BIST TPG for Faults in System Backplanes [*]

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

Abstract

A built-in self-test (BIST) methodology to test system backplanes by using BIST functionality in each of its constituent boards is presented. Since the configurations of systems changes frequently, at the system level, the proposed methodology employs a simple test schedule *which can be easily changed whenever the system configuration is changed. Since the boards used in such systems are designed for use in a wide variety of systems, the proposed methodology defines the* test objectives *to be achieved by a board's BIST circuit in terms of the board's edge pin connections, independent of the configurations of the systems in which the board may be used. It is shown that the combination of the proposed test schedule and the availability, on each board in the system, of any BIST circuit that satisfies the proposed test objectives, guarantees safe testing of faults in backplanes.*

A programmable test architecture *and an algorithm to program the architecture to obtain BIST that satisfies the test objectives is also presented. Finally, the applicability and effectiveness of the methodology is demonstrated via its application to multiple configurations of an example system that uses a VME backplane.*

1. Introduction

Large industrial systems are typically implemented using sub-systems, which are implemented on separate boards and integrated using a backplane. By providing standardized interconnects between boards, backplanes make it possible to build systems that can be constantly updated and/or grown using a wide variety of off-the-shelf boards. Since the backplane is the main communication link of such systems, its error free operation is crucial to the system's operability. Besides the faults caused over time by the hostile environments in which many such systems operate, faults can be introduced whenever a board is removed, replaced, or added. Such faults can be caused due to reasons such as loose or improper connections, or bent connector pins. Hence, built-in self-test (BIST) circuitry for backplane testing will find frequent use.

The objective of *backplane testing* is to test the interconnect structure of the *system bus* which resides in the backplane and the interconnect structure of the *edge pin connectors* which connect each board to the system bus. During backplane testing, all the *local nets* on each board are assumed to be well tested. Similar to how the IEEE 1149.1 boundary scan architecture (BSA) [12] test access port (TAP) port helps interconnect testing between chips at board level, we assume and exploit 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 the backplane level. This test bus can be either 1149.1 multi-drop extension [14, 26, 27] or IEEE 1149.5 module test and maintenance (MTM) bus [13].

Despite bearing significant similarities with testing of inter-chip interconnects via boundary scan at board level, backplane testing poses several new and unique requirements. Firstly, while at the board level, during BIST, all the boundary scan cells (BSCs) can be configured as a single scan chain under a single controller, at the backplane level, the cells in each board must form a separate chain, where each chain is controlled by a distinct on-board controller. Typically, the on-board controllers are controlled by a single master controller via the test bus. The presence of multiple chains and two-level hierarchical control necessitates coordination between the on-board controllers, altering the manner in which multi-driver conflicts are avoided and high fault coverage is obtained. More importantly, unlike the configuration of a board, which remains relatively fixed, the configuration of backplane interconnect changes continually as boards are removed, replaced, or added. Furthermore, the BIST circuitry in each board should be assumed to be designed by a different designer, without any a-priori knowledge of the exact system configuration. A new framework is required to develop BIST for backplanes that can take into account all these differences.

Most previous work in testing interconnects focused on the development of deterministic tests for interconnect between chips at the board level [7, 9, 15, 17, 22, 25, 28]. As pointed out above, the extension of these board level methods to backplane testing is non-trivial. In the interconnect test phase of [8], only board level interconnects are tested. Once the global knowledge of the system configuration and net lists of each constituent board are available, test methods used at board level are extended to test system level interconnect in [1, 2], where a dominant short fault model is

[*]This research was funded by Lucent Technologies and NSF CAREER Award no. MIP-9502300

also defined to address shorts between outputs of components employing different technologies. The walking enable algorithm [18] uses disable and enable vectors that enable one board at a time to make backplane testing independent of changes in system configuration. The decentralized BIST [23] addresses interconnect testing at both board and backplane levels. Even though the BIST circuitry is decentralized, the BIST circuitry on each board is required to be an exact copy of their BIST architecture. This limits the applicability of the method in scenarios where off-the-shelf boards from different manufactures are used.

