Vicis: A Reliable Network for Unreliable Silicon

David Fick, Andrew DeOrio, Jin Hu, Valeria Bertacco, David Blaauw, and Dennis Sylvester
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109
{dfick, awdeorio, jinhu, valeria, blaauw, dmcs}@umich.edu

ABSTRACT

Process scaling has given designers billions of transistors to work with. As feature sizes near the atomic scale, extensive variation and wearout inevitably make margining uneconomical or impossible. The ElastIC project seeks to address this by creating a large-scale chip-multiprocessor that can self-diagnose, adapt, and heal. Creating large, flexible designs in this environment naturally lends itself to the repetitive nature of network-on-chip (NoC), but the loss of a single link or router will result in complete network failure. In this work we present Vicis, an ElastIC-style NoC that can tolerate the loss of many network components due to wearout induced hard faults. Vicis uses the inherent redundancy in the network and its routers in order to maintain correct operation while incurring a much lower area overhead than previously proposed N-modular redundancy (NMR) based solutions. Each router has a built-in-self-test (BIST) that diagnoses the locations of hard fault and runs a number of algorithms to best use ECC, port swapping, and a crossbar bypass bus to mitigate them. The routers work together to run distributed algorithms to solve network-wide problems as well, protecting the networking against critical failures in individual routers. In this work we show that with stuck-at fault rates as high as 1 in 2000 gates, Vicis will continue to operate with approximately half of its routers still functional and communicating.

Categories and Subject Descriptors

B.4.5 [**Hardware**]: Input/Output and Data Communications—*Reliability, Testing, and Fault-Tolerance*

General Terms

Algorithms, Design, Reliability

Keywords

Network-on-Chip, Fault Tolerance, Hard Faults, N-Modular Redundancy, Reconfiguration, Torus, Built-in-Self-Test

1. INTRODUCTION

Silicon processes have continued to improve in transistor density and speed due to aggressive technology scaling. However, each generation increases variability and susceptibility to wearout as silicon features approach the atomic scale. It has been predicted that future designs will consist of hundreds of billions of transistors, where upwards of 10% of them will be defective due to wear-out and variation [5]. At that point we must learn to design reliable systems from unreliable components, managing both design complexity and process uncertainty [6].

The ElastIC project approaches this challenge by having a large network of processing cores that are individually monitored for various wearout mechanisms and operating conditions [20]. Each processing core can be tested, repaired, powered down, or quarantined individually. With proper lifetime management, the performance of some cores can even be boosted for short periods of time when needed [14]. The interconnect architecture becomes a single point of failure as it connects all other components of the system together. A faulty processing element may be shut down entirely, but the interconnect architecture must be able to tolerate partial failure and operate with partial functionality. Network-on-chip provides opportunities to address this issue, as redundant paths exist from point to point, potentially allowing for reconfiguration around failed components.

Network-on-chip is a distributed, router-based interconnect architecture that manages traffic between IPs (intellectual properties, *e.g.*, processing cores, caches, *etc.*) [4]. Traffic sent through the network is organized into packets, of arbitrary length, which are broken down into smaller, link-sized chunks called flits (FLow unITS). This style of interconnect scales bandwidth with network size, making it suitable for systems with many more communicators than what could be supported by bus-based architectures. Recent commercial designs include the Tilera TILE64 [2], Intel Polaris [21], and Ambric AM2045 [1, 12]. NoCs do not inherently support fault tolerance - the first link or router failure in the network will cause the entire network to deadlock.

The reliability of NoC designs is threatened by transistor wearout in aggressively scaled technology nodes. Wear-out mechanisms such as oxide breakdown and electromigration become more prominent in these nodes as oxides and wires become thinned to the physical limits. These breakdown mechanisms occur over time, so traditional post burn-in testing will not capture them. Additionally, so many faults are expected to occur that *in situ* fault management will be economically beneficial. In order to address these faults, we propose a framework with the following steps: 1) error detection, 2) error diagnosis, 3) system reconfiguration, and 4) system recovery. Error detection mechanisms notices new faults by invariance checks such as cyclic redundancy code checks on packets. Error diagnosis determines the location of the new fault via a built-in-self-test. System reconfiguration disables faulty components and configures functional components to work around the faulty ones. Lastly, system recovery restores the system to a previously known good state or checkpoint.

