QuARK: Quality-configurable Approximate STT-MRAM Cache by Fine-grained Tuning of Reliability-Energy Knobs

Amir Mahdi Hosseini Monazzah*, Majid Shoushtari†, Seyed Ghassem Miremadi*, Amir M. Rahmani‡†, Nikil Dutt†
*Department of Computer Engineering, Sharif University of Technology
ahosseini@ce.sharif.edu, miremadi@sharif.edu
‡Institute of Computer Technology, TU Wien
†Department of Computer Science, University of California, Irvine
{anamakis, amirr1, dutt}@uci.edu

***Abstract*—Emerging STT-MRAM memories are promising alternatives for SRAM memories to tackle their low density and high static power consumption, but impose high energy consumption for reliable read/write operations. However, absolute data integrity is not required for many approximate computing applications, allowing energy savings with minimal quality loss. This paper proposes QuARK, a hardware/software approach for trading reliability of STT-MRAM caches for energy savings in the on-chip memory hierarchy of multi- and many-core systems running approximate applications. In contrast to SRAM-based cache-way-level actuators, QuARK utilizes fine-grained cache-line-level actuation knobs with different levels of reliability for individual read and write accesses which are unique to STT-MRAM and suitable for systems running multiple applications with mixed accuracy sensitivity, thus avoiding inter-application actuation interference. Our experimental results with a set of recognition, mining and synthesis (RMS) benchmarks demonstrate up to 40% energy savings over a fully-protected STT-MRAM cache, with negligible loss in the quality of the generated outputs.**

I. Introduction

Deployment of SRAM memories in embedded systems is greatly challenging in advanced VLSI technologies in particular in sub-45nm features sizes [1] due to high leakage power, reliability issues as well as low density, making them less appealing for modern embedded systems. These limitations have led to the development of alternative memory technologies with different energy-performance-reliability characteristics such as Spin Transfer Torque Magnetic RAM (STT-MRAM). STT-MRAM offers a high-density, high-speed, non-volatile choice of random access memory, however, it suffers from a critical reliability issue, called *stochastic switching* [2], which if properly handled can make STT-MRAM a promising replacement for SRAM in many applications.

Stochastic switching affects both read and write operations. It affects the read operation by causing the *read disturbance* phenomenon which is defined as the process of unintentionally changing the stored value in a STT-MRAM cell while reading the cell. Similarly, a write operation may not succeed in correctly changing the value of the cell and a *write failure* occurs.

Conventional approaches that address these sources of unreliability to preserve data integrity use a conservatively high current for write operation as well as incorporating Error Correcting Codes (ECCs) to recover the potential errors [3], [4], [5]. Both solutions significantly exacerbate the energy consumption of STT-MRAMs, counteracting some of the advantages of STT-MRAM over SRAM.

Absolute correctness may not be required for all applications. Recently, researchers [6], [7], [8], [9] have shown that a system can save energy by exposing faults to a variety of *Approximate Computing* applications that are resilient to hardware errors during their execution. In particular Recognition, Mining and Synthesis (RMS) applications that are widely used in embedded systems could still process information usefully with error-prone elements [8]. At the same time, such applications often have a large working-set size which stresses the memory hierarchy, and therefore causes high energy consumption in the on-chip memory hierarchy.

Two recent studies have investigated reliability-energy trade-offs in on-chip cache memories for applications in the approximate computing domain [9], [10]. In [9], the authors present an approximate computing approach for SRAM caches which uses an SRAM-specific knob (i.e., voltage scaling), making it infeasible to be applied to STT-MRAM caches as the knob is technology-specific and shows different reliability-energy characteristics. The work presented in [10] targets STT-MRAM caches and utilizes the retention as a knob to trade accuracy for energy consumption in STT-MRAMs. This is done by changing the thermal stability factor during the manufacturing process of STT-MRAMs. The approach does not also provide a dynamically reconfigurable infrastructure for STT-MRAM caches. In this paper, we propose QuARK, a hardware/software approach for dynamically trading-off reliability (accuracy) for energy in STT-MRAM caches of systems running approximate computing workloads. In contrast to previous approaches, we exploit fine-grained tuning of energy-reliability knobs exclusive to STT-RAM technology to leverage energy savings.

QuARK enables software programmers to declare approximate data objects in the source code of the application along with an acceptable read and write reliability guarantees. This information is then passed to the hardware and used during the execution to adjust two main STT-MRAM specific reliability-energy knobs: read current and write current. For approximate read and write operations, these knobs are set to the lowest current that is sufficient to meet the programmer-specified acceptable level of reliability. Applying a lower current significantly reduces the energy consumed in the STT-MRAM cache at the cost of minor accuracy loss in the output. The main contributions of this paper are as follows:

- We introduce novel fine-grained STT-MRAM specific knobs to be used for approximate cache design that enable interference-free actuations independently tuned for different applications.
- We present the architectural and hardware support to utilize the reliability-energy knobs enabling multiple levels of reliability for individual read and write operations.
- We provide the necessary software support for the application to configure the cache reliability for different data objects in the program.

