A Distributed Concurrent On-Line Test Scheduling Protocol for Many-Core NoC-Based Systems

Jason D. Lee and Rabi N. Mahapatra
Texas A&M University
{jdlee,rabi}@cs.tamu.edu

Praveen S. Bhojwani
Sun Microsystems, Inc.
praveen.bhojwani@sun.com

***Abstract*—Concurrent on-line testing (COLT) of many-core systems-on-chip (SoC) has been recently proposed by researchers in response to the growing threat of electronic wear-out to system operational lifetimes and to the increasing reliability and availability demands of safety-critical applications. Previous research in concurrent on-line testing has focused on centralized approaches to manage core testing while the system is available to execute normal user applications. However, as technology scaling allows dozens and hundreds of processing cores to be placed on a single chip, these centralized approaches are not scalable solutions. In this paper, a distributed concurrent on-line test scheduling protocol is proposed and evaluated against previously developed solutions. Our experiments show that a distributed COLT scheduler can test a moderately-sized SoC with a speedup of 3.85 over centralized approaches while consuming 84% less energy, and performance benefits improve as the number of cores per chip increases. This research also presents a core test ordering algorithm – Code-Division Core Test Scheduling – that provides an additional 40% reduction in system test latency compared to other schedulers.**

I. INTRODUCTION

Rapid technology scaling has forced systems designers, reliability engineers and application programmers to rethink the fundamental design practices that have dominated computer system design for more than the past two decades. Multi-core systems-on-chip (SoC), with a handful of complex processing cores and integrated peripheral components, are predicted to be replaced by many-core SoC that contain hundreds or thousands of light-weight processing cores, memory and I/O subsystems. These many-core SoC will use packet switched networks-on-chip (NoC) for inter-core communication, as opposed to the current standard of on-chip busses, [1,7]. Notable examples of this architecture include the 64-core TILE64 from TILERA [24] and the 80-core Intel Terascale SoC [11].

As technology scaling has provided new opportunities for massively parallel and distributed computation to be performed on a single chip, new reliability challenges have also emerged. In addition to the well-understood circuit failures due to manufacturing imperfections, SoC components are also more susceptible to *electronic wear-out* – permanent failures that emerge during use – as feature sizes scale below 65nm [2,6,7].

In actuality, electronic wear-out is a combination of several physical degradation mechanisms, including electro-migration (EM), hot carrier injection (HCI) and negative bias temperature instability (NBTI), that are intensified by smaller feature sizes, higher current and power densities, and higher operating temperatures [2].

Because the most significant electronic wear-out mechanisms manifest as an increasingly severe delay fault at the circuit level, many researchers have proposed the use of SCAN-based delay testing for detecting this type of error [4,14,15]. Built-in self-test (BIST) architectures using pseudo-randomly generated test vectors are effective in discovering wear-out in memory elements. Unfortunately, BIST techniques cannot be applied to control portions of logic components, such as intellectual property (IP) cores, due to the low fault coverage levels provided by logic BIST [15].

Therefore, researchers have noted the need to apply production-quality, high-coverage SCAN delay test vectors to the cores within SoC once the system has been deployed in the field [14]. These tests can be applied as an isolated process – the entire system is taken off-line periodically or triggered by an event so that all cores can be tested for wear-out [16]. Alternatively, core testing can be performed *concurrently* with normally executing applications [4,5,15,16].

Concurrent on-line test (COLT) exploits the massive structural redundancy of multi- and many-core architectures by shutting down some subset of cores within the SoC for testing while the remaining cores run user applications as normal. This allows the system to achieve its reliability requirements and maintain an extremely high level of availability. The amount of overhead, in terms of power consumption, resource and network congestion, and chip area costs, incurred by testing system components during normal operation is known as *application intrusion*. Application intrusion must be minimized in order for COLT strategies to operate feasibly.

