A Non-Volatile Embedded System

Henriette Hofmeier
Friedrich-Alexander-Universität Erlangen-Nürnberg
heni.hofmeier@fau.de

ABSTRACT

The development of fast, byte-addressable non-volatile memory (NVM) technologies, with latencies and write endurance closer to SRAM and DRAM than to Flash, position them as possible replacements for the volatile technologies. While their non-volatility and low leakage power, among other beneficial features, make them attractive candidates for new system designs, the drawbacks of low write endurance and high write latency require hardware- and software-based adjustments. This work introduces two NVM technologies and discusses their applicability as processor registers, caches, and main memory and gives a possible design of an embedded system with non-volatile memory components.

1 INTRODUCTION

With the rise of Internet of Things (IoT)-systems and sensor networks, embedded systems are deployed more and more often, also in areas where no stable power supply is available. Therefore the power consumption of the system's components and also adequate speed as well as small system size gain more importance. While volatile memory technologies, like SRAM and DRAM, are optimized in terms of speed, this speed comes at the costs of high leakage power and high overall power demand as well as low density.

Here new non-volatile memory technologies like Phase-Change RAM (PCRAM) and Spin-Transfer Torque RAM (STT-RAM) present themselves as possible alternatives, as they provide byte-addressable data access, as well as lower power demands and higher density. Their non-volatility offers the additional benefit of providing durable memory to save, for example, the processor state across power outages. This feature can be utilized to improve the continuous forward progress of the computation of the embedded system, as less data has to be computed twice.

Though the new NVM technologies seem better fitting at first glance, there are physical drawbacks to be considered, for example, the read-write asymmetry. Writes take significantly longer than reads and also take longer than write operations on volatile memories. Also, the endurance of the NVM technologies falls short of the lifetime of volatile memories, limiting the applicability as caches and other components that are frequently written. These drawbacks have prompted researchers to find ways to lessen the performance losses and improve the lifetime and latencies. The question this work aims to answer is how a non-volatile embedded system can be designed without too much performance loss.

The following section first introduces two of the most promising non-volatile memory technologies and outlines their functionality. In Section 3, the applicability of NVMs at different levels of the memory hierarchy, namely memory elements of the processor, the caches, and the main memory, is discussed and a selection of implementation approaches are described. Section 3.4 then discusses the possible configurations of a non-volatile embedded system.

2 FUNDAMENTALS

This section introduces the properties of embedded systems as well as fundamental characteristics of non-volatile memory (NVM) technologies. Two of the most promising technologies, Phase-Change RAM (PCRAM) and Spin-Transfer Torque RAM (STT-RAM), are explained in more detail to illustrate up- and downsides of byte-addressable non-volatile memories.

2.1 What is NVM?

Byte-addressable, NVM technologies offer better read and write latency and much higher write endurance than flash memory, which enables them to contend with SRAM and DRAM. To store information, the two NVM technologies perform write operations that change the physical states. In contrast, the read operations only require low voltages to sense the resistivity. This results in a read-write asymmetry in regard to latencies and power consumption.

Two main benefits of these NVM technologies are the extremely low leakage power and the non-volatility, which renders the designs of new processors, caches, and main memories possible. Table 1 lists the read and write latency, as well as write endurance and minimal cell size of SRAM, DRAM, PCRAM, STT-RAM, and NAND-Flash to give an overview of the properties of the NVM technologies in relation to more commonly used memories. To better understand the possible implementations and challenges presented by incorporating PCRAM and STT-RAM, the structure and functionality of these technologies are presented in the following section paragraphs.

Phase-Change RAM (PCRAM) [18, 22, 28]

PCRAM makes use of chalcogenide alloys as phase-change material (PCM), which can adopt two different phases that are used to store logical values. PCMs have already been used in optical storage disks, like DVD-RAM [9] and Blu-Ray disks [6] and with recent research, the technology now positions itself as a possible alternative to DRAM. A PCRAM cell is composed of a heating element that is in contact with the PCM and is surrounded by a thermal insulator. Figure 1 shows the structure of a PCRAM cell. The insulator surrounding the heating element has been omitted to allow a clear arrangement. The region of the PCM at the contact point with the heater can then be changed between the amorphous and crystalline state by applying a current through the heating element, which causes the temperature of the programmable region of the PCM to rise. This constitutes the write operation. If the temperature is raised above the material's melting point and the material cools down quickly immediately afterward, the PCM is left in an amorphous state with high electrical resistance (RESET / '0'). Heating the material more slowly and not above the melting point but just above the crystallization point, followed by a more slowly cooling, the crystalline phase is achieved with low electrical resistance (SET,

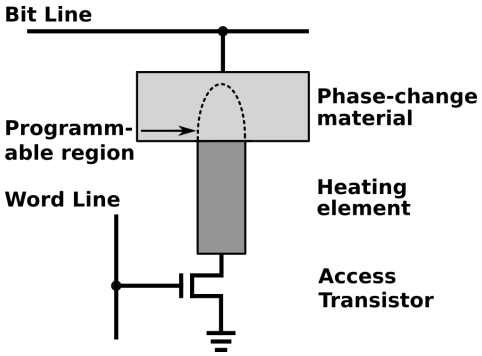

Figure 1: Structure of a Phase-Change RAM (PCRAM) cell. The dotted area indicates the volume of the programmable area.