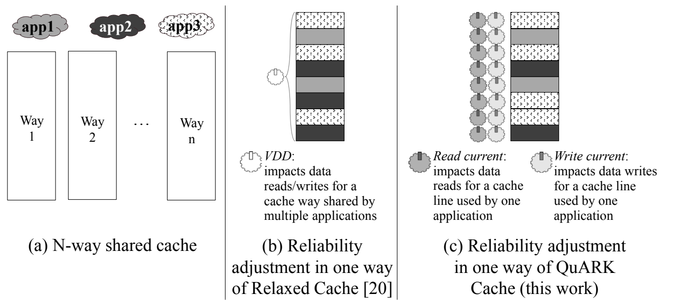

app1
app2
app3
Way 1
Way 2
...
Way n
(a) N-way shared cache
VDD: impacts data reads/writes for a cache way shared by multiple applications
(b) Reliability adjustment in one way of Relaxed Cache [20]
Read current: impacts data reads for a cache line used by one application
Write current: impacts data writes for a cache line used by one application
(c) Reliability adjustment in one way of QuARK Cache (this work)

Fig. 1. QuARK provides better flexibility compared to SRAM-based Relaxed Cache introduced in [9].

Compared to the related work on SRAM caches [9] that uses cache-way-level knobs, QuARK presents a more fine-grained (i.e., cache-line level) actuation capability (Fig. 1) enabling it to offer the following advantages: i) the knob actuations do not affect any other cache block unlike the actuation in [9] which requires flushing all the affected blocks, ii) multiple applications with different degrees of reliability can share the same cache without affecting each other's guaranteed level of reliability, and iii) the reliability level requested for a piece of data can be changed at runtime, if the quality of output is not satisfactory.

II. Motivational Example

In this section, we demonstrate the effectiveness of QuARK in making a trade-off between the output quality and energy consumption for two approximate computing applications from MiBench/-Susan [11]: image smoothing and edge detection. Both applications take an image as input. The data structures that hold the original image pixels as well as transformed pixels can tolerate certain levels of imprecise storage in the memory hierarchy. In this use case, we consider three levels of reliability for read and write operations in a STT-MRAM based L2 cache equipped with QuARK. Figure 2 shows energy consumption and accuracy delivered by QuARK compared to when the system uses a fully-protected STT-MRAM cache[1].

For the image smoothing application, the simulation results show that QuARK provides the opportunity to save energy by up to 34% while maintaining PSNR of the output image well above the threshold of 30 dB. Our experiments show that visually no perceptual differences are observed among the output images using different accuracy levels for this application. Thus, a programmer can select reliability level 3 for the L2 cache read and write operations without concern for quality degradation. Similarly, for the edge detection application, based on the different levels of accuracy, QuARK can deliver up to 39% of energy saving with only 4.85% loss in the quality of the output image. Intuitively, there is a noticeable quality degradation in the output when the reliability level 3 is used. If the application cannot tolerate this level of quality degradation, programmers can select level 1 or 2 for QuARK-enabled L2 cache and still benefit from 29%~34% energy savings delivered by QuARK.

In contrast to many previous efforts, QuARK enables energy savings for multi- or many-core systems that concurrently execute different applications with *mixed-reliability* requirements. For example, consider a scenario where it is desired to simultaneously run both image smoothing and edge detection applications in a multi-core platform equipped with QuARK. We could run image smoothing at reliability level 3 and edge detection at reliability level 2 and get acceptable quality outputs, while saving energy. However previous coarse-grained approximate cache storage approaches (e.g., SRAM-based approaches) require both applications to run at the same reliability (level 2) to achieve acceptable quality, resulting in higher energy consumption.

Our profiling for image smoothing and edge detection shows that over 57% and 70% of accesses to L2 cache in these applications are approximate accesses, respectively. A coarse-grained approximate computing approache cannot efficiently exploit this opportunity for power savings since the accesses to various L2 cache ways are a mix of accurate and approximate accesses with different levels of reliabilities. For instance, we observe that respectively for the image smoothing and edge detection applications, only in 11% and 16% of cache ways in all epochs, the entire access pattern is associated with a single reliability level. In other words, these are the only cases with no intra/inter application interference where a course-grained knob can be applied. This confirms that coarse-grained knob-tuning approaches would miss the opportunity to actuate on 100% of the approximate data, and therefore fine-grained actuation knobs (as exploited by QuARK) are required to turn this opportunity into energy savings by serving consecutive accurate and approximate accesses to each cache way.

[1]The accuracy and energy consumption reported in Fig. 2 are obtained by simulating a QuARK-enabled L2 cache in gem5 [12], using simulation parameters that will be presented in Section 5.

Image Smoothing				
Input	Golden	Reliability Level 1	Reliability Level 2	Reliability Level 3
Normalized Dynamic Energy	1	0.76	0.72	0.66
PSNR	-	50.6	51.1	39.2

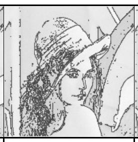
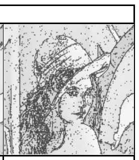
Edge Detection				
				
Normalized Dynamic Energy	1	0.71	0.66	0.61
Mean Pixel Difference	0	2.17×10^{-3}	5.67×10^{-3}	4.85×10^{-2}

Fig. 2. Energy savings vs. output fidelities delivered by QuARK with different knob settings for Image Smoothing and Edge Detection benchmarks.

