Memory Mapped SPM: Protecting Instruction Scratchpad Memory in Embedded Systems against Soft Errors

Hamed Farbeh¹, Mahdi Fazeli¹, Faramarz Khosravi¹, Seyed Ghassem Miremadi²

Department of Computer Engineering, Sharif University of Technology
Tehran, Iran
¹{farbeh, m_fazeli, fkhosravi} @ce.sharif.edu
²miremadi @sharif.edu

Abstract—Predictability, energy consumption, area and reliability are the major concerns in embedded systems. Using scratchpad memories (SPMs) instead of cache memories play an increasing role in satisfy these concerns. Both cache and SPM as on-chip SRAM memories are highly vulnerable to soft errors and as they contain the most frequently used blocks of the program, their errors can easily propagate in system leading to erroneous results. Unlike the instruction cache, an error in the instruction SPM cannot be corrected using only parity bits by invalidating the erroneous line. This study suggests a low-cost mechanism to protect the instruction SPM against soft errors. The main idea underlying the proposed mechanism includes four stages: 1) to use parity codes for error detection in the SPM, 2) to keep an address matching table in the main memory to store the address of the copy of SPM blocks in the main memory, in the case of dynamic SPM, 3) to allocate a specific segment of the main memory as an SPM backup, in the case of static SPM, and 4) to recover from errors using an interrupt service routine (ISR). Compared with a single error correction /double error detection (SEC-DED) scheme, by using a 2-bit interleaved-parity per word, the proposed mechanism can correct at least three bit errors; while SEC-DED is capable of correcting only single bit error and detecting 2-bit errors. The experimental results reveal that the energy consumption and area overheads of the proposed mechanism are approximately 22% and 15% less than that of SEC-DED for a 4Kbyte SPM, respectively. Moreover, this mechanism provides 10 times lower performance loss compared with SEC-DED.

Keywords—scratchpad memory; reliability; soft errors; error correction; embedded processors.

I. INTRODUCTION

Scratchpad memories (SPMs) are introduced as an alternative for cache memories in embedded systems [1], [2]. The SPM is an on-chip SRAM memory with disjoint address space from the off-chip memory [3], [4], [5]. The need for tag memory and hardware comparison circuit has made cache memories a bottleneck for energy constrained embedded systems [6], [7]. As these two hardware components are not included in SPMs, this supports SPMs to be more energy and area efficient as compared to cache memories. [8]. In Addition, unpredictable behaviour of cache memories prevents their widespread applications in real-time systems [3], [8], [9], [10], while software-managed SPMs not only simplify timing behaviour analysis, but also improve the performance [3].

Unlike cache memory, the SPM is managed explicitly by software [9]. To use the limited SPM space efficiently, several mapping algorithms have been proposed to manage the SPM. The SPM management algorithms consider data, instructions or both of them in various granularities as memory blocks that can be mapped to the SPM [2], [3]. However, there is no optimal algorithm to map program blocks to the SPM [2]. Although selecting between data and instructions for the SPM allocation is application-dependent, typically mapping the instruction is significantly more efficient for the improvement of performance and energy consumption.

Unlike data, instructions are fetched nearly in every clock cycle [11], [12], and fetching instructions from the main memory typically contributes to a large fraction of the system power consumption [7], [11], [13]. For example, in Motorola MCORE, more than 50% of the total processor power consumption is related to instruction cache [14]. Moreover, for mobile embedded applications, the code typically occupies a small fraction of the total program memory as compared to data [7], and the locality of code is more than data as well [15]. Consequently, mapping instructions instead of data to the SPM can result in greater energy savings [7], [15].

Furthermore, efficient instruction mapping can significantly improve the performance of the system [16]. Locality and predictability of the code is usually more than data in embedded applications [2], and typically, a small fraction of programs is responsible for most of the applications execution time [17]. Moreover, the number of applications executing in an embedded system is limited [18]; thus it is feasible to recognize the application execution patterns, to efficiently manage the SPM [18].