'1'). To read the currently stored value, a low voltage is applied, and the current flowing through the cell is then measured.

As the physical state of the PCM has to be changed only for the write and not for the read operation, a read-write asymmetry [2, 34] can be observed. Writing a value to the cell results in a latency around ten times higher than the read latency, with the same asymmetry in the required energy for the operations. These high write latencies, together with a write endurance of only up to 10^8 cycles currently render PCRAM unsuitable for first or second level caches. Since PCRAM offers low read latency, comparable with DRAM, high density, and low leakage power, it has the potential to replace DRAM. Section 3.3 introduces PCRAM main memory implementations and how the high write latency and energy demand of PCRAM are managed and compensated.

Spin-Transfer Torque RAM (STT-RAM) [18, 22, 28]
One alternative to PCRAM is the Spin-Transfer Torque RAM (STT-RAM) technology, which is a variation of Magnetic RAM (MRAM) and is based on a quantum mechanical phenomenon. A STT-RAM cell comprises the magnetic-tunnel junction (MTJ): two ferromagnets that are separated by a thin tunnel barrier. The value stored in a STT-RAM cell is determined by the resistance in the MTJ, which can be manipulated by the magnetization orientations of the ferromagnets. Figure 2 shows the structure of the MTJ in a STT-RAM cell. The orientation of the magnet connected to the access transistor is fixed and is typically referenced as fixed layer. The magnetic orientation of the other magnet can be switched and is therefore called the free layer. Changing the orientation requires applying a current pulse at the fixed layer. When a certain amount of electrons pass to the free layer, its orientation will be switched, resulting in a write operation. There is low resistance in the MTJ if the two orientations are in parallel, and electrons are more likely to tunnel through the tunnel barrier (depicted in Figure 2 with the dotted orientation in the free layer). This state represents a logical '1'. Consequently, high resistance in the MTJ is created if the magnetic orientations are anti-parallel, constituting a logical '0'. A bias voltage is applied to read the value stored in the cell, and the current passing through the cell is measured.

As a significantly lower current is required for reading than for writing a value, the read-write-asymmetry regarding latency and necessary energy for the operation is established. While the latency and energy demand of reading is comparable to SRAM, the STT-RAM write has a significantly higher latency and energy demand. Despite this drawback, STT-RAM is a possible contender to take the place of SRAM in caches, due to its lower leakage power and high density. Section 3.2 introduces STT-RAM cache implementations and presents approaches to reduce the write latency and energy demand.

2.2 Characteristics of an Embedded System

Embedded systems often face strict requirements, especially if deployed in environments with unstable power supply. The following sections point out the most important properties and the components relevant to meeting these requirements.

Systems Requirements As summarized by Lohmann et al. [13], three essential properties of embedded systems are energy, timeliness, and dependability. Meeting energy constraints is crucial when dealing with unstable power supplies. The energy consumption should be kept to a minimum to allow long run-times of the system and should be deterministic in order for the system to function reliably. Timeliness is essential if the embedded system is deployed in a real-time context and data has to be available after a specific time, so that it may be transferred to other systems or is at the disposal of the next task. If non-volatile memories are used for registers, caches, and main memory, consistency is a key feature that has to be ensured. Multiple factors can affect the data consistency of a system, for example, unexpected power outages can cause corrupted data structures in non-volatile memories if an update was interrupted. Here the interdependence of the properties comes to light. Consistency requires algorithms ensuring atomic updates and back-ups but also a deterministic energy consumption to correctly time these operations.

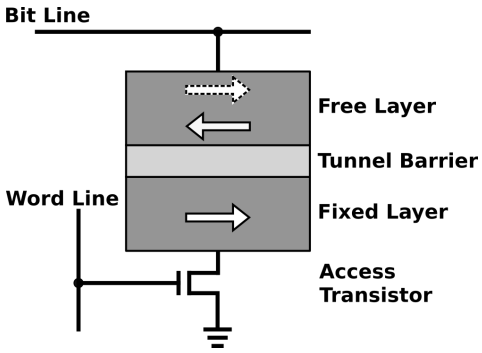

Figure 2: Structure of a Spin-Transfer Torque RAM (STT-RAM) cell, with focus on the magnetic-tunnel junction (MTJ), which comprises the two ferromagnets (free and fixed layer) and the tunnel barrier. The arrows visualize the magnetic orientation. The cell is in anti-parallel orientation, which constitutes high resistance (logical '0'). The dotted arrow depicts the upper magnet's orientation in a low resistance state(logical '1').

