IEICE TRANS. FUNDAMENTALS, VOL.E91–A, NO.7 JULY 2008

PAPER

Adopting the Drowsy Technique for Instruction Caches: A Soft Error Perspective*

Soong Hyun SHIN[†], **Sung Woo CHUNG**[††a)], ***Nonmembers***, **Eui-Young CHUNG**[†††], ***Member***, ***and*** **Chu Shik JHON**[†], ***Nonmember***

SUMMARY As technology scales down, leakage energy accounts for a greater proportion of total energy. Applying the *drowsy* technique to a cache, is regarded as one of the most efficient techniques for reducing leakage energy. However, it increases the Soft Error Rate (SER), thus, many researchers doubt the reliability of the drowsy technique. In this paper, we show several reasons why the *instruction cache* can adopt the drowsy technique without reliability problems. First, an instruction cache always stores read-only data, leading to soft error recovery by re-fetching the instructions from lower level memory. Second, the effect of the re-fetching caused by soft errors on performance is negligible. Additionally, a considerable percentage of soft errors can occur without harming the performance. Lastly, unrecoverable soft errors can be controlled by the *scrubbing* method. The simulation results show that the drowsy instruction cache rarely increases the rate of unrecoverable errors and negligibly degrades the performance.

key words: *instruction cache, soft error, drowsy technique, low-power*

1. Introduction

A soft error is a temporal malfunction caused by alpha particles and neutrons from cosmic radiation. It has become a crucial factor threatening the correctness of a system as process technology is scaled down. It makes memory cells upset or combinational logics perform inaccurate operations, which does not cause any permanent damage to the circuit [2]. Due to its severity, the soft error has been researched from different perspectives; physical phenomena of soft errors, protection schemes against soft errors, etc. Soft errors may result in serious problems in all parts of a system including combinational logics and sequential logics. As today's system-on-chips are evolving from logic-dominant to memory-dominant chips, the proportion of memories in a chip is rapidly increasing. Embedded memories are expected to occupy 94% of an SoC (System on Chip) by 2014 and the soft error hazard to sequential logics will increase as this proportion grows [3].

Among many sequential logics, a cache is an indispensable factor for system performance and its integrity is an essential requisite for system reliability. On the other hand, it consumes substantial leakage power. In addition, as the size of transistors becomes smaller, leakage accounts for higher proportion of power consumption. To reduce the increased leakage power, the drowsy cache (changing the supply voltage) was proposed [4]. In this technique, the supply voltage is lowered when the cache line is not expected to be accessed soon. Data is not lost when the cache line is in the leakage saving (low-voltage) mode, called 'drowsy mode.' In the drowsy mode, data is retained, although it cannot be accessed for read or write operations. Normal operations are performed after a wake-up, which is an operation to lift up the supply voltage to the normal voltage level. The drowsy technique indispensably reduces the reliability of a data cache due to soft errors [5], whereas it may not reduce the reliability of an instruction cache that stores read-only data by re-fetching instructions from lower-level memories (caches).

In this paper, we evaluate the reliability of an instruction cache and the performance overhead caused by soft error recovery (re-fetching the instructions from lower-level memories [1]). The instruction cache has been designed without any error correction technique, though it has parity bits that are for error detection. To the best of our knowledge, there has not been any quantitative study of the impact of soft errors on an instruction cache with the drowsy technique.

The remainder of this paper is organized as follows; Sect. 2 provides related works on soft errors. Section 3 describes a soft error model which is used for this study. Section 4 provides solutions to protect an instruction cache against soft errors. Section 5 classifies soft errors, depending on harmfulness and cache mode (normal/drowsy). Section 6 analyzes simulation results. Finally, Sect. 7 concludes this paper.

2. Related Works

2.1 Physical Perspective

The soft error is different from the hard error which is due to a permanent physical defect during fabrication [6]. The hard error is closely related to the process yield, which determines the chip cost [7]. Since today's chips are memory-dominant and embedded memories have aggressive design rules, semiconductor industries adopt yield optimization so-

Manuscript received November 30, 2007.
Manuscript revised March 26, 2008.

[†]The authors are with the School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea.
[††]The author is with the School of Computer and Information Technology, Korea University, Seoul 136-701, Korea (corresponding author; 82-2-3290-3194).
[†††]The author is with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea.

*This paper is an extended version of [1].

a) E-mail: swchung@korea.ac.kr

DOI: 10.1093/ietfec/e91–a.7.1772

lutions [3]. The yield can be improved by redundancy structures, which replace faulty cells. Redundancy policies are divided into a word redundancy, a word-line redundancy, a bit-line redundancy, and an IO redundancy [6], [8], [9]. Lastly, triple-modular redundancy (TMR), which is based on the two-out-of-three voting concept, has been suggested [10]–[12].

On the other hand, the soft error is a temporal upset caused by alpha particles and neutrons from cosmic radiation which cause memory cell upsets or induce inaccurate combinational operations [13]. Since the soft error was introduced [6], [14], the soft error has been studied from different perspectives: physical explanation of outbreak [15]–[17], modeling of the soft error rate [10], [13], [18]–[21], and detecting/correcting soft errors [22]–[28].

