AdAM: <u>Ad</u>aptive <u>A</u>pproximation <u>M</u>anagement for the Non-Volatile Memory Hierarchies

Mohammad Taghi Teimoori[1,2], Muhammad Abdullah Hanif[2], Alireza Ejlali[1], Muhammad Shafique[2]
[1]Computer Engineering Department, Sharif University of Technology, Iran
[2]Institute of Computer Engineering, Vienna University of Technology (TU Wien), Austria
teimoori@ce.sharif.edu, ejlali@sharif.edu, {muhammad.hanif, muhammad.shafique}@tuwien.ac.at

***Abstract*—Existing memory approximation techniques focus on employing approximations at an individual level of the memory hierarchy (e.g., cache, scratchpad, or main memory). However, to exploit the full potential of approximations, there is a need to manage different approximation knobs across the complete memory hierarchy. Towards this, we model a system including STT-RAM scratchpad and PCM main memory with different approximation knobs (e.g., read/write pulse magnitude/duration) in order to synergistically trade data accuracy for both STT-RAM access delay and PCM lifetime by means of an integer linear programming (ILP) problem at design-time. Furthermore, a run-time algorithm is proposed to adaptively tune the approximation knobs of both STT-RAM and PCM to obtain high energy savings while keeping error-per-second within acceptable ranges across the complete memory hierarchy. We evaluated our proposed technique (i.e., AdAM) in a baseline system consisting of 1-2MB STT-RAM scratchpad and 0.5-1GB PCM main memory. The experimental results demonstrate that AdAM improves the execution time and the lifetime of memory by up to 38.7% and 1.6X, respectively.**

I. INTRODUCTION

Approximate computing (AC) is a promising low-power and high performance design paradigm for embedded computing systems [1]. Most of the previous studies in AC have focused on optimizing the performance, area, and energy efficiency of arithmetic and logic units [2]. However, memories being the major bottlenecks for performance and power [3], have not received much attention in AC. Although Non-Volatile Memories (NVMs) including Phase Change Memory (PCM) and Spin Transfer Torque (STT) memory may solve the system's power consumption issue [4], STT technology as the alternative of SRAMs for on-chip caches/Scratch-Pad memories (SPMs), suffers from performance issues [5]. Furthermore, PCM technology as the alternative for the conventional off-chip memories confronts lifetime challenges [6] (i.e., lasting only for about 10^7-10^8 writes per cell). Thus, the performance and lifetime challenges should be addressed to make NVMs practical alternatives for traditional memories.

Previous studies have proposed different techniques to improve the energy consumption, storage density, reliability, and lifetime of PCM [6]- [7] and energy consumption, density, and performance of STT-RAM [4]. However, these works do not take advantage of the inherent error-resilience of applications. A few approximation techniques have also been proposed to improve the energy consumption, lifetime, density, and reliability of PCM [8]-[9] and performance, power efficiency and density of STT-RAM [10]-[11]. However, all the previous works on PCM/STT-RAM approximation have focused on approximating a single level in the complete memory hierarchy (MH), which usually consists of multiple levels including on-chip memory (e.g., cache, SPM) and off-chip memory (e.g., main memory). By considering approximations in multiple levels of the MH, there may be more potential for investigating more efficient low power and high performance optimizations.

Motivational Case Study: *1)* Considering Fig. 1 (a) and (b), STT-RAM's delay and PCM's lifetime analyses of Smoothing [12] and Epic [13] benchmarks, which are two image utilities, demonstrate that in general, applications experience different workloads on different levels of the MH. In this example, Smoothing is PCM-friendly because with its write traffic workload PCM's lifetime until wearing out is fairly long (about 40 years) as most of its Execution Time (ET) (more than 60%) is spent to access STT-RAM SPM. On the other hand, Epic indicates heavy write traffic workload on PCM main memory which results in its early wearout (i.e., 0.8 years) while the access delay of STT-RAM SPM is not an issue for this application (about 20%). *Thus, the Smoothing application requires effective STT-RAM approximations to improve performance (as PCM's lifetime is already high) while Epic application requires effective PCM approximations to enhance PCM's lifetime (as STT-RAM's delay is already moderate).*

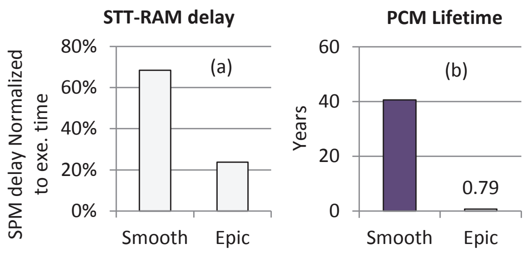

Fig. 1: STT-RAM's delay and PCM's lifetime for Smoothing [12] and Epic [13] applications in a system with 1MB STT-RAM SPM, 0.5GB PCM main memory and a 4GHz processing core.