		SRAM (512 KB)	DRAM	PCRAM	STT-RAM (1 MB)	NAND-Flash
Latency	Write	1.227 ns†	50 ns*	500 ns*	10.218 ns†	500 μs *
	Read	1.277 ns†	50 ns*	50 ns *	1.340 ns†	25 μs *
Endurance [cycles]		$> 10^{16}$ $^{\bullet}$	$> 10^{15}$ *	$> 10^{8}$ *	$> 10^{12}$ ^{a}	$> 10^{4}$ *
Minimum Cell Size [F^2]		140$^{\bullet}$	6*	4*	22^{a}	4*

Table 1: Overview of the properties of different memory technologies, as reported by [17]*, [18]$^{\bullet}$, [8]†, and [24]^{a}. *F* denotes the feature size of the transistor.

Relevant Components Bailey et al. [1] point out several implications NVM could have on the design of systems and operating systems. Having fast, long-lasting, byte-addressable, and persistent memory technologies can pave the way for systems with only one type of memory or only one level of physical memory. This would eliminate the need for paging, as all data could reside in a fast non-volatile main memory, though this design would require other means for memory protection and allocation techniques.

NVMs can also change the functioning of applications and processes. If data can be retained across reboots, the execution state of processes could be stored in non-volatile memory and allow the application to resume its execution at the point it left off. This requires reliable schemes to ensure data consistency, either through checkpointing or making use of atomic transactions. Checkpointing can also be performed on the processor level, for example, with the use of non-volatile registers, effectively creating back-ups of the pipeline state and avoiding re-execution of already processed instructions [12, 14, 15].

3 NON-VOLATILE EMBEDDED SYSTEM

The following section looks at the processor, the caches, and the main memory to discuss how these components can be rethought with non-volatile memory (NVM) technologies. The positive influences NVMs can have on an embedded system are examined as well as challenges due to less beneficial properties and how those can be compensated. Additionally, possible implementations are introduced. This section only provides a summary of the research in this area and does not claim to be complete.

3.1 Non-Volatile Processor (NVP)

All memory components that reside in the processor, starting from the register file and program counter (PC), up to the buffers, queues, and lists required for pipelining, are usually implemented in volatile memory technologies. Designing a non-volatile processor (NVP) is an elaborate undertaking, but the use of NVM offers the advantage of improved forward progression through the availability of back-ups. If the system is powered by a battery charged by energy harvesting, this feature even becomes essential. A system with a volatile processor loses its state if a power outage occurs along with all data not yet written to a NVM. At power-up, the old state has to be recalculated, resulting in several instructions being executed twice unnecessarily. Here a NVP can keep its state or at least a back-up of a recent state, depending on the chosen back-up scheme. The cycles saved may seem insignificant, but especially in the embedded context, frequencies are often lower, and even a few cycles saved have a significant impact on the forward progression of the system.

Design As mentioned above, the implementation of back-ups is an essential element when designing a NVP. The location of the back-up, the components to be backed-up, and timing, as well as the frequency of back-up operations, have to be determined. With only NVMs, all components, like the register file, are non-volatile, which allows them to keep their state during power outages. This design requires significant efforts to ensure that the system only transitions from one valid state to another. An additional central NVM block would allow actual back-ups [15] and enable the system to resume execution in a guaranteed valid state after a restore operation. Paired with a non-volatile controller to issue read and write signal as proposed by Liu et al. [12], this offers a suitable structure for the back-ups of a NVP.

Ma et al. [14] divide processor components into architecture state (register file, PC), microarchitecture state (e.g., pipelinelatches, reorder buffer, and load/store queue), and performance-enhancing components (e.g., branch history table, branch target buffer). Based on these three categories, different back-up strategies are developed, with their efficiency depending on the timing restrictions of the system. For example, including performance enhancers in the back-up improves the forward progress significantly, as the entire pipeline state is saved and does not have to be recalculated. However, this comes at the cost of longer back-up execution times, and also the restore operations require more time than before.

Determining the timing of the back-ups is as important as deciding which components to store. Two approaches present themselves: back-ups are either performed periodically or on-demand. Periodic back-ups can occur in fixed intervals or at statically determined code lines, as proposed by Xie et al. [29]. Their approach utilizes static code analysis to choose back-up points so that consistent states are stored. They identify load and store instructions to the same address, or rather the code in between those instructions, as potential error locations. Their proposed checkpointing algorithm identifies load/store pairs and tracks the distance between checkpoints. Checkpoints are inserted before stores, as well as after a predetermined number of instructions to ensure regular back-ups. If back-ups should only be triggered if necessary, the power supply presents a reasonable metric. Ma et al. [15] propose back-up schemes that are initiated by the power supply dropping beneath a certain threshold. The threshold is either chosen so that the remaining energy only just exceeds the level required for the completion of the back-up routine (On Demand All Back-Up). Alternatively, the remaining energy suffices to ensure the completion of the current PC and the entire back-up routine (On Demand Selective Back-Up). The latter option does improve the forward progression even further, as it saves an additional clock cycle when the last PC before

back-up is not executed twice. Though one cycle does not seem significant, it may be crucial in settings where power outages happen frequently, or the system's frequency is relatively low.

