StageNet: A Reconfigurable CMP Fabric for Resilient Systems

Shantanu Gupta   Shuguang Feng   Jason Blome   Scott Mahlke

Advanced Computer Architecture Laboratory
University of Michigan
Ann Arbor, MI 48109
{shangupt, shoe, jblome, mahlke}@umich.edu

ABSTRACT

Though CMOS feature size scaling has been the source of dramatic performance gains, this scaling has lead to mounting reliability concerns due to increasing power densities and on-chip temperatures. Given that most wearout mechanisms that plague semiconductor devices are highly dependent on these parameters, significantly higher failure rates are projected for future technology generations. Traditional techniques for dealing with device failures have relied on the coarse-grained replication of structures (typically at the processor core level) to maintain service in the face of failed components. In this work, we challenge the practice of core-level replication by identifying the inability of core-level replication to scale to high failure rate scenarios and investigating the advantages of finer-grained configurations. The case is made that supporting fine-grained reconfiguration not only enhances a system's robustness, but can improve overall throughput as well. We use this study to motivate the design of StageNet, a CMP architecture designed from its inception with reliability as a first class design constraint. StageNet relies on a reconfigurable network of replicated processor structures to maximize the useful lifetime of the chip, gracefully degrading performance toward end of life.

1. INTRODUCTION

Device scaling trends into the nanometer regime have lead to increasing current and power densities and rising on-chip temperatures, resulting in increasing device failure rates. Leading technology experts have begun to warn designers that device reliability will begin to deteriorate from the 65nm node onward [5]. Current projections indicate that future microprocessors will be composed of billions of transistors, many of which will be unusable at manufacture time, and many more which will degrade in performance (or even fail) over the expected lifetime of the processor [7]. To assuage these reliability concerns, computer designers must directly address reliability in computer systems through innovative fault-tolerance techniques.

The sources of computer system failures are widespread, ranging from transient faults, due to energetic particle strikes [26] and electrical noise [23], to permanent errors, caused by wearout phenomenon such as electromigration [9] and time dependent dielectric breakdown [25]. In recent years, industry designers and researchers have invested significant effort in building architectures resistant to transient faults and soft errors. Though there is significant evidence suggesting a growing rate of soft errors in future technology generations [7], this problem is actively being addressed in research [15, 15, 16, 24].

In contrast, much less attention has been paid to the problem of permanent faults, specifically transistor wearout due to the degrada-tion of semiconductor materials over time. Concerns about wearout are primarily due to increasing power and current densities, both of which lead to increasing on-chip temperatures. All of these three parameters have been shown to heavily influence most wearout mechanisms [3]. In fact, most wearout mechanisms exhibit an exponential dependence on temperature [12] [9] [20]. Furthermore, device scaling increases the susceptibility to wearout by shrinking the thickness of the gate and inter-layer dielectrics and increasing interconnect current density. Traditional techniques for dealing with transistor wearout have involved extra provisioning in logic circuits, known as guard-banding, to account for the expected performance degradation of transistors over time. However, the increasing degradation rate projected for future technology generations implies that traditional margining techniques will be insufficient. This necessitates revolutionary new designs for systems that can identify and adapt to wearout through reconfiguration.

The challenge of tolerating permanent faults can be broadly divided into three requisite tasks: fault detection, fault diagnosis, and system recovery. Fault detection mechanisms are used to determine that a fault is present in the system, while fault diagnosis is used to determine the source and nature of the fault. System recovery can consist of a number of different tasks, based on the nature of the fault. For example, if the fault is transient, the incorrect state may be corrected by simply flushing the processor pipeline [4]. However, if the fault is permanent, then a recovery mechanism which leverages system reconfiguration may be necessary to avoid propagating faults through the use of a failed component.

In general, system reconfiguration requires additional redundant resources, or the decommissioning of non-critical components. As an example, many computer vendors provide the ability to repair faulty memory and cache cells, through the inclusion of spare memory elements [18]. Recently, researchers have begun to extend these techniques to support sparing for additional on-chip resources [21], such as branch predictors [8] and registers [14].

Classical mechanisms such as dual and triple-modular redundancy (DMR and TMR) have been used in the past to address the problem of system recovery. However, replication at this granularity incurs a high hardware overhead and can only tolerate a small number of defects [10]. While such techniques may be appropriate for mainframes or mission-critical systems, they are generally too costly in terms of area and power requirements for mainstream desktop and embedded computer systems.

