Robust Concurrent Online Testing of Network-on-Chip-Based SoCs
Praveen S. Bhojwani, Student Member, IEEE, and Rabi N. Mahapatra

Abstract—Lifetime concerns for complex systems-on-a-chip (SoC) designs due to decreasing levels in reliability motivate the development of solutions to ensure reliable operation. A precursor to any proposed recovery scheme would require the identification of failures in the system. Non-concurrent in-field testing is an impractical solution due to prohibitive costs in terms of test power and test time. This novel research proposes the use of concurrent online testing (COLT) to circumvent these issues. A test infrastructure-intellectual property (TI-IP) is deployed within network-on-chip (NoC)-based SoC designs to provide online test support while managing intrusion of test into executing applications within the system. This research describes the architecture and operation of a TI-IP capable of COLT. To address scalability of this solution, we show how these would operate when more than one is deployed in an SoC. In the absence of benchmarks for the analysis of COLT, two baseline and eight TI-IP configuration variations within SoC test configurations were developed using application and test benchmarks from the research domain. The power profiles from the NoCSim simulation environment are reported here demonstrating how different configurations of TI-IPs would operate. A robust TI-IP protocol is also specified and possible hazards and their mitigations are identified.

Index Terms—Concurrent testing, infrastructure intellectual property (IP), network-on-chip (NoC), online testing.

I. INTRODUCTION

WITH A TREND towards increasing levels of integration in system-on-a-chip (SoC) designs, designers are faced with reliability concerns, specifically reliability degradation over lifetime of the SoC. Failure mechanisms, such as electro-migration, stress migration, time-dependant dielectric breakdown, and thermal cycling [1] affect lifetimes and pose a significant challenge to accurate estimation of mean-time-to-failure (MTTF) at design time and, consequently, failure detection at runtime. These issues trigger the development of mitigation techniques for recovery from failures. A precursor to such techniques is identification of failure prior to recovery initiation. Typical non-concurrent in-field test techniques provide for unacceptable solutions in light of prohibitive test costs: test power and test time. Post SoC deployment into the target system, these techniques simply turn “off” executing applications and test for possible failures. But with test costs being prohibitively large, it may not be possible to achieve the desired effect economically. The ideal solution entails an online integrity check of the system. This research proposes the integration of a test infrastructure intellectual property (TI-IP) into the design to provide concurrent online test (COLT)—test of SoC without turning off applications. The introduction of infrastructure IP (I-IP) blocks into SoC designs is an emerging trend that provides supplemental on-chip support infrastructure to address enhanced detection, test, diagnosis, and yield optimization solutions [2].

Providing concurrent online SoC test support cannot be achieved by simply deploying a test controller into the SoC. Test intrusion into concurrently operating applications has to be managed. This can be quantified as power consumed by the IP core in test mode and delay introduced into the finish time of executing applications. Classical research has shown test mode power consumption to be $1.5 \times$ normal mode power consumption [3], which will continue to scale with design complexity. Judicious management of available system power budget is essential. If exact information on executing applications is available, tests can be scheduled under power and performance constraints using existing optimization algorithms. But unavailability of this information requires the development of an online test strategy capable of collecting information across IP blocks in the SoC and managing concurrent online SoC testing. This research is a first step into demonstrating how this can be achieved. A robust protocol to manage concurrent online test is also essential towards ensuring correct operation of the TI-IP in a multi-core SoC environment.

A critical component to online SoC testing is the design and use of a test access mechanism (TAM). Networks-on-a-chip (NoC) have quickly emerged as a suitable interconnect infrastructure for complex SoC design with numerous cores communicating among each other and providing multiple functionality. Scalability, predictability, manageable power consumption, etc., make the utilization of NoCs an inviting prospect for SoCs [4]–[6]. The reuse of the NoC as a TAM has been proposed by researchers [7], [8] and is also considered in this research.

This research makes the following contributions:
- identification of challenges to COLT of SoCs.
- introduction of TI-IP to manage COLT of on-chip cores;
- specification of TI-IP architecture and design to manage online test [9];
- specification and validation of a robust TI-IP operation protocol capable of addressing scenarios that may hamper correct on-line test of cores [10];
- evaluation of TI-IP in a NoC-based SoC design using ten experimental configurations prepared using application
and test benchmarks; observations on power profile and application finish time variation are also reported;

- assessment of costs for deployment of TI-IP into SoCs.

Section II describes some related research in the SoC testing domain. Section III discusses some basic NoC concepts. Challenges to COLT are presented in Section IV. TI-IP architecture, operation, and limitations are examined in Section V. Section VI provides the robust TI-IP protocol specification and examines the possible hazards and mitigation techniques. The approach taken towards validating the proposed TI-IP design and observed results are discussed in Section VII.

