Soft Error and Energy Consumption Interactions: A Data Cache Perspective [*]

Lin Li, Vijay Degalahal, N. Vijaykrishnan, Mahmut Kandemir, Mary Jane Irwin
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802, USA
{lili, degalaha, vijay, kandemir, mji}@cse.psu.edu

ABSTRACT

Energy-efficiency and reliability are two major design constraints influencing next generation system designs. In this work, we focus on the interaction between power consumption and reliability considering the on-chip data caches. First, we investigate the impact of two commonly used architectural-level leakage reduction approaches on the data reliability. Our results indicate that the leakage optimization techniques can have very different reliability behavior as compared to an original cache with no leakage optimizations. Next, we investigate on providing data reliability in an energy-efficient fashion in the presence of soft-errors. In contrast to current commercial caches that treat and protect all data using the same error detection/correction mechanism, we present an adaptive error coding scheme that treats dirty and clean data cache blocks differently. Furthermore, we present an early-write-back scheme that enhances the ability to use a less powerful error protection scheme for a longer time without sacrificing reliability. Experimental results show that proposed schemes, when used in conjunction, can reduce dynamic energy of error protection components in L1 data cache by 11% on average without impacting the performance or reliability.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability and Fault-Tolerance

General Terms

Design, Reliability

Keywords

Soft Error, Energy-Efficiency, Data Cache

[*]This work was supported in part by grants from MARCO/DARPA GSRC-PAS, NSF Grants 0103583, 0082064, 0130143, 0202007 and CAREER Awards 0093082, 0093085.

1. INTRODUCTION

With shrinking feature sizes, the supply voltages and in turn threshold voltages of transistors have reduced. Though lowering supply voltage helps to reduce the dynamic power consumption, the consequent decrease in threshold voltage has increased the leakage energy [6]. As cache memories constitute a significant portion of the transistor budget of current microprocessors, leakage reduction for cache memories has been of particular importance. It has been estimated that leakage energy accounts for 30% of L1 cache energy and 70% of L2 cache energy for a 0.13 micron process [14].

Another consequence of the technology scaling is the smaller supply voltages and reduced capacitive values of the circuit nodes. This has raised reliability concerns due to the increased susceptibility to soft errors [15, 11, 9]. Soft errors or transient errors are circuit errors caused due to excess charge carriers induced primarily by external radiations, such as alpha particles and high energy neutrons. While these errors cause an upset event, the circuit itself is not damaged. In memory, these can cause a particular node to charge or discharge and thus cause a bit flip. This is particularly true for SRAM cells used in caches [9].

The focus of this work is on understanding the interactions between soft errors and the energy consumption behavior of the data caches. First, we investigate the impact of leakage energy optimizations on the soft errors. Next, we investigate the energy implications due to providing protection against soft errors.

There have been several efforts [3, 8, 2, 10] at the architectural level reducing the cache leakage energy when it is idle. Drowsy cache [8] controls the leakage of memory cells by using dynamic voltage scaling (DVS) while maintaining the values of cache blocks. In contrast, the cache decay scheme [12] completely turns off the cache blocks, losing the stored data, when they are not accessed for a long period of time. Different designs of low power cache memories can potentially have different immunity to soft errors due to supply voltage changes and circuit structure [7]. These differences in soft error vulnerability are important as they influence the complexity of error detection and correction circuitry employed in on-chip memories. We believe that the issues of energy efficiency and reliability are interrelated and should not be studied in isolation.

Consequently, we study the influence of soft errors on a cache with no leakage energy optimizations and on caches employing the drowsy cache and cache decay technique for energy savings. In order to perform the experiments, we have built a simulation framework on top of the SimpleScalar simulator [5] that permits the injection of soft errors into the cache and also allows the modeling of different cache interleaving schemes and error detection/correction schemes. Our experimental results show that the drowsy cache, while saving significant leakage energy also incurs a significant increase in soft errors as compared to the original cache configuration. In contrast, the cache decay can reduce both the leakage energy and the amount of effective soft errors. Consequently using the decay approach may be more effective when considering both reliability and energy optimizations together.