Trends in multicore systems have opened up a new design space for reliable system design. Recent work [17, 2] use the available redundancy in multicore systems to run duplicated copies of the same process or thread, thereby giving systems the ability to diagnose faulty cores and isolate them. Another interesting proposal, ElastIC [22], proposes dynamic reliability management for a mas-

sively multicore system that uses on chip wearout sensors to turn off cores that become defective over time. The focus of this paper is to present a generic framework for providing reconfiguration capabilities with a smaller granularity of replication in order to maximize the lifetime of CMP designs.

Although existing fault tolerance solutions with core level redundancy are viable, their effectiveness in the long run is not guaranteed. With the increasing defect rate in semiconductor technology, it will not be uncommon to see a rapid degradation in throughput for systems that have processor core level redundancy. This is because with any single device failure within a processor core, the entire core would have to be decommissioned, drastically reducing throughput of the system and leaving many working functional structures unused. This suggests the need for finer-grained control over system redundancy that enables reconfiguration for structures within a processor core. Over time as more and more devices fail, such a system will gracefully degrade its performance capabilities.

To this end, this work presents, and evaluates StageNet, a highly reconfigurable and adaptable CMP computing substrate. StageNet is a CMP architecture designed as a network of pipeline stages. This architecture naturally exploits the inherent redundancy in a CMP fabric to maintain higher system throughput over the duration of a system's life (even extending that lifetime) compared to a conventional multicore design. With a sea of pipeline stages at its disposal, an intelligent reliability management system can dynamically configure StageNet to meet changing reliability and performance demands. The primary contributions of this paper:

- A design space exploration of potential reconfiguration granularities for resilient system design

- A study of mean time to failure (MTTF) of different reconfiguration granularities

- A networked CMP architecture (StageNet) overview and evaluation

2. RECONFIGURATION GRANULARITY

An architecture for tolerating permanent faults requires the ability for system reconfiguration, where reconfiguration can refer to a variety of activities ranging from decommissioning non-functioning, non-critical processor structures to swapping in cold spare devices. In a reconfigurable architecture, recovery entails isolating defective component(s) and incorporating spare structures as needed. Support for reconfiguration can be achieved at various levels of granularity, from ultra-fine grained systems that have the ability to replace individual logic gates to coarser designs that focus on isolating entire processor cores. This choice presents a trade-off between complexity of implementation and potential lifetime enhancement, where finer grained solutions provide greater lifetime extensions than their coarser counterparts at significantly more cost. Generally speaking, the law of diminishing returns dictates the granularity of reconfiguration. This section presents experiments studying this trade-off and draws upon these results to motivate the design of StageNet.

2.1 Experimental Setup

In order to effectively model the reliability of different designs, a Verilog model of the OpenRISC 1200 (OR1200) core [1] was used as a representative design for the lifetime reliability experiments. The OR1200 is an open-source core with a conventional 5-stage pipeline design. The core was synthesized, placed and routed using industry standard CAD tools with a library characterized for a

OR1200 Core	
Area	1.0 mm^2
Power	123.9 mW
Clock Frequency	400 MHz
Data Cache Size	8 KB
Instruction Cache Size	8 KB
Technology Node	90 nm

(a) Overlay of the OR1200 floorplan on top of the placed and routed implementation of the CPU core.
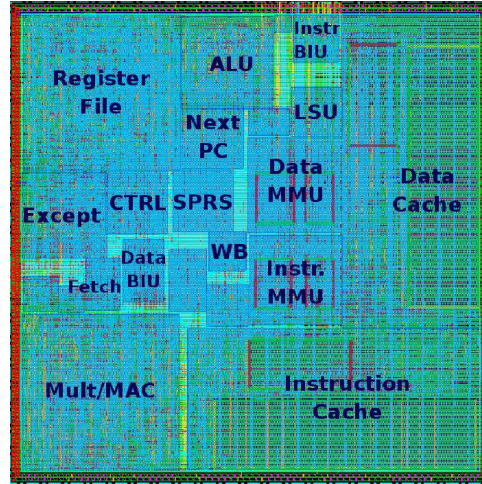
(b) Implementation details

Figure 1: OpenRisc 1200 embedded microprocessor

90nm process. The final floorplan along with several attributes of the design is shown in Figure 2.1.