II. ONLINE TESTING OF SOCS

A. Offline Testing

Traditional SoC test research focuses on test scheduling, test cost management, and TAM design. As summarized in [11], TAMs broadly fit into four categories: multiplexer-based, serial scan-based, bus-based, and network-based. The first two categories suffer from scalability issues and research literature has numerous works highlighting the possibility of reusing the NoC as a TAM [7], [8], [11]–[13]. Optimizing for test power consumption and test time via offline schemes have also been studied [7], [13]. These techniques are typically inapplicable for online testing due to the associated runtime costs.

B. Online Testing

Online testing of SoCs can be categorized into: 1) non-concurrent and 2) concurrent. Non-concurrent testing (also referred to as in-field testing) is typically executed by turning off executing applications in the device-under-test (DUT). Recent works have demonstrated in-field test in multi-core SoCs [14], [15].

Concurrent online testing needs to manage test of DUT in presence of executing applications. Software-based online health monitoring and diagnosis of microprocessors have been implemented in current systems. But these require the involvement of the operating system and are limited to the microprocessors. Sun Microsystems’ Online CPU Diagnostic Monitor is one such program for Ultra-SPARC III and Ultra-SPARC IV families of processors [16].

The proposed research eliminates the no-application restriction and demonstrates how concurrent online testing can be accomplished in heterogeneous designs.

III. NOC-BASED SOCS

A. NoC Architecture

Typical NoC-based SoC designs are constituted by regular layouts of network tiles connected to an interconnection network that routes communication with the help of on-chip routers. Network tiles are formed by one or more IP cores interfaced with the help of core-network interfaces (CNI). The common topologies considered by most researchers are the mesh and 2-D-folded torus [4], [17]–[19]. Fig. 1 illustrates this typical architecture.

B. Core-Network Interface

In the absence of a central network management component, the CNI has been introduced into NoC designs to provide services beyond basic packetization and depacketization [20]. End-to-end communication reliability, dynamic power management, communication scheduling, NoC reconfigurability, and NoC test support are the additional functionality provided by CNI. IP interface arbiter interfaces with IP cores and replays the corresponding interface protocols. Multiple IP cores may be connected to a single interface arbiter. This demonstrates the capability of the CNI to logically interface with multiple IP cores that may be present in a single network tile. Physically, the connection between the multiple IP cores in a network tile will be via a bus local to the network tile. Encoder/decoder provide for reliable end-to-end communication. The link controller interfaces with the NoC and is configured according to NoC specifications. The functionality of other CNI components is self-explanatory. This research utilizes the test controller to manage concurrent online test of IP cores in the NoC. The TI-IP communicates with the CNI to manage concurrent online test.

IV. CHALLENGES TO COLT

Management of COLT presents a challenge to system designers wishing to deploy an online health management solution in the target design. The key challenges to COLT are as follows.

1) Test Triggering: COLT for IP cores in a multi-core design can be triggered: 1) periodically or 2) on an event. Choosing between triggering techniques depends on expected MTTF and system criticality. The type of failure model chosen to model the lifetime of the IP core will impact the type of test triggering mechanism.

2) Test Feasibility: Since test and application have to execute concurrently in the SoC, test intrusion management is essential. This would determine the feasibility of performing online test on the on-chip cores. With complete system operation characteristics available, COLT management is trivial. In practical systems, this information will not be available. This requires COLT to determine feasibility of online test before it can be executed.

3) Test Prioritization: System criticality determines priority assigned to COLT. Critical applications and devices need to operate at high levels of confidence.

4) Restorative nature of test: Since test can execute on IP cores that may be executing applications, the context switch from application mode to test mode and from test back to application mode has to ensure that states are restored.

Section V discusses the TI-IP design and how it manages COLT.
V. TI-IP DESIGN

The objective of TI-IP is the management of online SoC testing in presence of executing applications in the system. The test management has to ensure the following:

- power constrained SoC testing;
- minimized application intrusion, while ensuring reliable operation.

This research addresses the power constrained concurrent testing and application intrusion management issues. For the online test of SoCs, the following components in the NoC are the main participants: TI-IP, CNI, and the test wrappers around IP cores. Analogous to a server-client environment, the CNI (client) acts as an intermediary between the TI-IP (server) and the test wrappers around the IP core under test. The CNI initiates test requests to the TI-IP and interfaces TI-IP with the test wrapper of the IP core providing test vectors when using SCAN testing and the necessary test control signals when using built-in self test (BIST). This research assumes SCAN test of IP cores due to the limited availability of benchmark data. But support for BIST has been incorporated into current designs of the TI-IP.

