

Self-Adaptive Data Caches for Soft-Error Reliability

Shuai Wang, Jie Hu, and Sotirios G. Ziavras

Abstract—Soft-error induced reliability problems have become a major challenge in designing new generation microprocessors. Due to the on-chip caches' dominant share in die area and transistor budget, protecting them against soft errors is of paramount importance. Recent research has focused on the design of cost-effective reliable data caches in terms of performance, energy, and area overheads, based on the assumption of fixed error rates. However, for systems in operating environments that vary with time or location, those schemes will be either insufficient or overdesigned for the changing error rates. In this paper, we explore the design of a self-adaptive reliable data cache that dynamically adapts its employed reliability schemes to the changing operating environments thus to maintain a target reliability. The proposed data cache is implemented with three levels of error protection schemes, a monitoring mechanism, and a control component that decides whether to upgrade, downgrade, or keep the current protection level based on the feedback from the monitor. Our experimental evaluation using a set of SPEC CPU2000 benchmarks shows that our self-adaptive data cache achieves similar reliability to a cache protected by the most reliable scheme, while simultaneously minimizing the performance and power overheads.

Index Terms—Data cache, reliability, reliable system design, self adaptation, soft error.

I. INTRODUCTION

Ionizing radiation induced single-event upsets (SEUs), also known as soft errors, in semiconductor memories have been recognized for a long time as a major reliability issue in electronic systems [1], [2]. Due to their large share of the transistor budget and die area, on-chip caches suffer from a significantly higher soft-error rate (SER) than on-chip combinational logic at the current and near future technologies [3]. A corrupted data value once readout from a cache may crash the subsequent computation/communication, external memory, or storage systems, leading to overall system failure or program inaccuracy. Consequently, protecting information integrity in on-chip caches is of paramount importance to the provision of reliable computing in new generation microprocessors.

Despite the fact that most of the previous work has studied tradeoffs between performance, energy, area overheads, and achieved cache reliability for their proposed schemes [4]–[6], their proposals are mainly targeting at cost-effective reliable cache designs with the assumption of fixed operating environments, e.g., supply voltage, operating temperature, system location, etc. However, many of the environmental conditions may change dynamically during the operation of the system under consideration, which makes the system more or less vulnerable to soft errors. For example, dynamic-voltage scaling [7] based on-chip power/energy optimization schemes may increase the chip vulnerability when the supply voltage is scaled down [8]. Systems deployed within mobile objects, such as transportation vehicles, ships, or airplanes, may experience very different intensities of (cosmic ray induced) neutron flux, and the SERs may change with the latitude or altitude during travel. For example, the cosmic ray flux can be $1000\times$

Manuscript received October 18, 2007; revised January 21, 2008 and March 12, 2008. This paper was recommended by Associate Editor K. Chakrabarty.

The authors are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: sw63@njit.edu; jhu@njit.edu; ziavras@njit.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCAD.2008.925789

more intense at the altitude of the commercial flight than at the sea level [1], [8]. Under such a situation, a conventional reliable design targeting a typical error rate will be either insufficient to provide the required reliability at harsh environments or overdesigned for low error-rate environments. Furthermore, microprocessor-based system design is a widespread design methodology that helps to expedite the design process, reduce the design cost, and improve the reliability of the designed system. While these systems may be designed for different application purposes, implying different reliability requirements and operating environments, off-the-shelf commercial microprocessors must be designed with sufficient flexibility and adaptability, in terms of reliability schemes, to support various applications. To address this new requirement, a self-adaptive reliable design will be of great value to future microprocessors.

In this paper, we propose a self-adaptive reliable data cache (SA-RDC) that dynamically adjusts its reliability scheme to the changing operating environments such that the reliability design target is guaranteed during its operation while performance and energy overheads are minimized simultaneously. Specifically, we provide three levels of soft error protection with increasing strength and recovery efficiency. To track variations in the operating environments that reflect on SERs, a mechanism is developed to dynamically monitor the detected error rate within a given sampling window. After each sampling window, the adaptive controller retrieves this information from a special counter, based on which the controller: 1) predicts the undetected error rate or silent data corruption (SDC) rate and the performance/energy overhead for correcting detected errors; 2) decides whether to upgrade, keep, or downgrade the current protection level such that the SDC rate is kept below the preset threshold while the performance and energy overheads can be further optimized; and 3) reconfigures the data cache protection strategies according to the decision made in 2). Our experimental evaluation with a set of SPEC CPU2000 benchmarks shows that the self-adaptive data cache scheme achieves similar reliability to the most robust scheme while maintaining the performance and energy overheads of a lightweight scheme.

