BIST TPG for Faults in Backplane Interconnect

Chen-Huan Chiang and Sandeep K. Gupta

CENG 97-17

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213) 740-2251

October 1997

BIST TPG for Faults in Backplane Interconnect *

Chen-Huan Chiang and Sandeep K. Gupta
Electrical Engineering – Systems
University of Southern California, Los Angeles, CA 90089-2562

Category: (2.6) BIST & DFT schemes, and boundary scan.

Corresponding Author

Sandeep K. Gupta
M/S 2562
Electrical Engineering – Systems
University of Southern California
Los Angeles, CA 90089-2562
Tel: (213)740-2251
Fax: (213)740-9803
Email: sandeep@boole.usc.edu

Primary Author

Chen-Huan Chiang
EEB332
Electrical Engineering – Systems
University of Southern California
Los Angeles, CA 90089-2562
Tel: (213)740-2354
Fax: (213)740-9803
Email: chenhuan@boole.usc.edu

*This research was funded by Lucent Technologies

BIST TPG for Faults in Backplane Interconnect

Abstract

In this paper, we present a new system interconnect BIST architecture and a methodology to program this architecture to generate test patterns for backplane interconnect faults via the test bus (IEEE1149.5 or IEEE1149.1 Extension) in the backplane and boundary scan architecture of each board in the system. The test architecture includes a Master BIST *in the master board and a* Slave BIST *in each slave board. One of the key features of the proposed BIST architecture is that its design is independent of the system configuration and a simple test scheduling algorithm is sufficient to test all interconnect faults in a system while avoiding multi-driver conflicts on backplane nets as well as local nets of each board. In this test scheduling, the shorts and stuck-at faults are tested by enabling the Master BIST, while opens are tested by the all BISTs in the system. Procedures to design the master TPG as well as slave TPGs are presented using the concepts presented in [7].*

The overall methodology guarantees the detection of all modeled faults using short test sequences with low area overheads, while ensuring that no circuit damage will occur due to multi-driver conflicts. All these features of the proposed BIST, along with its ability to easily adapt to changes in a system's configuration, are demonstrated via its application to test the backplane of three versions of a system that uses a VME backplane.

BIST TPG for Faults in Backplane Interconnect [1]

Abstract

In this paper, we present a new system interconnect BIST architecture and a methodology to program this architecture to generate test patterns for backplane interconnect faults via the test bus (IEEE1149.5 or IEEE1149.1 Extension) in the backplane and boundary scan architecture of each board in the system. The test architecture includes a Master BIST *in the master board and a* Slave BIST *in each slave board. One of the key features of the proposed BIST architecture is that its design is independent of the system configuration and a simple test scheduling algorithm is sufficient to test all interconnect faults in a system while avoiding multi-driver conflicts on backplane nets as well as local nets of each board. In this test scheduling, the shorts and stuck-at faults are tested by enabling the Master BIST, while opens are tested by the all BISTs in the system. Procedures to design the master TPG as well as slave TPGs are presented using the concepts presented in [7].*

The overall methodology guarantees the detection of all modeled faults using short test sequences with low area overheads, while ensuring that no circuit damage will occur due to multi-driver conflicts. All these features of the proposed BIST, along with its ability to easily adapt to changes in a system's configuration, are demonstrated via its application to test the backplane of three versions of a system that uses a VME backplane.

1 Introduction

A passive backplane architecture is widely used to build a system which may consist of a plug-in processor board and multiple plug-in add-on boards interconnected by a system bus in the backplane. A backplane architecture is much more advantageous than a motherboard architecture in terms of repair time, system upgrade and system change by board substitution and also has more system expansion slots required for complicated industrial applications. The rugged construction of a typical passive backplane system also provides reliable operation in an industrial environment. However, during the system assembly process or field operation, boards may be plugged into the wrong slot, pin connectors of a board may be bent, broken or shorted, and backplane traces may be damaged, open or even shorted with each other. Hence, testing the interconnection of a backplane is an important task during system assembly as well as field operation.

As the use of IEEE 1149.1 boundary scan architecture (BSA) [14] at board level becomes commonplace, the concepts of hierarchical test [1, 4, 29, 30] in a system environment get mature along with the advent of IEEE 1149.5 module test and maintenance (MTM) standard [15] for backplane, structural testing of system level interconnects becomes an interesting problem. Since the complexity of the overall interconnect test is large and the system configuration can change

[1]This research was funded by Lucent Technologies

constantly, generation of deterministic test patterns is difficult to manage and may even be impossible. Built-In Self-Test (BIST) can make the testing problem transparent to the users via automation. but also automate system testing. If the BISTs (in the following, this term will be used to refer to the hardware that implements BIST) of a system are programmable (for example, the BISTs can be reconfigured to target different fault models), then the BIST approach becomes more advantageous. During manufacture testing, when complete coverage of all types of faults and high diagnosis resolution are required, we can configure the BIST for the highest test quality. While for the testing during field operation, once shorts due to solder splash have been tested at assembly phase, we can reconfigure the BISTs for detecting open faults only. The resulting BISTs have shorter test time with reasonable fault coverage and diagnosis resolution.

There are similarities between testing for the inter-chip interconnects at board level and testing for the inter-board interconnects at backplane level. For instance, in both cases, test methodologies must avoid multi-driver conflicts and achieve complete fault coverage. However, test methodologies for testing backplane interconnects need to be independent of system configuration (which may change constantly) and test synchronization between boards in the system must be ensured. Furthermore, the test bus architecture used at board level is a single boundary scan chain, while at backplane level, a multi-drop test architecture (for example, a 1149.1 Extension [16, 29, 30] or 1149.5 MTM test bus) is used more often than the other architectures, such as a single long scan chain connecting all boards or a star configuration [23].

In the following, after a discussion of previous work, the objective of backplane testing will be given and test issues pertaining to backplane testing will be raised. The models of test bus and system bus at the backplane will be defined and followed by the target interconnect fault model and test conditions. We will then present a system interconnect test architecture and a BIST test pattern generation (TPG) framework. A TPG design procedure, based on the generalized input reduction techniques described in [7], will use the framework to program the test architecture to achieve safe testing and complete fault coverage in shorter time and less area overhead.

2 Background

The objective of *backplane testing* is to test the interconnect structure of the *system bus* which resides in the backplane and will be referred to as the circuit under test (CUT). The byproduct of backplane testing is the testing of the *edge pin connectors* which connect each board to the system bus. In backplane testing, all the *local nets* of a board are assumed to be well tested. Similar to how the boundary scan chain helps interconnect testing between chips at board level, we assume the presence of a *test bus* which is either a part of system bus or comprises of a few extra lines added to help the interconnect testing at backplane level.

