A Cache-Assisted ScratchPad Memory for Multiple Bit Error Correction

Hamed Farbeh, *Student Member, IEEE,* Nooshin Sadat Mirzadeh, Nahid Farhady Ghalaty, Seyed-Ghassem Miremadi*, Senior Member, IEEE,* Mahdi Fazeli, Hossein Asadi*, Senior Member, IEEE*

***Abstract*— *ScratchPad Memory* (SPM) is widely used in modern embedded processors to overcome the limitations of cache memory. The high vulnerability of SPM to soft errors, however, limits its usage in safety-critical applications. This paper proposes an efficient fault-tolerant scheme, called *Cache-Assisted Duplicated SPM* (CADS), to protect SPM against soft errors. The main aim of CADS is to utilize cache memory to provide a replica for SPM lines. Using cache memory, CADS is able to guarantee a full-duplication of all SPM lines. We also further enhance the proposed scheme by presenting *Buffered-CADS* (BCADS) that significantly improves the CADS energy efficiency. BCADS is compared with two well-known duplication schemes as well as single error correction scheme. The comparison results reveal that 1) BCADS imposes 13.6% less *Energy-Delay Product* (EDP) overhead than the duplication schemes and it does not require to modify the SPM manager and target application; 2) in comparison to the conventional *Single Error Correction-Double Error Detection* (SEC-DED) scheme, BCADS provides significantly higher error correction capability by correcting up to 4-bit burst errors using low-cost 4-bit interleaved parity code. Moreover, the area overhead for error correction and the performance overhead of BCADS are negligible (less than 1%), whereas the area and performance overheads are 21.9% and 6.1% for SEC-DED, respectively. Furthermore, BCADS imposes about 10.7% lower EDP overhead as compared to the SEC-DED scheme.**

Index Terms—Cache Memory, Data Duplication, Multiple-Bit Upset, ScratchPad Memory, Soft Error Correction.

I. Introduction

A wide range of modern embedded processors includes both *ScratchPad Memory* (SPM) and cache memory in their architectures to fulfill the application requirements of predictability, performance, and energy budget. Examples of such processors are ARM Cortex-R Series [1] and SH7785 [2] that are used in automotive, industry, and medical applications. These applications are safety-critical requiring highly reliable processors.

One major source of system failures in such applications is soft errors caused by radiation induced particles strike into chips [3]. *Single Event Upsets* (SEUs) and *Single Event Multiple Upsets* (SEMUs) are two types of soft errors in SPM and cache as on-chip SRAM memories [3]. It has been reported that more than 60% of the chip area is occupied by these memory cells [3]-[5] which makes them the most probable component to particles strike [3].

Correcting soft errors in on-chip memories (SPM or cache) can be categorized into two approaches [7]. The first approach is the use of *Error Correcting Codes* (ECCs), e.g., *Single Error Correction-Double Error Detection* (SEC-DED) and SEC-DED-*Double Adjacent Error Correction* (SEC-DED-DAEC), to detect and correct errors. All of the on-chip memories can be protected using this approach. However, this approach has two serious problems: a) a limited error correction capability [8][9], and b) a significantly higher overhead, when ECCs are employed to correct multiple bit errors such as SEMUs [8]-[10]. The second approach to detect and correct soft errors is a joint use of parity code and a duplication of memory entries; we call this approach as *parity-duplication*. The main advantage of this approach is its capability to correct all detected errors. This approach has been commonly applied to structures such as instruction-cache, instruction-SPM, and write-through data-cache. In these structures, a copy of all entries is inherently available in lower memory levels and the overheads of memory protection mechanism are as low as the parity code overheads. However, the parity-duplication approach does not offer full protection for data-SPM and write-back data cache since a fraction of data blocks in these structures does not have any copy in the lower memory levels for error correction.

Due to high error correction capability of parity-duplication approach, several studies have tried to utilize this approach for protecting the data-SPM and write-back data-cache [3][5][12][13]. To this end, these studies have proposed to provide the replica for non-duplicated data, i.e., *dirty data*. The replica of dirty data has been provided in three different ways: 1) an extra on-chip memory module is utilized to keep the replica of cache lines [13]; 2) the cache lines that may not be referred in a near future are utilized to keep the replica of other cache lines [3]; and 3) the SPM lines are utilized to keep a replica for other SPM lines [5][12]. The major drawback of these schemes is that only a subset of memory lines is duplicated meaning that a part of memory remains unprotected.

This paper proposes a duplication scheme, so called *Cache-Assisted Duplicated SPM* (CADS), to correct SEUs and SEMUs in data-SPM lines detected by low-cost error detecting code. The key idea in CADS to provide a replica for *software-managed* SPM is enforcing the *hardware-managed* cache to keep a copy of *non-cacheable* SPM lines. In particular, CADS duplicates all dirty SPM lines in cache memory considering the fact that clean SPM lines have inherently a copy in lower memory hierarchy. To this aim, we propose a cache controller circuitry that is capable to store a copy of non-cacheable SPM lines in cache memory. To reduce the energy consumption overhead of

H. Farbeh, N. S. Mirzadeh[1], N. F. Ghalaty[2], S. G. Miremadi and H. asadi are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (Email: farbeh@mehr.sharif.edu, {miremadi, asadi}@sharif.edu).

M. Fazeli is with the Department of Computer Engineering, Iran University of Science & Technology, Tehran, Iran (Email: m_fazeli@iust.ac.ir)

[1] N. S. Mirzadeh is currently a PhD Student at Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland (Email: nooshin.mirzadeh@epfl.ch).

[2] N. F. Ghalaty is currently a PhD at the Virginia Polytechnic Institute and State University (VT), Virginia, US (Email: farhady@vt.edu).

CADS, we furthure propose *Buffered-CADS* (BCADS) technique in which a mini buffer is inserted between SPM and cache to minimize the extra cache accesses for updating the replicas.

By taking advantages of the proposed cache controller circuitry, BCADS provides the following features: 1) it keeps a copy of all dirty SPM lines in the least recently-used cache lines, 2) it prevents the early eviction of SPM replicas from the cache due to cache replacement operation, 3) it releases the cache lines that contain the replicas whenever the corresponding replicas are no longer needed, 4) it overwrites the erroneous SPM lines by the error-free copy available in the cache or main memory, 5) using low-cost interleaved parity code to detect SEMUs, BCADS is able to correct all errors that are detected, and lastly 6) no modification in SPM management algorithm or target application is required.

The rest of this paper is organized as follows. In Section II, the related work is reviewed. Section III illustrates the motivation of this work. The proposed scheme is presented in Section IV. Section V explains the simulation system setup and the results are demonstrated in Section VI. In Section VII, several aspects of the proposed scheme is discussed. Finally, Section VIII concludes the paper.

II. Relater Work

Fault-tolerant schemes to protect SPM and cache memory can be broadly categorized into ECC-based and duplication-based schemes. The former schemes try to reduce the overheads of conventional ECCs or improve the correction capability of ECC codes [4][7][9]-[11][15][26][28][31]-[34]. The latter schemes keep a replica of memory contents for error recovery once an error is detected in a memory line [3][5][12][13][16][17][21][35][36]. Most of the previous studies on SPM protection are duplication-based while both duplication and ECCs have been used to protect cache memory. In the following, we first discuss the reliability improvement schemes in SPM and then explore duplication- and ECC-based schemes in cache memory.

A. SPM Reliability Enhancement

A data duplication scheme [12] uses dead blocks of SPM to keep a copy of live blocks. Dead blocks are identified at compiler level using an algorithm that analyses the patterns of accessing to SPM lines. At runtime, the redundant blocks are created if a free block exists in SPM. This scheme suffers from two major limitations. First, due to high utilization of SPM lines and shortage of dead blocks, a large fraction of SPM blocks remain unprotected. Second, updating the replica on every write operation to SPM will impose significant performance overhead.

Embedded RAIDs-on-Chip (E-RoC) scheme [5] protects on-chip distributed SPM modules in Chip Multi-Processor (CMP) systems. In this mechanism, SPM blocks are duplicated in the other SPM modules under the control of an E-RoC manager module. Significant energy overhead is imposed due to parallel accesses to SPMs and managing the SPM contents. Aggressive voltage downscaling is used to reduce this overhead, which its consequence is the exponential increase in the susceptibility of SPMs to soft errors. Considering distributed SPM in multicore processors, *In-Scrachpad Memory Replication* (ISMR) scheme [35] duplicates the active dirty SPM blocks into inactive SPM space. In this scheme, an offline profiling is performed to analyze the access patterns of SPM block. At runtime, the status of SPM blocks is determined by a tag included to all SPM lines and a dirty SPM blocks are replicated to inactive SPM blocks with the aid of a *Replica Management Unit* (RMU).

Memory-Mapped SPM (MM-SPM) scheme [21] has been introduced to protect instruction-SPM and is not applicable to data-SPM. Fault-Tolerant SPM (FTSPM) scheme [14] partitions SPM into three regions with different levels of soft error protection and maps data blocks to SPM regions according to the vulnerability of data blocks. *Data-recomputation* algorithm [22] recovers the erroneous dirty data block in SPM using its primary data elements by re-executing the instructions producing the data block.

B. Duplication Schemes in Cache

In-Cache-Replication (ICR) [3] replicates a fraction of dirty lines of data cache into lines which have not been used for a long time. *Replication Cache* [13] is based on keeping a redundant copy of dirty cache lines in a small embedded cache. *Multi-Copy Cache* (MC^2) [36] keeps multiple copies for cache lines to detect and correct process-variation induced faults in an aggressively voltage scaled cache architecture. This scheme can significantly reduce the energy consumption of the cache in embedded applications that their working set is considerably smaller than the cache size.

In *Tag Replication Buffer* (TRB) scheme [16] and *SimTag* scheme [17], tag array in cache is protected using the duplication approach. TRB scheme [17] inserts a fully-associative cache beside the main cache to keep a copy of recently-accessed tags. SimTag scheme [16] exploits the inherent similarity in tags of adjacent cache sets and considered the similar tags as the replica of each other.

C. ECC-based Schemes in Cache

Due to higher overheads of error correction in comparison to error detection, decoupling these two operations is a well-known approach to eliminate or minimize the overheads of ECCs in error-free system operation [7][28][31][32]. To reduce the latency and energy consumption of ECCs, *Punctured ECC Recovery Cache (PERC)* [7] uses a separate error detection and correction policies by using fast EDCs in cache and allocating a separate memory module for ECCs. *Memory-Mapped ECC* scheme [31] stores the ECC bits in memory hierarchy such as data, instead of dedicating SRAM cells to them. Employing different error coding schemes for clean and dirty lines of cache was introduced to protect cache lines [28]. *ECC FIFO* [32] proposed to use light error detection codes for each last-level cache line and to keep error correction codes in a FIFO structure located in off-chip DRAM.