QuARK enables each application running on a system to independently and in a fine-grained manner trade accuracy of storage in on-chip cache hierarchy for energy savings. This is enabled by using a set of properties specific to STT-MRAM which are discussed in the next section.

III. STT-MRAM Reliability/Energy Trade-off Knobs

In this section, we first review the structure of a STT-MRAM cell and the mechanisms of reading from and writing to it. Then, we discuss the available circuit-level knobs in STT-MRAMs which can be used to trade accuracy for energy.

A. STT-MRAM Basics

The standard STT-MRAM cell (1T-1J) includes a Magnitude Tunnel Junction (MTJ), and an access transistor. MTJ consists of an oxide barrier layer that is sandwiched between two ferromagnetic layers. One of the ferromagnetic layers has fixed magnetic field direction (i.e., reference layer), while the magnetic field direction of the other layer can be changed (i.e., free layer). Relative magnetic field direction of these layers delivers different resistances use to store values. Read and write operations in STT-MRAM cell are performed by applying either a small current to read MTJ resistance by sense amplifier, or a high current to change the resistance (i.e., write a new value) in the MTJ.

B. Reliability-Energy Knobs

As noted previously, STT-MRAM switching is a stochastic operation. Therefore, read and write operations can be performed at different levels of reliability and consequently energy consumption [13], [14].

Write Operation: During a write operation, depending on the amount and duration of the applied write current, the direction of the magnetic field of the free layer in MTJ may not change, which can result in a write failure. The write failure probability can be calculated by Equation (1) [15]:

$$P_{wf}(t_w) = exp(-t_w \times \frac{2\mu_B p(I_w - I_{C_0})}{(c + ln(\Pi^2 \frac{\Delta}{4})) \times (em(1+p^2))}) \qquad (1)$$

where Δ is the thermal stability factor, I_{C_0} is the critical MTJ switching current at 0°K, c is the Euler constant, e is the magnitude of electron charge, m denotes the magnetic momentum of the free

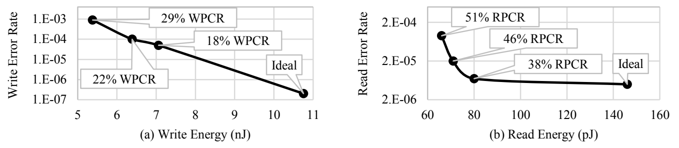


Fig. 3. STT-MRAM knobs for reliability-energy trade-off in 1MB cache. (a) Write Pulse Current Reduction (WPCR), and (b) Read Pulse Current Reduction (RPCR).

layer, p is the tunneling spin polarization, μ_B is the Bohr magneton, I_w is the write current, and t_w is the write pulse width. I_w is one of the effective circuit-level knobs available to control the reliability-energy trade-off during a write operation. Generally, a lower I_w decreases the write energy, but it also amplifies the probability of a write failure.

Based on the information reported in [16], we modeled a 1MB STT-MRAM cache in NVSim [17]. Figure 3(a) shows the write energy vs. write error rate of this cache at different write currents. We see that modulating I_w leads to savings in write operation energy, e.g., 2× improvement for an error probability of 9×10^{-4}.

Read Operation: During a read operation, depending on the amount and duration of the applied read current, an unintentional magnetic field direction flip in the free layer of MTJ may happen due to stochastic switching which can result in read disturbance. The read disturbance probability can be calculated using Equation (2) [18]:

$$P_{rd} = 1 - exp(-\frac{t_r}{\tau} \times exp(-\frac{\Delta(I_{C_0} - I_r)}{I_{C_0}})) \quad (2)$$

where τ is the attempt period, I_r is the read current, and t_r is the period of read pulse. Unlike write operation, both the read energy consumption and the read disturbance probability are decreased by reducing I_r. However, another constraint prevents us from applying a very small read current. Lowering the read current amplitude below a pre-defined sense amplifier threshold leads to different type of read errors in STT-MRAM which is called *decision failure*.

Accordingly, I_r is one of the effective circuit-level knobs to control the reliability-energy trade-off during a read operation. Figure 3(b) depicts the reliability-energy trade-off for read operation of aforementioned 1MB STT-MRAM cache. We see that decreasing the read pulse current results in energy savings for read operation. For example, modulating I_r leads to over 2.2× improvement in read energy consumption for an error probability of 9×10^{-5}.

IV. The QuARK Approach

QuARK enables a system with STT-MRAM caches running approximate computing applications to trade accuracy of storage in on-chip memory for energy savings. It captures the information about non-critical data objects from software and adjusts the reliability-energy knobs accordingly. This section presents the details of software and hardware supports for QuARK.

A. Software Support

Data structures of a program can be categorized into critical (i.e., corruption would most likely lead to catastrophic failure or significant quality degradation) or non-critical (i.e., quality degradation resulting from corruption would be acceptable). This concept has been used before in [7] where a simple partitioning of data – into critical and non-critical – has been shown effective to improve DRAM energy efficiency.