Next, we focus on how to protect the cache against soft errors in an energy efficient fashion. We propose an adaptive scheme that protects dirty and clean data using different complexities of error protection mechanisms. The clean data are protected by single error detection codes and data is retrieved from the secondary caches to recover from soft errors. However, for dirty data, a more powerful single error correction scheme is used to recover from the error. This adaptive scheme is in contrast to current commercial caches that use a uniform error protection scheme for all data. This is also different from other schemes provided for energy-efficient data protection through duplication of data [19], or protecting only the frequently used cache lines [17]. The approach involving cache line duplication is orthogonal to our approach and has been investigated using a cache using both ECC and parity. In contrast, our work focuses on providing different strength of protection for different cache blocks. In addition, we also decrease the effective number of dirty lines by using an early-write-back policy. This approach strikes a balance between additional overhead of frequent L2 accesses in a write-through cache that has no dirty lines and the additional overhead for using a more powerful error correction scheme for dirty lines in a write back approach for a longer period. Our experimental results show that using the adaptive protection with early-write-back can reduce the dynamic energy required for error protection in L1 data cache by 11% on average without impact the performance.

The remainder of this paper is organized as follows. In the next section, we introduce the soft error rate model used in our work and the implementation of soft error injection in cache memory. Section 3 studies the influence of two low power cache designs, drowsy cache and cache decay, on the soft error rate. The proposed adaptive error protection scheme with early-write-back policy is introduced and evaluated in Section 4. Section 5 provides conclusions.

2. SOFT ERROR MODEL

2.1 Soft Error Rates

The primary source of radiations that induce soft errors can be classified as alpha particles from the packaging materials, high energy neutrons from cosmic radiations, and the interaction of cosmic ray thermal neutron with the ^{10}B isotope of Boron [4]. While the elimination of ^{10}B using new process technologies and removing impurities that cause alpha particle radiation have helped in reducing the soft error rates, the smaller nodal capacitances and supply voltages have resulted in higher soft errors. The smaller charge is a particular concern because particles of lower energy occur far more frequently than particles of higher energy in atmospheric radiation [20]. Consequently, more particles can cause soft errors as CMOS device sizes decrease.

Table 1: Soft error rate (per cycle).

	Normal	Low Voltage	R/W
Single-bit Error	1E-7	1E-6	5E-7
Double-bit Error	1E-9	1E-8	5E-9
Multi-bit Error	1E-11	1E-10	5E-11

For a soft error to occur at a specific node in a circuit, the collected charge Q at that particular node should be more than $Q_{critical}$. If the charge generated by a particle strike at a node generates a charge that is more than the $Q_{critical}$, the pulse so generated is latched on, and results in a bit flip. This concept of critical charge is generally used to estimate the sensitivity of Soft Error Rate (SER). In [9], a method to estimate the SER in CMOS SRAM circuits was developed. In this model an exponential dependence of SER on critical charge was shown as

$$SER \propto N_{flux} * CS * exp(-\frac{Q_{critical}}{Q_s}) \qquad (1)$$

Where N_{flux} is the intensity of the Neutron Flux, CS is the area of the cross section of the node and Q_s is the charge collection efficiency (it's strongly dependent on doping). $Q_{critical}$ is proportional to the node capacitance and the supply voltage.

The SERs used in our experiments are shown in Table 1. The second column gives the SERs of the cache under the normal supply voltage, 1.0V in our work. The third column gives the SERs of the cache under the lower supply voltage used in drowsy cache mode, 0.3V in our work. The last column gives the SERs when a read/write operation on the target block, which reflects the fact that the cache blocks in read or write incur higher soft error rates.