2) The error models of each PCM [8] and STT-RAM [17] are different. For instance, the errors in PCM are time-dependent [21] and in systems with multi-tasking policies the Response Times (RTs) of applications fluctuate [22] resulting in PCM error fluctuation. *Thus, to keep the total Error-Per-Second (EPS) of an application within an acceptable range, it*

TABLE I: NVM Approximation Techniques

Mem. Tech.	Position in MH	Related Work	Technique	Short Description	Motivation (Improvement)	Role in MH	MH Aware
PCM	Off-Chip	[14][9]	Error-resilient Architectures	Before writing/reading to/from memory, data is coded/decoded or compressed/decompressed to reduce error-sensitive bit-patterns	Error Rate	Main Memory	No
		[8]	Approximate Main/Storage	*i)* the P&V iteration number is reduced *ii)* written data is still valid after retention time, *iii)* cells are used after wearing out	Energy, Lifetime	Main Memory/ Storage	No
		[15]	Error Scrubbing	The whole PCM storage is scrubbed once every scrubbing period and an ECC circuit is powered on to correct possible erroneous lines	Energy	Storage	No
		[16]	Selective Approximation	A circuit compares new data with old ones before writing to a PCM cell and the writing process is skipped if the difference is lower than a threshold	Energy, Lifetime	Main Memory	No
STT	On-Chip	[10]	Approximate Cache	The ECC circuits' overhead is eliminated for the relaxed data blocks of MLC STT-RAM while the safe area includes ECC circuits	Energy, Performance	Cache	No
		[17][18]	Fine Grain	Duration and/or intensity of write/read pulse is reduced to achieve efficient performance, power	Energy, Performance	SPM/ Cache	No
		[11][19]	Approx. Array	Exploitation of Different Retention Times	Energy	Cache	No
		[20]	Fine Grain	Access transistor width is aggressively scaled to reduce leakage power	Energy	SPM	No

is advantageous to monitor errors in both PCM and STT-RAM and tune the approximation knobs adaptively.

In this paper, to the best of our knowledge, we propose the first MH-aware NVM approximation utilization technique (AdAM) to trade accuracy for both PCM's lifetime and STT-RAM's delay. We consider a system including STT-RAM SPM and PCM main memory, which are both approximable. In order to improve system lifetime and performance at design time, we formulate an integer linear programming (ILP) optimization problem which considers the application workloads on the memory hierarchy levels. Furthermore, a run-time algorithm is proposed to monitor the EPS across the memory levels and to adaptively tune the approximation knobs in order to keep application's EPS within the required range. The system-wide approximation concept can also be extended to full-system approximations when applied to accelerator based architectures [23].

The main contributions of this paper are as follows:

1) We characterize application's workload impacts on the on-chip STT-RAM's delay and off-chip PCM's lifetime.
2) We consider the response time of applications in multi-tasking systems to estimate PCM's EPS.
3) We propose AdAM which consists of solving a design-time ILP optimization problem and a run-time algorithm to trade-off power/performance/accuracy.

To evaluate the proposed method in a system including 1-2MB STT-RAM SPM and 0.5-1GB PCM main memory, we used a system simulator [24], several image&voice benchmark applications from Mibench [12] and Mediabench [13] packages and the NVSim NVM simulator [25]. The experimental results show up to 38.7% (22.1% on average) improvement of the application execution time and up to 1.6X (1.2X on average) improvement of the memory lifetime.

Paper organization: Section II covers the background of STT and PCM technologies and their approximation techniques. Section III represents our proposed method. Our evaluation methodology and experimental results are represented in Section IV. Finally, Section V concludes the paper.

II. A Survey of NVM Approximation Techniques

An overview of approximation techniques for PCM and STT-RAM is presented in Table I. The table classifies the NVM approximation techniques into on-chip or off-chip techniques based on their target MH level. The mentioned on-chip works have investigated the potential of STT-RAM as approximate cache or SPM. Furthermore, the mentioned off-chip works have studied approximate PCM as main memory or permanent storage. However, computing systems usually include both on-chip and off-chip memory components to utilize data locality, thereby improving performance and power. Thus, it is important not to limit approximations to a single memory level. However, as the table reveals, there is a great potential for work on inter-memory-level approximations through the approximate memory hierarchies.