In the COLT strategies currently proposed in research, the delivery of test vectors from the test source (on-chip test storage or off-chip memory) to the individual cores

978-1-4244-5028-2/09/$25.00 ©2009 IEEE

within the SoC is the most significant contributor to testing costs in terms of test latency and energy consumption [4,15]. This problem will only become more critical as the number of cores per SoC increases over time. As the distance between test source and sink increases for deeply embedded cores, application intrusion in the form of test latency, energy and NoC congestion also increases.

To address this problem, a distributed test vector storage technique has been proposed which uses the topological properties of regular networks such as the 2D-torus to guarantee that each core can access its SCAN test vectors within a uniform bounded distance while minimizing storage overhead [14].

Using this distributed test vector storage technique, this research proposes a complete and fully distributed concurrent on-line test protocol.

The main contributions of this paper are summarized as follows:

- A *fully distributed* COLT scheduling protocol is introduced for the first time that addresses the scalable test vector delivery issues present in current COLT research.
- The proposed protocol and supporting architecture greatly reduces the application intrusion of running tests concurrently with user applications: system test delivery latency is reduced by 74%, and system test delivery energy consumption is reduced by 84% compared to centralized testing schemes for a moderately-sized 25-tile SoC. Performance improvements increase with SoC size.
- This research also presents a theoretically optimal core test ordering, called *Code-Division Core Test Scheduling*, when using distributed test vector storage. By simply ordering which subset of cores are tested simultaneously, system test delivery latency and energy consumption can be reduced by 50% when compared to other core test orderings.

The rest of this paper is organized as follows: Section II describes the current state of COLT research and explains the role of distributed test vector storage when implementing COLT. Section III introduces the proposed distributed COLT scheduling protocol and its supporting system architecture. Section IV details the experimental simulation platform, system assumptions and experiment set, and Section V presents the results of the evaluation in terms of system performance and overhead. Finally, concluding remarks are presented in Section VI.

II. PRELIMINARIES

A. Concurrent On-Line Testing (COLT)

In [10], the researchers analyzed the costs and benefits of reusing the NoC as a test access mechanism (TAM) for each core in the network. This was proposed as a response to the increasing difficulty of accessing deeply embedded cores within a SoC. It was shown that a NoC is indeed a feasible TAM due to minimal overheads; however, it was noted that test delivery times (and power consumption) depend on the distance between the test source and the core to be tested. Additionally, this variability in test delivery time may not be fully understood, since relatively small 5x7 and 4x8 2D-torus topologies were studied in that research.

Using the NoC as a test delivery infrastructure, researchers have proposed reusing infrastructure IP (I-IP) designed originally for manufacturing testing as a tool for on-line testing of the system [25].

Taking this concept a step further, [4,5] constructed a Test Infrastructure IP (TI-IP) capable of managing test scheduling, delivery, and intrusion for concurrent on-line test (COLT) of the SoC. COLT allows cores in the SoC to be tested in the presence of normally executing applications to maximize system availability. The original COLT scheme proposes that the test vectors are stored within the TI-IP. Due to the real-time constraints of these applications, COLT is extremely sensitive to application intrusion.

Similarly, Yi, Makar and Mitra have proposed CASP: Concurrent Autonomous chip Self-test using stored test Patterns [15]. Much like COLT, an on-chip test controller is proposed which manages test scheduling, test application and response comparison for processing cores of the OpenSPARC T1 chip multi-processor [22]. Their technique differs slightly from COLT in that the test vectors are stored off-chip in a nearby flash or HDD storage system. Consistent with previous research, the time required to test a core was almost completely dominated by test vector delivery latency. When using off-chip flash memory, 83% of the total test time was consumed by transferring test vectors to the core under test. Delivering test vectors via off-chip HDD memory accounted for over 99% of the total test time [15]. However, it was successfully demonstrated that CASP is a feasible solution to ensure lifetime reliability for certain applications.

Operating system and hardware virtualization support is required for any COLT scheme to operate invisibly to the system user. The OS/virtualization layer must track which cores are available to the operating system for normal applications and to initiate task migration when necessary during test. This topic is addressed in [16] and is outside the scope of this paper.

