A HIERARCHICAL, DESIGN FOR TESTABILITY (DFT) METHODOLOGY FOR THE RAPID PROTOTYPING OF APPLICATION SPECIFIC SIGNAL PROCESSORS (RASSP)

Richard Sedmak, Self-Test Services
John Evans, Lockheed Martin Advanced Technology Laboratories

Abstract

This paper describes a design for testability process, which is highly automated, hierarchical, and spans the entire life cycle. The process was developed for the DoD's RASSP Program and contributes significantly to the RASSP goals of 4x improvement in cycle time, design quality, and life cycle costs.

**Keywords - ** BIST, design for testability, rapid prototyping, RASSP, signal processors

1.0 Introduction

The Rapid Prototyping of Application Specific Signal Processors (RASSP) program is a DoD sponsored program to develop processes for the design, manufacturing, and fielding of signal processor based systems (the processes being referred to in this paper collectively as a "methodology") that will achieve four times (4x) improvements in cycle time, design quality, and life cycle costs. As part of that methodology, a design for testability (DFT) process or methodology has been developed that supports and facilitates the achievement of the RASSP goals and strives for testable systems at each phase of the life cycle and at each level of the packaging hierarchy. This paper describes the key features of the RASSP DFT Methodology.

The DFT methodology provides designers and test engineers a process for introducing test requirements and strategies early in the design cycle. It also provides procedures for test strategy selection and DFT implementation at the chip, MCM, board, and system levels. DFT strategy selection and implementation activities operate on successive refinements of the design, starting from the conceptual (system specification) level down to the physical implementation level. The impact of test decisions on meeting requirements, as well as the tracking of compliance to test requirements, is analyzed at all levels of the design and all phases of the life cycle. While the methodology is not driven by specific test tools, knowledge of the existing tools has been used to ensure the practicality of the methodology.

2.0 The Life Cycle Test Problem

In the RASSP DFT methodology, the test problem, along with DFT solutions, are addressed from a life cycle standpoint, as described in the sections that follow.

2.1 The Design Test Problem

Increased emphasis on shortening the time to first customer delivery, improving quality, managing the complexity of designs more effectively, and reducing overall life cycle costs, drives designers to find more efficient ways of performing design verification and prototype test. During integration of hardware and software, unanticipated iterations between hardware and software arise which can result n significant schedule delays.

Detecting and isolating these events/ problems can add weeks to months to tight schedules. At the same time, accessibility problems exist on systems with MCM's or conformally coated boards, after the unit enclosure is sealed, or when the UUT is placed in an environmental test chamber. The net result is hardware/software debug and integration problems are outpacing solutions.

2.2 The Production Test Problem

Testing high performance integrated circuits and high density boards o r MCMs is a difficult task because of their performance, complexity, and physical and/or electrical constraints. System clock frequencies are surpassing the capabilities of Automatic Test Equipment. Traditional test methods such as in-circuit probing cannot be applied to new technologies such as Ball Grid Arrays, Surface Mounted parts and MCMs unless test points are provided or new methodologies, such as boundary scan, are applied.

2.3 The Field Test Problem

The same problems outlined above for design and production test also apply to field test. The cost of procuring and maintaining test equipment, as well as the cost to develop and maintain TPSs, is outstripping budgets. It is no longer practical to require highly skilled troubleshooters to perform diagnosis of system problems. The cost of stocking spares is driven by unit cost and false alarms. Failures identified in the field which cannot be duplicated at the depot (i.e., LRU was good and/or system backplane was out of spec.) must be minimized.

The net result is that users are requiring built-in-test and diagnostics be supplied with electronic equipment as a requirement on par with size, weight and power limitations.

3.0 Significant Concepts in the DFT Methodology

There are several key concepts employed in the DFT methodology, which are addressed below.

3 .1 A Broadened Scope for the Concept of Testing and DFT

In the DFT Methodology, the term "testing is expanded to encompass any process which detects, isolates, and corrects anomalies (e.g., design flaws, manufacturing defects, field defects) in any phase of the life cycle. The definition of testing is expanded so that DFT provides the greatest contribution to achieving the RASSP 4x improvement goals in cycle time, quality, and cost.

3.1.1 The Concept of the Test Cycle

In all testing one common subprocess exists, which is referred to as "the test cycle." As depicted in Figure 1, the test cycle consists of three functions: detection, isolation, and correction.