2.2 Architectural Perspective

As the supply voltage is decreased with respect to technology scaling, a system cannot be protected against the soft error with only circuit-/process-level supports. Thus, architectural approaches have been proposed [5], [29]–[32]. Memory designers initially concentrated on soft errors in a DRAM chip, rather than an SRAM chip. An error-correcting code, which can correct up to two soft errors on each word line within a DRAM chip was proposed [22]. This was an early attempt to obtain reliability of a DRAM by using systematic analysis. Software redundancies were also introduced [26], [27]. Software redundancy refers to the fact that redundant instructions are executed to guarantee satisfactory reliability; for example, instruction binaries are modified prior to execution [27], or some instructions are duplicated to detect errors in the system during run-time [26]. Data cells were proposed for use as redundant cells [28]. The data cache cell, which is not expected to be used in the near future, is evicted, and the unoccupied cell is used as a redundant cell.

An unrecoverable error is an error which cannot be recovered from by an error correction scheme. As the usage of a memory cell increases, the SER in a system increases, resulting in an increase in unrecoverable errors. While scrubbing (periodically detecting and correcting soft errors in all cache lines[†]) effectively reduces unrecoverable errors [24], [33], performance is degraded by scrubbing. Thus, the scrubbing period should be carefully selected, especially considering unrecoverable error rates.

Several architectural approaches to the soft error problem were proposed recently. In [30], architectural simulation was done on a cache; soft errors in cache lines are particularly critical. On the other hand, tags of an instruction cache have greatest robustness against soft errors. Soft errors are classified according to whether the error affects program outcome or not [29], [32]. Some researches deal with the soft error problem in a low power cache; the soft error susceptibility of a low power data cache was analyzed [5].

Additionally, since the drowsy primitive [4] weakens system reliability [34], the influence of the drowsy technique should be analyzed from an architectural perspective. Though there have been many studies about soft errors and low-power caches, to the best of our knowledge, the reliability of a drowsy instruction cache has not been studied. Without evaluating the dependability of a drowsy instruction cache, the drowsy technique cannot be adopted for an instruction cache in future technology.

3. Soft Error Model

There are several factors which influence the probability of soft errors: technologies, doping and packaging materials, altitude, etc. All the parameters, except supply voltage, are assumed to be constant, in order to investigate the relation between the supply voltage and the soft error rate. Based on [13], we calculate the soft error rate at sea level to be:

$$SER \propto A \times \exp\left(-\frac{Q_{crit}}{Q_s}\right) \tag{1}$$

where,

A : the drain area,
Q_{crit} : the critical charge,
Q_s : the collection slope.

A soft error occurs if collected charges of a cell exceed the critical charge Q_{crit} of the cell. The collection slope Q_s means the speed at which charges of cells are collected, which strongly depends on the doping density and the supply voltage Vcc. Because the critical charge Q_{crit} is proportional to the supply voltage, the SER increases exponentially, as the supply voltage decreases. In 70 nm technology, an SRAM cell operates normally at 1.0 V and retains data at 0.3 V, which is the drowsy state. Thus, the SER in the drowsy state is ten times higher than the SER in the normal state. Based on [5], the SER of a 6T SRAM (70 nm technology) cell is 2.7×10^{-14} per cycle (1 ns clock period), called as the raw SER in this paper. According to [35], the probability that an error occurs depends on the raw SER and how long data stays in the cache. Consequently, we determine the Probability of the Soft Error occurrence (PSE) when a cache line resides in a cache during a T cycle to be:

$$\begin{aligned} &PSE(mode, T, word\_bit) \\ &= 10^{mode} \times T \times word\_bit \times raw\_SER \end{aligned} \tag{2}$$

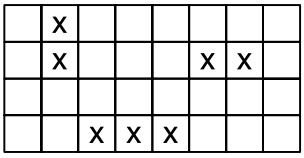
where,

$$mode = \begin{cases} 1 & \text{in the case of the drowsy state} \\ 0 & \text{in the case of the normal state} \end{cases}$$

$word\_bit$: the number of cells of a cache line.

In the current technology, most soft errors are Single Bit Errors (SBE)s. However, as the critical charge Q_{crit} is decreasing over time, there is a need for bit errors of two or greater to be addressed [36], [37]. The DBE (Double Bit Error) and the MBE (Multi-Bit Error) differ from multi-SBEs,

[†]A cache line is an access unit to a cache at a time.



(a) Example of DBEs and a MBE.

(b) Example of a multi-SBE.

Fig. 1 Two kinds of multiple bit errors. (The squares denote memory cells)

because they originate from a single upset. Therefore, in case of the DBE and the MBE, the erroneous cells are always adjacent, as shown in Fig. 1(a). By contrast, the multi-SBE implies that two or more SBEs occur in the same cache line. Erroneous cells are not adjacent, as shown in Fig. 1(b); in a 32-byte cache line, the probability that two erroneous cells, induced by a multi-SBE, are adjacent is quite low. Based on [5], [36], the DBE rate and the MBE rate are 0.01 and 0.0001 of the SBE rate, respectively, at 1.0 V supply voltage. By considering the multi-bit error, we can reformulate the equation and determine the PSE to be:

$$\begin{aligned} &PSE(error\_bit, mode, T, word\_bit) \\ &= 100^{1-error\_bit} \times 10^{mode} \times T \\ &\quad \times word\_bit \times raw\_SER \end{aligned} \tag{3}$$

where,

error_bit : the number of soft errors induced by a single upset.