In a centralized test vector storage scheme, energy consumption, network load, and test delivery latency become dependent on the distance of the core under test to the TI-IP. Aside from the obvious scalability issues, this increases the complexity (and potentially decreases the accuracy) of calculating intrusion costs of on-line testing by the TI-IP. Ultimately, this may negatively affect the availability of the system. This motivates the need to bound test delivery distances for all cores in the SoC.

B. *Distributed Test Vector Storage*

Lee and Mahapatra proposed a solution for bounding test vector delivery distances by applying the concept of *t-interleaving* [12] to optimally distribute test vectors across many-core SoC [14]. t-interleaving is a resource placement technique which guarantees any single resource is at least t hops apart within some regular topology.

This solution requires that the many-core SoC uses a NoC with a regular topology, such as the torus, ring or mesh network topology. For this research, we will only consider NoC constructed as 2D-tori; however, this concept may be generalized to other regular network topologies.

The basic premise of t-interleaving originated with Lee metric error correcting codes [8,9]. The Lee metric is the standard distance metric for the torus topology. The Lee distance between two nodes in a torus network is simply the hop count between those nodes.

Figure 1 illustrates a simple example of t-interleaving on a small 2D-torus. Specifically, a 3-interleaving is constructed so that all numbers are within one hop of every tile. In this example, the complete test vector set is divided into 5 segments corresponding to the 5 numbers distributed through the SoC. By inspection, it can be seen that any tile in the SoC can access all 5 segments of the test vector set within one hop.

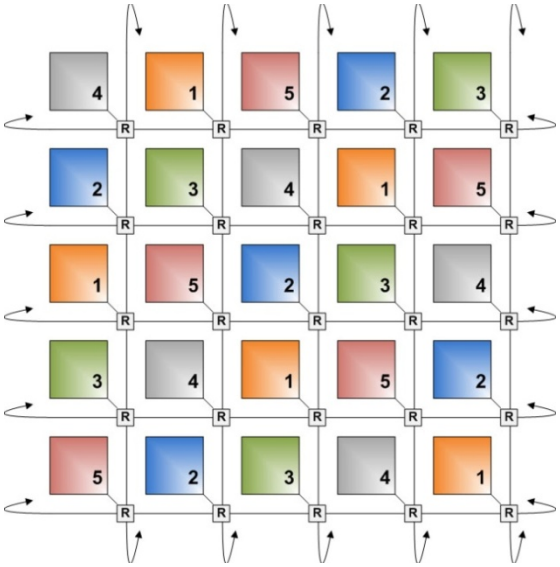

Figure 1. 3-interleaving on a 5x5 2D-Torus

A performance-storage trade-off is made in order to guarantee that all tiles have access to the complete test vector set within one hop – the complete test vector set must be replicated five times across the SoC. However, this has been proven to be an optimal trade-off for tori when a perfect t-interleaving is achieved. A more complete mathematical discussion of t-interleaving and its applications, such as distributed file storage, is provided in [12].

By using t-interleaving on tori for on-chip test vector storage, it is proven that any core in the network can access the complete set of test vectors within a bounded radius. Bounding this test delivery radius for each core not only guarantees scalability, but on-line test intrusion can be much more easily estimated. The level of intrusion for each core no longer depends on proximity to the test source, simplifying scheduling decisions for on-line testing.

Although this research focuses solely on the 2D-torus network topology, t-interleaving can be applied to any regular topology with a defined metric. Applicable alternative regular topologies that have been shown to be suitable for on-chip networks include the mesh, ring and circulant graph topologies [18].

III. DISTRIBUTED COLT SCHEDULING

A. *System Architecture*

Because the proposed test scheduling protocol is fully distributed, inserting dedicated TI-IP tiles into the SoC is unnecessary. Instead, each tile contains a small test controller within the Core-Network Interface (CNI) of each tile.

Figure 2 illustrates the distributed COLT architecture for each NoC tile within the SoC. NoC tiles communicate with each other via on-chip routers (R) and CNI. In the example system, each NoC tile contains a simple CPU core, such as the OpenSPARC T1 core, surrounded by DFT SCAN chains.

