Energy-Efficient Redundant Execution for Chip Multiprocessors

Pramod Subramanyan, Virendra Singh
Supercomputer Education and Research Center
Indian Institute of Science, Bangalore, India
pramod@rishi.serc.iisc.ernet.in
viren@serc.iisc.ernet.in

Kewal K. Saluja
Electrical and Computer Engineering Dept.
University of Wisconsin, Madison, WI
saluja@engr.wisc.edu

Erik Larsson
Dept. of Computer and Information Science
Linköping University, Linköping, Sweden
erila@ida.liu.se

ABSTRACT

Relentless CMOS scaling coupled with lower design tolerances is making ICs increasingly susceptible to wear-out related permanent faults and transient faults, necessitating on-chip fault tolerance in future chip microprocessors (CMPs). In this paper, we describe a power-efficient architecture for redundant execution on chip multiprocessors (CMPs) which when coupled with our per-core dynamic voltage and frequency scaling (DVFS) algorithm significantly reduces the energy overhead of redundant execution without sacrificing performance. Our evaluation shows that this architecture has a performance overhead of only 0.3% and consumes only 1.48 times the energy of a non-fault-tolerant baseline.

Categories and Subject Descriptors: C.1.0 [Processor Architectures]: General
General Terms: Reliability, Performance.
Keywords: Transient faults, permanent faults, redundant execution, microarchitecture.

1. INTRODUCTION

Over the last three decades, continued scaling of silicon fabrication technology has permitted exponential increases in the transistor budgets of microprocessors. In the past, higher transistor counts were used to increase the performance of single processor cores. However increasing complexity and power dissipation of these cores forced architects to turn to chip multiprocessors (CMPs) in order to deliver increased performance at a manageable level of power and complexity. While deep sub-micron technology is enabling the placement of billions of transistors on a single chip, it also poses unique challenges. ICs are now increasingly susceptible to soft errors [18], wear-out related permanent faults and process variations [3, 5].

Traditionally, high availability systems have been restricted to the domain of mainframe computers or specially designed fault-tolerant systems [4, 11]. However, the trend towards unreliable components means that fault tolerance is now important for the commodity market as well [1]. Fault-tolerant solutions for the commodity market have different requirements and present a different set of design challenges. The commodity market requires *configurable* [1] and *low cost* fault tolerance. CMPs are appealing in this context as they inherently provide replicated hardware resources which can be exploited for error detection and recovery. A number of proposals [1, 7, 12, 14, 19–23] have attempted to take advantage of these aspects of CMPs to provide fault tolerance.

An important aspect of low-cost fault tolerance is the energy-efficiency of fault-tolerant CMP designs. Power and peak temperature are key performance limiters for modern processors [8]. Since the power budget for a chip is fixed, decreasing the power consumed in any core increases the power available to other cores. This allows them to operate at a higher frequency, increasing overall system performance. Furthermore, reducing power dissipation has an additional advantage of reducing operating temperatures, which can increase chip lifetimes by an order of magnitude [13]. Consequently, we believe that there is a pressing need for energy-efficient fault-tolerant architectures for future microprocessors.

In this paper we propose an energy-efficient architecture for fault-tolerant CMPs. This architecture has a performance overhead of less than 0.3% and consumes only 1.48 times the energy of a non-fault-tolerant baseline processor. We compare our architecture to two previous proposals for fault-tolerant CMPs: (1) the parallelized verification architecture (PVA) introduced by Rashid et al. [14], and (2) Chip-level Redundantly Threaded (CRT) processors introduced by Mukherjee et al. [12]. Our proposal outperforms both PVA and CRT, and consumes lesser energy than either.

2. OVERVIEW AND DESIGN

Our architecture utilizes two cores of a CMP to execute a single logical thread. To an application executing in redundant mode, the two cores appear as a single logical processor. One of these cores is designated as the leading core, while the other is designated as the trailing core. The leading core is so named because it is temporally ahead of the trailing core. The two cores execute the same instruction stream, and process the same input data stream. Communication between the leading and trailing cores is carried out over the system bus of a shared memory CMP. A high level block diagram of the system architecture is shown in Figure 1.

The leading core *assists the execution* of the trailing core by forwarding branch outcomes and load values [12, 15]. This execution assistance increases the IPC of the trailing core. We exploit this increase by operating the trailing core at a lower voltage and frequency level, saving energy.

GLSVLSI'10, May 16–18, 2010, Providence, Rhode Island, USA.