In order to utilize the energy saving opportunities provided by QuARK, the programmer has to identify non-critical data structures. We believe a software programmer can easily make this distinction since (s)he is aware of the functionality and semantics of different parts of the application and related data structures. Frameworks like

TABLE I
QuARK APIs

Method	Parameters	Note
add_approx	BaseVA	Base virtual address of approx. memory region
	Size	Size of the approx. memory region
	ReliabilityLevel	Required reliability guarantee
remove_approx	BaseVA	Base virtual address of approx. memory region
	Size	Size of the approx. memory region

Rely [19] and Accept [20] could also assist the programmer in identifying non-critical data objects in a program.

Typically, non-critical data objects and their corresponding error rate are determined through a set of simulations with fault injection and finally measuring the quality of generated output against the golden output. These categorizations can be reflected in the program by annotating different data structures. One category of approaches define type qualifiers and dedicated assembly-level store and load instructions for this purpose [6], [19]. These approaches require major modifications to the compiler, ISA, and processor architecture. We present another approach for QuARK that uses dynamic declarations which are enforced at run-time. Our approach is ISA-independent, can be integrated with today's processor architectures easily and it requires only minor modifications to the compiler toolchain.

QuARK provides two Application Program Interface (API) calls to the programmer: `add_approx` and `remove_approx`. Table I shows their formats. `BaseVA` is the base address of the non-critical data object, `Size` is the size of non-critical data object, and `ReliabilityLevel` is the required reliability guarantee for that data object.

The QuARK APIs communicate with the QuARK hardware support (introduced in Section IV-B) to pass the information about non-critical data objects and their acceptable reliability level. There are two possible approaches for integrating these two APIs into the system: 1) The ISA of the processor can be modified to support the QuARK APIs. With this approach, the processor has to be modified to be able to directly communicate with the QuARK hardware support. 2) The QuARK APIs can be implemented within a special run-time library. With this approach, hardware components of QuARK become memory-mapped interfaces. The run-time library uses normal read/write instructions to transfer the information provided by the APIs to the hardware.

Figure 4 shows an example on how QuARK APIs can be used in the source code of a hypothetical face-detection application. According to [21], the environmental lighting affects the quality of face detection algorithm. Thus, considering a reasonable output quality in the worst case environmental lighting (when the lighting is either very low or very high), we can trade quality for energy

```
unsigned *image;
int reliability_level = RL0;
...
image = (unsigned int*)malloc(WIDTH*HEIGHT*sizeof(
     unsigned));
if(approximation_enabled) {
    switch(get_environment_lighting()) {
        case 0~30  :
            reliability_level = RL0;
        case 30~70 :
            reliability_level = RL1;
        case 70~100:
            reliability_level = RL0;
    }
}
add_approx(image,WIDTH*HEIGHT*sizeof(unsigned),
     reliability_level);
...
load_image(image);
face_detection(image);
...
remove_approx(image,WIDTH*HEIGHT*sizeof(unsigned));
...
```

Fig. 4. A pseudo-code example showing how QuARK APIs can be used in a face detection application.

saving in the cases that lighting is in normal condition (30~70 in this example).

B. Hardware Support

Figure 5 shows how the required hardware support for QuARK can be integrated into the architecture. Deploying QuARK for L1 private and L2 shared caches of memory hierarchy in a multi-core architecture requires adding two types of modules: (1) QuARK approximation table and (2) QuARK controllers. Additionally, the interconnect should be modified to carry the reliability level information. These modifications are highlighted in gray in Fig. 5.

QuARK Approximation Table: QuARK approximation table is responsible for storing the information provided by the QuARK APIs. Each core includes a private approximation table. Anytime an `add_approx` is called in the thread running on that core, its parameters (i.e, baseVA, size, and reliability level) are saved in one of the rows of this table. Similarly, `remove_approx` removes all or part of the information stored in a row.

This table is placed next to the Translation Look-aside Buffer (TLB) of each core. For every memory access, and during the TLB lookup, the QuARK approximation table is also searched. If the virtual address of the memory access falls within the boundaries of one of the rows in the table, the reliability of that access is set according to the reliability level stored in the corresponding row. For accesses that do not hit any row in the table, the reliability level is set to the maximum possible reliability. QuARK controls STT-MRAM knobs at cache block granularity. The virtual address ranges provided by the APIs should be aligned to cache block boundaries, because the cache blocks that contain a mix of critical and non-critical data should not be approximated. An address alignment module embedded in the approximation table performs the required alignment. The addresses provided by the QuARK APIs that will be stored in QuARK approximation table are Virtual Address (VA). However, in most architectures (e.g., ARM), caches work with Physical Addresses (PA).