2.1 Previous Work

Most previous work in testing interconnects focused on the development of deterministic tests for interconnect between chips at the board level [9, 11, 17, 19, 25, 28, 31]. Although there are similarities between interconnects at board and backplane levels, the differences between them make the extension of the board level methods to backplane testing non-trivial. In [2, 3], once the global knowledge of the system configuration and board level net lists are available at test time, test methods used at board level are extended to test system level interconnect. Furthermore, a dominant short fault model is defined to address shorts at system level where components made of different technologies are used. In the walking enable algorithm [20], suitable use of disable and enable vectors makes backplane testing independent of changes in system configuration.

In the interconnect test phase of [10], only board level interconnects are tested. The decentralized BIST [26] addresses interconnect testing at both board and backplane levels. Their BIST scheduling at backplane level is conceptually similar to the concept of walking enable algorithm [20]. Although the BIST TPGs are distributed in the individual boards, the responses need to be analyzed in a centralized master module. Besides, it is difficult to update the test architecture when the system is reconfigured.

2.2 Test Issues

All the test issues for inter-board interconnects at backplane level will be addressed briefly in the following.

2.2.1 Independence from System Configuration

There are continual changes and updates in a system configuration since different boards may be plugged into the same slot at different times. It is impossible to keep the information of all possible system configurations for testing purposes. Therefore, it is necessary to make testing relatively independent of system configuration. The following two methods can make backplane testing relatively independent of the system configuration:

1. Encapsulation of board level information: For each board, just a few boundary scan cells (BSCs), referred to as *pin connector boundary scan cells* (PCBSC), are connected to the pin connectors which link to the system bus in the backplane. The information about other nets on a board is irrelevant to the backplane testing. Therefore, if the information about pin connectors is encapsulated in a certain form, backplane testing can be made independent of system configuration.

2. Distributed approach: By putting as much test control and data as possible within the BIST on each slave module, the dependence on the master module can be minimized [22], making

easier the process of system integration. Also, system testing becomes faster because the data traffic between master and slave modules is decreased.

2.2.2 Test (Shift) Synchronization

The presence of more than one board in a system results in more than one boundary scan chain. All scan chains must be synchronized so that the BSCs connected to backplane nets are updated simultaneously with valid test patterns. Otherwise, intermediate test patterns may be applied and cause driver conflicts to damage the system [20].

2.2.3 Avoidance of Multi-Driver Conflict

Since boundary scan is used to test faults in interconnections, the application of a test pattern that enables multiple drivers driving opposite values on a given net can cause circuit damage due to excessive current flow. At system level, both backplane nets and *local nets*, i.e. the nets at board level, must be considered to ensure that no such test patterns can be applied. While the application of such patterns can be avoided easily if deterministicly generated test patterns are applied under external control, a BIST TPG must be carefully designed so that no such test patterns are applied to nets at backplane level as well as at board level.

2.2.4 Complete Fault Coverage

As shown in [3, 18], a circuit board and a system contain components that are made of different technologies. The fault behaviors vary from technology to technology. Especially for shorts, the traditional AND/OR type of shorts are not sufficient to model the short faults in a mixed technology environment. It is important to derive adequate fault models that can best describe fault behaviors in a mixed technology system.

If BIST is used, especially when distributed BIST is used for each board, it is necessary for complete fault coverage to ensure (i) the number of shifts selected does not cause *decimation* of the test sequence, i.e. repeated application of a fraction of the test vectors that would be applied for a different number of shifts; and (ii) no dependency/correlation exists between the patterns applied to backplane nets that are driven by BIST circuits in different boards.

3 Backplane Models

3.1 Test Bus Model

The multi-drop configuration shown in Figure 1 is adopted as our test bus architecture [5]. In a multi-drop test bus, there is a system test controller that controls the test bus via a master interface. A slave interface on each board links the board test resources (distributed BIST and

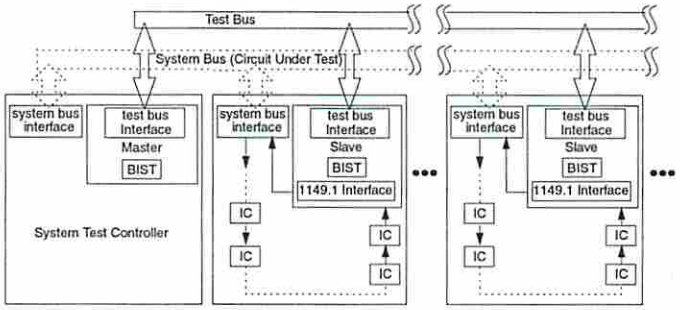

Figure 1: The multi-drop test bus architecture.

BSA) to the test bus. The boards are addressable so that one or more boards can be selected at the same time. That is, although the boards are physically connected to the test bus on the backplane, the logical connections depend on the addressing protocols. Since the boards are connected in a multi-drop fashion, adding or removing a board does not disrupt the connection of the others. Multi-drop test bus architecture, such as 1149.1 Extension and 1149.5 MTM bus, can be used as the test bus.

3.2 Circuit Model

3.2.1 Observations

The CUT in backplane testing is the system bus that resides in the backplane. Several system bus and backplane standards have been found to share the following characteristics.

1. In a multi-drop system bus, backplane nets are tri-state and no wire-nets exist.

2. The first slot is a system slot. However, the pin connectors for each slot are identical.

3. Physically, most backplane nets connect the pin in same position of the pin connectors at all slots, i.e. they physically create a broadcast interconnection. However, some pin connectors are connected in a daisy chain fashion.

4. There is usually a system bus interface that links each board to the system bus in the backplane as shown in Figure 1. In such a board, only a single driver drives a signal in the system bus. However, in some boards, multiple drivers drive a signal in the system bus.

5. The PCBSCs are mostly bidirectional. The usage of bidirectional BSCs eases the interconnection testing problem (especially for detecting shorts) by a great deal since they can simultaneously be a driver as well as a receiver and provide full controllability and observability of a net.

3.2.2 System Bus Model

Based on the above observations, the following assumptions are made about the characteristics of CUT in backplane testing:

1. The CUT is a multi-drop system bus, where a master board controls several slave boards.
2. Each backplane net is *boundary scannable*, i.e. (i) all cells of the backplane net are boundary scan cells, and (ii) each backplane net has at least one driver and at least one receiver.
3. All PCBSCs connected to the backplane interconnects are tri-state.
4. A backplane net is either a physically broadcast interconnection of all boards or a daisy chain connection from one board to another. Note that a private connection from one specific board to another can be established by dedicating a physically broadcast backplane net which consists of undefined pins in the system bus.