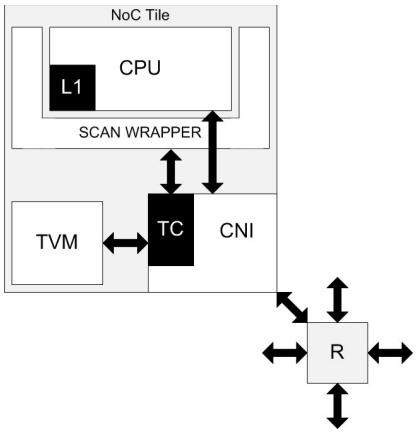

Figure 2. Distributed COLT Architecture

Each CNI contains a Test Controller (TC) which implements the proposed distributed COLT protocol. The TC contains a nominal amount of buffers to temporarily store incoming test vectors and responses from neighboring cores and communicates directly with the DFT structures surrounding the core under test. Each tile also contains a dedicated Test Vector Memory (TVM) which stores some portion of the complete test vector set. The TC is responsible for sending the local test vector subset to neighboring cores upon request.

The TC is not directly responsible for managing power consumption of the core or the system during test mode. Researchers have proposed that NoC-based SoC can manage power consumption by implementing a power management (PM) unit within each CNI [3, 13, 20]. This PM can manage power consumption of its associated NoC tile by throttling network traffic and the TC.

Because multiple NoC tiles may initiate tests simultaneously, a token-based protocol is used to manage the distributed TC. This protocol is detailed in the next section.

B. *Distributed COLT Scheduling Protocol*

This research proposes a simple distributed COLT scheduling protocol that allows each tile to manage its own self-test. The TC of each CNI implements the test scheduling protocol as illustrated in Figure 3.

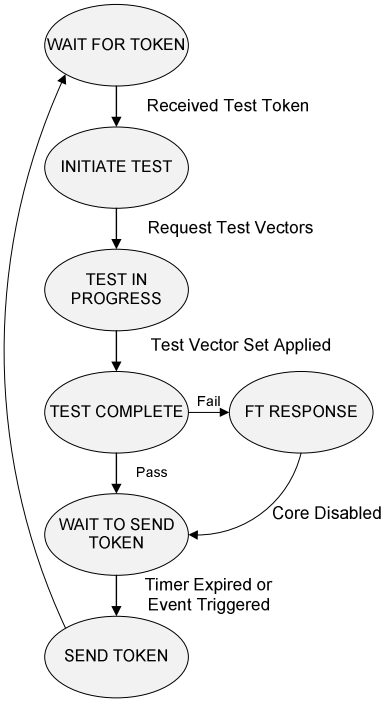

Figure 3. Distributed COLT Protocol

Depending on the operating and SCAN test characteristics of the CPU cores within the SoC, multiple cores may be tested simultaneously without violating the system power budget and application requirements. The number of cores that may be tested simultaneously is determined by the system architects and is system and application dependent. If *n* cores can be tested simultaneously, then the system is initiated with *n* tokens distributed across the NoC tiles. Tiles without tokens remain in the *WAIT FOR TOKEN* state before initiating COLT locally.

Once a tile possesses a token, the TC enters the *INITIATE TEST* state and may begin a local core test. Before actual testing can begin, it may be necessary to unload the core of its current task and migrate this task to an alternate core. The TC will request the other pieces of the test vector set from neighboring tiles and will begin applying SCAN tests to the test wrapper during the *TEST IN PROGRESS* state. All test vectors are applied and responses are compared until test completion. Once the TC is in the *TEST COMPLETE* state, it will determine whether the testing has passed or failed based on a standard response comparison. If the test has passed, the system can be configured so that the core may resume its task as described in [14]. If the testing has failed, the TC enters the Fault-Tolerance Response (*FT RESPONSE*) state and the appropriate action is taken. For instance, an appropriate response would be to disable the faulty core and replace it with a cold spare if one is available. Once this action is complete, or if the test passes, the TC enters the *WAIT TO SEND TOKEN* state. Concurrent on-line testing may be initiated periodically or due to specific system events. Once the appropriate condition is met, the TC will pass the testing token to the next tile.