In addition to the SPM management concerns, reliability in the SPM is also one of the main concerns in embedded systems design. The SPM as an SRAM-based memory [5], [9] similar to cache memory is one of the most vulnerable on-chip components to soft errors [20]. Moreover, SPM contains the most frequently used sections of the program [9] and an error in the SPM can easily be propagated in the system. Therefore, to have a reliable system, it is necessary to protect the SPM against soft errors. Traditional memory protection schemes...
such as Error Correcting Codes (ECCs) and parity bits are not efficient for embedded systems [10]. Error correction is not possible in parity scheme when no data backup is available; on the other hand, area, access time latency and energy overheads of ECCs typically limit their applicability in energy constrained embedded systems [21], [22]. In addition, the ECCs have limited error correction capability and are not efficient to tolerate multiple bit upsets (MBU) which play a significant role to cause soft errors in memory [22].

In the SPM, there is no information about the primary address of the SPM blocks in the main memory; unlike cache memories which the address of their copy in the main memory can be determined by tag bits. This means that, any error in instruction caches or write-through data caches can be corrected by only detecting it and invalidating the erroneous line, while in the SPMs, error detection is not merely sufficient for error recovery.

A few studies related to the reliability of SPM have been addressed in the past. The research in [10] has considered the reliability of the data SPMs. This work which is based on ‘data block duplication under compiler control’ focuses on array-based embedded applications from the video/image processing domain. Here, an algorithm is proposed to identify the dead blocks and use them for backup copies of the live blocks at runtime. Although this approach has no negative effect on the performance, it cannot guarantee to make a backup copy for all the SPM blocks and to correct all the detected errors. The research performed in [23], suggested a RAID-like protection scheme to protect distributed on-chip memories, which needs aggressive voltage scaling to reduce parallel memory accesses and also is not applicable in a single on-chip memory. The work in [24] and [25] try to improve the SPM reliability by managing the thermal behaviour of the SPM. No one of the above studies has considered the instruction SPM, which is a major source of system failure.

In this paper, Memory Mapped SPM (MM-SPM) mechanism, as a low-cost fault-tolerant mechanism for the instruction SPMs, is suggested. The proposed MM-SPM mechanism includes four stages: 1) to protect SPM lines by parity bits, 2) to have a table in the main memory containing the addresses of the SPM blocks and their primary copy in the main memory in the case of dynamic SPM mapping, 3) to store a backup of all the SPM blocks in the main memory, in case of the static SPM mapping, and 4) using an interrupt service routine (ISR) for error recovery whenever an error is detected. In this mechanism, any error detected by parity bits can be corrected by overwriting the backup copy of the erroneous SPM line from the main memory to the SPM.

The proposed MM-SPM mechanism is implemented in VHDL model of a 32-bit processor for SPM sizes ranging from 512 byte to 16 Kbyte. Experimental results show the overheads of MM-SPM are considerably lower than conventional SEC-DED protection scheme, and its protection capability is also higher than SEC-DED scheme.

The proceeding parts of this paper are organized as follows. Section II reviews the previous studies on fault-tolerant methods for the SPM and cache. Section III describes the suggested mechanism to protect the SPMs against soft errors. In Section IV, the experimental setup is explained. Section V presents the results of our evaluation, and Section VI concludes this paper.

II. RELATED WORK

As mentioned in the previous section, there are a few studies for reliability enhancement of the SPMs. In this section, we focus primarily on explaining the work presented by [10] which is the most relevant work to the scope of this paper; subsequently, other SPM reliability relevant works will be introduced, and finally previous fault-tolerant approaches in cache which have been proposed to overcome the drawbacks of traditional memory protection schemes such as parity and the ECCs will be presented. The main purpose of these approaches is to reduce the error detection overhead in error-free conditions, based on the fact that error detection is needed for every memory access while error correction is performed just on rare faulty conditions.

