Towards Scalable Reliability Frameworks for Error Prone CMPs

Joseph Sloan
Coordinated Science Laboratory
1308 W. Main Street
Urbana, IL 61801
jsloan@illinois.edu

Rakesh Kumar
Coordinated Science Laboratory
1308 W. Main Street
Urbana, IL 61801
rakeshk@illinois.edu

ABSTRACT

As technology scales and the energy of computation continually approaches thermal equilibrium [1,2], parameter variations and noise levels will lead to larger error rates at various levels of the computation stack. The error rates would be especially high for post-CMOS and nanoelectronic systems as well as for probabilistic [3] and stochastic architectures [4]. N-modular redundancy (NMR) at the core-level has been proposed as a way to attain system reliability goals for multi-core architectures. While core-level DMR and TMR have been shown to be effective when errors are rare, a large amount of core-level redundancy will be required for attaining system reliability goals in face of high error rates. This makes voting latency and bandwidth significant performance bottlenecks for such systems. In this paper, we present a scalable NMR framework for error prone chip multiprocessors(CMPs). The framework supports *in-network fault tolerance* where voting logic is integrated into routers to allow for truly distributed voting. The in-network fault tolerance router utilizes the expected redundancy in vote messages, to reduce some of the blocking overhead incurred at the leader, and also provide a mechanism to trade-off network bandwidth with latency. Our framework also supports *proactive checkpoint deallocation* which allows cores participating in voting to continue on with execution instead of waiting on notification from the voting logic. Finally, the framework supports *dynamic constitution* that allows an arbitrary core on this chip to be a part of an NMR group. This allows bypassing faulty cores as well as scheduling for performance. Our experiments show significant performance/bandwidth benefits from these optimizations.

Categories and Subject Descriptors

B.8.0 [**Performance and Reliability**]: General; C.1.3 [**Processor Architectures**]: Other Architecture Styles—*Adaptable architectures*

CASES'09, October 11–16, 2009, Grenoble, France.

General Terms

Design, Performance, Reliability

Keywords

In-network Fault Tolerance, Dynamic Consitution

1. INTRODUCTION

As technology scales and the energy of computation continually approaches thermal equilibrium [1, 2], parameter variations and noise levels will lead to larger error rates and, therefore, larger overheads of error management at various levels of the computation stack. For example, variations in process, voltage, and temperature add complexity and overhead at the circuit level in order to achieve correct operation over a range of parameter values. [2,5] With technology scaling, many of these variations are becoming very difficult to manage and tolerate at the circuit-level, forcing upper levels of computation stack to deal with a reduction in reliability [6, 7].

While errors have traditionally been rare in computer systems, error rates and hence the overhead of error management are expected to be high for systems built using post-CMOS [8] and nanoscale [9] devices. Such substrates exhibit extreme levels of non-idealities [10], especially in face of process, temperature, and voltage variations. These non-idealities can result in significant number of transient and permanent failures as confirmed by PTM [11] and CNFET [12] models.

Error rates will be high even for architectures where voltage and frequency are overscaled for throughput and power improvements. For example, probabilistic SOCs [3] aim at reducing the voltage of probabilistic co-processors to a subthreshold voltage. This overscaling results in a large number of architecture-level errors. Similarly, stochastic processors [4] aim to reduce the voltage of processors below their critical voltage. They also aim to increase the frequency of processors above the critical frequency. Under such situations, error rates can be high.

N-modular redundancy (NMR) at the core-level has previously been proposed as a way to attain system reliability goals for multi-core architectures. While the exact implementations may vary, core-level NMR typically involves running multiple copies of a program on different cores. Every write is buffered locally until voting is triggered (a trigger can be time-based or event-based) when one or more written values are voted on. Based on the results of voting, each copy of the program continues or rolls back to a check-

point (which was created using the previously validated architecture state). The attractiveness of core-level NMR for multi-core architectures is primarily due to the flexibility it provides in terms of dynamic power/reliability/performance tradeoffs.

While core-level dual-modular redundancy (DMR) and triple-modular redundancy (TMR) have been shown to be effective when errors are rare, a large amount of core-level redundancy will be required for attaining system reliability goals in face of high error rates. Figure 1 illustrates this fact,[1] showing that even with a plurality voter (2ofN), [2] NMR groups of more than 3-8 cores may be necessary for relatively high system reliability targets and component reliabilities (*1- error rate*) of future technologies. In fact, some applications with strict performance requirements may require additional redundancy in order to maximize availability.

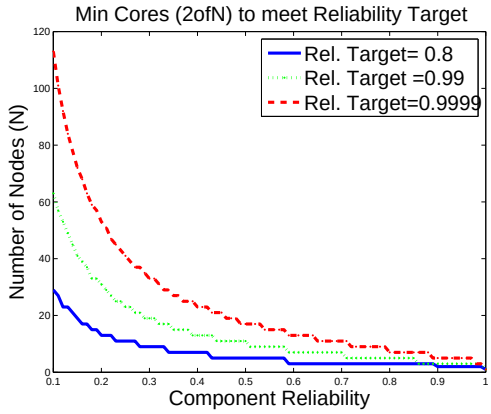

Figure 1: Minimum number of cores required in order to meet three different system reliability targets (i.e. 0.8, 0.99, and 0.9999) with varying component reliabilities shown on the x-axis.

