社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2006－ＳＬＤＭ－126（17）
2006／10／26

コンピュータシステムにおける信頼性と性能のトレードオフの解析と高信頼性キャッシュアーキテクチャ

杉原　真†　石原　亨††　村上　和彰†††

†財団法人九州システム情報技術研究所，〒814-0001 福岡市早良区百道浜 2-1-22 福岡 SRP センタービル 7F
††九州大学システム LSI 研究センター，〒814-0001 福岡市早良区百道浜 3-8-33
†††九州大学大学院システム情報科学研究院情報理学専攻，〒816-8580 春日市春日公園 6-1
E-mail: †sugihara@isit.or.jp

あらまし　本稿では，コンピュータシステムが様々なメモリ部品から構成される点に着目し，コンピュータシステム全体としての信頼性を議論する．具体的には，SRAM と DRAM のソフトエラー率の違いに着目し，一般的なコンピュータシステムにおいては，信頼性と性能の間にトレードオフが存在することを解析する．また，信頼性と性能を両立する「高信頼性キャッシュアーキテクチャ」を提案する．高信頼性キャッシュアーキテクチャにおいては，二つの動作モード，すなわち，信頼性モード及び性能モードを導入する．キャッシュメモリの動作モードを制御することにより，コンピュータシステムの信頼性と性能を両立する．計算機実験により，提案する高信頼性キャッシュアーキテクチャがコンピュータシステムの信頼性と性能を両立することを示す．

キーワード　ソフトエラー，信頼性，性能，キャッシュメモリ

An Analysis on a Tradeoff between Reliability and Performance and a Reliable Cache Architecture for Computer Systems

Makoto SUGIHARA†, Tohru ISHIHARA††, and Kazuaki MURAKAMI†††

† ISIT, 2-1-22 Momochihama, Sawara-ku, Fukuoka 814-0001 Japan
†† System LSI Research Center, Kyushu University, 3-8-33 Momochihama, Sawara-ku, Fukuoka 814-0001 Japan
††† The Department of Informatics, Kyushu University, 6-1 Kasuga-Koen, Kasuga 816-8580
E-mail: †sugihara@isit.or.jp

Abstract　In this paper, we discuss reliability of a computer system which consists of various IC components. We focus on the difference between SERs of SRAM and DRAM memories and analyze a tradeoff between reliability and performance of a computer system. We also propose reliable cache architectures for both reliability and performance. We introduce reliability and performance modes for reliable cache architectures. Controlling the operation mode of a computer system decreases vulnerability of a computer system under a certain performance constraint. Our experiments show that switching the operation modes of the cache memories affects performance and reliability of a computer system and is effective for tuning its performance and reliability.

Key words　Soft Error, Reliability, Performance, Cache Memory

1. Introduction

A hard error is caused by a permanent physical defect while a soft error usually refers to a transient defect which is caused by thermal neutrons, high-energy neutrons, or α particles. May first discovered that α particles emitted from radioactive substances caused soft errors in DRAM modules [6]. The feature size of integrated circuits has reached nano-scale and the nano-scale transistors become more soft error sensitive. Soft error estimation and highly-reliable design become of utmost concern in mission-critical systems as well as consumer products. Recently, several techniques for estimating soft errors have been proposed. In [3, 8, 15], soft error simulation with fault injection is discussed. In [3, 7], a soft error estimation technique is discussed for microprocessors. Soft error simulation in logic circuits are also being studied and developed [11–14]. The structure of memory modules is so regular and monotonous that it is comparatively easy to treat their soft errors. Once their SERs are obtained, the values can be utilized as references. Their SERs can be obtained with field and accelerated tests experimentally [4]. These SERs of memory modules become pessimistic when they are embedded into computer systems. In obtaining the SERs of memory modules by field or accelerated tests, every single event upset (SEU) on memory modules is regarded as a critical error. This implicitly

assumes that every soft error on memory cells of a module make a computer system faulty. In computer systems which behave dynamically, memory modules are used temporally and spatially and every soft error on the memory modules does not necessarily make the computer system faulty. In this sense, the soft errors in computer systems should be treated in a different way from those in memory modules.