The Possibility of Multi-SBE occurrence (PMSBE) is:

$$\begin{aligned} &PMSBE(m\_error\_bit, mode, T, word\_bit) \\ &= PSE(m\_error\_bit, mode, T, word\_bit)^{m\_error\_bit} \\ &= (10^{mode} \times T \times word\_bit \\ &\quad \times raw\_SER)^{m\_error\_bit} \end{aligned} \tag{4}$$

where,

m_error_bit: the number of soft errors occurring in a cache line.

4. Architectural Solutions for Soft Errors

For a reliable drowsy instruction cache, soft errors should be resolved by selecting the most suitable error correction methods, depending on their characteristics.

4.1 Single Bit Error

Until now, many error correcting codes have been introduced. They are categorized into time redundancies and hardware redundancies [38]. The time redundant techniques, which compare results of repeated executions to detect errors, are not suitable for detecting and correcting soft errors in an instruction cache. Since time redundant techniques repeatedly execute an operation for several times and check results, these techniques inevitably increase execution time. However, instruction cache access directly affects performance. Thus, time redundant techniques are not suitable for cache error detection/correction. Therefore, we concentrate on the hardware redundancies. The ability to detect and correct errors is highly dependent on the amount of hardware.

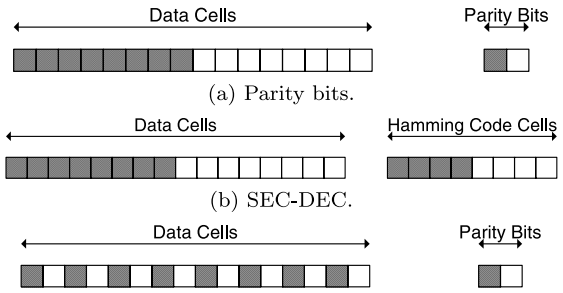


Fig. 2 Examples of error detection/correction codes in the case of a 2 byte cache line.

Among all types of soft errors, the SBE is most predominant, accounting for 98 ∼ 99 % of all soft errors. The SBE is detected by parity bits and corrected by Single Bit Correction Double Bit Detection (SBC-DBD), that is, the integration of parity bits and ECCs [39], as shown in Fig. 2. We assume that the cache line in Fig. 2 consists of 2 bytes, for simple explanation. Gray cells and white cells correspond to the first byte and the second byte, respectively. Fig. 2(a) depicts parity bits which are commonly used in instruction caches. Parity bit cells are additional data cells, the proportion of which is 1:8, in general. This is a very simple algorithm and incurs low hardware overhead; however, it cannot correct a single bit error or detect a multiple (even number) bit error. The SEC-DED corrects single bit errors. Its hardware overhead is substantial, as shown in Fig. 2(b). Though the SEC-DED can correct single bit errors, checking a parity bit is suitable for an instruction cache. This is because its hardware overhead is reasonable for commercial processors, since the parity bits were already applied to an instruction cache. Another reason is that erroneous instructions can be re-fetched from lower-level memory (or cache), in order to recover from soft errors in an instruction cache. In this case, re-fetching overhead is included in the memory access time when a single bit error is detected. However, the overhead is expected to be negligible, which is shown in Sect. 6.

4.2 Double or Multiple Bit Error

The DBE and the MBE were previously considered insignificant. However, as technology shrinks transistor sizes and the supply voltage decreases, both the probability of the DBE and the MBE increase, as does that of SBE. Additionally, the drowsy technique, which is one of the most popular

low power techniques, increases the SER as explained in Sect. 3.

Many researchers were concerned about the DBE and published a number of papers about detecting or correcting the DBE. This is because its probability is increasing, but it is not detected by parity bits. Though many error correcting codes have been introduced, most of them are designed for communication, where the error rate is high. To guarantee reliability of network communication, data is highly redundant. However, if these error correcting codes were applied to an instruction cache, they would incur unnecessary hardware overhead.

For this reason, the bit interleaved parity was recommended, which is commonly used to minimize the error rate of multi-bit errors [36]. The bit interleaved parity assigns physically adjacent bits to different parity units, since the DBE or the MBE always has adjacent erroneous bits. When the parity bits are interleaved, a DBE can be treated as two SBEs, as shown in Fig. 2(c).

4.3 Multi-SBE

The multi-SBE is two or more SBEs which occur in the same cache line in a certain cache. The multi-SBE can be avoided by checking parity bits and correcting SBEs, in most cases. Though the error rate is still low, it is necessary to guarantee a reasonable level of reliability, due to an increased SER in the future.

The most appropriate scheme for the prevention of multi-SBE is scrubbing [24], [33]. During the scrubbing, all cache lines and their parity bits are checked to remove a single-bit error. We simulated the scrubbing effect with various scrubbing periods, and the results are shown in Sect. 6.

5. Harmful/Harmless Error in Normal/Power-Saving Mode

Soft errors of an instruction cache disappear without causing any damage, when the erroneous cache line is evicted without further read operations. Thus, the residence time for each cache line can be separated into harmful periods or harmless periods according to whether the following reference exists or not. In Fig. 3, data are fetched and stored in an instruction cache at time t1 and evicted at time t5. There are two reads, at times t1 and t3, and the cache line is woken up at these times. The cache line is put in the drowsy mode at times t2 and t4, when the supply voltage is lowered from 1.0 V to 0.3 V. The period prior to the previous read, 'Read_2,' is a harmful period, and the subsequent period is a harmless period. Each of the harmful or harmless periods is separated into a drowsy mode or a normal mode, based on the supply voltage. In summary, the residence time for each cache line is classified into harmful periods in normal mode, harmful periods in drowsy mode, harmless periods in normal mode, and harmless periods in drowsy mode. We show the proportion of each period for applications in Sect. 6.