Under such high degrees of core-level redundancy, traditional core-level NMR framework may not be effective as voting latency and bandwidth can become significant performance bottlenecks for such systems. This is due to the fact that the voting latency and bandwidth scale with the number of cores that form an NMR group. In this paper, we present a scalable NMR framework for error prone CMPs. The framework supports *in-network fault tolerance* where voting logic is integrated into routers to allow for distributed voting. The integrated router utilizes the expected redundancy in vote messages, to reduce some of the blocking overhead incurred at the leader, and also provide a mechanism to trade-off network bandwidth with latency. Our framework also supports *proactive checkpoint deallocation* which allows cores participating in voting to continue on with execution instead of waiting on notification from the voting logic. Finally, the framework supports *dynamic constitution* that allows an arbitrary core on this chip to be a part of an NMR group. This allows bypassing faulty cores as well as scheduling for performance. Our experiments show significant performance/bandwidth benefits from these optimizations.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the need for a low-latency, low-bandwidth NMR framework in face of high error rates. It also motivates the need for the core-level NMR framework to allow scalable dynamic coupling of cores. Section 4 presents a scalable NMR framework with support for in-network fault tolerance and proactive checkpoint deallocation. Section 6 analyzes the benefits of our optimizations. Section 7 summarizes and concludes.

2. RELATED WORK

There exists much previous work in using NMR to improve a computation structure's reliability. [13] This includes classical fault-tolerant systems using NMR, such as the Sperry Univac 1100/60, Tandem S2, C.vmp, Software Implemented Fault Tolerance(SIFT), and FTMP systems. [14–16] The C.vmp system utilized triplicates and voting whenever accesses to the bus occurred. The FTMP divided programs into tasks which each ran on a processor/scratch pad pair (with three copies of memory), where values are voted on before being written to or from the scratch pad. Many of these previous fault-tolerant systems used similar techniques, including physical isolation to prevent fault propagation, tightly synchronized logic, adaptive policies in the case of failures.

More recent work commonly uses a DMR approach to processor logic, such as DIVA [17] which utilizes cores in a leader-checker fashion, such that the simplified more reliable core acts as a checker and runs behind the leader, verifying its operation dynamically to protect against transient faults. The slack between the cores is compensated for by the leader providing prefetch and branch prediction results to the trailing core. Tandem Integrity S2 is a classic commercial example of a TMR type system which utilizes multiple layers of dual redundancy on system components and triple redundancy for processor logic. [18] IBM Power6 is also another commercial example of lockstep execution between two cores on-chip. [19] All of these approaches are limited in their extension to a flexible NMR framework as the cores are tightly coupled together.

The Reunion approach [20] introduces an architecture for utilizing DMR groups on a chip multiprocessor. They also describe techniques allowing overheads associated with coherence complexities to be reduced, termed relaxed input coherence. However, their approach is also limited by the use of tightly statically coupled cores. Their DMR groups, similar to previous commercial examples, also used short comparison intervals.

There is some work involving the use of a CMP with the capability to reconfigure groups similar to the work in this paper. Dynamically Coupled Cores(DCC) [21] utilizes a shared bus for the communication fabric and devises basic protocols for supporting the dynamic coupling of DMR groups. Due to bandwidth constraints though, they are restricted to using only large comparison intervals (3000-10000 cycles). Their technique is limited in both the error rates and degree of NMR groups supported. Dynamic DMR(DDMR) [22] attempts to solve this problem by uti-

[1]Figures 1to 3 are derived from an analytical model of NMR based reliability frameworks.

[2]In a plurality voter, a minimum of 2 of N modules must agree on an output value in order to proceed with the program. If the largest set of values is unique, it is always chosen. Otherwise, one of the non-unique largest sets is chosen at random in order to proceed.

lizing a partitioned design with a ring network in each partition. DDMR also uses short validation intervals (128 instructions or 24 stores). This partitioned strategy allows for microarchitecture/network designs, such as a ring buffer, which reduce the overall communication latency within the voting groups. As discussed below though, in face of heterogeneity, these designs can be very limited in aggregate performance and throughput.

Finally, there has been a study on applying the DCC approach to a multi-core environment using a traditional network on chip mesh. [23]. They find similar high overheads to using the basic DCC approach with a direct network. They also use larger comparison intervals, which require more complex protocols for synchronization and input coherence between cores within a DMR group. Further, they find that the overhead associated with their synchronization protocols are a limiting factor to the scalability of the DCC approach on a CMP.

The use of combining networks has also been studied in previous systems [24]. The micro-architectural approach for *in-network fault tolerance* similarly includes additional logic in the network to gather aggregate results and support bandwidth/latency tradeoffs. However, our approach also supports further integration with the reliability framework in order to allow for more efficient network architectures and voting/checkpointing protocols (including *proactive checkpoint deallocation* and *dynamic constitution*).

3. REQUIREMENTS OF AN NMR FRAMEWORK FOR ERROR-PRONE CMPS

Designing an NMR framework for error prone CMPs starts with the question of dynamic redundancy vs static redundancy. Static NMR for CMPs may be too expensive in terms of area and power in face of high error rates due to the large number of cores required to meet system reliability targets. Dynamic redundancy, on the other hand, allows dynamic performance/power/reliability tradeoffs and is, therefore, more attractive under such scenarios.

The second desirable feature of an NMR framework for error prone CMPs is *dynamic constitution*. Dynamic constitution refers to selecting an arbitrary core on the chip to constitute an NMR group. As opposed to dynamic redundancy (or dynamic coupling [21]), the composition of cores of an NMR group in a dynamic constitution framework can change completely over time. Dynamic constitution is useful for several reasons. First, it allows scheduling around cores with permanent faults. For example, in the face of permanent faults, if NMR groups are statically constituted and the number of failed cores in an NMR group drops below the threshold to meet the basic reliability target, the rest of the cores in the group are essentially rendered useless. Figure 2 illustrates this type of behavior by showing the number of working NMR groups over time with a CMP of N=90 cores divided into multiple partitions and a reliability target requiring 3 redundant cores for each thread. Each core is assumed to have a permanent failure rate which is normally distributed ($\mu = 500$, $\sigma^2 = 300$) and coupling is allowed only with cores belonging to the same partition (i.e., when one of the cores belonging to an NMR group fails, a new NMR group can be formed only with a core within the same partition). Figure 2 shows that a system with no partitions (i.e., where any core can join into an NMR group with any other core on the chip) shows the best overall throughput over the period shown.