The rest of this paper is organized as follows. The experimental framework is presented in Section II. Section III presents reliable data cache designs and their architectural vulnerability factor (AVF) analysis. In Section IV, we elaborate on the design of the proposed self-adaptive data cache and present the analysis of the experimental results. We point out the limitation of this paper in Section V and draw conclusions in Section VI.

II. EXPERIMENTAL FRAMEWORK

A. Microprocessor Model

For our experimental evaluation of the proposed SA-RDC, a performance and power simulator that models a contemporary high-performance microprocessor similar to Alpha 21264 [9] is developed based on SimpleScalar V3.0 [10] and Wattch [11]. Table I shows the detailed configuration of the simulated microprocessor. For experimental evaluation, we use a set of SPEC CPU2000 benchmarks compiled for the Alpha instruction set architecture with “peak” tuning. We use the reference input sets and the early single simulation point specified by SimPoint [12] for this paper. For parity checking, we assume a 10% energy overhead (of a cache access) for encoding/decoding a word data [5].

B. Error Model and Soft Error Injection

To evaluate the error resilience of our schemes, we conducted soft error injection during the execution-driven simulation. The soft error

TABLE I
PARAMETERS FOR THE SIMULATED MICROPROCESSOR

Processor Core	
Int/FP issue queue	20/15 entries
Load/Store Queue	64 entries
Active list (ACL)	80 entries
Int/FP Register File	80/72 registers
Datapath width	4 instructions per cycle
Function Units	4 IALU, 1 IMULT/IDIV
	2 FALU, 1 FMULT/FDIV/FSQRT
	2 MemPorts
Branch Predictor	
Branch Predictor	Tournament predictor with a 4K meta-table, a 4K bimodal predictor table, and a 2-level gshare predictor with 12-bit history
	2048-entry, 2-way BTB, and 32-entry RAS
Memory Hierarchy	
L1 I/D-Cache	64KB, 2 ways, 64B blocks, 2 cycle latency
L2 U-Cache	4MB, 8 ways, 128B blocks, 12 cycle latency
Memory	225 cycles first chunk, 12 cycles rest
TLB	Fully-associ., 128 entries, 30-cycle miss penalty
Technology Parameters	
Vdd	0.9V
Clock frequency	3GHz
Technology	70nm

injection flips one bit or multiple bits in a selected cache line. Since the multiple-bit error rate is several orders of magnitude lower than the single-bit error rate [13], we assume a single-bit error model in this paper. Therefore, our error injection scheme simulates SEUs in the data cache. At each clock cycle, a uniformly distributed random function is called to locate a cache line and a specific bit within that line. Then, an error is injected with a given probability (e.g., 10^{-6} shown as e-6), i.e., single-bit SER per selected bit. As a general way to perform architectural-level error injection [13], we assume accelerated error rates in order to expose the error behavior and evaluate the reliability of the system.

III. RELIABLE DATA CACHES BUILT UPON BYTE-LEVEL PARITY CODING

Our paper has shown that at the same area overhead the byte-level parity coding provides the same error detection capability as typical (72, 64) ECC codes in the data cache under extremely high SERs. In the meantime, byte-level parity coding can be seamlessly integrated into the data cache without modifying the regular cache access procedure. Therefore, in our reliable data caches, we choose byte-level parity coding as the error detection scheme. For this paper we also assume that the L2 cache is protected by some means of ECC coding and is error free.

Note that systems targeting different applications and different operating environments may have very different reliability requirements. For a given SER in a target environment, we consider three levels of soft error protection schemes for the data cache: reliable data cache with byte-level parity coding (RDC-P), RDC-P with in-cache replication (RDC-P-ICR), and RDC-P-ICR with early writeback and clean cache line invalidation (RDC-P-ICR-EWB-CCI). We elaborate on and evaluate these three schemes in the following sections.

