has been regarded as future replacement or complement for flash memory.

In Tanakamaru et al. (2014), the authors studied a hybrid storage system composed of a small volume of ReRAM and a large volume of flash memory. The integration of ReRAM was horizontal, with the objective being to store mirroring information and parity and error records, as ReRAM is non-volatile, page rewritable, and has better performance and endurance than flash. The objective of the proposed Unified Solid State Storage system (USSS), which implements reverse mirroring, shift mirroring, error reduction synthesis, and page-RAID, was to improve the acceptable raw bit error rate (ABER) of the memory.

In Jung et al. (2013), the authors highlighted the main issues in designing a large-scale crossbar ReRAM, such as write disturbance and read sensing margin (Jung et al. 2013). The authors argued that without taking this issue into consideration, the sensing margin would be unacceptable for a large-scale ReRAM. The authors proposed some microarchitectural solutions based on a new access strategy to deal with this issue. They also proposed a macroarchitecture design and evaluated the performance of an 8GB ReRAM as compared to two flash memory configurations (integration as a replacement of flash memory). The proposed architecture gave up to eight times better bandwidth and between 66% and 88% shorter latency.

Other studies used ReRAM horizontally integrated in a solid-state device to absorb some specific patterns and relieve flash memory from intensive and random fragmented data updates. In Sun et al. (2014) and Fujii et al. (2012), the authors proposed a horizontal integration of ReRAM together with NAND flash memory; the idea was to use ReRAM for highly written and random data to optimize the overall performance.

6.3 Open Research Issues

Open research issues and challenges related to ReRAM have been discussed in Xue et al. (2011) and (Yang and Williams 2013) and are summarized in this section.

6.3.1 The Switching Endurance. The switching endurance is related to the number of cycles that the device can be switched ON and OFF reversibly before its performance degrades to a non-tolerable threshold. The endurance values reported in the literature are far from the limit of metal oxide memristive devices. The key element resides in the choice of the switching material. Many switching materials have been investigated, such as Pt/TaO$_x$/Ta, TiO$_x$, HfOx (Yee et al. 2010), and WOx (Chien 2010). Some showed an endurance of close to 10 billion switching cycles, which is still far from the looked-for universal memory endurance of about 10^{15}–10^{16}.

6.3.2 The Device Yield. To make ReRAM viable for manufacturing, the device yield should be enhanced, as it has been reported as being low (Yang et al. 2010). One of the key parameters in determining this yield is the control of the switching channels inside the device (Yang and Williams 2013). Some state-of-the-art solutions have improved the yield through seeding nanoscale switching centers (Yang et al. 2010).

6.3.3 The Operation Energy. Even though the energy required to switch the ReRAM cell is very small compared to other NVMs such as flash and PCM, it is still very important to lower its value to increase the memory density. The energy required for switching is larger than that required for reading. There are two main solutions to lowering the energy consumed in switching: (1) lowering the junction size, or (2) optimizing the switching material (Miao et al. 2011).

6.3.4 Electroforming Process. Electroforming is the application of high voltage or a current, which needs to be performed at one time to enable the switching of the resistive memory cell. This process produces a reduction in electronic conductivity (Xue et al. 2011; Yang et al. 2009). After this, the device operates with widely varying properties according to the material that is used. It is very important to limit this variance in computing circuit applications (Yang et al. 2009).

6.3.5 Sneak Path and Nonlinearity. The sneak path is an unexpected path in the crossbar structure, which may lead to write and read failures (Jung et al. 2013). It occurs due to the absence of a transistor or any access device on the ReRAM crossbar structure. Preventing sneak paths is a necessary property for achieving a large-scale ReRAM memory (Chen et al. 2011a, 2011b). To solve the issue of sneak paths without introducing an access device, a highly nonlinear current-voltage relation for the ON state is required (Yang and Williams 2013). In fact, the higher the non-linearity, the better the memory cell as the sneak current is significantly reduced (Niu et al. 2012), and the better the scaling of the write current and the energy consumption. As discussed in Jung et al. (2013), state-of-the-art sneak path solutions may introduce some other issues such as write disturbances and a read sensing margin that can prevent ReRAM from scaling and thus from becoming a candidate for replacing storage systems.

To perform design space exploration of ReRAM, many tools were developed such as NVSim (Dong et al. 2012), and models were built to study and optimize many parameters such as area, latency, energy (Xu et al. 2011), and reliability (Xu et al. 2014), or to study trade-offs to achieving a multi-level cell ReRAM (Xu et al. 2013).

6.4 ReRAM Maturity and the State of the Market

The ReRAM is still under development, and some prototypes have been provided by the Rambus Company and the Crossbar Company. In February 2012, Rambus bought a ReRAM company called Unity Semiconductor, and Panasonic launched a ReRAM evaluation kit in May 2012. The evaluation kit is based on a tantalum oxide 1T1R memory cell architecture. In 2013, Crossbar introduced a ReRAM prototype in the form of a chip about the size of a postage stamp that can store 1 TB of data. In August 2013, the Crossbar Company claimed that large-scale production of their ReRAM chips was scheduled for 2015. This same year, Intel and Micron introduced a so-called 3D Xpoint memory bearing similarities with ReRAM and that was being developed by Crossbar.