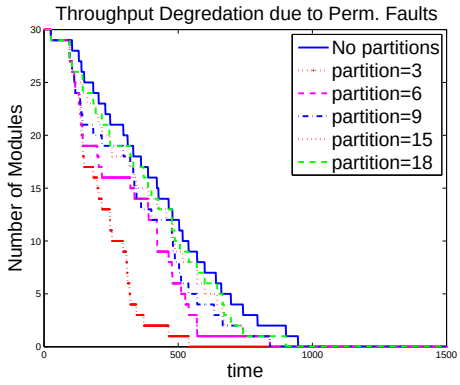

Figure 2: Throughput Degradation due to normally distributed permanent faults. N=90, NMR=3, Faults Normal($\mu = 500$,$\sigma^2 = 300$)

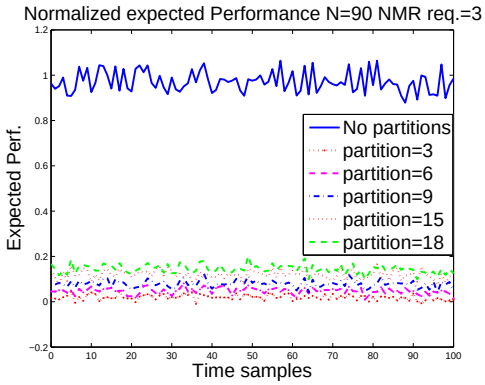

Figure 3: Normalized expected performance with optimal scheduling. N=90, NMR=3. Systems with finite partition sizes can drastically limit the overall throughput and expected performance.

Similarly, dynamic constitution facilitates better scheduling for throughput in face of heterogeneity due to manufacturing variations as well as core-level voltage/frequency scaling. For example, since performance of redundant cores are dictated by the slowest core in group, better overall throughput can be achieved when the system has flexibility in scheduling of the NMR groups (i.e., the cores used to constitute an NMR group). Figure 3 shows an example of how expected single thread performance is affected by the use of smaller partitions restricting the scheduling of NMR groups based on relative performance. Each core's performance (IPC) is normally distributed($\mu = 100$, $\sigma^2 = 50$) and cores are optimally scheduled within each partition to maximize the expected performance of threads within the group. Figure 3 plots one sample path for normalized expected performance with alternative partition sizes. The system with no partitions shows much greater expected single thread performance versus the partitioned systems, assuming an ideal scheduling mechanism.

Dynamic constitution may also be helpful in face of thermal constraints (to prevent NMR hotspots) and multiprogramming.

Yet another characteristic we want an NMR framework for error-prone CMPs to have is low voting latency. The equation below illustrates a simple relationship between the voting frequency, voting latency, and error rate in determining the associated system overhead. Assuming the voting group's recovery rate is exponentially distributed ($exp(\lambda)$), and the following primary parameters:

- Cf =Checkpoint Frequency.
- CV_t = The latency of tasks associated with the reliability framework per comparison interval (i.e. checkpointing, voting, input coherence).
- T_o = Random variable representing the total execution time between checkpoints, which is a function of the number of re-executions, Cf, and CV_t.

We then find that the expected overhead ($E[O_r]$) for a single unit of computation of length $1/C_f$ as:

$$E[O_r] = C_f * E[T_o] - 1 = e^{\lambda/C_f} * (1 + C_f * CV_t) - 1$$

With a fixed recovery rate and reliability framework latency, the optimal checkpointing frequency(for the least system overhead) can then be found and is shown below.

$$CkptFreq_{optimal} = \lambda + \sqrt{\lambda^2 + 4\lambda/CV_t}$$

Figure 4 utilizes these equations to plot the expected performance overhead of voting for different voting latencies for several potential error rates ($10^{-4} - 10^{-7}$) that have been reported in literature [8–10] for post-CMOS and nanoelectronic systems. These error rates are meaningful for probabilistic SoCs and stochastic architectures [3,4] as well. For each recovery rate (which is likely a fraction of individual core error rates, as some faults may be masked) shown on the x-axis, the optimal checkpointing frequency is used (from above equation). The results show that the voting latency represents a major restriction to scaling for error-prone CMPs, especially as longer delays are incurred between cores with the further technology scaling, increased core counts, and larger NMR sizes.

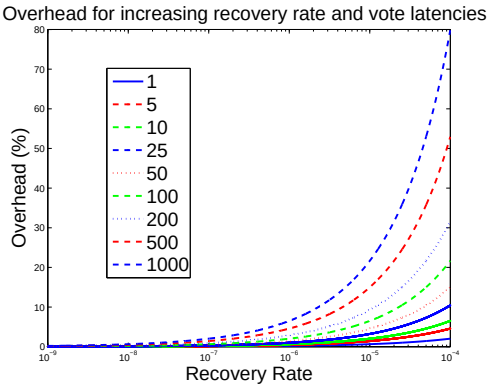

Figure 4: Voting module overhead vs error rate. The voting latency bounds the optimal comparison frequency, limiting how much the framework's overhead can be minimized.

Finally, an NMR framework for error prone CMPs needs to be bandwidth-sensitive. Figure 5 illustrates one situation in which bandwidth can become a bottleneck for future error rates. Assuming an optimal square mesh, the number of packets injected into the network for one comparison interval are easily calculated; as a finite number of votes from each core are sent to the leader. And a finite number of response messages are returned from the leader in a single interval. This constant is multiplied by the expected number of re-executions due to errors. Figure 5 shows the approximate utilization of the routers in the NoC based on the NMR size and comparison interval (Checkpointing frequency=200) Even for smaller numbers of NMR sizes the approximate bandwidth consumed by the voting framework alone can be significant (almost 25%). However with increased NMR sizes and non-optimal NMR mappings on chips with larger total numbers of cores, the bandwidth consumed can become even larger, (almost 40-50%).