Network-on-chip provides inherent structural redundancy and interesting opportunities for fault diagnosis and reconfiguration. Each router is comprised of input ports, output ports, decoders and FIFOs, which are duplicated for each channel. By leveraging the redundancy that is naturally present in routers and the network, Vicis is able to provide robustness at a low area cost of 42% while exhibiting greater fault tolerance than the previously proposed N-modular redundancy based solutions. We experimentally show that Vicis is able to sustain as many as one stuck-at fault per every 2000 gates, and still maintain half of its routers.

DAC'09, July 26–31, 2009, San Francisco, California, USA.

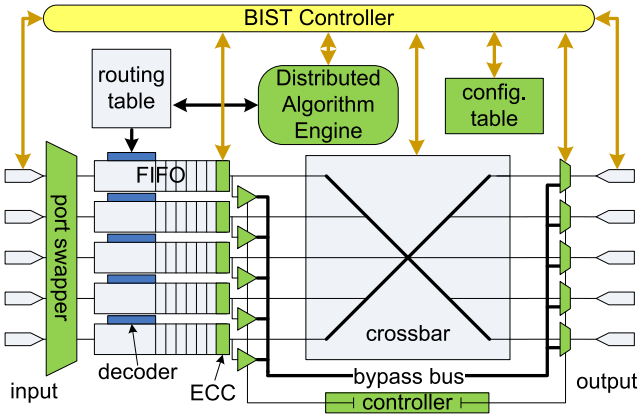

Figure 1: Architecture of the Vicis router. Enhancements include a port swapper, ECC, crossbar bypass bus, BIST, and distributed algorithm engine.

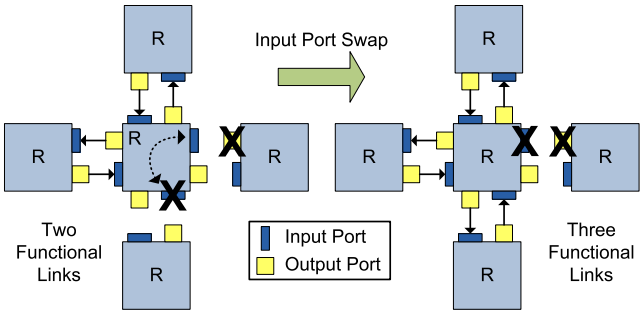

Figure 2: An example of port swapping. On the left side a failed input port and a failed output port are part of two different links. By swapping the failed input port to the link connected to the failed output port, Vicis increases the number of functional links from two to three.

2. RELATED WORK

Reliable network design was first introduced by Dally *et al.* with a reliable packet-switching network, focusing on protecting against faulty board traces due to physical and thermal stress [8]. This network used link-level monitoring and retransmission to allow for the loss of a single link or router anywhere in the network, without interruption of service.

Constantinides *et al.* demonstrated the BulletProof router, which efficiently used NMR techniques for router level reliability [7]. However, NMR approaches are expensive, as they require at least N times the silicon area to implement. Additionally, network level reliability needs to be considered since some logic is impossible or expensive to duplicate (*e.g.*, clock tree) or spares may run out, resulting in the loss of a router, and subsequently the network.

Pan *et al.* explored another strategy that exploits redundant components [16]. Instead of using NMR, inputs are sent through two copies of the same component and the outputs are then compared. If they do not match, then an error is detected and can be repaired. In addition, they included on-line testing and a built-in-self-test (BIST) for correction.

A recent architecture was proposed by Park *et al.* that focused on protecting the intra-router connections against soft errors [17]. They explored a hop-by-hop, flit retransmission scheme in case of a soft error with a three-cycle latency overhead. Though each component has localized protection, there is no guarantee that the network will be operational if certain hard faults occur. Researchers have also explored efficient bus encoding schemes to protect against transient errors [3, 24].

Routing algorithms for fault-tolerant networks have been extensively explored for network level reconfiguration [9–11, 13, 18, 19, 22, 23]. These algorithms direct traffic around failed network components in a way that avoids network deadlock. Although Vicis must also accomplish this, it additionally diagnoses the location of hard faults and reconfigures around them using other methods as well. Vicis adopts an implementation of the routing algorithm described in [9] for part of its network level reconfiguration.

3. THE VICIS NETWORK

Vicis is able to maintain and prolong its useful lifetime by accurately locating faults, reconfiguring around them at the router level, and then reconfiguring at the network level. The following section is split into three parts, giving a top down view of Vicis. The first section discusses how Vicis reconfigures around failed routers and links at the network level. The second section looks at how Vicis uses ECC and a crossbar bypass bus to tolerate faults internally. The last section discusses the built-in-self-test and how it is able to accurately diagnose faults within the router with 100% coverage.