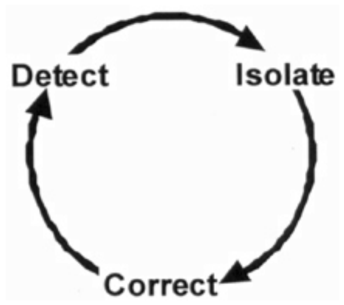

Figure 1 - The Test Cycle

- "Detection" determines the presence of an anomaly, including design flaws, manufacturing defects, and field defects.
- "Isolation" determines the location of the cause of the anomaly.
- "Correction" is the process of removing and replacing the item causing the anomaly, such as would occur in a manufacturing or field repair process. Depending upon context, it is also considered to be the process of removing the damaging effect of the item causing the anomaly, such as would be the case in a fault-tolerant system.
- Other test related functions that have widespread application in the life cycle of a system, such as test control, initialization, recording and reporting of anomalies are important functions. But they are not as fundamental and common to every test process in every phase of the life cycle, as the functions defined in the "test cycle".

3.2 Compliance Tracking

Continuous test requirements compliance tracking is implemented through overlapping processes of prediction, verification, and measurement. "Prediction" is the process of determining compliance to test requirements prior to the availability of a complete, detailed design. It is accomplished through comparison with similar library elements or through analytic techniques, such as circuit level testability measures or topological dependency models. "Verification" makes use of such techniques as deterministic fault simulation, probabilistic fault grading, and closed form proofs, for such techniques as exhaustive type BIST approaches.

"Measurement" compliance tracking process applies when test performance measurements can be made on actual hardware and software, such as physical prototypes, manufactured or fielded systems.

Together, the three processes of prediction, verification, and measurement provide feedback to track test requirements compliance and to allow correction of deficiencies. In addition, the same feedback is used tD correlate the results of the methods and tools used in the three overlapping compliance tracking processes to assess their adequacy and permit improvement in the tracking processes themselves.

3.3 DFT Reuse Concept

One of the key ways in which the DFT Methodology contributes to the achievement of the RASSP goals is through the enforcement of reuse in four different dimensions. The four dimensions of DFT reuse are as follows:

a. Across the life cycle phases within a given model year. From system definition to field support, each DFT/BIST embodiment or instantiation is designed for maximum reuse in the phases to follow.
b. Across the packaging hierarchy within a given model year. From chip to system, each DFT/BIST embodiment or instantiation is designed for maximum reuse in testing the package levels above and below it.
c. Across a single packaging level within a given model year. Each DFT/BIST embodiment or instantiation is designed for maximum reuse in testing other entities within the same package level. For example, a board level BIST technique is examined to see if it can be used for testing other boards in the system.
d. Across new model years or as new systems/ products unfold. The RASSP reuse library is used to provide the outputs of some steps in the process (rather than perform the step from scratch), and outputs of each process step become candidates for encapsulation in the reuse library for future model years or systems. Test related reuse items include, for example, test requirements; test strategies; DFT/BIST techniques for certain logic structures; testable chips, MCMs, etc.; BIST software modules; and test vector sets as appropriate.

3.4 Relationship of Test Requirements, Test Strategies, and Test Architectures

Figure 2 illustrates the relationship between test requirements, test strategies, and test architectures. Test requirements specify the fault coverage, test time, and other parameters associated with testing the Unit-UnderTest (UUT). The UUT may be a chip, MCM, board, or system. Test requirements do not include specification of test means, but instead, are allocated to various test means during the development of "test strategies."