C. *Code-Division Core Test Scheduling*

Given that it will be possible for some many-core SoC, containing hundreds or thousands of processing cores, to test multiple cores simultaneously, optimizing which cores can be tested simultaneously for latency, network congestion, power consumption and thermal effects is a critical consideration for COLT.

As described in Section 2.2, a many-core SoC which uses a NoC with a regular topology, such as the 2D-torus, can be represented as a set of Lee-metric error correcting codes. Each code contains a set of NoC tiles located by each codeword. For example, Figure 1 shows a 25-core SoC that contains 5 different error correcting codes. Each code is labeled by a distinct number – the tiles that are labeled 1 all belong to the same error correcting code. By the definition of error correcting codes, each tile belonging to the same code must be *t* hops apart.

Because each tile belonging to the same code – tiles that will be tested simultaneously – are t hops apart, and each core under test must access test vectors from at most t – 1 hops away, there exist no resource conflicts due to test. In other words, any tile in the SoC will send its segment of the test vector to only one requesting core under test at any time. This design constraint greatly simplifies the testing architecture; no dual ported TVM are required, and there is no need to broadcast or multicast test vector segments across the network. Therefore, Code Division Core Test Scheduling dictates that only cores belonging to the same error correcting code may be tested simultaneously.

However, there are fundamental limits to this test ordering scheme. If more than n cores in an n x n 2D-torus SoC can be tested simultaneously, then resource conflicts

will occur. Also, if only one core can be tested at a time, the order in which cores are tested will have no effect on performance, neglecting user application effects.

IV. EXPERIMENTAL SETUP

To measure the performance of using this distributed test vector technique, we simulate a NoC-based SoC using the NoCSim on-chip network simulator [19]. NoCSim is a SystemC cycle-accurate simulator which models IP cores, on-chip routers, CNI, and network links for any network topology to form a complete system. Table 1 describes the baseline simulation configuration and parameters used during our experiments.

Table 1. Simulation Parameters

SoC Topology	25, 64, 100, 169 cores with 2D-torus
Test Vector Set Size	256KB
Network Configuration	64-bit flits, 8 flits per packet, 8 VCs per link, 8 flit buffer depth, 1 GHz, wormhole routing, credit-based flow control
Process Technology	180nm

The simulated on-chip network utilizes 64-bit links, and each IP core can transmit information in 64-byte packets. The test vector sets used in all experiments are 256KB in size; therefore, 4096 packets of information are transmitted from test source to test sink. The measurement of energy consumption is based on energy models developed in [13] which considers a system operating at 1GHz using 180nm technology.

Four systems are constructed for our experiments: a 25-node SoC in a 5x5 torus topology, a 64-node SoC in a 8x8 torus topology, a 100-node SoC in a 10x10 torus topology, and a 169-node SoC in a 13x13 torus topology. For each system, we simulate the behavior of delivering test vectors to each node in the system using a standard centralized approach described in [4] and the proposed distributed protocol based on t-interleaving. We allow *n* cores to be tested simultaneously in each *n* x *n* configuration. For example, five cores are tested simultaneously in the 5x5 torus system.

For the 5x5 and 10x10 torus systems, 3-interleaving is used to distribute the test vectors across the network; the test vector set is split into 5 pieces, and each node can access all test vectors within 1 hop. For the 8x8 torus, 4-interleaving is used, therefore the test vector set is split into 8 pieces, and each node can access all test vectors within 2 hops. Finally, the 13x13 torus system uses 5-interleaving to distribute the test vectors. In this scheme, the test vector set is split into 13 pieces, and each node can access all test vectors within 2 hops.