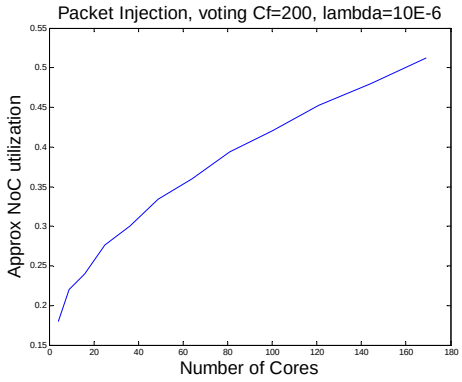

Figure 5: Approximate NoC Utilization. The voting process saturates the NoC bandwidth as more cores are included in the NMR group and as delays increase from additional routing on-chip.

4. PROPOSED FRAMEWORK

In this section, we describe an NMR framework for error prone CMPs that supports dynamic constitution, in-network fault tolerance, and proactive checkpoint deallocation. Before we describe the three optimizations, we discuss the basic reliability logic that the framework uses.

4.1 Basic Reliability Logic

The baseline reliability logic includes a simple hardware voter distributed throughout the mesh, shown in figure 6.

The sphere of replication includes the core and local L1 cache. As opposed to software oriented NMR, memory is not replicated and all requests to lower parts of the memory hierarchy are validated before being committed. [25]. Validation occurs by comparing a CRC hash of the processor states. The NoC, voters, checkpoints are all assumed to be of high reliability.

A multi-cycle version of the voting module is shown in figure 7 consisting of a basic FSM for voter control, a register file for data and statistics collection, and an adder used to tally the results. A register file is used to hold the temporary vote values as they arrive from the children cores. They are routed over the same network that other on-chip traffic utilizes. As vote values are passed to the hardware voter they are stored into the vote register file. Upon receipt of all messages the analysis phase begins by utilizing

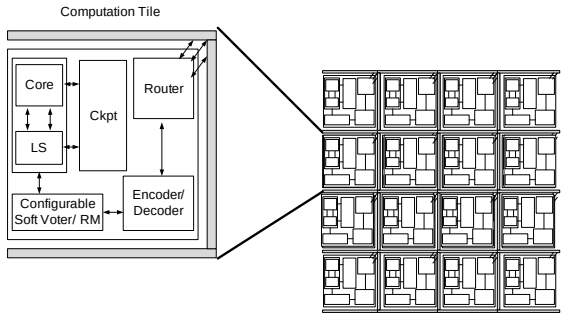

Figure 6: Basic tile included in reliability framework for CMP. Configurable logic associated with voting and reliability management resides between local processing resources and the network-on-chip interface.

the array of comparators to compare all values and tally the total counts for each value. Unique values are also recorded with control bits associated with each entry in the register file. Upon checking all entries, the FSM utilizes the Decision/Selection logic to determine whether a quorum is met and what actions should be taken for each core (continue, roll-forward, roll-back) Various policies and other information can be utilized by the decision/selection unit to determine what actions to take. The FSM then creates control packets communicating the actions to the children of the NMR group.

Conversely, a single cycle implementation utilizing a sorting network implemented as a butterfly network, shown in figure 8, could also be used. However, since the vote values are arriving asynchronously, vote values can also be analyzed in parallel with the time period in which the leader is waiting for receipt of all the children results. Hence, the vote register file and the comparator array function essentially as a Content-Addressable-Memory(CAM). The decision/selection unit can then determine the result without any delay after receipt of the last vote value.

The hardware voters also contain basic reliability management which maintains strict control over its locally associated core. This includes the ability to stall the core and L1 cache, disallowing any data to pass to external resources. The reliability manager also contains the ability to save local processor and L1 state as well as transfer remote state to the core and restart execution. The reliability management also is coupled to the local core to detect exceptions or halts. When these scenarios are detected the leader is signaled so appropriate action can be taken to recover. Additionally, the voter has an integrated timer/heartbeat mechanism to detect when a core has potentially stalled.

The cache itself is modified to maintain a "verified" bit, along with some slight coherence modifications discussed in [21] to account for redundant executions. When unverified lines are evicted a vote is triggered within the group. Upon verification, all cache lines in the group are marked as verified. Additionally, hardware storage for saving processor state and L1 cache are also included in the tile. In the baseline, checkpoints/recoveries are implemented by a basic direct copy, however future work involves the extension of the hardware checkpoint functionality for reducing storage capacity based on analysis of execution history.

4.2 Dynamic Constitution

Hardware voters have software-accessible configuration bits (see Figure 8). Dynamic constitution is supported by allowing the OS/runtime to manage the configuration of the NMR group's voters. Also, the routers along the static links between cores can be configured, assuming a deterministic routing policy (X-Y routing).

For the core-voter pair assigned as the leader, the basic configuration bit settings include the mode(leader), Ids for assigned children, initial timeout values on waiting for votes and the voting policy. For the cores assigned to the leader, their basic configuration settings include the mode(child), Id of their assigned leader, and threshold values used to trigger voting on certain events (number of writes, cycles, etc). Upon a triggered vote, the local core's reliability manager(RM) creates a vote message with the assigned leader as it's destination. The message can then be queued by the router similar to other outgoing traffic. Based on the leader's initial configuration, the router can then receive children messages, or make decision on events when cores are non-responsive.

After arbitration is completed by the voter a decision is made based on a given policy on how to continue. For example, if a majority of results match, the leader then sends control messages to each of the children signaling them to checkpoint and continue. If a majority is not reached, recovery is implemented by signaling all cores to rollback to a previously saved state.

Current objectives and system/environment conditions determine the best degree of redundancy and the actual mapping of cores within the NMR group.