In the proposed technique an instruction cache considers the SBEs as the cache misses, thus, the recovery (refetching) is simple. As a side effect, the cache miss rate and the AMAT (Average Memory Access Time) will increase. We calculated the AMAT to be:

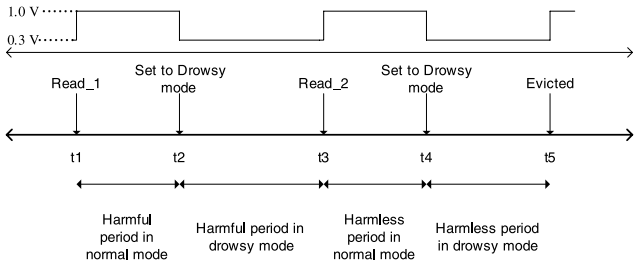


Fig. 3 Error classifications in the drowsy instruction cache.

Table 1 Notations.

Notation	Description
HT_{cache_name}	Hit time of *cache_name*
WR_{cache_name}	Wake-up rate of *cache_name*
WP_{cache_name}	Wake-up penalty of *cache_name*
MER_{cache_name}	Miss+Error rate of *cache_name*
MR_{cache_name}	Miss rate of *cache_name*
MP_{cache_name}	Miss penalty of *cache_name*
ER_{cache_name}	Error rate of *cache_name*

$$AMAT = HT_{L1} \times (1 + WR_{L1} \times WP_{L1}) + (HT_{L2} + MR_{L2} \times MP_{L2}) \times MER_{L1} \quad (5)$$

$$MER_{L1} = MR_{L1} + ER_{L1}. \quad (6)$$

Detailed notations are shown in Table 1.

6. Simulations

We implement a soft error generator in 'SimpleScalar' 3.0 [40] and evaluate soft error damage in a drowsy instruction cache. The default configurations of the simulator are: a 16 KB, 32 B cache line, 4-way L1 instruction cache; a 16 KB, 32 B cache line, 4-way L1 data cache; a unified 256 KB, 64 B cache line, 4-way L2 cache. The access cycles for the L1, L2, and memory are 1, 8, and 40, respectively, at 1 GHz clock frequency. In the drowsy instruction cache, cache lines are put into the drowsy mode every 2000 cycles and they are protected by bit interleaved parity bits and scrubbing. The benchmarks are selected from 'SPEC2000' [41]. Each benchmark is initially fast forwarded for 300 million instructions, then, simulated for 1 billion instructions. Table 2 gives the SER of a bit cell, for each voltage mode of the 6T SRAM (70 nm technology).

6.1 Performance Overhead of Re-Fetching Instructions

Figure 4 shows a decomposition of soft errors. In all the benchmarks, the number of DBEs and MBEs is less than 1%; it is very difficult to find them in Fig. 4. Every DBE is detected by bit interleaved parity bits and corrected by

Table 2 The soft error rate of a bit cell per cycle [5].

	Normal State	Drowsy State	Read /Write
Single Bit Error Rate	$2.7e-14$	$2.7e-13$	$1.4e-13$
Double Bit Error Rate	$2.7e-16$	$2.7e-15$	$1.4e-15$
Multi Bit Error Rate	$2.7e-18$	$2.7e-17$	$1.4e-17$

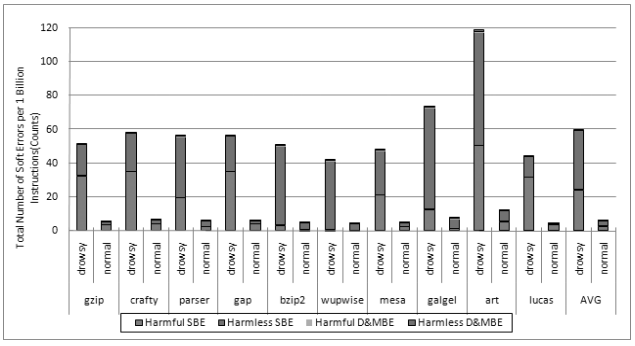


Fig. 4 Decomposition of soft errors.

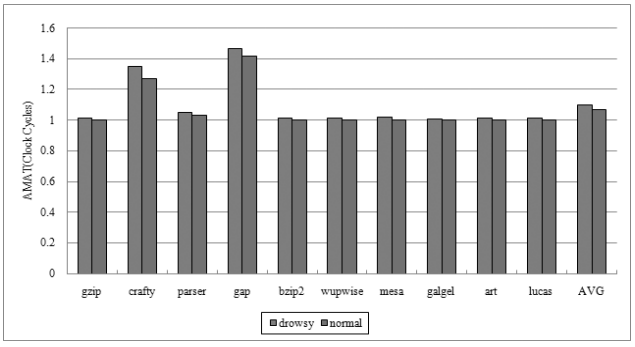


Fig. 5 The AMAT, including wake-up overheads of the drowsy instruction cache.

re-fetching erroneous instructions. During simulations, a multi-SBE (an unrecoverable soft error) does not occur. This is because of its extremely low probability, due to the high temporal locality of instruction caches. For example, the average inter-reference time of 'gzip' is 689.7 clock cycles, and the probability of an unrecoverable soft error is 6.22e−17 during the period. Note instruction caches are accessed repeatedly for a short period; thus, the inter-reference time is short.