In [10], a compiler-based approach has been presented to protect data SPM against soft errors. The main contribution of [10] is to use dead blocks of the SPM to store a copy of live data blocks under the control of the compiler without any performance loss. To eliminate any effect on the performance, it is not always possible to assign a duplicate copy for all live data blocks. Therefore, the proposed algorithm identifies a subset of live data blocks to have a duplicate copy, and at runtime, the duplicated blocks are created if there is a free block in the SPM. The SPM lines are protected by parity bits and whenever an error is detected in a data block, it will be corrected if there is a duplicate copy for the faulty block. As reported in [10], in normal operation of the system, the execution time overhead of this approach is the same as the pure parity-based method while its power consumption is much lower than the ECC method.

Some limitations of the approach presented in [10] are as follows:

- The proposed algorithm designed to identify dead blocks and to determine which SPM blocks can have a duplicate copy. It is limited to data blocks in nested loops and is not applicable to instructions and other data blocks such as stack, heap and global data.
- There may be several live blocks without a duplicate copy; thus there is no guarantee to correct all detected errors.
- No implementation and result have been reported by the authors to present the applicability and overheads of this approach.

The work in [24] tries to distribute the SPM accesses evenly all over the memory region to reduce hot spots. In [25] the on-chip memory accesses are evenly distributed between the SPM and cache to reduce the on-chip memory temperature. Both [24] and [25] try to reduce the vulnerability of SPM against soft errors, but they are fault avoidance techniques and have no solution in case of error occurrence, while this paper presents a fault-tolerant mechanism which can recover errors.
Unlike the SPMs, several novel fault-tolerant mechanisms for cache memories have been proposed with low overhead in normal operation of the system. In [21], ICR (In-Cache Replication) technique replicates the active blocks of data cache to blocks which do not need in near future to protect the active cache blocks.

The memory mapped ECC technique in [20] uses memory hierarchy instead of high-cost SRAM cache cells in order to store costly redundant error correction codes. As a result, it is possible to apply inexpensive error detection or light error correction codes to protect all the SRAM cache lines and use a low-cost DRAM memory to store more powerful and complicated ECCs for the dirty lines of the cache.

In [22], a low-cost error detection and correction approach based on the two-dimensional parity scheme [26] has been presented. It has been reported that timing and overheads of the proposed approach is the same as the standard parity method and it is applicable in the first level cache memories and other fast memories such as register files.

Authors of [27] have suggested protecting only the most frequently used portion of cache lines. In their parity caching technique, only the most frequently used lines of the cache are protected instead of uniformly applying check bits to all lines; and selective checking technique has been employed to protect a subset of lines for each set in the set-associative cache based on their access frequency.

III. MEMORY MAPPED SPM: THE PROPOSED MECHANISM

Due to the preference of instructions over data to be mapped to the SPM in the most embedded systems [7], [16], [12], [13], [18], [11], [15], in this study, we focus on protecting the instruction SPM against soft errors.

The SPM management approaches are categorized as static and dynamic approaches [10], [3], [6], [8]. Both of these two approaches have their own advantages and disadvantages. For large applications with several hotspots, the dynamic approach may result in better allocations compared to the static approach [3]. However, the execution time and energy consumption overheads to copy memory blocks from the main memory to the SPM at runtime in the dynamic approach are considerable [3]. In contrast, there is no runtime overhead for the static approach except the jump instructions when mapping basic blocks to the SPM [6]. Additionally, although data are likely to be allocated dynamically, even the static allocation of code can have a considerable improvement on energy consumption and performance [2]. In this paper, we considered both the static and the dynamic allocation approaches.

The main idea underlying the suggested MM-SPM approach is to protect the instructions in the SPM using its primary backup in a higher memory hierarchy. Since the instructions are supposed not to be modified at run-time and no update of the primary backup is needed, it can be expected that the error detection is sufficient to have the error recovery capability, as in instruction cache; however there is a basic difference between instruction cache and the instruction SPM.

Unlike cache memories, there is no additional information in the SPM about the addresses of the primary backup copies of the SPM blocks in a higher memory hierarchy. Consequently, to correct an error in the instruction SPM, it is necessary to have a mechanism to determine the addresses of the backup copies of the SPM blocks.

