# Reliability Effects of Process and Thread Redundancy on Chip Multiprocessors

Dakai Zhu
Department of Computer Science
University of Texas at San Antonio
San Antonio, TX, 78249
dzhu@cs.utsa.edu

Hakan Aydin
Department of Computer Science
George Mason University
Fairfax, VA 22030
aydin@cs.gmu.edu

## I. Introduction and Background

The phenomenal performance gains in successive computer technologies have been obtained at the cost of drastic increases in power densities. This fact promoted energy to a first-class system resource, and energy-aware computing has recently become a major research area. At the same time, with the continued scaling of CMOS technologies and reduced design margins, VLSI circuits have become more susceptible to transient faults that are induced by energic particles (e.g., neutrons and alpha particles), and today, reliability concerns are pronounced more strongly for all computing systems [7]. The widely popular energy management technique, *dynamic voltage scaling (DVS)* has been shown to have direct and negative effects on the system reliability due to the increased transient fault rates [2], [11]. Therefore, there is an interesting tradeoff between system reliability and energy efficiency. For reliable systems that have limited energy budgets, exploring *energy-efficient fault tolerance* techniques becomes a necessity.

Transient faults (also called soft errors), caused by high-energy particles in computing systems, have been studied extensively, especially for memory sub-systems due to the fact that it is relatively easy to detect and model such errors for memory circuits. Error detection/correction schemes (such as *parity* and SEC-DED codes) have been proposed to enhance the data integrity. However, such techniques require significant space redundancy and complex error-correcting circuits increase memory access times, limiting their usage, especially for L1 caches [2]. Moreover, a recent model predicts that, with technology advancements and reduced feature sizes, the transient fault rate in combinational logic circuits will be comparable to that of memory elements [9].

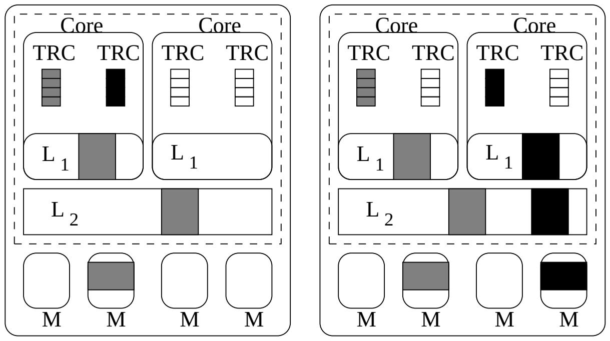
*Simultaneous multithreading (SMT)* [10] and *chip multiprocessor (CMP)* [5] architectures were originally proposed to increase the system performance. Recently, they have been also explored for fault tolerance purposes, mainly to enhance system reliability through their inherent hardware redundancies [3], [8]. The main idea is to create/run a duplicated thread, on the same or a different core, simultaneously to detect and/or recover from transient faults. With the support of the operating system, one can also migrate/group threads to execute them on a certain core for managing power density in a system [6]. Very recently, an architecture-level power-efficient fault tolerance scheme is proposed to exploit redundant cores in multicore systems, where the redundant executions for verification are performed at lower frequencies for energy savings [7]. By utilizing the idle cores to duplicate part of computation in array-intensive applications, [1] demonstrates the tradeoff among performance, power and reliability.

## II. Thread vs. Process Duplication

However, all previous work has focused on *thread level duplication (TLD)*, where the duplicated threads share common data/code segments. Although error-checking circuits in memory cells could detect (and possibly correct) a single transient fault, multiple transient faults could create corrupt data blocks in cache or main memory [2]. These, in turn, can affect duplicated threads and result in a system failure.

In this work, we illustrate the trade-off between system reliability and energy consumption with different redundancy granularities (specifically, TLD and *process-level duplication (PLD)*). Moreover, by taking into account the effects of DVS on transient fault rates [11], the challenging problem of exploring CMP architectures for both reliability and energy efficiency is identified and possible research directions are discussed.

As an example, consider an application that is duplicated with two threads running on a dual-core CMP system. In Figure 1, the dotted rectangle represents the CMP and each core has two *thread running contexts (TRC)*. Suppose that two duplicated threads, represented by the *shaded* and *darkened* TRCs, respectively, are scheduled to run in the same core (Figure 1a). We can put the second core and unused memory blocks (shown as *blank*) to a low-power sleep state for energy savings, [2]). However, in this case, any undetected transient faults in the memory sub-system (i.e., the *shaded* L1 cache which is in use, L2 cache or main memory blocks that are referenced) could lead to an error for *both* threads. Consequently, we may have a *false positive* (for the case where two threads form a duplex system to detect faults) or a failure (for the case where sanity check is performed at the end of the application for fault detection).



a. Thread level duplication b. Process level duplication

Fig. 1. Computation duplication using threads vs. processes

Assume that the probability of having transient fault(s) in data blocks in use is given by $\rho_m$. Similarly, $\rho_c$ denotes the probability of having transient fault(s) affecting the execution of one thread on the chip. For the case where sanity checks are used for fault detection, the probability of failure for the TLD approach $\rho_{TLD}$ when executing the application will be:

$$\rho_{TLD} = \rho_m + (1 - \rho_m) * \rho_c^2 \tag{1}$$

where the first part corresponds to having transient fault(s) in the memory sub-system while the second part corresponds to the case where both threads are subject to transient fault(s) on the chip (while there are no faults affecting memory subsystem).