In this paper, we propose a framework to design BIST that uses the walking enable approach [18] in its backplane level *test schedule*, in which one board is activated at a time, but employs a more compact set of patterns to test the faults in the activated board's *edge pin connector nets*, i.e. the interconnection between the edge pin connector boundary scan cells (PCCs) and each pin of the board. By identifying the required test information and defining test objectives for BIST on each board in the system, the proposed methodology achieves near independence from system configuration and complete independence from how the BIST circuitry in each board achieves the specified test objectives, and guarantees the achievement of complete fault coverage in short test time and the avoidance of any multi-driver conflicts during backplane testing.

A BIST architecture that satisfies the requirements of the proposed methodology is presented. A test pattern generator (TPG) design procedure, based on the generalized input reduction techniques [5] where the notions of *incompatibility* [4] and *conditional incompatibility* [5] are used to describe the test objectives that a TPG must satisfy, can be used to program the test architecture to achieve safe testing and complete fault coverage in short time and low area overhead. Finally, all the features of the proposed backplane test methodology, along with its ability to easily adapt to changes in a system's configuration, are demonstrated via its application to test the backplanes of three versions of a system that uses a VME backplane.

2. Background

2.1. Backplane Nets

According to the characteristics of the drivers, backplane nets can be classified as 3-state nets, bidirectional nets, simple nets, wired-AND nets and wired-OR nets. A 3-state net is driven by one or more 3-state drivers, each of which has a data cell and a corresponding control cell. A 3-state driver is said to be *enabled (disabled)* when the enable (disable) value is assigned to the corresponding control cell. A bidirectional net is driven by at least one bidirectional driver which acts like a receiver when disabled (by the assignment of its disable value to the corresponding control cell) and becomes a 3-state driver with a receiver when enabled. It can be shown that, for our problem, bidirectional nets and 3-state nets are equivalent. Hence, in the following, we only consider 3-state nets, but our methodology can be applied to backplanes with bidirectional nets. A simple net is driven by a single 2-state driver, while a wired-AND(OR) net is driven by multiple drivers of a suitable design.

A 3-state net is said to be *disabled* when all its drivers are simultaneously disabled. To simplify the discussion, we assume that a disabled 3-state net holds a logic-1 value; however, our results can also be applied to nets that hold a logic-0. We also assume that the driving strength of an enabled 3-state driver is significantly higher than that of a disabled one. We say that a simple net is *disabled*, when a logic-1 value is assigned to its driver, so that it holds the same value as that of a disabled 3-state net. A driver of a wired-AND(OR) net is said to be *disabled* when a logic-1(0) is assigned. A wired-AND(OR) net is said to be *disabled* when all of its drivers are disabled simultaneously. Note that the resulting value of a wired-AND(OR) net is the same as that obtained by the evaluation of a logic AND(OR) gate whose inputs are the drivers of the wire net.

In the following, we consider the faults associated with the *edge pin connector nets* (i.e. the connections between the PCCs of a board and the pins of its edge pin connectors), *edge pin connectors*, and the *backplane nets* (i.e. the interconnections in the backplane chassis). We assume that any two driver pins in the edge pin connector of a board always constitute distinct backplane nets.

2.2. Conflict-Avoidance Constraints

If boundary scan is used to test faults in backplane nets, then the application of a test pattern that enables multiple drivers driving opposite values on a given net can cause circuit damage by causing excessive current flow. While the application of such patterns can be avoided easily if deterministically generated test patterns are applied under external control, BIST TPG must be carefully designed to ensure that the test sequence generated is constrained in such a manner that it does not apply any illegal patterns. These constraints are referred to as *conflict-avoidance constraints* (referred to as essential constraints in [5]).

2.3. 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, for all types of nets, except wired-OR, that a logic-1 value is captured by a receiver when one or more opens disconnect it from the drivers. For wired-OR nets, when disconnected due to opens, the receivers are assumed to capture a logic-0.

The technology and driving strengths of the output drivers involved in a short affect the behavior of a short fault [2, 16, 10]. The behavior can be either deterministic or non-deterministic. The deterministic behaviors due to shorts between two backplane nets can be further characterized as 0(1)-dominant and net-dominant faults [2], where 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.