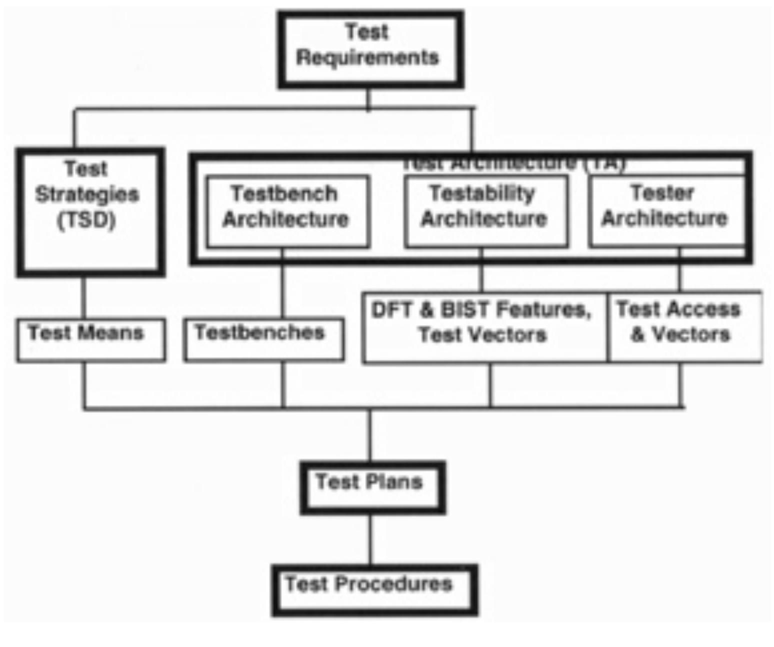

Figure 2 - Relationship of Test Requirements, Test Strategies, and Test Architectures

Requirements are checked for realizability, consistency, and validity when they are received, generated o r specified. This process minimizes the problem where an invalid requirement [such as 99% single stuck-at gate level fault detection for a design with 100%

commercial off-the-shelf (COTS) components] flows all the way down the board level before it is found to be impossible to meet. A requirement is considered realizable if it can be achieved, given current test technology, and system constraints (size, weight, power, degree of COTS, etc.). A requirement is consistent, if it does not contradict any other requirements and is traceable to a higher level requirement. Finally, a test requirement is valid if there is currently a practical and cost effective means to predict, verify, and measure compliance. A requirement is amended or deleted if it is not consistent, realizable, and valid.

A test strategy) defines the mix of test means by which the UUT will be tested and is developed for each phase of testing (design, manufacturing, and field. Test means can be built-in self-test (BIST hardware and/or software based), test equipment, and manual procedures. Test requirements drive the selection and mix of test means and are allocated each of the selected test means. Test strategies, which are specified and managed by the use of "test strategy diagrams" described below, drive the development of test plans and test procedures at each level of the packaging hierarchy. Test plans describe a high level view of testing the UUT, including budget, schedule, resources, capital equipment, and a description of the test flow. Test procedures give the detailed, step by step procedures for testing the UUT. As with test requirements, test strategies are checked for realizability, consistency, and validity when they are generated and documented, before they are flowed down to the next lower level of abstraction.

A test architecture is a suite consisting of the UUT and any test equipment required, based on the developed test strategy. The test architecture consists of the "testbench architecture" which describes the simulation-based testbench for design-verification; the "testability architecture" which describes the DFT and BIST features in the UUT; and the "tester architecture" which describes the configuration of "test equipment" required for any externally based testing. During the development of the testability architecture, the impact on the functional design, such as real estate, 1/0 pins, performance, power, and reliability, is constantly evaluated be ensure it is acceptable. Test architectures are checked for realizability, consistency, and validity when they are generated and documented, before they are flowed down to the next level of abstraction (as was done with the test requirements and strategies). Note, certain types of testability architectures, such as those based on BIST and boundary-scan, are more robust than others, in that they are applicable across the packaging hierarchy, as well as across all life cycle testing phases.

As the system is decomposed into its constituent elements, test requirements, test strategies, and test architectures are "flowed-down" to each level of hardware packaging and software abstraction level. At each level, compliance to the higher level test requirements, strategies, and architectures is checked. If at a lower level, it is impossible to adhere to the requirements, strategy, or architecture, the previous level or levels are re-examined and analyzed to correct the problems in those higher level requirements, strategy, or architecture. Note, the amount of backtracking required should be minimized and problems should be identified earlier than in normal methodologies, since the strategy and architecture are driven by the requirements and all three are checked for realizability, consistency, and validity when they are generated initially at the front end of the process and also when captured at each successive level.

3.5 Test Strategy Diagrams

The test strategy diagram (TSD) is a key construct used to bridge from requirements to implementation, manufacturing, and field; to "knit" all of the test processes together; to provide a means of carrying information between and within steps of the process; and manage the requirements specification and compliance tracking.