3.1 Network Level Reconfiguration

Network level reconfiguration addresses network visible problems by running distributed algorithms to optimize connectivity and redirect traffic. The input port swapping algorithm runs first in order to increase the number of available links in the network. The network routing algorithm then rewrites the static routing table (shown in Figure 1) to detour traffic around failed network links and routers. These algorithms use fault information provided by the router level reconfiguration discussed in Section 3.2.

3.1.1 Input Port Swapping

The routing algorithm needs fully functional bidirectional links in order to safely route through the network. Each link is comprised of two input ports and two output ports, all four of which must be functional to establish a link. If one of these four ports fail, then there are still three functional ports that might be used in other places (three spares).

As shown in Figure 1, the input ports are comprised of a FIFO and a decode unit which are identical for each lane of traffic. Vicis includes a port swapper to the input of the FIFOs in order to change which physical links are connected to each input port. A port swapper is not included for the output ports due to their relatively small area. The area of the input ports, however, is majority of the total area of the router and will therefore attract the most faults, so adding an input port swapper gives Vicis great ability to move faults around the router. Since the outputs of the input ports are connected to a crossbar, they do not need a second swapper.

Figure 2, shows an example of the input port swapper in use. On the left side there is a failed input port and a failed output port. Since these two failed ports are on different links, both links fail and become unusable by the network routing algorithm. One of the failed ports is a input port of the router in the center, so the physical channel that is connected to can be changed. The port swapping algorithm reconfigures the port swapper so that the failed input port is instead connected to the neighboring failed output port, as shown on the right side. By doing this, Vicis takes advantage of the inherent redundancy in the router, and increase the number of functional links from two to three.

The port swapper does not need to be fully connected, that is, not every input port needs to be connected to every physical link. In this implementation of Vicis, the link to the local network adapter

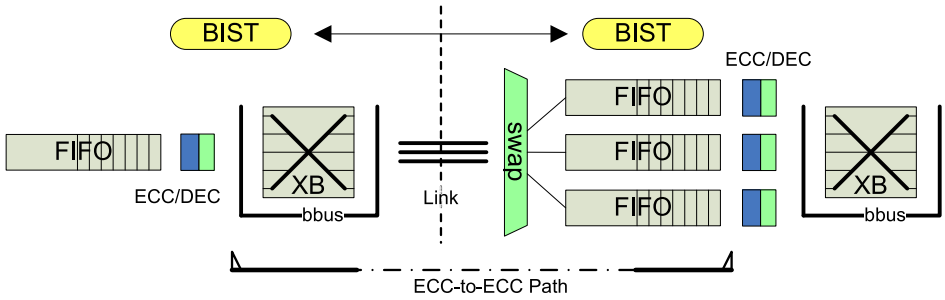

Figure 3: Convergence path of ECC-mitigated faults. Faults in the datapath can be corrected using ECC as long as no more than one is encountered between corrections. The bypass bus and port swapper provide alternate paths between routers to reduce the number of faults in each ECC-to-ECC path. There are six available paths in this example.

is able to connect to three different input ports, and the other links are able to connect to two input ports. The port swapping algorithm is a greedy algorithm that takes into account the ECC related information described in Section 3.2.2. It also gives priority to the local network adapter link, making sure that it is always connected when possible.

After the port swapping algorithm finishes running, each router knows which of its links are functional, and provides that information to the routing algorithm, which completes the network reconfiguration.

3.1.2 Network Rerouting

For a non-fault tolerant network, or a network that supports only router level reliability, the first link or router failure will cause the entire network to fail. Traffic that gets directed to that link or router gets lost, and eventually the network will deadlock when traffic flow tokens fail to be back propagated. In order to detour traffic around faulty links and routers, Vicis uses a routing algorithm to rewrite the routing tables shown in Figure 1. We selected the algorithm described by Fick *et al.* in [9], since it does not need virtual channels, it supports link-level faults, and it has a low overhead implementation. Since it is a distributed algorithm, the loss of any number of links or routers will not prevent it from running, and the remaining routers will have correctly written routing tables.

3.2 Router Level Reconfiguration

Router level reconfiguration is used to contain faults within the router so that they are not visible at network level in the form of failed links and routers. The Vicis architecture, shown in Figure 1, includes a crossbar bypass bus to protect against crossbar failures, and single error correct, single error detect error correction codes (SEC-SED ECC) to protect datapath elements. Information about the location of faults is provided by the hard fault diagnosis stage, described in Section 3.3.