In the static allocation of instructions, the contents of the SPM remain unchanged during the application execution and in the dynamic allocation, the blocks of instructions can be moved to the SPM at runtime when needed [2], [3], [6]. Hence, the SPM contents are not fixed when the blocks are allocated dynamically.

Due to the difference between static and dynamic instruction SPM, the problem of finding the address of the backup copies of the SPM blocks is treated differently.

A. MM-SPM FOR STATICALLY ALLOCATED SPM

In static allocation approach, program blocks are loaded to the SPM when application starts and the SPM contents remain unchanged during the application execution. On the other hand, the primary copy of the SPM blocks is in a predetermined fixed location of the permanent memory. Consequently, there is a one-to-one relation between SPM blocks and their backup copy that are located in a constant offset from the SPM addresses. Therefore, to recover from an error occurrence in the SPM, the error-free backup copy is read using the address of the erroneous SPM line and the offset.

Error recovery is done by executing an interrupt service routine (ISR) that gets an SPM address as input argument. Whenever an error is detected, error signal from the error detection unit, as an interrupt, causes the processor to execute the ISR to correct the error. This ISR simply reads a memory line from the source address and stores it in the destination address. The ISR adds the address of the erroneous SPM line by the offset address to calculate the address of backup copy of the SPM line, then read from this address and write the data to the SPM address. Fig. 1 shows the ISR for error recovery in MM-SPM.

```plaintext
Procedure correct (SPM_address)
begin
    MM_address = SPM_address + offset;
    temp = Read (MM_address);
    Write (temp, SPM_address);
end Procedure
```

Figure 1. Interrupt Service Routine (ISR) for static SPM error correction

The operation of reading the backup copy from the permanent memory is significantly time consuming, though, it occurs only when an error is detected. If the latency of the recovery operation in error conditions, that caused by reading from permanent memory, is not tolerable, it is possible to have a backup of SPM in the main memory. In other words, MM-SPM suggests dedicating a specific portion of the main memory to store a backup copy of the SPM blocks.

Since the size of SPM is usually much less than the size of the main memory, e.g. ~Kbyte vs. ~Mbyte, the memory...
overhead of the backup copy is negligible. Fig. 2 depicts the MM-SPM architecture for the static allocated SPM with a backup copy in the main memory.

![MM-SPM architecture for statically allocated SPM](image)

**Figure 2. MM-SPM architecture for statically allocated SPM**

### B. MM-SPM FOR DYNAMICALLY ALLOCATED SPM

In dynamic allocation approach, all program blocks are primarily loaded in the main memory and they can move to the SPM when needed. In this case, although the SPM blocks are somewhere in the main memory, since mapped blocks are changing over time, it is not possible to locate the corresponding block in the main memory for each SPM block. The address of the backup blocks changes as the SPM blocks are replaced by new blocks. Consequently, the backup of the SPM blocks is distributed over the main memory.

In the dynamic SPM, the error recovery ISR cannot merely locate the backup copy of the SPM lines using the SPM address and the offset. Instead, MM-SPM suggests considering an address matching table to store the address of the backup copy of each SPM block. In fact, when a new block is loaded to the SPM, the address of that block in the main memory and its address in the SPM will be saved in this table. Fig. 3 illustrates the structure of this address matching table. The first column of the table is the block number; the second and third columns of the table are the start and the end addresses of blocks in the SPM, and the forth column is the start address of the blocks in the main memory. It is noteworthy that there is no need to store the end address of blocks in the main memory.