In 2016, Western Digital announced at the Flash Memory Summit that it intends to use 3D Resistive RAM (ReRAM) as storage class memory (SCM) for its future special-purpose ultra-fast SSDs (Shikov 2016).

7 CONCLUSION

In this article, we have introduced four very promising NVM technologies: PCM, MRAM, FeRAM, and ReRAM, each of which has its pros and cons. Those NVMs are in different stages of development, with some are already being manufactured (such as FeRAM in some MCUs) and others still in the prototype phase. Those NVMs have been discussed in state-of-the-art studies and integrated according to many factors such as endurance, write latency, energy, and process fabrication compatibility.

Two main ways of integrating NVMs have been proposed: horizontal or vertical. Horizontal integration consists of complementing the existing level of memory in the memory hierarchy pyramid, thereby providing at least two options for storing data (with generally no redundancy). Data placement in this case obeys some pattern-related properties based mainly on two characteristics: types of operation and access patterns. In vertical integration an additional level is inserted into the memory hierarchy. The inserted level acts as a cache or a buffer to cope with the performance gap between two memory technologies.

From the applicative or even the operating system points of view, there are mainly two options for the integration. First, it is possible to hide the introduced memory from the operating system and/or the application, just as if there were different levels of cache memory, where the operating system knows about the existence of a cache, but does not know how many levels there are, nor does it have the ability to manage the cache, as this managing is done in the hardware. The second option consists of making the newly integrated memory visible to the operating system and eventually to the applications, so it is possible to choose the memory in which to store data.

NVM brings new research opportunities for optimizing computing systems. It should be an interesting direction to continuously exploit specific characteristics of NVM. For instance, compared with DRAM, all NVM techniques have superior static power but with worse write performance and endurance. Therefore, more effective hardware/software techniques need to be developed to utilize these features. Moreover, integrating NVM into the memory/storage systems may introduce new heterogeneous traffics. For instance, I/O traffics can be passed to the memory system (Chen et. al. 2016). It will be a challenge to develop effective architectural techniques to handle such heterogeneity with small modification to existing systems. Finally, with NVM integration, it is possible to build a uniform memory/storage system. It should be interesting to investigate how to fully exploit this uniform memory/storage system at the software level.

REFERENCES

J. Ahn and K. Choi. 2012. Lower-bits cache for low power STT-RAM caches. In *Proceedings of ISCAS*, 480–483.

A. Akel, A. M. Caulfield, T. I. Mollov, R. K. Gupta, and S. Swanson. 2011. Onyx: A protoype phase change memory storage array. In *Proceedings of HotStorage*.

H. Akinaga and H. Shima. 2010. Resistive random access memory (ReRAM) based on metal oxides. *Proc. IEEE* 98, 2 (2010), 2237–2251.

A. Athmanathan, M. Stanisavljevic, N. Papandreou, H. Pozidis, and E. Eleftheriou. 2016. Multilevel-cell phase-change memory: a viable technology. *IEEE J. Emerg. Select. Top. Circ. Syst.* 6, 1 (2016), 87–100.

A. Awad, S. Blagodurov, and Y. Solihin. 2016. Write-aware management of NVM-based memory extensions. In *Proceedings of ICS*. Article 9.

S. Baek, J. Choi, D. Lee, and S. H. Noh. 2013. Energy-efficient and high-performance software architecture for storage class memory. *ACM Trans. Embed. Comput. Syst.* 12, 3 (2013). Article 81.

R. Bishnoi, F. Oboril, M. Ebrahimi, and M. B. Tahoori. 2014. Avoiding unnecessary write operations in STT-MRAM for low power implementation. In *Proceedings of ISQED*. 548–553.

M. Bjørling, P. Bonnet, L. Bouganim, and N. Dayan. 2013. The necessary death of the block device interface. In *Proceedings of CIDR*.

S. Bock, S. B. Childers, R. Melhem, D. Mosse, and Y. Zhang. 2011. Analyzing the impact of useless write-backs on the endurance and energy consumption of PCM main memory. In *Proceedings of ISPASS*. 56–65.

J. Boukhobza. 2013. Flashing in the cloud: shedding some light on NAND flash memory storage systems. 2013. *Data Intensive Storage Services for Cloud Environments*. IGI Global, 241–266.

J. Boukhobza and P. Olivier. 2017. Flash memory integration, Elsevier.

G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy. 2008. Overview of candidate device technologies for storage-class memory. *IBM J. Res. Dev.* 52, 4 (2008), 449–464.

J. Carter and K. Rajamani. 2010. Designing energy efficient servers and data centers. *Computer* 43, 7 (2010), 76–78.

A. M. Caulfield, J. 0 Coburn, T. Mollov, A. De, A. Akel, J. He, A. Jagatheesan, R. K. Gupta, A. Snavely, and S. Swanson. 2010. Understanding the impact of emerging non-volatile memories on high-performance, io-intensive computing. In *Proceedings of SC*, 1–11.