Several studies tried to improve the detection/correction coverage of ECCs [4][9]-[11][15][26][33][34]. *PSP-cache* scheme [9] exploits the parallel access of cache lines to apply ECCs in larger data granularity. Using the same number of bits as SEC-DED, a SEC-DED-DAEC code [15] provides a higher error correction capability. The goal of the matrix-based ECC scheme [26] is to provide the capability of correcting adjacent multiple errors. The coding scheme proposed by *Ma et al.* [33]

is able to correct double random errors as well as burst errors of length three and four bits. The ECC scheme proposed by *Neale et al.* [10] has the ability of DAEC as well as scalable *Adjacent Error Detection* (xAED). The goal of the scheme presented in [11] is to maximize the probability of detecting double adjacent bit errors in SEC and triple adjacent bit errors in SEC-DED. A modified version of Hamming code [34] provides the ability to detect 2- and 3-bit burst errors in addition to correcting single bit errors. A two dimensional coding scheme [4] detects errors by parity code in cache rows and corrects errors by keeping the column XOR of data written to the cache as well as column XOR of all dirty data removed from the cache. In [8], a two dimensional cache protection method has been proposed in which multi-bit errors are corrected by interleaving data array rows among vertical parity rows.

We conclude this section by highlighting the main differences between our proposed scheme and previous data-duplication schemes. All previous data SPM duplication schemes keep the replicas in SPM, whereas we propose to keep the replicas in the cache. Keeping the replica in SPM requires complex application profiling and SPM management modification [12][35] or requires a complicated hardware module to manage the replicas [5][35]. Moreover, it imposes significant performance overhead due to reducing the SPM usable space. On the other hand, due to architectural difference between SPM and cache memory, duplication schemes for cache protection either are not applicable to SPM [3][16][17] or impose significant overheads to provide replicas for all SPM lines [13][36]. By keeping the SPM replicas in the cache, as proposed in this paper, neither extra hardware module nor modification in SPM management are required. In addition, the locality in accessing the cache and SPM, as will be explained in Section III, minimizes the impact of reducing the cache usable space on the performance of the system.

III. MOTIVATION

Until recently, SEUs were regarded as a main effect of particles strike in digital circuits. However, in today's nanoscale technology, SEMUs due to particles strike have become more probable than previous technology generations. Fig. 1 depicts the percentage of SEMU and SEU caused by particles strike for different technology feature sizes [6]. According to [6], the probability of SEMU in 65nm technology is about 20%, whereas this probability for 40nm technology has increased to 40%.

A. Shortcomings of Conventional Protection Schemes

Error correction codes, e.g., SEC-DED, are extensively used to protect data against SEUs. However, the presence of SEMUs makes these codes inefficient to be used in highly reliable systems [14]. Using more powerful error correction code, e.g., *Double Error Correction-Triple Error Detection* (DEC-TED), *Single Error Correction-Double Error Detection-Double Adjacent Error Correction* (SEC-DED-DAEC), and interleaving ECCs can effectively be used to protect the system against SEMUs [15]. Interleaving ECCs, however, imposes severe energy, area, and/or performance overheads [8] and the system still remains vulnerable to multiple errors greater than two bit-flips for DEC-TED and SEC-DED-DAEC codes.

Rapid increase of SEMU rates has made the data duplication approach as one of the most promising fault-tolerant approaches for on-chip memories. However, the huge overheads in area and energy consumption of full memory module duplication limit its application in on-chip SPM and cache memory. The main goal of previous duplication-based schemes is to minimize overheads while providing replica of high fraction of data. The main drawback of all previous duplication-based approaches is that they cannot guarantee the full-duplication of memory with an acceptable energy, area, or delay overhead [3][5][12][13][17][35].

Using write-through policy instead of write-back policy can improve the cache reliability. However, for software-managed SPM, which is in the memory address space, simultaneous writes to higher memory hierarchy is not straightforward. In addition to its huge energy and performance overheads, updating the main memory upon each SPM data update also complicates the SPM management mechanism.

To efficiently utilize the limited SPM space, this space is shared between various data blocks of the application and these blocks will be dynamically transferred to SPM on demand. As proposed in previous work [5][12][35], SPM space can be used to keep a replica of other SPM lines. However, exploiting a fraction of SPM space to keep a replica of SPM blocks not only complicates the SPM management, but also leads to significant performance overhead; moreover, it does not provide a replica for all SPM blocks if there is no enough free space in SPM.

B. A Key Observation

A wide range of the modern embedded processors such as Cortex-R Series [1], SH7785 [2], and ColdFire MCF5 [18], have employed both SPM and cache memory in their architecture. The main aim of the inclusion of both cache memory and SPM in these architectures is to enhance both predictability and performance. Cache memory is used to enhance performance by compensating the limitations of SPM software management due to dynamic behavior of the program. On the other hand, by optimally mapping of program blocks to SPM, most of the memory references would be complied with SPM; thus, the number of references to the main memory and consequently the number of cache accesses would significantly decrease compared to non-SPM processors.

Several studies illustrated that the hybrid SPM-cache architecture outperforms both pure SPM and pure cache architectures in several aspects. It was shown by *Kang et al.* [37] that employing both cache and SPM together not only improves the timing predictability of the system but also reduces the total number of cache misses. In [38], it was illustrated that the hybrid cache-SPM architecture outperforms both pure cache and pure SPM architectures in term of the execution time. By employing the hybrid SPM-cache [39][40], the total energy consumption is reduced. As reported [41], the timing predictability of hybrid SPM-cache architecture is better than that of pure cache and the performance of this hybrid architecture is better than that of pure SPM architecture. It was shown that compared with pure cache architecture, hybrid architecture reduces not only the number of cache misses, but also the total energy consumption and execution time of the applications [42]. In [43], the leakage energy of the cache is reduced by aggressively placing the cache lines into low power mode when the majority of transactions are SPM accesses. The temperature of on-chip

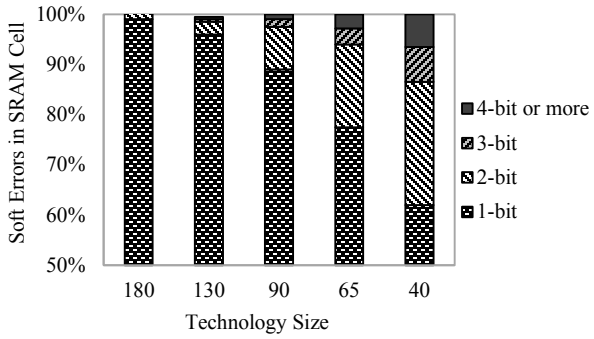


Fig. 1. Percentage of SEU and SEMU caused by particles strike in SRAM cells [6]

memories is managed by *Jia et al.* [44] via dividing the loops into several segments and alternately allocating data to the cache and SPM.

According to the principle of locality, it can be expected that there would be a locality when accessing on-chip memories. In other words, it is expected that for a specific time slot, most of the memory references are accesses to SPM (or cache) and there are small number of cache (or SPM) accesses. To verify the existence of such locality, we have carried out a set of experiments on ARM926EJ-S processor [1] to extract the number of accesses to cache and SPM in different cycles. Fig. 2 depicts the number of cache and SPM accesses during the workload execution for Rijndael benchmark [20] in a sliding window of one thousand accesses, i.e., one thousand most recent accesses sent to the memory system. The gap between the two curves indicates the locality in SPM and cache accesses; the wider gap, the more access locality is experienced. According to Fig. 2, SPM accesses contribute more than 85% of the total accesses in most of time intervals. In the time intervals that cache is accessed more frequently, SPM is less busy than the other time intervals. Our observations for workloads in SPEC CPU2006 [19] and MiBench [20] benchmark suites show that more than 80% of memory references are access to either SPM or cache for about 87% of time intervals, on average. This observation is the consequence of the software management of SPM. The SPM space is mostly allocated to data arrays accessed in loops. This leads to high utilization of SPM and low utilization of the cache in the time interval between entering and exiting these loops. The reverse utilization exists in other execution phases or inside loops that their data arrays are not mapped to SPM. The observation of the locality in accessing the cache and SPM motivates us to employ the least-frequently used lines in the cache to keep a replica for SPM lines.

Availability of a cache memory along with SPM is a promising solution for exploiting cache memory to keep a replica of dirty SPM lines. Such protection mechanism is expected to impose low energy and performance overhead as the cache is reused to protect SPM. In hybrid SPM-cache processors, the static power due to leakage current is wasted regardless the cache is accessed or not. On the other hand, the static power contributes a large fraction of the total power consumed in cache memory. Therefore, by using already available cache lines as a replica for SPM, only dynamic energy consumption overhead will be imposed to the system due to extra cache accesses. Hence, by adding no hardware to store the replicas, the static power consumption remains the same as baseline system configuration.

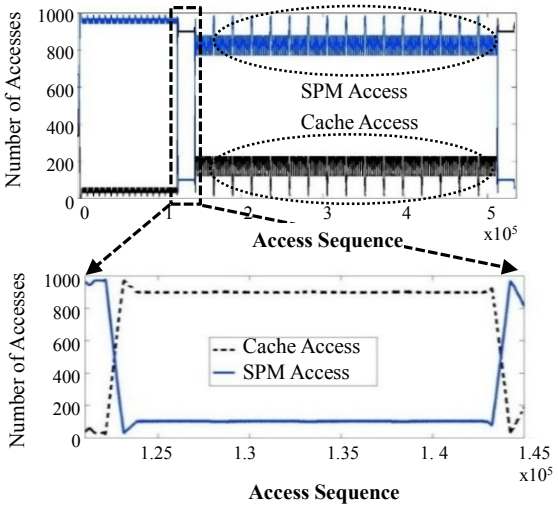


Fig. 2. Number of SPM and cache accesses in 1000 sliding window accesses

IV. Proposed CADS Scheme

The idea in proposed architecture, named *Cache-Assisted Duplicated SPM (CADS)*, is to utilize existing cache and main memory to keep a replica for all SPM data lines. To this aim, CADS considers a copy of clean SPM data in the main memory and enforces the cache to store the non-cacheable dirty SPM data. Since SPM blocks are transferred between the main memory and SPM at runtime, there is always an inherent copy of clean SPM lines in the main memory. CADS uses the resident data lines in the main memory as a copy of clean SPM lines. For dirty SPM lines, which have no valid copy in the main memory and are the major reliability concern, we propose to exploit the cache lines for replication. The SPM address space is normally non-cacheable and the cache ignores the accesses to SPM. To enforce the cache to store a copy of dirty SPM lines, we modify the cacheable detection unit. The proposed cache controller provides the ability to allocate and update the replica in the cache for SPM write accesses, to ignore the SPM read accesses, and to retrieve the replica for error recovery.