It is also important to note that one TI-IP in an SoC will control online testing of several IP cores and is capable of doing so concurrently. For this research, we assume different TI-IP configurations, one where we have a single TI-IP managing online test of the complete SoC and the second where we have multiple TI-IPs managing online test of multiple IP cores. Fig. 2 illustrates the envisioned system. The following subsections elaborate on TI-IP architecture and its operation.

A. TI-IP Components

The key TI-IP components (see Fig. 3) are: 1) input/output queues; 2) test memory; and 3) TI-IP engine. Since we envision the use of a complete NoC tile for TI-IP, the input/output queues basically handle incoming and outgoing requests and responses between the TI-IP engine and CNI of the NoC tile within which it resides. This research uses queues instead of single buffers, to allow for non-blocking TI-IP operation to support multiple test requests from CNIs in the NoC. When utilizing SCAN test, the test memory essentially stores test vectors that will be needed to test IP cores in the SoC. At current technology trends, for a NoC tile of size 4 mm × 4 mm, after setting aside silicon area for the TI-IP engine, the TI-IP core can accommodate 512 kB of test memory. With migration of IP test towards BIST, the need for larger on-chip test memory will diminish. The TI-IP engine is heart of the TI-IP and it can be better understood via its operation.

B. TI-IP Operation

Operations to be supported by TI-IP include: 1) servicing test requests and 2) managing concurrent online testing. It collaborates with CNIs in the NoC and other TI-IPs in the design. The CNI 1) initiates test requests; 2) forwards test data to IP cores from TI-IP; 3) test responses from the IP cores to TI-IP; and 4) monitors IP core utilization with the help of an IP core monitoring unit. The function of the IP core monitor is to determine IP core utilization. We do not provide technical details regarding this, since it involves a simple circuit capable of tracking IP core utilization by examining its current drain. For the purpose of our experiments, we treat this as a black box. This collected information will aid the TI-IP in assessing feasibility of concurrent test.

As stated earlier, for concurrent online testing of SoCs, the CNI is responsible for issuing test requests to the TI-IP for the IP cores attached to it. Determining the frequency of test is an interesting and challenging task by itself, but is beyond the scope of this paper. For the purpose of this research, we assume that CNI requests for test periodically. The frequency of test request depends on: 1) technology; 2) utilization; and 3) criticality of the core. Future research will formalize this concept. Test request can also be event-based and is an option being explored in other research.

The online testing of an SoC is performed in a number of stages. A finite-state machine (FSM) summarizing these is shown in Fig. 4, while Fig. 5 elaborates on the expected timeline of execution for a multi-core SoC with online testing enabled. The various states in the FSM are discussed in this section, while details regarding the signal arcs are presented in Section VI-A.

The rest of this section describes the activities in the various stages of TI-IP operation.
By virtue of using a packet-switched interconnection network to provide communication between on-chip cores, network latency can degrade the “freshness” of the system snapshot information collected by the TI-IP. In smaller NoCs, this is not a major concern. But in larger designs, this issue can be resolved by weighting information delivered from farther CNIs. This aspect is beyond the scope of this paper. The test throttling mechanism discussed later in this paper provides sufficient protection against power budget violations.

**TA:** In the TA stage, a routing solution for test vector delivery from test memory to the target IP core is determined and using SS information, the feasibility of test is made. Test feasibility is determined using utilization metrics. The following expression aids in deciding whether a requested test can be accepted within intrusion bounds:

\[
\left( \sum_{i \in \text{IP}} U_i + \sum_{j \in \text{NoC components}} U_j \right) + \left( \sum_{k \in \text{CNI}} U_k + \sum_{l \in \text{path}} U_l \right) \leq 1.0
\]

where the \( U \) terms are the utilization indices of the components in the SoC. These utilization indices are normalized using: 1) the measured IP core utilization; 2) a multiplier; and 3) total power budget. The multiplier normalizes measured core utilization with system power budget and can be determined from the core specifications. By using this utilization index as a prediction of SoC utilization we determine feasibility of test. Once accepted, test is scheduled and queued into the output queue of TI-IP. When test requests are rejected, corresponding messages have to be provided to the requesting CNIs.

**Test Delivery (TD):** In the TD stage, test vectors are delivered over the NoC via the route determined in TA stage. The intended target is the CNI attached to the IP core that requested the test. When SCAN testing multiple cores simultaneously, to prevent starvation test vector delivery to the multiple cores is interleaved. Test vector delivery is paced to prevent buffer overflow on intermediate hops and at the destination test site.