Accurate soft error estimation at system level is one of the issues of urgent concern. Several soft error vulnerability estimation techniques have been proposed [1, 2, 5, 10]. The authors of this paper discussed an accurate soft error estimation technique which can be adopted early in development of computer systems [10]. It is necessary that the dynamic behavior of computer systems is taken into account in order to accurately estimate the reliability of them. The time during which a memory module or a logic circuit is utilized depends on the behavior of a computer system with a program and affects its reliability. It is hard to distinguish the soft errors in memory which are or are not critical to the computer system without knowing its behavior. The complicated structure of the hierarchical memory modules which are utilized in computer systems makes soft the error estimation of them harder. In such a computer system, it is hard to judge which part of memories are active to operate the computer system. Our soft error estimation technique computes the number of soft errors on all memory modules of the computer system by cycle-accurate instruction-set simulation (ISS) with their dynamic behavior taken into account.

The soft errors on electronics are getting probable as the technology node of ICs proceeds. The reliability issues must be taken into account in the mission critical applications as well as in consumer products. In this paper, we analyze a tradeoff between reliability and performance of computer systems and also propose a novel memory hierarchy for making computer systems reliable. The recent research on the single event upsets (SEUs) on memory modules identified that the SRAM modules are getting more vulnerable than the DRAM modules. In the sense, there exists a tradeoff between the performance and the reliability of computer systems because their cache memories are made from vulnerable SRAM memories and contribute to the performance improvement. In our memory hierarchy, a cache memory is adaptively activated or deactivated for controlling performance and reliability. The cache memory is activated in the phase where the performance is required while it is deactivated in the phase where reliability is required. Switching the mode of the cache memory contributes to controlling performance and reliability of computer systems. This paper experimentally validates that switching the mode of the cache memories affects the performance and the reliability of computer systems and is effective for performance and reliability of computer systems.

2. Performance and Reliability of Computer Systems

A soft error rate (SER) is often utilized for measuring and evaluating vulnerability of a memory component. All possible single event upsets (SEUs) are considered critical to the memory component on measuring its SER. The SER is directly inapplicable to estimating vulnerability of computer systems, because they dynamically behave to use memory modules temporally and spatially. This means that some of SEUs on memory modules make the computer systems faulty and the others not. In the sense, it is pessimistic to adopt the SERs for evaluating reliability of computer systems. From the point of executing programs, it is not an SER but the number of soft errors occurred during a certain job that should be the metric for estimating the reliability of computer systems.

Several soft error vulnerability estimation techniques have been proposed [1, 2, 5, 10]. The authors of this paper recently proposed a methodology and an algorithm for estimating reliability of computer systems [10]. Their estimation methodology adopted cycle-accurate simulation to identify which part of memory is utilized spatially and temporally during executing programs. Identifying the spatial and temporal usage of memory modules contributed to more accurate estimation of reliability of computer systems. The proposed method estimated 96.7% less number of soft errors occurred during executing a program than a naive estimation method which basically calculated the reliability with the product of SERs and its runtime.

In this section, we review the relation between performance and reliability of ICs. We examined the number of soft errors during the execution of a program on a microprocessor-based system consisting of an ARM processor (ARMv4T, 200MHz), an instruction cache module,a data cache module, and a main memory module. The cache line size and the number of cache-sets are 32-byte and 32, respectively. We adopted the Least Recently Used (LRU) policy as the cache replacement policy. We evaluated the reliability of the computer system with the two write policies, write-through and write-back ones. The cell-upset rates (CUR) of both SRAM and DRAM modules are shown in Table 1. We used the cell-upset rates shown in [9] as the cell-upset rates of non-ECC SRAMs and non-ECC DRAMs and assumed that adopting an ECC circuit makes the reliability of a memory module 1000 times higher.

表 1 Cell-upset rates.