Mean-time-to-failure (MTTF) was used as the metric to estimate lifetime of various modules in the OR1200 design. This study was conducted for the time-dependent-dielectric-breakdown (TDDB) wearout mechanism[1], employing an empirical model similar to that found in [19]. Equation 1 gives the per device MTTF in the design with respect to the TDDB wearout mechanism. Furthermore, module level MTTFs were calculated by identifying the minimum MTTF across all logic gates within each top-level module of the OR1200 core. More details about these calculations can be obtained from [6], which uses a similar experimental setup.

$$MTTF_{TDDB} \propto \left(\frac{1}{V}\right)^{(a-bT)} e^{\frac{(X+\frac{Y}{T}+ZT)}{kT}} \qquad (1)$$

where,

- V = operating voltage
- T = temperature
- k = Boltzmann's constant
- $a, b, X, Y,$ and Z are all fitting parameters based on [19]

The purpose of this setup was to generate a per module MTTF in the OR1200 design. In this experiment, it is assumed that the fastest

---

[1]A similar analysis can be done for other wearout mechanisms including negative bias threshold inversion (NBTI), hot carrier injection (HCI) and electromigration (EM)

failing component in the design (the one with the smallest MTTF) determines the operational lifetime of the core. Using this MTTF data, the next subsection will discuss advantages and disadvantages of reconfiguring the hardware at different levels of granularity.

2.2 Choosing the Granularity

The granularity of reconfiguration is used to describe the unit of isolation or replication for components within the processor core. Implicitly it also states the level at which redundancy is maintained by the system, because the replacement of the faulty component is done by the redundant spares. It is important to note that it is not strictly necessary to use cold spare structures in place of failed components, in certain situations the isolation of non-critical faulty component suffices. Various options for reconfiguration in the order of increasing granularity are as follows:

Gate level: Given this level of reconfiguration, a system can replace logic gates in the design as and when they fail. Unfortunately, such designs are typically impractical because they both require capability to diagnose faults precisely at the level of individual gates, and require tremendous overhead due to redundant components and wire routing area.

Module level: At this level of reconfiguration, a processor core can replace broken micro-architectural structures such as an ALU or branch predictor.

Stage level: The microarchitectural modules of a processor core are grouped together to form pipeline stages that make a coarser grained reconfiguration level. Stage level reconfiguration suggests replacement at this granularity, for example, a fault in the operand muxes feeding the ALU will require the replacement of the entire Execute stage.

Core level: This forms the coarsest level of reconfiguration where an entire processor core can be isolated from the system upon its failure. From the perspective of a system designer, this is the easiest technique to implement but at the same time has poorest returns in terms of the lifetime extension.

Figure 2 demonstrates the effectiveness of the above granularities of reconfiguration (gate-level reconfiguration is not included in this study due to the complexity of implementation). MTTF values for the modules were computed as described in the previous subsection. The module level MTTF values were then used to get stage level MTTF values by taking the minimum of module MTTF values belonging to a stage. The same reasoning was extended to compute the core level MTTF by using the minimum MTTF value among all the modules. The figure overlays three separate plots, one for each level of reconfiguration. The redundant spares matched the granularity of reconfiguration and were provisioned to add as much as 300% area overhead. The area overhead presented here is from the redundant components.

The data shown in Figure 2 demonstrates that going towards finer-grained reconfiguration is categorically beneficial as far as the gain in MTTF is concerned. But, it overlooks the design complexity aspect of the problem. As one goes towards finer-grained reconfiguration, hardware challenges for supporting redundancy aggravate, e.g. muxing logic, wiring overhead, circuit timing management, etc. At the same time, very coarse grained reconfiguration is also not an ideal candidate since the MTTF gain scales poorly with the area overhead. Therefore, a middle-ground solution is desirable that is amiable to the reconfiguration and has a better life expectancy.
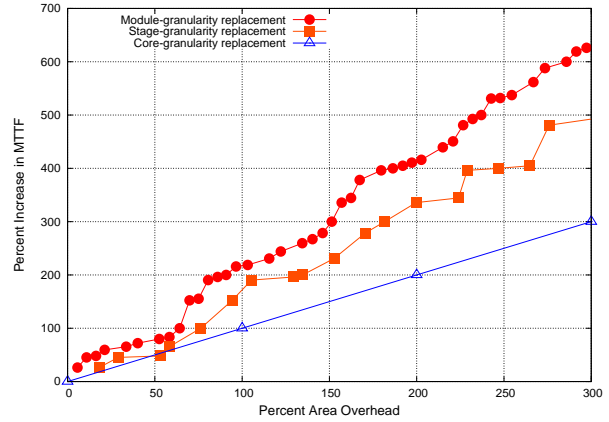

Figure 2: Gain in MTTF from the addition of cold spares at the granularity of micro-architectural modules, pipeline stages, and processor core. The gains shown are cumulative, and spare modules are added in the order they are expected to fail (the markers indicate the times when a cold spare is added to the system). The base system is a single core machine. A higher slope indicates better returns on the area investment but at the same time involves more design complexity.