**Test Application (TAP):** Test vectors are applied to the IP core in this stage. CNI receives test vectors from the TI-IP. The test wrapper prepares these for application to the IP cores. COLT does not advocate a particular type of test for the IP cores. This will be IP core specific. The TI-IP can support both SCAN and BIST. Due to large test times, the IP core may not be completely testing in a single attempt. Partial test may be performed based on the availability of idle periods.

**Test Response Collection (TRC):** Once test vector application completes, test response collection and comparison is performed before the TI-IP can respond with a recovery scheme. The comparison can either be performed at the application site, i.e., the CNI attached to the IP core being tested or at the TI-IP. In the former case, the expected response will have to be sent to the CNI, while in the latter, the test responses need to be forwarded to the TI-IP. Tradeoffs involved in selecting between TRC implementation includes buffer storage at the CNI, processing delays at TI-IP due to multiple test response comparisons and test response delivery energy consumption.

---

**TABLE I**

<table>
<thead>
<tr>
<th>SYSTEM SNAPSHOT TABLE (SST) FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender ID</td>
</tr>
<tr>
<td>Measured Utilization</td>
</tr>
<tr>
<td>Timestamp</td>
</tr>
<tr>
<td>Test Request</td>
</tr>
</tbody>
</table>

---

**Fig. 4. TI-IP FSM.**

**Fig. 5. SoC operation timeline with online test integrated.**
TABLE II
TI-IP CONFIGURATION TABLE FIELDS

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile ID</td>
<td>This field indicates the network tile ID.</td>
</tr>
<tr>
<td>Test Type</td>
<td>This field indicates the type of test support by the IP core, i.e., SCAN or BIST.</td>
</tr>
<tr>
<td>Test Volume</td>
<td>This field indicates the test volume associated with this test entry.</td>
</tr>
<tr>
<td>Test Memory Pointer</td>
<td>This field provides a pointer to the test vectors in the test memory.</td>
</tr>
</tbody>
</table>

**TI-IP Response (TPR):** After evaluation of test response completes, the TI-IP has to respond to detected phenomenon. This response could include system reconfiguration, IP core shutdown, communication rerouting, and application termination. This response depends on the type of SoC configuration and is a part of the online health management system. Since this research is focused on COLT, SoC recovery is not discussed here.

**CNI Alarm Handler:** In this stage, when the CNI detects utilization misprediction, it sends a CNI Alarm message to TI-IP. The TI-IP throttles IP core test by modifying pace of test vector delivery, thereby controlling test power consumption.

**TI-IP Configuration:** In the TI-IP configuration stage, the TI-IP configuration table is populated with the test information and is loaded when the SoC is deployed and initialized. The fields of this configuration table are listed in Table II.

**C. TI-IP Limitations**

To ensure minimal intrusion of online test to executing application, the TI-IP uses SS information. The quality of this data and utilization prediction affects intrusion of test on application. Also, prioritizing between test and application execution (and communication), will affect SoC test and application performance.

**D. Multi-TI-IP Enhancement**

To manage test power efficiently in larger SoCs a single TI-IP in the design will not be sufficient. Energy consumed in delivering test vectors from test memory to IP cores is directly proportional to the distance between them and volume of the test data. Fig. 6 demonstrates how this energy consumption grows with test volume and hop distance. This provides adequate motivation for the formulation of a multi-TI-IP technique to test larger SoCs. In such a scenario, multiple TI-IPs provide for energy efficient testing by reducing the average hop distance for test vector delivery. But the introduction of additional TI-IPs makes it difficult to ensure power constrained testing without cooperation between these TI-IPs.

Introducing multiple TI-IPs into an SoC design is not a trivial solution, since they have to cooperate in test decisions. For this research, we configured the TI-IPs to operate in a token ring manner. In this case, only the TI-IPs with the test management tokens are permitted to schedule test of the IP cores. To manage this aspect, the test management token handler state was added into the TI-IP controller FSM in Fig. 4. The multi-TI-IP configuration also introduces an additional task of mapping IP core test responsibility to the individual TI-IPs. For our initial experiments, we have assigned test responsibilities based on proximity. A formal technique considering test parameters and design constraints is currently being assessed.

**E. COLT For NoC Components**

The primary focus of this paper is COLT management for on-chip IP cores. For a complete design, the TI-IP should also be able to manage COLT for NoC components, i.e., CNIs, routers, and links. Test for these is relatively easier and lower in cost (test power and test time), when compared to IP cores.