3.2.3 Circuit Information Model

The circuit information needed for backplane BIST at each board (including master and slave boards) can be divided into the following three categories:

1. Information that is required at each board for designing distributed BIST: (i) the location (PCBSC or non-PCBSC) and functionality (data or control) of each BSC; (ii) the enable value of each control BSC; (iii) *pin connector net list* which describes the interconnection of PCBSCs to each pin of the board, and (iv) *backplane net list* of the system which describes the interconnection of pins of each board in the system bus backplane.
2. Information that each board must provide for backplane test scheduling. During backplane testing, a board is said to be *enabled* when enable value is assigned to at least one of the PCBSC control cells of its backplane nets; a board is said to be *disabled* when the disable values are assigned to all its PCBSC control cells. *Test scheduling* describes how the master module controls each slave module by enabling or disabling it to accomplish backplane testing. Therefore, each board must provide circuitry to decode the enable/disable command issued by the master module for test scheduling.
3. Information that each board must provide for test synchronization: a test synchronization finite state machine (TS-FSM) and the boundary scan chain length of the board.

Additional information is required for the system test controller at the master board for test scheduling and test synchronization, which includes the number of slots, the number of boards, and the boundary scan chain length of each board. Besides, a test synchronization control procedure which communicates with the TS-FSM in each slave module and a test scheduling algorithm must be programmed into the system test controller.

3.3 Fault Model and Test Conditions

3.3.1 Fault Model

A *stuck-at* fault affects the entire net while an *open* may only affect a part of the net. The behavior of an open depends on the technology of the receivers of the net. In the following, we assume that a logic-1 value is captured by a receiver when one or more opens disconnect it from the drivers.

Only *pairwise shorts* are considered in the following because multiple net shorts are automatically detected if pairwise shorts are detected. The technology and driving strengths of the output drivers involved in a short affect the behavior of a short fault [3, 18, 12]. The behavior can be either deterministic or non-deterministic. Deterministic short fault behavior between two backplane nets can be characterized as 0(1)-dominant and net-dominant faults [3]. 0(1)-dominant faults are generalizations of the traditional AND(OR) short faults and a net-dominant fault is equivalent to the traditional strong driver short.

A tri-state backplane net is said to be *disabled* when all its drivers are simultaneously disabled. In the following, we assume that a disabled net holds a deterministic value. To simplify the discussion, we assume that a disabled tri-state net holds a logic-1 but our results are applicable even for nets that hold a logic-0. We also assume that the driving strength of an enabled tri-state driver is significantly higher than that of a disabled one.

The dominant short fault model addresses the issue of mixed technologies in backplane interconnects while the open and stuck-at fault models cover most of the defects that are usually seen at system level, such as bent connector and loose connection. We will consider open, stuck-at, and pairwise dominant faults in the backplane testing.

3.3.2 Test Conditions for Backplane Interconnection Faults

Test conditions for a fault represent all possible tests that can detect the fault. They are used by our BIST design methodology to reduce TPG length while ensuring complete fault coverage.

Since a stuck-at fault on a net affects the whole net, it is necessary and sufficient to drive a logic-0(1) value at any driver to detect the stuck-at-1(0) fault on the net.

To detect an open on a tri-state backplane net, a test must: (i) enable one of the drivers and disable all other drivers of the net; (ii) apply a logic-0 at the enabled driver (since an open is assumed to result in the capture of a logic-1 at the receivers of the net); (iii) check the response captured at *all* the receivers of the net.

Test conditions for pairwise shorts between two tri-state backplane nets are either (i) enable one tri-state net with logic-0 (since a disabled tri-state backplane net is assumed to hold a logic-1) and disable the other, or (ii) enable both nets and drive a pair of opposite values on them, i.e. '0' on one net and '1' on the other and vice versa, to detect the dominant shorts. The first approach

of enabling one and disabling the other tri-state net can detect the short even if the behavior of the short is non-deterministic when the both tri-state nets are enabled. It also provides safer test patterns to detect shorts by avoiding large current that may otherwise flow via the short.

3.3.3 Definitions

The notions of *incompatibility* [6] and *conditional incompatibility* [7] can be used to describe the constraints that a TPG must satisfy to guarantee the satisfaction of the test conditions of all target faults while avoiding the application of any test pattern that can cause circuit damage. Their definitions in the context of interconnect testing are given in the following and detailed examples can be found in [7].

Definition 1 (Incompatibility) If two boundary scan cells a and b cannot be assigned to the same stage of a TPG with the same polarity, then a and b are said to be *incompatible* and denoted by $a \not\sim b$. □

For example, the essential constraint to avoid multi-driver conflicts by enabling at most one driver of a multi-driver net now be written as:

$$c_i \not\sim c_j, \tag{1}$$

for all control cells c_i and c_j that control drivers on the same net.

Definition 2 (Conditional incompatibility) If two boundary scan cells w and x cannot be assigned to the same stage of a TPG when some other constraints Γ are not satisfied, then w and x are said to be *conditionally incompatible w.r.t.* Γ and denoted by $\{(w \not\sim x) \vee \Gamma\}$, or $\{\mathit{if}\ (w = x)\ \mathit{then}\ \Gamma\}$. □

In general, for two tri-state nets i and j with the same net degree (k) at a board, the constraints for the satisfaction of test conditions of pairwise shorts between them can be captured by the following $k!$ conditional incompatibilities:

$$\begin{cases} \text{if } (c_{i,1} = c_{j,1} \wedge c_{i,2} = c_{j,2} \wedge \ldots \wedge c_{i,k} = c_{j,k}) \\ \text{then } (d_{i,1} \not\sim d_{j,1} \vee d_{i,2} \not\sim d_{j,2} \vee \ldots \vee d_{i,k} \not\sim d_{j,k}), \\ \vdots \\ \text{if } (c_{i,1} = c_{j,k} \wedge c_{i,2} = c_{j,(k-1)} \wedge \ldots \wedge c_{i,k} = c_{j,1}) \\ \text{then } (d_{i,1} \not\sim d_{j,k} \vee d_{i,2} \not\sim d_{j,(k-1)} \vee \ldots \\ \qquad \ldots \vee d_{i,k} \not\sim d_{j,1}) \end{cases} \tag{2}$$

where $c_{i,\alpha}$ ($c_{j,\alpha}$) is the control cell corresponding to the data cell $d_{i,\alpha}$ ($d_{j,\alpha}$), $\alpha = 1, 2, \ldots, k$ on tri-state nets i and j of a board.