Usually soft error rates are thought of as single bit upsets. But as the technology scales, the nodal cross section decreases, hence multiple bit upsets (MBU) are also probable. In [18], the magnitude difference between single and double error rates was found to differ from three orders to one order of magnitude based on the operating conditions such as supply voltage. To reflect this observation, we used two orders of magnitude difference between the error rates of single, double and multiple bit errors. Also as shown by Equation 1, error rates for low power mode was fixed as one order of magnitude higher than the corresponding high power mode, as the error rate is exponential dependent on the supply voltage [7]. During read and write operations, the error can manifest in either the SRAM cell or the peripheral circuits like sense amps. Also, when the wordline in SRAM is asserted, charge sharing occurs and hence reduces the $Q_{critical}$ of the cell [15]. To account for these facts the error rate during read and write is increased by 5X from the corresponding steady state error rate. Since it is very time consuming to experimentally simulate for multiple hour executions, when current soft error rates will manifest in actual errors, we used the base probability of 1E-7 errors. While significantly higher than the error rate observed in current technology, it is still effective to mimic soft error problems that would occur in long running applications. And we do use accurate relative numbers of the different soft error cases to provide a realistic evaluation.

2.2 Soft Error Injection

In this work, we implement a random soft error injection in the cache memories using the SimpleScalar [5] simulator. A random variable generated by the simulator along with the SERs provided in Table 1 is used to determine whether a soft error happens and the number of bits influenced by the error. In addition, three independent random variables are used to decide the set, way, and bit of block in the cache at which the external radiation hits. All these random variables are generated based on pseudo-random numbers using the linear congruential algorithm. These three random variables guarantee an even distribution of soft errors in a cache. Whether a soft error happens and how many bits error happens depends on not only the strength of external radiation, but also the state of the cache block. Cache blocks in different state have different susceptibility to external radiations.

3. INFLUENCE OF LEAKAGE OPTIMIZATIONS ON SOFT ERROR RATES

In this section, we investigate the susceptibility to soft errors of two popular architectural level leakage optimizations: drowsy cache and cache decay.

3.1 Drowsy Cache

Drowsy cache is based on controlling the leakage by using dynamic voltage scaling (DVS) while using the standard 6T SRAM cell structure [8]. This method makes use of the fact that to retain a value in the SRAM cell, the source voltage of the cell can be just about 1.5 times of V_t. Thus, for a 70-nanometer technology based SRAM cell that normally operates at 1.0V, the voltage can be reduced to up to 0.3V when accesses are not required while still retaining the data. Substantial leakage energy saving can be gained when placing the cache line in this low voltage drowsy state [8]. However, one cycle penalty is incurred when accessing a drowsy cache line, as the supply rails have to be restored to 1.0V before a read or write operation. For voltages less than 0.3V it was noted that the cache line lost its values. In the drowsy cache scheme, all the cache lines are periodically switched to a lower voltage assuming a periodic change in the working set of the application.

When in the drowsy state, the cache cells due to the reduced supply voltage are more susceptible to soft errors compared to the normal state. This observation can be deduced from Equation 1. The parameters influencing SER, except $Q_{critical}$, shown in the Equation 1 are the same since the underlying circuit structure remains the same. $Q_{critical}$ reduces superlinearly with the supply voltage (since capacitance is also a function of the supply voltage). Thus, the use of DVS provides an interesting opportunity for trade off between leakage reduction and soft error immunity.

3.2 Cache Decay

In [12], Kaxiras et al. present a leakage energy reduction technique, called Cache Decay, for cache memories. Cache Decay exploits the temporal behavior of cache blocks to reduce leakage consumption. This technique is based on the idea that a cache block that is not used for a sufficiently long period of time can be considered dead. More specifically, with each cache block, they associate a small 4-state FSM (finite state machine). The FSM steps through these states as long as the cache block is not accessed. When the last state is reached, the cache block is turned off completely after committing any changes back to higher levels of memory hierarchy. This can be implemented using the same circuit fabric as in drowsy cache but by changing the sleep mode voltage to 0V instead of 0.3V. Since the data is completely destroyed, soft errors in the idle state is not a concern for this leakage control mode. Furthermore, since cache lines are evicted from the cache close to their anticipated dead times, the time for which they are exposed to soft errors in the normal mode is also smaller as compared to the original cache. In an original cache, the entries will be evicted only when replaced by another cache line when employing the commonly used write-back approach.