2.3 Implications on StageNet

Stage level reconfiguration granularity presents itself as a good candidate because of following reasons:

- Stages can be looked upon as boundaries in a logical as well as a circuit sense. A logical boundary because the pipeline architectures divide work at the level of stages (like fetch, decode, etc.). A circuit boundary because the data signals gets latched at the end of every pipeline stage. Both these factors are helpful when reconfiguration is desired with a minimum impact on the original performance.

- Stage based reconfiguration scales well with the increase in available redundant spares (see fig 2).

- A stage based reconfigurable design is easy to validate, because there is a very limited interaction with the micro-architecture.

- And lastly, in the proposed architecture (StageNet), pipelines share the stages among them as spare components which makes the system inherently redundant.

A high-level picture of the StageNet is presented in Figure 3. Processor cores within this system are designed as part of a high speed network-on-a-chip, where each stage in a coarse-grained processor pipeline corresponds to a node in the network. A horizontal slice of the architecture is equivalent to a logical processor core. Such a system would isolate nodes that are deemed defective, and configure pipelines to share certain stages. As these nodes wearout and eventually fail, system will exhibit graceful degradation in its performance, and a gradual decline in throughput. The next section elaborates on the design of StageNet architecture and shows performance evaluation for the same.

3. STAGENET: A RECONFIGURABLE CMP FABRIC

By making structures on the chip both simple and regular with a straightforward communications interconnect, architectures can
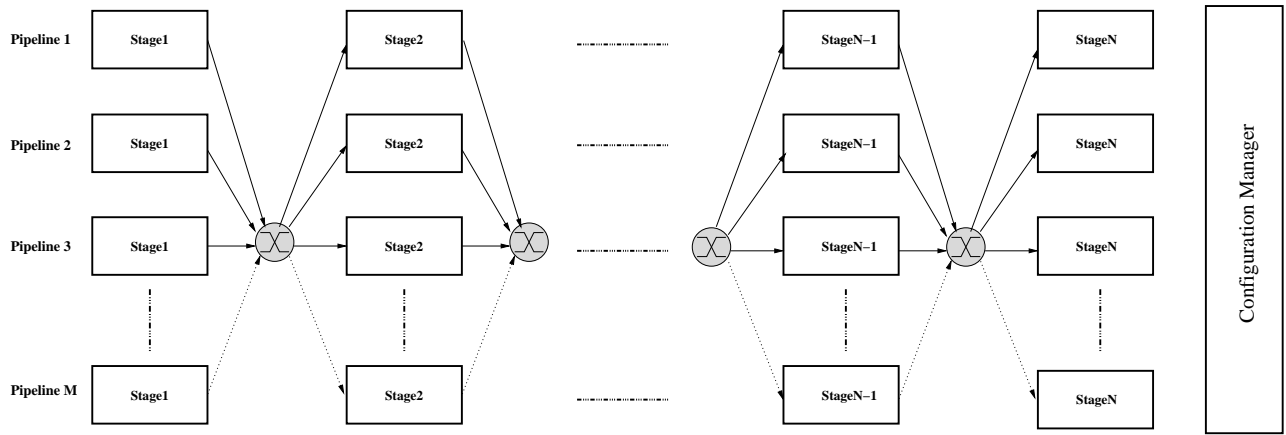
Figure 3: StageNet architecture. The figure above shows logical pipelines that have their stages interconnected for maximum opportunity of reconfiguration. Each horizontal slice (a single pipeline) is equivalent to a logic processing core. This figure has M, N-stage pipelines.

be built which can maintain operation despite a large number of non-working components. The StageNet architecture is inspired by this observation. It is a multiprocessor architecture where the pipeline stages of a traditional microarchitecture, are used as the unit of replication. Stages are organized as a tightly-coupled, high-performance network-on-a-chip, communicating with each other through an interconnect rather than pipeline latches, allowing for a high degree of system reconfigurability. The overall objective of this design is to have a scalable fault tolerant multiprocessor system with built in redundancy and reconfiguration capabilities. This section describes the proposed architecture and shows preliminary performance figures.

3.1 Reconfigurable CMP Fabric

The StageNet architecture, shown in Figure 3, consists of an interwoven fabric set of simple pipeline stages. Each row forms a logical processing core, a StageNet pipeline. The number of stages is variable and dependent upon the base pipeline architecture that is used to form the StageNet. Each stage is connected to simple network routers (represented by the shaded circles) allowing it to communicate with the neighboring stages in the pipeline. These routers can be considered surrogates for the pipeline latches in a conventional pipeline. The configuration manager (shown at the far right in Figure 3) handles system-wide control (i.e., definition of logical cores, dynamically re-routing around ailing components, etc.). At initialization time, each logical core is assigned a core ID and allocated a single instance of each pipeline stage. This allocation of resources defines the routing tables for the intermediate routers. The simplest policy is to allocate stages belonging to the same row to the same core. But, by no means are these allocations static, failure of pipeline stages can trigger system reconfiguration forcing neighboring pipelines to share stages on a time-multiplexed basis.