J. Chen, R. C. Chiang, H. H. Huang, and G. Venkataramani. 2012. Energy-aware writes to non-volatile main memory. *ACM SIGOPS Operat. Syst. Rev.* 45, 3 (2012), 48–52.

Y. Chen, H. Li, W. Zhang, and R. E. Pino. 2011a. 3D-HIM: A 3D high-density interleaved memory for bipolar RRAM design. In *Proceedings of NANOARCH*. 59–64, 2011.

Y. Chen, H. Li, Y. Chen, and R. E. Pino. 2011b. 3D-ICML: A 3D bipolar ReRAM design with interleaved complementary memory layers. In *Proceedings of DATE*. 1–4.

Y. Chen, W. Wong, H. Li, C. Koh, Y. Zhang, and W. Wen. 2013. On-chip caches built on multilevel spin-transfer torque RAM cells and its optimizations. *ACM J. Emerg. Technol. Comput. Syst.* 9, 2 (2013).

R. Chen, Z. Shao, and T. Li. 2016. Bridging the I/O performance gap for big data workloads: A new NVDIMM-based approach. In *Proceedings of MICRO*. 1–12.

H. Y. Cheng et al. 2016. LAP: Loop-block aware inclusion properties for energy-efficient asymmetric last level caches. In *Proceedings of ISCA*. 103–114.

W. C. Chien. 2010. Unipolar switching behaviors of RTO WOX RRAM. *IEEE Electron. Device Lett.* 31, 2 (2010), 126–128.

B. J. Choi et al. 2005. Resistive switching mechanism of TiO2 thin films grown by atomic-layer deposition. *J. Appl. Phys.* 98, 033715 (2005).

L. Chua. 1971. Memristor—the missing element. *IEEE Trans. Circuit Theory.* 18, 5 (1971), 507–519.

F. Clermidy, N. Jovanovic, S. Onkaraiah, H. Oucheikh, O. Thomas, O. Turkyilmaz, E. Vianello, J. Portal, and M. Bocquet. 2014. Resistive memories: Which applications? In *Proceedings of DATE.*

G. Dhiman, R. Ayoub, and T. Rosing. 2009. PDRAM: A hybrid PRAM and DRAM main memory system. In *Proceedings of DAC.* 664–469.

I. H. Doh, J. Choi, D. Lee, and S. H. Noh. 2007. Exploiting non-volatile RAM to enhance flash file system performance. In *Proceedings of EMSOFT.* 164–173.

W. Dong et al. 2015. Minimizing update bits of NVM-based main memory using bit flipping and cyclic shifting. In *Proceedings of HPCC/CSS/ICESS.* 290–295.

X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. 2012. NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 31, 7 (2012), 994–1007.

X. Dong, N. P. Jouppi, and Y. Xie. 2013. A circuit-architecture co-optimization framework for evaluating emerging memory hierarchies. In *Proceedings of ISPASS.* 140–141.

Y. Du, M. Zhou, B. R. Childers, D. Mossé, and R. Melhem. 2013. Bitmapping for balanced PCM cell programming. In *Proceedings of ISCA.* 428–439.

S. R. Dulloor, A. Roy, Z. Zhao, N. Sundaram, N. Satish, R. Sankaran, J. Jackson, and K. Schwan. 2016. Data tiering in heterogeneous memory systems. In *Proceedings of EuroSys.* Article 15.

K. Eshraghian et al. 2011. Memristor MOS content addressable memory (MCAM): Hybrid architecture for future high performance search engines. *IEEE Trans. on VLSI Systems.* 19, 8 (2011), 1407–1417.

Everspin Technologies Inc. 2015. Retrieved November 2016 from https://www.everspin.com/64mb-spin-torque-mram-ddr3-dram-compatible.

H. Fujii, K. Miyaji, K. Johguchi, K. Higuchi, C. Sun, and K. Takeuchi. 2012. x11 performance increase, x6.9 endurance enhancement, 93% energy reduction of 3D TSV-integrated hybrid ReRAM/MLC NAND SSDs by data fragmentation suppression. In *Proceedings of VLSIC.* 134–135.

M. Fujimoto, H. Koyama, Y. Hosoi, K. Ishihara, and S. Kobayashi. 2006. High-speed resistive switching of TiO2/TiN nano-crystalline thin film, japan. *J. Appl. Phys.* 45, 8/11 (2006), L310–L312.

Y. Fujisaki. 2013. Review of emerging new solid-state non-volatile memories. *Japan. J. Appl. Phys.* 52 (2013).

S. Gao, B. He, and J. Xu. 2015. Real-time in-memory checkpointing for future hybrid memory systems. In *Proceedings of ICS.* 263–272.

J. F. Gibbons and W. E. Beadle. 1964. Switching properties of thin NiO films. *Solid State Elect.* 7, 11 (1964), 785–790.