**F. CNI Modification to Support COLT**

Since the CNI is an active participant in COLT, it is modified to accept (transmit) commands from (to) the TI-IP. This modification is primarily in the CNI controller (CNIC) which is responsible for the setting and interpreting of control signals in flits exchanged between the TI-IP and the CNI.

In the Test Request protocol step, a Test Request command is issued by the CNI to the TI-IP. The CNI enables the TI-IP field in the packet, which ensures that the command flit is forwarded to the TI-IP.

In the System Snapshot Determination protocol step, the CNI responds to System Snapshot Request command with a System Snapshot Response. This response provides utilization information for the IP core attached to the CNI. This response command enables the TI-IP field ensuring that the data is forwarded to the TI-IP.

The CNI is also responsible for issuing a CNI Alarm command, if it tracks utilization of the IP core to have changed when compared to the previous observation period.

Test vectors to be applied to the IP core attached to the CNI are tagged by the Test Vector Data command. This instructs the CNI to forward test data to the Test Controller.

**VI. ROBUST TI-IP OPERATION PROTOCOL**

This section addresses the robustness aspect of COLT communication protocol by specifying the communication protocol.
identifying protocol hazards and describing hazard mitigation techniques.

A. Protocol Specification

Protocol Step 1—Test Request: This step of the protocol involves the TI-IP and CNI of IP core requesting test. On the occurrence of the test request event, the CNI issues a test request (TReq) command to the TI-IP. Before issuing a test response, the TI-IP executes the System Snapshot Collection protocol step and performs the Test Acceptance operation discussed in the previous section. The TA step decision is provided to the CNI of IP requesting test via a test request response (TReqResp) command. Fig. 7(a) illustrates this protocol step.

Protocol Step 2—System Snapshot Collection: This protocol step involves the TI-IP and CNIs of all network tiles in the NoC. The TI-IP issues a system snapshot request (SSReq) command to all the CNIs. The CNIs respond with system snapshot response (SSResp) command to all TI-IPs in the NoC. Fig. 7(b) illustrates this protocol step. This requires the CNIs to be aware of all TI-IPs in the SoC. As mentioned earlier, this system snapshot response provides utilization information for the IP core attached to that CNI.

Protocol Step 3—Test Vector Delivery: This protocol step involves the TI-IP and CNIs of IP cores requesting test. When using SCAN test, TI-IP delivers test vectors to these CNIs in an interleaved manner. The interleaved delivery is needed to prevent test starvation due to large test vector volume. Fig. 7(c) assumes a scenario where three IP cores have successfully requested concurrent online test and are in Test Vector Delivery COLT protocol step.

Protocol Step 4—Test Management Token Transfer: In the multiple TI-IP configuration, to allow for cooperative testing of IP cores, TI-IPs need a Test Management Token. As discussed in Section V-D, the TI-IP with the Test Management Token is the only TI-IP allowed to schedule COLT. The multiple TI-IPs are configured in a token ring configuration and they periodically exchange the Test Management Token. Fig. 7(e) illustrates a three TI-IP configuration where the TI-IPs periodically exchange tokens.

Protocol Step 5: Test Throttle: Since test acceptance and scheduling is predictive in nature and depends on history information in the form of IP core utilization, the introduction of new applications into the system, may render collected information to be invalid. To account for mispredicted system utilization information (also referred to as misbehaving applications), the TI-IP has to be able to throttle concurrent online test. The need for such a mechanism is dominant in scenarios with dynamically changing application characteristics. Periodic and well-behaved applications would not utilize this protocol step. The participants in this protocol step are the TI-IPs, CNIs of “misbehaving” IP cores and CNIs of IP cores-under-test. When misbehavior is detected, the CNI issues a CNI alarm (CNIAlarm) command to the TI-IP. The TI-IP then determines whether any concurrent tests need to be reconfigured. It can control two aspects of concurrent test reconfiguration: 1) alter test vector delivery pace and 2) halt test application on IP...
cores-under-test. Fig. 7(d) illustrates the CNI Alarm issuance and the corresponding test throttle (TThrottle) command delivery to CNIs of IP cores-under-test.

B. Protocol Hazards

The following are hazards presented by the protocol specification to concurrent online testing of the SoC. The impact of each hazard on the communication protocol steps is also described.