Instead of using threads to duplicate the application, we can exploit the possibility of using duplicated single-threaded processes, which will have *separate* data/code segments, to enhance system reliability. As shown in Figure 1b, two memory blocks (and the cache blocks[1]) are used to hold the different address spaces of the two single-threaded processes. Considering the same data block size and the same duration of computation, the probability of having transient fault(s) in the data block(s) for one process is assumed to be $\rho_m$. The same holds for $\rho_c$. Therefore, the probability of failure for the PLD approach is:

$$\begin{aligned} R_{proc} &= (1 - \rho_m) * (1 - \rho_c) & (2) \\ \rho_{PLD} &= (1 - R_{proc})^2 & (3) \end{aligned}$$

where $R_{proc}$ is the probability of finishing one process correctly considering separate memory blocks are used. The application fails only if both processes fail during their executions. Simple algebra shows that $\rho_{PLD} < \rho_{TLD}$, that is, PLD achieves higher reliability than TLD. However, for the case shown in Figure 1b, where two threads are scheduled on separate cores, the energy consumption for PLD could be almost double of that for TLD.

## III. Effects of DVS

It is worth noting that, by scheduling two processes on separate cores, PLD could finish the application earlier than TLD, where two threads may compete for the same functional units. Considering the convex relation between the power consumption and operating frequency (and supply voltage), we can scale down the execution of the processes to save energy in the PLD scheme. However, the transient fault rate is exponentially-related to the *critical charge* (the smallest charge needed to cause a soft error) $(Q_{crit})$, which is proportional to the system supply voltage [4]. Therefore, as we scale down the processing on each core, the probability of incurring transient faults on each core will increase drastically due to increased fault rates [11].

Thus, more duplications may be used to compensate for the reliability loss due to the power management on CMPs and to obtain a certain level of system reliability. Moreover, such duplications may be executed on different cores with different speeds. Hence, the efficient/optimal use of the CMP architecture to improve the system reliability and energy savings is an open problem to a considerable extent.

## IV. Conclusion

Simultaneous multithreading (SMT) and chip multiprocessors (CMP) techniques were originally proposed to enhance the performance for modern processors, but they have been also explored for power-aware computing, recently. Moreover, due to their inherent redundancy, SMT/CMP techniques have been also used to enhance system reliability by duplicating the threads of execution to tolerate transient faults, which become more prominent with reduced technology sizes and design margins. All previous studies have focused on *thread-level duplication (TLD)*, which can detect/tolerate transient faults that affect the semi-conductor circuits. However, to the best of our knowledge, no work has addressed the effects of *shared* data/code segments among threads on system reliability, where a single corrupt memory cell could cause the failure of all threads.

Based on the idea of *process-level duplication (PLD)*, we illustrate the trade-off between system reliability and energy consumption for TLD and PLD in CMP systems. Although TLD could save more energy by using less resource (e.g., memory blocks), sharing data/code segments among threads could hurt system reliability. In contrast, PLD uses separate data/code segments for duplicated processes, which is more reliable but incurs more energy consumption.

In our future work, we will further explore the variations of TLD/PLD schemes by mapping threads/processes onto the same/different core(s). Moreover, considering the complications due to the energy management and transient faults, we will study the optimal management schemes (e.g., the number of duplications and processing speeds for the duplicated threads/processes) to maximize system reliability or to minimize energy consumption for a given application running on a CMP system.

[1]Although it is possible for two memory blocks being mapped to the same cache block, the transient fault(s) for one process will not affect the other. For simplicity, we consider the case of two memory blocks being mapped to two different cache blocks.

## References

[1] G. Chen, M. Kandemir, and F. Li. Energy-aware computation duplication for improving reliability in embedded chip multiprocessors. In *Proc. of the* $11^{th}$ *Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2006.

[2] V. Degalahal, L. Li, V. Narayanan, M. Kandemir, and M. J. Irwin. Soft errors issues in low-power caches. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 13(10):1157–1166, Oct. 2005.

[3] M. Gomaa, C. Scarbrough, T. Vijaykumar, and I. Pomeranz. Transient-fault recovery for chip multiprocessors. In *Proc. of the Int'l Symposium on Computer Architecture (ISCA)*, June. 2003.

[4] P. Hazucha and C. Svensson. Impact of cmos technology scaling on the atmospheric neutron soft error rate. *IEEE Trans. on Nuclear Science*, 47(6):2586–2594, 2000.

[5] K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson, and K. Chang. The case for a single-chip multiprocessor. In *Proc. of the Int'l Symp. Architectural Support for Programming Languages and Oper ating Systems (ASPLOS*, Oct. 1996.

[6] M. D. Powell, M. Gomaa, and T. N. Vijaykumar. Heat-and-run: Leveraging smt and cmp to manage power density through the operating system. In *Proc. of the Int'l Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Oct. 2004.

[7] M. W. Rashid, E. Tan, M. Huang, and D. Albonesi. Exploiting coarse-grain verification parallelism for power-efficient fault tolera nce. In *Proc. of the* $14^{th}$ *Int'l Conference on Parallel Architectures and Compilation Techniques (PACT)*, Sep. 2005.

[8] E. Rotenberg. Ar-smt: A microarchitectural approach to fault tolerance in microprocessors. In *Proc. of the Int'l Symposium on Fault-Tolerant Computing (FTCS)*, June. 1999.

[9] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proc. of the International Conference on Dependable Systems and Networks*, 2002.

[10] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *Proc. of the Int'l Symposium on Computer Architecture (ISCA)*, Jun. 1995.

[11] D. Zhu, R. Melhem, and D. Mossé. The effects of energy management on reliability in real-time embedded systems. In *Proc. of the International Conference on Computer Aidded Design (ICCAD)*, Nov. 2004.