Table 2: Benchmarks and execution cycles.

	Execution Cycle		Execution Cycle
bzip	793,768,890	parser	992,353,260
gap	731,340,328	perlbmk	1,455,161,111
gcc	1,521,913,064	twolf	1,661,660,355
gzip	769,888,583	vortex	1,758,572,035
mcf	2,147,483,647	vpr	945,542,126

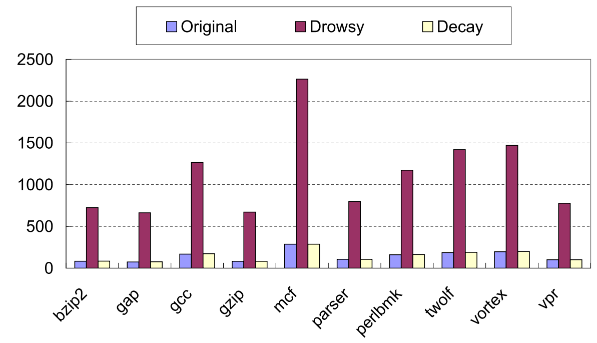

Figure 1: Number of soft errors when using the original cache with no leakage optimizations, drowsy caches and cache decay.

3.3 Error Behavior of Different Approaches

We implement the soft error injection, drowsy cache, and cache decay in the simulator SimpleScalar 3.0 [5]. Our benchmarks are from SPEC CINT2000 [1]. Each benchmark is first fast forwarded 300 million instructions and then simulated 1 billion instructions. Table 2 gives the total number of cycles taken by each benchmarks. In drowsy cache, the entire data cache was put into a low voltage mode for every 2000 cycles. In cache decay scheme, the decay interval of L1 data cache is set as 10K. The default configuration of the simulator with a 16KB, 32 byte, 1-way L1 instruction cache, a 16KB, 32 byte, 4-way L1 data cache and a unified 256KB, 64 byte, 4-way L2 cache was used in the experiments.

Figure 1 shows the total number of soft errors injected in the L1 data cache in original cache, drowsy cache, and decay cache for each benchmark. Since the original cache and decay cache use the same memory cells, they have the same SER and, therefore, incur the same number of soft errors. But in the drowsy cache, a large portion of cache blocks that operate at a lower voltage when in drowsy mode are more susceptible to soft errors (See table 1). Therefore, the total number of soft errors induced in the drowsy cache is significantly more than that in the normal cache.

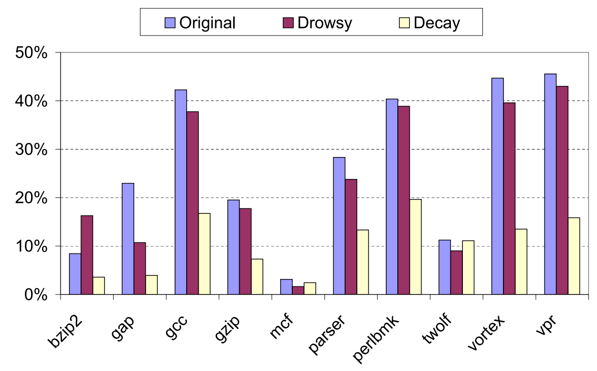

Figure 2: Percentage of soft errors read into datapath.

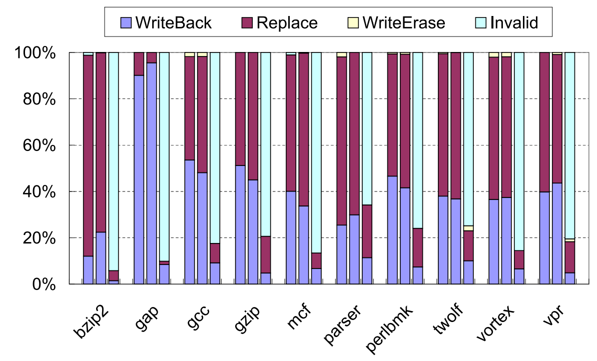

Figure 3: Distribution of how L1 cache blocks with soft errors propagate in the memory hierarchy (from left to right columns are original, drowsy, and decay cache).