CADS employs parity code to detect errors in SPM. Error detection mechanism is also independently employed from error recovery process. This can provide two main benefits: 1) the overheads for error correction are not imposed in the normal system operation; 2) according to the degree of reliability needed for each application, an appropriate error detection mechanism can be employed, which makes this architecture applicable to a wide range of applications. To detect both SEU and SEMU, interleaved parity code is considered in this paper for error detection which has significantly lower overhead as compared to ECC codes.

The error recovery process in SPM needs to differentiate between dirty and clean data lines. Errors occurring in clean data lines can be corrected by its original copy in the main memory whereas the copy of dirty data lines is available in the cache memory. The mechanism to distinguish between clean and dirty data lines in SPM and the other aspects of the proposed architecture will be discussed in the next subsections.

A. Keeping a replica for all SPM lines

There are two approaches for allocating limited SPM space to data blocks: static allocation and dynamic allocation [12]. In the static approach, a subset of data blocks is placed in SPM at the beginning and remains there for a while [23]. In the dynamic approach, which is typically preferred for data allocation, data blocks are placed in the main memory at the beginning and will be transferred between SPM and main memory at runtime [23].

Based on the dynamic SPM allocation, to provide a full-duplication for SPM lines, we propose to store the original copy of SPM lines in main memory as a replica for clean lines and to exploit the existing cache memory to keep a replica for dirty lines. However, in conventional processor architectures, SPM and cache are located at the same level of memory hierarchy and SPM address space is not in the cacheable address space [23]; therefore, the cache controller ignores all SPM references. In CADS, the cache controller is modified to be aware of SPM accesses. To keep a replica for a dirty SPM line in the cache, when a line is written to SPM, this data line must be written in the cache, as well. This means that we propose two different caching policies in the cache for writing to SPM and reading from SPM.

Every write access to SPM is defined as a cacheable transaction in CADS; therefore, once a data line in SPM becomes dirty, a cache line is allocated to keep a replica for this new dirty SPM line. In the subsequent updates of this data line in SPM, its replica in the cache is updated as well. On the other hand, read operation does not modify the SPM contents and no cache access is needed. Thus, the conventional caching policy is employed for SPM read transactions. Consequently, in the proposed architecture, each dirty line in the SPM has a replica data line in the cache while clean data lines in SPM have their replica in the main memory.

B. Designing cacheable detection logic

To make the SPM write accesses as cacheable operations and to keep SPM read accesses as non-cacheable operations, CADS redesigns the *cacheable detection unit* of the cache controller. Conventional cacheable detection unit decides whether the address generated by the processor is cacheable or not. In CADS, the read/write signal is also checked by the cacheable detection unit. If the read/write signal indicates the write access and the address generated by the processor is in the SPM region, the cacheable detection unit activates the cacheable signal.

Fig. 3 depicts the proposed architecture and an abstract view of the modified cacheable detection unit. To define SPM write accesses as cacheable operations, a *SPM Cacheable Unit* (SCU) is added to the conventional cacheable detection unit. SCU activates its output if the address is within SPM region and the write signal is active. The output of the conventional cacheable detection unit and SCU are ORed to produce the final cacheable signal. SCU has also an input signal, so called *Error Recovery* signal, to make "SPM read" accesses as cacheable operation in SPM error recovery phase. In the next subsection, the functionality of this signal will be discussed in detail.

Fig. 4 illustrates the detailed architecture of the modified cacheable detection unit for addressing areas of a typical processor [24] that its cacheable table is according to Table I. As shown in Table I, PROM and RAM address areas are cacheable whereas I/O and SPM address areas are non-cacheable accesses.

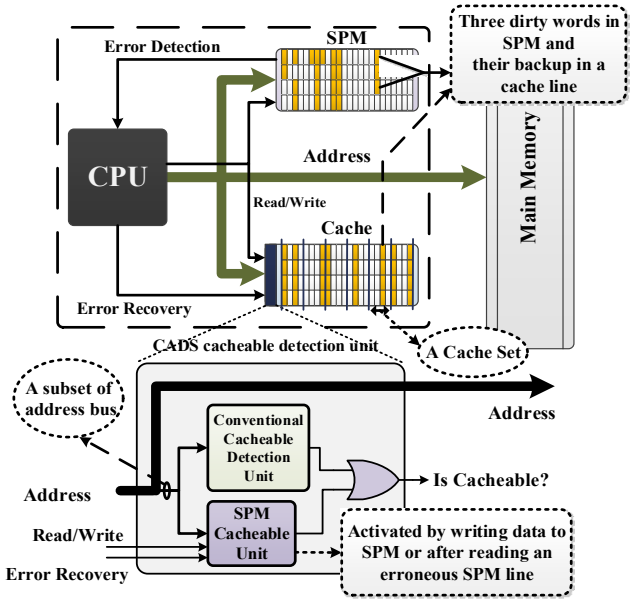


Fig. 3. Proposed CADS architecture

C. Challenges in Design and Implementation of CADS

Despite high degree of reliability provided by CADS, some challenges in design and implementation of the proposed architecture need to be addressed. In particular, the following questions should be addressed.

- How to guarantee the full-duplication of dirty SPM lines?
- How to restore error-free replicas of dirty data from cache?
- How to distinguish between clean and dirty data in SPM?

In the following, we discuss these concerns in detail.

1) Guaranteeing full duplication of dirty SPM lines

One major threat to replicate the SPM data lines in the cache is the probability of eviction of these replicas from the cache. This can possibly occur upon a cache miss. Once a replica line is evicted from the cache, SPM becomes vulnerable to soft errors as the original dirty data line in SPM has no replica any longer.

To guarantee that there would be always a replica for all dirty SPM lines, we can simply prevent the cache lines containing the replicas to be evicted. Most of the cache memories in today's embedded processors provide the ability to lock a cache line to prevent replacing a line in the cache memory [1]. By locking the cache lines that store the replicas of SPM, no replica may be evicted from the cache. It is assumed that the size of the cache is at least equal to the SPM size to guarantee the full-duplication of all SPM contents in the worst-case scenario, i.e., the situation in which all SPM lines are dirty.

Because of dynamic transferring of data lines between SPM and main memory, a dirty line in SPM may be replaced by a new line. In this case, the replica will be unlocked along with the dirty line eviction. It is noteworthy that the locking/unlocking operation is not required for every SPM transaction. The locking operation is needed only when a new cache line is allocated for SPM replica, which happens for small fraction of the writes to SPM. The unlocking operation is needed only when a

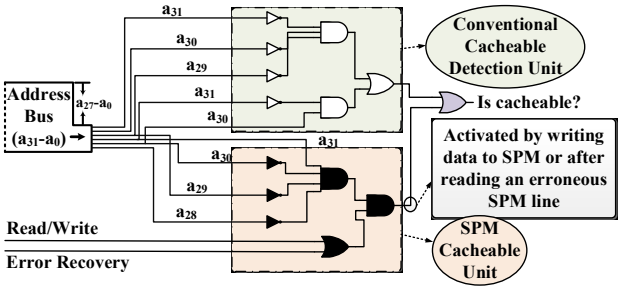


Fig. 4. Proposed cacheable detection unit to support SPM write access caching

TABLE I
CACHABLE TABLE FOR SYSTEM ADDRESSABLE AREA

Address range	Area	Cached
0x00000000 – 0x1FFFFFFF	PROM	Cacheable
0x20000000 – 0x3FFFFFFF	I/O	Non-Cacheable
0x40000000 – 0x7FFFFFFF	RAM	Cacheable
0x8E000000 – 0x8FFFFFFF	SPM	Non-Cacheable

replicated SPM block is evicted. No locking/unlocking operation is needed for read operations. For write operations into SPM, only one out of the following three cases requires the locking operation: 1) *A write operation into an unreplicated clean word:* in this case, the clean word becomes dirty and a cache line is allocated for replicating the word. The allocated cache line needs to be locked; 2) *A write operation into a dirty word:* in this case, a cache line has already been allocated to the dirty word and it is enough to update the data word in SPM and its replica in the cache. The cache line has already been locked and no further locking operation is required; 3) *A write operation into an already-replicated clean word:* A cache line allocated for replication contains the replica of eight adjacent SPM words. A cache line has already been allocated and locked for a clean SPM word that at least one of its eight adjacent words (8-1) is dirty. Since the SPM space is mostly allocated to arrays accessed in loops, the replica for the majority of clean SPM words that become dirty have already been allocated and locked (seven out of eight in the best case).

2) Restoring error-free replica of dirty data lines from cache

Upon error detection in a dirty line, the processor should read the error-free copy of the dirty line from the cache and overwrite the erroneous line of SPM by the error-free replica. Since read accesses from SPM are not cacheable operations, conventional read access cannot restore the backup copy available in the cache. To read an error-free line from the cache during error recovery, after activation of error detection signal from SPM, the SPM is disabled by inactivating SPM chip select and error recovery signal is activated in the next clock cycle. Activation of the error recovery signal makes the read operation from SPM address space as a cacheable operation in order to force the cache to send out the replica data line to the processor. After reading the error-free line from the cache by the processor, it is then written into the erroneous line in the SPM. An abstract view of corresponding signals has been shown in Fig. 3.

3) Distinguishing between clean and dirty data lines in SPM

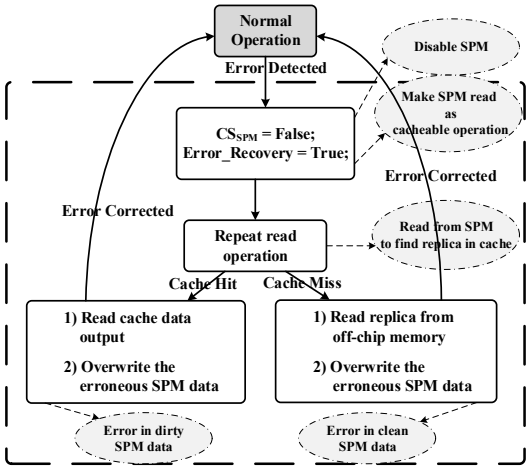


Fig. 5. Error correction procedure in CADS

A traditional way to identify the clean and dirty data is to employ a dirty flag bit for every SPM line which leads to about 3% energy and area overheads in normal system operation. We propose a solution to overcome this problem without any overhead in the normal system operation. In the proposed architecture, all dirty SPM lines are stored in the cache and hence cache contains no clean SPM line. When an error is detected, we first look up the cache to find the replica. If a cache hit occurs, the erroneous data is a dirty line and will be corrected using its replica in the cache. Otherwise, a cache miss indicates that the erroneous line is clean and the replica will be read from the main memory. Fig. 5 depicts the SPM error correction procedure.