A. Limits of Conventional Reliable Data Caches

1) *RDC-P Data Cache*: Basically, the error detection ability of parity coding applied to byte-level data is sufficient to handle relatively low error rates, such as e-6 or e-5. Therefore, our first level of error protection scheme only uses the parity coding to protect each byte in the data cache. When some data in the data cache is read by CPU or written back to the L2 cache, the parity checking is performed for each byte of that data item. Any single-/odd-bit error in a byte will be detected and signaled for error handling. If the erroneous

data is in a clean cache line, it can be recovered by invalidating the current cache line and refetching the data copy from the L2 cache. If the error happens in a dirty line, we assume that hardware exception will be generated and the operating system (OS) will take over for error recovery based on some checkpointing schemes [14]. We optimistically assume 1000 cycles for the OS to handle the error situation. Note that in real cases this OS recovery latency may be much longer.

The RDC-P scheme works best at relatively low-error-rate conditions. However, as the error rate increases, RDC-P scheme will suffer from two major problems. First, a high error rate may introduce a large number of double-bit or multibit errors that cannot be detected by the parity scheme. Fig. 1(c) shows that when the error rate reaches e-3, the undetected error rate in the RDC-P scheme is dramatically increased to 1.8%. This SDC presents the potential of crashing the program execution or resulting in erroneous output [15]. Second, the OS recovery overhead in the RDC-P scheme becomes significant as the error rate increases. Fig. 1(a) shows that the performance loss of RDC-P scheme at error rate e-6 is negligible. This number increases to 0.23% and 2.3% at error rates of e-5 and e-4, respectively. A dramatic performance loss of 25.2% is observed, when the error rate is extremely high ($e - 3$). Similarly, the total energy consumption of the data cache and L2 cache increases by more than 70% as the error rate reaches e-3, as shown in Fig. 1(b). This huge performance and energy overhead at high error rates is mainly due to the frequent OS invocations in the RDC-P scheme.

2) *RDC-P-ICR Data Cache*: To cost-effectively recover from error-corrupted dirty lines at relatively high error rates, we borrow the ICR scheme [5], which duplicates the dirty cache line within the cache, as the alternative solution to alleviate the performance overhead. When erroneous data are detected in a dirty line of the RDC-P-ICR, the recovery scheme will first search its replica and check the parity of the replica. If the replica passes the parity check, it is assumed to be error free and is used to recover the corrupted primary data copy. Therefore, a successful recovery from the in-cache replica can significantly reduce the recovery overhead. However, if the replica is also error corrupted and detected by the parity checking, RDC-P-ICR use the same strategy, OS recovery, as in RDC-P. Fig. 1(a) shows that at high error rates such as e-4 and e-3, compared to the RDC-P scheme, RDC-P-ICR improves performance by 2.2% and 33.5%. However, due to cache line replication and replica maintenance, in general, the power consumption of the data cache and L2 cache in RDC-P-ICR is higher than in RDC-P. Fig. 1(b) shows that the energy number for RDC-P-ICR is around 19.5% higher than for RDC-P when the error rate is lower than e-3. However, the picture changes when the error rate reaches e-3, where RDC-P consumes significantly higher energy in the data cache and L2 cache than RDC-P-ICR. A noticeable feature of the RDC-P-ICR scheme is that it performs consistently well in terms of performance and energy consumption, across vastly different error rates.

3) *RDC-P-ICR-EWB-CCI Data Cache*: While the ICR scheme solves performance and energy issues related to the software (OS) recovery strategy at high error rates, the parity coded data cache still suffers from high SDC rates. Fig. 1(c) shows that the SDC rate reaches 1.8% and 2.1% in the RDC-P and RDC-P-ICR schemes at the error rate of e-3. We consider two techniques, namely EWB and CCI for reducing the cache vulnerability to soft errors and, therefore, the SDC rate.