When a soft error happens, it happens either on a valid cache block or an invalid cache block. Later, the valid cache block with soft error can be read into datapath, overwritten by new data, replaced by new cache block if it is clean, or written back to L2 cache if it is dirty. It is obvious that not every injected soft error will impact the correctness of the operation performed by the system. Therefore, we call the soft errors being read into datapath and written back to the L2 cache as the effective soft errors as they propagate the error beyond the L1 cache.

Figure 2 shows the percentage of injected soft errors being read into datapath. We observe that the percentage of errors that propagate to the datapath is significantly less when employing the decay cache. This happens because the decay cache increases the number of invalid cache blocks by applying cache block shut down. All errors that occur in invalid blocks do not propagate to the datapath. In contrast, the original and drowsy modes do not perform any additional tasks to increase the number of invalid cache blocks. Also, note that the absolute number of errors that propagate to the datapath for the drowsy cache is much higher (on the average 6.7 times more) than that of the original cache.

Figure 3 shows how the error in the L1 cache block affected by soft errors will propagate in the memory hierarchy. The problematic case occurs when the dirty data is written back to the L2 cache. We observe that this portion is 43.3%, 43.4% and 7.1% on the average for the original, drowsy and decay caches, respectively. Consequently, decay cache not only reduces the leakage energy consumption for the cache block which is not accessed anymore, but also reduces the chance of those blocks being exposed to external radiation. Specifically, when doing shutdown, if there is a dirty block it is written back early, reducing the amount of time it will be exposed to a soft error.

Table 3: Drowsy cache with bit interleaving.

	without Interleaving			with Interleaving		
	1-bit error	2-bit error	Multi-bit error	1-bit error	2-bit error	Multi-bit error
bzip2	716	9	0	734	0	0
gap	656	7	0	670	0	0
gcc	1246	20	0	1286	0	0
gzip	663	8	0	679	0	0
mcf	2232	32	0	2296	0	0
parser	789	10	0	809	0	0
perlbmk	1159	14	0	1187	0	0
twolf	1400	19	0	1438	0	0
vortex	1450	20	0	1490	0	0
vpr	767	10	0	787	0	0

We also observe that double-bit errors also occur, especially in the case of drowsy cache lines that operate at a lower voltage. These double bit errors are those caused by particle strikes with an impact area spanning more than a single memory cell and not due to two independent single events upsets that we found to be rare. Handling multi-bit errors will necessitate more powerful error correction schemes than single bit correction or parity-based recovery for clean data. A more effective optimization for such cases is by employing bit interleaving [13] that interleaves bits from different cache block in a row. This technology distributes the impact of a multi-bit soft error on a single cache block into multiple single bit soft errors on multiple cache blocks. The result of the number of soft errors that occur in the drowsy cache with and without bit interleaving scheme are shown in the Table 3. Based on the results in Table 3, most double-bit errors are converted into single bit errors. Therefore, the double bit errors and multi-bit errors are reduced significantly with the increase in single bit errors. This alleviates the burden of more powerful error detection and error correction circuits.

4. ENERGY-EFFICIENT SOFT ERROR PROTECTION

In this section, we consider on how to protect the cache blocks in an energy efficient fashion.

4.1 Adaptive Error Protection

Error detection or correction codes are a way of introducing redundant information in the form a codeword. Error protection codes are widely used to improve the memory reliability. Most simple form of coding involves adding a single bit to store the parity (odd or even) of each data word. But this can only detect the errors and not correct them. Another commonly used scheme is SEC-DED. (38, 32) Hamming code and odd-weight code belong to this class of code. The most important feature of this code is its fast encoding and error detection in the decoding phase. This code can correct the single bit errors and detect double bit errors with extra bits overhead. Therefore, the more powerful the error protection coding scheme is, the higher reliability it guaran-

Table 4: Power consumption and delay of different coding schemes.