3.2.1 Crossbar Bypass Bus

The bypass bus, parallel to the crossbar, allows flits to bypass a faulty crossbar (see Figure 1). The crossbar controller is configured by the BIST to direct traffic to either the crossbar or to the bypass bus on a packet by packet basis. If multiple flits needs to use the bypass bus (simultaneously), then one flit is chosen to proceed first while the others wait until a later cycle. This spare path allows Vicis to maintain operation when multiple faults appear in the crossbar, trading off performance to maintain correct operation.

3.2.2 Error Correcting Codes (ECC)

ECC-protected units allow each datapath component to tolerate a small number of faults while maintaining correct functionality at the cost of minimal overhead. Any fault that manifests along an ECC-guarded datapath can be corrected when the flit passes though an ECC unit. In order to take full advantage of ECC, the BIST tracks the location of every datapath fault and ensures that every ECC-to-ECC path has at most one hard fault.

In Figure 3 we show the ECC-to-ECC path in Vicis. After encoding, a flit travels through the crossbar and bypass bus, the link between the two routers, the input port swapper, and finally the FIFO. At each mentioned unit along the path, the faults (if any) are diagnosed and cataloged by the two BISTs. If two faults accumulate in the same datapath, the bypass bus and port swapper provide alternative configurations to either avoid one of the faults or move one of the faults to a different datapath, respectively.

For example, one fault could be in the crossbar, another in a link, and a third in the default FIFO for the datapath. The ECC implementation in Vicis can only correct one of these faults, so the crossbar bypass bus and input port swapper need to mitigate two of them. The bypass bus will be used to avoid the crossbar fault, potentially resulting in a loss of performance. The input port swapper will be used to swap in a fault-free input port to the datapath, moving the single-fault input port to another physical link that does not have any faults. By doing this, full functionality is maintained, even with three faults originally in the same datapath.

The FIFOs were designed to take advantage of ECC as well. The FIFO is implemented as a circular queue, which results in a given flit only ever entering a single line of the FIFO. In this way, a FIFO could have upwards of one fault per entry while remaining operational, as it effectively adds only a single fault to the ECC-to-ECC path.

3.3 Hard Fault Diagnosis

The techniques in Vicis rely on knowing precisely where faults exist. A port cannot be swapped if it is not known what ports are faulty and which are functional. The use of ECC requires that Vicis knows precisely how many faults are in each part of the datapath. In this section we discuss the strategies that Vicis uses to locate faults with low cost, yet obtain 100% coverage.

3.3.1 Built-in-Self-Test (BIST)

The built-in-self-test (BIST) unit controls all of the router diagnosis and reconfiguration. It is power gated during normal operation in order to protect it from wearout. Because of this, the BIST controls other units through a configuration table, which stays powered on at all times. The configuration table is tested as well, as part of testing the units that it controls. Non-wear-out faults in the BIST need to be found after fabrication, and the router disabled by the manufacturer.

Figure 4 shows the built-in-self-test procedure that occurs when the BIST is activated. One of network adapters will detect a network failure by using a cyclic redundancy check or another invari-

```
 1: (*)  Propagate Error Status (Error Unit)
 2: ( )  Power up BIST
 3: (*)  Synchronize Network
 4: (D)  Test Router Configuration Table
 5: (P)  Test Error Propagation Unit
 6: (P)  Test Crossbar Controller
 7: (P)  Test Routing Table
 8: (P)  Test Decode/ECC Units
 9: (P)  Test Output Ports
10: (P)  Test FIFO Control Logic
11: (D)  Test FIFO Datapath
12: (D*) Test Links
13: (D*) Test Swapper
14: (D)  Test Crossbar
15: (D)  Test Bypass Bus
16: (*)  Communicate ECC-to-ECC Path Information
17: (*)  Run Port Swapping Algorithm
18: (*)  Run Network Rerouting Algorithm
19: ( )  Power Down BIST
20: ( )  Resume Operation
```

Figure 4: Built In Self Test procedure. Distributed algorithms are marked (*), datapath testing is marked (D), and pattern based testing is marked (P).

ance check. Once a network adapter detects an error, it will send an error status bit to its connected router, at which point the routine in Figure 4 begins.