Dead-Time-Based EWB: Previous work [16]–[19] has shown that the time between the last write and the replacement (WPL) of a cache data item contributes the largest part to the vulnerability factor. The dead-time-based EWB scheme proposed in [13] and [16] writes back a dirty cache line after it has not been accessed for a certain time

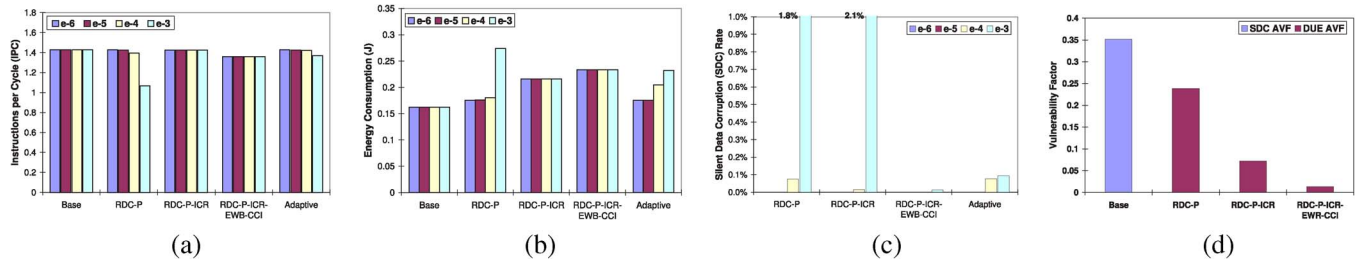


Fig. 1. Performance, energy consumption, and SDC rate comparison between the SA-RDC and fixed RDC schemes under error injection and an AVF comparison for fixed RDC schemes. (a) Performance. (b) Energy consumption. (c) SDC rate. (d) AVF of data caches.

interval. This scheme can reduce the WPL component while avoiding the dramatic increase in accesses to the L2 cache.

4) *CCI*: After applying EWB for optimizing the WPL phase, the RR time (the time between the first read and last read in a clean cache line) arises as the major part in the vulnerability factor calculation of the data cache [19]. Based on the observation that the majority of RRs (around 90%) have a short time interval (≤ 0.5 K cycles) and account for only a small percentage (around 10%) of the accumulated RR vulnerable intervals, a recent work [19] proposed a technique called CCI to invalidate clean lines after a certain time interval. We adopt this CCI technique for the RR phase optimization.

Combined Scheme: Our RDC-P-ICR-EWB-CCI scheme is a combination of the parity coding, ICR, dead-time-based EWB, and CCI. In our evaluation, we choose a 1 K-cycle interval for both deadness prediction and clean line invalidation. We use a similar implementation as the cache decay scheme [20]. Each cache line maintains a two-bit local counter which is ticked every 256 cycles by a global counter. Both the dead-time-based EWB scheme and the CCI scheme use the same local counter. The dirty bit controls whether a simple invalidation or an EWB shall be performed when the local counter saturates. The local counter is reset to zero upon any access to the cache line.

By reducing the time of cache lines staying in the vulnerable WPL and RR phases, the RDC-P-ICR-EWB-CCI scheme significantly decreases the possibility of data being corrupted by soft errors and being loaded by the CPU or written back to the L2 cache. Consequently, the occurrence of SDC will also be dramatically reduced. Fig. 1(c) shows that at the error rate of e-3, RDC-P-ICR-EWB-CCI achieves a significantly low SDC rate of 0.01%, compared to 1.8% and 2.1% for RDC-P and RDC-P-ICR, respectively. On the other hand, due to periodic EWB and CCI, the performance of RDC-P-ICR-EWB-CCI is about 4.6% lower than RDC-P-ICR, and the energy consumption of the data cache and the L2 cache increases by 8.6% over RDC-P-ICR.

B. Computing the AVFs

Different from error-injection-based reliability analysis, AVF analysis [21] is independent of the lower device-level error rate (usually in FITs, failures in 1 000 000 000 h). We use a similar lifetime model as in [22] to compute and compare the AVFs for different reliable data caches discussed in this section. A single-bit error model is also assumed in this AVF study. Fig. 1(d) shows the AVF numbers for different data caches, an average across the selected benchmarks. The AVF of a base data cache is around 35.1%, all contributing to SDC AVF. With parity protection, all single-bit errors can be detected and some of them (in clean lines) can be recovered by the L2 cache. Thus, RDC-P reduces the architecturally correct execution (ACE) time and converts the SDC AVF into detected unrecoverable error (DUE) AVF. As shown in the figure, the DUE AVF is reduced to 23.9%. By duplicating dirty lines, RDC-P-ICR is capable of recovering errors in dirty lines provided their duplicates are error free, which improves the

cache reliability with a significantly reduced DUE AVF of 7.2%. With EWR and CCI, the ACE time of the data cache can be further reduced and the RDC-P-ICR-EWB-CCI achieves an impressive AVF of 1.4%.