V. RESULTS

The goal of any concurrent on-line testing scheme is to minimize application intrusion. Therefore, the system impacts of COLT are measured in terms of system test latency – the amount of time required to test all cores of the SoC, and system test energy consumption, which is critical in safety-critical embedded applications. Additionally, the effect of the order of core testing is measured to determine the effect of network and resource contention on system test latency. Finally, the Test Controller (TC) portion of the CNI, which implements the proposed distributed COLT scheduling protocol is synthesized to determine area and power overhead.

A. System Test Latency

System test latency, or the amount of time required to test each core of the SoC using COLT, is a critical parameter in determining the feasibility of including a COLT scheme in a SoC.

Figure 2 illustrates the performance differences between using a standard centralized COLT protocol and the proposed approach. In each n x n SoC, *n* tiles are replaced with TI-IP as described in [4] to ensure that n cores are tested simultaneously throughout the simulation for fairness.

For the 5x5 2D-torus system, distributed COLT can test each core of the system in 120µs, while the centralized COLT scheme requires 460µs to test the entire system. This equates to a 74% reduction in system test time when using distributed COLT. System test latency improvements increase with SoC size as Figure 4 shows; at the 169-core 13x13 SoC, distributed COLT can test the entire system in 260µs, while centralized COLT requires 1700µs. This equates to a 84% reduction in system test time.

In safety-critical applications with real-time constraints, shortening the system test time as much as possible is critical to ensuring that tests and applications can meet their deadlines.

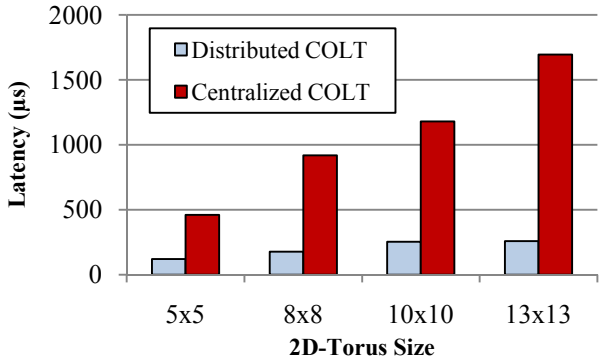

Figure 4. Scalability of System Test Latency

B. System Test Energy Consumption

Increasing test vector delivery distances affect energy consumption due to test, just as latency is affected in Section 5.1. Energy consumption is a critical factor to many safety-critical applications, since a large portion of safety-critical applications require mobile systems where a finite amount of energy is available. Network energy parameters used in these experiments are based on the results obtained in [13].

Figure 5 illustrates the effect of SoC size on energy consumption during COLT. Note that energy consumption is represented on a log scale due to the exponential growth of energy consumption as SoC size increases.

For the 5x5 2D-torus system, using the proposed distributed COLT protocol results in a 83% reduction in energy consumption for an entire system test. As with latency, energy consumption improves as the SoC size increases. For the 13x13 SoC, energy consumption is reduced by 93%.

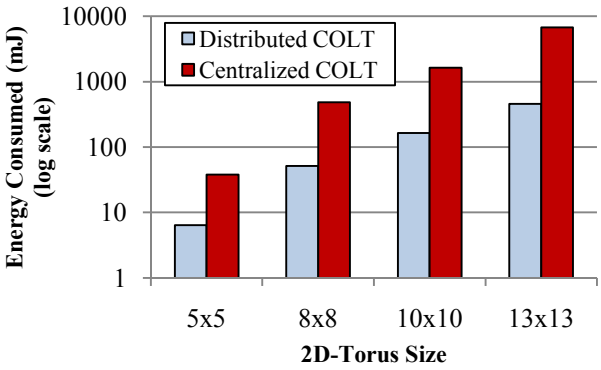

Figure 5. Scalability of Energy Consumption during COLT

C. Effect of Core Test Scheduling

Distributed test vector storage allows a core under test to access the complete test vector set stored across its neighbors within a certain radius. Specifically for torus-based on-chip networks, a communication pattern in the form of a Lee-metric sphere of radius *r* centered at the core under test is created. In other words, the core under test will communicate with neighboring cores at most *r* hops away. Therefore, no two cores with overlapping Lee-metric spheres of radius *r* should be tested simultaneously. This experiment shows the effect of testing cores with overlapping test vector communication compared to using the proposed Code-Division Core Test Scheduling; the time to test all cores of the SoC using a standard centralized approach is also included for the purposes of comparison.