The StageNet design enables the re-provisioning of resources in order to maintain operation (with potentially degraded performance) in the presence of multiple component failures. The resources allocated to a logical core can also be changed over time to reduce hot spots on the chip. During system reconfiguration, the available resources are partitioned into logical cores, and the routing tables are updated. For example (see figure 5), if a stage (X) in row 2 encounters a failure, the router may be re-configured to forward instruction packets from the preceding stage ($X - 1$) in row 2 to a working stage (X) of, say, row 1. Since the num-

ber of non-defective components on the chip will decrease over time, the number of complete logical cores that can be formed will also decrease with age. Thus, the overall system performance will gracefully degrade as nodes periodically fail.

Advantages of the StageNet design:

- **Reconfiguration Flexibility:** The fundamental interconnection change (from pipeline latches to programmable routers) provides a tremendous opportunity for fault tolerance through reconfiguration. Failed stages can be easily bypassed by routing instructions to an alternate spare while still utilizing the remaining stages in the original pipeline.

- **Inherent Redundancy:** When used to form a multicore system, such a design can share its pipeline stages with neighboring processors. This ability to share stages provides redundancy by-design without the additional costs of cold-spares.

- **Scalable issue width:**[2] In addition to its reliability features, the StageNet architecture also supports configurations that optimize for performance. A single logical core can potentially be allocated more than its share of pipeline stages. For example, the issue stage of one pipeline can potentially scatter computation to multiple execute stages and the results can be gathered by a single writeback stage (emulating a wide-issue system). This is made possible by the flexibility of interconnect between the stages.

Along with these benefits, there is one major challenge facing this architecture, namely the overhead of the routers that support the inter-stage communications. The performance overhead associated with these routers is evaluated and discussed in the next subsection.

As a preliminary study, the number of stages in the StageNet pipeline was varied to observe the impact of pipeline depth on performance. The router communication overhead was fixed at 1 cycle for every transfer between the stages. Figure 4 shows the performance measured as CPI for a set of benchmarks chosen from the MiBench [13] benchmark suite while the pipeline depth is varied. The performance impact from the increase in pipeline depth, as

---

[2]Although the evaluation for this feature of StageNet is beyond the scope of this paper, it was one of the advantages that motivated the design

seen in figure 4, is negligible when compared to the overhead associated with the router. Even for the case with five stage pipelines (second bar for every benchmark), almost all the benchmarks show approximately 2X increase in the CPI over the baseline architecture [3]. Therefore, the major component of performance degradation comes from the communication overhead that is discussed further in the following subsection.
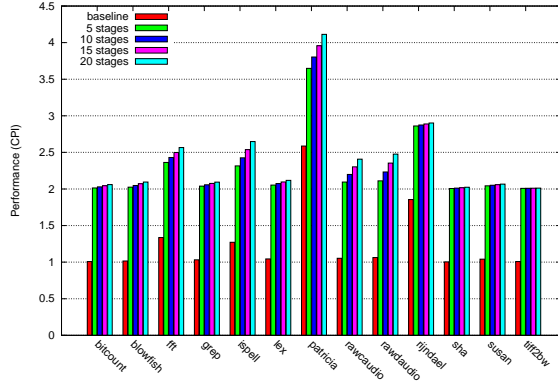

Figure 4: Single thread performance for a StageNet pipeline with variation in the number of pipeline stages. The transmission delay is fixed at 1 cycle.

3.2 Communication Overheads

The communication between stages happens via on-chip network routers. The router for StageNet, as shown in figure 5, is a standard textbook design [11] with buffering at its inputs and round-robin allocation logic for the non-blocking crossbar switch. It performs the task of receiving instruction packets from stages upstream and transmitting them to available stages downstream. The communication overhead in the StageNet architecture comes from two sources:

Transmission delay: This is the delay incurred from transmitting an instruction from one pipeline stage to the next. The reason for this delay is the limited communication bandwidth between the pipeline stages. For instance, if the instruction size is 128 bits (including operands), then a 64 bit wide communication bus would require two cycles to transmit it.

Congestion delay: If the number of input stages to a router is more than the number of output stages, then that router is said to be congested. Such a scenario arises when the pipeline stages start failing and the remaining stages have to be time-multiplexed between the existing threads (see figure 5). The delay introduced is referred to as congestion delay. Early in the lifetime when all stages are functioning properly, no congestion delay is observed.