QuARK Controller: The reliability level for each access is set by looking up the QuARK approximation table. Then the memory request, now augmented with a reliability level, is passed to QuARK controller(s). QuARK controller sits next to the cache controller and it is responsible for setting the STT-MRAM reliability-energy trade-off knobs (i.e., read current and write current) for each access. The feasibility of designing efficient peripheral circuits for changing the current (voltage) level during the runtime is evaluated in [22], [23], [16]. QuARK controller receives the reliability level set by the approximation table and selects the minimum current setting that satisfies the proper level of reliability for that access. One QuARK

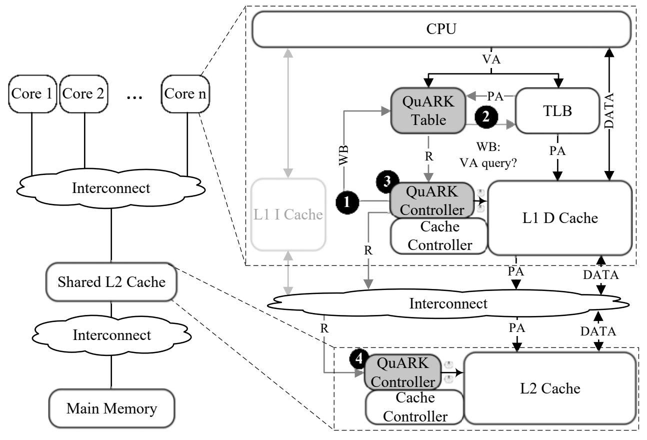


Fig. 5. Integrating QuARK into the architecture. Required changes are highlighted in gray.

controller is needed for each private L1 cache and one for each shared L2 cache.

Support for Cache Fillings and Write-backs: Special consideration might be needed for interactions between L1 and L2 caches anytime there is an L1 data cache miss.

■ L1 Cache Filling: Anytime a cache miss occurs in L1 data cache, a cache filling is required. The memory access that resulted in a cache miss is already augmented with a reliability level when the approximate table was searched during TLB lookup. The QuARK controller will use this reliability level to fill the L1 cache line when the request is served by the L2 cache.

■ L2 Cache Filling: Cache filling in L2 is required whenever a L1 cache request is missed in the L2 cache. Similar to the previous case, the reliability level of this request is already provided by the approximation table. The QuARK controller of L2 will use this reliability level during the filling operation of L2 cache line from upper-level memories.

■ Write-backs from L1 to L2: There are some times that L1 cache controller decides to evict a dirty cache line from L1 cache. This happens when either a new block should be replaced by dirty a one, or the last modified version of the block in L1 cache should be written to upper-level memories. For write-back from L1 data cache to L2 cache, QuARK follows a different flow. Following Fig. 5, the detailed steps to write back a dirty block from L1 data cache to L2 cache are as follows:

1) QuARK controller in L1 data cache sends the PA of dirty block to the QuARK table.
2) QuARK table produces the VA queries for all of its entries and checks the TLB PA output to find the PA of dirty block in address intervals of entries. Then, QuARK table determines the reliability level of this write back request.
3) The QuARK controller of L1 data cache reads the data of dirty block with the provided reliability level and passes the data and reliability level to L2 cache.
4) The QuARK controller of L2 cache adjusts the current level and performs the L1 data write back operation.

■ Write-backs from L2 to upper-level memory: Write-backs from L2 cache are issued due to the same reasons that mentioned for L1 write-back requests. For all of the write-back operations from L2 cache, QuARK considers the highest reliability level for reading the victim blocks from L2 cache and writing them to the upper-level memory.

V. Evaluations

In the following, we show experiments with a mix of approximate applications to evaluate the QuARK's capabilities in saving energy in the on-chip memory hierarchy under different levels of accuracy demanded by applications.

A. Experimental Setup

In order to implement our scheme in detail, we modified the cache architecture in the gem5 framework [12] for a multi-core architecture. In this work, to assess the efficiency of the QuARK in enhancing the energy characteristics of multi-core architectures, we enable QuARK for L2 cache as an exemplar. We used gem5's pseudo-instructions for implementing QuARK's language extensions. The common gem5 parameters used in all our simulations are summarized in Table II.

We randomly injected faults into different cache blocks during read and write operations, with uniform distribution over different bit positions. We used bit error rate (BER) and access current data

TABLE II
GEM5 SETTINGS

Parameter	Value	Parameter	Value
ISA	ARMv7-A	L1 $ Size, Assoc.	64KB, 4
No. of Cores	4	L2 $ Size, Assoc.	1MB, 16
Cache Config	L1 (Private) L2 (Shared, QuARK-enabled)	Cache Block Size	64B

TABLE III
ACCURACY-ENERGY TRANSDUCER MAP FOR 1MB QUARK-ENABLED STT-MRAM CACHE. ENERGY CONSUMPTIONS ARE REPORTED FOR 64-BYTE CACHE LINE.