N. Goswami, C. Bingyi, and L. Tao. 2013. Power-performance co-optimization of throughput core architecture using resistive memory. In *Proceedings of HPCA.* 342–353.

X. Guo, E. Ipek, and T. Soyata. 2010. Resistive computation: avoiding the power wall with low-leakage, STT-MRAM based computing. In *Proceedings of ISCA.* 371–382.

S. Gurumurthi. 2009. Architecting storage for the cloud computing area. *IEEE Micro.* 29, 6 (2009), 68–71.

A. Hassan, H. Vandierendonck, and D. S. Nikolopoulos. 2015. Energy-efficient in-memory data stores on hybrid memory hierarchies. In *Proceedings of DaMoN.*

Y. Hosoi et al. 2006. High speed unipolar switching resistance RAM (RRAM) technology. *Proceedings of IEDM.* 30.7.1–30.7.4.

Y. Huai. 2008. Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects. *AAPPS Bulletin.* 18, 6 (2008), 33–40.

H. F. Huang and T. Jiang. 2014. Design and implementation of flash based NVDIMM. In *Proceedings of NVMSA.* 1–6.

A. Jadidi, M. Arjomand, and H. Sarbazi-Azad. 2011. High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement. In *Proceedings of ISLPED.* 79–84.

L. Jiang, Y. Zhang, and J. Yang. 2012. Elastic RESET for low power and long endurance MLC based phase change memory. In *Proceedings of ISPLED.* 39–44.

L. Jiang, Y. Zhang, and J. Yang. 2014. Mitigating write disturbance in super-dense phase change memories. In *Proceedings of DSN.* 216–227.

L. Jiang, B. Zhao, J. Yang, and Y. Zhang. 2014. A low power and reliable charge pump design for phase change memories. In *Proceedings of ISCA.* 397–408.

Y. Jin, M. Shihab, and M. Jung. 2014. Area, power and latency considerations of STT-MRAM to substitute for main memory. In *Proceedings of The Memory Forum.*

A. Jog, A. K. Mishra, X. Cong, X. Yuan, V. Narayanan, R. Iyer, and C. R. Das. 2012. Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs. In *Proceedings of DAC.* 243–252.

M. R. Jokar, M. Arjomand, and H. Sarbazi-Azad. 2016. Sequoia: A high-endurance NVM-based cache architecture. *IEEE Transactions on VLSI Systems.* 24, 3 (2016), 954–967.

Y. Joo, D. Niu, X. Dong, G. Sun, N. Chang, and Y. Xie. 2010. Energy-and endurance-aware design of phase change memory caches. In *Proceedings of DATE.* 136–141.

J. Jung, Y. Won, E. Kim, H. Shin, and B. Jeon. 2010. FRASH: Exploiting storage class memory in hybrid file system for hierarchical storage. *ACM Trans. On Storage* 6, 1 (2010), Article 3.

J. Jung, Y. Nakata, M. Yoshimoto, and H. Kawaguchi. 2013. Energy-efficient spin-transfer torque RAM cache exploiting additional all-zero-data flags. In *Proceedings of ISQED*. 216–222.

M. Jung, J. Shalf, and M. Kandemir. 2013. Design of a large-scale storage-class RRAM system. In *Proceedings of ICS*. 103–114.

H. Kanaya et al. 2004. A 0.602 μm^2 nestled chain cell structure formed by one mask etching process for 64 Mbit FeRAM. In *Proceedings of VLSl Technology*. 150–151.

D. Kang, S. Baek, J. Choi, D. Lee, S. H. Noh, and O. Mutlu. 2015. Amnesic cache management for non-volatile memory. In *Proceedings of MSST*. 1–13.

S. Kannan, A. Gavrilovska, and K. Schwan. 2016. pVM: Persistent virtual memory for efficient capacity scaling and object storage. In *Proceedings of EuroSys*. Article 13.

Y. Kato, T. Yamada, and Y. Shimada. 2005. 0.18-lm nondestructive readout FeRAM using charge compensation technique. *IEEE Trans. Elect. Dev.* 52, 12 (2005), 2616–2621.

K. Kim and S. J. Ahn. 2005. Reliability investigations for manufacturable high density PRAM, Reliability. In *Proceedings of IRPS*. 157–162.

K. Kim. 2008. Future memory technology: challenges and opportunities. In *Proceedings of VLSI-TSA*, 5–9.

H. Kim, S. Seshadri, C. L. Dickey, and L. Chiu. 2014. Evaluating phase change memory for enterprise storage systems: A study of caching and tiering approaches. *ACM Trans. Storage* 10, 4 (2014), Article 15.

Y. Kim et al. 2011. Bi-layered RRAM with unlimited endurance and extremely uniform switching. In *Proceedings of VLSIT*, 52–53.

P. Kogge et al. 2008. Exascale computing study: technology challenges in achieving exascale systems. *DARPA Information Processing Techniques Office.*

M. P. Komalan et al. 2013. Design exploration of a NVM based hybrid instruction memory organization for embedded platforms. *Design Auto. Embed. Syst.* 17, 3–4 (2013), 459–483.