Hazard 1: Starvation Due to Application/Test: Since application and test communication flow over a common communication infrastructure, large volumes of application (test) data can starve test (application) data.
Impact of hazard on the communication protocol steps are discussed in the following.
1) Test Request: Starvation due to application can prevent test requests from flowing to the TI-IPs, prevent online test.
2) System Snapshot Collection: Starvation due to application can prevent the CNI from responding to system snapshot request commands.
3) Test Vector Delivery: Starvation due to applications can prevent test vector delivery to wrappers. High test vector volume can also lead to starvation due to test at TI-IP.
4) Test Management Token Transfer: Starvation due to high test vector volume can prevent test management token transfer.
5) Test Throttle: Starvation due to application can prevent the CNI alarm from being delivered to the TI-IP. Starvation due to high test vector volume can also prevent Test Throttle command from being delivered from TI-IP to CNI of cores-under-test.

Hazard 2—Test Input Queue Buffer Overflow: The Test Input Queue buffer overflow hazard can prevent the TI-IP from receiving commands critical to concurrent online testing. This hazard has an impact on all protocol steps of COLT.

Hazard 3: TI-IP Failure: The TI-IP Failure hazard can prevent the concurrent online testing of the SoC. In the single TI-IP configuration, the major impact of the failure is loss of concurrent online test capability. In a multi-TI-IP configuration, the testability of the SoC is reduced, since test of IP cores assigned to the failed TI-IP may not be transferable to other TI-IPs in multi-TI-IP configuration.
Impact on the following communication protocol steps.
1) Test Request: TI-IP failure can prevent test request command handing.
2) System Snapshot Collection: TI-IP failure prevents snapshot collection.
3) Test Vector Delivery: TI-IP failure prevents on the delivery of test vectors to IP cores being tested by the failed TI-IP.
4) Test Management Token Transfer: This hazard can result in the loss of the Test Management Token in the multi-TI-IP configuration.
5) Test Throttle: This hazard can lead to constraint violation due to concurrent online test.

Hazard 4: Test Wrapper Buffer Overflow: The test wrapper buffer overflow hazard will lead to loss of test vector data for test of IP cores. This hazard only has impact on the Test Vector Delivery protocol step.

Hazard 5: CNI Failure: This hazard prevents correct operation of the CNI. CNI failure can lead to the isolation of the network tile from the rest of the NoC.

C. Hazard Mitigation

This subsection discusses the approaches taken to circumvent potential hazards identified in Section VI-B.

Hazard 1 Mitigation—Starvation Due to Application/Test:
1) Test Request: For this protocol step, to mitigate hazard 1, test would need to be assigned priority over application. Temporarily swapping priorities can resolve this hazard.
2) System Snapshot Collection: Hazard mitigation for this protocol step involves the usage of the timeout value. Once timeout occurs, the CNI being starved by application is assumed to be busy and in high utilization mode.
3) Test Vector Delivery: To mitigate this hazard, priorities of application and test would need to be swapped.
4) Test Management Token Transfer: Interleaving outgoing communications from Test Output Queue of TI-IP addresses all starvation concerns.
5) Test Throttle: To mitigate this hazard, priorities of application and test would need to be swapped.

Hazard 2 Mitigation—Test Input Queue Buffer Overflow: The test input queue buffer overflow hazard is mitigated by requesting a retransmission from the source CNI. This technique would require additional buffering at the source CNI.

Hazard 3 Mitigation—TI-IP Failure: Hazard 3 mitigation techniques are only applicable in multi-TI-IP configuration.
1) Test Request: CNIs in the NoC need to communicate with all TI-IPs in the SoC. When a TI-IP failure event is detected, all CNIs need to be informed. When using SCAN test, IP core test responsibilities cannot be handed over to alternative TI-IPs since test vector data is only assigned to TI-IP responsible for the test. Responsibility hand off is easier for BIST of cores.
2) System Snapshot Collection: Mitigating the hazard during system snapshot collection protocol step is not possible. It requires hand off of test responsibilities, if possible.
3) Test Vector Delivery: TI-IP failure during the Test Vector Delivery protocol step will prevent successful test of the IP cores assigned to it.
4) Test Management Token Transfer: Mitigating this hazard is considerably more difficult when more than two TI-IPs are deployed in the multi-TI-IP configuration. A complex election protocol among the TI-IPs is required to recover the lost token.
5) Test Throttle: TI-IP failure can prevent test throttling of tests initiated by the failed TI-IP. Mitigation of this hazard can be achieved by exchanging IP core test status between TI-IPs in the SoC. This can be coupled with Test Management Token Transfer protocol step.

Hazard 4 Mitigation—Test Wrapper Buffer Overflow: Test wrapper buffer overflow is prevented by scheduling test vector delivery to test wrapper of IP core-under-test. Since test vector
delivery at the TI-IP source is interleaved, test vector block sizes sent out in a single iteration have an upper bound stipulated by the test wrapper buffer capacity.