Corresponding to the on-demand back-up schemes, the restore process can only start if enough power is available to guarantee the completion of the restore as well as the next back-up routine. While this ensures consistency and forward progress, it may result in longer sleep phases, as the power supply has to surpass a certain threshold, which may take some time without a stable power supply. One method to reduce the required energy is not restoring the entire processor state immediately but reading them from the NVM when required by for the next computation (On-demand recovery [14]).

Challenges & Alternatives The memory components of a processor are written very frequently, and current NVM technologies do not offer write endurance that would make the use of only non-volatile components feasible. Instead, non-volatile components could be used to duplicate volatile ones and thus offering individual back-up locations. This would allow the all-in-parallel back-up scheme proposed by Ma et al. [15].

3.2 Non-Volatile Caches

Current SRAM cache implementations suffer from low density as well as high power leakage, affecting the power consumption of the entire system [16]. This causes limitations to the deployment of systems that include environment sensors and do not have access to a stable power supply. Non-volatile memory technologies have significantly lower leakage power and lower overall power demand. Additionally, their higher density allows for larger on-chip caches to be incorporated.

Although the PCRAM offers the highest density, the STT-RAM technology is more suitable for the implementations of hardware caches. Its density, while lower than PCRAM, still surpasses SRAM's density (see Table 1). In addition, it offers much shorter latencies than PCRAM, with the read latency already coming close to the SRAM read operation. Together with STT-RAM's write endurance of up to 10^{12} cycles [24], and the lower power consumption, the STT technology is a strong contender for replacing SRAM.

Challenges Two main drawbacks of STT-RAM lie with the write operation (the high latency and high energy consumption) and the comparatively low write endurance.

(i) Write Latency and Energy: While STT-RAM's high write latency may be tolerable in a last level cache (LLC), it would have a significant negative impact on the performance of the whole system if implemented as an L1 cache. One way to make use of STT-RAM as a higher-level cache is to relax its non-volatility, through reducing the retention time, in exchange for write performance. The retention time denotes the expected time until a random bit-flip occurs due to thermal instability. While high thermal stability is desirable for reliable, non-volatile memories, it also causes the high write latency and energy, as more time or higher currents are necessary to perform the write operation [23]. Thus, to increase the write performance, the required switching current of the MTJ has to be decreased, which will simultaneously result in the lower retention time. There are multiple possibilities to achieve this effect, in the following approaches by Smullen et al. [23] and by Sun et al. [24] are introduced.

To improve the write performance of STT-RAM Smullen et al. reduce the magnetic-tunnel junction (MTJ) planar area from 32 F^2 down to 10 F^2, where F is the feature size of the transistor[1]. This change shortens the retention time to 56 µs, as the required switching density is reduced. This results in read and write latencies comparable to SRAM, with a write requiring only one cycle more, when using STT-RAM. The change also has a considerable impact on the write energy, which is reduced by over 69% compared to the non-relaxed implementations. However, this comes at the cost of slightly higher leakage power.

Sun et al. [24] argue that retention relaxation by reducing the planar area is not feasible, as the MTJ will usually already be designed as small as possible. Instead, they change different physical properties, for example, the thickness of the free layer to reduce the required switching density. Table 2 lists their evaluation results for L1 caches with a remaining retention time of 3.24 s and 26.5 µs, compared to an unrelaxed STT cache and a SRAM cache. While the short retention times may induce the need for refresh actions (which could counter the positive impact on the power demand), they observe that the write latency comes close to the one of SRAM. This comes at the cost of higher leakage power, though it is still below the amount of SRAM leakage.

(ii) Write Endurance: Apart from the high write latency and energy, STT-RAM's low write endurance also limits its applicability as a cache technology. Compared to SRAM, which has a write endurance of up to 10^{16} cycles before erroneous behavior is expected, STT-RAM can only guarantee up to 10^{12} writes [24]. There are two techniques to compensate for this limitation and improve the lifetime of STT-RAM: *(a)* wear-leveling and *(b)* reducing the number of writes.

(a) Wear-Leveling aims to distribute writes evenly across the entire STT-RAM cache. This positively impacts the endurance of the cache, since writes to a cache typically cluster to a specific region during the execution of an application. Chen et al. [4] propose one variation of wear-leveling. They remap all cache set indices after a predefined amount of time to distribute the cache writes.

(b) Reducing the number of writes to a STT-RAM cache can be accomplished by performing read-before-write operations on bit level and cache-access level. One such implementation is the Early Write Termination (EWT) proposed by Zhou et al. [35], which is a variation of bit-level read-before-write operations. Their write operation incorporates sensing the old bit-value, comparing it to the new one, and issuing a signal to stop the write operation, if the two values are the same. This method does not induce additional overhead, as voltage used for reading has to be applied for a write operation as well.