In order to see the impact of varying transmission delays ($tdelay$) on single thread performance, the same benchmarks as before were executed. The number of stages was kept fixed at 10, and the transmission delay was varied from 0 to 8 (see figure 6). The resulting performance scaled almost linearly with the transmission delay. This is also intuitive because after every instruction in the pipeline, the transmission delay will force the insertion of NOPs (no-operations). Thus, the pipeline stages sit idle for at least 'transmission delay' number of cycles after processing every instruction.

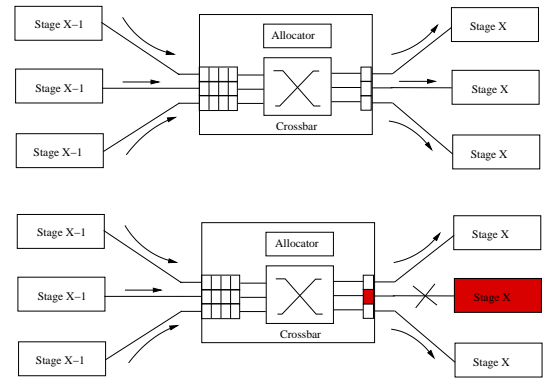[3] A conventional five-stage inorder pipeline

Figure 5: StageNet fabric handling a fault in a pipeline stage. The top figure shows the router under normal conditions. The bottom figure shows the configuration after the failure of Stage X in the second pipeline. Here, the router redirects the incoming traffic to the working stages downstream.

The impact of congestion delay is harder to quantify for a single thread because congestion is seen only when multiple benchmarks are running on the system. Furthermore, it is necessary to model failed pipeline stages to induce congestion. This is a topic for section 4 where the system throughput is computed in the presence of stage failures.
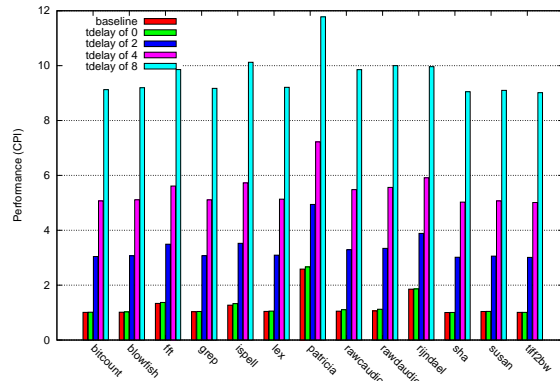

Figure 6: Performance of a StageNet pipeline with different transmission delays. The number of stages is fixed at 10.

3.3 Performance Enhancement

The performance of the design suffers immensely from the overhead of transferring instructions between stages since every instruction has to go through a network with a variable amount of delay. Another observation is that since each of these stages usually take a single cycle to execute the instruction, they are sitting idle for rest of the time waiting for the next instruction to arrive. A very fruitful optimization would be to increase the granularity of communication between the stages to a collection of instructions rather than sending one at a time. Let this bundle of instructions be called a *macro-op*. A pipeline stage will take multiple cycles to process such a macro-op because the hardware resources within a stage remains unchanged. As a result, the transmission of the next macro-op can be overlapped with the processing of the current macro-op, increasing overall hardware utilization. Lessons learned from out-of-order architectures support the fact that a higher resource utiliza-

tion usually translates into better overall performance.

For this work, the macro-ops were formed by combining consecutive instructions in the program assembly code. The macro-ops were generated during compilation time with two guiding constraints:

- the number of instructions that can belong to a single macro-op had an upper bound
- macro-ops cannot span branch instructions

Essentially, macro-ops are similar to the program basic blocks, only with a limit on the maximum number of instructions. Performance results for experiments that varied the macro-op size are shown in figure 7. In these experiments, the number of pipeline stages was fixed at 10 and the transmission delay was set at 4 cycles. The results demonstrate that performance greatly benefits from the use of macro-ops. Generally speaking, benchmark performance increases proportionally with increasing macro-op sizes. But, as evident from the figure 7, this holds true only as long as the macro-op size is less than or equal to the transmission delay. When the macro-op size exceeds the transmission delay (bars corresponding to macro-op sizes 6 and 8), the performance improvement saturates. This is an expected result because initially when the macro-op size is increased, the longer execution times required for macro-ops masks the transmission delays and the performance improves. Once the macro-op size becomes greater than the transmission delay, the bottleneck comes from the pipeline stages that execute the macro-ops, and this introduces stall cycles.
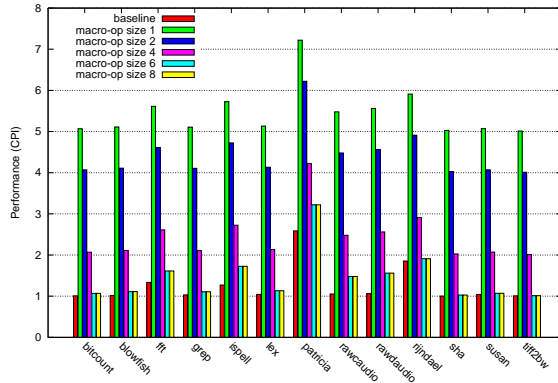

Figure 7: Performance of a StageNet pipeline with variable sizes of macro-ops. The transmission delay is fixed at 4 and number of stages at 10.