The network first broadcasts an error status bit via a low-overhead error unit. The error unit also powers up the BIST for that router. Once the BIST has been powered up, it performs a distributed synchronization algorithm so that every BIST in the network runs the rest of the routines in lock-step. Once synchronization completes, each unit is diagnosed for faults. The diagnosis step does not use information from previous runs of the routine, or from the detection mechanism that started the routine, so all faults will be diagnosed regardless of whether they caused the error or not. Once all units have been tested, distributed configuration algorithms run, and finally the BIST powers down and normal operation resumes.

3.3.2 *Functional Unit Wrappers*

Each functional unit has a test wrapper so that the built-in-self-test can control it during fault diagnosis. The wrapper consists of multiplexers for each of the functional unit inputs, which switch between the normal unit inputs, and a testing bus coming from the BIST. A signature bus takes a copy of each of the outputs back to the BIST for analysis. The wrappers are interlocked to guarantee fault-coverage is not lost between each of the units. Although a hard fault on a wrapper gate may be caught trivially, a hard fault may occur on the gate driving the multiplexer select bit, forcing it to always be in test mode. By reading the inputs after the multiplexer to the next unit, this case will also be caught. Additionally, synthesis might insert buffers between where the signature bus connects and the other units connecting to the output - those buffers would not be covered without interlocking wrappers. Cooperative testing between routers is needed to test links, the input port swapper, and their associated wrappers.

3.3.3 *Datapath Testing*

Lines marked with (D) in Figure 4 are part of a datapath test. Units in the datapath need an exact count of ECC errors for each unit so that the maximum number of errors is not exceeded for any ECC-to-ECC path. The tests each send all 1's or all 0's through the datapath, looking for bit-flips faults. A specially designed bit-flip count unit determines if the datapath has zero, one, or more bit flips so that multiplexers are not needed to inspect each bit individually.

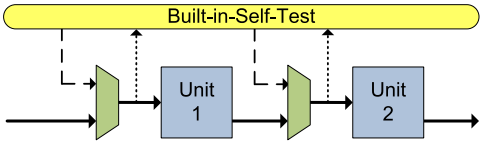

Figure 5: Functional Unit Wrappers. Vicis uses interlocking functional unit wrappers to provide the BIST with access to each unit for hard-fault diagnosis. Incoming patterns from the BIST are shown with long dash arrows, outgoing signatures are shown with short dash arrows, and normal operation paths are shown with solid arrows.

Units like the FIFO reuse the same test for each instance, which helps maintain a low overhead.

3.3.4 *Pattern Based Testing*

Lines marked with (P) in Figure 4 are part of a pattern based test. These units are implemented with synthesized logic, so a hand-written test would not have full coverage. To test these units, Vicis uses a linear feedback shift register (LFSR) to generate a number of unique patterns, and a multiple input signature register (MISR) to generate a signature. Each unit tested with the pattern based test receives the same sequence of patterns from the LFSR, but each has its own signature. Identical units, such as all of the decoders, have the same signature. A signature mismatch will flag the corresponding unit as unusable, and the unit is then disabled by the BIST. Implementation of the pattern based test is lightweight due to the simplicity of the LFSR and MISR.

Pattern based testing dominates the total runtime of the BIST, taking approximately 200,000 cycles when 10,000 patterns are used per test. The time between faults should be measured in hours, however, so this is a relatively short amount of time.

4. EXPERIMENTAL RESULTS

In this section, we discuss Vicis' effectiveness in improving network reliability. First we discuss the experimental setup used to evaluate Vicis. Next, we look at network reliability and compare Vicis to a TMR based implementation. Then we look at router reliability and its *Silicon Protection Factor*. Finally, we show results for network performance degradation.

4.1 Experimental Setup

We implemented Vicis in a 3x3 torus topology, as well as a baseline router design. Each design routes flits with a 32-bit data portion, and have 32-flit input port buffers, which is consistent with the Intel Polaris router [21]. The baseline router has the same functionality as Vicis, except for the reliability features. Both Vicis and the baseline NoC were written in Verilog and were synthesized using Synopsis Design Compiler in a 42nm technology. Compared against the baseline NoC, Vicis has an area overhead of 42%, which is much lower than the 100+% needed for NMR.

To test the reliability features of Vicis, we randomly injected stuck-at faults onto gate outputs in the synthesized netlist. Selection of the gates was weighted based the area of the gate, which is consistent with the breakdown patterns found experimentally in Keane *et al.* [15].