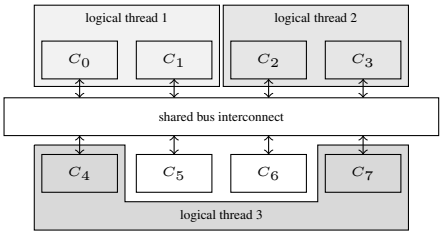

Figure 1 – System-level block diagram.

2.1 Core Architecture

To enable energy-efficient redundant execution, we augment a conventional out-of-order superscalar processor core with some additional structures. Figure 2 shows a block diagram of the modified core. The rest of the section describes these modifications in more detail.

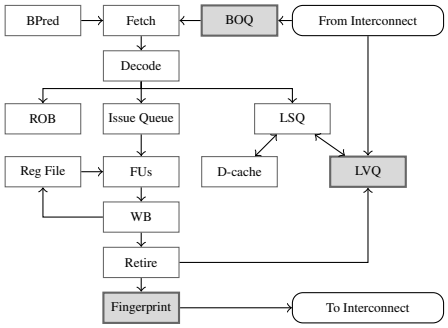

Figure 2 – Diagram showing processor core augmented with structures required for redundant execution. Newly added structures are shaded.

The *Load Value Queue* (LVQ) [15] structure is used to hold load values in the trailing core which are forwarded from the leading core. Forwarding load values has two benefits: (1) solving the problem of input replication [15] and avoiding input incoherence [20] and (2) speeding up the trailing core.

Previous implementations of the LVQ accessed it at the same time as the access to the data cache, i.e., after the effective address computation. This is an unnecessary delay because the LVQ entry from which the value has to be read is known at the time of instruction decode. Therefore, we introduce the *early-write* optimization. This optimization reads the LVQ and writes the result into the destination register at the time of instruction dispatch. The effective address computation takes places later and is used only for accessing the TLB and in the fingerprint computation. As a result of this optimization instructions which are dependent on load instructions can begin execution immediately after the load instruction is dispatched. This optimization *breaks data-dependence chains* containing load instructions, improving trailing core IPC by over 30%.

Branch outcomes from the leading core are forwarded to the trailing core's *Branch Outcome Queue* (BOQ)[15]. During instruction fetch the trailing core does not use the branch predictor. Instead it accesses the BOQ to get the direction and target address of the branch. In the absence of soft errors, the BOQ provides perfect branch prediction to the trailing core.

Periodically, the leading core and trailing core compute a hash value that summarizes updates that have been made to the state of the processor. This hash value is referred to as a *fingerprint* [19]. The two cores swap and compare fingerprints to detect errors. If no error has occurred, the architectural updates will be exactly the same, guaranteeing that the fingerprints will also be equal. If an error occurs, the fingerprints are extremely likely to be different. A mismatch in fingerprints indicates the occurrence of an error.

When the fingerprints are compared and found to match, the register state is stored in a *checkpoint store*. Memory state is also saved by using a modified L1 data cache similar to speculative version caches. The details of checkpointing, recovery and fault isolation are given in [22]. Fault coverage is discussed in [21] and [22].

2.2 Voltage and Frequency Control

Forwarding load values and branch outcomes to the trailing core allows it to execute faster than leading core. This speedup can be exploited by operating it at a reduced voltage and frequency level. The challenge here is to design an algorithm that can dynamically set the voltage/frequency levels of the trailing core based on program phase behaviour.

In this context, we make the key observation that the sizes of the BOQ and LVQ are an indication of the difference in execution speed between the two cores. To understand why, let us assume for a moment that the trailing core has infinite sized BOQ and LVQ structures. If the trailing core is operating at lower than its optimal frequency, its execution will be temporally behind the leading core, and the number of elements in the LVQ and BOQ will continuously increase. On the other hand, if the trailing core is operating at higher than its optimal frequency, the LVQ/BOQ structures will likely be empty. This suggests that an algorithm which varies the frequency of the trailing core based on the number of entries in the queues will be able to track IPC variations in leading core.

2.2.1 DVFS Algorithm

Our algorithm periodically samples the size of the BOQ and LVQ after a fixed time interval T_s. There are two thresholds associated with the BOQ and LVQ, a high threshold and a low threshold. If the occupancy of both structures structure is greater than the high threshold, then the frequency of operation is increased. If the occupancy of both the structures is less than the low threshold, then the frequency of operation is decreased. In effect the algorithm attempts to maintain the occupancy of the structures in between the low and high thresholds.

The thresholds can be set either statically or dynamically. Our results in section 3 show that a single static threshold for all programs provides significant power savings with a small performance degradation. Hence, we only use a statically set threshold value.