IV. SA-RDC

A. Why Self-Adaptive Scheme?

For the three levels of reliable data caches we proposed in the previous section, each could be the best cost-effective scheme at a particular error rate. For systems with stringent performance and energy constraints and changing operating environments (i.e., SERs), a fixed reliable scheme will be either insufficient (at harsh environments) or too costly (at low-error-rate environments) in terms of performance and energy overheads. Thus, a self-adaptive scheme will be of critical importance to meet the reliability requirements as well as performance/energy constraints.

B. Soft-Error Monitoring Mechanism

Due to the limits of the conventional reliable data caches, we propose a SA-RDC based on the three levels of protection schemes, RDC-P, RDC-P-ICR, and RDC-P-ICR-EWB-CCI. The SA-RDC scheme dynamically monitors the number of detected errors during the parity check and logs the error occurrences into a special error counter during each monitoring window. Based on the value of the error counter at the end of each monitoring window and the currently applied protection scheme, the SA-RDC predicts the error rate that the system is currently experiencing and determines whether to change the protection scheme or just maintain the current scheme. The error counter is reset to zero at the beginning of each new monitoring window. From this paper, a monitoring window of 100 K cycles is chosen, the minimum window size to effectively change the protection scheme. Notice that the selection of the window size is strongly related to the error rates. For instance, in the case of more realistic low error rates, the window size should be much larger.

C. Control of Self Adaptation

After fixing the size of the monitoring window, we also need to determine the thresholds of error occurrences for triggering the different schemes. Therefore, we simulated the different protection schemes at different error rates and profiled the number of errors detected in each 100 K monitoring window. Based on the analysis of our profiling results, the threshold for upgrading from the RDC-P scheme to the RDC-P-ICR scheme is four errors. It means that if more than four errors are detected in the last window with RDC-P protection scheme, we predict that the current error rate might be larger than e-4 and the protection scheme needs to be changed to RDC-P-ICR in order to reduce the performance loss. If the current protection scheme is RDC-P-ICR and more than 16 errors are detected in the last monitoring

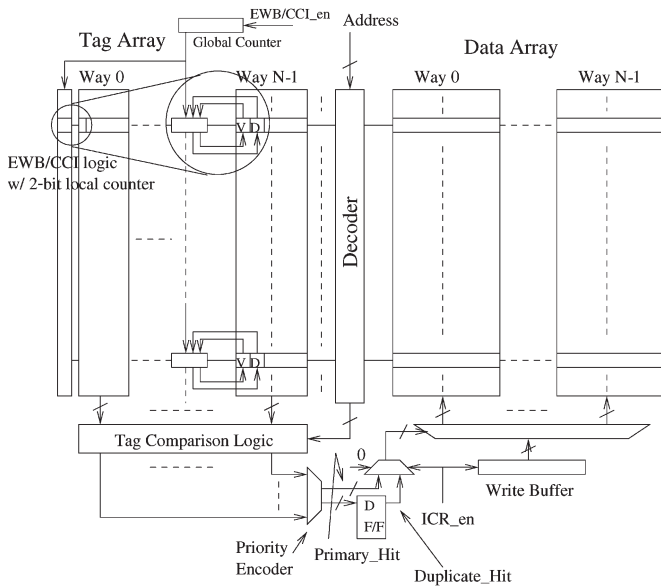


Fig. 2. Microarchitectural schematic of the proposed SA-RDC.

window, we predict that the current error rate is quite high such that double- or multibit errors will occur within a single byte. Therefore, the RDC-P-ICR-EWB-CCI protection scheme should be invoked in order to reduce the SDC.

To downgrade from the RDC-P-ICR-EWB-CCI scheme to the RDC-P-ICR scheme, we use the history information of last three consecutive monitoring windows. If there is no error detected in the last three monitoring windows, we degrade the protection scheme to RDC-P-ICR. From RDC-P-ICR to RDC-P, we use two consecutive monitoring windows. If no error is detected in the previous two monitoring windows under the RDC-P-ICR protection scheme, we downgrade to the RDC-P scheme. Notice that the threshold is also strongly related to the error rates and the monitoring window size.