For the evaluation of a CMP system, in addition to the single thread performance, throughput is an important metric. Especially in the face of failing design components, the rate of throughput degradation gives insight into the resilience of a design. This is discussed in more detail by the following section.

4. THROUGHPUT EVALUATION

The benefits of prolonging the life of a processor are marginalized if its computational capacity is severely diminished. Therefore, the key metric used to evaluate the StageNet architecture is its throughput over time. In other words, the "quality of life" of the processor is as important as the magnitude of the lifetime extension achievable. In the context of this paper, throughput is defined as the amount of work done per unit time where work refers to the number of instructions committed, and the unit of time is CPU cycles.

Therefore, throughput is simply the summation of IPCs (instructions per cycle) for all threads running on the CMP.

In the event of failures, throughput is expected to diminish because fewer resources are available in the system. A system that degrades its throughput gracefully (i.e. at a slow rate) with respect to the number failures is naturally a good resilient design. This was the primary motivation for using throughput against failures as a metric for StageNet CMP fabric. In addition to this, congestion delay in the StageNet architecture comes into play only when some of the stages in the pipeline fail (see section 3.2). Thus, a throughput study would shed light on this aspect of the design as well.

A Monte-Carlo analysis was conducted (see figure 8) to measure the throughput with respect to the number of failures for the StageNet CMP. Each simulation for the Monte-Carlo was an experiment where faults were injected into the pipeline stages of a StageNet CMP instance, and the throughput was computed as the failures accumulated in the system. Each core of the StageNet CMP ran a benchmark from the MiBench suite. The parameter values for the StageNet are mentioned in the caption for figure 8. As shown in the figure, a similar analysis was done for a traditional CMP system.
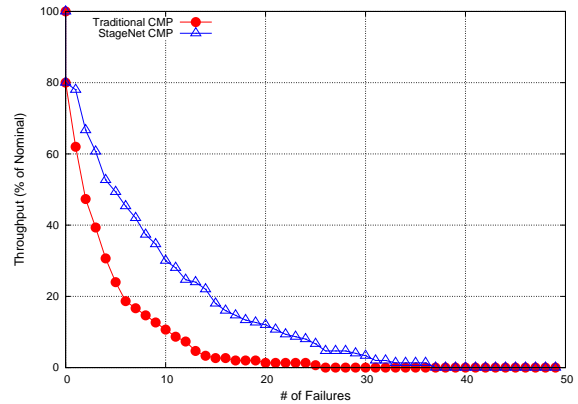

Figure 8: Throughput degradation with the injection of failures in the StageNet CMP and a traditional CMP. The StageNet architecture has macro-op size = 6, transmission delay = 4, and number of stages = 10. Both the systems start off with resources equivalent to 4 cores, and the throughput is normalized to that of the traditional CMP design when no failures exist.

The throughput degradation results highlight the resilience of the StageNet CMP over a traditional CMP design. Note that for a traditional system, any failed device in a core leads to its failure. The rate of degradation is significantly less for the StageNet CMP, which implies that it can tolerate more failures and maintain a higher throughput, throughout its operational lifetime. The plot also shows that the congestion delay does not have a significant impact on the system throughput.

5. CONCLUSION

Technology trends project high failure rates for future CMOS technology generations indicating a growing need for revolutionary new designs that can maintain functionality in the presence of multiple failed components.

This paper demonstrates that traditional approaches to coarse-grained replication for reliability is insufficient for tolerating high failure rates. In general, the finer the granularity of reconfiguration, the better the projected lifetime of the system. One caveat

to this generalization is that beyond a certain level, the complexity of implementing this finer-grained reconfiguration nullifies the gains in MTTF. As a solution, this paper presented StageNet, a highly reconfigurable CMP fabric composed of interwoven pipeline stages communicating with each other over an on-chip network. StageNet allows for reconfiguration at the granularity of a pipeline stage without adding significant interconnection design complexity. The advantage of being reconfigurable at a finer granularity than a processor core results in a longer lifetime for the system.