3. EVALUATION

3.1 Simulation Methodology

Our evaluation uses an appropriately modified version of the SESC execution-driven simulator [16]. The simulator models an out-of-order superscalar processor in a detailed manner and fully simulates "wrong-path" instructions. Our CMP configuration models a CMP with private L2 caches. Details of the model are given in Table 1.

In order to put our results in context, we compare our architecture against two previous proposals: (1) the Parallelized Verification Architecture (PVA) from [14] and (2) Chip-level Redundantly Threaded (CRT) processors from [12]. PVA-specific configuration values are taken from [14].

Workload: We used twelve benchmarks from the SPEC CPU 2000 benchmark suite. For each of the benchmarks, we skipped the first

Table 1 – CMP configuration

Fetch/issue/retire	4/4/4	Mem/Int/FP units	4/6/4	Branch predictor	hybrid/16k/16k/16k
ROB size	128 instructions	I-cache	32k/64B/4-way/2 cycles	BTB	4k entries/4-way
Integer/FP window	64/32 instructions	D-cache	64k/64B/4-way/2 cycles	RAS	32 entries
Load/store queue	32 instructions	Private L2 cache	2 MB/64B/8-way/24 cycles	LVQ size	128
Interconnect latency	48 cycles	Memory	400 cycles	BOQ size	128
Checkpointing interval	32k instructions	DVFS update interval	1 μs	DVFS update latency	100 ns
DVFS voltage levels	0.5 - 1.0 V	DVFS frequency levels	1.5-3.0 GHz	# of DVFS levels	6

three billion instructions and simulated the next one billion instructions.

3.2 IPC Results

Figure 3 shows the IPC of each of the SPEC benchmarks normalized by the IPC of non-fault-tolerant execution. The performance degradation of PVA is 4.2%. CRT's performance degradation is 4.4%. Our proposal, Energy-Efficient Redundant Execution (EERE), has a performance degradation of only 0.3%.

PVA uses a structure called the Post Commit Buffer (PCB) to hold buffered stores until they are verified. When the PCB becomes full, the primary core cannot make progress. Buffered stores are removed from the PCB only when the corresponding chunk of instructions is executed by the trailing cores and the checkpoints match. We simulate a CMP configuration with large interconnect delays which increases PCB occupancy, thus degrading PVA's performance. In CRT, a store cannot retire from the leading core's store buffer until it is verified by the trailing core. This creates additional pressure on the store buffer. This is the reason why CRT's performance suffers for some benchmarks.

3.3 Energy Results

Figure 4 shows the energy consumption of the SPEC benchmarks normalized by the energy consumption of a non-fault-tolerant execution. On an average CRT consumes 1.99 the energy of non-fault-tolerant execution. PVA consumes 1.79 times the energy of non-fault-tolerant execution. Our proposal, Energy-Efficient Redundant Execution (EERE), consumes 1.48 times the energy of non-fault-tolerant execution.

3.4 Sensitivity to Interconnect Latency

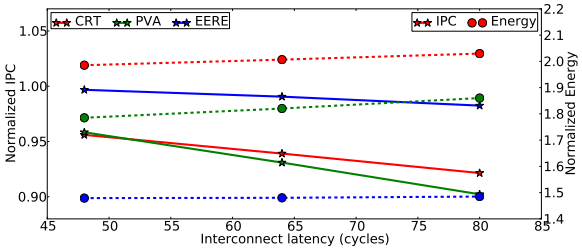

Figure 5 – Performance with increasing interconnect delay.

Figure 5 shows the variation of normalized IPC and normalized energy with increasing interconnect delay. Increasing interconnect delays have a detrimental impact on performance. Two points are apparent from the from the figure. Firstly, even with a very high interconnect latency of 80 cycles, our proposal is able to deliver the same level of energy-efficiency with only a 2% loss in performance. In contrast, PVA and CRT suffer a higher performance degradation. Secondly, the impact of higher interconnect latencies on energy consumption is minimal.

4. RELATED WORK

Fault Tolerant Architectures:

Current high availability systems like the HP Nonstop Advanced Architecture [4] and the IBM zSeries [6] are high cost systems that spare no expense to meet reliability targets. Although they provide excellent fault coverage, they impose a high cost of 100% hardware duplication and 100% additional energy consumption. For high availability systems targeted at the commodity market, these high costs are unacceptable.

DIVA is a novel fault detection architecture which detects faults in a larger out-of-order core by using an in-order checker core to verify computations at the time of retirement [2]. Unlike our proposal, a DIVA checker is always-on and its resources cannot be used when fault tolerance is disabled.