![Address matching table between SPM blocks and their backup copy in main memory](image)

**Figure 3. Address matching table between SPM blocks and their backup copy in main memory**

<table>
<thead>
<tr>
<th>Block Number</th>
<th>Start Address in SPM</th>
<th>End Address in SPM</th>
<th>Start Address in Main Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x08E00000</td>
<td>0x08E000C0</td>
<td>0x040010D8</td>
</tr>
<tr>
<td>1</td>
<td>0x08E000C4</td>
<td>0x08E00140</td>
<td>0x0400210C</td>
</tr>
<tr>
<td>2</td>
<td>0xFFFFFFFF</td>
<td>0xFFFFFFFF</td>
<td>0xFFFFFFFF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>0x08E00108</td>
<td>0x08E001E0</td>
<td>0x040023CC</td>
</tr>
</tbody>
</table>

In the case of error detection, the ISR searches the table according to the address of the erroneous SPM line in order to find the row that includes this address. Afterwards, the ISR reads a line from the main memory according to forth column and erroneous line offset from first column; then writes the data to the erroneous line of the SPM and the error will be recovered. The interrupt routine is presented in Fig. 4.

![Interrupt service routine for dynamic SPM error correction](image)

**Figure 4. Interrupt service routine for dynamic SPM error correction**

It should be noted that the ISR and address matching table are stored in the off-chip memory, and it has been supposed that the off-chip memory is protected against soft errors.

In this paper, the MM-SPM mechanism is based on the following facts:

- Latency and energy consumption of the conventional ECC protection schemes restrict applying them in the last level caches [21]; thus they cannot be suitable for the SPMs used in embedded systems with more energy consumption constraints [10].

- The error occurrences are extremely rare and even 10ns latency for correcting the error is acceptable [20]. This means that error correction latency of MM-SPM which is less than 1µs for static SPM and less than 500µs for dynamic SPM with address matching table with 100 entries is tolerable.

- Error detection is necessary for every access [20], however error correction is required only when an error occurs. Consequently, decoupling these two steps can have a significant effect on the reduction of performance loss and the energy consumption.

- The size of SPM is much less than that of the off-chip memory; therefore, the overheads of the backup copy of the SPM, for static SPMs, or the address table for dynamic SPMs, in the off-chip memory would be insignificant.

- Similar to instruction cache, no write-back operation is needed in the instruction SPM. Thus, the backup copy would not be updated and there is no write-back latency overhead for the backup copy of the SPM.
IV. EXPERIMENTAL SETUP

The proposed mechanism has been implemented in a 32-bit LEON2 processor [28]. A SRAM with single clock cycle access time is considered as the SPM and a SDRAM with 25 clock cycle access latency is adopted as the off-chip main memory according to [29], [30]. In order to estimate the access time and energy consumption of the SPM, we used CACTI SRAM model with 65nm feature size [31]. Energy per access of the off-chip memory is assumed 50 times higher than energy per access of a 16Kbyte SRAM, according to [29], [30]. Processor clock frequency is assumed to be 800MHz as well.

The SPM size is considered as a power of two ranging from 512bytes to16Kbytes. Mapping the program blocks the SPM is considered as the 0/1 knapsack problem [2], [3], [4], [6], [7], [26] to allocate the best subset of memory blocks to the SPM. A set of MiBench suite [32] programs including Basicmath, CRC32, Rijndael and Dijkstra is used as our benchmarks. To perform more comprehensive evaluations, we have further packed these programs in a single large program to have a program with various access patterns and several hotspots. We called this program All_in_One.

We have compared the suggested MM-SPM mechanism protected by a 2-bit interleaved parity per line with a SPM protected by classic SEC-DED (39,32) code, and assumed that both parity and SEC-DED generator as well as the checker circuits are in the critical path of memory accesses. Parity and SEC-DED circuits are synthesized by Synopsis Design Compiler® tool [33] for 65nm feature size, and their delays are added to the memory access time. Table I presents the delays of these circuits reported by Synopsis Power Compiler® tool.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>DELAY OF PARITY AND SEC-DED CIRCUITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parity circuit</td>
<td>SEC-DED circuit</td>
</tr>
<tr>
<td>0.37 ns</td>
<td>0.89 ns</td>
</tr>
</tbody>
</table>

V. RESULTS

In this section, we compared the suggested MM-SPM mechanism with the SEC-DED method in terms of performance, energy consumption and area overheads as well as reliability achievement. The results for both the static and dynamic SPMs are presented.