Though the total number of soft errors in the drowsy cache is approximately 9.8 times higher than that of the normal cache, the total number of soft errors in the drowsy cache is 59.9, on average. Moreover, 60% of the soft errors in the drowsy cache are classified as harmless soft errors. Thus, the drowsy instruction cache has only 24.37 soft errors while executing 1 billion instructions. Additionally, the harmless error rates for some benchmarks, such as 'bzip2,' 'wupwise,' 'galgel,' and 'art,' are relatively high, because of the proportions of harmless periods (Fig. 4). Program sizes of these benchmarks are small, therefore, a small number of instructions are executed repeatedly. Thus, a large proportion of the cache is not used and the soft errors are classified as harmless errors. For instance, the harmless period of execution of 'bzip2' is 91.8% and only 3.12 errors are harmful errors, out of 50.7 soft errors.

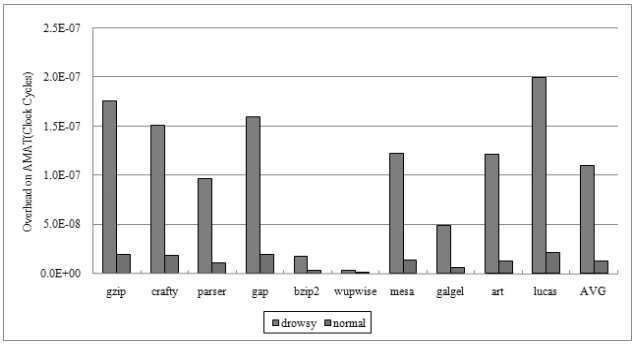


Fig. 6 The AMAT overhead caused by soft errors, excluding wake-up overheads of the drowsy instruction cache.

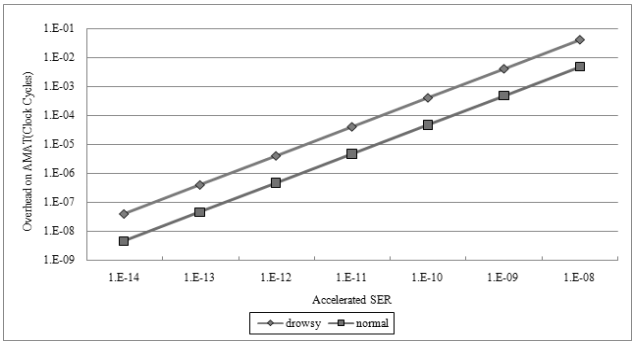


Fig. 7 The comparison between the AMAT overhead in a normal cache and the AMAT overhead in a drowsy cache, with various soft error rates.

To evaluate soft error damage to the instruction cache, we examined the AMAT. As shown in Fig. 5, the AMAT of the drowsy cache is 1.096 and that of the normal cache is 1.071; the AMAT of the drowsy cache is approximately 2.3% longer than that of the conventional cache. Note, Fig. 5 includes the wake-up overhead of the drowsy cache in the AMAT. The AMAT overhead caused by soft errors is negligible, as shown in Fig. 6, because its probability is 1.09e−7 in the drowsy cache. Thus, most of the AMAT overhead shown in Fig. 5 is not caused by soft error recovery, but by the drowsy primitive itself, because the cache line in the drowsy state suffers additional wake-up overhead when it is accessed. Consequently, the SER is still too low to affect the performance of the drowsy cache, for current technology.

Though the effect of the soft error on the drowsy instruction cache is negligible for current technology, the SER is expected to increase in the future. To evaluate the SER for future technology, we simulated the AMAT with accelerated SERs, from 1e−14 to 1e−8 (for a bit cell). Table 2 shows that the current SER of a bit cell per a cycle is 2.7e−14. Fig. 7 depicts the comparison between the AMAT overhead in a normal cache and the AMAT overhead in a drowsy cache. The results were obtained with future SERs. Ob-

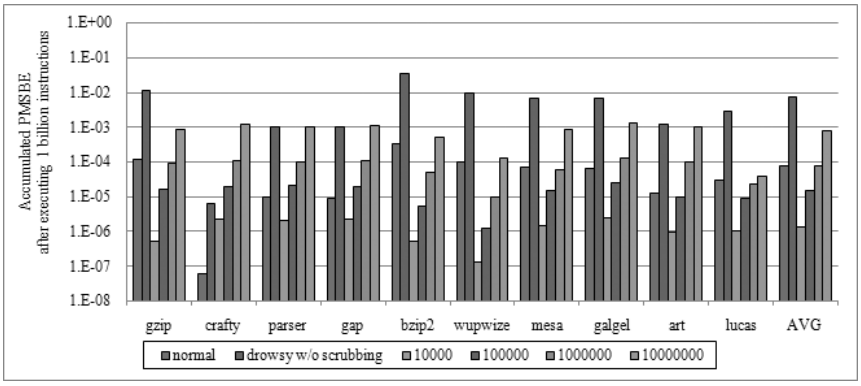


Fig. 8 The accumulated PMSBE values.

viously, the overhead for the AMAT increases as the SER increases. Even though the SER is 1e−8 (which means that the SER is 2.7e+5 times higher than the current SER level), the overhead of the AMAT is 0.04 cycles[†]. Consequently, when the SER of a cache line is below 1.e−8, the drowsy instruction cache can be used with negligible overhead to handle the erroneous cache lines.