Transient fault detection using simultaneous multithreading was introduced by Rotenberg in AR-SMT [17] and Reinhardt and Mukherjee [15] in Simultaneously and Redundantly Threaded (SRT) processors. An SRT processor augments SMT processors with additional architectural structures like the branch outcome queue and load value queue for transient fault detection. The branch outcome queue enhances the performance of the redundant thread, while the load value queue provides input replication. Since an SRT processor provides an unpredictable combination of space and time redundancy, it cannot guarantee the detection of all permanent faults. Muhkerjee et al. also introduced chip level redundant threading (CRT) [12], which extends SRT to simultaneously multithreaded chip multiprocessors. Our design provides better performance and energy characteristics than CRT. Gomaa et al. studied Chip Level Redundant Threading with Recovery (CRTR) [7], which uses the state of the trailing thread to recover from an error. Our design also provides recovery from errors like CRTR, but unlike CRTR, it is faster and more energy-efficient than CRT.

Dynamic Voltage and Frequency Scaling (DVFS): The idea of per-core voltage and frequency levels was first explored Isci et al. [8]. They introduced a set of policies to manage per-core voltage and power levels in CMPs. Their policies aim to maximize performance while keeping power dissipation within the budget. These policies are managed either by a dedicated micro-controller, or a daemon running on a dedicated core. Kim et al. [10] described the detailed design of on-chip regulators; showing that it is possible to perform voltage changes in time periods of the order of a few hundred nanoseconds. Although current commercial processors do not yet have the ability to set per-core voltage levels, the AMD Quad Core Opteron [9] allows the frequency of each core to be set independently.

Energy-efficient Fault Tolerance: Rashid et al. [14] proposed the parallelized verification architecture (PVA) for fault-tolerant CMPs which saves energy by parallelizing the verification by executing it on two cores. These two "verification cores" are operated at half frequency and voltage levels. An important point here is that PVA uses three cores to execute a single logical thread, while our design uses only two. Our design also has better performance and energy-efficiency than PVA.

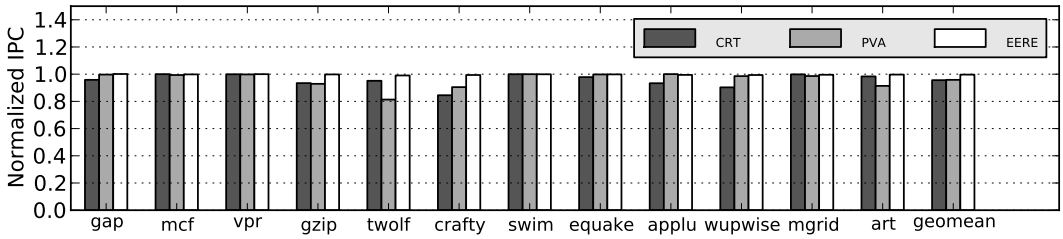

Figure 3 – Normalized IPC of the SPEC benchmarks. PVA and CRT refer to the proposals in [14] and [12] respectively. EERE is our proposal

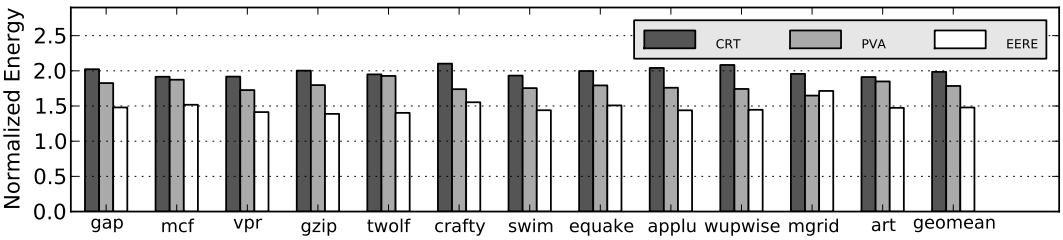

Figure 4 – Normalized energy consumption of the SPEC benchmarks. PVA and CRT refer to [14] and [12] respectively. EERE is our proposal.

5. CONCLUSION

Decreasing feature sizes, lower design tolerances and higher operating temperatures have resulted in the emergence of wear-out related permanent faults and transient faults as significant concerns in modern microprocessors.

In this paper, we showed the design of an energy-efficient fault-tolerant microarchitecture for chip multiprocessors. Our microarchitecture exploits the slack due to branch mispredictions and data cache misses to operate the trailing core at a lower frequency, significantly reducing the energy-cost of fault-tolerance. Our results showed that this architecture has a performance overhead of less than 0.3% and energy consumption that is only 1.48 times that of non-fault-tolerant execution. We compared our architecture with two previous proposals for fault-tolerant CMPs and found that our architecture outperforms both proposals while consuming less energy. Our results also showed that our architecture performs well even with very high interconnect delays, making it suitable for future CMPs with several tens or hundreds of cores.