Implementations With current restrictions regarding the high write latency and energy as well as the low write endurance of non-volatile STT-RAM, using only this technology to implement non-volatile hardware caches does not seem feasible. Instead, if

[1]The feature size F of a transistor is defined as the minimum length between its drain and source.

		SRAM	Non-Relaxed	Relaxed Opt. 1	Relaxed Opt. 2
Latency [ns]	Read	1.113	0.802	0.792	0.951
	Write	1.082	10.378	5.370	1.500
Energy [nJ]	Read	0.075	0.083	0.032	0.043
	Write	0.059	0.958	0.466	0.198
Leakage Power [mW]		57.7	1.82	1.78	2.41
Cell Size [F^2]		125	23	22	40.3
Retention Time (at 350 K)		–	4.27 yr	3.24 s	26.5 µs

Table 2: Results of the evaluation of STT-RAM caches with different levels of retention relaxation from Sun et al. [24]. The 32 KB caches were configured as L1 caches. The relaxed implementations were relaxed with the same technique, only differing regarding the retention time.

the non-volatility of STT-RAM is required, hybrid approaches are advisable with volatile caches at the higher levels.

Li et al. [11] propose one such hybrid implementation, a combination of STT-RAM and SRAM. They utilize compiler analyses and knowledge provided by the operating system (OS) to swap write-intensive data from non-volatile to volatile memory, thus prolonging the lifetime of the STT-RAM cells.

A different approach, which focuses on the advantageous energy and capacity features of STT-RAM, is the approach by Sun et al. [24]. They propose a cache hierarchy of STT-RAM caches with different retention times, to reduce the performance loss caused by the high write latencies of regular STT-RAM in exchange for the non-volatility of the higher-level caches.

3.3 Non-Volatile Main Memory

Both the low power consumption and the high density of byte-addressable NVM technologies make them an attractive alternative to DRAM. For the main memory, most researchers suggest PCRAM as the NVM technology of choice. This is due to its higher density, which compensates for the higher latencies of PCRAM. As the need for systems with larger memories increases, the capacity of the main memory has to grow as well. With DRAM, this can lead to the power consumption of the main memory dominating the system's overall energy consumption. For example, Lefurgy et al. [10] reported that 41% of a mid-range IBM servermachine's energy consumption could be attributed to its main memory subsystem. Additionally, a non-volatile main memory allows for new operating system designs, as pointed out by Bailey et al. [1].

Challenges This paragraph focuses on how to ensure consistency and how to tackle low write endurance. Limitations due to high write latency and energy as well as low write endurance of NVM technologies have already been discussed in Section 3.2, though some proposed countermeasures are not applicable for non-volatile main memory implementations.

(i) Consistency: With the use of NVMs, questions concerning the consistency of data in the system arise [20]. These lead system developers to take properties into account the database community has researched for years and which are often referred to as the *ACID* properties: atomicity, consistency, isolation, and durability [5]. Consequently, NVM main memory implementations rely mostly on transactions to ensure consistency. For example, Volos et al. [25] utilize durable memory transactions and manage transactions by write-ahead logging (WAL) to allow in-place updates.

As argued by Zhao et al. [33], transactions impose a significant overhead, that negatively impacts the performance. To perform transactions, mostly either WAL or copy-on-write (COW) is performed, both of which result in multiple write operations. To avoid this overhead, they propose utilizing the memory hierarchy to avoid additional logs and copies. A similar approach is also brought forward in the work of Moraru et al. [19]. To realize multiversioning, Zhao et al. make use of a non-volatile LLC. With this design, the updated data in a dirty cache line constitutes one version, while the data residing in the main memory is a second version.

(ii) Write Endurance: Wang and Wong [26] propose OS assisted wear leveling. They utilize the knowledge of the OS about files to predict the frequency of updates in the file, among other things, to adapt and improve wear-leveling algorithms. Though their work is based on NAND Flash, the applicability on PCRAM should be researched.

Another approach to treating low write endurance is to introduce fault-tolerance mechanisms. For example, Yoon et al. [30] use still functional cells of failing blocks for storing the address of the remapped block. Alternatively, blocks that have exceeded their lifetime can still be of use by marking them as *read-only* [3].

Implementations & Alternatives Systems designed with only non-volatile main memory are hard to come by. This is mostly due to the relatively low write endurance, that has to be compensated. More common are hybrid implementations, for example, *Mnemosyne* developed by Volos et al. [25]. *Mnemosyne*'s DRAM main memory is supported by PCRAM to utilize its non-volatility, fast access time, and to provide user-mode access to persistent memory. The compiler ensures that no volatile addresses are assigned to non-volatile pointers, which, in case of power failures, would then point to lost data.