The error correction routine is activated after detecting an error on a read operation from SPM. The activation of *Error Detection* signal is a hardware interrupt to the processor which requires immediate attention. To respond this interrupt, the processor suspends its current activities, saves its state, and executes an interrupt handler operating according to 'Error Correction Procedure' depicted in Fig. 5. To properly activate and deactivate the corresponding signals, negligible modification in the processor control unit is required.

D. Design Optimization

To replicate all dirty SPM lines in the cache, every write access to SPM needs an extra access to the cache. Extra cache accesses are imposed due to update the replicas in the cache or to allocate new cache lines for the replicas of SPM lines if they are not currently available. This leads to significant dynamic energy overhead in CADS. The energy overhead of extra accesses to memory for allocating and updating the replicas exists for all previous duplication schemes [3][5][12][13][17][35][36]. Our evaluations show that the dynamic energy overhead for updating the replica in the most related work [12] and CADS is 52% and 43%, respectively.

To overcome this overhead and reduce the number of cache accesses due to SPM line duplication, we enhance the CADS architecture by inserting a buffer between SPM and the cache. This enhanced architecture is named *Buffered-CADS* (BCADS). This buffer reduces the cache accesses by storing the replica of the most-recently updated SPM line. The size of the buffer is considered the same as the size of single cache line,

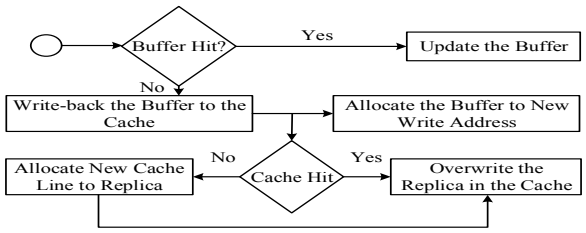


Fig. 6. Control flow of a write access to SPM in BCADS

e.g., 32-byte in our configuration. Using this buffer, we can keep the replica of eight 32-bit SPM words residing in the same cache line.

Fig. 6 depicts the control flow of a write access to SPM in BCADS. For every write operation to SPM, the write address is compared with the address of data line stored in the buffer. If the replica is already in the buffer, which can be interpreted as a buffer hit, the buffer entry will be updated. Otherwise, on a buffer miss, after writing back the buffer entry to the cache, the buffer will be allocated to the new data written to SPM. For the write-back data line from buffer to the cache, if the replica line is already allocated in the cache, the line will be rewritten. Otherwise, a new cache line is selected for replication before writing back the buffer entry.

BCADS can significantly reduce the extra cache accesses by utilizing the locality of references in SPM access. The SPM lines are mainly allocated to data arrays accessed in loops, which have a highly localized access pattern. Therefore, it is highly probable that consecutive requests for updating the replicas refer to the same cache line. By taking advantage of such locality, the buffer will be able to catch the majority of replica update requests.

V. Simulation System Setup

The proposed architecture has been evaluated by a cycle-accurate simulator called FaCSim [25], which models the ARM926EJ-S processor core and its memory subsystem [1]. The on-chip memory configuration used in our simulation consists of a D-SPM, a D-cache, and an I-cache. A 4-way set associative write-back cache with 32-byte lines is considered as D-cache. The evaluations are performed for three different cache and SPM size, i.e., 4-Kbyte, 8-Kbyte, and 16-Kbyte. We have modified the cacheable detection unit to support CADS and BCADS requirements.

Program blocks are mapped to SPM by modifying the application source code. For this purpose, the instructions required for transferring data blocks between the main memory keeping the original SPM contents and SPM are inserted in the source code. Application profiling is used to analyze the patterns of accessing data blocks and determine the blocks that are worthy to be transferred to SPM, i.e., data blocks with high temporal locality are the best candidates to be mapped to SPM. A subset of SPEC CPU2006 [19] and MiBench [20] benchmark suite are used as the target workloads.

To evaluate the proposed architecture, it has been compared with the data duplication schemes presented in [12] and [35]. As described in Section II, there are three duplication-based schemes in the literature to protect data SPM, i.e., [12], [5], and [35]. It is noteworthy that [5] and [35] targeted distributed SPMs in multicore processors, whereas [12] and BCADS provide the duplications in local memories of a core and are independent of the number of cores. In this regard, [12] is the most comparable scheme to BCADS and [35] is the most state-of-the-art scheme to protect SPM. Hereafter, the schemes in [12] and [35] are referred as Li et al. and ISMR, respectively.

The results for CADS are also included to quantitatively show why the optimization proposed in BCADS is required and how much the energy efficiency is improved. In addition, we have included the results for the well-known SEC-DED scheme, in which 32-bit data words are protected by 7-bit check bits, i.e., SEC-DED(39,32). It should be noted that comparison between data duplication schemes and ECCs may not be straightforward and fair. Because, SEC-DED and duplication are two different error correction approaches with different correction capabilities and correction mechanisms. SEC-DED is capable of correcting only single errors, whereas duplication is capable of correcting any detectable errors, regardless of the number of erroneous bits. SEC-DED is a forward error correction scheme, whereas duplication is based on a backward correction. The main reason that we discussed SEC-DED is to provide an insight about the overheads of BCADS, Li et al., and ISMR by comparing them with a non-protected SPM and a protected-SPM by a conventional ECC scheme.

For error detection in BCADS, 1-bit parity, 3-bit interleaved parity, and 4-bit interleaved parity [26] have been considered as case studies. Hereafter, these configurations are referred as BCADS-p1, BCADS-p3, and BCADS-p4, respectively. For error detection in Li et al., ISMR, and CADS, single bit parity code is considered and referred as Li et al.-p1, ISMR-p1, and CADS-p1, respectively.

To estimate the energy consumption of the SPM and cache, we use CACTI 6.5 [27] and Synopsis Design Compiler® [29] for 65nm feature size. The dynamic energy per access and static power of the cache, SPM, and the buffer are extracted from CACTI 6.5 [27]. The dynamic and static power of parity and SEC-DED codecs are extracted from Synopsis Design Compiler® [29] using Nangate 65nm Open Cell Library and added to the energy of SPM. The values reported by CACTI 6.5 [27] and Synopsis Design Compiler® [29] for 4-Kbyte cache, 4-Kbyte SPM, and the buffer are shown in Table II.

The access time of SPM, cache, and main memory is presented in Table III. It is assumed that parity code does not increase the access time of the cache and SPM while the latency of SED-DED read path is assumed to be one clock cycle. It is noteworthy that the impact of BCADS controlling logic on the latency of the cache and SPM has been taken into account. As explained in Section IV, there is no modification in the SPM logic. Meanwhile, according to Fig. 4, there is a minor modification in the cacheable detection unit of the cache controller. As illustrated in Fig. 4, the "SPM Cacheable Unit" has the same logic depth and so the same delay as "Conventional Cacheable Detection Unit". The only overhead to the logical critical path is a 2-input OR gate. The cacheable detection unit has a small contribution in the total critical path of the cache controller. We have synthesized the cache controller logic by Synopsis Design Compiler® [29] using Nangate 65nm Open Cell Library and the results show that there is no increase in the critical path of the cache access for BCADS. The main memory configuration is the default configuration used for SDRAM in FaCSim [25]. The

TABLE II
POWER AND ENERGY PARAMETERS for 4-Kbyte SPM and Cache

		Static power (mW)	Dynamic energy per access (pJ)
	Cache	6.72	42.31
	Buffer	0.05	7.75
SPM	on-protected	6.15	5.48
SPM	SEC-DED	7.27	6.73
SPM	Li et al.-p1	6.31	5.63
SPM	ISMR-p1	6.98	5.86
SPM	BCADS-p1	6.31	5.63
SPM	BCADS-p3	6.63	5.94
SPM	BCADS-p4	6.79	6.09

TABLE III
MEMORY ACCESS LATENCY (CLOCK)

SPM Latency					
Non-protected	BCADS	Li et al.	ISMR	SEC-DED Wr	SEC-DED Rd
1	1	1	1	1	2
Main Memory Sequential		Main Memory Non-sequential			Cache Latency
30		33			1

size of SDRAM is 128-Mbyte consisting of 4 banks. Each bank consists of 8096 rows and the size of each row is 4-Kbyte.

VI. SIMULATION RESULTS

A. Performance

The performance overhead imposed by the SEC-DED scheme is due to adding an extra cycle for SPM read operations. Whereas, the increase in execution time of Li et al. and ISMR is due to reducing the usable SPM space as well as the extra operations required for allocating and updating the replicas in SPM. BCADS imposes performance overhead due to reducing the useable cache space.

Fig. 7 reports the execution time for BCADS as well as Li et al. and ISMR normalized to the non-protected SPM for three different SPM and cache sizes. According to the results reported in Fig. 7(a), for a 4-Kbyte memory size, SEC-DED, Li et al., and ISMR schemes increase the average execution time by 6.1%, 5.1%, and 6.6%, respectively. The performance overhead imposed by BCADS is 0.7%, which is significantly lower than that in the other schemes. As expected, occupying cache lines for SPM replicas in BCADS resulting in reduced useable cache space has a negligible impact on the overall system performance. Whereas, the performance overhead in the scheme presented by Li et al. [12] and ISMR caused by occupying the SPM useable space is comparable to SEC-DED overhead.

For workloads that the majority of memory references are accessing to SPM, e.g., *String-Search* and *FFT*, the effect of the SEC-DED latency penalty is considerably higher than the other workloads. The high performance overhead of *Rijndael*, *String-Search*, and *Sjeng* workloads in Li et al. scheme can be due to high fraction of dirty data in SPM space and/or long residence time of dirty data; this overhead also is observed for BCADS in *Mcf* workload.

Increasing the memory size to 8-Kbyte and 16-Kbyte in Fig. 7(b) and Fig. 7(c), respectively, has negligible effect on the performance overhead of SEC-DED. This is because the overhead of SEC-DED is proportional to the memory access latency and the same latency is assumed for three memory sizes used in the evaluations. On the other hand, increasing the memory size leads to the availability of more free SPM lines for the scheme presented by Li et al. [12] as well as ISMR and more unutilized cachelines for BCADS. As a result, the performance overhead of Li et al. reduces from 5.1% in 4-Kbyte to 3.9% and 3.3% in 8-Kbyte and 16-Kbyte memories, respectively. The same trend is observed in ISMR. These reductions for BCADS are from 0.7% in 4-Kbyte to 0.5% and 0.4% in 8-K and 16-Kbyte memories, respectively.

