

Fingerprinting Across On-Chip Memory Interconnects

Srinivas Chellappa, Frédéric de Mesmay, Jared C. Smolens, Babak Falsafi, James C. Hoe, and Ken Mai
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract—Pairs of cores in a chip multiprocessor (CMP) can execute programs redundantly to detect and recover from soft errors. Prior work assumes dedicated cross-core buses to compare the redundant cores’ outputs for error detection. In this paper, we investigate using the CMP’s existing on-chip memory interconnect for comparing hashes of architectural state updates, called fingerprints, across redundant cores. We show that the memory system can support periodic fingerprint comparison. Furthermore, our simulation-based results show that for reasonable comparison intervals, the added load does not affect memory access latency and matches the performance of a dedicated comparison bus.

1. INTRODUCTION

The trend of increasing device density, coupled with reduced capacitance and noise margins mean that soft error rates in processor datapaths will rise [1], [2]. One method to protect the datapath is to execute programs redundantly across cores in a chip multiprocessor (CMP) [3]–[5]. In order to detect errors in execution, the cores must continually compare their architectural state updates with each other. Because of the substantial amount of state retired in each clock cycle by aggressive superscalar processors, equal to the L1 cache store port bandwidth or even the architectural register file, researchers have investigated ways to efficiently compare the two executions.

One method which has been proposed is architectural fingerprints [6]. Architectural fingerprints compress architectural state updates into a small hash for periodic comparison across redundant processor cores. The fingerprint is accumulated over a contiguous interval of instructions. Before retirement, the cores exchange their respective fingerprint values to detect single event upsets that have affected instructions in the interval. Because a single fingerprint (e.g., 16 bits) can cover multiple instructions, fingerprinting offers significant bandwidth savings over directly comparing execution results.

Current techniques to compare fingerprints rely on dedicated datapaths between processor cores, as illustrated in Figure 1. However, there are disadvantages to using dedicated datapaths: they require global hardware additions and restrict fingerprinting only between core pairs determined at design time. The alternative, datapaths between all pairs, is unscalable as the number of cores increases.

The observation in this paper is that CMPs already provide a datapath suitable for transporting fingerprints: the on-

chip core-cache memory interconnect. Reusing the on-chip memory interconnect eliminates dedicated hardware datapaths, while allowing dynamic reconfiguration of redundant execution across cores. In this paper, we show that transporting fingerprints across the on-chip memory interconnect is viable and offers performance comparable to dedicated datapaths.

2. APPROACH

The redundant execution design considered in this paper is Reunion [5], briefly summarized below. The Reunion execution model divides logical processor pairs in a CMP into *vocal* and *mute* cores. The vocal core executes a program and releases updates (e.g., stores) to shared caches and memory, as in a non-redundant CMP. The mute core checks the vocal’s execution by periodically exchanging architectural fingerprints with the vocal. While the mute uses a private cache hierarchy for its own loads and stores, the mute’s outputs are never observable by other cores in the system. The cores support rollback-recovery by leveraging the existing precise exception support. This design preserves the traditional non-redundant design, including the out-of-order execution core, core-cache interface, and cache coherence protocol. The main changes from a traditional non-redundant CMP are limited to the shared cache controller and adding fingerprint generation and comparison support to the cores.

However, the system evaluated in Reunion [5] assumes dedicated datapaths between cores for fingerprint comparison. This incurs additional global buses, dedicated solely to support redundant execution. We observe that, because the fingerprints are small (comparable to a cache request message), they can instead accompany cache memory transactions on the memory interconnect, leveraging existing datapaths for communication between individual processor cores and the shared cache.

Related work. Other researchers have also investigated soft error detection and correction with redundant execution in a CMP. Mukherjee et al. [3] propose directly comparison of store values between redundant cores to provide detection. This requires a wide, dedicated cross-core datapath that matches the L1 cache store bandwidth. Gomaa et al. [4] propose a similar design that supports recovery. However, recovery requires timely error detection for all updated architectural state. To avoid comparing every instruction result, the authors propose comparing the ends of instruction dependence chains, which

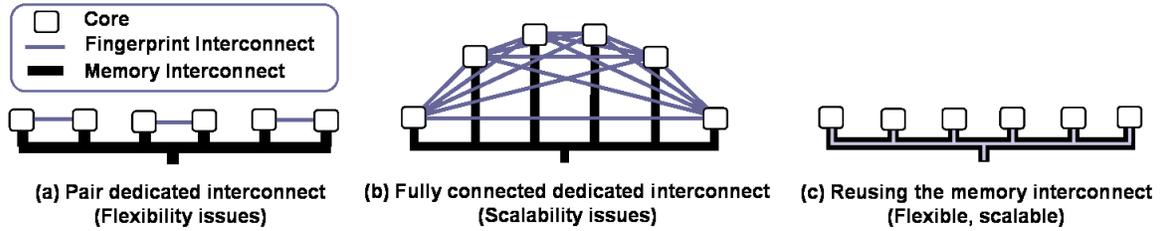


Fig. 1. Advantage of using the on-chip memory interconnect over fixed datapaths. (a) fixes the pairs that can perform DMR at design time. (b) overcomes this, but requires hardware datapaths between all cores, which does not scale. (c) solves these problems by sending fingerprints across the existing memory interconnect.

avoids approximately 20% of the comparisons. The results from this paper will apply if fingerprints are used for error detection in these redundant execution designs.