Moraru et al. [19] present a different hybrid approach. They propose a main memory consisting of DRAM as well as NVRAM and utilize OS virtual memory mechanisms to map non-volatile memory to the address space of a process. Additionally, they offer *NVMalloc* as a form of wear-leveling. The *NVMalloc* implementation does not organize its memory as a LIFO list but instead adds freed blocks to a FIFO queue. The head of the queue is removed and marked as available for reallocation if it has been in the queue for a

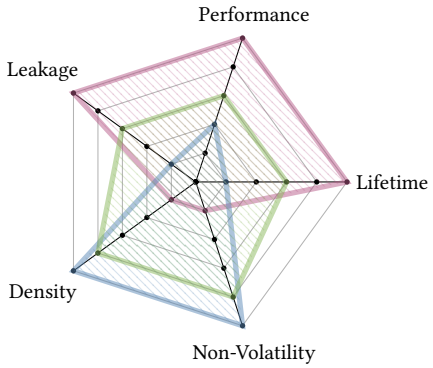

Figure 3: Visualization of the expected characteristics of a volatile, a non-volatile, and a mixed system.

predetermined amount of time. The free lists and other meta-data are managed in DRAM, as they have to be written frequently.

3.4 Putting It All Together

The previous sections discussed the applicability of NVM technologies to different memory components, but the question of whether an all non-volatile embedded system is feasible is still unanswered.

Are We There Yet? Implementing an entirely non-volatile system requires all volatile processor memory components, as well as caches and the main memory to be replaced. With PCRAM and STT-RAM two NVM technologies are available that come close to the read latencies of DRAM and SRAM respectively. Additionally, they offer higher density and less power consumption, due to their almost non-existent leakage power, as detailed in Table 1 and Table 2. However, their write endurance and write latency prove problematic, especially if deployed as high-level caches or processor registers. While in the area of embedded systems, a slow working system might not be as catastrophic as in the supercomputing area, the latencies introduced are still hard to handle and might render the entire system infeasible. The performance is even further limited by the introduction of transactions to maintain data consistency.

Additionally, the issue of the low write endurance has to be addressed. STT-RAM, which is the most suitable candidate to replace SRAM, would, on average last only 1.3 years if it were deployed as a first-level cache without any optimizations [27]. Even software- and hardware-based wear-leveling combined with reducing the total number of writes does most likely not enhance the lifetime to an acceptable level.

Best of Both Worlds With the restrictions imposed by the current NVM technologies an entirely non-volatile system does not seem feasible. It does allow the design of a smaller system with significantly less leakage power and improved overall power consumption, though the higher energy required for writing does lessen the effect [23]. However, these advantages are not able to balance out the much lower performance and reduced lifetime.

Therefore the question arises whether a hybrid system could combine the best features of a volatile and a non-volatile design: high performance, low leakage power, high density, long lifetime, and non-volatility on some levels. After examining the different features, a possible design can be derived: A volatile processor equipped with a non-volatile memory block to allow back-ups, volatile high-level caches, a non-volatile last level cache (LLC), and a non-volatile main memory with a volatile buffer. Figure 3 shows the properties of a volatile system (red), a non-volatile design (blue), and of the suggested hybrid system (green).

The performance of the volatile processor is only limited by the overhead of back-up and restore operations. Nonetheless, the ability to pick up execution at the state before a power outage makes up for the additional operations [12, 15]. With volatile high-level caches, the performance improvement of caches can be maintained, and a non-volatile LLC allows taking advantage of a non-volatile main memory and introducing transactions without additional logs [33]. The volatile caches could be implemented as a STT-RAM-SRAM hybrid [11], with the help of retention relaxation [23, 24] so that the system profits of the low leakage power of STT-RAM and the system's power demand is significantly reduced. The PCRAM main memory could make use of a DRAM buffer for write-intensive data, to enhance its lifetime. The entire design relies on the support of the system software. Endurance-aware memory-allocation, like *NVMalloc* [19], prolongs the lifetime, and knowledge of the OS can be utilized to transfer write-intensive files to the volatile buffer and perform more effective wear-leveling [26]. Without these software techniques, a feasible system would not be possible.

4 CONCLUSION

This work introduced current research into the use of byte-addressable non-volatile memory technologies and discussed their potential to replace the volatile SRAM and DRAM. While the non-volatile STT-RAM and PCRAM offer high density as well as minimal leakage power, their limited write endurance and high write latency prevent them from taking the place of SRAM and DRAM respectively. A hybrid design consisting of volatile and non-volatile memory components can utilize features of both technologies to acquire non-volatility and low power consumption without significant performance losses. Enabling the processor to perform back-ups improves its forward progress in case of power outages. A combination of modified STT-RAM and SRAM allows for less power-intensive caches, improving the overall power consumption of the system. A non-volatile LLC combined with a non-volatile main memory, that is supported by a volatile buffer, offer non-volatility to keep the system state across power-downs, without too much performance loss. To improve the lifetime of a hybrid system, the system software is essential for providing wear-leveling and consistency algorithms.