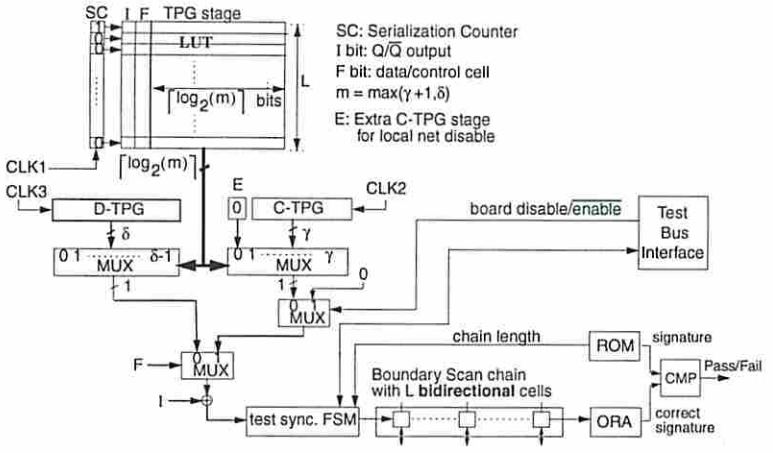

Figure 2: Master BIST architecture.

4 System Interconnect Test Architecture

Since both the system bus under test and the test bus are assumed to be multi-drop architectures, a master board must exist in the system and the master test module can be incorporated into the master board. Since there is only a single master board of a system, we can afford a comprehensive BIST at the master board. Algorithms to schedule and synchronize BISTs on each board must be provided for the system test controller which resides in the master board. The availability of comprehensive set of test capabilities in the master can help greatly simplify the BIST in all other boards of a system — a great reduction in cost since a backplane may contain one master but over a dozen slaves.

In the proposed system interconnect test architecture, the BIST architecture shown in Figure 2 is implemented in the master board of the system and referred to as *Master BIST*, while the BIST architecture shown in Figure 3 is distributed in each slave board and referred to as *Slave BIST*. The independence from system configuration, test synchronization, and the avoidance of multi-driver conflicts on local nets can be achieved by the new hardware features in the proposed system test architectures. Primitives for enabling/disabling a board are described in the proposed architecture and they can then be used in test scheduling algorithms (which will be described in Section 5) to safely test all target faults independently of system configuration.

4.1 The Master BIST

The Master BIST, as shown in Figure 2, contains a look-up table (LUT), two test pattern generators (a C-TPG that generates test patterns for the control cells in the boundary scan chain and a D-TPG that generates test patterns for the data cells), a read-only memory (ROM), a TS-FSM, an output response analyzer (ORA) and a test bus interface. Finally, PCBSCs in the master board must be bidirectional.

4.1.1 Data Encapsulation

The board level information required for backplane testing is stored in the LUT. The number of entries in the LUT is equal to the length of the board's boundary scan chain (L). Each entry of this table corresponds to a specific BSC in the scan chain and has total of ($\lceil \log_2 m \rceil + 2$) bits (where m is the larger of the C-TPG size (γ) and D-TPG size (δ)) and contains the following information: (i) one bit, F, is used to specify whether the cell is a control or data cell and is used to obtain the data to be scanned either from C-TPG or D-TPG, (ii) $\lceil \log_2 m \rceil$ bits are then used to select the appropriate stage of the selected TPG, and (iii) one bit, I, is used to scan into BSC either the content of the selected TPG stage or its complement. For each control cell entry, the I-bit represents the enable value of corresponding control cell. The I-bit of a data cell and the $\lceil \log_2 m \rceil$ bits for C-TPG/D-TPG stage are programmed in different manner for the following two types of BSCs:

1. PCBSC: These are cells connected to the backplane nets. For PCBSCs, the I bit of each data cell entry represents the polarity (Q or $\overline{Q}$) of the output of the appropriate D-TPG stage, while each control cell scans in only the positive output (i.e. Q) of the appropriate C-TPG stage. The assignments of the appropriate TPG stages are determined by the TPG design procedure described later in Section 6.

2. Non-PCBSC: These are cells connected to the local nets on the board. Each non-PCBSC entry of the LUT is designed in such a manner that all local tri-state nets are disabled to avoid conflicts by an extra stage of the C-TPG that always outputs a '0' value which generates a disable value for each non-PCBSC control cell according to bit I of the LUT. To further ensure the avoidance of multi-driver conflicts, all non-PCBSC data cells can be assigned to the same stage of the D-TPG, say, stage 0, so that identical values are driven on all multi-driver local nets.

4.1.2 BIST Operation

In the Master BIST, the ROM, the ORA and two TPGs form the core logic for BIST. The on-board ROM contains the correct signature of the expected output response for ORA and the length of the boundary scan chain at the board (L) for test synchronization. The C-TPG is a γ-stage one-hot counter while the D-TPG can be a δ-stage linear feedback shift register (LFSR), a δ-stage one-hot counter, or a counting sequence generator that generates $\lceil \log_2(2\delta + 2) \rceil$ patterns.

When the master board is enabled by the system test controller via the test bus interface, the Master BIST generates appropriate test patterns and synchronizes with other boards. The synchronized test patterns are then scanned into the boundary scan chain and applied to the backplane nets while avoiding conflicts on local nets and the response is captured simultaneously. Finally, Master BIST collects output response signature when the captured response is shifted out.

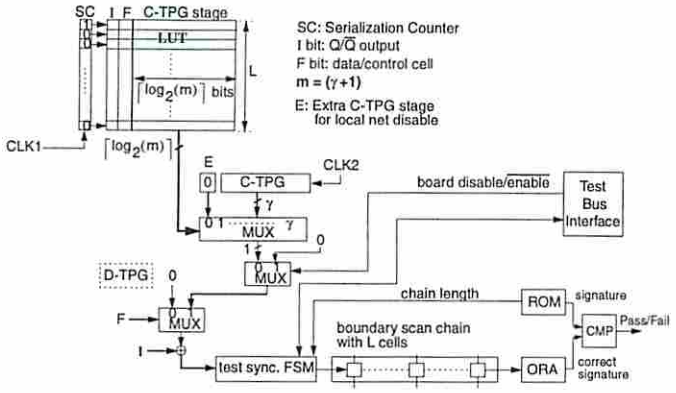

Figure 3: Slave BIST Architecture.

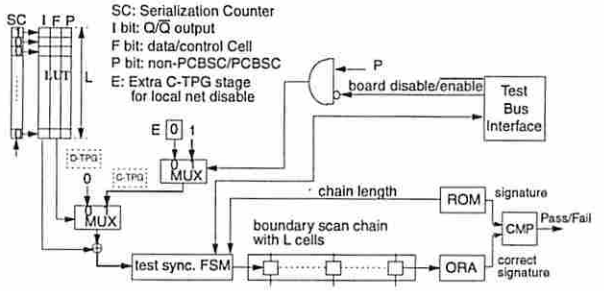

Figure 4: The simplified Slave BIST.

Since the boundary scan chain length may be different for each board in a system, a test synchronization control procedure in the system test controller at the master board and a TS-FSM on each board are proposed to resolve the test synchronization problem. Based on the boundary scan chain length stored in each on-board ROM and sent to the master board by each on-board TS-FSM, the longest boundary scan chain length, L_{max}, is selected by the system test controller and sent back to the TS-FSM on each slave board. TS-FSM then appends additional PAUSE states (defined in the BSA test access port (TAP) controller) after the desired test patterns are shifted into the scan chain so that the total number of shifts on each board before the application of the test patterns are equal.