Accuracy Level	**Read Current**	**Read Error Rate**	**Read Energy**	**Write Current**	**Write Error Rate**	**Write Energy**
L0 (Baseline)	$49\mu A$	protected	$0.146nJ$	$653\mu A$	protected	$10.755nJ$
L1	$30\mu A$	7×10^{-6}	$0.080nJ$	$529\mu A$	5×10^{-5}	$7.059nJ$
L2	$26\mu A$	2×10^{-5}	$0.071nJ$	$503\mu A$	1×10^{-4}	$6.386nJ$
L3	$24\mu A$	9×10^{-5}	$0.066nJ$	$461\mu A$	9×10^{-4}	$5.378nJ$

TABLE IV
LIST OF APPROXIMATE APPLICATIONS

Benchmark	**Domain**	**Quality Metric**
Corner Detection	Computer Vision	Mean Pixel Difference
Edge Detection	Computer Vision	Mean Pixel Difference
Image Smoothing	Computer Vision	PSNR
Blackscholes	Financial Analysis	Average Relative Error
Image Scale	Multimedia	PSNR
Sobel Filter	Computer Vision	Mean Pixel Difference

from [16] and modeled a 1MB STT-MRAM cache in NVSim [17] to extract energy consumption. The details of error rates and energy consumptions of the modeled cache can be seen in Table III.

Benchmarks: Table IV lists the RMS applications we use in our evaluations and their quality metrics. We annotated each application by inserting `add_approx` and `remove_approx` in the source code for non-critical data objects .

Figure 6 shows the distribution of overall and approximate read-/write operations in L2 cache for the benchmarks listed in Table IV when they run on a single-core configuration. As it can be seen, on average, 80% of accesses to L2 cache in these benchmarks can be done in the approximate mode, showing the significant energy saving potentials in such applications. Furthermore, about 45% of these accesses are for read requests and 35% are for write requests.

To evaluate the efficiency of QuARK, we consider the effects of multi-programming in a shared L2 cache equipped with QuARK. To this end, we run different mixes of benchmarks mentioned in Table V in two modes: fixed-reliability mode, and mixed-reliability mode. In fixed-reliability mode, we run each workload mix in four reliability levels (RLs): RL0 (i.e., fully-protected), RL1, RL2 and RL3 (i.e., least-reliable) configurations. In the mixed-reliability mode, we consider different levels of reliability for each benchmarks in each combination. Our empirically-chosen criteria to set the RL for each application is based on the user-level QoS desirability. For instance, as shown in the motivational example, RL3 provides an acceptable QoS for Image Smoothing, while for Edge Detection RL2 is more desirable. Note that these RLs can be chosen by the user/designer w.r.t any other policies. The workload mixes along with their reliability levels are listed in in Table V.

B. Experimental Results

Figure 7 shows a summary of our experiments. We evaluate QuARK in a multi-core platform from these perspectives: 1) flexibility in running different applications with different reliability levels, 2) energy savings, and 3) delivered quality.

Figure 7(a) depicts the average reliability level distribution of accesses to the shared L2 cache equipped with QuARK. To evaluate the flexibility of QuARK in delivering the L2 cache accesses with these various required reliability levels, over the approaches that use

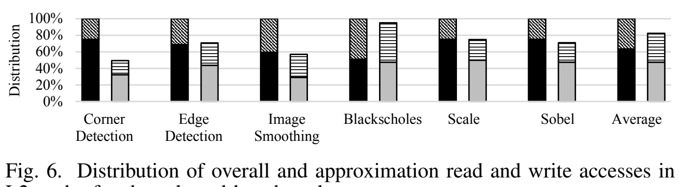


Fig. 6. Distribution of overall and approximation read and write accesses in L2 cache for the selected benchmarks.

TABLE V
LIST OF WORKLOAD MIXES

Workload Mixes	**Benchmarks**	**RLs (respectively)**
Comb1	(Corner Detection, Sobel, Image Smoothing, Blackscholes)	(RL1, RL3, RL2, RL3)
Comb2	(Scale, Blackscholes, Sobel, Image Smoothing)	(RL2, RL2, RL1, RL3)
Comb3	(Sobel, Corner Detection, Scale, Edge Detection)	(RL2, RL3, RL2, RL1)
Comb4	(Blackscholes, Scale, Edge Detection, Image Smoothing)	(RL3, RL1, RL2, RL2)

coarse-grained actuation knobs, we conduct an experiment, where we divide the execution time of workloads to 10ms epochs. Unlike QuARK that uses fine-grained actuation knobs and assigns the required reliability level to each accesses on-demand, the coarse-grained approaches either should decide about the reliability level of the cache ways (memory banks) statically, or in the best case, dynamically and at the start of each epoch.

Thus, in each epoch, the coarse-grained approaches always should set the reliability level of cache ways, based on the most vulnerable data that should be served by that way. However, both static and dynamic fine-grained approaches can modify cache replacement policy and restrict the traffics of vulnerable data to susceptible ways by paying significant performance penalty. According our experimental results, for all workloads mentioned in Fig. 7(a), all the 16 ways of the simulated shared L2 cache contains at least one vulnerable block (accessed in RL0 mode) at each epoch. Thus, in all epochs, coarse-grained approaches should pay extra energy overhead to maintain the integrity of vulnerable blocks in the cache ways (even for the blocks that contain imprecise data), while fine-grained actuation knobs in QuARK not only keeps the integrity of vulnerable data, but also provide the opportunity for energy saving in the blocks that contain imprecise data.