A. RESULTS OF MM-SPM FOR STATIC SPMS

Performance: The SPM access latency is a single clock cycle. Therefore, if the error detection and correction mechanisms add any clock cycle to the SPM access, they will degrade the performance of the system compared to the non-protected SPM. Table II shows the access times have been reported by CACTI, for the non-protected SPM, the MM-SPM mechanism and the SEC-DED protected SPM, for different SPM sizes.

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>SPM ACCESS TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPM Size (Bytes)</td>
<td>non-protected SPM</td>
</tr>
<tr>
<td>512</td>
<td>0.45</td>
</tr>
<tr>
<td>1024</td>
<td>0.43</td>
</tr>
<tr>
<td>2048</td>
<td>0.48</td>
</tr>
<tr>
<td>4096</td>
<td>0.51</td>
</tr>
<tr>
<td>8192</td>
<td>0.58</td>
</tr>
<tr>
<td>16384</td>
<td>0.61</td>
</tr>
</tbody>
</table>

For our experimental system setup with 800MHz processor clock frequency, two clock cycles are needed to access the SEC-DED protected SPM, while the access latency of the MM-SPM is a single clock cycle, the same as that of the non-protected SPM.

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>SPM ACCESS TIME IN TERM OF CLOCK CYCLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Frequency (MHz)</td>
<td>non-protected SPM</td>
</tr>
<tr>
<td>660</td>
<td>1</td>
</tr>
<tr>
<td>750&lt; f &lt;1000</td>
<td>1</td>
</tr>
<tr>
<td>1190&lt; f &lt;1330</td>
<td>1</td>
</tr>
<tr>
<td>1500&lt; f &lt;1630</td>
<td>1</td>
</tr>
</tbody>
</table>
Fig. 5 illustrates the performance gain for the MM-SPM and the SEC-DED protected SPM for different SPM sizes in five benchmarks as well as the average case, compared to the non-SPM system. The performance gain is defined as the ratio of the workload execution time in the non-SPM processor and the execution time of the SPM-based processor. As the performance gain increases for larger SPM sizes, the performance gain difference between the SEC-DED protected SPM and the MM-SPM mechanism increases as well. On average, for a 512-byte SPM, the performance gain for the MM-SPM and the SEC-DED protected SPM is 163.6% and 158.8%, respectively; resulting in nearly 0.6% performance loss by using the SEC-DED protected SPM. However for an 8Kbyte SPM, the performance gain for the MM-SPM is approximately 419.2% and for the SEC-DED protected SPM is nearly 368.2%; experiencing nearly 12.1% of the performance loss in the SEC-DED protected SPM compared to the MM-SPM. Generally, the performance loss of the SEC-DED scheme caused by additional clock cycle in the SPM access increases proportionally to the performance gain of the unprotected SPM. The higher performance gain using the SPM, the higher performance loss in the SEC-DED protected SPM compared to the non-protected SPM or the MM-SPM is experienced.

Energy consumption: Both of the SEC-DED and MM-SPM mechanisms increase the SPM energy per access; however their effect on the energy consumption is not the same, nor is it constant for different SPM sizes. The energy overhead consists of two components: the energy consumed by the error checker circuit and the energy consumed by the redundant bits. The energy consumption per access of the error checker circuit is independent of the SPM size because the circuit is the same for different SPMs. Table IV depicts the average energy per access of the error checker circuit for the MM-SPM as well as the SEC-DED protected SPM. On the other hand, the total number of the redundant bits in the entire SPM is proportional to its size; i.e. for the MM-SPM, two redundant bits for every 32 bits and for the SEC-DED protected SPM, seven bits for every 32 bits are needed. Therefore, the total energy consumption of the redundant bits is likely to have a constant ratio of the energy consumption in the memory cells of SPMs.