Figure 6 illustrates the effect of resource conflicts due to improper core test ordering. For this experiment, four scheduling algorithms are used to determine which cores of a 5x5, 8x8 and 10x10 2D-torus based SoC are tested simultaneously: Random, Code, Linear, and Centralized. In the random scheduler, cores are chosen at random to be tested such that each core is tested once until the entire SoC is tested; this algorithm is used as a baseline for comparison. Simulations using the random scheduler were run until a latency result converged. The Code scheduler is the proposed algorithm described in Section 3.3. The Linear scheduler simply tests all n cores of a n x n SoC line-by-line. The Centralized scheduler is the algorithm assumed by previous research and does not use distributed test vector storage.

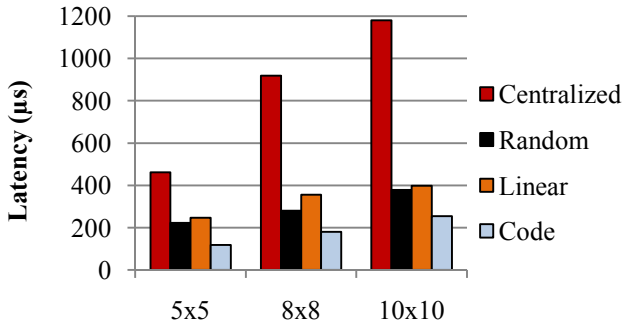

Figure 6. Effect of Core Test Scheduling

As Figure 6 shows, the proposed Code-Division based algorithm reduces system test latency by approximately 40% compared to both the random and linear schedulers. This demonstrates that resource conflict, in terms of network congestion and shared test vector memory usage, is a significant contributor to system test latency.

D. Distributed Test Controller Overhead

To determine the area and power costs of including the distributed COLT-based Test Controller (TC) in each CNI of the system, and HDL model of the TC was developed. This model was functionally verified using Verilog testbenches.

The HDL model was synthesized using Synopsys Design Compiler [23] and Oklahoma State University's 45nm FreePDK library [21]. The gate count of the TC was estimated to be 5.8K, while power consumption was estimated to be 6.2mW. This overhead includes the realization of the scheduling state machine, test vector and response buffers, core state buffers to save the microprocessor state, and test controller mechanisms such as the SCAN test interface.

E. Test Vector Memory Overhead

As stated in previous sections, more on-chip storage will be required to implement the distributed test vector storage scheme used with the proposed distributed COLT protocol when compared to centralized schemes [14]. For most systems, n copies of the test vector set will need to be

stored on-chip for a SoC of size n x n. However, as on-chip cache sizes increase exponentially as feature size shrinks, test vector storage will consume a relatively small percentage of on-chip memory.

For example, a 25-core SoC using the OpenSPARC T1 core would require approximately 2MB of test vector storage using distributed COLT assuming standard test compression is used and based on findings from [15]. Due to these increased storage requirements, this research envisions that distributed COLT will be most applicable for safety-critical applications.

VI. CONCLUSION

Concurrent on-line test of many-core SoC is only feasible if application intrusion is sufficiently reduced, allowing user applications to function correctly while detecting the formation of hard errors due to electronic wear-out quickly. Previous research has shown that the most significant contributor to application intrusion is the delivery of test vectors to each core within the system.

This research has proposed and evaluated a fully distributed concurrent on-line test scheduling protocol using distributed test vector storage for many-core SoC. At the cost of increased storage requirements, our test protocol can reduce the latency of system test by 74% and energy costs by 84% for a 5x5 2D-torus SoC.

VII. REFERENCES

[1] L. Benini and G. De Micheli, "Networks on Chip: A New SoC Paradigm," *IEEE Trans. Computer*, Vol. 35, No. 1, 2002.