	Cell upset rates			
	[FIT/bit]		[errors/word/cycle]	
	non-ECC	ECC	non-ECC	ECC
SRAM	1.0×10^{-4}	1.0×10^{-8}	4.4×10^{-24}	4.4×10^{-28}
DRAM	1.0×10^{-8}	1.0×10^{-12}	4.4×10^{-28}	4.4×10^{-32}

Figure 1 shows vulnerability and runtime of a computer system which adopts non-ECC L1 cache memory and a non-ECC main memory. The horizontal axis indicates the number of cache ways which translates the size of the cache memory. In this memory configuration, the CUR of the non-ECC SRAM is 10k times higher than that of the non-ECC DRAM, and the SER of the non-ECC SRAM resultantly degraded the reliability of the computer system. The vulnerability increases as the size of the cache memory increases as shown in Figure 1.

Figure 2 shows vulnerability and runtime of a computer system

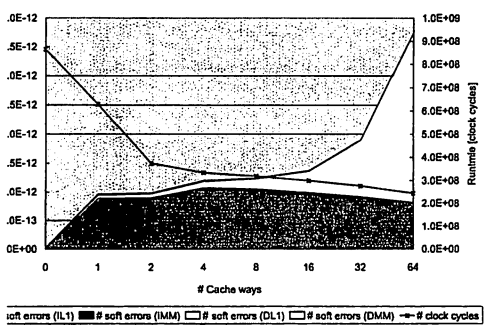

図 1 Vulnerability vs cache size (non-ECC L1, non-ECC main memory).

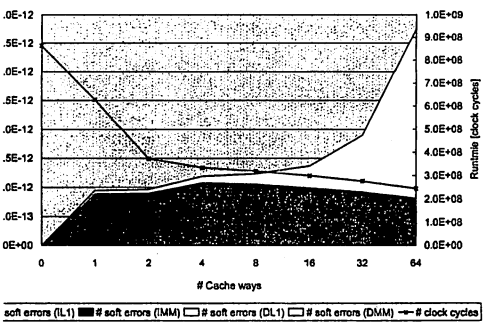

図 2 Vulnerability vs cache size (non-ECC L1, ECC main memory).

which adopts a non-ECC L1 cache memory and an ECC main memory. In this memory configuration, the CUR of a non-ECC SRAM is 1M times higher than that of a non-ECC DRAM. Comparing Figure 1 and with Figure 2, adding an ECC circuit to the main memory hardly decrease vulnerability of the computer system because vulnerability of not the main memory but the L1 cache memory is dominant in the computer system.

Figure 1 shows that the vulnerability of a non-ECC L1 cache memory is dominant in the computer system and connotes that adding an ECC circuit to the L1 cache memory would decrease vulnerability of the computer system. Figure 3 shows adding an ECC circuit to an L1 cache memory decreased much vulnerability. The CURs of the L1 cache memory and the main memory, consequently, become same as the other, that is 1.0×10^{-8} [FIT/bit]. The total SER of the main memory becomes larger than that of the L1 cache memory because the size of the main memory is larger than that of the L1 cache memory. In this memory configuration, it is deduced that the vulnerability of the main memory would be dominant in the vulnerability of the computer system. Figure 3 shows that the vulnerability of the computer system decreases as increasing the size of the cache memory increases the performance of the computer system. Higher performance decreases the vulnerability of the main memory because time the computer system use the main memory is reduced.

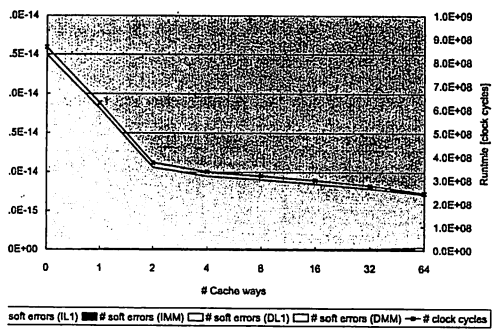

図 3 Vulnerability vs cache size (ECC L1, non-ECC main memory).