Test packets were generated at each network-adapter. It was verified that each packet made it to the correct destination with the correct data. For each error combination, we ran 10,000 packets of a random uniform traffic distribution, each packet having between 2 and 11 flits.

Network throughput was measured using a custom made, cycle accurate C++ simulator. Profiles of the network were gen-

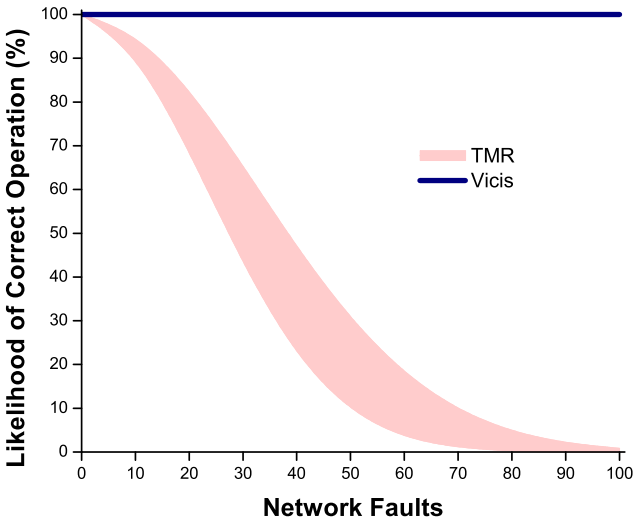

Figure 6: Reliability Comparison with TMR. TMR has probabilistic reliability while maintaining performance, where Vicis has probabilistic performance while maintaining reliability (see Figure 8.)

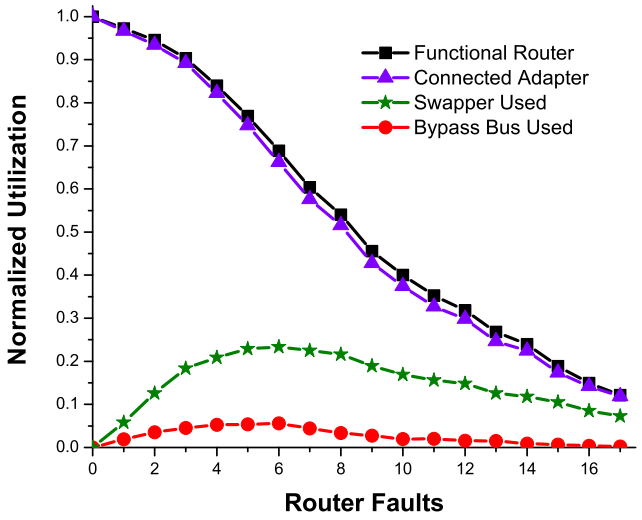

Figure 7: Router reliability with increasing faults. Reliability of an individual router as it receives stuck-at faults inside of a faulty network. Utilization of individual features is also shown. *Connected Adapter* shows how many functional routers also have a functional link to its adapter.

erated with the Verilog simulations described above, which were then given to the C++ simulator to model longer simulation traces. Throughput was measured using random uniform traffic for this test as well.

4.2 Network Reliability

To test the reliability of the simple network, we randomly injected faults into the system and monitored the packet traffic. Although the full system is prone to wear-out, we primarily focused on injecting faults into the router - a total of 20,413 gates. Since we only consider wear-out induced faults, we power-gated the BIST, thus making it immune to fault injection.

To rigorously test the system, we considered eleven different cases with varying simultaneous faults: 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100, 100 being one fault for every 2000 gates. For each number of simultaneous faults, we considered 1200 different (random) error combinations.

In this series of tests there was a single failure that manifested at 30-40 faults, but disappeared at 50 faults. The disappearance of the failure is likely due to new faults being injected into the same unit as the fault(s) that were previously overlooked by the BIST. We expect the failure to be an implementation error, as opposed to a problem with the described techniques. Either way, the reliability of the network is dominated by the routing algorithm for larger networks, as described in [9].

We compare the reliability of our design to a TMR-based implementation. Note that TMR only provides probabilistic reliability - since the voter takes the most common signal of the three replicated units, two well-placed faults could cause the system to fail. In the worst case, a single fault could cause system failure if it occurred in a clock tree or another non-replicable cell. Unlike BulletProof [7] and other prior work that relies on maintaining total functionality, Vicis is able to tolerate multiple, simultaneous faults, including ones that render entire routers useless. Thus, Vicis is able to maintain 100% reliability regardless of the number or occurrence of faults, although performance becomes degraded.