	Power (mW)		Delay (ns)	
	Coding	Decoding	Coding	Decoding
Parity	6.9232	7.2239	1.41	1.41
SEC	14.4871	26.2962	1.45	2.66

tees and the more energy consumption and area overhead it incurs.

Compared to clean blocks, dirty blocks in L1 data cache have no corresponding duplicate copies in the L2 cache (or memory, if no L2 cache is present) when using the commonly used write-back approach. This means that data in the dirty blocks may be damaged permanently, in case one of the bits flip due to soft errors. In contrast, data with soft errors in the clean blocks can be recovered from the duplicate copy in the L2 cache (We do not consider the rare error in both L1 and L2 cache). Consequently, dirty blocks need to be protected more strongly than clean data.

Error protection schemes are widely used in the cache memories, such as parity and ECC coding. However, they employ the same protection for the cache lines. This can be wasteful as we have noticed that clean and dirty blocks can operate with different strength of error protection schemes while providing the desired protection. Our adaptive protection scheme treats these cache lines differently. In order to eliminate 1-bit soft error, dirty blocks need Single Error Correction (SEC) coding at least. But the clean blocks may only use Single Error Detection (SED) coding and then get the correct data from the L2 cache. Since error cases are rare, the performance and energy overhead of additional L2 accesses are insignificant.

Since the energy consumption of SEC and SED are quite different, the proposed scheme provides significant savings over a scheme that uses single error correction approach for all cache lines. Table 4 gives the power value and delay of coding and decoding for SED (implemented by parity) and SEC (implemented by Hamming (38,32) code) custom implemented using 250 nm libraries. In order to support the adaptive error protection, the dirty bit is used to determine whether SED or SEC is used. An additional 6 bits are associated with each cache line to support the more powerful SEC. In the case where only SED is required (dirty bit not set), the additional 5-bits are supply gated to reduce leakage and only the single bit is used for parity.

4.2 Early-Write-Back Policy

Next, we explore an optimization to decrease the amount of time that single error correction needs to be employed. This can be achieved by reducing the duration for which cache lines remain in the dirty state. Traditional cache designs employ the write-back policy, which keep dirty blocks in L1 data cache as long as possible and write the data in this dirty block back to L2 cache until this block is replaced by another block. Write-back policy significantly reduces the number of writes to L2 cache. At the same time, it makes the dirty blocks vulnerable to soft errors for a long time. Intel Pentium® M processor uses write-back policy in L1 data cache.

Another common write policy is write-through, which writes the data back to L2 cache whenever it is changed in L1 cache. Intel Itanium® 2 processor uses write-through policy

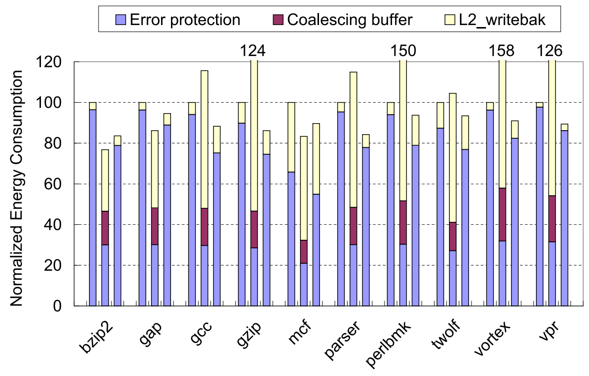

Figure 4: Energy consumption of different schemes (from left to right columns are Scheme 1, 2, and 3).

in L1 data cache. This method keeps the blocks in L1 cache in clean mode, but it significantly increases the number of L2 cache accesses. Therefore, a coalescing write-buffer [16] is always employed with write-through cache to merge the writebacks to L2 cache. But the total number of L2 cache accesses of write-through cache is still significantly larger than that of write-back cache.