Figure 4 shows vulnerability and runtime of a computer system which adopts an ECC L1 cache memory and an ECC main memory. In this memory configuration, the SER of the cache memory is dominant in the computer system and the vulnerability of the computer system increases as the size of the L1 cache memory increases.

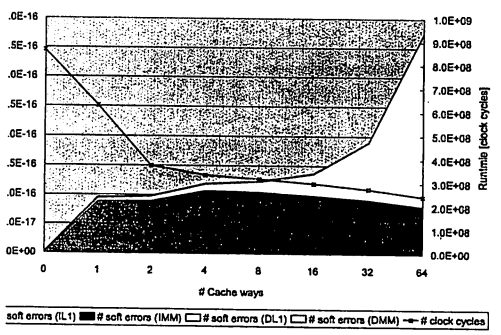

図 4 Vulnerability vs cache size (ECC L1, ECC main memory).

As shown in Figures 1, 2, and 4, there exists a tradeoff between reliability and performance of a computer system, in which the SER of a cache memory is dominant in the reliability of the computer system. The experimental examination also shows that cache memory sizing contributes to adjusting the reliability of the computer system.

3. Cache Memory Sizing Architecture

As discussed in the previous section, decreasing the size of a cache memory contributes to increasing the reliability of a computer system with some performance degradation. In this section, we propose a reliable cache memory architecture which dynamically switches its size, controls its SER, and affirmatively accepts some performance degradation for increasing its reliability. The reliable cache memory architecture is useful when the deadline time at which a program finishes is given.

Before we discuss the reliable cache memory architecture, we review the general cache memory architecture. Figure 5 shows a read mechanism for a 4-way set associative cache memory. The cache memory behaves on a read as follows. Given an address from a CPU for read, the index address of the given address selects all the corresponding lines of all cache ways. Then their tags are compared with the tag of the given address for examining whether or not the corresponding data item is on the cache memory. In the meantime, the words of the lines are selected by the block offset of the given address. The data item is finally outputted if it is on the cache memory.

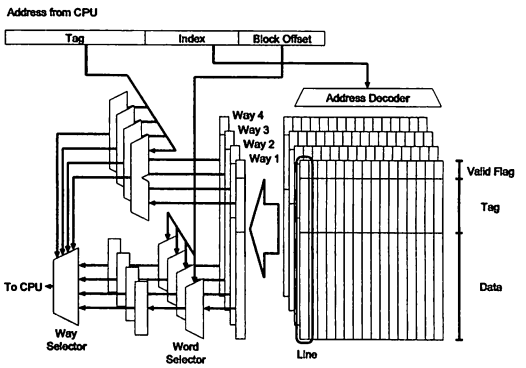

図 5 Read mechanism for 4-way set associative cache memory.

Figure 6 shows an LRU-based write mechanism for a 4-way set associative cache memory. LRU bits of each line holds the index for the least recently used cache way. The cache memory behaves on a write as follows. Given an address from a CPU for write, the index address of the given address selects all the corresponding lines of all cache ways. Then their tags are compared with the tag of the given address for examining whether or not the corresponding data item is on the cache memory. If the data item is on the cache memory, the date item is updated. Otherwise, the line of the LRU way is written out to a lower memory and the line of the data item is allocated to the line of the LRU way. The data item from the CPU finally is written on the line of the LRU way.

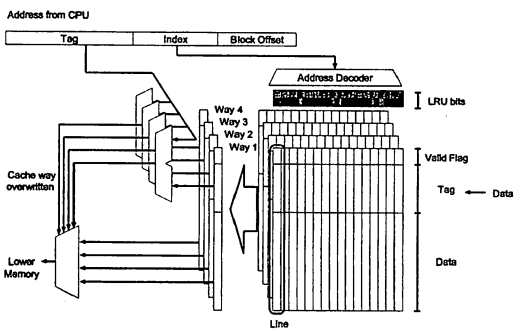

図 6 LRU-based write mechanism for 4-way set associative cache memory.