In the following, the dominant short fault model captures shorts in backplane interconnects, while the open and stuck-at fault models cover most of the other defects that are usually seen at backplane level, such as bent connector pins and loose connection. Only *pairwise shorts* are considered because multiple net shorts are automatically detected if pairwise shorts are detected.

2.4. Test Conditions

Test conditions for a fault characterize all possible tests that can detect the fault. They are used by our BIST design methodology to reduce the TPG size and/or test 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 the net with a logic-0(1) value to detect the stuck-at-1(0) fault on the net.

Any set of test patterns that detects opens at pins of each driver of a backplane net also guarantees the detection of all other *detectable open faults* (defined in [20]) in the net. To detect such an open on a backplane net, a test must: (i) enable one of its drivers and disable all its other drivers; (ii) apply a logic-0 (logic-1 for wired-OR nets) to the enabled driver; (iii) check the response captured at *all* its receivers.

Test conditions for pairwise shorts between two backplane nets a and b, where either a or b is a 3-state or wired-AND(OR) net, are either (i) disable one net, say a, and enable the other net, say b, and apply a value that is opposite to the value of the disabled net a, or (ii) enable both nets and drive both possible sets of opposite values on them, i.e. '0' on net a and '1' on net b **and** vice versa, to detect the dominant shorts. If nets a and b are both simple nets, test condition (ii) above must be satisfied.

3. Methodology for Backplane Testing

The first characteristic of the proposed BIST methodology is its ability to easily adapt to changes in the system configuration, without requiring large amounts of information about the constituent boards and their BIST features. This is accomplished by the use of a simple *test schedule* that relies on the availability of multi-mode BIST circuitry on each board. Secondly, the requirements of the multi-mode BIST in each board is specified in terms of the information that is normally available during board design. Thirdly, the methodology does not require any specific BIST implementation; any BIST circuitry that satisfies the requirements is acceptable. This is especially attractive, since typically systems use boards from multiple vendors; in such a context, the availability of desired BIST functionality is significantly more realistic to assume than the availability of a specific BIST circuitry.

3.1. Test Scheduling

We assume that the test bus and system bus architectures belong to multi-drop architecture where a master controls several slaves in the system. Further, we assume that we can incorporate the test bus master into the same board as the system master. The main task of the test bus master is to execute a *test schedule* comprised of multiple *test sessions*. In each test session, the master is responsible for appropriately configuring the BIST circuitry of each board in the system, allowing the BIST circuits to perform self-test, and finally collecting and checking the signatures. At the beginning of each session, the master also communicates the expected value of the signature to each slave.

The BIST circuitry in each board can be configured in two main modes: active and inactive (described in detail in the following). In the active mode, a board's BIST circuitry applies a sequence of tests to, and captures test responses from, its PCCs; in the inactive mode, it applies a fixed pattern but captures response as in the active mode. Since the boundary scan chain length may be different for each board in the system, each session begins with the master determining the number of shift clocks required to apply tests in that session. This number is used for *test synchronization* — the BIST circuit in board that is activated during the session uses this number to apply test patterns and the BIST circuits in both the active and inactive boards during the session use this number to determine when they should capture responses.

The walking enable strategy [18] is used as the overall test schedule, i.e. in each session the BIST circuitry of one board in the system is configured in the active mode while those of all other boards in the system are configured in the inactive mode. The overall test schedule is comprised of sessions in which each board is activated in turn, and requires only the knowledge of the number of boards in the system and their locations (i.e. the backplane slot to which they are connected). The simplicity of the strategy allows easy update of the backplane self-test after each system update.

3.2. Specifications of Multi-mode BIST

The requirements of the BIST circuitry on each board will now be specified in such a way that any BIST circuitry that satisfies them, when used along with the above schedule, will cover all faults in and among the nets in any given backplane and guarantee the avoidance of multi-driver conflicts.