4.3 In-network Fault Tolerance

The first optimization, in-network fault tolerance, includes aggregation logic within each router. The additional logic buffers and analyzes the vote messages coming into a router and forwards a smaller number of messages that represent the unique values being voted on and their current count. For example, if a router receives three vote messages, with values A, A, and B, it will forward two messages - (A,2) and (B,1). If the three vote messages have values A, A, and A, the router will forward one message - (A,3). So, the optimization utilizes the expected redundancy in vote messages, to reduce some of the blocking overhead incurred at the leader, and also provide a mechanism to trade-off network bandwidth with latency.

To implement in-network fault-tolerance, the routers are initially programmed with the number of expected votes for a given group, that may pass through a router. The router detects 'vote' related flits in parallel with the virtual channel and switch allocation stages. Multiple CAMs for each voting group (or a partition-able CAM) are used to cache the temporary vote values, shown in figure 9. The value of the vote is used as the address to the CAM, and the count is used as the data stored in the table. As values are matched in the CAM, the count is simply added to previous count. Upon receiving the expected number of votes for a given NMR group, the voter control within the router creates messages to forward on the contents of the CAM table in the same format. The *reduction network* (we call the NoC augmented with the aggregation logic a reduction network) can also be adjusted to change the degree at which bandwidth is traded off with latency, by adding limits to the time routers wait

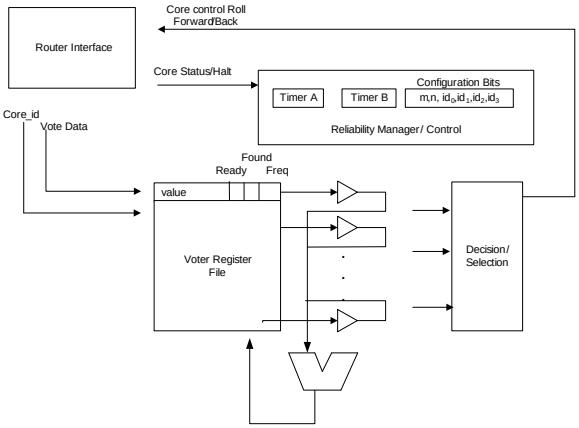

Figure 7: Multi-cycle implementation of hardware voter/Reliability manager. Limited by the area/power required for the sorting network included in critical path.

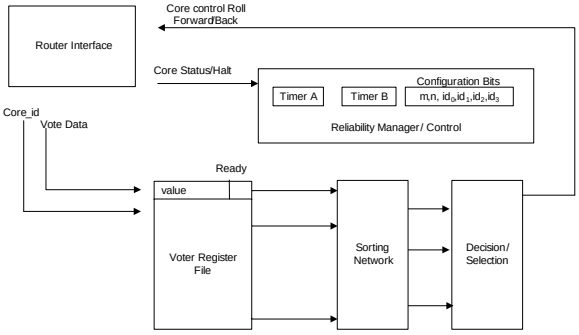

Figure 8: Single-cycle implementation of hardware voter/Reliability manager. Including Serial/parallel interface to vote data from cores and control includes FSM to solve mutual exclusion problem for data set and then create returned control messages.

for additional vote messages of the same voting phase before forwarding aggregated results onward.

Note that by incorporating the voter register file and associated comparison logic in the router as multiple CAMs or a single partitionable CAM, the router can participate in multiple concurrent voting phases. Also, this optimization does require an additional byte of control data to be communicated, associated with the number of values seen so far. However, this overhead is minimal compared to the bandwidth reduced due to aggregation.

4.4 Proactive Checkpoint Deallocation

The second optimization includes additional checkpoint buffer storage in each core, in order to partially hide the requirement that cores wait until receiving verification to continue on with execution.

For example, the proactive checkpoint deallocation can be used when each core has three local checkpoint buffers. Each of the cores and routers in this case appends two more bytes of control information along with the vote message. These two bytes contain the unique id of the core which currently holds the checkpoint associated with the particular vote value. The other byte is set by the originating core

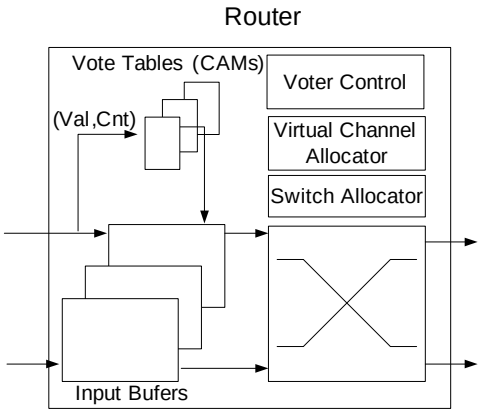

Figure 9: Basic router architecture including additional voting related logic.

and contains the current number of used checkpoint buffers. The vote tables (CAMs) within each router are extended to maintain the current core assigned to checkpoint (and its total number of checkpoints) for every unique value in table. As vote values are matched within the reduction network, the number of checkpoints stored at the originating core for the currently arriving vote and the previously received vote stored in the vote table are compared. The lesser of the two is chosen and the router then sends a 'checkpoint deallocate' message to the core id associated with the lesser of the checkpoint counts. The process continues, potentially deallocating more checkpoints depending on values received, and with the guarantee that at least one core will always store a checkpoint associated with a unique vote. This then allows recovery by rolling state back to this unique state, in the case the overall vote fails to reach agreement. In the ideal case all but one of the cores are allowed to deallocate their particular checkpoint and continue on with execution, while one lone core must wait until receipt has been received from the leader that the vote succeeded. So although the NMR group is dictated by the slowest core, in aggregate, the policy above will automatically adapt to various performance variations and previous checkpoint histories to minimize the aggregate time spent waiting for verification over multiple comparison intervals.

As vote values are matched within the reduction logic the voter can make a local decision and send control back to the corresponding core to deallocate its checkpoint and continue. Using this scheme the actual checkpoints are more efficiently balanced across the group, reducing the buffer size required for each core. The disadvantage to this approach though, is the increased potential penalty for roll-forward events(or remote transfers). Since these events are not as common as other framework events (less than 5%), the penalty from remote transfers can be amortized in the total recovery costs. Similar to the first optimization, additional marginal bandwidth/storage is required for communicating the number of allocated checkpoints and the core which is currently storing a particular checkpoint for a phase of execution. This allows for a simple load balancing policy to schedule checkpoints on the core with more buffer slots available.