In Figure 6 we show the reliability of TMR, which is based on analysis from [7]. Where TMR provides 100% performance and degrading reliability for 200+% area overhead, Vicis provides 100% reliability and degrading performance for 42% overhead. Comparing TMR reliability degradation to Vicis performance degradation, TMR degrades much more quickly - at 100 network faults, TMR's reliability approaches zero whereas Vicis continues to operate with half of its routers still functional and communicating.

4.3 Router Reliability

In Figure 7, we present the different internal router statistics based on the tests described above. To fairly evaluate and collect results for each router component, we injected faults on standard-cell outputs in a network level netlist in order to consider the states of the surrounding routers. A router is considered operational if it has at least two functional bidirectional ports connected to either other routers or network adapters.

4.3.1 Input Port Swapper

We found that the input port swapper was very successful at keeping the cores (IPs) connected to the network. As shown in Figure 7, only eight percent of the available routers not have a functional network adapter. The swapper had a high utilization, being used nearly 24% of the time for routers with seven faults.

4.3.2 Crossbar Bypass Bus

At seven faults, the crossbar bypass bus was used less than six percent of the time. This is due to two reasons: 1) the crossbar is relatively small - less than five percent of the total area of the router, and 2) the crossbar is protected by both the input port swapper and ECC as well. We expect that an improved implementation of the bypass bus will have an greater effect, and are pursuing that in future work.

4.3.3 Silicon Protection Factor (SPF)

Constantinides *et al.* introduce the concept of *Silicon Protection Factor* for router level reliability, defined as the number of faults a router can tolerate before becoming inoperable, normalized by the area overhead of the technique [7]. The normalization step is to take into account the increase of gates since having more gates means that the design will also experience more faults.

From Figure 7 we can interpolate to get the median value, which is 9.3 faults before the system fails. Normalized to our area overhead of 42%, this gives Vicis an SPF of 6.55. In comparison, the best SPF provided by the Bulletproof router is 11.11, which incurs a 242% area overhead. The lowest area overhead Bulletproof configuration is 52% overhead, but provides an SPF of only 2.07.

Actual network reliability cannot be compared since a router failure in Bulletproof causes full network failure, whereas a router failure in Vicis only renders that single router inoperable. Additionally, Bulletproof does not give a breakdown of what faults were critical (first fault causes a failure) versus cumulative (multiple faults interact to cause a failure). The number of critical failure points in the Bulletproof router would place a limit on overall network reliability, where that is not the case in Vicis.

4.4 Network Performance

In Figure 8, we demonstrate how network level performance gracefully degrades as the number of faults in the network increases. The black line with squares shows the number of available IPs connected through the network to at least one other IP. At 90 faults, which is more than 10 faults per router, or one fault per 2000 gates, over 50% of the original IPs are still available. Since faults are injected at the gate level, there would not likely be any functional IPs to connect, as they would be experiencing the same fault levels.

In Figure 8, we also show the normalized network throughput across different number of simultaneous hard faults (data points as triangles). For the first 30-40 faults, the network experiences a loss in normalized throughput. This is due to link failures (2+ faults within links), forcing packets to take longer paths and avoid network obstructions. After 40 faults, however, the network loses enough routers to effectively create a smaller network that intrinsically supports higher normalized throughput. The light shading behind the data line shows 5th-95th percentiles of normalized throughput - the line itself is the median.

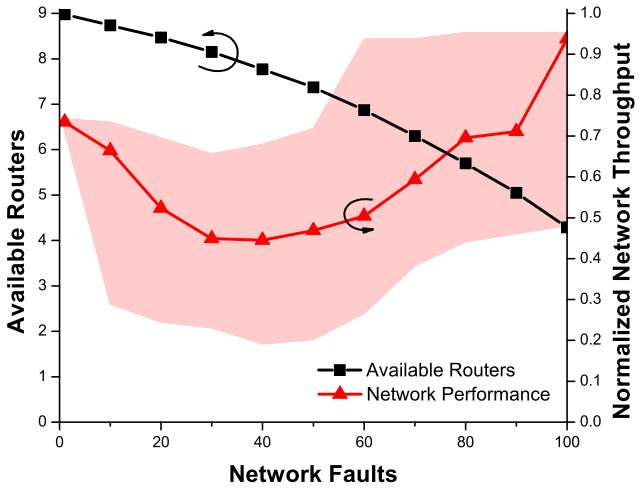

Figure 8: Network performance with increasing faults. Normalized network throughput is shown as the number of faults in the network increases. Throughput is normalized to the bandwidth of the remaining network adapter links. The shaded region show the 5th-95th percentiles, while the line is the median.