Fig. 6 demonstrates the SPM energy per access for different SPM sizes for the MM-SPM mechanism and SEC-DED protected SPM, normalized to that of the non-protected SPM. As the SPM size increases, the energy overhead is likely to have a constant ratio in the total SPM energy. It is due to the energy consumption of error checker circuit which remains unchanged for different SPM sizes and is not considerable for large SPM sizes, e.g. 8Kbyte and larger, while the energy consumption of the redundant bits is proportional to the SPM size and its ratio is the same for different SPM sizes. Consequently, increasing the SPM size will diminish the contribution of error checker circuit and the major factor of energy overhead would be the redundant check bits.

As depicted in Fig. 6, when the SPM size increases, the energy overhead for the MM-SPM approaches 6.4%, while for the SEC-DED, it is about 22%. For small SPM sizes e.g. 512bytes, the energy consumption overhead of the MM-SPM and the SEC-DED mechanisms is about 16.5% and 64.5%, respectively; however for larger SPMS e.g. 8Kbytes, the energy overhead of the MM-SPM and the SEC-DED protected SPM correspondingly decreases to 6.9% and 26.6%. Although the energy consumption of the SPM increases proportionally to the SPM size and the total consumed energy
is higher for the larger SPM, the energy dissipated by accessing the off-chip memory is much higher than accessing the SPM. Therefore, since more program blocks can be mapped to larger SPMs, the total number of off-chip memory references and consequently the total system energy consumption is reduced in such SPMs. Meanwhile, the performance gain is improved. Thus, it can wrongly be concluded that a larger SPM leads to the improvements in both the performance and the total energy consumption, regardless of the higher energy consumption of the larger SPM.

**TABLE IV**

<table>
<thead>
<tr>
<th>Error Checker Circuits</th>
<th>Parity circuit</th>
<th>SEC-DED circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.101 pJ</td>
<td>0.857 pJ</td>
</tr>
</tbody>
</table>

**Figure 6.** SPM energy per access normalized to non-protected SPM

However, the performance gain improves or at least remains unchanged as the SPM size increases, as long as the SPM latency is the same, while the total energy consumption of the system can increase for larger SPMs. The total energy consumption is decreased using the SPM, due to reducing the off-chip memory access. Consequently, the larger SPMs can lead to better energy consumptions if the reduction of the total energy consumption caused by less off-chip memory accesses can compensate the increased total energy consumed by the SPM.

To accomplish a more reasonable conclusion about the energy and performance improvement when increasing the SPM size, two parameters are defined: performance gain and energy gain; which are the performance or energy in the non-SPM system divided by performance or energy in the SPM-based system.

According to these definitions, our goal is to provide a higher performance gain while achieving higher energy gain. In order to define a parameter to show a compromise between energy consumption and performance gain in different SPM sizes, we defined SPM gain as the multiplication of performance gain and energy gain. As aforementioned, the SPM gain factor is used to show the relation between the energy consumption, performance gain and SPM size. As illustrated in Fig. 7, by increasing the SPM size, the SPM gain primarily increases and afterwards decreases for all the employed benchmarks. As an example, for Rijndael workload, the SPM gain increases by expanding the SPM size to 8Kbytes, subsequently the SPM gain decreases. Consequently, the optimum SPM size for this workload is 8Kbytes. As another example, for All_in_One workload, the SPM gain increases by increasing the SPM size to 16Kbytes. It reveals that increasing the SPM size is still commodious, however the SPM gain is saturating for the larger SPMs. Thus, it can be expected that for the SPMs larger than 16Kbyte, the SPM gain can decrease.

**Area:** Similar to the energy overhead, the area overhead consists of two components: the error checker circuit and the redundant bits. Compared to the redundant bits, the overhead of the error checker circuit is negligible. Therefore, the area
overhead of the protection mechanism is nearly proportional to the number of redundant bits. The area overhead of the MM-SPM and the SEC-DED Mechanisms is approximately 6.2% and 21.8%, respectively. Consequently, the area overhead of MM-SPM Mechanism is much less than that of the SEC-DED protected SPM.