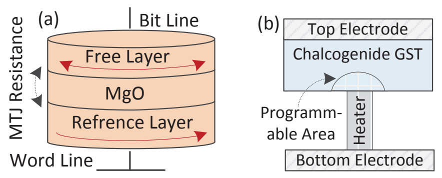

Fig. 2: Conventional STT-RAM [19] and PCM cells

III. Background

STT-RAM and PCM cells are depicted in Figures 2 (a) and (b), respectively. A brief background on these memory technologies is presented in this section.

A. Spin Transfer Torque Memory

STT-RAM read and write operations are carried out by applying currents smaller and larger than the critical switching current (I_{co}) across the MTJ [26], respectively. Since read accesses to STT-RAM SPM are on the critical path of the execution, the Read Disturb (RD) approximation technique has been proposed to reduce its delay [17]. The RD technique increases the read pulse current magnitude to enable a reduction in the STT-RAM's read pulse duration. However, as the current approaches I_{co}, RD-induced bit-flip probability ($P_{disturb}$) increases following Equation 1 [27]

$$P_{disturb} = 1 - exp(\frac{-t_{read}}{\tau_0\, exp(\Delta(1 - \frac{I_{read}}{I_{co}}))}). \qquad (1)$$

where Δ is the thermal stability of an STT-RAM cell and τ_0 is 1ns. Based on this expression, by increasing the read current (i.e., I_{read}), the STT-RAM read pulse duration (i.e., t_{read}) can be decreased to improve the read performance at the cost of increased $P_{disturb}$.

B. Phase Change Memory

By applying electrical current, PCM element's state switches between amorphous state (high resistance or RESET) and crystalline state (low resistance or SET) [6]. In MLC-PCM the wide range of PCM's resistance spectrum (i.e., 1k to 1M) is partitioned into 2^n levels, by multiple SET/RESET iterations [28], where n is the number of bits stored in a cell.

Since PCM cell write endurance is limited, the lifetime of a PCM system depends on write traffic of applications. PCM's lifetime can be calculated as [6]

$$System\ Lifetime = \frac{W_{Max} \times S}{B}\ Seconds \qquad (2)$$

where, W_{Max}, S and B are PCM cell's endurance (i.e., 10^7), PCM capacity in GB and the write traffic in GBps.

In the PCM approximation technique [8], the SET/RESET iteration number is reduced which may result in overlapping adjacent logic states. Furthermore, the drift phenomenon which continuously increases the amount of the resistance, worsens the error rate in the approximate PCM. Although applications' ET are usually not long enough for large drifts, it may become problematic when the RT of an application in a multi-tasking system becomes longer than its ET. Based on the MLC-PCM's soft error probability which is presented in [21][8], the error rate ranges from 10^{-7} to 8% for the most conservative and the most aggressive approximations, respectively.

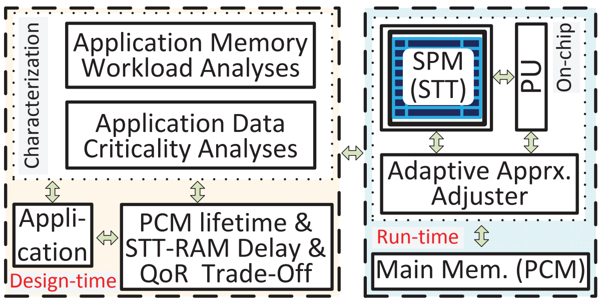
Characterization
Application Memory Workload Analyses
Application Data Criticality Analyses
Appli-cation
PCM lifetime & STT-RAM Delay & QoR Trade-Off
Design-time
SPM (STT)
PU
On-chip
Adaptive Apprx. Adjuster
Run-time
Main Mem. (PCM)

Fig. 3: System overview and AdAM's methodology

IV. Proposed Technique

Our proposed technique, AdAM, consists of solving a design-time ILP problem and a run-time algorithm. The ILP utilizes application's workloads on SPM and main memory to guarantee that the PCM's lifetime and the STT-RAM's access delay are acceptable while application's EPS across the entire MH is minimized. Then at run-time, our technique predicts the response time of an application and evaluates PCM's EPS based on the prediction, and tunes approximation knobs of PCM and STT-RAM to control overall EPS across the memory hierarchy. An overview of the proposed method is illustrated in Fig. 3.