Reliable Cache Memory Architecture

As discussed in Section 2., decreasing the size of a cache memory contributes to increasing the reliability of a computer system with some performance degradation. It is relatively easy to dynamically change the size of a cache memory as shown in Figure 7. The easiest way for this is only to activate or deactivate some cache ways.

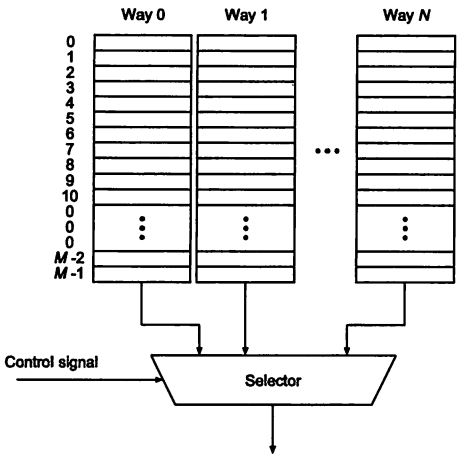

図 7 Simplified read mechanism for N-way set associative cache memory.

There are basically three approaches to activate or deactivate cache ways as follows.

Naive cache architecture

In this cache architecture, some cache ways are activated under a reliability mode while all of cache ways are activated under a performance mode as shown in Figure 8. Ideally speaking, every cache way may change into either active or inactive mode. From the viewpoint of hardware implementation, some of cache ways may change its operation mode.

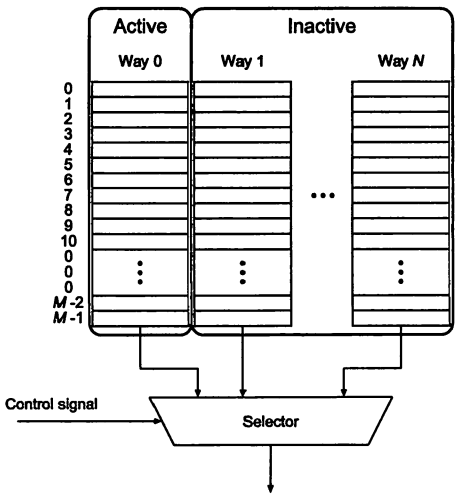

図 8 Naive mechanism for N-way set associative cache memory.

Detection-oriented cache architecture

This cache architecture is error-detection-oriented. In this cache architecture, two cache ways are regarded as a redundant pair and they have the same content as each other as shown in Figure 9. If an SEU occurs on a redundant pair of cache ways, the SEU is promptly detected and the CPU manages the SEU. The simple way the CPU manages is to halt the computer system.

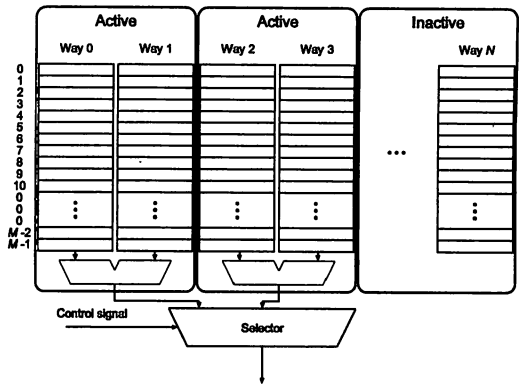

図 9 Detection-oriented mechanism for N-way set associative cache memory.

Correction-oriented cache architecture

This cache architecture is error-correction-oriented. In this cache architecture, three or more cache ways are regarded as a redundant set and they retain same content as each other as shown in Figure 10. If an SEU occurs on a redundant set of cache ways, the SEU is promptly detected and corrected by majority rule. A correct value is decided by majority among the corresponding cache ways.