There is no additional configuration setup for this optimization, although some policy variations could be included

in the basic configuration.

5. METHODOLOGY

The M5 simulator [26] was used to implement the primary components of the reliability framework, and a set of workloads from the Spec benchmark suite was simulated on top of the framework. Aspects of the reliability framework that were modeled carefully included the voting logic associated with aggregation and mutual exclusion policies, which were implemented in a simple router architecture included in the simulator. The vote and control messages are 8 bytes. Recovery mechanisms for rolling back to checkpoints were implemented by saving the processor state, L1 dcache, and I/O state. Additionally, we included a simple modification to the cache reflecting the status of verification for lines, and triggering voting.

Our experiments used a mesh topology as the interconnect, with regular dimensions and optimal leader placement minimizing hops. Multiple parameter values were simulated for the leading hardware voter – latency (1-2n cycles), Vote Bandwidth (4-100 bytes), and various triggers(write back, write limit = 2-128, cycle limit). However, we present most results for relatively short comparison intervals due to expected error rates and a reduction in size/complexity requirements for the checkpoints per core.

Basic error behavior was simulated by injecting faults as single bit flips into the actual data being compared. Fault arrivals to each core were modeled as a Poisson process (time between faults being exponentially distributed) Given a fault at time t, a single bit flip, which is uniformly distributed, is then applied. This model guarantees the injection of only single bit errors at a time per core, similar to other hardware focused fault injection campaigns. The impact of more complex error scenarios on the system will potentially lead to further opportunity for performance improvement from *in-network fault tolerance* . Additionally, the experiments included in the following section are based on using redundancy with single threaded benchmarks. The framework is generalized to multi-threaded workloads as well though. And these types of workloads will also further exacerbate reliability framework related overheads. (i.e. ensuring that all external inputs to redundant groups are coherent.)

Some of the basic parameters used in the simulations are shown below:

Component	Parameter
Interconnect Topology	2-Mesh
Routing	X-Y Routing
Channel Width	8 bytes
Processors	Alpha 21264
Clock Frequency	2GHz
L1 Cache per tile	32KB ICache, 64KB DCache
L2 Cache	4MB
Execution	Inorder
Voting Policy	Plurality

6. EXPERIMENTAL RESULTS

In this section, we analyze the potential benefits from using a reduction network and from proactive deallocation of checkpoints.

6.1 Fault Free Operation

Figures 10 and 11 show the relative speedups over the baseline for the reduction network and proactive checkpoint deallocation technique (which reduces most of the latency overhead). In most scenarios the reduction network does not show larger speedups ($> 10\%$). The benefits of the first optimization predominantly come from a reduction in the overall network bandwidth. With fault free simulations, the total number of bus cycles occupied by vote packets using the reduction network is reduced approximately $7\% - 34\%$, from 4 to 32 participating cores.

The figures (10 and 11) also illustrate the effect of delay at the leading hardware voter for NMR analysis and control message creation. With a delay of 2n cycles (n is the number of cores in an NMR group) in the leading voter, the optimizations exhibit greater speedups due to the increased penalty included in the voting process.

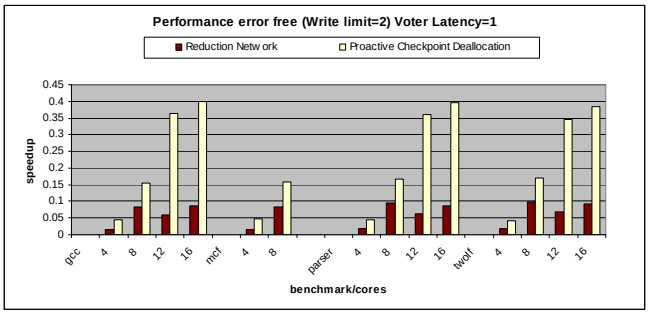

Figure 10: Performance speedups over the baseline for the first two optimizations (w/ WL=2, Voter latency=1). The x-axis is divided between each of the benchmarks with varying number of cores for the corresponding benchmark shown in between the benchmarks.

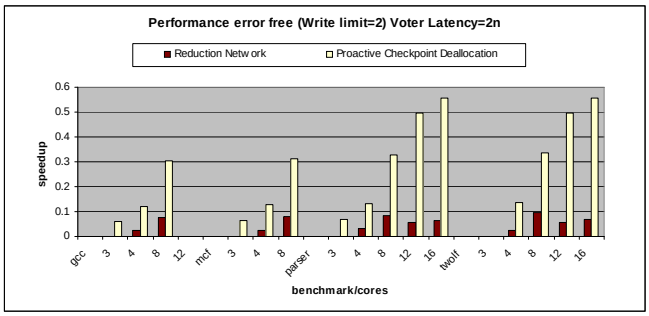

Figure 11: Performance speedups over the baseline for the first two optimizations (w/ WL=2, Voter latency=2n). The x-axis is divided between each of the benchmarks with varying number of cores for the corresponding benchmark shown in between the benchmarks.

Additionally, we see that although the reduction network imposes additional delays at each network hop which aggregates values, the reduction network more than compensates for this by the decrease in time spent on forwarding additional vote messages along the network to the leader. (in average or best case error scenarios).

Using a write limit (the number of writes after which vot-

ing is triggered) greater than 30 reduces the fault free performance improvements to about $2-15\%$ across the benchmarks. We also performed simulations using a write limit of 30, with alternative comparison bandwidths (i.e. 8-100 bytes, with 100 bytes showing $10-35\%$ improvements over $3-8$ cores). Figure 12 shows performance speedups for a slightly more moderate comparison rate (write limit of 10). Similar results are seen as in Figures 10 and 11 So these optimizations have the greatest affect on systems with short validation intervals due to high error rates or additional restrictions on the size/complexity of actual checkpoints.