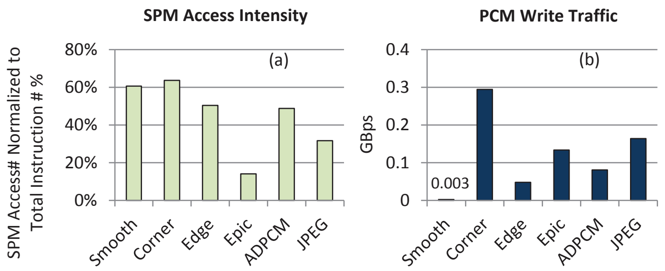

Fig. 4: Application workload on the MH levels. Most of the applications are memory bound (i.e., ≥30% memory instructions).

A. Workload Characterization

The proposed technique selects heavier STT-RAM/PCM approximations for applications with a high number of accesses to STT-RAM/PCM. Based on this, in this step of AdAM, we characterize workloads of applications on STT-RAM (i.e., delay of accesses to STT-RAM) and PCM (i.e., write traffic to PCM). Figures 4 (a) and (b) compare six applications with their on-chip and off-chip workloads, respectively. This information is exploited to estimate PCM's lifetime and STT-RAM's delay in every approximation level.

B. Memory Hierarchy Approximation Trade-Off

AdAM utilizes the asymmetric workloads of applications on the on-chip and the off-chip memories to enable an application-aware trade-off across the entire MH for improving memory lifetime and application execution time. The following ILP optimization problem formulates the memory hierarchy performance-lifetime-accuracy trade-off problem. Throughout the ILP, it is assumed that n tasks $t_i \in \{t_1, t_2 \ldots t_n\}$ execute in the system.

1) PCM's lifetime: By selecting the PCM approximation level for task t_i, we indirectly decide the write traffic to PCM because the heavier the approximation, the lower the write iteration number [8]. Thus, the PCM's lifetime is guaranteed to be longer than the required lifetime LT_i by Equation 3:

$$\frac{W_{Max}.S}{\sum_{j=1}^{PL} B_{i,j} \times XP_{i,j}}\ Seconds \geq LT_i \qquad (3)$$

where, $B_{i,j}$ is the write traffic of task t_i at the $j-th$ PCM approximation level, PL is the number of PCM approximation levels, LT_i is the required lifetime, and $XP_{i,j}$ is a binary variable which is defined as

$$XP_{i,j} = \begin{cases} 1 & \text{if } j_{th} \text{ PCM approx. level is assigned to } t_i \\ 0 & \text{otherwise.} \end{cases} \qquad (4)$$

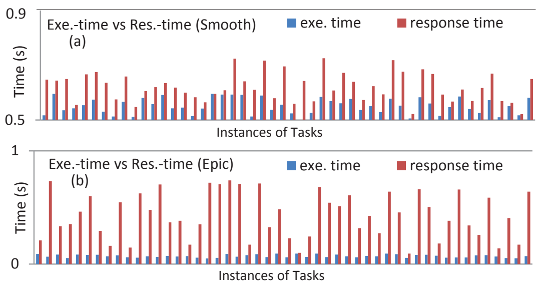

Fig. 5: Response Time Vs. Execution Time

2) STT-RAM delay: By deciding the STT-RAM approximation level for the task, we also determine its access delay to STT-RAM because when using RD technique, the higher the approximation level, the lower the STT-RAM's access delay [17]. Thus, for every task t_i Equation 5 guarantees the performance requirement (i.e., SD_i), where the binary variable $XS_{i,k}$ is 1 if $k-th$ STT-RAM approximation level is assigned to the task t_i, SL is the number of STT-RAM approximation levels, and SD_i is the required STT-RAM delay of the application.

$$\sum_{k=1}^{SL} STT\_delay_{i,k} \times XS_{i,k} \leq SD_i \tag{5}$$

3) Error per second: Equations 3 and 5 may try to select the highest approximation levels to achieve the longest PCM lifetime and the least STT-RAM's delay if the amount of the approximations is not controlled through the ILP. However, not controlling application's EPS amount can result in significant Quality-of-Result (QoR) degradations. We consider the approximations amount by minimizing application's EPS. Based on this, the objective of the ILP is defined as

$$\sum_{j=1}^{PL} EPS_{PCM}^{i,j} \times XP_{i,j} + \sum_{k=1}^{SL} EPS_{STT}^{i,k} \times XS_{i,k} \tag{6}$$

where $EPS_{PCM}^{i,j}$ and $EPS_{STT}^{i,k}$ are estimations of the error-per-second of PCM and STT-RAM, respectively. Thus, PCM's write iteration number, which determines PCM's approximation level, and STT-RAM's access pulse duration, which determines STT-RAM approximation level, are two knobs for trading accuracy for the PCM's lifetime and the STT-RAM delay.