The application of a single pattern begins by clocking of the C-TPG and/or D-TPG to generate a new test pattern. In the following, we assume that the D-TPG is clocked once for each pattern while the C-TPG is clocked after the application of each set of |D-TPG| patterns, where |D-TPG| denotes the length of the sequence generated by the D-TPG. Once a new pattern is generated, the contents of both TPGs are held constant while the serialization counter is clocked L times and holds for $(L_{max} - L)$ clocks while the BSCs in the scan chain are clocked L_{max} times for test synchronization. In each clock during this period, the LUT entry corresponding to the BSC whose content is being shifted in is accessed and used to select appropriate stage of either the C-TPG or D-TPG and its content inverted, if so specified by the table entry, and shifted into the BSC.

When the master board is disabled during test scheduling, Master BIST disables all the PCBSCs control cells but still collects responses from the backplane nets and computes a signature for ORA.

4.2 The Slave BIST

The Slave BIST (as shown in Figure 3) in each slave board differs from the Master BIST in only two ways: (i) it contains no D-TPG, and (ii) its PCBSCs need not be bidirectional. The Slave BIST can be further simplified to the one shown in Figure 4 if there is no multi-driver net in the pin connector net list of the board, which is the case when a single system bus interface chip is

used to link the board to the system bus. In the simplified Slave BIST, bit P of each LUT entry, instead of multiple bits for TPG stage, is used to decide whether the cell corresponding to the entry is a PCBSC or not.

The application of the test patterns and test synchronization are the same as described above for the Master BIST. When the slave board is enabled by the system test controller via the test bus interface, the Slave BIST generates all '0' test patterns to PCBSC data cells while avoiding conflicts on local nets and computes output response signature when the output is shifted out. When the slave board is disabled during test scheduling, Slave BIST disables all PCBSCs control cells but still collects responses from the backplane nets and computes a response signature for ORA.

5 Proposed BIST TPG Framework

As described above, the independence from the system configuration, test synchronization and the avoidance of local net conflicts are achieved via new features of the system interconnect test architecture. In the proposed BIST TPG framework, the avoidance of driver conflicts and complete fault coverage at backplane level will be addressed by adding constraints to the TPG design. These constraints at backplane level can be added to the generalized input reduction TPG design procedure described in [7]. The key objective of the TPG design is to assign all BSCs to corresponding TPGs with minimum number of stages, subject to constraints which ensure that no multi-driver conflict occurs and coverage of all faults is guaranteed.

5.1 Constraints to Achieve Complete Fault Coverage

5.1.1 Pairwise Short Faults

Consider the case that nets i and j are a pair of tri-state backplane nets with the same net degree k, and the system configuration and complete backplane net list are given. Hence, the board level BIST TPG methodology described in [7] can be applied to design a highly centralized BIST TPG for backplane testing. Therefore, according to the board level BIST TPG methodology, the following $k!$ conditional incompatibilities need to be satisfied for testing pairwise short fault between nets i and j:

$$
\begin{cases}
\text{if } ((c_{i,1})_{B_{x,1}} = (c_{j,1})_{B_{y,1}} \wedge (c_{i,2})_{B_{x,2}} = (c_{j,2})_{B_{y,2}} \wedge \ldots \wedge (c_{i,k})_{B_{x,k}} = (c_{j,k})_{B_{y,k}}) \\
\text{then } ((d_{i,1})_{B_{x,1}} \not\sim (d_{j,1})_{B_{y,1}} \vee (d_{i,2})_{B_{x,2}} \not\sim (d_{j,2})_{B_{y,2}} \vee \ldots \vee (d_{i,k})_{B_{x,k}} \not\sim (d_{j,k})_{B_{y,k}}), \\
\vdots \\
\text{if } ((c_{i,1})_{B_{x,1}} = (c_{j,k})_{B_{y,k}} \wedge (c_{i,2})_{B_{x,2}} = (c_{j,(k-1)})_{By,(k-1)} \wedge \ldots \wedge (c_{i,k})_{B_{x,k}} = (c_{j,1})_{B_{y,1}}) \\
\text{then } ((d_{i,1})_{B_{x,1}} \not\sim (d_{j,k})_{B_{y,k}} \vee (d_{i,2})_{B_{x,2}} \not\sim (d_{j,(k-1)})_{B_{j,(k-1)}} \vee \ldots \\
\qquad \ldots \vee (d_{i,k})_{B_{x,k}} \not\sim (d_{j,1})_{B_{y,1}})
\end{cases}
\tag{3}
$$

where $c_{i,\alpha}$ ($c_{j,\alpha}$) is the PCBSC control cell corresponding to the PCBSC data cell $d_{i,\alpha}$ ($d_{j,\alpha}$), $\alpha = 1, 2, \ldots, k$, at boards $B_{x,1} \ldots B_{x,k}$ ($B_{y,1} \ldots B_{y,k}$) which represent different boards or the same board, depending on the system configuration. Note that these PCBSCs may be distributed all over different boards in the system. Constraint (3) is not guaranteed to be satisfied unless the system configuration is known in advance. In order to be relatively independent of system configuration, we propose a method for detecting short, which consists of a test scheduling and the Master BIST architecture shown in Figure 2. The test scheduling for shorts is to enable the Master BIST in master board while disabling all the other boards in the system. Since each PCBSC in the Master BIST is implemented by a bidirectional BSC, each PCBSC is a driver with complete observability of its backplane net which is assumed to be a physically broadcast interconnect. Therefore, the board with the Master BIST contains the driver signals of all backplane nets. Under the test scheduling for shorts, because all the slave boards are disabled, constraint (3) can be simplified as (2), where $c_{i,\alpha}$ ($c_{j,\alpha}$) is now the PCBSC control cell corresponding to the PCBSC data cell $d_{i,\alpha}$ ($d_{j,\alpha}$), $\alpha = 1, 2, \ldots, k$ at the master board.

Furthermore, since only the Master BIST is used for test shorts, no dependence and correlation can occur in the test patterns and the output response analysis can also be performed within the Master BIST. When enabling Master BIST to test shorts, only the result of ORA at master board is checked while the result of ORA of each slave board is ignored.