Finally, a Monte-Carlo analysis was done to evaluate the throughput degradation of the StageNet CMP with the injection of failures. The results demonstrated a significantly more graceful degradation of throughput for the StageNet CMP as compared to a conventional CMP, with a difference of as much as 20% in throughput for the same number of failures.

6. REFERENCES

[1] Openrisc 1200, 2006. http://www.opencores.org/projects.cgi/web/or1k/openrisc_1200.

[2] N. Aggarwal, P. Ranganathan, N. P. Jouppi, and J. E. Smith. Configurable isolation: building high availability systems with commodity multi-core processors. In *Proc. of the 34th Annual International Symposium on Computer Architecture*, pages 470–481, 2007.

[3] J. S. S. T. Association. Failure mechanisms and models for semiconductor devices. Technical Report JEP122C, JEDEC Solid State Technology Association, Mar. 2006.

[4] T. Austin. Diva: a reliable substrate for deep submicron microarchitecture design. In *Proc. of the 32nd Annual International Symposium on Microarchitecture*, pages 196–207, 1999.

[5] K. Bernstein. Nano-meter scale cmos devices (tutorial presentation), 2004.

[6] J. A. Blome, S. Feng, S. Gupta, and S. Mahlke. Online timing analysis for wearout detection. In *Proc. of the 2nd Workshop on Architectural Reliability (WAR)*, pages 51–60, 2006.

[7] S. Borkar. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, 2005.

[8] F. A. Bower, P. G. Shealy, S. Ozev, and D. J. Sorin. Tolerating hard faults in microprocessor array structures. In *Proc. of the 2004 International Conference on Dependable Systems and Networks*, page 51, 2004.

[9] A. Christou. *Electromigration and Electronic Device Degradation*. John Wiley and Sons, Inc., 1994.

[10] K. Constantinides et al. Bulletproof: A defect-tolerant CMP switch architecture. In *Proc. of the 12th International Symposium on High-Performance Computer Architecture*, pages 3–14, Feb. 2006.

[11] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[12] D. Dumin. *Oxide Reliability: A Summary of Silicon Oxide Wearout, Breakdown, and Reliability*. World Scientific Publishing Co. Pte. Ltd., 2002.

[13] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *Proc. of the 4th IEEE Workshop on Workload Characterization*, pages 10–22, Dec. 2001.

[14] P. Shivakumar, S. Keckler, C. Moore, and D. Burger. Exploiting microarchitectural redundancy for defect tolerance. In *Proc. of the 2003 International Conference on Computer Design*, page 481, Oct. 2003.

[15] D. Siewiorek et al. *Reliable Computer Systems: Design and Evaluation, 3rd Edition*. AK Peters, Ltd., 1998.

[16] J. Smolens, J. Kim, J. Hoe, and B. Falsafi. Efficient resource sharing in concurrent error detecting superscalar microarchitectures. In *Proc. of the 37th Annual International Symposium on Microarchitecture*, pages 256–268, Dec. 2004.

[17] J. C. Smolens, B. T. Gold, B. Falsafi, and J. C. Hoe. Reunion: Complexity-effective multicore redundancy. In *Proc. of the 39th Annual International Symposium on Microarchitecture*, pages 223–234, 2006.

[18] L. Spainhower and T. Gregg. IBM S/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective. *IBM Journal of Research and Development*, 43(6):863–873, 1999.

[19] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The case for lifetime reliability-aware microprocessors. In *Proc. of the 31st Annual International Symposium on Computer Architecture*, pages 276–287, June 2004.

[20] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The impact of technology scaling on lifetime reliability. In *Proc. of the 2004 International Conference on Dependable Systems and Networks*, pages 177–186, June 2004.

[21] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *Proc. of the 32nd Annual International Symposium on Computer Architecture*, pages 520–531, June 2005.

[22] D. Sylvester, D. Blaauw, and E. Karl. Elastic: An adaptive self-healing architecture for unpredictable silicon. *IEEE Journal of Design and Test*, 23(6):484–490, 2006.

[23] S. Vrudhula, D. Blaauw, and S. Sirichotiyakul. Estimation of the likelihood of capacitive coupling noise. In *Proc. of the 39th Design Automation Conference*, pages 653–658, 2002.

[24] C. Weaver and T. M. Austin. A fault tolerant approach to microprocessor design. In *Proc. of the 2001 International Conference on Dependable Systems and Networks*, pages 411–420, Washington, DC, USA, 2001. IEEE Computer Society.

[25] E. Wu et al. Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides. *Solid-State Electronics*, 46:1787–1798, 2002.

[26] J. Zeigler. Terrestrial cosmic ray intensities. *IBM Journal of Research and Development*, 42(1):117–139, 1998.