D. Microarchitecture of the SA-RDC

In order to implement our SA-RDC, the hardware should support all three schemes and the mechanism to switch between them. In fact, the hardware requirement of our SA-RDC is similar to that of the RDC-P-ICR-EWB-CCI scheme, since in the three chosen schemes, the higher level (more reliable) scheme is built on top of the lower level (less reliable) scheme, which makes it easy to integrate them together. Fig. 2 shows the microarchitecture-level schematic of our SA-RDC. To support ICR, the priority encoder in the tag match logic is slightly augmented to generate both Primary_Hit and Duplicate_Hit way numbers. The figure shows that during a write operation the Duplicate_Hit way number is delayed by one cycle, and data will be buffered in the write buffer once ICR is enabled such that writing the duplicate copy is performed in the following cycle. To support EWB and CCI, an N -bit global counter ticked by the clock signal and a per line two-bit local counter ticked by the global counter every 2^N cycles are introduced. The local counter is reset to zeros once the cache line is accessed. If the local counter saturates, either EWB is performed (if both valid and dirty bits are set) or CCI is performed (if valid bit is set and dirty bit is cleared), then the local counter is reset to zero. Notice that the global counter is enabled by signal EWB/CCI_en, and it is the control unit of the SA-RDC that generates the ICR_en and EWB/CCI_en signals to shut down or turn on ICR and/or EWB/CCI to adaptively choose a protection level. Thus, the hardware overhead of supporting SA-RDC should be at the same level as the RDC-P-ICR-

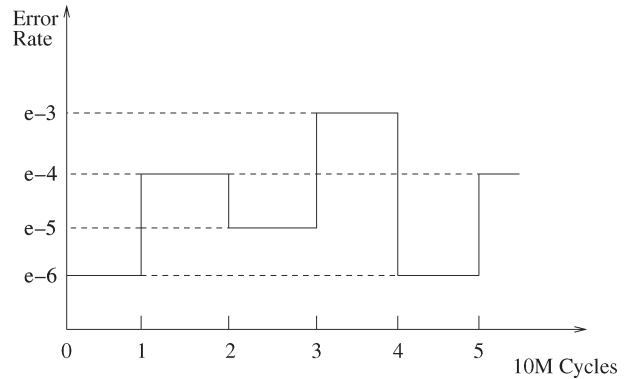


Fig. 3. SER profile to simulate the changing error rate.

EWB-CCI scheme. Moreover, because all three schemes are based on parity coding, the parity bit and the encoding/decoding units are shared by the three schemes to reduce the hardware cost and complexity of the SA-RDC.

E. Evaluation of the SA-RDC Scheme

Fig. 1 shows that our SA-RDC can self-adapt to the best scheme at a fixed error rate. For example, at the error rate e-3, SA-RDC performs nearly the same as RDC-P-ICR-EWB-CCI, which means it automatically tunes itself to the RDC-P-ICR-EWB-CCI scheme in order to reduce the SDC rate. Furthermore, in order to evaluate the efficiency of our SA-RDC scheme, we also simulate SA-RDC under changing operation environments by varying the soft error injection rate. We randomly construct a SER profile as shown in Fig. 3, which is similar to the one in [23]. The profile simulates a pattern of varying SER at four levels between e-6 and e-3. Each error rate lasts for 10 M cycles, then the pattern repeats.

Fig. 4(a) shows the performance between our SA-RDC scheme and fixed RDC schemes for the varying error rate pattern shown in Fig. 3. The large performance loss in RDC-P is mainly due to the huge recovery overhead at the error rate of e-3. Similarly, the performance degradation in RDC-P-ICR-EWB-CCI is caused by periodic EWB and CCI. RDC-P-ICR consistently performs better than other fixed RDC schemes since the error rate has less impact on the performance of RDC-P-ICR. Our SA-RDC scheme adaptively adjusts the protection schemes to the detected error rate. It avoids to apply the RDC-P scheme at high error rates or the RDC-P-ICR-EWB-CCI scheme at low error rates, thus eliminating unnecessary performance loss. Fig. 4(a) shows that SA-RDC achieves a performance within the 0.8% of the best scheme RDC-P-ICR.