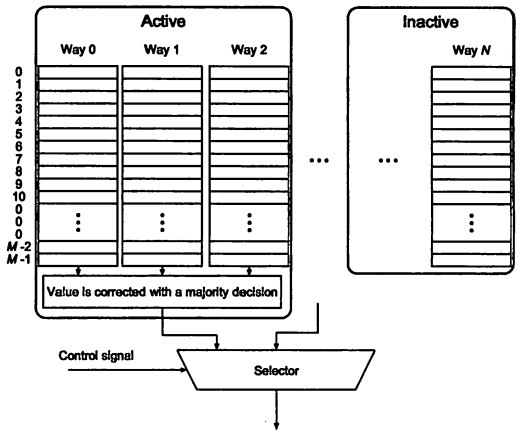

図 10 Correction-oriented mechanism for N-way set associative cache memory.

The merits and demerits of the three approaches are summarized in Table 2.

表 2 Three cache memory architectures.

	Reliability	Area Overhead	Performance	Power
Naive	low	low	high	low
Detection	middle	middle	middle	high
Correction	high	high	low	high

4. Case Study

We calculated performance and vulnerability of computer systems on switching the operation mode of cache memory, that is the performance mode or the reliability mode. We used the GNU C

compiler and debugger for ARMv4T architecture to generate address a trace. Vulnerability of the computer system was calculated with the simulation-based soft error estimation method presented in [10]. We utilized *Compress*, that is a data compression program, as a benchmark program for this experiment. The trace of a benchmark program is 100 million instructions long.

We adopted the operation mode under which a cache memory is completely disabled as the reliability mode. We also adopted the operation mode under which a cache memory is fully activated as the performance mode. Our experiment was done under several computer systems which have a cache memory whose size is 2kB, 4kB, 8kB, 16kB, 32kB, 64kB, or 128kB.

We adopted cell upset rates shown in Table 1 in Section 2. for our experiment. We used the cell-upset rates shown in [9] as the cell-upset rates of non-ECC SRAMs and non-ECC DRAMs. We assumed that a cell upset rate of a non-ECC memory is 1,000 times higher than that of an ECC memory.

We ran the program as the following procedure.

(1) Begin to run a program under the reliability mode.
(2) Switch the operation mode from the reliability mode to the performance mode.
(3) The program finishes.

In this procedure, the operation mode of a program is switched once a job.

Figures 11, 12, 13, 14, 15, 16, and 17 show vulnerability and performance of a computer system for various timing of switching the operation mode from the reliability mode to the performance mode. In the figures, accumulated area graphs show the number of soft errors occurred during the execution of the program, and the lines show performance of the computer system. In the figures, The "Switched cycle" indicates when the computer system switches its mode from the reliability mode to the performance mode. The figures clearly show that there exists a tradeoff between reliability and performance on the cache memory sizing architecture.

5. Conclusion

In this paper, we analyzed a tradeoff between reliability and performance of computer systems. It was experimentally found that there exists a tradeoff between reliability and performance of computer systems in which an SER of a cache memory is more critical than that of a main memory.

We proposed three cache memory architectures: (i) Naive cache architecture, (ii) detection-oriented architecture, and (iii) correction-oriented cache architecture. Each cache architecture has its own merits and demerits. System designers should choose their one of them for the requirements of their products.

We experimentally examined correlation between runtime and vulnerability of a computer system. The experimental results imply that the operation mode under which a cache memory operates should be adaptively determined. Urgent programs should be executed under the performance mode while non-urgent ones should be executed under the reliability mode. The operation mode of a program may be adaptively switched in accordance with the situation around the computer system. This aspect is important in RTOS systems. Controlling reliability and performance by an RTOS is one of our future work.

文　献

[1] H. Asadi, V. Sridharan, M. B. Tahoori, and D. Kaeli. Vulnerability analysis of L2 cache elements to single event upsets. In *Proceedings of the conference on Design, Automation and Test in Europe Conference (DATE)*, Munich, Germany, March 2006. IEEE.

[2] A. Biswas, P. Racunas, R. Cheveresan, J. Emer, S. S. Mukherjee, and R. Rangan. Computing architectural vulnerability factors for address-based structures. In *Proceedings of IEEE International Symposium on Computer Architecture (ISCA)*, pages 532–543, Madison, WI, USA, June 2005. IEEE.