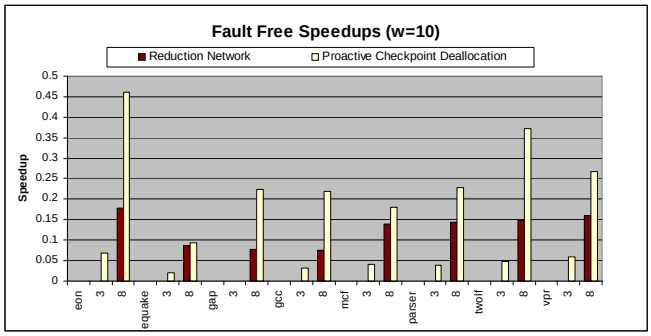

Figure 12: Performance speedups for the reduction network and proactive checkpoint deallocation with a write limit=10 and n=3,8

Figure 13 shows how the effective comparison rate changes when using the writeback trigger alone. For the most part , because comparison rates we simulated (controlled by write and time limits) were relatively high, not very many write-back triggered votes occurred. With lower comparison rates, we observed that the associativity of the cache had a much larger impact in determining the effective rate of comparison.

Figure 14 shows how the effective buffer size is a small proportion of the actual buffer size allocated. For these evaluations the biggest speedup came from having 2-3 additional checkpoints slots per core. Results shown in Figure 12, for the proactive checkpoint deallocation optimization using only three available checkpoint buffers also confirm that a small set of checkpoint buffers is sufficient to provide good performance gains for these types of environments.

6.2 Faulty Operation

We now discuss the operation of the framework in the face of simulated errors and recoveries.

Figure 15 shows the average performance speedups over a small sample set. Most benchmarks on average had an increase in speedups in the face of faults and recoveries. We found that performance depends not only on the number of additional votes triggered due to errors, but also on the error patterns upon which recoveries were based (because recovery penalties associated with re-execution depend directly on the error pattern). This stresses the importance of using an appropriate combination of triggers, to ensure proper comparison intervals with good fault detection characteristics.

Finally, Figure 16 shows how much the average voting latency across executions decreases with the reduction network. The graph shows that reduction network continues to provide performance benefits even in face of errors.

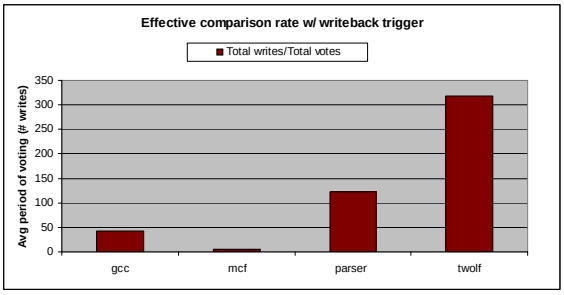

Figure 13: Effective comparison rate in terms of number of writes using the write back trigger alone.

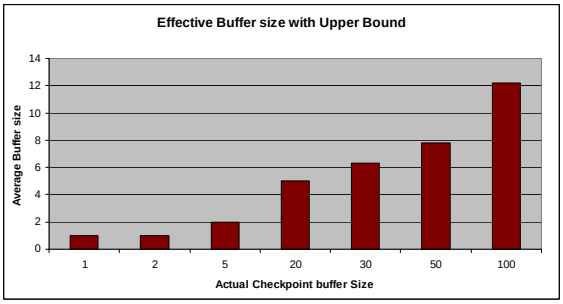

Figure 14: Sample of effective checkpoint buffer capacity with fixed sizes. In most cases the average buffer size was a small fraction of maximum size and showed very limited decrease in performance

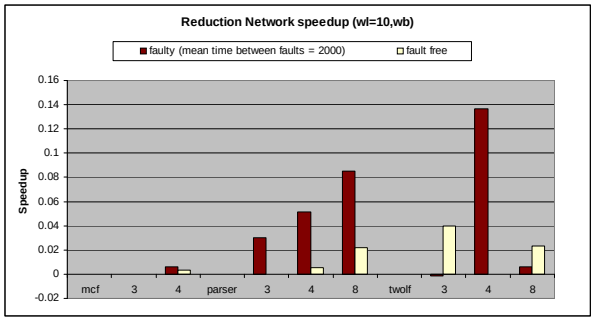

Figure 15: Performance speedups for the reduction network in the face of faults and recovered executions.

7. CONCLUSIONS

For future technologies which are inherently error prone, due to various deep submicron challenges, using higher degrees of NMR may become essential. Moreover, due to many additional sources of heterogeneity across the cores on the chip, the ability to dynamically constitute an NMR group with arbitrary cores on chip may also be needed. In this paper, we presented a scalable NMR framework for error prone CMPs. The proposed framework supports dynamic constitution of NMR groups. The framework also recognizes that large-scale NMR can be severely limited by the latency and bandwidth overheads included with inter-NMR and intra-NMR group communications. To that effect, the frame-

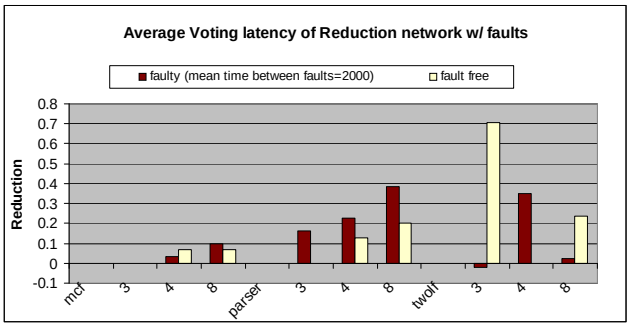

Figure 16: Decrease in average voting latency for the reduction network.

work supports in-network fault-tolerance that minimizes the bandwidth overhead of NMR. The framework also supports proactive checkpoint deallocation that allows reduced latency for NMR-based fault tolerance. Overall, we found that the proposed framework provides good performance improvements across future operating conditions characterized by adverse error behavior and the trend toward increasingly parallel CMPs.