Though the proposed framework makes use of BIST circuitry in all boards, it relies heavily on the BIST circuitry in the master board. The reliance on the master is justified due to several reasons. Firstly, the master is always present in the system, even in a system with minimal configuration. Secondly, since, functionally, the system design is intricately related to the design of the master board, it can be assumed that the system designer has significant say in the design of the master board (at least greater say than in the design of off-the-shelf boards used as slaves). Finally, since a system contains a single master but multiple slaves, it is more economical to design more complicated BIST circuitry in the master, especially if that helps simplify the design of BIST in all the slave boards. Clearly, a methodology that imposes simpler requirements on the BISTs of slave boards will gain wider acceptance among the manufacturers of off-the-shelf slave boards.

Due to above reasons, in its active mode, the BIST in the master board is required to test all the faults that it can. For most backplane bus standards, such as VME, since most backplane nets are connected to the master, most faults can be tested by the master. Under some conditions, this can simplify the design of BIST circuitry in each slave board to the point that it requires only the inactive mode — greatly decreasing its complexity. Also, the BIST in master board functions as the master test controller that controls the modes of the BIST circuitry in slave boards and implements the overall test schedule. Next, we describe the requirements that need to be satisfied by the BIST circuitry in various boards in their active and inactive modes.

3.3. Inactive Mode

In the inactive mode, the drivers of all 3-state, simple and wired-AND(OR) backplane nets on the board are disabled. Hence, in this mode, the board applies a single fixed pattern to its edge pin connector nets. In addition, all its local nets are disabled. The BIST is required to capture the response at all the receiver PCCs, at specified intervals, and to compress the response.

3.4. Active Mode

During the active mode, the BIST TPG of a board must generate patterns that achieve the test objectives of the board. In addition, it should capture response at the receiver PCCs and hold each board level local net in its disabled state.

As discussed above, the test objectives are defined more aggressively for the master board. For this purpose, in addition to the backplane net classification given in Section 2.1, the backplane nets are further classified according to the location of their drivers as: (i) *type-m nets*, at least one of whose drivers is on the master board, and (ii) *type-s nets*, none of whose drivers is on the master board. (Note that each type-m net is, by definition, connected to at least one driver PCC of the master board.) Next, the test objectives for the master and each slave are defined in terms of the faults in their edge pin connector nets that they must cover.

3.4.1. Active Mode of the Master Board

The set of tests generated during the active mode by the BIST in the master board must guarantee the detection of (i) stuck-at faults in each type-m backplane net, (ii) open faults at the output pins of all driver PCCs on the board, and (iii) pairwise shorts between all pairs of type-m backplane nets. In addition, each test must ensure the avoidance of all multi-driver conflicts in the board's edge pin connector nets. Since the overall test schedule ensures that only a single board BIST is activated during any given session, this condition only imposes requirements on the board's driver PCCs that drive a given net.

3.4.2. Active Mode of a Slave Board

The test objectives of the slave board can be simplified by eliminating the faults that have been detected during the active mode of the master board. Hence, during the active mode of a slave board it should generate a set of patterns that guarantees the detection of (i) stuck-at faults in each type-s backplane net that is driven by one or more of its PCCs, (ii) open faults at the output pins of all driver PCCs on the board, and (iii) pairwise shorts between all pairs of type-s backplane nets that have at least one driver on the board. In addition, each pattern must ensure the avoidance of multi-driver conflicts on the board's edge pin connector nets.

Note that all the above requirements are defined in terms of the information about the board's own PCCs and the knowledge of which of its PCCs belong to certain type-m nets. The former is also required for functional design of the board and hence available; the latter is also known a-priori for most nets in typical backplane standards. If a backplane net driven by one or more of a board's PCCs cannot be definitely classified as a type-m net, it can be treated as type-s. This will still guarantee safe and complete testing; however, the test length would be higher.

3.5. Completeness of the Proposed Strategy

In this section, we will first show how the BIST methodology proposed above is guaranteed to detect all faults in any given backplane.