We propose a new write policy, early-write-back, which writes the data back to L2 cache after a fixed time gap from last write operation. In our experiments, we use the 10K cycles as the time period. This means when 10K cycles pass after the last write operation, the dirty block is written back to L2 cache, if not evicted earlier. This method keeps the blocks in clean mode longer and does not increase the number of write to L2 significantly. The implementation of the time period monitoring is achieved in a fashion similar to that used in the decay cache.

4.3 Evaluation

We implement three different error protection schemes for evaluation. Scheme 1 uses a L1 data cache with SEC protection for all cache blocks and uses a write-back policy. Scheme 2 uses a write-through L1 data cache with a coalescing write-buffer. The coalescing write-buffer contains four entries and each entry is the same size as the cache line[1]. SED protection is used for all cache lines and SEC protection is used for coalescing write-buffer. The last one is our approach that combines the adaptive error protection scheme and the early-write-back policy.

Figure 4 shows the energy consumption of these three different schemes that provide equal protection to soft errors. All these schemes have equivalent error behavior but exhibit different energy behavior for providing the protection. They differ in the energy required for the error encoding and decoding with cache line accesses, any additional L2 accesses incurred because of the differences in the write policies, and error encoding and decoding with coalescing write-buffer accesses in Scheme 2. Energy consumption for accesses to coalescing write-buffer is very small and is omitted in our simulation.

We observed that Scheme 2 focuses on minimizing the energy consumption for error coding protection since there

[1]The design issues and tradeoffs for coalescing write-buffer with different parameters are beyond the discussion of this paper.

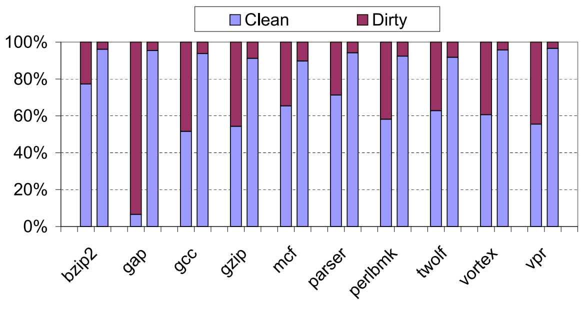

Figure 5: Distribution of clean and dirty blocks for Scheme 1 (left) and Scheme 3(right).

are no dirty blocks. However, it incurs a significant number of additional L2 accesses. In seven cases, the total energy consumption of Scheme 2 is larger than those of Scheme 1 and 3. The average energy overhead of Scheme 2 is 14% over Scheme 1. In contrast, Scheme 1 minimizes the energy required for additional L2 accesses while requiring significant energy for using SEC for all cache blocks. Our approach balances between previous two schemes and achieves the minimum overall total energy consumption. The average of energy reduction of Scheme 3 is 11% as compared to Scheme 1.

Figure 5 shows the time distribution of blocks in clean mode and dirty mode. It is obvious that all blocks are in clean mode under write-through policy. It also can be observed that dirty blocks are around 56% under write-back policy for most benchmarks. In comparison, the early-write-back significantly increases blocks in the clean mode, around 94%. Increasing the overall percentage of clean blocks results in more read operations accessing clean blocks. Consequently, many of the SEC operations translate to the less power consuming SED operations when employing early-write-back with the adaptive protection scheme. Our results also show that our technique incurs little additional performance penalty (less than 0.1%) as compared to the write-back scheme used in Scheme 1.

5. CONCLUSION

With dramatic scaling in feature size of VLSI technology, both energy-efficiency and reliability are becoming very important criteria in system designs. In this work, we first study the influence of two low power cache design, drowsy cache and cache decay, on the soft error rate. Results show that the drowsy cache, while saving significant leakage energy, also incurs a significant increase in soft errors as compared to the original cache configuration. In contrast, the cache decay can reduce both the leakage energy and the amount of effective soft errors. Then we propose an adaptive error protection scheme with early-write-back policy in L1 data cache for the purpose of energy efficient error protection. Experimental results show that proposed scheme can reduce dynamic energy of error protection components in L1 data cache by 11% in average without impact the performance.