We believe that the effectiveness of our framework (and optimizations) in achieving greater availability is only going to increase as the the number of cores on chip and within the NMR groups scales.

8. REFERENCES

[1] "International Technology Roadmap for Semiconductors 2005, http://public.itrs.net."

[2] C. Constantinescu, "Trends and challenges in vlsi circuit reliability," *Micro, IEEE*, vol. 23, no. 4, pp. 14–19, July-Aug. 2003.

[3] L. N. Chakrapani, P. Korkmaz, B. E. S. Akgul, and K. V. Palem, "Probabilistic system-on-a-chip architectures," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 3, pp. 1–28, 2007.

[4] *Stochastic Processors (or processors that do not always compute correctly by design)*, NSF Workshop on Science of Power Management. [Online]. Available: http://scipm.cs.vt.edu/Slides/2.RakeshKumar.pdf

[5] 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," 2002, pp. 389–398.

[6] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2003, p. 29.

[7] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. patel, "Characterizing the effects of transient faults on a high-performance processor pipeline," in *DSN '04: Proceedings of the 2004 International Conference on Dependable Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2004, p. 61.

[8] A. Ionescu, "New functionality and ultra low power: key opportunities for post-cmos era," April 2008, pp. 72–73.

[9] K. Tsukagoshi, N. Yoneya, S. Uryu, Y. Aoyagi, A. Kanda, Y. Ootuka, and B. W. Alphenaar, "Carbon nanotube devices for nanoelectronics," *Physica B: Condensed Matter*, vol. 323, no. 1-4, pp. 107 – 114, 2002, proceedings of the Tsukuba Symposium on Carbon Nanotube in Commemoration of the 10th Anniversary of its Discovery.

[10] A. van Roosmalen and G. Zhang, "Reliability challenges in the nanoelectronics era," *Microelectronics and Reliability*, vol. 46, no. 9-11, pp. 1403 – 1414, 2006, proceedings of the 17th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. Wuppertal, Germany 3rd-6th October 2006.

[11] *Predictive Technology Model*, Arizon State University, School of Engineering. [Online]. Available: http://www.eas.asu.edu/ ptm/

[12] B. C. Paul, S. Fujita, M. Okajima, and T. Lee, "Modeling and analysis of circuit performance of ballistic cnfet," in *DAC '06: Proceedings of the 43rd annual conference on Design automation*. New York, NY, USA: ACM, 2006, pp. 717–722.

[13] M. L. Shooman, *Reliability of Computer Systems and Networks: Fault Tolerance,Analysis,and Design*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[14] D. Siewiorek, V. Kini, H. Mashburn, S. McConnel, and M. Tsao, "A case study of c.mmp, cm*, and c.vmp: Part i.experiences with fault tolerance in multiprocessor systems," *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1178–1199, Oct. 1978.

[15] D. Avresky, S. Geoghegan, and Y. Varoglu, "Evaluation of software-implemented fault-tolerance (sift) approach in gracefully degradable multi-computer systems," *Reliability, IEEE Transactions on*, vol. 55, no. 3, pp. 451–457, Sept. 2006.

[16] J. Hopkins, A.L., I. Smith, T.B., and J. Lala, "Ftmp.a highly reliable fault-tolerant multiprocess for aircraft," *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1221–1239, Oct. 1978.

[17] T. M. Austin, "Diva: A dynamic approach to microprocessor verification," *Journal of Instruction-Level Parallelism*, vol. 2, p. 2000, 2000.

[18] D. Jewett, "Integrity s2: a fault-tolerant unix platform," *Fault-Tolerant Computing, 1991. FTCS-21. Digest of Papers., Twenty-First International Symposium*, pp. 512–519, Jun 1991.

[19] P. N. Sanda, J. W. Kellington, P. Kudva, R. Kalla, R. B. McBeth, J. Ackaret, R. Lockwood, J. Schumann, and C. R. Jones, "Soft-error resilience of the ibm power6 processor," *IBM Journal of Research and Development*, vol. 52, no. 3, pp. 275–284, 2008.

[20] J. C. Smolens, B. T. Gold, B. Falsafi, and J. C. Hoe, "Reunion: Complexity-effective multicore redundancy," *Microarchitecture, IEEE/ACM International Symposium on*, vol. 0, pp. 223–234, 2006.

[21] C. LaFrieda, E. Ipek, J. F. Martinez, and R. Manohar, "Utilizing dynamically coupled cores to form a resilient chip multiprocessor," in *DSN '07: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Washington, DC,

USA: IEEE Computer Society, 2007, pp. 317–326.

[22] A. Golander, S. Weiss, and R. Ronen, "Ddmr: Dynamic and scalable dual modular redundancy with short validation intervals," *Computer Architecture Letters*, vol. 7, no. 2, pp. 65–68, Feb. 2008.

[23] D. Sánchez, J. L. Aragón, and J. M. García, "Evaluating dynamic core coupling in a scalable tiled-cmp architecture," in *Proc. of the 7th Int. Workshop on Duplicating, Deconstructing, and Debunking (WDDD), in conjunction with ISCA'08*, Jun 2008.

[24] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir, "The nyu ultracomputer—designing a mimd, shared-memory parallel machine," in *ISCA '98: 25 years of the international symposia on Computer architecture (selected papers)*. New York, NY, USA: ACM, 1998, pp. 239–254.

[25] A. Shye, T. Moseley, V. J. Reddi, J. Blomstedt, and D. A. Connors, "Using process-level redundancy to exploit multiple cores for transient fault tolerance," in *DSN '07: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 297–306.

[26] "The M5 Simulator System, University of Michigan, http://www.m5sim.org/wiki/index.php/mainpage."