The performance overhead of the replication schemes is mainly due to increase in the number of off-chip memory accesses. SPM-SPM replication schemes increase the number of off-chip memory accesses by reducing the usable SPM space, whereas this increase is due to increase in cache miss rate for BCADS. Fig. 8 depicts the total number of off-chip memory accesses caused by cache misses as well as transferring data blocks between SPM and the off-chip memory. On average, BCADS increases the number of off-chip memory accesses by about 5%, whereas this value for Li et al. is about 28%. Li et al. scheme increases the off-chip memory accesses by reducing the SPM usable space, which results in increasing the traffic between SPM and off-chip memory. The results for ISMR is almost the same as Li et al. due to the similarity in their policy for replication. On the other hand, BCADS has no effect on the traffic between SPM and off-chip memory but slightly increases the cache miss rate.

This observation illustrates an interesting feature of BCADS, as the only SPM-cache replication scheme, in comparison to the SPM-SPM replication schemes. The usable SPM space in SPM-SPM replication schemes as well as the usable cache space in BCADS decrease when the locality of transactions on SPM increases. This is the consequence of increase in the number of dirty SPM lines. In other words, the higher need for usable SPM space due to increase in the locality of transactions on SPM, the lower usable SPM space is available in SPM-SPM schemes, i.e., ISMR and Li et al. This leads to higher performance overhead when the dirty SPM lines increase. On the other hand, the higher need for usable SPM space, the lower usable cache space is needed and available in SPM-cache scheme, i.e., BCADS. Therefore, there is a balance between the usable cache lines and the fraction of transactions referred to the cache in BCADS for most of the program execution phases.

B. Energy Consumption

There are two sources of energy overhead in the SEC-DED code: 1) the energy consumed by the SEC-DED combinational circuit for error detection and correction, and 2) the energy consumed by the redundant SEC-DED bits. Li et al. scheme imposes three sources of energy overhead: 1) parity combinational circuit for error detection, 2) redundant parity bits, and 3) extra operations for each SPM write operation. Finally, for BCADS there are also three sources for increasing the energy consumption: 1) parity combinational circuit for error detection, 2) redundant parity bits, and 3) extra accesses to the buffer and cache for SPM write operations.

Energy consumption can be divided into static energy and dynamic energy. The static energy overheads of SEC-DED, Li et al., ISMR, CADS, and BCADS are only proportional to the extra hardware components added by these three protection

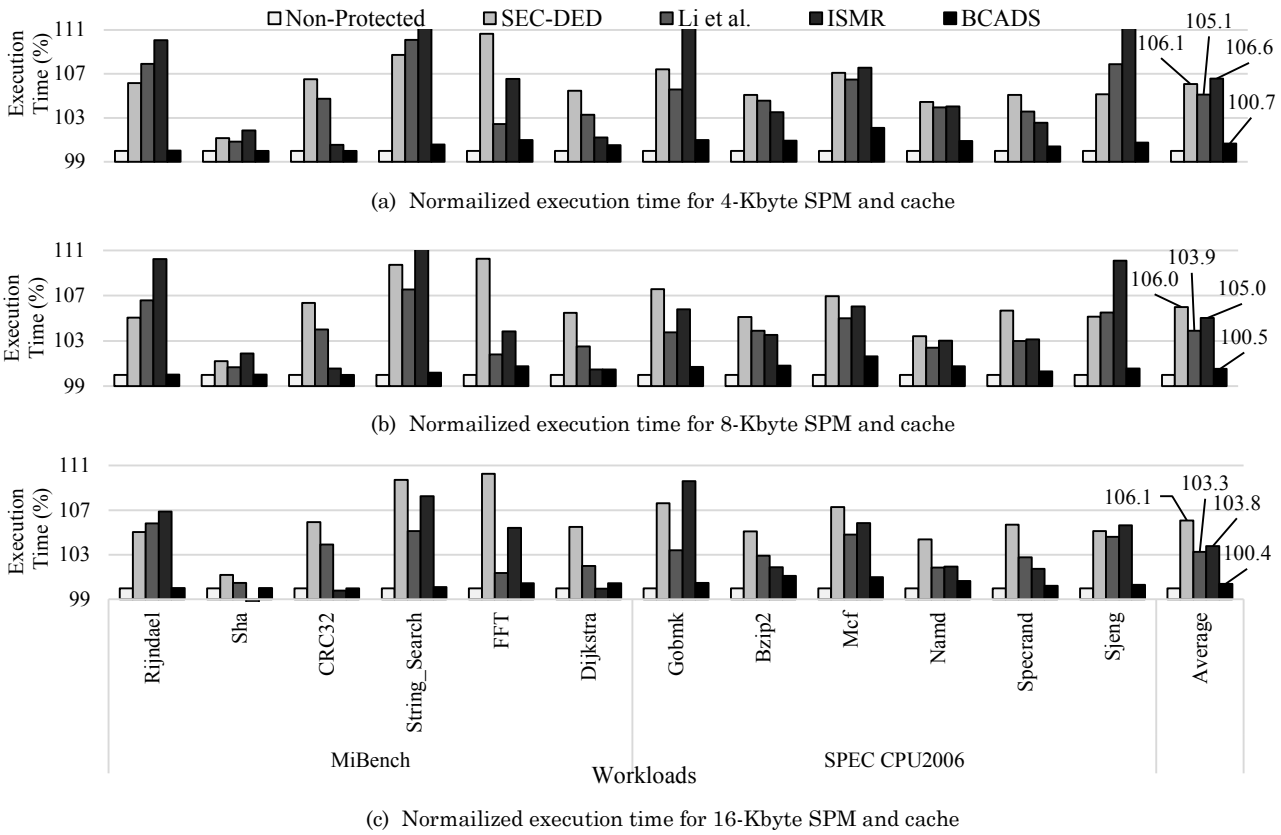


Fig. 7. Normailized execution time for SEC-DED, Li et al, ISMR, and BCADS for three different memory sizes of cache and SPM (4-Kbyte, 8-Kbyte, and 16-Kbyte)

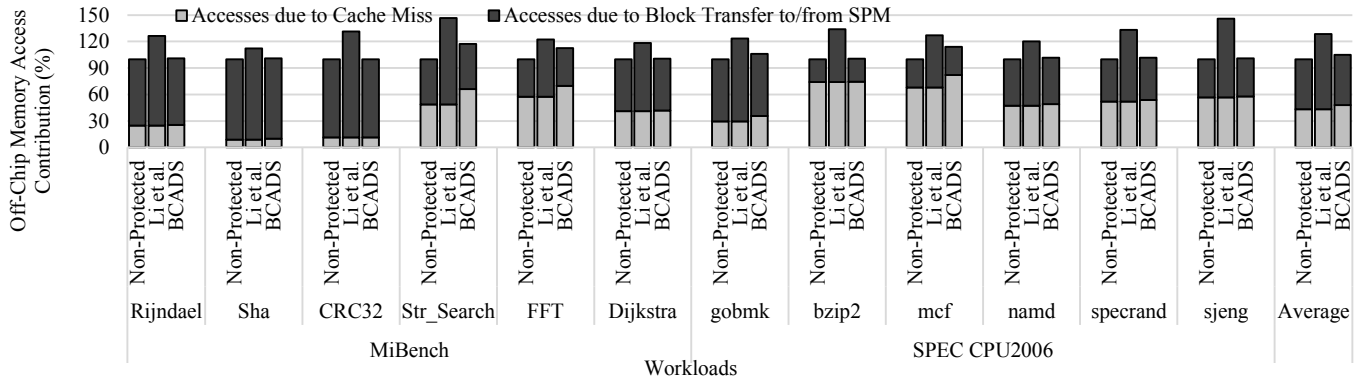


Fig. 8. Total number of off-chip memory accesses for Li et al. and BCADS normalized to non-protected SPM

schemes. The calculation of the dynamic energy consumption requires the dynamic energy per access of each component and the total number of accesses to each component. We use Equation (1) to compute static energy in BCADS.

$$Static\ Energy = [(Cache)_{Static\ Power} + (SPM)_{Static\ Power} + (Buffer)_{Static\ Power}] \times (Execution\ Time) \quad (1)$$

The term *'(Buffer)$_{Static\ Power}$'* is removed from Equation (1) when calculating the static energy consumption in the baseline, SEC-DED, Li et al.-p1, ISMR-p1, and CADS-p1 schemes.

The dynamic energy consumption in BCADS is calculated according to Equation (2):

$$Dynamic\ Energy = (Cache)_{Energy\ per\ Access} \times Total\ Number\ of\ cache\ Accesses + (SPM)_{Energy\ per\ Access} \times Total\ Number\ of\ SPM\ Accesses + (Buffer)_{Energy\ per\ Access} \times Total\ Number\ of\ buffer\ Accesses \quad (2)$$

The term *'(Buffer)$_{Energy\ per\ Access}$'* is removed from Equation (2) for calculating the dynamic energy consumption in the baseline, SEC-DED, Li et al., ISMR, and CADS schemes.

The total energy consumption in each scheme is the sum of the static energy and dynamic energy calculated in Equation (1) and Equation (2). Fig. 9 depicts the energy consumption of the evaluated schemes for 4-Kbyte, 8-Kbyte, and 16-Kbyte memory size, normalized to a non-protected SPM. The energy consumption of CADS-p1 is also presented to illustrate how the buffer in BCADS can reduce energy overhead of the proposed method. Considering 4-Kbyte memories, the energy consumption overhead of the SEC-DED, Li et al.-p1, and ISMR-p1 are 12.8%, 16.5%, and 23.8%, respectively. The energy consumption overhead for CADS-p1 is 19.4% and it significantly re-