M. Komalan, J. I. Gómez Pérez, C. Tenllado, P. Raghavan, M. Hartmann, and F. Catthoor. 2014. Feasibility exploration of NVM based I-cache through MSHR enhancements. In *Proceedings of DATE,* Article 21.

M. H. Kryder and C. S. Kim. 2009. After hard drives—what comes next? *IEEE Trans. Magnet.* 45, 10 (2009), 3406–3413.

E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu. 2013. Evaluating stt-ram as an energy-efficient main memory alternative. In *Proceedings of ISPASS*. 256–267.

K. Kwon, S. H. Choday, Y. Kim, and K. Roy. 2014. AWARE (asymmetric write architecture with redundant blocks): a high write speed STT-MRAM cache architecture. *IEEE Trans. on VLSI Systems.* 22, 4 (2014), 712–720.

B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. 2009. Architecting phase change memory as a scalable dram alternative. In *Proceedings of ISCA*. 2–13.

H. Y. Lee et al. 2010. Evidence and solution of over-RESET problem for HfOX based resistive memory with sub-ns switching speed and high endurance. In *Proceedings of IEDM*. 19.7.1–19.7.4.

D. Lee et al. 2006. Excellent uniformity and reproducible resistance switching characteristics of doped binary metal oxides for non-volatile resistance memory applications. In *Proceedings of IEDM*. 30.8.1–30.8.4.

E. Lee, H. Bahn, S. Yoo, and S. H. Noh. 2014. Empirical study of NVM storage: An operating system's perspective and implications. In *Proceedings of MASCOTS*. 405–410.

S. Lee, H. Bahn, and S. H. Noh. 2014. CLOCK-DWF: A write-history-aware page replacement algorithm for hybrid PCM and DRAM memory architectures. *IEEE Trans. Comput.* 63, 9 (2014), 2187–2200.

D. Li, J. S. Vetter, G. Marin, C. McCurdy, C. Cira, Z. Liu, and W. Yu. 2012. Identifying opportunities for byte-addressable non-volatile memory in extreme-scale scientific applications. In *Proceedings of IPDPS*. 945–956.

J. Li, C. Xue, and Y. Xu. 2011. STT-RAM based energy-efficiency hybrid cache for CMPs. In *Proceedings of VLSI-SoC*. 31–36.

X. Li, K. Lu, X. Wang, and X. Zhou. 2012. NV-process: A fault-tolerance process model based on non-volatile memory. In *Proceedings of APSys*. Article 1.

Y. Li, Y. Chen, and A. Jones. 2012. A software approach for combating asymmetries of non-volatile memories. In *Proceedings of ISLPED*. 191–196.

Q. Li, J. Li, L. Shi, M. Zhao, J. C. Xue, and Y. He. 2014. Compiler-assisted STT-RAM-based hybrid cache for energy efficient embedded systems. *IEEE Trans. VLSI Syst.* 22, 8 (2014), 1829–1840.

Y. Liu, C. Zhou, X. Cheng, Hybrid SSD with PCM. 2011. In *Proceedings of NVMTS*. 1–5.

J. S. Meena, S. M. Sze, U. Chand, and T. Tseng. 2014. Overview of emerging non-volatile memory technologies. *Nanoscale Res. Lett.* 9, 526, (2014), 1–33.

F. Miao et al. 2011. Anatomy of a nanoscale conduction channel reveals the mechanism of a high-performance memristor. *Adv. Mater.* 23 (2011) 5633–5640.

A. Mirhoseini, M. Potkonjak, and F. Koushanfar. 2012. Coding-based energy minimization for phase change memory. In *Proceedings of DAC*. 68–76.

S. Mittal. 2013. Using cache-coloring to mitigate inter-set write variation in non-volatile caches. Technical report, Iowa State University.

S. Mittal and J. S. Vetter. 2015a. **A**YUSH: A technique for extending lifetime of SRAM-NVM hybrid caches. *IEEE Comput. Architect. Lett.* 14, 2 (2015), 115–118.

S. Mittal, J. S. Vetter, and D. Li. 2015b. A survey of architectural approaches for managing embedded DRAM and non-volatile on-chip caches. *IEEE Trans. Parallel Distrib. Syst.* 26, 6 (2015), 1524–1537.

S. Mittal and J. S. Vetter. 2016. A survey of software techniques for using non-volatile memories for storage and main memory systems. *IEEE Trans. Parallel Distrib. Syst.* 27, 5 (2016), 1537–1550.

S. Mittal. 2016. A survey of techniques for architecting processor components using domain-wall memory. *J. Emerg. Technol. Comput. Syst.* 13, 2, (2016), Article 29.

J. C. Mogul, E. Argollo, M. Shah, and P. Faraboschi. 2009. Operating system support for NVM + DRAM hybrid main memory. In *Proceedings of HotOS*.

O. Mutlu. 2015. Main memory scaling: challenges and solution directions. *More than Moore Technologies for Next Generation Computer Design*, Springer. 127–153.