**Hazard 5 Mitigation: CNI Failure:*** CNI failure is a hazard that cannot be mitigated. However, once detected, the event should be registered with TI-IPs to prevent wasted effort due to unnecessary communication between TI-IP and failed CNIs.

### VII. Design Validation

#### A. NoCSim

NoCSim [18], [21] is a SystemC-based [22] model for an NoC simulator and is configured to the feature set shown in Table III. NoCSim includes models for the CNI and on-chip routers. For the task-graph-based execution, NoCSim can also track deadline misses (if deadline is provided) and the average task graph execution time over multiple iterations. NoCSim as an experimental platform can be configured with the desired IP cores in each NoC tile to act as traffic generators, processing elements and functionally testable components.

A SystemC functional model of the proposed TI-IP has been embedded into a network tile in NoCSim and is used to manage test of IP cores embedded in the design. It does so through the embedded CNI model and generic test wrapper around the IP core modules. Minor modifications to the packet headers were also required to allow for special handling of flits in the CNIs.

#### B. ITC’02 SoC Test Benchmark

In the absence of an actual SoC design with corresponding applications and test configurations, the ITC’02 SoC Test Benchmarks [23] and Embedded Systems Synthesis Benchmark Suite (ES3) [24] are used to provide the test configuration and application configuration information, respectively. The ITC’02 benchmarks provide sufficient test data for modular designs, to allow researchers to solves problems in the domain of TAM design, test scheduling, and wrapper design.

The benchmark provides typical SoC test parameters—test volume and test execution time. Test power consumption for most of these is unavailable. Since research has shown that power consumption in test mode to be about 1.5x normal mode power consumption [3], we utilize this knowledge to estimate test power consumption for the selected IP cores in our test configurations.

#### C. Embedded Systems Synthesis Benchmark Suite

The ES3 is used to provide application related information for our test configurations. Relevant ES3 information for this research included power, area and execution times for 47 tasks from 5 application domains on 17 commercial processors. Task graphs from the five application domains were embedded into two SoC configurations in NoCSim as illustrated in Fig. 8.

#### D. Test Configurations

Assuming a chip size of 22 mm × 22 mm [17], consisting of network tiles sized at 4 mm × 4 mm, the IBM 405 GP processor was selected from ES3 processor list. Using the relevant specifications of the IBM 405 GP (idle power, peak power and clock rate), the network tiles in NoCSim were configured accordingly to provide the compute support to applications.

Two baseline SoC configurations were also developed using the tasks graphs in ES3. soc_config1 was assigned 11 task graphs spanning 4 application domains—office automation, consumer, networking, and auto industry. These were mapped onto a 4 × 4 2-D torus utilizing 13 network tiles (i.e., execute on 13 IBM 405 GPs). The mapping of the tasks has an impact on the performance of the design [18], but for the purpose of this research, we mapped the task graphs by hand. The test parameters of the cores were assigned based on the g1023.soc ITC’02 benchmark.

The second SoC configuration (soc_config2) also utilized the IBM 405 GP and had 9 task graphs from the telecom application domain mapped over 11 network tiles. Test parameters were assigned based on the d695.soc ITC’02 SoC benchmark. Fig. 9 illustrates the mapping of the test core specifications onto the corresponding network tiles. The digits in the tile correspond to the core id in the benchmark specification. Using these two baseline configurations, four additional configurations were prepared, differing in the number of TI-IPs assigned and their locations in the design. Fig. 8 illustrates these
additional configurations. To illustrate how the TI-IP would operate in the absence of executing applications, four additional configurations with just TI-IPs were also prepared. We report power profiles and finish time variations on these ten configurations in the following sub-section. It must be noted here, that the simulations in NoCSim do not actually run the applications or the test benchmarks, but instead characterizes execution profiles of the system which is reflected in the power profiles.

E. TI-IP Validation

SoC Configuration 1: Fig. 10 illustrates the power profile comparison between the three variations (applications only, test only and applications + test) of this configuration. In the simulation interval specified and for the configured input test parameters (from ITC’02 benchmarks), only a single IP core was tested. The power variation in this case is not much. When the parameters were altered to allow two additional IP cores to be tested, the power profile in Fig. 11 was observed. This demonstrated the sensitivity of the concurrent testing to the test parameters. For comparison purposes, the power profile for just the IP cores testing has also been plotted in the same.

This demonstrated the sensitivity of the concurrent testing to the test parameters. For comparison purposes, the power profile for just the IP cores testing has also been plotted in the same.