The system backplane interconnect faults to be considered include: (a) stuck-at faults at edge pin connector nets on each board; (b) open faults at edge pin connector nets on each board; (c) pairwise shorts between edge pin connector nets on each board; (d) stuck-at faults in backplane nets; (e) open faults in backplane nets, and (f) pairwise shorts between backplane nets. Since a stuck-at fault is assumed to affect the faulty net completely, faults in the category (a) are equivalent to those in (d). Faults in the category (c) are equivalent to those in (f); therefore, no pairwise shorts between an edge pin connector net and a backplane net need to be considered. Pairwise shorts between backplane nets (category (f)) can be further classified into: (f-1) pairwise shorts between two backplane nets which are driven by drivers on the same board, and (f-2) those between two backplane nets which are driven by drivers on the different boards. The test objectives for each board explicitly require coverage of faults in categories (d) and (f-1). The faults in category (f-2) are implicitly tested during the test schedule because it is guaranteed that, sometime during testing, one net involved in such a pairwise short will be on an active board, while both logic-1 and logic-0 values will be applied to test stuck-at faults, and the other will be on an inactive board holding a steady disabled value.

As mentioned in the test conditions for opens, once the open faults at pins of all the drivers of a net are detected, all detectable opens at the net are detected. Therefore faults in category (e) dominate those in category (b) which are explicitly covered by the test objectives of each board. Hence, in the test schedule, the combination of the test objectives of the master board and those of all the slave boards in the system covers all the interconnect faults in the system backplane. Therefore the complete fault coverage can be guaranteed by the proposed methodology.

Up to this point we have shown how the proposed backplane test methodology is independent from system configuration and that it guarantees complete coverage of system backplane interconnect faults. We now address how the proposed methodology avoids multi-driver conflicts on both local nets and backplane nets. Since the local nets of each board are assumed to be well tested and remain disabled at all times during backplane testing, the avoidance of multi-driver conflicts on local nets is guaranteed. For backplane nets, the test schedule guarantees that, during any session, drivers on all but the active board are disabled. Furthermore, it is assumed that any two driver pins on a board always constitute distinct nets in the backplane. Hence, the multi-driver conflicts can be avoided by merely satisfying the conflict-avoidance constraints described as a part of the test objectives of each board.

The above test objectives can be used by the TPG design procedure described in Section 5 to program the test architecture described next.

4. Proposed BIST Architecture

In this section, a system interconnect test architecture which can be used to implement the proposed system backplane test methodology is proposed. The proposed test architecture consists of: (i) a *Master BIST* on the master board, and (ii) a *Slave BIST* on each slave board in the system. The architecture of both the Master BIST and Slave BIST is as shown in Figure 1. Typically each Slave BIST can be further simplified as the test objectives are reduced due to the absence of certain types of drivers on the slave board. The active and inactive modes are provided, and the independence from system configuration, test synchronization, and the avoidance of multi-driver conflicts on local nets can be achieved by the use of hardware features in the proposed system test architecture.

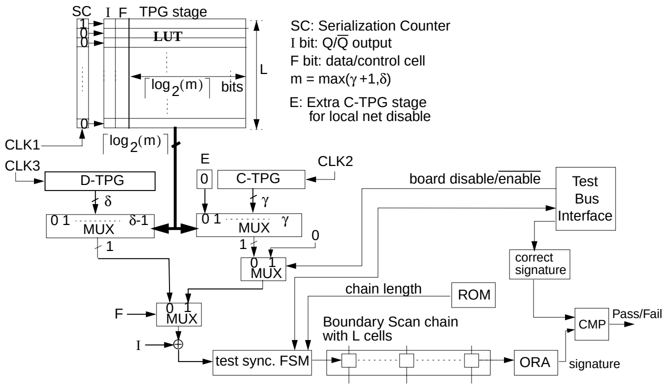
SC
I
F
TPG stage
1
0
0
0
LUT
$\lceil \log_2(m) \rceil$ bits
L
SC: Serialization Counter
I bit: Q/$\overline{\text{Q}}$ output
F bit: data/control cell
m = max(γ+1,δ)
E: Extra C-TPG stage for local net disable
CLK1
$\lceil \log_2(m) \rceil$
CLK3
D-TPG
δ
0 1 δ-1
MUX
1
E
0
C-TPG
CLK2
γ
0 1 γ
MUX
1
0
0 1
MUX
board disable/$\overline{\text{enable}}$
Test Bus Interface
correct signature
chain length
ROM
F
0 1
MUX
I
Boundary Scan chain with L cells
test sync. FSM
ORA
signature
CMP
Pass/Fail

Figure 1. Master (Slave) BIST architecture.