D. Niu, C. Xu, N. Muralimanohar, N. P. Jouppi, and Y. Xie. 2012. Design trade-offs for high density cross-point resistive memory. In *Proceedings of ISLPED*. 209–214.

F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori. 2015. Evaluation of hybrid memory technologies using SOT-MRAM for on-chip cache hierarchy. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 34, 3 (2015), 367–380.

S. Oikawa. 2014. Independent kernel/process checkpointing on non-volatile main memory for quick kernel rejuvenation. In *Proceedings of ARCS*. 233–244.

X. Ouyang, D. Nellans, R. Wipfel, D. Flynn, and D. K. Panda. 2011. Beyond block I/O: Rethinking traditional storage primitives. In *Proceedings of HPCA*. 301–311.

J. Park, D. Shin, and H. G. Lee. 2015. Design space exploration of row buffer architecture for phase change memory with LPDDR2-NVM interface. In *Proceedings of VLSI-SoC*. 104–109.

S. Park, Y. Kim, B. Urgaonkar, J. Lee, and E. Seo. 2011. A comprehensive study on energy efficiency of flash memory storages. *J. Syst. Architect.* 57, 4 (2011), 354–365.

S. P. Park, S. Gupta, N. Mojumder, A. Raghunathan, and K. Roy. 2012. Future cache design using STT MRAMs for improved energy efficiency: Devices, circuits and architecture. In *Proceedings of DAC*. 492–497.

Y. Park, S. K. Park, K. H. Park, Linux Kernel Support to Exploit Phase Change Memory. 2010. In *Proceedings of the Linux Symposium.* 217–224.

M. K. Qureshi, V. Srinivasan, and J. A. Rivers. 2009. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of ISCA*. 24–33.

M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-Montano. 2010. Improving read performance of phase change memories via write cancellation and write pausing. In *Proceedings HPCA*.

M. K. Qureshi, S. Gurumurthi, and B. Rajendran. 2011. Phase change memory: From devices to systems. Synthesis Lectures on Computer Architecture, Morgan & Claypool.

P. Ranganathan. 2011. From microprocessors to nanostores: Rethinking data-centric systems. *Computer* 44, 1 (2011), 39–48.

M. Rasquinha, D. Choudhary, S. Chatterjee, S. Mukhopadhyay, and S. Yalamanchili. 2010. An energy efficient cache design using spin torque transfer (STT) RAM. In *Proceedings of ISLPED*. 389–394.

D. Roberts, T. Kgil, and T. Mudge. 2009. Integrating NAND flash devices onto servers. *Commun. ACM* 52, 4 (2009), 98–106.

U. Russo, D. Ielmini, A. Redaelli, and A. L. Lacaita. 2008. Modeling of programming and read performance in phase-change memories—Part I: Cell optimization and scaling. *IEEE Trans. Electron Devices* 55, 2 (2008), 506–514.

R. Salkhordeh and H. Asadi. 2016. An operating system level data migration scheme in hybrid DRAM-NVM memory architecture. In *Proceedings of DATE*. 936–941.

M. H. Samavatian, H. Abbasitabar, M. Arjomand, and H. Sarbazi-Azad. 2014. An efficient STT-RAM last level cache architecture for GPUs. In *Proceedings DAC*. 1–6.

G. S. Sandhu. 2013. Emerging memories technology landscape. In *Proceedings of NVMTS*. 1–5.

S. Schechter, G. H. Loh, K. Straus, and D. Burger. 2010. Use ECP, not ECC, for hard failures in resistive memories, SIGARCH Comput. *Architect. News* 38, 3 (2010), 141–152.

S. Senni, L. Torres, G. Sassatelli, A. Bukto, and B. Mussard. 2014. Exploration of magnetic RAM based memory hierarchy for multicore architecture. In *Proceedings of ISVLSI*. 248–251.

S. Senni, L. Torres, G. Sassatelli, A. Gamatie, and B. Mussard. 2015. Emerging non-volatile memory technologies exploration flow for processor architecture. In *Proceedings of ISVLSI*. 460–460.

S. Sheu et al. 2011. A 4Mb embedded SLC resistive-RAM macro with 7.2ns read-write random-access time and 160ns MLC-access capability. In *Proceedings of ISSCC*. 200–202.

H. Shiga et al. 2009. A 1.6GB/s DDR2 128Mb chain FeRAM with scalable octal bitline and sensing schemes. In *Proceedings of ISSCC*. 464–465.

A. Shilov. 2016. Western digital to use 3D ReRAM as storage class memory (SCM) for special-purpose SSDs. ANANDTECH.

C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan. 2011. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *Proceedings HPCA*. 50–61.

SNIA. 2015. NVM Programming Model (NPM). Retrieved from http://www.snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.1.pdf.

D B. Sturkov, G. S. Snider, and R. S. William. 2008. The missing memristor found. *Nature* 453 (2008), 80–83.

G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. 2009. A novel architecture of the 3D stacked MRAM L2 cache for CMPs. In *Proceedings of HPCA*. 239–249.