5.1.2 Open Faults

For the detection of opens on the backplane nets, at most one driver of a net can be enabled at a time. That is,

$$(c_k)_{B_i} \not\sim (c_l)_{B_j}, \tag{4}$$

for PCBSC control cells c_k and c_l that control drivers on the same backplane net at boards B_i and B_j, where B_i may be either the same as or different from B_j. Note that these PCBSCs may be distributed on different boards in the system. Again, constraint (4) can not be satisfied without knowing the system configuration in advance. An alternative solution using the proposed method for test scheduling, which is relatively independent of system configuration, also exists. In this method, for each net, one driver pin of one board is enabled at a time. Hence, at most one *driver pin* of a backplane net is enabled at any time. Only the number of slots and boards in the system and the pin connector net list of each board are required as inputs by the test scheduling algorithm. To increase test parallelism and save test time, this solution can be further enhanced to not only enable one driver pin of one board at a time for all pins, but to enable all driver pins of the enabled board with all '0's pattern (since an open is assumed to result in the capture of a logic-1 at the receivers of the net). Hence, at most one driver pin of every single backplane net is enabled at any time and all opens in backplane nets can be detected in parallel. However, in order to achieve diagnostic resolution of open faults in the multi-driver pin connector nets that connect to all the

enabled driver pins of the enabled board, at most one control cell of the multiple drivers of the pin connector nets can be enabled at a time, which can be written as:

$$c_i \not\sim c_j, \tag{5}$$

for PCBSC control cells c_i and c_j that control drivers on the same pin connector net at the enabled board.

The enhanced solution using test scheduling for open faults pushes the test parallelism to the limit without any TPG for data cells, since only the all '0' pattern is applied. The simple hardware for the enhanced solution for opens must be distributed to each slave board in the system as shown in the Slave BIST in Figure 3 and constraint (5) must be satisfied for master and each salve board during TPG design. The test scheduling algorithm for opens is to enable one board at a time while disabling the others for all boards in the system. This test scheduling covers the test scheduling for shorts, which is to enable the Master BIST in the master board while disabling all the other boards in the system. When a slave board is enabled during test scheduling for opens, the output response analysis for opens at each salve module is relatively simple since each board expects all '0' pattern when testing for open and the result of ORA of each board must be checked. Hence, opens at the driver pins of the enabled salve board and opens at the receiver pins of any disabled boards can be tested. Since each PCBSC in the Master BIST is bidirectional, an open at any pin of the master board, which can acts as a receiver or driver, can be tested when any one of the slave boards is enabled. When the master board is enabled during the testing of opens, because the D-TPG of Master BIST generates test patterns other than all '0' pattern, the result of ORA at each board is ignored. (If there are nets that are only driven by the master, then the correct signature of the test patterns generated by Master BIST would be required at one of the slaves.) Therefore, all opens on backplane nets are tested when each board is enabled at a time for all boards while ORA at each board is checked.

5.1.3 Stuck-at Faults

Coverage of stuck-at faults can be obtained as a byproduct when the master board, which contains the Master BIST architecture, is enabled. Since the patterns applied to the PCBSC data cells depend on the D-TPG (which can be implemented by a one-hot counter or a counting sequence generator) and TPG assignment in the LUT, both '0' and '1' are applied to each backplane net during backplane testing. Therefore stuck-at fault coverage is guaranteed.

5.2 Constraints to Avoid Multi-driver Conflicts

For backplane nets, the multi-driver conflicts can be avoided by enabling at most one driver on each backplane net at a time. Both test scheduling for testing open and short/stuck-at faults enable at most one board at a time while constraint (5) ensures that at most one PCBSC driver is enabled

for each backplane net at the enabled board. Therefore, the avoidance of multi-driver conflicts on backplane is guaranteed.

5.3 Constraints at Board Level and Extension of the Framework

There are two types of nets on each board: pin connector nets and local nets. The avoidance of multi-driver conflicts and the complete fault coverage of pin connector nets of a board can be achieved as byproducts of the satisfaction of constraints (2) and (5). As discussed in Section 2, in the backplane interconnect testing, all the local nets of a board are assumed to be well tested and the avoidance of multi-driver conflicts on local nets can be achieved by the new features in the proposed system interconnect test architecture.

Some constraints in the framework can be removed or simplified depending on the structure of a specific target backplane. The framework is also flexible to include additional constraints to re-program the BIST architecture for higher diagnostic resolution. If there exist other system buses in the test hierarchy (such as mezzanine cards and private chases) besides the one in the backplane, the same system interconnect test architecture and BIST TPG design framework can be recursively applied.

6 BIST TPG Design Procedure

Since many constraints for BIST TPG design are satisfied by new features in the system interconnect BIST architecture, such as the avoidance of multi-driver conflicts at local nets, Slave BIST for open faults at each slave board and Master BIST for short/stuck-at faults at the master board. Also, via test scheduling, constraint (3) can be reduced to (2) and constraint (5) can be used to avoid multi-driver conflicts on backplane nets. The objective of the TPG design is to assign all BSCs to corresponding TPGs with minimum number of stages, subject to constraints for the avoidance of multi-driver conflict and complete fault coverage. Hence, for backplane nets as well as pin connector nets on each board, the BIST TPG design of system interconnect testing can be divided into two parts: (i) a BIST TPG design procedure for Master BIST at the master board, which must satisfy constraints (2) and (5), and (ii) a BIST TPG design procedure for Slave BIST at each slave board, which must satisfy constraint (5). Note that if there is no multi-driver net in the pin connector net list of the board, the simplified Slave BIST in Figure 4 can be used and no BIST TPG design procedure for Slave BIST is needed. They can be written as follows:

$$\begin{array}{ll} \text{For Master BIST:} & \text{For Slave BIST:} \\ \begin{cases} \text{Minimize } |\text{C-TPG}| \times |\text{D-TPG}|, \\ \text{subject to constraints (2) and (5).} \end{cases} & \begin{cases} \text{Minimize } |\text{C-TPG}| \times |\text{D-TPG}|, \\ \text{subject to constraint (5).} \end{cases} \end{array} \tag{6}$$

where |TPG| denotes the length of the sequence generated by a TPG. The branch-and-bound algorithm proposed in [7] can also be used to solve (6) for the TPG design.

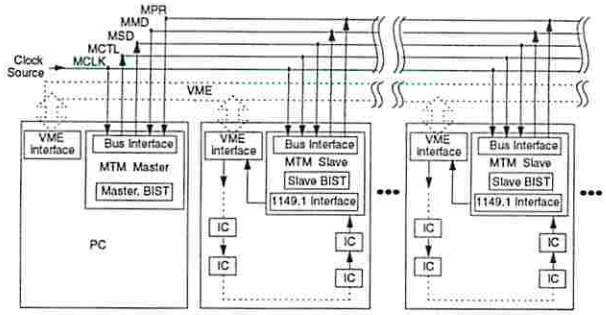

Figure 5: 1149.5 MTM in a VME system.