As shown in Figure 1, the proposed BIST architecture contains a look-up table (LUT), two test pattern generators (a C-TPG that generates tests for the control cells in the boundary scan chain and a D-TPG that generates tests for the data cells), a read-only memory (ROM), a test synchronization finite state machine (TS-FSM), an output response analyzer (ORA) and a test bus interface.

4.1. 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 board's scan chain and has a total of $(\lceil \log_2 m \rceil + 2)$ bits (where γ is the C-TPG size, δ is the D-TPG size, and $m = max(\gamma, \delta)$) and contains the following information: (i) the bit F specifies 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) the next $\lceil \log_2 m \rceil$ bits are then used to select the appropriate stage of the selected TPG, and (iii) the 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. PCC: These cells are connected to the backplane nets. For PCCs, 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 the positive (negative) output of the appropriate C-TPG stage if it is enable high (low). The assignment to the appropriate TPG stage is determined by the TPG design procedure described later in Section 5.
2. Non-PCC: These cells are connected to the local nets on the board. Each non-PCC entry of the LUT is designed in such a manner that all local nets are disabled to avoid conflicts by their assignment to an extra stage

of the C-TPG that always outputs a '0' value which generates the disable value for each non-PCC control cell, by using the bit I of the LUT.

4.2. BIST Operation

The signature register, the ORA and two TPGs form the core of the BIST architecture. The signature register contains the correct signature of the expected output response for ORA, which is obtained from the master at the beginning of each test session. 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 board is activated by the system test controller via the test bus interface, the BIST architecture generates appropriate test patterns and synchronizes with other boards. The test patterns are then scanned into the boundary scan chain and applied to the backplane nets (while avoiding conflicts on local nets); after test application, the response is captured. Finally, the BIST circuit compresses the output response into a signature.

Since the boundary scan chain length may be different for each board in a system, test synchronization is achieved by using appropriate circuitry in the system test controller on the master board and the TS-FSM on each board. Based on the boundary scan chain length (L) stored in each on-board ROM, which is 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 TAP controller) after the desired test patterns are shifted into the scan chain. This helps synchronize the application of a pattern by the active board and the capture of the responses by all boards.

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 held constant for $(L_{max} - L)$ for test synchronization. In the first L clocks 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 (slave) board is in-activated, Master (Slave) BIST disables all the PCCs control cells but still collects responses from the backplane nets and computes a signature.

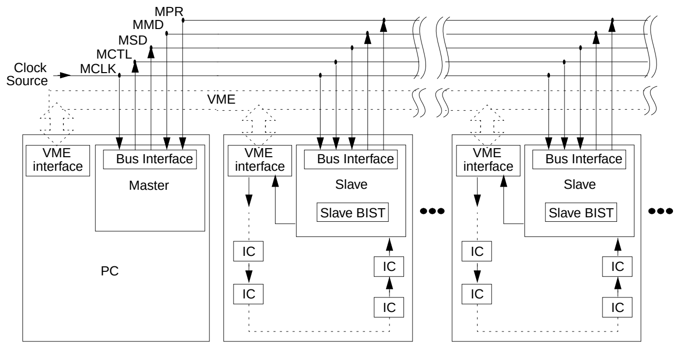

Figure 2. 1149.5 MTM/1149.1 Ext. in a VME backplane.

5. BIST TPG Design Procedure

The key objective of the TPG design is to assign all BSCs to corresponding TPGs with minimum number of stages, subject to conflict-avoidance constraints and constraints which ensure that the test objectives specified by the methodology are achieved. That is, the BIST TPG design procedure is used to program the LUT of the proposed system backplane test architecture. The constraints can be written using the notions of *incompatibility* [4] and *conditional incompatibility* [5]. Therefore, the TPG design procedure can be written as follows:

$$\begin{cases} \text{Minimize } |\text{C-TPG}| \times |\text{D-TPG}|, \\ \text{subject to (i) conflict-avoidance constraints;} \\ \qquad\quad \text{(ii) constraints for achieving test objectives,} \end{cases} \quad (1)$$

where $|$TPG$|$ denotes the length of the sequence generated by a TPG. The branch-and-bound algorithm proposed in [5] can also be used to solve the problem specified by (1) to obtain a TPG design.