3. DESIGN

This section discusses the baseline CMP microarchitecture and implementation of fingerprinting in the on-chip memory interconnect and the key design tradeoffs.

3.1 Baseline CMP

We consider a baseline CMP with multiple aggressive superscalar, out-of-order processor cores. Each core has its own private L1 instruction and data caches which are connected to a shared L2 cache through a non-blocking crossbar. As in the Sun OpenSPARC T1 [7], the on-chip memory interconnect consists of crossbar paths between cores and independent cache controller banks. Each crossbar port can both send and receive one message on each cycle; cores do not communicate directly.

3.2 Fingerprint Comparison

During redundant execution, each core generates a fingerprint covering an interval of instructions and sends that fingerprint to its redundant partner core. The instructions being checked are buffered in the core’s out-of-order queues (e.g., re-order buffer and store buffer) until a corresponding, matching fingerprint has been received from the partner core. If the fingerprints match, the instructions are irrevocably retired. If the fingerprints do not match, the instructions are discarded and the cores re-execute the instruction interval.

We consider two designs for transmitting fingerprints between the redundant cores. First, our baseline redundant design provides a dedicated datapath that incurs a fixed latency to send each message and matches the retirement bandwidth of the core. Second, we consider a design where fingerprints are transmitted as messages through the existing L1 cache store request datapath. The fingerprint transmission consumes the processor’s cache store port and traverses the crossbar to reach its destination. Two crossbar traversals are necessary to reach the destination core because direct core-core communication paths are unavailable. Fingerprint messages compete with existing on-chip cache and coherence requests.

3.3 Design Factors

The fingerprint interval is a key design parameter which affects (1) the bandwidth required for comparison and (2) the amount of instruction buffering needed before comparison.

Fingerprints are transmitted at fixed instruction intervals. To the first order, the bandwidth (measured in messages per cycle) required for fingerprint comparisons is proportional to the IPC (instructions per cycle) of the executing program and inversely proportional to the fingerprint interval. The interaction of fingerprint messages with existing cache transactions creates contention at the processor’s cache ports and the shared cache. Contention from this additional bandwidth demand can become a dominant performance bottleneck.

Because a fingerprint cannot be generated until all instructions in the interval have completed execution, longer intervals also incur additional pipeline resource occupancy (e.g., re-order buffer and store buffer entries) while instructions are being collected and checked. However, speculative execution can continue, so the overall performance impact from this effect is small.

Finally, the Reunion evaluation shows that the comparison latency, the time required to transmit and compare a fingerprint, is a critical performance factor in multithreaded workloads [5]. While most comparisons can overlap with further execution, serializing instructions (such as traps, memory barriers, and non-idempotent I/O) cause execution to stall until the serializing instruction has been checked. This exposes execution stalls for the entire comparison latency, directly impacting performance. The observed delay is fixed for dedicated datapaths, while the latency observed in an on-chip interconnect varies based upon memory system contention. Our evaluation quantifies this latency.

4. RESULTS

In this section, we first present our simulation methodology, followed by our results and analysis. We discuss the viability of our approach, the factors affecting the performance of our design, and finally, the effect of our design on rest of the system.

4.1 Methodology

We evaluate the designs using the Flexus full-system CMP performance simulator. Parameters and workloads are

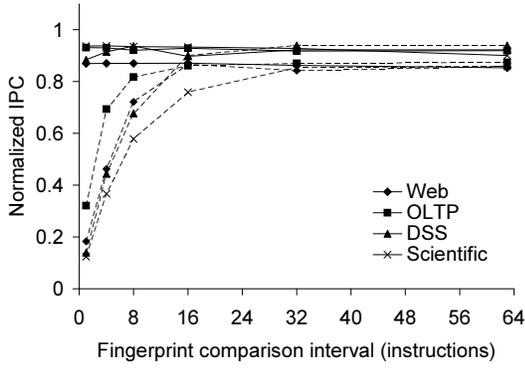


Fig. 2. Performance sensitivity to fingerprint comparison interval for dedicated datapaths (solid lines) with a fixed latency of 10 cycles, and on-chip memory interconnect (dashed lines). Normalized to a baseline lockstepped system.