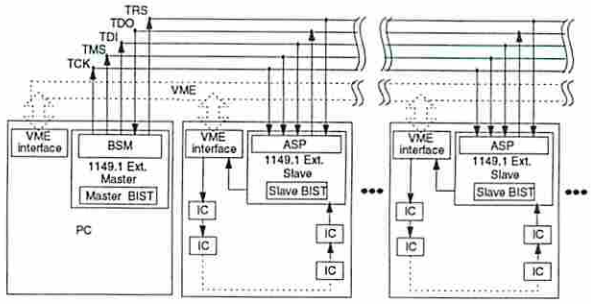

Figure 6: 1149.1 Extension in a VME system.

If the objective of TPG design becomes the minimization of the LUT size, it can be changed to minimize the maximal of |C-TPG| and |D-TPG| instead of minimizing the product of them in (6).

7 Case Study — VME Backplane

VME is a popular open standard system for industrial applications such as telecommunication, real-time systems, and multi-processing. It includes definitions for VME boards, backplanes and protocols. It is so well defined that some other standards, such as VIX (mostly seen in automatic test and data acquisition systems) and CompactPCI (a rugged version of PCI), are also based on it. As demands for system testing are getting higher, IEEE1149.5 MTM bus is being proposed to be a part of the new VME64 Extensions [27].

Either the 1149.5 MTM bus or 1149.1 Extension can be used as the test bus for backplane testing. In the MTM test bus environment, as shown in Figure 5, a system test controller (which itself can be controlled by a PC) is assumed to be at the master board and a slave interface [10, 13, 24] which can communicate with the 1149.1 BSA on each board is also assumed. In the 1149.1 Extension environment, as shown in Figure 6, a BSM controlled by a PC resides in the master board and TI ASP [16, 29, 30] is proposed to be used as the slave interface [21]. Our BIST architecture can communicate to both test bus environments provided that there exists an appropriate interface which can decode the instructions sent by the master. The core BIST architectures are independent of the test bus architecture.

In the case study, each VME board in the system is assumed to be equipped with a VME interface (such as Tundra's Universe Chip [8]). In the VME interface chip, there are 98 signals which can drive the VME bus. The maximum number of data BSCs that a control cell controls is 32 and this number decides the minimum D-TPG size. Except the bus grant and interrupt acknowledge signals, which are daisy chained, the signals are implemented as bidirectional BSCs. None of the other chips on the board connect to the backplane. Since only the VME interface chip connects to the VME bus, each backplane net is driven by only one output or bidirectional BSC per board.

In VME backplane, there are at most 21 slots. Let us consider three different system configurations: (a) a master board and a slave board; (b) a master board and nine slave boards;

and (c) the maximum configuration, a master board with 20 slave boards. In the proposed system interconnect BIST architecture, the Master BIST and the Slave BISTs are independent from the system configuration. Hence they are identical for the all three system configurations. According to the above information, pairwise shorts of the 98 VME backplane nets used in this case study can be tested by the Master BIST consisting of a 1-stage C-TPG and a 49-stage one-hot D-TPG for minimum test time — or a 4-stage C-TPG and 16-stage one-hot D-TPG for the minimum LUT area overhead — using the TPG design procedure described in Section 6 [7]. As we further investigate this case, the result of using the one-hot counter D-TPG and input reduction can be improved by the generalized input reduction TPG design procedure for counting sequence. Therefore, only a 7-stage counting sequence D-TPG can replace the 49-stage one-hot D-TPG for minimum test time while a 6-stage counting sequence D-TPG can replace the 16-stage one-hot D-TPG for the minimum LUT area overhead. Since each backplane net is driven by only one output or bidirectional BSC per board, the simplified Slave BIST, where no TPG design is needed, is used for each slave board. The size of the LUT in each board depends on the scan chain length of the board.

The test scheduling for short fault detection is also identical for all three system configurations and can be overlapped with the test scheduling for opens. The only major difference between these three system configurations in the proposed system interconnect BIST architecture is the test scheduling for open faults. If the longest boundary scan chain lengths in the system are l_a, l_b and l_c for configurations (a), (b) and (c), respectively, then the time needed to test opens in each configuration in terms of number of scan shifts are $(2 \times l_a)$, $(10 \times l_b)$ and $(21 \times l_c)$.

Compared with the decentralized BIST approach in [26] and walking enable algorithm [20], our open detection test architecture enables all backplane nets with at most one enabled driver at the enabled board and still guarantees to detection of all opens, while walking sequence is used in decentralized BIST approach and walking enable algorithm to enable only one driver on one backplane net at the enabled board. Therefore, if the decentralized BIST approach or walking enable algorithm is used for the above three configurations, $(98 \times 2 \times l_a)$, $(98 \times 10 \times l_b)$ and $(98 \times 21 \times l_c)$ scan shifts are needed. The test time can be decreased significantly by the proposed BIST architecture.

The walking enable algorithm is developed for deterministic test generation and does not consider the test issues for BIST. Conceptually, the test scheduling of our proposed system interconnect test architecture is an extension of the walking enable algorithm and using LUT for data encapsulation and the avoidance of local net conflicts are the BIST implementation of enable and disable vectors in walking enable algorithm. However, test parallelism can be achieved in our proposed framework and complicated external test equipment can be replaced by the simple system test control and the Master BIST architecture. In the decentralized BIST approach, output response analysis is handled in the master module while in our proposed test architecture, it can be done distributed in each slave board. Furthermore, BIST hardwares for decentralized BIST approach are much larger but are designed to accommodate the maximal possible system

configuration and combine board and backplane level interconnect testing, while our test architecture is relatively independent from system configuration and is developed for backplane testing assuming that each board has been tested by board level BIST such as the one described in [7]. Research on testing interconnects simultaneously at board and backplane levels in the proposed BIST architecture will be carried out.

8 Conclusion

In this paper, we have presented a new system interconnect BIST architecture and a methodology to program this architecture to generate test patterns for backplane interconnect faults, via the test bus (IEEE1149.5 or IEEE1149.1 Extension) in the backplane and boundary scan architecture of each board in the system. The test architecture includes a *Master BIST* in the master board and a *Slave BIST* in each slave board. The key components in both BISTs are a look-up table to encapsulate board level information, a C-TPG that generates test patterns for the control cells in the boundary scan chain, a D-TPG that generates test patterns for the data cells, a ROM, a controller for test synchronization and an output response analyzer. The contents of the lookup table are determined using the proposed BIST TPG design procedure, which uses the notions of *incompatibility* [6] and *conditional incompatibility* [7].

One of the key features of the proposed BIST architecture is that its design is nearly independent of the system configuration and a simple test scheduling algorithm is sufficient to test all interconnect faults in a system while avoiding multi-driver conflicts on backplane nets as well as local nets of each board. In this schedule, the shorts and stuck-at faults are tested by enabling the Master BIST, while opens are tested by all the BISTs in the system.