Considering the energy consumption reported in Table III, we can find that with the current distribution of approximate read and write operations in the selected benchmarks, and the significant difference between L2 cache read and write energy consumptions, the write energy saving offered by QuARK dominates the gain achieved by approximate read operations in QuARK. Thus, to keep a reasonable trade-off between output quality and energy consumption, we decided to only use the write operation knob in our experiments in following. It should be noted that QuARK read operation knob would be still useful for the read intensive approximation benchmarks or future STT-MRAM technologies that alleviate the difference between read and write energy consumptions.

From Fig. 7(b), we see that QuARK delivers up to 40% energy saving for Comb3 at level 3 of fixed-reliability mode. For mixed-reliability mode, it can be observed that the assigned reliability level to each benchmarks as well as the combination of benchmarks in each workload have a major contribution in the energy saving of L2 cache equipped with QuARK. For example, considering mixed-reliability mode, it can be seen that changing the reliability level of Blackscholes from level 3 to 2, and also replacing Corner Detection by Scale are the two most important contributors for less energy saving in Comb2 compared with Comb1.

To compare the quality of the benchmarks, we used the quality metrics introduced in Table IV. Figure 7(c) depicts the average results of relative error for Blackscholes in all experiments. We see that using level 1 or 2 for the reliability of L2 cache keeps the quality degradation less than 5%. However, high energy saving delivered by using level 3 may not be acceptable for some applications due to high degree of quality degradation (23%). Figure 7(d) shows the average PSNR used as quality metric for image smoothing and Scale. We see that Scale is more susceptible to approximation computing. The average calculated PSNR of Scale outputs at level 3 is less than 30dB threshold, which may not be acceptable for some applications (still level 1 or 2 can be selected for these applications). However, the average PSNR of image Smoothing in all of the reliability levels are higher than 30db threshold. Figure 7(e) shows the mean pixel difference used as quality metric for Corner detection, Edge detection,

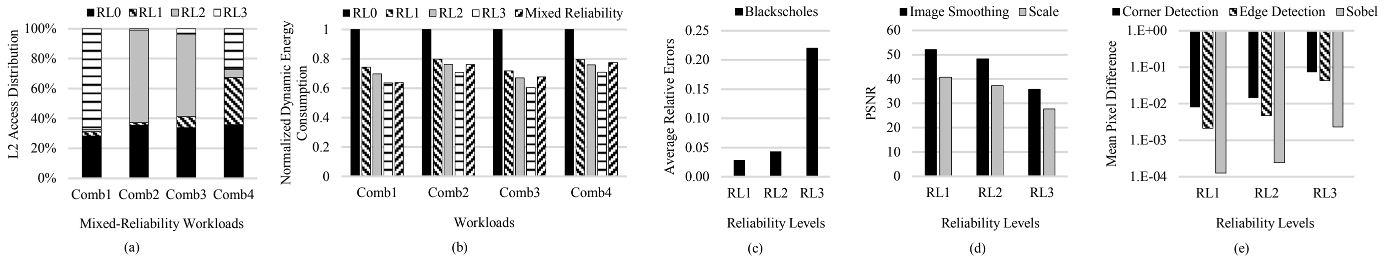


Fig. 7. QuARK evaluation results: (a) Distribution of accesses in mixed-reliability workloads, (b) Energy savings (normalized to fully-protected STT-MRAM L2 cache), (c) Average relative error for blackscholes benchmark, (d) PSNR for Scale and Image Smoothing benchmarks, and (e) Mean pixel difference for Corner Detection, Edge Detection and Sobel benchmarks.

and Sobel filter. We see that among these three applications, on average, Corner detection is more susceptible to approximate cache storage. However, its output quality is still degraded by less than 10% using RL3 execution mode.

VI. Related Work

While most of the early research in approximate computing have focused on computation and programming language support, there are a few works that apply approximation to different components of the memory hierarchy. SRAM memories consume significant static power on retaining data. The authors of [9] have shown how a number of cache ways of a SRAM cache can operate at lower than the nominal voltage. These relaxed ways are used to hold approximate data and the protected ways are used to hold critical data. [24] exploits the approximate similarity of data in last level caches to use the cache storage more efficiently and hence save static and dynamic energy. Two recent works have considered applying the idea of approximation to SST-MRAM structures. The authors of [16] have utilized similar energy-reliability knobs that are used in QuARK, however they are applied to scratchpad memories used in a vector processor running single application. [25] proposes an approximation-aware Multi-Level Cells (MLCs) STT-MRAM cache architecture to trade reliability with large capacity delivered by MLC STT-MRAM cells. [10] uses approximate computing to tolerate the increased retention failure rate caused by relaxing the thermal stability factor of STT-MRAM. Similarly, DRAM off-chip memories can reduce the power spent on refresh cycles where bit flips are allowed [7], [26]. The work in [27] shows how approximate storage in persistent memories where storage cells are at the risk of wear out, can reduce the number of flipped bits to prolong the device lifetime.