6.2 Unrecoverable Error

Though no unrecoverable errors occurred during simulations, the unrecoverable error rate of the drowsy cache must be higher than that of the normal cache. We use an accumulated PMSBE value for a cache (not a cache line) during harmful periods. In our simulations (one billion instructions executed), the accumulated PMSBE in the drowsy instruction cache and in the normal instruction cache are 7.52e−3 and 7.48e−5, respectively, as shown in Fig. 8. Though the unrecoverable error rate is still low in any cache, the drowsy instruction cache is approximately 100 times more vulnerable than the normal instruction cache.

Scrubbing, explained in Sect. 4.3, is an effective way to mitigate the occurrence of unrecoverable errors. This technique is based on the idea that SBEs can be fixed by periodically checking the parity bits against the data bits. To find the optimal scrubbing period, we investigated the accumulated PMSBE with respect to the inter-reference period. We performed simulations using different values for the scrubbing period; 1.e4, 1.e5, 1.e6, and 1.e7 cycles. As shown in Fig. 8, the reliability of the drowsy instruction cache approaches that of the normal instruction cache, when the scrubbing period is 1.e6.

6.3 Power Consumption Evaluation

Though it is clear that the drowsy cache consumes less leakage power, scrubbing the cache periodically increases dynamic energy consumption. Thus, we evaluated the energy consumption of the drowsy cache including the dynamic energy overhead by scrubbing. The leakage energy parameters are obtained from [42] and the dynamic energy parameters are calculated from 'CACTI' [43]. The parameters of parity encoding/decoding logics are obtained from [5], and the scrubbing period is 1.e6 cycles, which is selected in the prior subsection. According to our CACTI calculation, the cache access energy for reading a cache line is 0.02528 nJ. The values for leakage power/bit in the normal mode and in the drowsy mode are 0.0778 and 0.0167 μW, respectively. The energy values of parity encoding and decoding are 6.0232 and 7.2239 pJ, respectively.

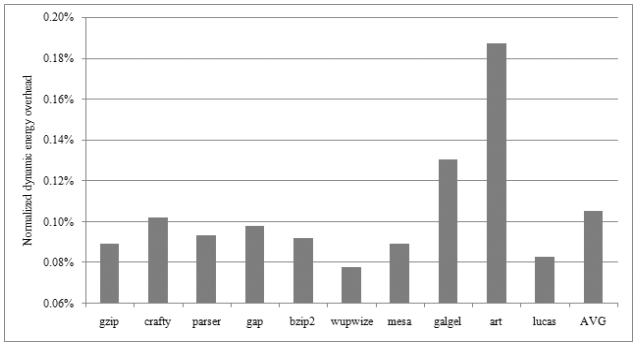


Fig. 9 Dynamic energy overhead.

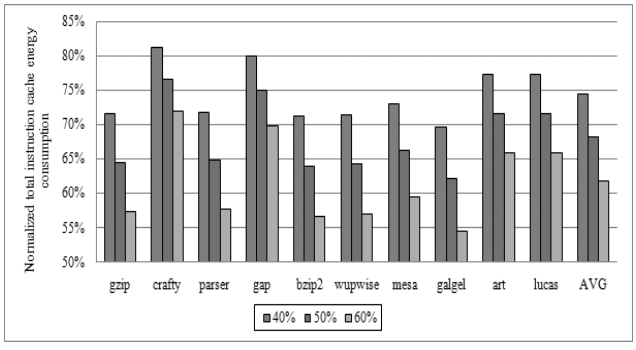


Fig. 10 Total energy consumption.

As shown in Fig. 9, scrubbing periodically increases the dynamic energy consumption by 0.12% on average, and 0.18% in the worst case. Technology advance makes the

[†]Note that the overhead is not for the CPI(clock per instruction) but for the AMAT.

leakage energy consumption attract more attention than before. In [44], the leakage power will exceeds the active power beyond 65nm technology node. Fig. 10 shows the total energy consumption of the drowsy cache, which is normalized to that of the conventional cache. We could not find reliable research showing parameters of the leakage power of the instruction cache and the dynamic power of parity logic. Thus, for fair comparison, we simulated with several ratios of the leakage power to the total power. Simulated ratios were 0.4, 0.5 and 0.6, because the ratio is around 0.5 in 70 nm technology [44]. If the ratio is 0.5, the drowsy cache with periodical scrubbing reduces the total energy consumption by 32%, onverage. Consequently, the power reduction of the drowsy cache is significant even if the scrubbing overhead is considered.

7. Conclusion

This paper presents the first quantitative evaluation of the reliability of the drowsy instruction cache, which is regarded as one of the best power-saving techniques. It has been widely considered that the drowsy cache is inefficient for use in future applications because the SER increases exponentially when the supply voltage is suppressed. The key observation is that instruction caches always store read-only data. Thus, the drowsy instruction cache overcomes the soft error problems by re-fetching the corresponding data. Though the SER is expected to increase in the future, refetching has negligible effect on performance. Therefore, it is reasonable to apply the drowsy technique to an instruction cache. The multi-SBE threatens system reliability, because this problem cannot be resolved with bit interleaved parity bits. However, the unrecoverable error rate of the drowsy instruction cache can be effectively reduced to that of normal instruction caches by scrubbing.

Acknowledgements

This work was supported by the Second Brain Korea 21 Project, a Korea University Grant and the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. R01-2007-000-20750-0).