If the objective of TPG design is the minimization of the LUT size, the above formulation can be modified to minimize max($|$C-TPG$|$, $|$D-TPG$|$), instead of minimizing their product.

6. 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. With increasing demand for system testing, 1149.5 MTM bus is being proposed to be a part of the new VME64 Extensions [24].

Either the 1149.5 MTM bus or 1149.1 Extension can be used as the test bus for backplane testing as shown in Figure 2. In the MTM test bus environment, a system test controller (which itself can be controlled by a PC) is assumed to be at the master board; a test bus slave interface [8, 11, 21] which can communicate with the BSA on each board, is also assumed to be present on each slave board. In the 1149.1 Extension environment, a BSM [3] controlled by a PC resides in the master board and TI ASP [14, 26, 26] can be used as the test bus slave interface [19]. Our BIST architecture can communicate with both test bus environments, provided that there exists an appropriate interface which can decode the instructions sent by the master, and is otherwise independent of the test bus architecture.

In the following case study, each VME board in the system is assumed to be equipped with a VME interface (such as Tundra's Universe Chip [6]). The VME interface chip connects to the VME backplane bus which consists of 104 signals: one input signal, one output signal, 5 pairs of daisy chain input/output signals (4 pairs for bus grant signals and one pair for interrupt acknowledge signal) and 92 bidirectional signals. These signals form 92 type-m bidirectional nets, 5 daisy chain connections (5 type-m 3-state nets on the master board and 5 type-s 3-state nets on each slave board) in the backplane and two untestable nets (the input-only and the output-only signals), which are ignored during testing. The maximum number of data BSCs that a control cell controls in the 92 type-m bidirectional nets is 32 and this number decides the minimum D-TPG size. Except for a few buffer chips between the VME interface chip and the VME backplane, none of the other chips on the board connect to the backplane; therefore, each backplane net is driven by only one output or bidirectional BSC per board.

In a VME backplane, there are at most 21 slots. Let us consider three different system configurations: (a) the master board and one slave board; (b) the master board and 9 slave boards; and (c) the maximum configuration, the master board with 20 slave boards. In the proposed system interconnect BIST methodology, the Master BIST and the Slave BISTs are independent from the system configuration. Hence they are identical for the all three system configurations. However, because of the different test objectives, the Master BIST is required to cover faults on and between (92+5 = 97) type-m nets, while each Slave BIST is required to cover faults on its 5 type-s nets as described in the test objectives of the proposed methodology. The BIST TPG design procedure then generates Master BIST designs consisting of either (a) a 1-stage C-TPG and a 49-stage one-hot D-TPG (minimum test time), or (b) a 4-stage C-TPG and a 16-stage one-hot D-TPG (minimum LUT area overhead). It generates a Slave BIST design consisting of a 1-stage C-TPG and a 3-stage one-hot D-TPG that provides the minimum test time as well as LUT area overhead. 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 to design a counting sequence D-TPG [5]. For the Master BIST, a counting sequence D-TPG with only 6 stages can replace the 49-stage one-hot D-TPG for minimum test time while a 5-stage counting sequence D-TPG can replace the 16-stage one-hot D-TPG for the minimum LUT area overhead. No further improvement can be obtained for Slave BISTs by using counting sequence.

The only major difference between these three system configurations in the proposed system interconnect BIST architecture is the total test time. Let us consider one of the BIST architectures above, which has minimum test time for the Master (Slave) BIST using a 1(1)-stage C-TPG and a 6(3)-stage counting sequence D-TPG, which generates $1 \times 6(1 \times 3)$ test vectors. 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 total test time in each configuration in terms of number of test clocks (considering number of clocks for scan shifts, one more clock for applying the test vector and capturing the response simultaneously, and one additional cycle to shift out the final response) are $(6(l_a+1)+3(l_a+1)+l_a \approx 10l_a)$, $(6(l_b+1)+3(l_b+1)\times 9+l_b \approx 34l_b)$, and $(6(l_c+1)+3(l_c+1)\times 20+l_c \approx 67l_c)$.