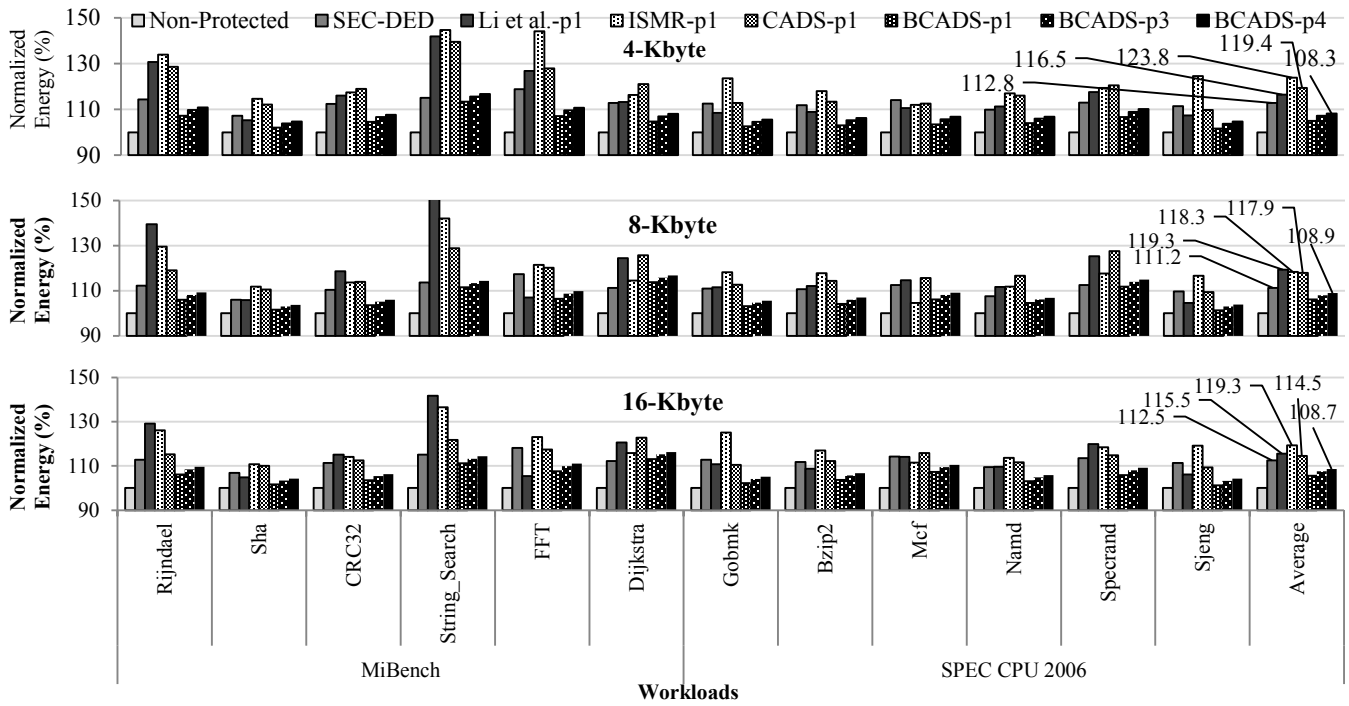


Fig. 9. Normalized energy consumption of SEC-DED, Li et al-p1, ISMR-p1, CADS-p1, and three configurations of BCADS (BCADS-p1, BCADS-p3, and BCADS-p4) for 4-Kbyte, 8-Kbyte, and 16-Kbyte SPM and cache

duces to 5.0% after enhancing the scheme to BCADS-p1. Considering higher error protection capability, this overhead in BCADS-p3 and BCADS-p4 is 7.2%, 8.3%, respectively. According to Fig. 9, BCADS-p1 imposes significantly lower energy overhead than ISMR-p1, Li et al.-p1, and SEC-DED with the same error correction capability. On the other hand, BCADS-p4 consumes 4.5% less energy than the SEC-DED code, meanwhile it provides much higher error correction capability than SEC-DED.

As observed in the experiments, BCADS significantly reduces the energy consumption overhead of CADS by decreasing the number of accesses to the cache for SPM replication. The evaluations show that about 79% of requests for SPM data replication are responded by the buffer without the cache interaction in 4-Kbyte memories. Considering the total dynamic energy consumption in BCADS, the contribution of SPM writes and reads are 10.2% and 26.7%, whereas the contribution of cache writes and reads are 14.7% and 31.9%, respectively. The buffer consumes 1.9% of the total dynamic energy and the remaining 14.6% is mainly due to accessing the cache for data replications.

Considering 8-Kbyte memories, the average of energy consumption overhead in BCADS-p4 is 8.9%, whereas this value for SEC-DED, Li et al., ISMR, and CADS are 11.2%, 19.3%, 18.3%, and 17.9%, respectively. The energy consumption overheads in 16-Kbyte memories for SEC-DED, Li et al., ISMR, and CADS are 12.5%, 15.5%, 19.3%, and 14.5%, respectively. This value for BCADS is 8.7%.

As another comnparison metric, Energy-Delay Product (EDP) overhead in 4-Kbyte SPM and cache for SEC-DED, Li et al.-p1, and ISMR-p1 is 19.7%, 22.6%, and 32.2%, respectively. EDP overhead for CADS-p1, BCADS-p1, BCADS-p3, and BCADS-p4 is 11.3%, 5.8%, 7.9%, and 9.0%, respectively. Considering 8-Kbyte memories, EDP overhead in BCADS-p4 is 9.5% on average, whereas the EDP overheads in SEC-DED, Li et al.-p1, ISMR-p1, and CADS-p1 are 17.9%, 25.4%, 24.5%, and 11.3%, respectively. The EDP overhead in SEC-DED, Li et al.-p1, ISMR-p1, and CADS-p1 in 16-Kbyte memories are 19.4%, 19.4%, 24.0%, and 11.3%, respectively. Whereas, the EDP overhead in BCADS-p4 is 9.1% which is significantly lower than that in the other schemes.

D. Area Overhead

BCADS does not impose a considerable area overhead because it is mainly based on hardware reuse in the system. There are three sources of area overhead in BCADS: 1) *Cacheable Detection Unit*: modification of cacheable detection unit is limited to addition of a few numbers of logic gates and it is negligible. 2) *Parity Bits and Parity Generator/Checker Circuit*: the main source of area overhead in BCADS is the area overhead of its error detection mechanism. 3) *The inserted buffer to reduce the cache accesses*: This overhead is less than 1%.

The area overhead of Li et al.-p1, ISMR-p1, BCADS-p1, BCADS-p3, BCADS-p4, and SEC-DED is about 3%, 8.1%, 3.4%, 9.4%, 12.9%, and 22%, respectively. This means that the area overhead of BCADS-p1 with the same error correction capability to SEC-DED is about 19% less than that of SEC-DED.

E. Reliability Analysis

Two reliability threats in data duplication schemes are: 1) the replica is not available; 2) the replica is erroneous once it is needed for error correction. The unavailability of replica is a severe reliability threat for partial-duplication schemes, as mentioned in Section III. Whereas, the main feature of BCADS is to guarantee the full-duplication, which resolves the first threat. Therefore, we need to focus on the second reliability threat in BCADS, i.e., the probability of occurring the error in both original data and its replica.

We use the Markov chain model to analyze the reliability of SPM in BCADS. It is assumed that the failure of the system in BCADS is when both replica in cache and the original line in SPM are erroneous. In addition, the failure of non-protected baseline SPM is when an error occurs in a SPM line, which can only be detected and not corrected. Therefore, the system is operational in BCADS as long as the original data in SPM or its replica in cache is error-free; and, it is operational in the baseline SPM as long as the original data in SPM is error-free. Accordingly, the Markov chain of a SPM line in the baseline SPM and BCADS are as shown in Fig. 10 (a) and Fig. 10(b), respectively. Table IV shows the description of the notations used in Fig. 10. The failure state are (D') and (D', R') in Fig. 10(a) and Fig. 10(b), respectively.

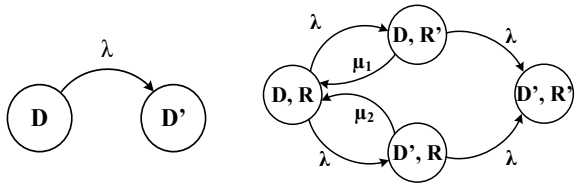
In Fig. 10(b), there is a transition from (D, R') to (D, R). This transition occurs when the original data in SPM is updated due to a write access or replacement. In this case, the replica will be updated to new error-free version. The transition from (D', R) to (D, R) occurs in both read and write accesses to the data. On a write access, the data and its replica will be updated to new error-free version, regardless of their current status. In a read access, on the other hand, the error in the erroneous data will be detected by the error detection mechanism and it will be corrected using the replica according to the procedure illustrated in Fig. 5. The system fails when it enters to (D', R') from one of the states (D, R') or (D', R).

For reliability analysis, we use the *Mean Time To Failure* (MTTF) parameter. To calculate MTTF of BCADS and the baseline SPM, we need to assign values to λ, μ_1, and μ_2 parameters. The value of λ is proportional to the intrinsic SEU rate in SRAM cells, which is 1,150 SEUs per 10^9 hours for 1Mbit memory [30]. For a 32-bit SPM word, λ is about 3.68×10^{-11} per hour. For μ_1 we assume a wide range from 10^{-6} accesses per hour, which is extremely pessimistic, to 10^6 accesses per hour, which is not far from reality. We typically assume μ_2, i.e., rate of read from a SPM line, is 2x larger than μ_1, i.e., rate of write/replacement in SPM.

Fig. 11 illustrates MTTF of both BCADS and baseline SPM. For μ_1 values around 10^6, MTTF of a SPM line in BCADS is about 16 orders of magnitude higher than that of normal SPM. This improvement in MTTF for extremely pessimistic value of 10^{-6} for μ_1 is more than 4 orders of magnitude.

The Markov chain models in Fig. 11 and the reported MTTFs are for one SPM line. The SPM is reliable as long as all of its lines are error-free. Therefore, to extend the reliability analysis from one SPM line to all SPM lines, it is enough to consider SPM as a system with N units arranges in *series* configuration. In this series system, a failure of any unit results in the failure of the system.

It is worth noting that in the reliability analysis as well as the evaluations, no error correction scheme is assumed in the cache. This assumption is necessary when evaluating the overheads of SPM protection schemes in the hybrid SPM-cache architecture. Considering a protection mechanism for the cache affects the performance and energy consumption overheads of all schemes and biases the results. However, the reliability analysis and the Markov model are still valid when considering some protection mechanism for the cache.



(a) Baseline (b) BCADS

Fig. 10. Markov chain model of a SPM line in (a) baseline and (b) BCADS

TABLE IV
DESCRIPTION OF NOTATIONS USED IN MARKOV CHAIN MODELS OF FIG. 10

Notation	Description
D	Original data is error-free
R	Replica is error-free
D′	Original data is erroneous
R′	Replica is erroneous
λ	Soft error rate in a 32-bit word
μ_1	Rate of write or replacement access to a data word in SPM
μ_2	Rate of read access to a data word in SPM

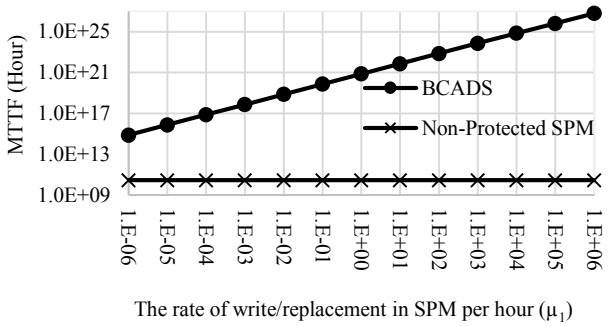


Fig. 11. MTTF of a SPM line in BCADS and baseline non-protected SPM

VII. DISCUSSIONS

In this section, we provide a comparative discussion on the proposed hybrid SPM-to-cache duplication scheme and the available SPM-to-SPM duplication schemes. We further discuss several aspects of the proposed scheme.