The proposed test schedule can be viewed as a generalization, for self-test, of the walking enable algorithm, which is developed for deterministic test generation and does not consider issues particular to BIST. The main similarity between the proposed schedule and walking-enable is that the proposed schedule sequentially enables the master BIST and each of the slave BIST. However, the proposed methodology obtains shorter test length by exploiting test parallelism. Test time is also low since, besides the test schedule, messages exchanged for test synchronization constitute the only traffic on the test bus. Furthermore, the features of the master and slave BISTs are utilized to make backplane testing insensitive to changes in system configuration, while ensuring complete fault coverage and guaranteeing no circuit damage.

The overall methodology guarantees the detection of all modeled faults using short test sequences and at low area overheads, while ensuring that no circuit damage will occur due to multi-driver conflicts. All these features of the proposed BIST, along with its ability to easily adapt to changes in a system's configuration, are demonstrated via its application to testing the backplane of three versions of a system that uses a VME backplane.

The proposed architecture can be recursively applied throughout the test hierarchy of the system. Also, the TPG framework is flexible and, whenever necessary, changes in test quality and diagnostic resolution can be easily accommodated by adding/removing constraints.

The proposed test methodology for backplane testing assumes that the faults in the inter-chip interconnect on each board have been tested by board level BIST, such as the one described in [7]. Research is being conducted on testing simultaneously board and backplane level interconnects using the proposed BIST architecture.

References

[1] J. Andrews. Roadmap for Extending the IEEE 1149.1 for Hierarchical Control of Locally-Stored, Standardized Command Set, Test Programs. In *Proceedings IEEE International Test Conference*, pages 300–306, 1994.

[2] F. W. Angelotti. Modeling for Structured System Interconnect Test. In *Proceedings IEEE International Test Conference*, pages 127–133, 1994.

[3] F. W. Angelotti, W. A. Britson, K. T. Kaliszewski, and S. M. Douskey. System Level Interconnect Test In A Tristate Environment. In *Proceedings IEEE International Test Conference*, pages 45–53, 1993.

[4] D. Bhavsar. An Architecture for Extending the IEEE Standard 1149.1 Test Access Port to System Backplanes. In *Proceedings IEEE International Test Conference*, pages 768–776, 1991.

[5] C. Champlin. Backplane Test Bus Selection Criteria. In *Proceedings IEEE International Test Conference*, page 1021, 1994.

[6] C.-A. Chen. *Test Generation and Embedding for Built-In Self-Test.* PhD thesis, Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, 1995.

[7] C.-H. Chiang and S. K. Gupta. BIST TPGs for Faults in Board Level Interconnect via Boundary Scan. In *IEEE VLSI Test Symposium*, pages 376–382, 1997.

[8] T. S. Corporation. Tundra Universe User Manual.

[9] P. Goel and M. T. McMahon. Electronic Chip-In-Place Test. In *Proceedings IEEE International Test Conference*, pages 83–90, 1982.

[10] O. F. Haberl and T. Kropf. Self Testable Boards with Standard IEEE 1149.5 Module Test and Maintenance (MTM) Bus Interface. In *Proceedings European Design and Test Conference*, pages 220–225, 1994.

[11] A. Hassan, V. K. Agarwal, B. Nadeau-Dostie, and J. Rajski. BIST of PCB Interconnects Using Boundary-Scan Architecture. *IEEE Transactions on CAD*, 11(10):1278–1288, Oct. 1992.

[12] W.-C. Her, L.-M. Jin, and Y. El-Ziq. An ATPG Driver Selection Algorithm for Interconnect Test with Boundary-Scan. In *Proceedings IEEE International Test Conference*, pages 382–388, 1992.

[13] J.-H. Hong, C.-H. Tsai, and C.-W. Wu. Hierarchical testing Using the IEEE Std 1149.5 Module Test and Maintenance Slave Interface Module. In *Proceedings Asian Test Conference*, pages 50–55, 1996.

[14] IEEE. *IEEE Standard Test Access Port and Boundary-Scan Architecture.* IEEE, 1993.

[15] IEEE. *IEEE Standard Module Test and Maintenance (MTM) Bus Protocol.* IEEE, 1995.

[16] T. I. Inc. SN74ABT8996, Addressable Scan Ports, Multidrop-addressable IEEE STD 1149.1 TAP Transceivers, Aug 1994. TI Web Site, SCBS489.

[17] N. Jarwala and C. W. Yau. A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects. In *Proceedings IEEE International Test Conference*, pages 63–70, 1989.

[18] F. D. Jong, J. S. Matos, and J. M. Ferreira. Boundary Scan Test, Test Methodology, and Fault Modeling. *Journal of Electronic Testing: Theory and Applications*, 2:77–88, Mar. 1991.

[19] W. H. Kautz. Testing for Faults in Wiring Networks. *IEEE Transactions on Computers*, c-23(4):358–363, Apr. 1974.

[20] W. Ke. Backplane Interconnect Test In A Boundary-Scan Environment. In *Proceedings IEEE International Test Conference*, pages 717–724, 1996.

[21] W. Ke, D. Le, and N. Jarwala. A Secure Data Transmission Scheme for 1149.1 Backplane Test Bus. In *Proceedings IEEE International Test Conference*, pages 789–796, 1995.

[22] D. Landis, C. Hudson, and P. McHugh. Applications of the IEEE P1149.5 Module Test and Maintenance Bus. In *Proceedings IEEE International Test Conference*, pages 984–992, 1992.

[23] D. Le and W. Ke. An Evaluation of System Test Architectures, 1995. AT&T Bell Labs Memo.

[24] C. Poirier. IEEE 1149.5 To 1149.1 Data and Protocol Conversion. In *Proceedings IEEE International Test Conference*, pages 527–535, 1993.

[25] W. Shi and W. K. Fuchs. Optimal Interconnect Diagnosis of Wiring Networks. *IEEE Transactions on VLSI Systems*, 3(3):430–436, Sept. 1995.

[26] C. Su, S.-J. Jou, and Y.-T. Ting. Decentralized BIST for 1149.1 and 1149.5 Based Interconnects. In *Proceedings European Design and Test Conference*, 1996.

[27] VITA. *VME64 Extension Draft Standard.* VITA, 1996.

[28] P. T. Wagner. Interconnect Testing with Boundary Scan. In *Proceedings IEEE International Test Conference*, pages 52–57, 1987.

[29] L. Whetsel. A Proposed Method of Accessing 1149.1 in a Backplane Environment. In *Proceedings IEEE International Test Conference*, pages 206–216, 1992.

[30] L. Whetsel. Hierarchically Accessing 1149.1 Applications in a System Environment. In *Proceedings IEEE International Test Conference*, pages 517–526, 1993.

[31] C. W. Yau and N. Jarwala. A Unified Theory for Designing Optimal Test Generation and Diagnosis Algorithms for Board Interconnects. In *Proceedings IEEE International Test Conference*, pages 318–324, 1989.