C. Run-time Approximation Adjustment

In systems with multi-tasking policies, usually a task's RT is longer than its ET due to the system level task management decisions [22]. Fig. 5 shows the difference between the response time and the execution time for Smoothing and Epic applications in a system with six tasks. The RT of Epic considerably fluctuates compared to Smoothing because Epic's ET is relatively low and the order of its execution in the multi-tasking system affects its RT. The RT fluctuations influence the drift-based error probability which has a direct relation with time [21]. Based on this analysis, this step of our proposed method adaptively adjusts PCM and STT-RAM approximation knobs based on RT prediction [22] and monitoring in order to keep application's overall EPS around the amount that ILP decided at design-time.

Algorithm 1: Adaptive adjustment of PCM and STT-RAM approximation knobs

```
Input: ILP's decisions on PCM and STT-RAM approximation levels
Output: PCM and STT-RAM approximation level at run-time
    On task release:
 1: Predict the $RT$ for the task;
    $RT_m = \alpha \times RT_{m-1} + \beta \times RT^*$
 2: if (PRT! =ET) then
 3:     calculate $EPS_{PCM}^{OPL}(PRT)$;
 4:     find $p \in \{1 \dots PL\}$ such that
        $EPS_{PCM}^{p}(PRT) == EPS_{PCM}^{OPL}(ET)$;
 5: end if
    On task execution:
 6: Record task ART;
 7: if (PRT! =ET)&(ART! =PRT) then
 8:     Calculate $EPS_{PCM}^{p}(ART)$;
 9:     decide $s \in \{1 \dots SL\}$ such that
        $EPS_{STT}^{s} + EPS_{PCM}^{p}(ART) == EPS_{STT}^{OSL} + EPS_{PCM}^{p}(PRT)$;
10: end if
```

Our adaptive approximation adjustment algorithm for both PCM and STT-RAM is presented in Alg. 1. On task release, the algorithm is invoked to predict the RT (i.e., PRT) based on the RT prediction method [22] (Line 1), where RT_{m-1} and R^* denote the previous RT of task and the last predicted RT, respectively. If PRT is not equal to ET (Line 2), PCM's EPS will be different from what is supposed to happen during ILP. Thus, the algorithm finds PCM approximation level p which makes $EPS_{PCM}^{p}(PRT)$ equal to $EPS_{PCM}^{OPL}(ET)$ (Lines 3-4). This adaptive decision tries to control PCM's EPS near the designed amount (OPL) against the possible fluctuations.

RT prediction methods are not always precise (20% to 30% prediction error) [22]. Therefore, when a tasks starts executing, Alg. 1 monitors whether the actual RT (i.e., ART) equals the PRT or not. If the ART is higher/lower than PRT (Lines 6-7), PCM's EPS amount (i.e., $EPS_{PCM}^{p}(ART)$) has been more/less than the predicted EPS amount (i.e., $EPS_{PCM}^{p}(PRT)$). Thus, the algorithm finds STT-RAM approximation level s which brings the overall EPS (i.e., $EPS_{STT}^{s} + EPS_{PCM}^{p}(ART)$) near to the ILP-decided EPS amount (i.e., $EPS_{STT}^{OSL} + EPS_{PCM}^{p}(PRT)$). This adaptive decision tries to control the overall EPS against the RT miss-predictions.

V. Evaluation and Experimental Results

To evaluate relative efficiency of AdAM in terms of ET, PCM's lifetime and EPS we used six image and voice processing benchmarks from Mibench [12] and Mediabench [13] packages. We compared our results with previous works. i.e., the PCM approximation technique [8] and the STT-RAM

TABLE II: Baseline system configuration and parameters

	Memory Parameters		Execution	
	SPM	Main Mem.		
Technology	STT-RAM	PCM	Operating Frequency	4GHz
Size	1MB	1GB		
Read energy	659.2pJ	877pJ	# of Cores	1
Read delay	2.5ns	130.5ns	Simulator	Simple-scalar
Write delay	11ns	93-482ns		
Write Energy	93.5pJ	6.7-26.8nJ	ISA	Pisa