6. REFERENCES

[1] SPEC CPU2000 benchmark. http://www.spec.org/.

[2] A. Agarwal, H. Li, and K. Roy. DRG-cache: a data retention gated-ground cache for low power. In *Proc. of the 39th Design Automation Conference*, pages 473–478, 2002.

[3] N. Azizi, A. Moshovos, and F. N. Najm. Low-leakage asymmetric-cell SRAM. In *Proc. of ISLPED*, pages 48–51, 2002.

[4] R. Baumann. The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction. In *Digest of International Electron Devices Meeting*, pages 329–332, 2002.

[5] D. C. Burger and T. M. Austin. The SimpleScalar tool-set, Version 2.0. Technical Report 1342, Dept. of Computer Science, UW, June 1997.

[6] J. A. Butts and G. S. Sohi. A static power model for architects. In *Proc. of the 33rd International Symposium on Microarchitecture*, pages 191–201, 2000.

[7] V. Degalahal, N. Vijaykrishnan, and M. J. Irwin. Analyzing soft errors in leakage optimized SRAM designs. In *Proc. of 6th International Conference on VLSI Design*, 2003.

[8] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: simple techniques for reducing leakage power. In *Proc. of the 29th annual International Symposium on Computer Architecture*, pages 148–157, 2002.

[9] P. Hazucha and C. Svensson. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. *IEEE Transactions on Nuclear Science*, 47(6), 2000.

[10] Z. Hu, P. Juang, P. Diodato, S. Kaxiras, K. Skadron, M. Martonosi, and D. Clark. Managing leakage for transient data: decay and quasi-static 4T memory cells. In *Proc. of ISLPED*, pages 52–55, 2002.

[11] T. Karnik, B. Bloechel, K. Soumyanath, V. De, and S. Borkar. Scaling trends of cosmic ray induced soft errors in static latches beyond 0.18u. In *Digest of Technical Papers of Symposium on VLSI Circuits*, pages 61–62, 2001.

[12] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. In *Proc. of the 28th annual International Symposium on Computer Architecture*, pages 240–251, 2001.

[13] J. Maiz, S. Hareland, K. Zhang, and P. Armstrong. Characterization of multi-bit soft error events in advanced SRAMs. In *Digest of International Electron Devices Meeting*, 2003.

[14] M. Powell, S. Yang, B. Falsafi, K. Roy, and N. Vijaykumar. Reducing leakage in a high-performance deep-submicron instruction cache. *IEEE Transactions on Very Large Scale Integration Systems*, 9(1):77–89, 2001.

[15] N. Seifert, D. Moyer, N. Leland, and R. Hokinson. Historical trend in alpha-particle induced soft error rates of the Alpha^{TM} microprocessor. In *Proc. of 39th International Reliability Physics Symposium*, pages 259–265, 2001.

[16] K. Skadron and D. W. Clark. Design issues and tradeoffs for write buffers. In *Proc. of the Third International Symposium on High-Performance Computer Architecture*, pages 144–155, 1997.

[17] A. K. Somani and S. Kim. Area efficient architectures for information integrity checking in cache memories. In *Proc. of International Symposium on Computer Architecture*, pages 246–256, 1999.

[18] F. Wrobel, J.-M. Palau, M.-C. Calvet, O. Bersillon, and H. Duarte. Simulation of nucleon-induced nuclear reactions in a simplified SRAM structure: scaling effects on SEU and MBU cross sections. *IEEE Transactions on Nuclear Science*, 48(6):1946–1952, 2001.

[19] W. Zhang, S. Gurumurthi, M. Kandemir, and A. Sivasubramaniam. ICR: in-cache replication for enhancing data cache reliability. In *Proc. of the International Conference on Dependable Systems and Networks*, 2003.

[20] J. Ziegler. Terrestrial cosmic ray intensities. *IBM Journal of Research and Development*, 40(1):19–39, 1996.