[3] V. Degalahal, S. Cetiner, F. Alim, N. Vijaykrishnan, K. Unlu, and M. J. Irwin. SESEE: soft error simulation and estimation engine. In *Proceedings of MAPLD International Conference*, Washington, D.C., USA, October 2004. IEEE.

[4] H. Kobayashi, K. Shiraishi, H. Tsuchiya, M. Motoyoshi, H. Usuki, Y. Nagai, K. Takahisa, T. Yoshiie, Y. Sakurai, and T. Ishizaki. Soft errors in SRAM devices induced by high energy neutrons, thermal neutrons and alpha particles. In *Technical digest of IEEE International Electron Devices Meetings*, pages 337–340, San Francisco, December 2002. IEEE.

[5] X. Li, S. V. Adve, P. Bose, and J. A. Rivers. SoftArch: An architecture level tool for modeling and analyzing soft errors. In *Proceedings of IEEE International Conference on Dependable Systems and Networks (DSN)*, Yokohama, Japan, June 2005. IEEE.

[6] T. C. May and M. H. Woods. Alpha-particle-induced soft errors in dynamic memories. *IEEE Transactions on Electron Devices*, Vol. 26:2–9, 1979.

[7] S. S. Mukherjee, J. Emer, and S. K. Reinhardt. The soft error problem: an architectural perspective. In *Proceedings of IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 243–247, San Francisco, CA, USA, 2005. IEEE.

[8] M. Rebaudengo and M. Violante M. S. Reorda. An accurate analysis of the effects of soft errors in the instruction and data caches of a pipelined microprocessor. In *Proceedings of the conference on Design, Automation, and Test in Europe - Volume 1*, pages 10602–10607, Munich, Germany, 2003. IEEE Computer Society.

[9] C. W. Slayman. Cache and memory error detection, correction and reduction techniques for terrestrial servers and workstations. *IEEE Transactions on Device and Materials Reliability*, 5(3):397–404, September 2005.

[10] M. Sugihara, T. Ishihara, K. Hashimoto, and M. Muroyama. A simulation-based soft error estimation methodology for computer systems. In *Proceedings of IEEE International Symposium on Quality Electronic Design*, pages 196–203, San Jose, CA, USA, March 2006. IEEE.

[11] Y. Tosaka, H. Ehara, M. Igeta, T. Uemura, H. Oka, N. Matsuoka, and K. Hatanaka. Comprehensive study of soft errors in advanced CMOS circuits with 90/130 nm technology. In *Technical digest of IEEE International Electron Devices Meetings*, pages 941–948, San Francisco, CA, USA, December 2004. IEEE.

[12] Y. Tosaka, H. Kanata, T. Itakura, and S. Satoh. Simulation technologies for cosmic ray neutron-induced soft errors: models and simulation systems. *IEEE Transactions on Nuclear Science*, 46:774–780, June 1999.

[13] Y. Tosaka, S. Satoh, and T. Itakura. Neutron-induced soft error simulator and its accurate predictions. In *Proceedings of IEEE International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pages 253–256, Cambridge, MA, USA, September 1997. IEEE.

[14] Y. Tosaka, S. Satoh, and H. Oka. An accurate and comprehensive soft error simulator NISES II. In *Proceedings of IEEE International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pages 219–226, Munich, Germany, September 2004. IEEE.

[15] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. Patel. Characterizing the effects of transient faults on a high-performance processor pipeline. In *Proceedings of IEEE International Conference on Dependable Systems and Networks (DSN)*, pages 61–70, Florence, Italy, July 2004. IEEE.

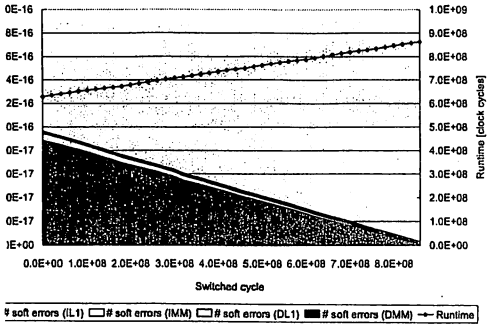

図 11 Vulnerability and performance on switching non-cache to direct mapped cache.