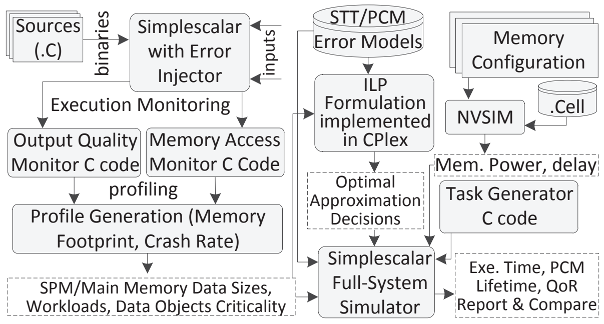
Sources (.C)
binaries
Simplescalar with Error Injector
inputs
STT/PCM Error Models
Memory Configuration
Execution Monitoring
ILP Formulation implemented in CPlex
NVSIM
.Cell
Output Quality Monitor C code
Memory Access Monitor C Code
profiling
Mem. Power, delay
Optimal Approximation Decisions
Task Generator C code
Profile Generation (Memory Footprint, Crash Rate)
Simplescalar Full-System Simulator
Exe. Time, PCM Lifetime, QoR Report & Compare
SPM/Main Memory Data Sizes, Workloads, Data Objects Criticality

Fig. 6: Flow of our experimental methodology

approximation technique [17]. In the PCM approximation technique, both write errors and drift-based errors can occur through the application data. The error rates reported in [8] include both of these types of errors. In PCM's EPS estimation, the error rates in [8] are used to calculated the number of expected errors in the application data caused by the approximate PCM. Moreover, in the STT-RAM technique, accesses to STT-RAM are approximated using the RD technique and STT-RAM's EPS estimation is the expected number of STT-RAM errors during its RT. To implement these techniques, detailed approximate STT-RAM and PCM memories are integrated in the Simplescalar simulator [24] as SPM and the main memory, respectively. The delay and power parameters of the main memory and SPM have been obtained by simulating in NVsim NVM simulator [25] and also reusing parameters reported in [17], where the SPM uses LRU replacement policy [29]. Table II illustrates the simulation parameters used in our experiments.

Our experimental methodology and used tools are depicted in Fig. 6. First of all, application's workloads on MH levels and the criticality of application's data objects are characterized by the modified Simplescalar tool. Then, *1)* application's workload characteristics, *2)* PCM, STT-RAM error models and *3)* memories' delay/lifetime parameters are fed into the ILP trade-off maker to decide PCM and STT-RAM approximation levels which are used to calculate the EPS estimation, lifetime and performance. At last, *1)* the selected approximation levels, *2)* data objects' criticality information and *3)* the error models are fed into Simplescalar system simulator to obtain the QoR.

Errors in approximate STT-RAM occur totally randomly following the probability function in equation 1. Furthermore, using this technique, read errors lead to bit-flips in STT-RAM cells. However, in PCM in addition to random write errors, time-dependent drift based read errors occur. We have considered time-dependent errors in addition to random errors throughout EPS estimation.

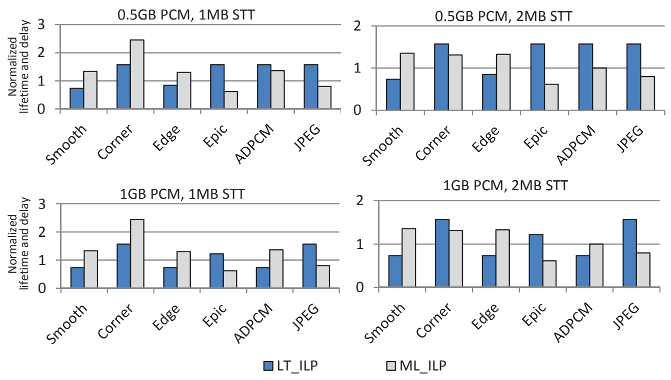

Fig. 7: PCM's lifetime in the ILP (LT_ILP) normalized to the baseline method (LT_Mod) and STT-RAM's delay in the baseline method (ML_Mod) normalized to the ILP (ML_ILP)

Fig. 4 demonstrates that among our used benchmarks, *Smoothing*, *ADPCM*, *Corner* and *Edge* benchmarks are STT-RAM-Costly (SRC) benchmarks because more than 50% of their instructions are accessing STT-RAM SPM and except Corner, the other SRC benchmarks experience moderate workloads on PCM main memory. Furthermore, *Corner*, *Epic* and *JPEG* are PCM-Costly (PRC) benchmarks because their high write traffics to the PCM lead to its early wear-out. In the following, we report PCM's lifetime/STT-RAM's delay/EPS of proposed the ILP (i.e., LT_ILP/ML_ILP/EPS_ILP), EPS of the proposed adaptive algorithm (i.e., EPS_Adp) and EPS of baseline method (i.e., EPS_Mod). LT_ILP, ML_ILP are normalized to baseline method which uses the intermediate approximation level in both PCM and STT-RAM while AdAM selects either heavy, intermediate or light levels depending on application's workloads on PCM and STT-RAM.