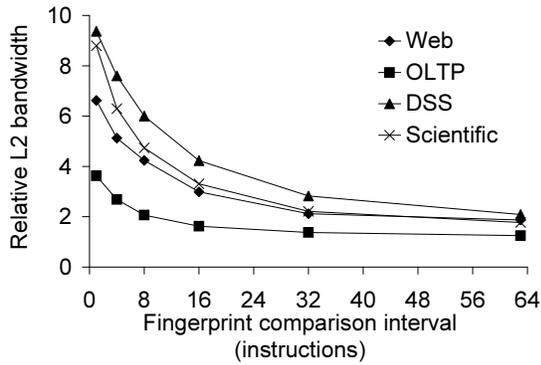


Fig. 3. Interconnect bandwidth utilization as the fingerprint comparison interval increases. Normalized to the baseline shared cache request traffic.

described in [5]; we summarize the parameters here. The baseline design is a eight-core CMP with idealized lockstepped processor pairs (four logical contexts) that have no performance overhead from redundant execution or error checking. The processors are four-wide out-of-order SPARC v9 cores, with a 256 entry re-order buffer, 64-entry store buffer and 64kB private L1 caches with two load ports and one store port. The shared cache is a 16MB with four banks, connected to cores through the crossbar described earlier. We also model a Reunion system with a fixed 10-cycle inter-core fingerprint comparison latency across a dedicated datapath and the experimental system with fingerprint messages transmitted over the on-chip memory interconnect.

Our workloads include Apache and Zeus web servers running SPECWeb99, Oracle 10g and IBM DB2 ESE 8 with a 100 warehouse online transaction processing (OLTP) application, DB2 with decision support system (DSS) queries 1 and 2, and scientific workloads *em3d*, *moldyn*, *ocean*, and *sparse* which show a range of memory access behaviors. Detailed workload parameters are in [5].

4.2 Evaluation

Factors affecting performance. To measure the system performance impact of our design, we compare the user-mode IPC

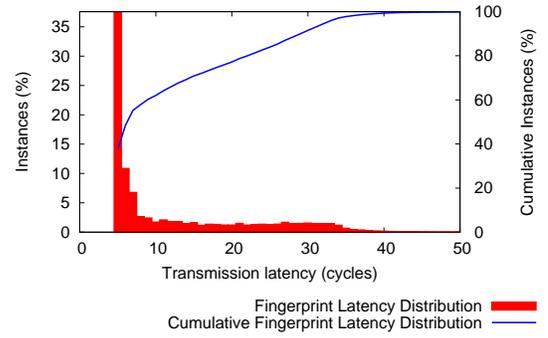


Fig. 4. Fingerprint latency distribution: Representative scientific workload (*ocean*, 32-cycle comparison interval).

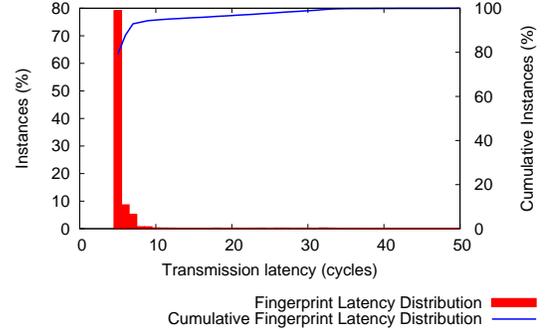


Fig. 5. Fingerprint latency distribution: Representative commercial workload (DSS query 1, 32-cycle comparison interval).

of the systems. Figure 2 shows the effect of increasing the fingerprint comparison interval on performance for a system with a 10-cycle dedicated datapath and a system using the on-chip memory interconnect. The comparison interval determines the additional pressure applied to the memory interconnect, which in turn impacts memory access latencies and system performance. These results indicate that for the larger comparison intervals (32 instructions or larger), the impact of such pressure is negligible. However, for the shorter comparison intervals (16 instructions or less), bandwidth pressure on the cache ports and crossbar severely impacts performance. This demonstrates the viability of on-chip memory interconnects for larger comparison intervals.

Figure 3 shows our design’s memory interconnect bandwidth demands, normalized to the baseline L2 memory bandwidth for each workload class, as the fingerprint comparison interval increases. The results clearly show much higher bandwidth demands at shorter comparison intervals—many times the bandwidth required to support the existing workload traffic—which causes contention for buffers and increased latencies for both cache transactions and fingerprint messages.