G. Sun, Y. Joo, Y. Chen, D. Niu, Y. Xie, Y. Chen, and H. Li. 2010. A hybrid solid-state storage architecture for the performance, energy consumption, and lifetime improvement. 2010. In *Proceedings of HPCA*. 1–12.

G. Sun, D. Niu, J. Ouyang, and Y. Xie. 2011. A frequent-value based PRAM memory architecture. In *Proceedings of ASP-DAC*. 211–216.

H. Sun, K. Miyaji, K. Johguchi, and K. Takeuchi. 2014. A high performance and energy-efficient cold data eviction algorithm for 3D-TSV Hybrid ReRAM/MLC NAND SSD. *IEEE Trans. Circ. Syst.* I, 61, 2 (2014), 382–392.

Z. Sun, X. Bi, H. Li, W. Wong, Z. Ong, X. Zhu, and W. Wu. 2011. Multi retention level STT-RAM cache designs with a dynamic refresh scheme. In *Proceedings of MICRO*. 329–338

Z. Sun, Z. Jia, X. Cai, Z. Zhang, and L. Ju. 2015. AIMR: An adaptive page management policy for hybrid memory architecture with NVM and DRAM. In *Proceedings of HPCC/CSS/ICESS*. 284–289.

A. Suresh, P. Cicotti, and L. Carrington. 2014. Evaluation of emerging memory technologies for HPC, data intensive applications. In *Proceedings of CLUSTER*. 239–247.

S. Syu, Y. Shao, and I. Lin. 2013. High-endurance hybrid cache design in CMP architecture with cache partitioning and access-aware policy. In *Proceedings of GLSVLSI*. 19–24.

S. Tanakamaru, M. Doi, and K. Takeuchi. 2014. NAND flash memory/ReRAM hybrid unified solid-state-storage architecture. *IEEE Trans. Circ. Syst.* I, 61, 4 (2014), 1119–1132.

G. Uh, Y. Wang, D. Whalley, S. Jinturkar, C. Burns, and V. Cao. 1999. Effective exploitation of a zero overhead loop buffer. In *Proceedings of LCTES*. 10–19.

H. Venkatesh and S. Singh. 2015. FRAMs fit wearable electronics like a glove, electronic design. Retrieved November 2016 from http://electronicdesign.com/ memory/frams-fit-wearable-electronics-glove.

J. S. Vetter and S. Mittal. 2015. Opportunities for nonvolatile memory systems in extreme-scale high-performance computing. *Comput. Sci. Eng.* 17, 2 (2015), 73–82.

J. Wang, X. Dong, Y. Xie, and N. P. Jouppi. 2013. i2WAP: Improving non-volatile cache lifetime by reducing inter- and intra-set write variations. In *Proceedings of HPCA*. 234–245.

J. Wang, X. Dong, Y. Xie, and N. P. Jouppi. 2014. Endurance-aware cache line management for non-volatile caches. *ACM Trans. Archit. Code Optim.* 11, 1 (2014a).

J. Wang, X. Dong, and Y. Xie. 2014. Building and optimizing MRAM-based commodity memories. *ACM Trans. Archit. Code Optim.* 11, 4 (2014b).

R. Wang, L. Jiang, Y. Zhang, L. Wang, and J. Yang. 2015a. Selective restore: An energy efficient read disturbance mitigation scheme for future STT-MRAM. In *Proceedings of DAC*. Article 21.

R. Wang, L. Jiang, Y. Zhang, and J. Yang. 2015b. SD-PCM: Constructing reliable super dense phase change memory under write disturbance. In *Proceedings of ASPLOS*. 19–31.

R. Wang, L. Jiang, Y. Zhang, L. Wang, and J. Yang. 2015c. Exploit imbalanced cell writes to mitigate write disturbance in dense phase change memory. In *Proceedings of DAC*. Article 88.

Z. Wang, D. A. Jimenez, X. Cong, S. Guangyu, and Y. Xie. 2014. Adaptive placement and migration policy for an STT-RAM-based hybrid cache. In *Proceedings of HPCA*. 13–24.

Z. Wang, Z. Gu, M. Yao, and Z. Shao. 2015. Endurance-aware allocation of data variables on NVM-based scratchpad memory in real-time embedded systems. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 34, 10 (2015), 1600–1612.

Q. Wei, J. Chen, and C. Chen. 2015. Accelerating file system metadata access with byte-addressable nonvolatile memory. *ACM Trans. Storage* 11, 3 (2015), Article 12.

W. Wei, D. Jiang, S. A. McKee, J. Xiong, and M. Chen. 2015. Exploiting program semantics to place data in hybrid memory. In *Proceedings of PACT*. 163–173.

R. S. William. 2008. How we found the missing memristor. *IEEE Spectrum*. 45 (2008), 28–35.

R. Winter. 2008. Why are data warehouses growing so fast? Retrieved from http://www.b-eye-network.com/channels/1138/view/7188.