**Reliability:** The MM-SPM mechanism is able to correct all errors which can be detected by the parity checker unit. Hence, it is possible to correct three adjacent errors and all odd numbers of errors in a line using a 2-bit interleaved parity for each memory line. On the other hand, the SEC-DED protected SPM is capable of correcting single errors and detecting double errors. Although it is possible to use more powerful ECCs in order to have higher error detection and correction capability, it requires meeting the cost of more energy, area and delay in ECC protected SPM.

Moreover, the error correction coverage of the MM-SPM mechanism is explicitly related to the error detection method used in the SPM, and it can be expected that the cost of such error detection method is much less than that of the ECC with the same error detection coverage.

**B. RESULTS OF MM-SPM FOR DYNAMIC SPM**

The difference between protecting a static SPM and a dynamic SPM is that in the dynamic SPM, the MM-SPM uses a table to save the addresses of the backup copies of all the SPM blocks which are scattered in the main memory.

**Performance:** Unlike the static SPM, the MM-SPM needs to update its table at run-time when a new memory block transferred to the dynamic SPM. Updating this table includes invalidating the rows of the victim blocks and assigning new values to the row of the new SPM allocated block. Thus, the MM-SPM increases the execution time compared to the non-protected SPM for the dynamic SPM. Fig. 9 displays the execution time overhead of the MM-SPM as well as the SEC-DED protected SPM.

According to Fig. 8, for the larger SPMs, the execution time overhead of the MM-SPM and the SEC-DED scheme decreases and increases, respectively. The reason for this observation is that in the MM-SPM, for the larger SPMs, the total number of transferring memory blocks from the main memory to the SPM, and consequently the frequency of updating the table decreases; while the execution time overhead of the SEC-DED scheme compared to the non-protected SPM is proportional to the number of the SPM accesses and for the larger SPMs with more SPM accesses, the execution time overhead of the SEC-DED scheme increases. On the average, the execution time overhead of the MM-SPM is 1.3% for a 512byte SPM and decreases to 0.5% for a 16Kbyte SPM; while that of the SEC-DED scheme is 5.2% for a 512byte SPM and increases to 16.5% for a 16Kbyte SPM.

**Energy:** The energy overhead of the parity bits in the MM-SPM for a dynamic SPM is the same as that in the static SPM which was about 19% less than that of the SEC-DED scheme. In the dynamic SPM, updating the address table increases the energy overhead of the MM-SPM mechanism; however, since the energy overhead of updating the table is less than 1%, it can be ignored.

**Area:** In the dynamic SPM, the table is located in the main memory and the order of its size is the total number of blocks which can be mapped to the SPM. The memory overhead of this table is less than 1% and is negligible. Thus, the area overhead of the MM-SPM for the dynamic SPM is the same as the static SPM. By protecting the SPM using a 2-bit interleaved parity per memory line, the area overhead of the MM-SPM is approximately 6.2%, while it is nearly 22% for the SEC-DED protected SPM.

**Reliability:** The error correction capability of the MM-SPM for the dynamic SPM is the same as the static SPM and all detected errors are corrected. Using 2-bit interleaved parities, at least three adjacent errors can be corrected, while in SEC-DED protected SPM, only a single bit error is correctable.
VI. CONCLUSION

The classical fault-tolerant techniques in the memory hierarchy such as the ECCs cannot efficiently be used to protect SPMs in embedded systems due to the execution time, energy consumption and area overheads. This study proposed a low-cost fault-tolerant mechanism for instruction SPMs named MM-SPM. In the proposed mechanism, for dynamic SPMs, a table is considered to save the address of each SPM block. MM-SPM is based on backup copy of SPM contents and all errors that are detected can also be corrected. The SPM lines are protected by a 2-bit interleaved parity per line and it can detect three adjacent bit errors. The overheads of this mechanism are negligible compared to the pure interleaved parity-protected SPM in error-free conditions. As compared to the SEC-DED scheme, energy consumption and area overheads of the MM-SPM are about 22% and 15% less than that of the SEC-DED protected SPM. Additionally, MM-SPM provides 10 times lower performance loss compared to SEC-DED protected SPM.

REFERENCES