The decentralized BIST approach in [23] as well as the walking enable algorithm [18] (try to) enable only one backplane net at a time to avoid conflicts. Therefore, if the decentralized BIST approach or walking enable algorithm (the intra-board counting version) is used for the above three configurations, at least, $(7(l_a+1)\times 2+l_a \approx 15l_a)$, $(7(l_b+1)\times 10+l_b \approx 71l_b)$, and $(7(l_c+1)\times 21+l_c \approx 148l_c)$ (where $7 = \lceil log_2(97+2)\rceil$) test clocks are needed. The proposed TPG achieves the same objectives at a lower test length because it can avoid conflicts while enabling multiple backplane nets. Furthermore, faults detected during the active mode of the master board are eliminated from the test objectives of the slaves to simplify Slave BIST circuitry and further decrease the test time.

The walking enable algorithm [18] was developed for deterministic test generation and did not consider the test issues for BIST. While our test methodology adopts its walking approach as the test schedule at the higher level, it achieves test parallelism at the board level and replaces complicated external test equipments. In the decentralized BIST approach, output response analysis is handled in the master module while in our proposed test architecture, this is done in a distributed fashion in each slave board. Finally, the proposed BIST architecture is truly independent of the system size. As has been discussed earlier, the need to replicate identical BIST TPGs compromises the applicability of the decentralized BIST approach [23] in a multi-vendor context.

7. Conclusion

In this paper, we have presented a new BIST methodology for testing faults in and between the nets in system backplane, pins of the board's edge pin connectors, and edge pin connector boundary scan nets on the boards.

The proposed test methodology assumes that each board in the system contains BIST circuitry that can be configured in *active* or *inactive* mode. In the active mode, the BIST circuitry on a board applies a set of test patterns to the board's driver PCCs that achieve specified test objectives. It also captures and compresses the response to each test. In the inactive mode, a specified fixed value is applied to each driver PCC on the board and the response is collected and compressed. To ensure that the BIST circuitry is useful in any system in which the board is used, the test objectives are specified in terms of the board's PCC nets, independent of the configurations of the systems in which the board will be used.

To simplify the test objectives (and hence the BIST circuitry) for the large number of slave boards used in a system, the BIST in the (single) master board in the system, in its active mode, is required to generate tests to detect all faults in the backplane which can be detected by the master alone.

At the system level, to ensure that BIST can be easily adapted to changes in system configuration, a simple *test schedule* is adopted. The test schedule comprises of a set of sessions, where in each session, the BIST circuitry of one board is configured in its active mode while those of all other boards are in-activated. While this test schedule can be viewed as a generalization of the enable strategy proposed in [18], the proposed methodology achieves faster testing by using a more efficient test set in the enable mode.

It has been shown that the combination of this test schedule and the availability, in each board, of *any BIST circuitry that satisfies the above test objectives* guarantees safe testing of all faults in the backplane and edge pin connector nets. Note that, unlike the decentralized BIST proposed in [23], the proposed methodology does not require the BIST circuitry on each board to be identical. We believe that this makes the proposed methodology significantly easier to apply to practical systems, which employ boards designed by a number of different vendors.

A programmable test architecture and procedures to program this test architecture to obtain BIST circuitry for the master as well as slave boards are also presented. The applicability and efficiency of the proposed methodology is demonstrated via its application to test the VME backplane in three configurations of an example system.

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

Research is currently being conducted on testing simultaneously the board and backplane level interconnects, and on testing backplanes of systems that contain some boards which do not have the BIST capability required by the above methodology.

References

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

[2] 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.

[3] AT&T. *AT&T Boundary-Scan Master Manual*. AT&T, 1993.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

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

[15] 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.

[16] 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.

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

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

[19] 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.

[20] J.-C. Lien and M. A. Breuer. Maximal Diagnosis For Wiring Networks. In *Proceedings IEEE International Test Conference*, pages 96–105, 1991.

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

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

[23] 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.

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

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

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

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

[28] 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.