However, one may argue that if SA-RDC always favors choosing the RDC-P-ICR scheme over others, it could still achieve the performance shown in Fig. 4(a). To illustrate that the SA-RDC scheme does adapt itself by invoking different schemes at different error rates during the simulation, we present a comparison of the energy consumption of the data cache and L2 cache for all the schemes in Fig. 4(b). Overall, the RDC-P scheme has the lowest energy consumption. By choosing the RDC-P scheme at low error rates and RDC-P-ICR-EWB-CCI only at high error rates, SA-RDC achieves a much lower energy consumption than RDC-P-ICR or RDC-P-ICR-EWB-CCI, which is only 3.7% higher than RDC-P.

Finally, we present the SDC rate comparison in Fig. 4(c). Obviously, neither RDC-P nor RDC-P-ICR works well as the SER varies between e-6 and e-3. On the average, the SDC rate is 1% and 0.5% for RDC-P and RDC-P-ICR, respectively. In contrast, the SDC rate is almost zero in RDC-P-ICR-EWB-CCI due to the effective EWB and CCI. In the

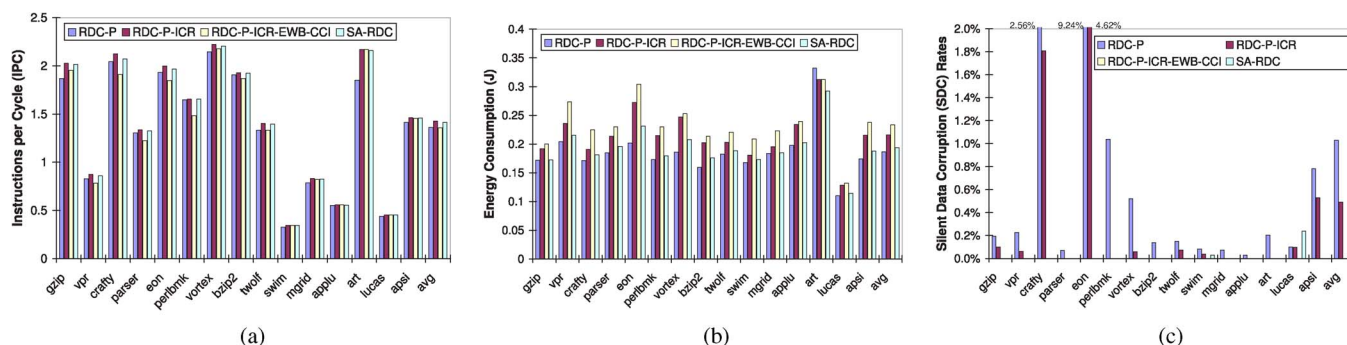


Fig. 4. Performance, energy consumption, and SDC rate comparison between the SA-RDC and fixed RDC schemes. (a) Performance. (b) Energy consumption. (c) SDC rate.

meantime, RDC-P-ICR-EWB-CCI suffers from the worst performance loss and energy overhead, as shown in Fig. 4(a) and (b). On the other hand, our SA-RDC achieves a significantly reduced SDC rate, 0.02% on the average, while simultaneously minimizing the performance and energy overheads. These results confirm us that the SA-RDC scheme is very effective for systems operating under changing environments.

V. LIMITATIONS OF THIS STUDY

First, as we discussed in Section II-B, the error rates we use in this paper are extremely high compared to real ones because of limitations in a simulation study. Second, the dynamic soft error model in Fig. 3 is also quite different from real world situations, where the changing speed and rate are much lower than assumed. We choose this error model just for simulation purposes. To make our self-adaptive scheme work in the real world situations, we need to change the monitoring window size and the threshold accordingly. Third, the cache access latency may increase by incorporating the protection schemes. However, a detailed study of this timing impact is out of the scope of this paper. We assumed that the cache access latency is maintained at the same level. We will measure this kind of hardware overhead in a future study.

VI. CONCLUSION

We propose in this paper a new methodology for designing reliable systems that can self-adapt to the changing operating environments, by adjusting the applied reliability scheme to be the best match to the current erroneous situation. We exemplify this methodology by presenting the design of a SA-RDC that supports three levels of protection schemes targeting at different error rates with different performance and energy impacts. The monitoring component of the adaptive design continuously monitors the error incidents within the preset windows and sends the error information to the control component. The latter decides whether to replace the current reliability scheme or not. Our simulation results show that this SA-RDC can effectively take the best advantage of each provided metascheme while avoiding their deficiencies. This limited study conveys a very encouraging message in the area of self-adaptive reliable system design. We plan to investigate this design methodology at the microprocessor and system levels in our future work.