PCM lifetime: As Fig. 7 depicts, ILP step of our technique improves the PCM's lifetime for PRC benchmarks (i.e., Corner, Epic and JPEG) up to 1.6X (1.4X on average) and 1.2X on average for all used benchmarks. PCM's lifetime has a direct relation with PCM's capacity (i.e., S in Equation2), so AdAM's efficiency is higher for the 0.5GB PCM relatively to the 1GB PCM. However, the lifetime improvement for the 1GB PCM is still notable.

Execution time: Considering Fig. 7, ILP step of our technique improves the STT-RAM's delay for SRC benchmarks (i.e., Smoothing, Corner, ADPCM and Edge) up to 2.5X (1.5X on average) and 1.18X on average for all used benchmarks. This memory delay reduction contributes up to 38.7% (22.1% on average) improvement on ET of SRC applications. Comparing STT-RAM delay results of 1MB and 2MB SPM configurations, AdAM's efficiency is higher for the smaller STT-RAM SPM size.

Error per second: We assume that the tasks are executed using FIFO policy. Thus, when a task wait in the FIFO queue for execution, its RT increases considerably. Fig. 8 (a), demonstrates that although the tasks except the Smoothing, have

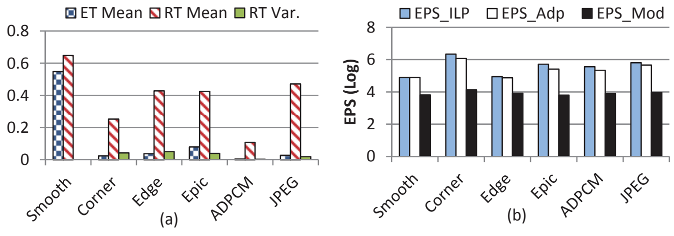

Fig. 8: Adaptive control of EPS against RT fluctuations

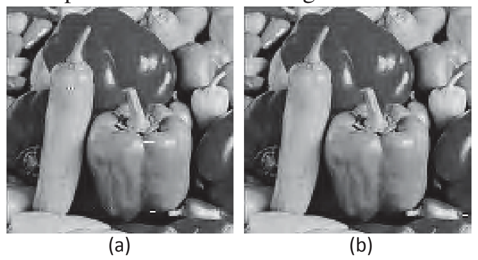

Fig. 9: QoR of Epic with Peppers image, when EPS of Epic is (a) 5.7 with the ILP, and (b) 5.4 with the algorithm

relatively low execution time, their response times increase in the FIFO waiting for the other tasks. For the Smoothing, however, the execution time is relatively high, so its wait time is not considerable compared to its execution time. Based on this, the adaptive step of the proposed technique, as shown in Fig. 8 (b), reduces EPS by adjusting PCM and/or STT-RAM approximation knobs at run-time in comparison with the ILP step which decides the knobs once at design-time. Moreover, comparing Figures 9 (a) and (b) which show the output of the Epic benchmark for the ILP step and adaptive step, adaptive step enhances the QoR.

VI. Conclusion

In this paper, we employed approximations at multiple MH levels in order to exploit the full potential of approximations throughout the MH. The proposed technique, AdAM, selects heavier PCM/STT-RAM approximation levels for PRC/SRC benchmarks and also adaptively controls application's EPS at run-time. We evaluated the proposed technique in the baseline system with 4GHz single core, 1-2MB STT-RAM SPM and 0.5-1GB PCM main memory. The experimental results show substantial improvements of the ET of applications (up to 38.7%) as well as system's lifetime (up to 1.6X) with a moderate EPS increase.

Acknowledgment

This work is supported in parts by the German Research Foundation (DFG) as part of the GetSURE project in the scope of SPP-1500 priority program Dependable Embedded Systems.