When multiple TI-IPs were deployed into the design (soc_config1_tiip2), similar power trends were observed (see Fig. 12), with one exception: fewer cores were tested in this configuration, since the TI-IP responsible for testing those cores did not have the test management token, highlighting the consequence of the IP core test responsibility mapping to individual TI-IPs.

When the average task execution times of the task graphs in soc_config1 were analyzed, there were no variations observed. This is attributed to the fact that the IP core that was tested, was not executing any tasks at that time, which highlights the test scheduling aspect and the need for an accurate online monitoring scheme to detect when to test an IP core. Fig. 13 compares the power profiles of the two TI-IP configurations.

SoC Configuration 2: In this test configuration, the observed power profile was similar in observed trends to that of SoC configuration 1 (see Figs. 14 and 15). Since the test configuration for this was based on d695.soc, the different test specifications demonstrated different operational characteristics (in terms of test power profile), especially in TI-IP only scenarios. In TI-IP configuration 1 (single TI-IP), nine IP cores were tested, while in TI-IP configuration 2 (two TI-IPs), four IP cores were tested. As observed earlier, this is due to the IP core test responsibility mapping to the individual TI-IPs discussed earlier. Fig. 16 illustrates the power profile comparison between the single TI-IP and multi-TI-IP configurations for SoC configuration 2. For this test configuration, a finish time variation of 1% was observed for
task graph 4 in the system. But this variation did not lead to any deadline misses.

F. Robust Protocol Validation

Fig. 17 demonstrates the energy profile for the system snapshot collection and beginning of test application on an IP core in the NoC. The normal online test mode curve is compared to the case of test starvation due to communication intensive application. The timeout-based mitigation technique delays start of test application. In the absence of this technique, no test application would take place.

A test case to demonstrate test throttling due to misbehaving application is demonstrated in Fig. 18. Since only SCAN test data was available, the TI-IP reduced the pace of test vector delivery to reduce test energy consumption.

G. Synthesis Results

An HDL implementation of TI-IP was synthesized with Synopsys Design Compiler [25] using Virginia Tech VLSI for Telecommunications TSMC-0.25-μm, 2.5-V standard cell library [26], [27], the gate count for the TI-IP was estimated at 83 K and power consumption estimated was 520 mW (leakage: 32 uW).

VIII. CONCLUSION

With reducing lifetime reliability in complex SoC designs and challenges in estimating MTTF, deployment of a concurrent online SoC test framework is the next step to address rising test costs. This research proposes the introduction of a test infrastructure-IP into a SoC design to provide online test support in presence of executing applications. Previous works consider SoC test in the absence of application, but prohibitive test times make turning off of the system infeasible. To address scalability aspects, the designed TI-IPs are capable of operating in a multi-TI-IP configuration, in which they cooperatively handle concurrent online test responsibilities. An NoC-based SoC design is considered as it provides for a superior communication infrastructure. Basic operation of the TI-IP is outlined and a functional model is included into NoCSim. Using application...
benchmarks (E3S) and SoC test benchmarks (ITC’02 SoC Test bench), two baseline SoC configurations and eight TI-IP variations were built and simulated on. Experiments demonstrated the correct operation of the TI-IP. For this research, we only consider SCAN testing of IP cores due to the lack of BIST test data. BIST support has been included in the current version of the TI-IP, but has not been tested with actual BIST test benches. Communication protocol specifications are also presented. Potential hazards to concurrent online testing have been identified and techniques to mitigate them have been proposed. Experimental results demonstrate robust operation of the communication protocol.

ACKNOWLEDGMENT

The authors would like to thank J. Lee for his assistance in the implementation and cost analysis of the TI-IP.

REFERENCES


Praeven S. Bhojwani (S’05–M’08) received the B.Tech. (H) degree in computer science and engineering from the Indian Institute of Technology (IIT), Kharagpur, India, in 2001, and the M.S. and Ph.D. degrees in computer engineering from Texas A&M University, College Station, in 2003 and 2007, respectively.

His research interests include networks-on-chip, low-power design, computer architecture, and electronic design automation.

Rabi N. Mahapatra is an Associate Professor with the Department of Computer Science, Texas A&M University, College Station, where he is currently directing the Hardware-Software Codesign Research Group. His research interests include the area of embedded systems, low-power design, SoC and VLSI Design, and computer architecture. He has published more than 100 research papers in international journals/conferences. His research has been funded by NSF, DoT, NASA, and other major industries.

Prof. Mahapatra is a Ford Fellow and a Distinguished Visitor of IEEE Computer Society.