REFERENCES

- [1] J. F. Ziegler *et al.*, "IBM experiments in soft fails in computer electronics (1978–1994)," *IBM J. Res. Develop.*, vol. 40, no. 1, pp. 3–18, Jan. 1996.
- [2] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 305–316, Sep. 2005.
- [3] P. Shivakumar *et al.*, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. Int. Conf. Dependable Syst. Netw.*, Jun. 2002, pp. 389–398.
- [4] S. Kim and A. Somani, "Area efficient architectures for information integrity in cache memories," in *Proc. Int. Symp. Comput. Architecture*, May 1999, pp. 246–255.
- [5] W. Zhang *et al.*, "ICR: In-cache replication for enhancing data cache reliability," in *Proc. Int. Conf. Dependable Syst. Netw.*, 2003, pp. 291–300.
- [6] V. Sridharan *et al.*, "Reducing data cache susceptibility to soft errors," *IEEE Trans. Dependable Secure Comput.*, vol. 3, no. 4, pp. 353–364, Oct.–Dec. 2006.
- [7] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *Proc. ISLPED*, Aug. 1998, pp. 76–81.
- [8] T. J. O’Gorman *et al.*, "Field testing for cosmic ray soft errors in semiconductor memories," *IBM J. Res. Develop.*, vol. 40, no. 1, pp. 41–50, Jan. 1996.
- [9] R. E. Kessler, "The Alpha 21264 microprocessor," *IEEE Micro*, vol. 19, no. 2, pp. 24–36, Mar./Apr. 1999.
- [10] D. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," *Comput. Sci. Dept., Univ. Wisconsin, Madison, WI, Tech. Rep. 1342*, 1997.
- [11] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. High-Performance Comput. Architecture*, 2000, pp. 83–94.
- [12] T. Sherwood *et al.*, "Automatically characterizing large scale program behavior," in *Proc. 10th ASPLOS*, Oct. 2002, pp. 45–57.
- [13] L. Li *et al.*, "Soft error and energy consumption interactions: A data cache perspective," in *Proc. ISLPED*, 2004, pp. 132–137.
- [14] N. J. Wang and S. J. Patel, "ReStore: Symptom-based soft error detection in microprocessors," *IEEE Trans. Dependable Secure Comput.*, vol. 3, no. 3, pp. 188–201, Jul.–Sep. 2006.
- [15] C. Weaver *et al.*, "Techniques to reduce the soft error rate of a high-performance microprocessor," in *Proc. Int. Symp. Comput. Architecture*, 2004, pp. 264–275.
- [16] W. Zhang, "Computing cache vulnerability to transient errors and its implication," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Oct. 2005, pp. 427–435.
- [17] G. Asadi *et al.*, "Balancing performance and reliability in the memory hierarchy," in *Proc. IEEE Int. Symp. Performance Anal. Syst. Softw.*, Mar. 2005, pp. 269–279.
- [18] H. Asadi *et al.*, "Vulnerability analysis of L2 cache elements to single event upsets," in *Proc. DATE*, Mar. 2006, pp. 1–6.
- [19] S. Wang, J. Hu, and S. G. Ziavras, "On the characterization of data cache vulnerability in high-performance embedded microprocessors," in *Proc. Int. Conf. Embedded Comput. Syst.: Architectures, Modeling, Simul.*, Jul. 2006, pp. 14–20.
- [20] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power," in *Proc. Int. Symp. Comput. Architecture*, 2001, pp. 240–251.
- [21] S. S. Mukherjee *et al.*, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proc. ACM/IEEE Int. Symp. Microarchitecture*, Dec. 2003, pp. 29–40.
- [22] A. Biswas *et al.*, "Computing architectural vulnerability factors for address-based structures," in *Proc. Int. Symp. Comput. Architecture*, Jun. 2005, pp. 532–543.
- [23] L. Li *et al.*, "Adaptive error protection for energy efficiency," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2003, pp. 2–7.