References

[1] S.H. Shin, S.W. Chung, and C.S. Jhon, "On the reliability of drowsy instruction caches," Asia-Pacific Computer Systems Architecture Conference, pp.445–451, 2006.
[2] T. Karnik, P. Hazucha, and J. Patel, "Characterization of soft errors caused by single event upsets in cmos processes," IEEE Trans. Dependable and Secure Computing, vol.01, no.2, pp.128–143, 2004.
[3] Y. Zorian and S. Shoukourian, "Embedded-memory test and repair: Infrastructure ip for soc yield," IEEE Des. Test Comput., vol.20, no.3, pp.58–66, 2003.
[4] K. Flautner, N.S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: Simple techniques for reducing leakage power," SIGARCH Comput. Archit. News, vol.30, no.2, pp.148–157, 2002.
[5] V. Degalahal, L. Li, V. Narayanan, M. Kandemir, and M.J. Irwin, "Soft errors issues in low-power caches," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.13, no.10, pp.1157–1166, 2005.
[6] J.F. Ziegler and W.A. Lanford, "Effect of cosmic rays on computer memories," Science, vol.206, pp.776–788, Nov. 1979.
[7] A. Silvagni, G. Fusillo, R. Ravasio, M. Picca, and S. Zanardi, "An overview of logic architecture inside flash memory devices," Proc. IEEE, vol.91, no.4, pp.569–580, 2003.
[8] E. Rondey, Y. Tellier, and S. Borri, "A silicon-based yield gain evaluation methodology for embedded-srams with different redundancy scenarios," MTDT'02: Proc. 2002 IEEE International Workshop on Memory Technology, Design and Testing, pp.57–61, 2002.
[9] S.E. Schuster, "Multiple word/bit line redundancy for semiconductor memories," IEEE J. Solid-State Circuits, vol.13, no.5, pp.698–703, 1978.
[10] R.E. Lyions and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," IBM J. Res. Dev., vol.6, no.2, pp.200–209, 1962.
[11] K. Mohanram and N.A. Touba, "Partial error masking to reduce soft error failure rate in logic circuits," DFT'03: Proc. 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp.433–440, 2003.
[12] K. Nikolic, A. Sadek, and M. Forshaw, "Fault-tolerant techniques for nanocomputer," Nanotech, vol.13, pp.357–362, 2002.
[13] P. Hazucha and C. Svensson, "Impact of cmos technology scaling on the atmospheric neutron soft error rate," IEEE Trans. Nucl. Sci., vol.47, no.6, pp.2586–2594, 2000.
[14] T.C. May and M.H. Woods, "A new physical mechanism for soft error in dynamic memories," Proc. IEEE International Reliability Physics Symposium, pp.33–40, 1978.
[15] R. Baumann, "Soft errors in advanced computer systems," IEEE Des. Test of Comput., vol.22, no.3, pp.258–266, 2005.
[16] D. Lambert, J. Baggio, G. Hubert, V. Ferlet-Cavrois, O. Flament, F. Saigne, F. Wrobel, H. Duarte, J. Boch, B. Sagnes, N. Buard, and T. Carriere, "Neutron-induced seu in srams: Simulations with n-si and n-o interactions," IEEE Trans. Nucl. Sci., vol.52, no.6, pp.2332–2339, 2005.
[17] T.J. O'Gorman, "The effect of cosmic rays on the soft error rate of a dram at ground level," IEEE Trans. Electron Devices, vol.41, no.4, pp.553–557, 1994.
[18] H. Asadi and M.B. Tahoori, "Soft error modeling and protection for sequential elements," DFT'05: Proc. 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05), pp.463–474, 2005.
[19] H.T. Nguyen, Y. Yagil, N. Seifert, and M. Reitsma, "Chip-level soft error estimation method," IEEE Trans. Device and Materials Reliability, vol.5, no.3, pp.365–381, 2005.
[20] R.R. Rao, K. Chopra, D. Blaauw, and D. Sylvester, "An efficient static algorithm for computing the soft error rates of combinational circuits," DATE'06: Proc. Conference on Design, Automation and Test in Europe, pp.164–169, 2006.
[21] K. Yamaguchi, Y. Takemura, K. Osada, K. Ishibashi, and Y. Saito, "3-d device modeling for sram soft-error immunity and tolerance analysis," IEEE Trans. Electron Devices, vol.51, no.3, pp.378–388, 2004.
[22] P. Mazumder, "An on-chip double-bit error-correcting code for three-dimensional dynamic random-access memory," Proc. International Test Conference, pp.279–288, 1988.
[23] P.J. Meaney, S.B. Swaney, P.N. Sanda, and L. Spainhower, "Ibm z990 soft error detection and recovery," IEEE Trans. Device and Materials Reliability, vol.5, no.3, pp.419–427, 2005.
[24] S.S. Mukherjee, J. Emer, T. Fossum, and S.K. Reinhardt, "Cache scrubbing in microprocessors: Myth or necessity?," Proc. IEEE Pacific Rim International Symposium on Dependable Computing, pp.37–42, 2004.
[25] M. Nicolaidis, "Design for soft error mitigation," IEEE Trans. Device and Materials Reliability, vol.5, no.3, pp.405–418, 2005.
[26] N. Oh, P.P. Shirvani, and E.J. McCluskey, "Error detection by duplicated instructions in super-scalar processors," IEEE Trans. Reliab., vol.51, no.1, pp.63–75, 2002.