5. CONCLUSIONS

In this work we presented Vicis, a network-on-chip that is highly resilient to hard-faults. By using the inherent redundancy in the network and its routers, Vicis can maintain higher reliability than NMR based solutions while incurring only a 42% overhead. Each router uses a built-in-self-test (BIST) to diagnose the locations of hard faults and runs a number of algorithms to best use ECC, port swapping, and a crossbar bypass bus to mitigate them. The routers work together to run distributed algorithms to solve network-wide problems as well, protecting the networking against critical failures in individual routers. Ultimately, we show that with stuck-at fault rates as high as 1 in 2000 gates, Vicis will continue to operate with approximately half of its routers still functional and communicating. Additionally, we provide results detailing the utilization of some of the architectural features, as well as a reliability comparison with triple modular redundancy and prior work.

6. ACKNOWLEDGMENT

This research was funded in part by the Gigascale Systems Research Center and the United States National Science Foundation. We would like to thank Synopsys for their generous support of this project, and the reviewers for their detailed feedback.

7. REFERENCES

[1] Massively Parallel Processing Arrays Techhnology Overview. *Ambric Technology Overview*, 2008.

[2] S. Bell et al. TILE64 processor: A 64-core SoC with mesh interconnect. *Proc. ISSCC*, 2008.

[3] D. Bertozzi, L. Benini, and G. De Micheli. Low power error resilient encoding for on-chip data buses. *Proc. DATE*, 2002.

[4] T. Bjerregaard and S. Mahadevan. A survey of research and practices of network-on-chip. *ACM Computer Survey*, 2006.

[5] S. Borkar. Microarchitecture and design challenges for gigascale integration. *Proc. Micro, keynote address*, 2004.

[6] S. Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *Proc. Micro*, 2005.

[7] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky. BulletProof: a defect-tolerant CMP switch architecture. *Proc. HPCA*, 2006.

[8] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos. The reliable router: A reliable and high-performance communication substrate for parallel computers. *Proc. PCRCW*, 1994.

[9] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw. A highly resilient routing algorithm for fault-tolerant NoCs. *Proc. DATE*, 2009.

[10] C. J. Glass and L. M. Ni. Fault-tolerant wormhole routing in meshes without virtual channels. *IEEE Trans. on Parallel and Distributed Systems*, 1996.

[11] M. E. Gomez et al. An efficient fault-tolerant routing methodology for meshes and tori. *IEEE Computer Architecture Letters*, 2004.

[12] T. R. Halfhill. Ambric's New Parallel Processor: Globally Asynchronous Architecture Eases Parallel Programming. *Microprocessor Report*, 2006.

[13] C.-T. Ho and L. Stockmeyer. A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers. *IEEE Trans. on Computers*, 2004.

[14] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge. Reliability modeling and management in dynamic microprocessor-based systems. *Proc. DAC*, 2006.

[15] J. Keane, S. Venkatraman, P. Butzen, and C. H. Kim. An array-based test circuit for fully automated gate dielectric breakdown characterization. *Proc. CICC*, 2008.

[16] S.-J. Pan and K.-T. Cheng. A framework for system reliability analysis considering both system error tolerance and component test quality. *Proc. DATE*, 2007.

[17] D. Park, C. Nicopoulos, and J. K. N. V. C. Das. Exploring fault-tolerant network-on-chip architectures. *Proc. DSN*, 2006.

[18] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide. Immunet: A cheap and robust fault-tolerant packet routing mechanism. *ACM SIGARCH Computer Architecture News*, 2004.

[19] S. Rodrigo, J. Flich, J. Duato, and M. Hummel. Efficient unicast and multicast support for CMPs. *Proc. Micro*, 2008.

[20] D. Sylvester, D. Blaauw, and E. Karl. ElastIC: An Adaptive Self-Healing Architecture for Unpredictable Silicon. *IEEE Design & Test*, 2006.

[21] S. R. Vangal et al. An 80-tile sub-100w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 2008.

[22] J. Wu. A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model. *IEEE Trans. on Computers*, 2003.

[23] J. Zhou and F. C. M. Lau. Multi-phase minimal fault-tolerant wormhole routing in meshes. *Parallel Computing*, 2004.

[24] H. Zimmer and A. Jantsch. A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip. *Proc. CODES+ISSS*, 2003.