VII. Conclusion and Future Work

This paper introduced QuARK, a hardware/software approach for trading reliability of STT-MRAM caches for energy savings in the on-chip memory hierarchy of systems running approximate applications. QuARK utilizes fine-grained actuation knobs to efficiently control reliability-energy trade-off for individual accesses of concurrently running applications. Our simulation results demonstrated up to 40% energy savings over a fully-protected STT-MRAM L2 cache, with acceptable quality loss. Our future work will extend our platform to explore the efficiency of QuARK at different levels of memory hierarchy, as well as to analyze the effects of concurrent running of non-approximate and approximate applications.

Acknowledgments

This paper is dedicated to the memory of our wonderful colleague, Professor Seyed Ghassem Miremadi who passed away in April 2017.

We acknowledge financial support by the Marie Curie Actions of the European Union's H2020 Programme.

References

[1] B. Raj *et al.*, "Nanoscale FinFET Based SRAM Cell Design: Analysis of Performance Metric, Process Variation, Underlapped FinFET, and Temperature Effect," *IEEE Circuits and Systems Magazine*, 2011.
[2] T. Devolder *et al.*, "Single-Shot Time-Resolved Measurements of Nanosecond-Scale Spin-Transfer Induced Switching: Stochastic Versus Deterministic Aspects," *Phys. Rev. Lett.*, 2008.
[3] H. Sun *et al.*, "Design Techniques to Improve the Device Write Margin for MRAM-based Cache Memory," in *Proc. of GLSVLSI*, 2011.
[4] W. Kang *et al.*, "High Reliability Sensing Circuit for Deep Submicron Spin Transfer Torque Magnetic Random Access Memory," *Electronics Letters*, 2013.
[5] Z. Azad *et al.*, "An Efficient Protection Technique for Last Level STT-RAM Caches in Multi-Core Processors," *IEEE TPDS*, 2016.
[6] A. Sampson *et al.*, "EnerJ: Approximate Data Types for Safe and General Low-power Computation," in *Proc. of PLDI*, 2011.
[7] S. Liu *et al.*, "Flikker: Saving DRAM Refresh-power Through Critical Data Partitioning," in *Proc. of ASPLOS*, 2011.
[8] V. K. Chippa *et al.*, "Analysis and Characterization of Inherent Application Resilience for Approximate Computing," in *Proc. of DAC*, 2013.
[9] M. Shoushtari *et al.*, "Exploiting Partially-Forgetful Memories for Approximate Computing," *IEEE Embedded Systems Letters*, 2015.
[10] F. Oboril *et al.*, "Fault Tolerant Approximate Computing Using Emerging Non-volatile spintronic Memories," in *Proc. of VTS*, 2016.
[11] M. R. Guthaus *et al.*, "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," in *Proc. of WWC*, 2001.
[12] N. Binkert *et al.*, "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, 2011.
[13] K. Munira *et al.*, "A Quasi-Analytical Model for Energy-Delay-Reliability Tradeoff Studies During Write Operations in a Perpendicular STT-RAM Cell," *IEEE TED*, 2012.
[14] H. Li *et al.*, "Performance, Power, and Reliability Tradeoffs of STT-RAM Cell Subject to Architecture-Level Requirement," *IEEE TMAG*, 2011.
[15] M. M. de Castro *et al.*, "Precessional spin-transfer switching in a magnetic tunnel junction with a synthetic antiferromagnetic perpendicular polarizer," *Journal of Applied Physics*, 2012.
[16] A. Ranjan *et al.*, "Approximate Storage for Energy Efficient Spintronic Memories," in *Proc. of DAC*, 2015.
[17] X. Dong *et al.*, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *IEEE TCAD*, 2012.
[18] R. Takemura *et al.*, "A 32-Mb SPRAM With 2T1R Memory Cell, Localized Bi-Directional Write Driver and '1'/'0' Dual-Array Equalized Reference Scheme," *IEEE JSSC*, 2010.
[19] M. Carbin *et al.*, "Verifying Quantitative Reliability for Programs That Execute on Unreliable Hardware," in *Proc. of OOPSLA*, 2013.
[20] A. Sampson *et al.*, "Mobile Systems IV," University of Washington, Tech. Rep., 2015.
[21] Forczmański *et al.*, "An Algorithm of Face Recognition Under Difficult Lighting Conditions," *Electrical Review*, 2012.
[22] F. Ishihara *et al.*, "Level conversion for dual-supply systems," *IEEE TVLSI*, 2004.
[23] D. Lee *et al.*, "High-performance Low-energy STT MRAM Based on Balanced Write Scheme," in *Proc. of ISLPED*, 2012.
[24] J. S. Miguel *et al.*, "DoppelgÄNger: A Cache for Approximate Computing," in *Proc. of MICRO*, 2015.
[25] F. Sampaio *et al.*, "Approximation-aware Multi-level Cells STT-RAM Cache Architecture," in *Proc. of CASES*, 2015.
[26] A. Raha *et al.*, "Quality Configurable Approximate DRAM," *IEEE TC*, 2016.
[27] A. Sampson *et al.*, "Approximate Storage in Solid-State Memories," *ACM Trans. Comput. Syst.*, 2014.