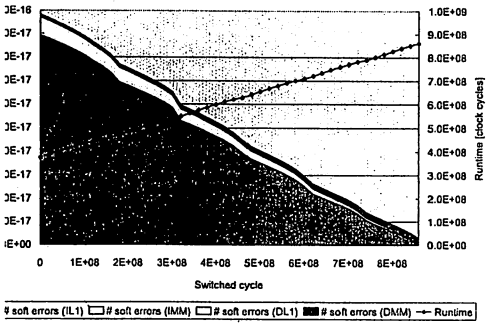

図 12 Vulnerability and performance on switching non-cache to 2-way set associative cache.

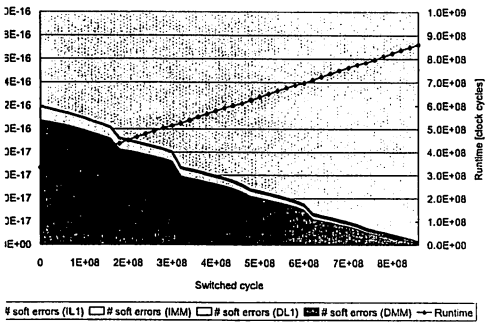

図 13 Vulnerability and performance on switching non-cache to 4-way set associative cache.

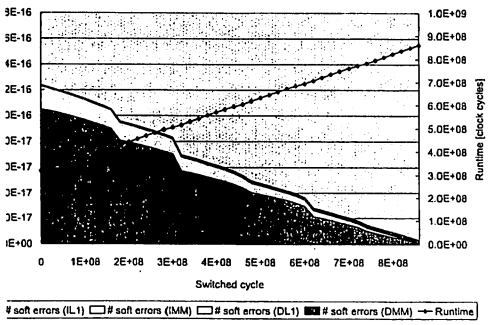

図 14 Vulnerability and performance on switching non-cache to 8-way set associative cache.

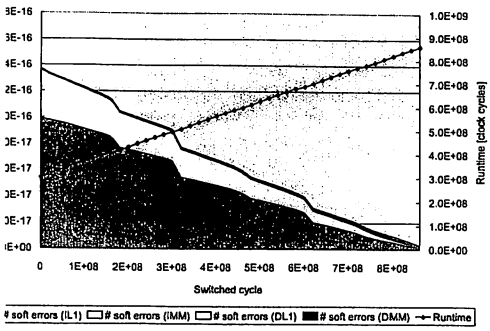

図 15 Vulnerability and performance on switching non-cache to 16-way set associative cache.

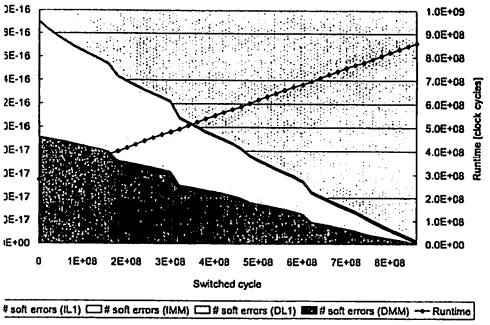

図 16 Vulnerability and performance on switching non-cache to 32-way set associative cache.

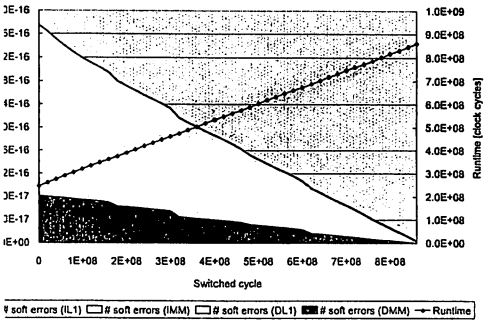

図 17 Vulnerability and performance on switching non-cache to 64-way set associative cache.