References

[1] Nidhi Aggarwal, Parthasarathy Ranganathan, Norman P. Jouppi, and James E. Smith. Configurable isolation: building high availability systems with commodity multi-core processors. *SIGARCH Comput. Archit. News*, 35(2), 2007.

[2] Todd Austin. DIVA: A Reliable Substrate For Deep Submicron Microarchitecture Design. *Proceedings of the 32nd MICRO*, 1999.

[3] Todd Austin, V. Bertacco, S. Mahlke, and Yu Cao. Reliable Systems on Unreliable Fabrics. *IEEE Des. Test*, 25(4), 2008.

[4] D. Bernick, B. Bruckert, P. D. Vigna, D. Garcia, R. Jardine, J. Klecka, and J. Smullen. Nonstop®advanced architecture. In *DSN '05: Proc. of DSN*, 2005.

[5] S. Y. Borkar. Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. *IEEE Micro*, 25(6), 2005.

[6] M.L. Fair, C.R. Conklin, S. B. Swaney, P. J. Meaney, W. J. Clarke, L. C. Alves, I. N. Modi, F. Freier, W. Fischer, and N. E. Weber. Reliability, Availability, and Serviceability (RAS) of the IBM eServer z990. *IBM Journal of Research and Development*, 2004.

[7] M. Gomma, C. Scarbrough, T. N. Vijaykumar, and I. Pomeranz. Transient-Fault Recovery for Chip Multiprocessors. *Proceedings of the 30th ISCA*, 2003.

[8] C. Isci, A. Buyuktosunoglu, C-Y. Cher, P. Bose, and M. Martonosi. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. *Proc. of the 39th MICRO*, 2006.

[9] J. Dorsey et al. An Integrated Quad-core Opteron processor. *International Solid State Circuits Conference*, 2007.

[10] W. Kim, M. S. Gupta, Wei Gu-Yeon, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. *Proceedings of the 14th HPCA*, 2008.

[11] Israel Koren and C. Mani Krishna. *Fault Tolerant Systems*. Morgan Kaufmann Publishers Inc., 2007.

[12] S. S. Mukherjee, M. Kontz, and S. K. Reinhardt. Detailed Design and Evaluation of Redundant Multithreading Alternatives. *Proceedings of the 29th ISCA*, 2002.

[13] I. Parulkar, A. Wood, J. C. Hoe, B. Falsafi, S. V. Adve, and J. Torrellas. OpenSPARC: An Open Platform for Hardware Reliability Experimentation. *Fourth Workshop on Silicon Errors in Logic-System Effects (SELSE)*, 2008.

[14] M. W. Rashid, E. J. Tan, M. C. Huang, and D. H. Albonesi. Exploiting Coarse-Grain Verification Parallelism for Power-Efficient Fault Tolerance. *Proc. of the 14th International Conference on Parallel Architectures and Compilation Techniques*, 2005.

[15] S. K. Reinhardt and S. S. Mukherjee. Transient Fault Detection via Simultaneous Multithreading. *Proceedings of the 27th ISCA*, 2002.

[16] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos. SESC Simulator. `http://sesc.sourceforge.net/`, 2005.

[17] E. Rotenberg. AR-SMT: A Microarchitectural Approach to Fault Tolerance in a Microprocessor. *Proceedings of FTCS*, 1999.

[18] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi. Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic. *Proceedings of the 32nd DSN*, 2002.

[19] J. C. Smolens, B. T. Gold, J. Kim, B. Falsafi, J. C. Hoe, and A. G. Nowatzyk. Fingerprinting: Bounding soft error detection latency and bandwidth. *Proceedings of the 9th ASPLOS*, 2004.

[20] J. C. Smolens, B. T. Gold, B. Falsafi, and J. C. Hoe. Reunion: Complexity-Effective Multicore Redundancy. *Proceedings of the 39th MICRO*, 2006.

[21] P. Subramanyan, V. Singh, K. K. Saluja, and E. Larsson. Power-Efficient Redundant Execution for Chip Multiprocessors. *Proc. of 3rd WDSN*, 2009.

[22] P. Subramanyan, V. Singh, K. K. Saluja, and E. Larsson. Mulitplexed Redundant Execution: A Technique for Efficient Fault Tolerance in Chip Multiprocessors. *Proc. of DATE*, 2010.

[23] P. Subramanyan, V. Singh, K. K. Saluja, and E. Larsson. Energy-Efficient Fault Tolerance in Chip Multiprocessors Using Critical Value Forwarding. *To appear in Proc. of DSN*, 2010.