A. SPM-to-SPM vs. SPM-to-Cache replication

Considering SPM-to-SPM duplication schemes, exploiting the already available SPM to keep the replica of SPM contents complicates the replication process and imposes significant overheads due to several challenges. First, when the replication is managed by the software, as in Li et al [12], complex profiling, analysis, and algorithms are required to identify and allocate free spaces to replicas of new data and finding the replica of already replicated data. When the replication is managed by the hardware, as in E-RoC [5] and ISMR [35], a complex manager unit is required to monitor the SPM contents and perform the replication process. Second, updating the replica is performed sequentially after the operation of allocating free space to the replica or finding the replica location. This imposes performance overhead. Third, SPM-to-SPM schemes cannot guarantee the full-replication when more than half of the SPM space is dirty. Forth, the locality of accesses to SPM and cache, which is the motivation of this work, has not been reported for SPM-SPM accesses.

When replicating in the cache, as proposed in BCADS, all operations needed for replication, including the allocation of free space to the replicas, finding the replicas, and updating the replicas, are performed inherently by one normal cache access.

From another point of view, the main source of performance overhead in data duplication schemes is the reduction in the usable memory space. The usable memory space decreases when the number of dirty lines increases. In the best case, there is no dirty data and no replication is required. In this case, the replication schemes impose no performance overhead. In the worst case, SPM is fully utilized and all data lines are dirty, which may happen in write intensive phases of the workloads.

To guarantee the full duplication of dirty SPM lines, BCADS allocate all cache lines to replicas in the worst case and no usable cache space remains until the eviction of some dirty lines from SPM. Due to the locality in accessing the cache and SPM, the utilization of the cache is expected to be extremely low and the unavailability of usable cache lines in these phases has not significant effect on the performance of the system. On the other hand, the full duplication of SPM dirty lines in SPM-to-SPM schemes requires to keep the number of SPM dirty lines lower than 50% of total SPM space. In other words, the usable SPM space is not more than 50% in a scenario that 100% utilization of SPM is required. This underutilization of SPM when high utilization is required leads to diverting a large fraction of SPM transactions to the off-chip memory in SPM-to-SPM schemes resulting in extra latency and energy consumption.

B. Scability of BCADS in Multicore and Multi-Threaded Programs

The proposed BCADS architecture is extendable to multicore processors. In a multicore processor each core has a local cache as well as a local SPM and the replication in each core is performed independently without any interference with the other cores. The replicas of SPM in each core are stored in its corresponding cache. The cache coherence protocols, however, need to be adapted to support the replication process in BCADS. When a new cache line is allocated to replica, this replication should be interpreted as a cache line invalidation and the corresponding operations should be performed across the CPU cores. The cache coherence protocol treats the replica lines in the cache as invalid lines and ignores the write accesses to these lines triggered by updating the SPM contents.

BCADS is also applicable to multi-threaded programs and concurrent applications. In a multi-threaded program, each thread has its own view of the SPM space. Since each thread requires its own data set to be transferred to SPM, the conflicts among the threads to use the limited SPM space is managed by the SPM mapping algorithm. This is not the concern of BCADS. For a thread running concurrently with other threads, four scenarios can happen. 1) The thread has no data in SPM and the majority of SPM space is filled with data blocks of other threads. In this case, the usable space of the cache is decreased. 2) The thread has no data in SPM and there is no data block from other concurrent threads in SPM. In this case, no replication is required and BCADS has no effect on the thread. 3) A part of the SPM space is allocated to the thread and there is no data block from other concurrent threads in SPM. In this case, the replicas of this thread reduce the usable cache space for other threads. 4) A part of the SPM space is allocated to the thread and the remaining SPM space is allocated to other threads. In this case, each thread that has dirty data blocks in SPM occupies a fraction of cache lines for replication.

It is noteworthy that all of the abovementioned scenarios can happen in different phases of the programs execution. The overheads of the replication depend on the frequency and duration of each scenario. In general, the overheads of replication mostly depend on the fraction of dirty data in the total SPM space and the utilization of the cache.

C. Efficiency of BCADS in Multi-Tasking Systems

On a context switch, two situations may occur based on the SPM mapping algorithm. 1) The data blocks of the first process are transferred back to main memory on the context switch. In this situation, the corresponding cache lines containing the replicas are unlocked and the replication of the first process has no effect on the second process. 2) The data blocks of the first process remain in SPM during the execution of the second process. In this situation, the replicas of the first process reduce the usable cache lines for the next process. This situation happens only when the SPM is allocated to the first process and the other processes have no data block to be transferred to SPM. Since all processes try to exploit the limited SPM space on their own turn, the first situation is more probable in reality.

D. Comparision of BCADS with Other Schemes

We conclude this section by providing a comparison of duplication schemes to have an insight about the capabilities and overheads of various data duplication schemes. This comparison is presented in Table V. Duplication schemes have been classified into cache duplication and SPM duplication schemes. All previous schemes in SPM duplication keep the replica in SPM or main memory, while all previous schemes in cache duplication keep the replica in the cache, an extra cache, and/or main memory. The major feature of CADS/BCADS that makes it distinguished is to open a way to utilize the cache to keep the replica of SPM. According to our evaluations and Table V, it can be concluded that utilizing the cache is a better choice in comparison to previous solutions for SPM replication.

VIII. Conclusions

SPM is one of the most vulnerable parts in embedded processors to soft errors. Conventional error correction codes have severe limitations to be employed in embedded processors due to their overheads on energy consumption, performance, and area. Moreover, increasing the contribution of SEMUs in system failure caused by soft errors has resulted in more restriction on ECCs applicability. The main aim of the proposed architecture, called BCADS, is to protect SPM against SEUs and SEMUs and to provide high error correction capability with negligible performance loss. The simulation results show that BCADS provides significant reliability improvement with negligible area overhead (less than 1%) for error correction and 10% less area overhead for error detection as compared to the SEC-DED scheme. The performance loss and EDP overhead of BCADS are 0.7% and 5.8%, respectively; while the performance loss and EDP overhead of previous SPM protection scheme are 5.1% and 22.6%, respectively. Ability to correct all detectable errors, the independence of error detection from error correction

TABLE V
COMPARISON OF DUPLICATION SCHEMES IN CACHE AND SPM

	Duplication Schemes	Replica location	Replication capability	Performance overhead	Energy overhead	Area overhead	Hardware modification	Software modification
SPM replication	Li et al. [12]	SPM	Partial	5.1%	16.5%	0.0%	No	Yes
	ISMR [35]	SPM	Full	6.6%	23.8%	8.1%	Yes	Yes
	MM-SPM [21]*	Main Memory	Full	0.5%	<1%	0.0%	No	Yes
	E-RoC [5]**	SPM	Full	39%	184%	15%	Yes	No
	Proposed CADS	Cache	Full	0.7%	19.4%	<1%	Yes	No
	Proposed BCADS	Cache	Full	0.7%	5.0%	<1%	Yes	No
Cache replication	R-Cache [13]	Extra Cache	Partial	0.0%	5.6%	1.5%	Yes	No
	TRB [17]	Extra Cache	Partial	0.0%	19.9%	16.3%	Yes	No
	ICR [3]	Cache	Partial	21.0%	26.9%	<1%	Yes	No
	SimTag [16]	Cache	Partial	0.0%	2.9%	<1%	Yes	No
	MC^2[36]***	Cache	Full	3.5%	-59.5%	0%	Yes	No

*: Only instruction-SPM (not applicable to data-SPM) **: Without aggressive voltage scaling ***: With aggressive voltage scaling

scheme, and no requirement for software modification makes the proposed architecture an efficient and scalable fault-tolerant mechanism for a wide range of SPM-based embedded applications from handheld devices to safety-critical systems.

REFERENCES

[1] (2012) The ARM website. [Online]. Available: http://www.arm.com/ products/processors/classic/arm11/index.php

[2] (2012) The Renesas Website. [Online]. Available: http://www. renesas.com/products/mpumcu/superh/sh7780/sh7785/index.jsp.

[3] W. Zhang, S. Gurumurthi, M. Kandemir, and A. Siavasubramaniam, "ICR: In-Cache Replication for Enhancing Data Cache Reliability," International Conference on Dependable Systems and Networks (DSN), Jun. 2003, pp. 291-300.

[4] M. Manoochehri, M. Annavaram, and M. Dubois, "CPPC: Correctable Parity Protected Cache," International Symposium on Computer Architecture (ISCA), Jun. 2011, pp. 223-234.

[5] L. A. D. Bathen and N. D. Dutt, "Embedded RAIDs-on-Chip for Bus-based Chip-Multiprocessors," ACM TECS, vol. 13, no. 4, pp. 83:1-83:36, 2014.

[6] A. Dixit and A. Wood, "The Impact of New Technology on Soft Error Rates," International Reliability Physics Symposium (IRPS), Apr. 2011, pp. 486-492.

[7] N. N. Sadler and D. J. Sorin, "Choosing an Error Protection Scheme for a Microprocessor's L1 Data Cache," International Conference on Computer Design (ICCD), Oct. 2006, pp. 499-505.

[8] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe, "Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding," International Symposium on Microarchitecture (Micro), Dec. 2007, pp.197-209.

[9] H. Farbeh and S. G. Miremadi, "PSP-Cache: A Low-Cost Fault-Tolerant Cache Memory Architecture," Design, Automation & Test in Europe (DATE), Mar. 2014.

[10] A. Neale and M. Sachdev, "A New SEC-DED Error Correction Code Subclass for Adjacent MBU Tolerance in Embedded Memory," IEEE TDMR, vol. 13, no. 1, pp. 223-230, 2013.

[11] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro, "Enhanced Detection of Double and Triple Adjacent Errors in Hamming Codes through Selective Bit Placement," IEEE TDMR, vol. 12, no. 2, pp. 357-362, 2012.

[12] F. Li, G. Chen, and M. Kandimer, "Improving Scratch-Pad Memory Reliability through Compiler-Guided Data Block Duplication," International Conference on Computer-Aided Design (ICCAD), Nov. 2005, pp. 1002-1005.

[13] W. Zhang, "Replication Cache: A Small Fully Associative Cache to Improve Data Cache Reliability," IEEE TC, vol. 54, no. 12, pp. 1547-1555, 2005.

[14] A. M. H. Monazzah, H. Farbeh, S. G. Miremadi, M. Fazeli, and H. Asadi, "FTSPM: A Fault-Tolerant ScratchPad Memory," International Conference on Dependable Systems and Networks (DSN), Jun. 2013, pp. 1-10.