[27] M. Rebaudengo, M.S. Reorda, M. Torchiano, and M. Violante, "Soft-error detection through software fault-tolerance techniques," DFT'99: Proc. 14th International Symposium on Defect and Fault-Tolerance in VLSI Systems, pp.210–218, 1999.
[28] W. Zhang, S. Gurumurthi, M. Kandemir, and A. Sivasubramaniam, "Icr: In-cache replication for enhancing data cache reliability," Proc. International Conference on Dependable Systems and Networks, pp.291–300, 2003.
[29] S.S. Mukherjee, J. Emer, and S.K. Reinhardt, "The soft error problem: An architectural perspective," HPCA'05: Proc. 11th International Symposium on High-Performance Computer Architecture, pp.243–247, 2005.
[30] M. Rebaudengo, M.S. Reorda, and M. Violante, "An accurate analysis of the effects of soft errors in the instruction and data caches of a pipelined microprocessor," DATE'03: Proc. Conference on Design, Automation and Test in Europe, pp.10602–10607, 2003.
[31] M. Sugihara, T. Ishihara, M. Muroyama, and K. Hashimoto, "A simulation-based soft error estimation methodology for computer systems," ISQED'06: Proc. 7th International Symposium on Quality Electronic Design, pp.196–203, 2006.
[32] C. Weaver, J. Emer, S.S. Mukherjee, and S.K. Reinhardt, "Techniques to reduce the soft error rate of a high-performance microprocessor," SIGARCH Computer Architecture News, vol.32, no.2, pp.264–275, 2004.
[33] A. Saleh, J. Serrano, and J. Patel, "Reliability of scrubbing recovery techniques for memory systems," IEEE Trans. Reliab., vol.39, no.1, pp.114–122, 1990.
[34] V. Narayanan and Y. Xie, "Reliability concerns in embedded system designs," Computer, vol.39, no.1, pp.118–120, 2006.
[35] X. Li, S.V. Adve, P. Bose, and J.A. Rivers, "Softarch: An architecture level tool for modeling and analyzing soft errors," International Conference on Dependable Systems and Networks (DSN), pp.496–505, 2005.
[36] J. Maiz, S. Hareland, K. Zhang, and P. Armstrong, "Characterization of multi-bit soft error events in advanced srams," Digest of International Electron Devices Meeting, pp.21.4.1–21.4.4, 2003.
[37] G. Neuberger, F. de Lima, L. Carro, and R. Reis, "A multiple bit upset tolerant sram memory," ACM Trans. Des. Autom. Electron. Syst., vol.8, no.4, pp.577–590, 2003.
[38] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K.S. Kim, "Robust system design with built-in soft-error resilience," Computer, vol.38, no.2, pp.43–52, 2005.
[39] C.L. Chen and M.Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," IBM J. Res. Dev., vol.28, no.2, pp.124–134, 1984.
[40] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," Computer, vol.35, no.2, pp.59–67, 2002.
[41] J.L. Henning, "Spec cpu2000: Measuring cpu performance in the new millennium," Computer, vol.33, no.7, pp.28–35, 2000.
[42] S.W. Chung and K. Skadron, "On-demand solution to minimize i-cache leakage energy with maintaining performance," IEEE Trans. Comput., vol.57, no.1, pp.7–24, 2008.
[43] P. Shivakumar and N.P. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," WRL Research Report, 2001.
[44] B. Coyle, R. Arghavani, D. Barlage, S. Datta, M. Doczy, J. Kavalieros, A. Murthy, and R. Chau, "Transistor elements for 30 nm physical gate lengths and beyond," Intel Tech. J., vol.6, no.2, pp.42–54, 2002.

Soong Hyun Shin is a Ph.D. candidate in the School of Electrical Engineering and Computer Science at Seoul National University in Korea. His research interests include soft error correction, new generation smart card and low power computer architecture. He received BSE and MSE in Electrical Engineering and Computer Engineering from Seoul National University in Korea.

Sung Woo Chung received the B.S. degree in computer engineering and the Ph.D. degree in electrical and computer engineering from Seoul National University, Korea, in 1996 and 2003, respectively. He worked as an academic visitor for the IBM T.J. Watson Research Center, Yorktown Heights, New York, in 2002. From 2003 to 2005, he worked for Samsung Electronics as a senior engineer. In 2005, he worked as a research scientist at the University of Virginia at Charlottesville. He joined the Division of Computer and Communication Engineering at Korea University, Seoul, as an assistant professor in 2006. His research interests include technology-aware design for microarchitecture and SoC (system on chip) and architectural supports for flash memories.

Eui-Young Chung received the B.S. and M.S. degrees in electronics and computer engineering from Korea University, Seoul, Korea, in espectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2002. Currently, he is an Associate Professor with the School of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea. From 1990 to 2005, he was a Principal Engineer with SoC R&D Center, Samsung Electronics, Kiheung, Korea. His research interests include system architecture and VLSI design including all aspects of computer aided design with the special emphasis on low power applications and on-chip interconnection network.

Chu Shik Jhon is a professor in the Electrical Engineering and Computer Science at Seoul National University in Koera. His research interests include VLSI/CAD and parallel computer architecture. He received his B.S. (1974) degree in Applied Mathematics from Seoul National University, his M.S. (1976) degree in Computer Science from Korea Advanced Institute of Science and Technology, and his Ph.D. (1982) degree in Computer Science from University of Utah. From 1982 to 1984, he worked as a faculty member of University of Iowa.