REFERENCES

[1] Katelin Bailey, Luis Ceze, Steven D Gribble, and Henry M Levy. 2011. Operating System Implications of Fast, Cheap, Non-Volatile Memory. In *Proceedings of the 13th USENIX Workshop on Hot Topics in Operating Systems (HotOS '11)*, Vol. 13. 2–2.

[2] Rajendra Bishnoi, Fabian Oboril, Mojtaba Ebrahimi, and Mehdi B Tahoori. 2014. Avoiding Unnecessary Write Operations in STT-MRAM for Low Power Implementation. In *Proceedings of the 15th International Symposium on Quality Electronic Design (ISQED '14)*. 548–553.

[3] Yu Cai, Onur Mutlu, Erich F Haratsch, and Ken Mai. 2013. Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation. In *Proceedings of the 31st International Conference on Computer Design (ICCD '13)*.

123–130.
[4] Yiran Chen, Weng-Fai Wong, Hai Li, Cheng-Kok Koh, Yaojun Zhang, and Wujie Wen. 2013. On-chip Caches Built on Multilevel Spin-transfer Torque RAM Cells and Its Optimizations. *J. Emerg. Technol. Comput. Syst.* 9 (2013), 16:1–16:22.
[5] Theo Haerder and Andreas Reuter. 1983. Principles of Transaction-Oriented Database Recovery. *ACM Computing Surveys (CSUR)* 15, 4 (1983), 287–317.
[6] Yung-Chiun Her, Wei-Ting Tu, and Ming-Hsin Tsai. 2012. Phase transformation and crystallization kinetics of a-Ge/Cu bilayer for blue-ray recording under thermal annealing and pulsed laser irradiation. *Journal of Applied Physics* 111 (2012), 043503.
[7] Yiming Huai et al. 2008. Spin-Transfer Torque MRAM (STT-MRAM): Challenges and Prospects. *AAPPS Bulletin* 18, 6 (2008), 33–40.
[8] Navid Khoshavi and Ronald F Demara. 2018. Read-Tuned STT-RAM and eDRAM Cache Hierarchies for Throughput and Energy Optimization. *IEEE Access* 6 (2018), 14576–14590.
[9] Alexander V Kolobov, Paul Fons, Junji Tominaga, and T Uruga. 2006. Why DVDs work the way they do: The nanometer-scale mechanism of phase change in Ge–Sb–Te alloys. *Journal of Non-Crystalline Solids* 352 (2006), 1612–1615.
[10] Charles Lefurgy, Karthick Rajamani, Freeman Rawson, Wes Felter, Michael Kistler, and Tom W Keller. 2003. Energy Management for Commercial Servers. *Computer* 36, 12 (2003), 39–48.
[11] Yong Li, Yiran Chen, and Alex K. Jones. 2012. A Software Approach for Combating Asymmetries of Non-volatile Memories. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '12)*. 191–196.
[12] Yongpan Liu, Zewei Li, Hehe Li, Yiqun Wang, Xueqing Li, Kaisheng Ma, Shuangchen Li, Meng-Fan Chang, Sampson John, Yuan Xie, et al. 2015. Ambient Energy Harvesting Nonvolatile Processors: From Circuit to System. In *Proceedings of the 52nd Annual Design Automation Conference (DAC '15)*. 150.
[13] Daniel Lohmann, Wolfgang Schroder-Preikschat, and Olaf Spinczyk. 2005. Functional and Non-Functional Properties in a Family of Embedded Operating Systems. In *Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS '05)*. 413–420.
[14] Kaisheng Ma, Xueqing Li, Shuangchen Li, Yongpan Liu, John Jack Sampson, Yuan Xie, and Vijaykrishnan Narayanan. 2015. Nonvolatile Processor Architecture Exploration for Energy-Harvesting Applications. *IEEE Micro* 35, 5 (2015), 32–40.
[15] Kaisheng Ma, Yang Zheng, Shuangchen Li, Karthik Swaminathan, Xueqing Li, Yongpan Liu, Jack Sampson, Yuan Xie, and Vijaykrishnan Narayanan. 2015. Architecture Exploration for Ambient Energy Harvesting Nonvolatile Processors. In *IEEE 21st International Symposium on High Performance Computer Architecture (HPCA '15)*. 526–537.
[16] Sparsh Mittal. 2014. A Survey of Architectural Techniques for Improving Cache Power Efficiency. *Sustainable Computing: Informatics and Systems* 4 (2014), 33–43.
[17] Sparsh Mittal and Jeffrey S Vetter. 2015. A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems. *IEEE Transactions on Parallel and Distributed Systems* 27, 5 (2015), 1537–1550.
[18] Sparsh Mittal, Jeffrey S Vetter, and Dong Li. 2014. A Survey Of Architectural Approaches for Managing Embedded DRAM and Non-Volatile On-Chip Caches. *IEEE Transactions on Parallel and Distributed Systems* 26, 6 (2014), 1524–1537.