[15] A. Dutta and N. A. Touba, "Multiple Bit Upset Tolerant Memory using a Selective Cycle Avoidance based SEC-DED-DAEC Code," VLSI Test Symposium (VTS), May 2007, pp. 349-354.

[16] J. Hong, J. Kim, and S. Kim, "Exploiting Same Tag Bits to Improve the Reliability of the Cache Memories," IEEE TVLSI, vol. 23, no. 2, pp. 254-265, 2015.

[17] S. Wang, J. Hu, and S. G. Ziavras, "Replicating Tag Entries for Reliability Enhancement in Cache Tag Arrays," IEEE TVLSI, vol. 20, no. 4, pp. 643-654, 2012.

[18] (2012) The Freescale Website. [Online]. Available: http://www.freescale.com/webapp/sps/site/homepage.jsp?code=PC68KCF/

[19] SPEC CPU2006, Standard Performance Evaluation Corporation, http://www.spec.org

[20] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," Annual Workshop on Workload Characterization, Dec. 2001, pp. 3-14

[21] H. Farbeh, M. Fazeli, F. Khosravi, and S. G. Miremadi, "Memory Mapped SPM: Protecting Instruction Scratchpad Memory in Embedded Systems against Soft Errors," European Dependable Computing Conference (EDCC), May 2012, pp. 218-226.

[22] H. Sayyadi, H. Farbeh, A. M. H. Monazzah, and S. G. Miremadi, "A Data Recomputation Approach for Reliability Improvement of Scratchpad Memory in Embedded Systems," International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Oct. 2014, pp. 228-233.

[23] P. R. Panda, N. D. Dutt, and A. Nicolau, "Efficient Utilization of Scratch-pad Memory in Embedded Processor Applications," European Design and Test Conference (ED&TC), Mar. 1997, pp. 7-11.

[24] Gaisler Res., Leon2 Processor User's Manual, Ver. 1.0.30, XST ed., Goteborg, Sweden, Jul.2005.

[25] J. Lee, J. Kim, C. Jang, S. Kim, B. Egger, K. Kim, and S. Han, "FaCSim: A Fast and Cycle-Accurate Architecture Simulator for Embedded Systems," Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), Jun. 2008, pp. 89-100.

[26] C. A. Argyrides, P. Reviriego, D. K. Pradhan, and J. A. Maestro, "Matrix-based Codes for Adjacent Error Correction," IEEE TNS, vol. 57, no. 4, pp. 2106-2111, 2010.

[27] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "Hp labs cacti v5.3," CACTI 5.1, HP Laboratories, Tech. Rep. HPL-2008-20, 2008.

[28] L. Li, V. Degalahal, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Soft Error and Energy Consumption Interactions: A Data Cache Perspective," International Symposium on Low Power Electronics and Design (ISLPED), Aug. 2004, pp. 132-137.

[29] Synopsys Design Compiler, Synopsys Inc.

[30] Semiconductor Industries Association, International Technology Roadmap for Semiconductors (ITRS), 2007.

[31] D. H. Yoon and M. Erez, "Memory Mapped ECC: Low-cost Error Protection for Last Level Caches," International Symposium on Computer Architecture (ISCA), 2009, pp. 116-127.

[32] D. H. Yoon and M. Erez, "Flexible Cache Error Protection using an ECC FIFO," Conference on High Performance Computing Networking, Storage and Analysis (SC), Nov. 2009, pp 1-12.

[33] W. Ma, X. Cui, and C. L. Lee, "Enhanced Error Correction against Multiple-bit-upset based on BCH Code for SRAM," International Conference on ASIC (ASICON), Oct. 2013, pp. 1-4.

[34] L. J. Saiz-Adalid, P. Gil, J. Baraza-Calvo, J. C. Ruiz, D. Gil-Tomás, and J. Gracia-Morán, "Modified Hamming Codes to Enhance Short Burst Error Detection in Semiconductor Memories," European Dependable Computing Conference (EDCC), May 2014, pp. 62-65.

[35] L. Delshadtehrani, H. Farbeh, and S.G. Miremadi, "In-Scratchpad Memory Replication: Protecting Scratchpad Memories in Multicore Embedded Systems against Soft Errors," ACM TODAES, vol. 20, no. 4, pp. 61:1-61:28, 2015.

[36] A. Chakraborty, H. Homayoun, A. Khajeh, N. Dutt, A. Eltawil, and F. Kurdahi, "E< MC2: Less Energy through Multi-copy Cache," International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), Oct. 2010, pp. 237-246.

[37] S. Kang and A.G. Dean, "Leveraging both Data Cache and Scratchpad Memory through Synergetic Data Allocation," Real-Time and Embedded Technology and Applications Symposium (RTAS), Apr. 2012, pp. 119-128.

[38] G. Wang, L. Ju, Z. Jia, and X. Li, "Data Allocation for Embedded Systems with Hybrid On-Chip Scratchpad and Caches," International Conference on High Performance Computing and Communications (HPCC), Nov. 2013, pp. 366-373.
[39] L. Wu, Y. Ding, W. Zhang, "Characterizing Energy Consumption of Real-Time and Media Benchmarks on Hybrid SPM-Caches," International Conference on High Performance Computing and Communications (HPCC), Aug. 2014, pp. 526-533.
[40] W. Zhang and L. Wu, "Exploiting Hybrid SPM-Cache Architectures to Reduce Energy Consumption for Embedded Computing," International Conference on High Performance Computing and Communications (HPCC), Aug. 2014, pp. 340-347.
[41] W. Zhang and Y. Ding, "Hybrid SPM-Cache Architectures to Achieve High Time Predictability and Performance," International Conference on Application-Specific Systems, Architectures and Processors (ASAP), Jun. 2013, pp. 297-304.
[42] Z. Zhou, J. Lei, J. Zhiping, and L. Xin, "Fast and Accurate Code Placement of Embedded Software for Hybrid On-Chip Memory Architecture," International Conference on High Performance Computing and Communications (HPCC), Aug. 2014, pp. 1008-1015.
[43] H. Wen and W. Zhang, "Reducing Cache Leakage Energy for Hybrid SPM-Cache Architectures," International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), Oct. 2014, pp. 1-9.
[44] Z. Jia, Y. Li, Y. Wang, M. Wang, and Z. Shao, "Temperature-aware Data Allocation for Embedded Systems with Cache and Scratchpad Memory," ACM TECS, vol 14, no. 2, pp. 30:1-30:24, 2015.

Hamed Farbeh (S12) received the B.S. and M.S degrees in computer engineering from Sharif University of Technology (SUT), Tehran, Iran, in 2009 and 2011, respectively. He is currently pursuing the Ph.D. degree in computer engineering at SUT. He is the member of Dependable Systems Laboratory from 2007. He was with Embedded Computing Laboratory at KAIST University, Korea, as a visiting researcher from October 2014 to May 2015. His research interests include fault-tolerant embedded system design, reliable memory hierarchy, and reliability challenges in emerging memory technologies.

Nooshin Sadat Mirzadeh received the B.S. degree in computer engineering from SUT, Tehran, Iran, in 2013. She is currently pursuing the Ph.D. degree in Computer and Communication Sciences at École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. Her research interests include high performance energy-efficient computer architecture, memory systems including 3D integration, and near-memory processing.

Nahid Farhady Ghalaty received the M.S degree in computer engineering from SUT, Tehran, Iran, in 2011. She is currently pursuing the Ph.D. degree in electrical and computer engineering at Virginia Tech, Blacksubrg, VA, USA. Her research interests include security, reliability & performance issues in embedded systems and hardware software codesign.

Seyed Ghassem Miremadi (SM07) is a Professor of Computer Engineering at Sharif University of Technology. As fault-tolerant computing is his specialty, he initiated the "Dependable Systems Laboratory" at Sharif University in 1996 and has chaired the Laboratory since then. The research laboratory has participated in several research projects which have led to several scientific articles and conference papers. Dr. Miremadi and his group have done research in Physical, Simulation-Based and Software-Implemented Fault Injection, Dependability Evaluation Using HDL Models, Fault-Tolerant Embedded Systems, Fault-Tolerant NoCs, Fault-Tolerant Real-Time Systems, and Fault-Tolerant Storage Systems. He was the Education Director (1997-1998), the Head (1998-2002), the Research Director (2002-2006), and the Director of the Hardware Group (2009-2010) of Computer Engineering Department at Sharif University. During 2003 to 2010, he was the Director of the Information Technology Program at Sharif International Campus in Kish Island. From 2010 to 2012, Dr. Miremadi was the Vice-President of Academic Affairs (VPAA) of Sharif University. Since 2014, he is VPAA of Sharif University. He served as the general co-chair of the 13th Int'l CSI Computer Conference (CSICC 2008), the executive chair of the 2013 Int'l Conference on Engineering Education, and the general co-chair of the 2015 Int'l CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST 2015). He is currently the Editor of the Scientia Transactions on Computer Science and Engineering. Dr. Miremadi got his M.Sc. in Applied Physics and Electrical Engineering from Linköping Institute of Technology and his Ph.D. in Computer Engineering from Chalmers University of Technology, Sweden. He is a senior member of the IEEE Computer Society and IEEE Reliability Society.

Mahdi Fazeli received the M.Sc and Ph.D. degrees in computer engineering both from the Sharif University of Technology, Tehran, Iran, in 2005 and 2011, respectively. He has been with the department of computer engineering, Iran University of science and technology (IUST), since 2011, where he is currently an Assistant Professor. He has established and chaired two research laboratories at IUST since 2012, namely Dependable Systems and Architectures Laboratory (DSA) and Networked and Embedded System Laboratory (NESL). He has authored and co-authored more than 50 papers in reputable journals and conference proceedings. His research interests include reliable issues in VLSI circuits and emerging technologies, dependable embedded systems, Low power circuits and systems, fault-tolerant computer architectures, Fault injection and reliability modeling and evaluation..

Hossein Asadi (M08, SM14) received the B.S. and M.S. degrees in computer engineering from SUT, Tehran, Iran, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from Northeastern University, Boston, MA, USA, in 2007. He was with EMC Corporation, Hopkinton, MA, USA, as a Research Scientist and Senior Hardware Engineer, from 2006 to 2009. From 2002 to 2003, he was a member of the Dependable Systems Laboratory, SUT, where he researched hardware verification techniques. He has been with the Department of Computer Engineering, SUT, since 2009, where he is currently a tenured Associate Professor. He is the Founder and Director of the Data Storage Systems Laboratory at SUT. He has authored and co-authored more than sixty technical papers in reputed journals and conference proceedings. His current research interests include data storage systems and networks, solid-state drives, operating system support for I/O and memory management, and reconfigurable and dependable computing.