In addition to the comparison interval, the comparison latency and the memory access patterns of the workloads are factors that affect the performance of our design. Fingerprint transmission latencies affect performance because cores stall retirement until fingerprints are received. Because our design uses the memory interconnect to transmit fingerprints, the transmission latency varies and is dependent upon contention in the interconnect. Our experiments show that for the shorter comparison intervals, the transmission latency is affected heavily by the length of the interval, because in these cases,

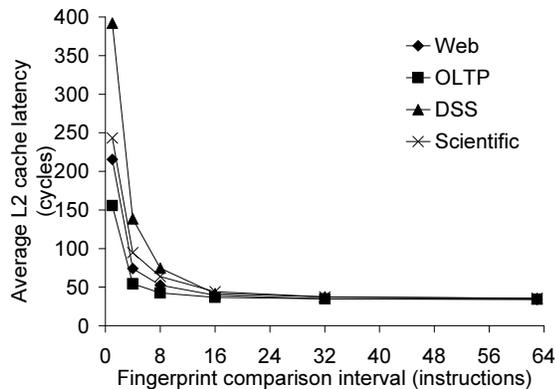


Fig. 6. Average shared L2 cache access latency with on-chip memory interconnects, as the comparison interval increases.

TABLE I

TRANSMISSION LATENCIES (CYCLES), 32 INSTRUCTION INTERVAL.

| | Range | Mean |
|-------------------|--------|------|
| Web | 5 - 89 | 6.6 |
| OLTP | 5 - 84 | 7.1 |
| DSS | 5 - 75 | 6.6 |
| Scientific | 5 - 66 | 11.5 |

fingerprint messages dominate the total interconnect traffic. For larger comparison intervals, the nature of the workload (in particular, the workload’s interconnect bandwidth usage) determines the fingerprint transmission latency, and thus also overall performance.

Figures 4 and 5 show the distribution of the fingerprint transmission latencies for representative scientific and commercial workloads, respectively. The difference in distribution between the scientific and commercial workloads is explained by the higher interconnect bandwidth utilization of the scientific workloads, and therefore higher L2 latency. Furthermore, the higher IPC and higher L2 request volume of the scientific workloads cause fingerprint messages to be queued behind cache requests, resulting in fingerprint messages that are delayed by L2 requests (a 35-cycle latency), as seen in Figure 4.

Table I summarizes the fingerprint transmission latency ranges and averages for the workloads when using the interconnect to transmit fingerprints. The comparison interval is 32 instructions for these experiments. For 32-instruction and larger comparison intervals, a constant delay of 10 cycles on the baseline system with dedicated datapaths is a good approximation of the performance of our design that uses the interconnect, as corroborated by several performance charts presented in this section.

System Impact. To confirm the reasons behind our performance results and to examine the effects of our design on the rest of the system, we measured the increase in the system’s mean L2 access latency, which directly impacts performance.

The consequence of inducing additional traffic in the memory interconnect is increased L2 access latencies. Figure 6 shows the impact of our design on the L2 latency across a range of comparison intervals. The correlation of the L2 access latencies (Figure 6) and bandwidth utilization (Figure 3), with the corresponding impact on throughput (Figure 2) indicates that the poor performance of the shorter comparison intervals is caused by contention for the memory interconnect. Interconnect contention becomes a negligible issue as the comparison intervals grow larger.

5. CONCLUSION

Fingerprinting is an efficient mechanism for error detection across redundant cores in a chip multiprocessor. Past fingerprinting approaches have assumed dedicated datapaths between cores to transport fingerprints. Our main contribution is to show that fingerprints can instead be transported using existing infrastructure, namely the on-chip memory interconnect. Advantages of our design include simplicity through reuse of existing hardware, flexibility, and scalability. We show that for reasonable fingerprint comparison intervals, our design does not affect memory access latency, and carries virtually no overhead.

REFERENCES

- [1] T. Juhnke and H. Klar, “Calculation of the soft error rate of submicron cmos logic circuits,” *IEEE Journal of Solid State Circuits*, vol. 30, no. 7, pp. 830–834, July 1995.
- [2] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, “Modeling the effect of technology trends on soft error rate of combinational logic,” in *International Conference on Dependable Systems and Networks*, June 2002.
- [3] S. Mukherjee, M. Kontz, and S. Reinhardt, “Detailed design and evaluation of redundant multi-threading alternatives,” in *Proceedings of 29th Annual International Symposium on Computer Architecture*, 2002.
- [4] M. Goma, C. Scarbrough, T. Vijaykumar, and I. Pomeranz, “Transient-fault recovery for chip multiprocessors,” in *Proceedings of 30th Annual International Symposium on Computer Architecture*, 2003.
- [5] J. C. Smolens, B. T. Gold, B. Falsafi, and J. C. Hoe, “Re-union: Complexity-effective multicore redundancy,” in *Proceedings of ACM/IEEE International Symposium on Microarchitecture (MICRO-39)*, Dec 2006.
- [6] J. C. Smolens, B. T. Gold, J. Kim, B. Falsafi, J. C. Hoe, and A. G. Nowatzky, “Fingerprinting: Bounding soft-error detection latency and bandwidth,” in *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct 2004, pp. 224–234.
- [7] *OpenSPARC T1 Microarchitecture Specification*, Sun Microsystems, Aug 2006.