[19] Iulian Moraru, David G. Andersen, Michael Kaminsky, Niraj Tolia, Parthasarathy Ranganathan, and Nathan Binkert. 2013. Consistent, Durable, and Safe Memory Management for Byte-addressable Non Volatile Main Memory. In *Proceedings of the 1st ACM SIGOPS Conference on Timely Results in Operating Systems (TRIOS '13)*. Article 1, 1:1–1:17 pages.
[20] Benjamin Ransford and Brandon Lucia. 2014. Nonvolatile Memory is a Broken Time Machine. In *Proceedings of the Workshop on Memory Systems Performance and Correctness (MSPC '14)*. 5.
[21] Nitin Rathi, Swaroop Ghosh, Anirudh Iyengar, and Helia Naeimi. 2016. Data Privacy in Non-Volatile Cache: Challenges, Attack Models and Solutions. In *Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC '16)*. 348–353.
[22] Rodrigue Rizk, Dominick Rizk, Ashok Kumar, and Magdy Bayoumi. 2019. Demystifying Emerging Nonvolatile Memory Technologies: Understanding Advantages, Challenges, Trends, and Novel Applications. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '19)*. 1–5.
[23] Clinton W Smullen, Vidyabhushan Mohan, Anurag Nigam, Sudhanva Gurumurthi, and Mircea R Stan. 2011. Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches. In *Proceedings of the 17th IEEE International Symposium on High Performance Computer Architecture (HPCA '11)*. 50–61.
[24] Zhenyu Sun, Xiuyuan Bi, Hai (Helen) Li, Weng-Fai Wong, Zhong-Liang Ong, Xiaochun Zhu, and Wenqing Wu. 2011. Multi Retention Level STT-RAM Cache Designs with a Dynamic Refresh Scheme. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '11)*. 329–338.
[25] Haris Volos, Andres Jaan Tack, and Michael M. Swift. 2011. Mnemosyne: Lightweight Persistent Memory. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. 91–104.
[26] Chundong Wang and Weng-Fai Wong. 2013. SAW: System-Assisted Wear Leveling on the Write Endurance of NAND Flash Devices. In *Proceedings of the 50th Annual Design Automation Conference (DAC '13)*. 164.
[27] Jianxing Wang, Yenni Tim, Weng-Fai Wong, Zhong-Liang Ong, Zhenyu Sun, and Hai Helen Li. 2014. A Coherent Hybrid SRAM and STT-RAM L1 Cache Architecture for Shared Memory Multicores. In *Proceedings of the 19th Asia and South Pacific Design Automation Conference (ASP-DAC '14)*. 610–615.
[28] L Wang, C-H Yang, and J Wen. 2015. Physical Principles and Current Status of Emerging Non-Volatile Solid State Memories. *Electronic Materials Letters* 11, 4 (2015), 505–543.
[29] Mimi Xie, Mengying Zhao, Chen Pan, Jingtong Hu, Yongpan Liu, and Chun Jason Xue. 2015. Fixing the Broken Time Machine: Consistency-Aware Checkpointing for Energy Harvesting Powered Non-Volatile Processor. In *Proceedings of the 52Nd Annual Design Automation Conference (DAC '15)*. 184:1–184:6.
[30] Doe Hyun Yoon, Naveen Muralimanohar, Jichuan Chang, Parthasarathy Ranganathan, Norman P Jouppi, and Mattan Erez. 2011. FREE-p: Protecting Non-Volatile Memory against both Hard and Soft Errors. In *Proceedings of the 17th International Symposium on High Performance Computer Architecture (HPCA '11)*. 466–477.
[31] Daming Zhang, Shuangchen Li, Ang Li, Yongpan Liu, X Sharon Hu, and Huazhong Yang. 2014. Intra-task Scheduling for Storage-less and Converter-less Solar-Powered Nonvolatile Sensor Nodes. In *Proceedings of the IEEE 32nd International Conference on Computer Design (ICCD '14)*. 348–354.
[32] Daming Zhang, Yongpan Liu, Xiao Sheng, Jinyang Li, Tongda Wu, Chun Jason Xue, and Huazhong Yang. 2015. Deadline-aware Task Scheduling for Solar-powered Nonvolatile Sensor Nodes with Global Energy Migration. In *Proceedings of the 52nd Annual Design Automation Conference (DAC '16)*. 126.
[33] Jishen Zhao, Sheng Li, Doe Hyun Yoon, Yuan Xie, and Norman P Jouppi. 2013. Kiln: Closing the Performance Gap Between Systems With and Without Persistence Support. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '13)*. 421–432.
[34] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. 2009. A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA '09)*. 14–23.
[35] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. 2009. Energy Reduction for STT-RAM Using Early Write Termination. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD '09)*. 264–268.