[2] J. Bernstein, et al, "Electronic Circuit Reliability Modeling," *Microelectronics Reliability*, Vol. 46, 2006, pp. 1957-1979.

[3] P. Bhojwani, and R. Mahapatra, "Core network interface architecture and latency constrained on-chip communication," *Proc. IEEE Intl. Symp. on Quality Electronic Devices,* 2006, pp. 363-368.

[4] P. Bhojwani, and R. Mahapatra, "An Infrastructure IP for On-Line Testing of Network-on-Chip Based SoCs," *Proc. IEEE Intl. Symp. on Quality Electronic Devices,* 2007, pp. 867-872.

[5] P. Bhojwani, and R. Mahapatra, "A Robust Protocol for Concurrent On-Line Test (COLT) of NoC-based Systems-on-a-Chip," *Proc. Design Automation (DAC),* 2007, pp. 670-675.

[6] S. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degredation," *IEEE Micro*, Nov-Dec 2005.

[7] S. Borkar, "Thousand Core Chips – A Technology Perspective," *Proc. Design Automation* (DAC), 2007, pp. 746-749.

[8] B. Broeg, et al, "Lee Distance and Topological Properties of k-ary n-cubes," *IEEE Trans. Computer*, Vol. 44, No. 8, 1995.

[9] B. Broeg, et al. "Lee Distance, Gray Codes, and the Torus," *Telecommunication Systems*, Vol. 10, Nos. 1-2, 1998, pp. 21-32.

[10] E. Cota, L. Carro, F. Wagner, and M. Lubaszewski, "Power-Aware NoC Reuse of the Testing of Core-Based Systems," *Proc. Intl. Test Conference (ITC)*, 2003, pp. 612-621.

[11] Intel Tera-scale Computing Research Program, http://www.intel.com/research/platform/terascale

[12] A. Jiang, M. Cook, and J. Bruck, "Optimal Interleaving on Tori," *SIAM Journal of Discrete Math*, Vol. 20, No. 4, 2006, pp. 841-879.

[13] Y. Jin, E. Kim, and K. Yum, "Peak Power Control for a QoS Capable On-chip network," *Proc. Intl. Conference on Parallel Processing* (ICPP), 2005, pp. 585-592.

[14] J. Lee, and R. Mahapatra, "In-Field NoC-Based SoC Testing with Distributed Test Vector Storage," *Proc. Intl. Conference on Computer Design* (ICCD), 2008, pp. 206-211.

[15] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns," *Proc. Design, Automation, and Test in Europe* (DATE), 2008, pp. 885-890.

[16] Y. Li, and S. Mitra, "VAST: Virtualization-Assisted Concurrent Autonomous Self-Test," *Proc. International Test Conference* (ITC), 2008, pp. 1-10.

[17] A. Manzone, et al, "Integrating BIST Techniques for On-line SoC Testing," Proc. *IEEE Intl. On-Line Testing Symposium (IOLTS)*, 2005, pp. 235-240.

[18] C. Martinez, et al, "Dense Gaussian Networks: Suitable Topologies for On-chip Multiprocessors," *Intl. Journal of*

[19] NoCSim. http://codesign.cs.tamu.edu/nocsim

[20] L. Shang, L.S. Peh and N.K Jha, "PowerHerd: A Distributed Scheme for Dynamically Satisfying Peak-Power Constraints in Interconnection Networks," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 1, 2006.

[21] J. Stine, et al, "FreePDK: An Open-Source Variation-Aware Design Kit.", Proc. International Conference on Microelectronic Systems Education, 2007.

[22] Sun Microsystems, OpenSPARC T1 Processor, http://www.opensparc.com

[23] Synopsys Design Compiler, http://www.synopsys.com

[24] Tilera Corporation. http://www.tilera.com

[25] Y. Zorian, "Guest Editor's Introduction: What is Infrastructure IP?," *IEEE Design & Test of Computers*, Vol. 19, 2002, pp. 3-5.