References

[1] M. Shafique *et al.*, "Cross-layer approximate computing: From logic to architectures," in *Design Automation Conf.*, 2016.
[2] S. Rehman *et al.*, "Architectural-space exploration of approximate multipliers," in *Int. Conf. on Computer-Aided Design*, 2016.
[3] F. Sampaio *et al.*, "dsvm: energy-efficient distributed scratchpad video memory architecture for the next-generation high efficiency video coding," in *Design, Automation & Test in Europe Conf.*, 2014.
[4] F. Sampaio *et al.*, "Energy-efficient architecture for advanced video memory," in *Int. Conf. on Computer-Aided Design*, 2014.
[5] C. W. Smullen *et al.*, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *Int. Symp. on High Performance Computer Architecture*, 2011.
[6] M. K. Qureshi *et al.*, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *Int. Symp. on Microarchitecture*, 2009.
[7] M. Arjomand *et al.*, "Boosting access parallelism to pcm-based main memory," *SIGARCH Comput. Archit. News*, 2016.
[8] A. Sampson *et al.*, "Approximate storage in solid-state memories," *ACM Trans. on Computer Systems*, 2014.
[9] M. Jalili and H. Sarbazi-Azad, "A compression-based morphable pcm architecture for improving resistance drift tolerance," in *Int. Conf. on Application-specific Systems, Architectures and Processors*, 2014.
[10] F. Sampaio *et al.*, "Approximation-aware multi-level cells stt-ram cache architecture," in *Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, 2015.
[11] A. Ranjan *et al.*, "Staxcache: An approximate, energy efficient stt-mram cache," in *Design, Automation & Test in Europe Conf.* IEEE, 2017, pp. 356–361.
[12] M. R. Guthaus *et al.*, "Mibench: A free, commercially representative embedded benchmark suite," in *Int. Workshop on Workload Characterization*, 2001.
[13] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems," in *Int. Symp. on Microarchitecture*, 1997.
[14] W. Zhang and T. Li, "Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system," in *Int. Conf. on Dependable Systems & Networks*, 2011.
[15] M. Awasthi *et al.*, "Efficient scrub mechanisms for error-prone emerging memories," in *Int. Symp. on High Performance Computer Architecture*, 2012.
[16] Y. Fang, H. Li, and X. Li, "Softpcm: Enhancing energy efficiency and lifetime of phase change memory in video applications via approximate write," in *Asian Test Symp.*, 2012.
[17] A. Ranjan *et al.*, "Approximate storage for energy efficient spintronic memories," in *Design Automation Conf.*, 2015.
[18] A. M. H. Monazzah *et al.*, "Quark: Quality-configurable approximate stt-mram cache by fine-grained tuning of reliability-energy knobs," in *Int. Symp. on Low Power Electronics and Design*, 2017.
[19] N. Sayed *et al.*, "Exploiting stt-mram for approximate computing," in *IEEE Test Symp. (ETS)*, 2017.
[20] B. Zeinali, D. Karsinos, and F. Moradi, "Progressive scaled stt-ram for approximate computing in multimedia applications," *IEEE Trans. on Circuits and Systems II: Express Briefs*, 2017.
[21] S. Yeo, N. H. Seong, and H.-H. S. Lee, "Can multi-level cell pcm be reliable and usable? analyzing the impact of resistance drift," in *Annual Workshop on Duplicating, Deconstructing and Debunking*, 2012.
[22] C. Y. Tatibana, C. Montez, and R. S. De Oliveira, "Soft real-time task response time prediction in dynamic embedded systems," in *Int. Workshop on Software Technolgies for Embedded and Ubiquitous Systems*, 2007.
[23] W. El-Harouni *et al.*, "Embracing approximate computing for energy-efficient motion estimation in high efficiency video coding," in *Design, Automation & Test in Europe Conf.*, 2017.
[24] D. Burger and T. M. Austin, "The simplescalar tool set, version 2.0," *ACM SIGARCH Computer Architecture News*, 1997.
[25] X. Dong *et al.*, "Nvsim: A circuit-level performance, energy, and area model for emerging non-volatile memory," in *Emerging Memory Technologies*, 2014.
[26] Z. Diao *et al.*, "Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory," *Journal of Physics: Condensed Matter*, 2007.
[27] D. Apalkov *et al.*, "Spin-transfer torque magnetic random access memory (stt-mram)," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2013.
[28] J.-T. Lin *et al.*, "Operation of multi-level phase change memory using various programming techniques," in *Int. Conf. on IC Design and Technology.*, 2009.
[29] S. Mittal *et al.*, "Replacement policies for scratch pad memory in embedded systems," in *TENCON Conf.*, 2011.