W. Wong. 2016. FRAM delivers unified memory for 16-bit microcontroller. *Electronic Design*.

J. Y. Wu et al. 2015. Greater than 2-bits/cell MLC storage for ultrahigh density phase change memory using a novel sensing scheme. In *Proceedings of VLSI Technology*. T94–T95.

P. Wu, D. Li, Z. Chen, J. S. Vetter, and S. Mittal. 2016. Algorithm-directed data placement in explicitly managed non-volatile memory. In *Proceedings of HPDC*. 141–152.

X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie. 2009. Hybrid cache architecture with disparate memory technologies. In *Proceedings of ISCA*. 34–45.

F. Xia, D. Jiang, J. Xiong, and N. Sun. 2015. A survey of phase change memory systems. *J. Comput. Sci. Technol.* 30, 1 (2015), 121–144.

C. Xu, X. Dong, N. P. Jouppi, and Y. Xie. 2011. Design implications of memristor-based RRAM cross-point structures. In *Proceedings of DATE*.

C. Xu, D. Niu, N. Muralimanohar, N. P. Jouppi, and Y. Xie. 2013. Understanding the trade-offs in multi-level cell ReRAM memory design. In *Proceedings of DAC*. Article 108.

C. Xu, D. Niu, N. Muralimanoha, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie. 2015. Overcoming the challenges of crossbar resistive memory architectures. In *Proceedings of HPCA*. 476–488.

C. Xu, D. Niu, Y. Zheng, S. Yu, and Y. Xie. 2014. Reliability-aware cross-point resistive memory design. In *Proceedings of GLSVLSI*. 145–150.

W. Xu, J. Liu, and T. Zhang. 2009. Data manipulation techniques to reduce phase change memory write energy. In *Proceedings of ISPLED*. 237–242.

C. J. Xue, Y. Zhang, Y. Chen, G. Sun, J. J. Yang, and H. Li. 2011. Emerging non-volatile memories: Opportunities and challenges. In *Proceedings of CODES+ISSS*. 325–334.

B. D. Yang, J. E. Lee, J. S. Kim, J. Cho, S. Y. Lee, and B. G. Yu. 2007. A low power phase-change random access memory using a data-comparison write scheme. In *Proceedings of ISCAS*. 3014–3017.

J. J. Yang and R. S. Williams. 2013. Memristive devices in computing system: Promises and challenges. *ACM J. Emerg. Technol. Comput. Syst.* 9, 2 (2013), Article 11.

J. J. Yang, F. Miao, M. D. Pickett, D. A. A. Ohlberg, D. R. Stewart, C. N. Lau, and R. S. Williams. 2009. The mechanism of electroforming of metal oxide memristive switches. *Nanotechnology* 20 215201 (2009).

J. J. Yang et al. 2010. Diffusion of adhesion layer metals controls nanoscale memristive switching. *Adv. Mater.* 22, 36 (2010), 4034–4038.

S. H. Yang, and Y. S Ryu. 2013. A buffer management for STT-MRAM based hybrid main memory in sensor nodes. In *Proceedings of ICCNCE*. 286–289.

S. Yazdanshenas, M. R. Pirbasti, M. Fazeli, and A. Patooghy. 2014. Coding last level STT-RAM cache for high endurance and low power. *Comput. Architect. Lett.* 13, 2 (2014), 73–76.

J. H. Yoon, E. H. Nam, Y. J. Seong, H. Kim, B. S. Kim, S. L. Min, and Y. Cho. 2008. Chameleon: A high performance flash/FRAM hybrid solid state disk architecture. *Comput. Architect. Lett.* 7, 1 (2008), 17–20.

K. Yoongu, R. Daly, J. Kim, C. Fallin, L. Ji Hye, L. Donghyuk, C. Wilkerson, K. Lai, and O. Mutlu. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *Proceedings of ISCA*. 361–372.

S. Yu, P. Y. Chen, Emerging Memory Technologies: Recent Trends and Prospects. 2016. *IEEE Solid-State Circuits Mag.* 8, 2 (2016), 43–56.

J. Yue, Y. Zhu. 2013a. Accelerating write by exploiting PCM asymmetries. In *Proceedings of HPCA*. 282–293.

J. Yue and Y. Zhu. 2013b. Exploiting subarrays inside a bank to improve phase change memory performance. In *Proceedings of DATE*. 386–391.

J. Yun, S. Lee, and S. Yoo. 2012. Bloom filter-based dynamic wear leveling for phase-change RAM. In *Proceedings of DATE*. 1513–1518.

Q. Zhang, L. Chang, and R. Boutaba. 2010. Cloud computing: State-of-the-art and research challenges. *J. Int. Services Appl.* 1, 1 (2010), 7–18.

P. Zhou, B. Zhao, J. Yang, and Y. Zhang. 2009a. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of ISCA*. 14–23.

P. Zhou, B. Zhao, J. Yang, and Y. Zhang. 2009b. Energy reduction for STT-RAM using early write termination. In *Proceedings of ICCAD*. 264–268.

Received January 2017; revised May 2017; accepted July 2017