The TSD is based on the simple concept of allocating a given "anomaly population" onto given "test means" for the purposes of detection, isolation and correction. A “test means" is a vehicle used to detect, isolate, and possibly correct an anomaly in an item under test. The nature of the test means and item under test depends on the life cycle step and level of system hierarchy at which the test is applied. For example, during the early stages of the development process, the test means could be specified in general terms such as peer reviews, simulation, ATE, and BIST, while the "item under test" might be specified as requirements specifications, behavioral model, boards, and chips, respectively. On the other hand, later in the detailed design stage, the vehicle might be DSP ASIC team review, structural VHDL simulation, PC-based boundary-scan tester, and circular BIST respectively, while the item under test might be DSP ASIC flowed down manufacturing test requirements, DSP ASIC structural VHDL model, signal processing board prototype, and DSP ASIC in manufacturing test.

The anatomy of a test strategy diagram is shown in Figure 3. It consists of a three by "n" array of cells, comprised of one row for each of the three test functions, detection, isolation, and correction. The n columns correspond to n test means, such as BIST, ATE, or manual procedures. Test means are listed in priority order from left to right, starting with the means that has the highest fault detection coverage. The input to the array is the total population of anomalies. Each cell in the array can be thought of as a node in a flow graph, having one input and one or two outputs.

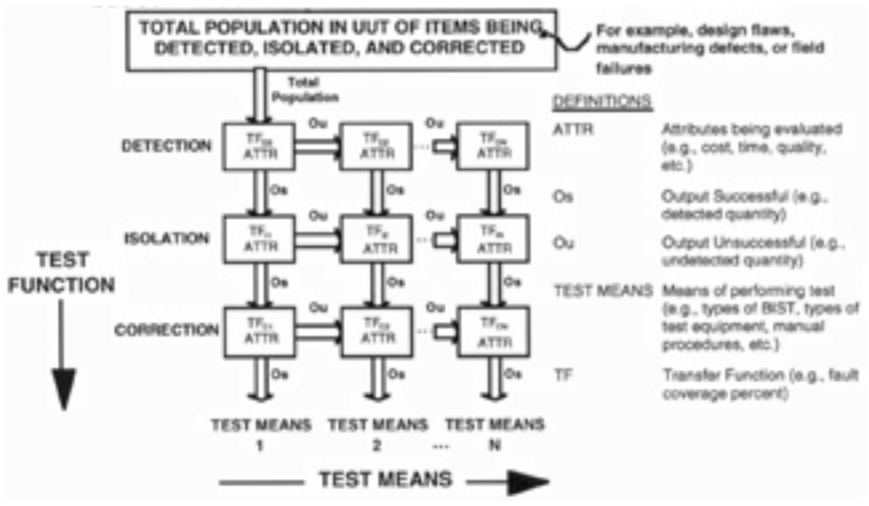
TOTAL POPULATION IN UUT OF ITEMS BEING DETECTED, ISOLATED, AND CORRECTED
For example, design flaws, manufacturing defects, or field failures
Total Population
DEFINITIONS
ATTR
Attributes being evaluated (e.g., cost, time, quality, etc.)
Os
Output Successful (e.g., detected quantity)
Ou
Output Unsuccessful (e.g., undetected quantity)
TEST MEANS
Means of performing test (e.g., types of BIST, types of test equipment, manual procedures, etc.)
TF
Transfer Function (e.g., fault coverage percent)
DETECTION
ISOLATION
CORRECTION
TEST FUNCTION
ATTR
Os
Ou
TEST MEANS 1
TEST MEANS 2
TEST MEANS N
TEST MEANS

Figure 3 - The Anatomy of a test Strategy Diagram

Inside each cell is a transfer function or operator that acts on the input to produce the output(s). The most common transfer function in the TSD is the coverage operator, which expresses the degree of detection, isolation, and correction coverage. It is stored as a decimal value, displayed as a percentage and is multiplied times the quantity coming into the cell. Note, the transfer functions of test means are position dependent. That is, the transfer functions in the nth column operate on the residual population from the previous (n-1).

In addition to the transfer function, one or more attributes are associated with the cell. Examples of important attributes are test time, test cost, real estate penalty, quality levels, etc. For example, a test time stored in the upper left cell in Figure 3 could represent the time for test means #1 to detect a fault. A cost factor could include a one time (non-recurring) cost for test generation and test evaluation, plus a recurring cost for test application that would be incurred for each unit. The attribute is normally stored as a worst case value, although average values can be used as appropriate.

Early in the process, initial values for transfer functions are requirements, while the attributes and the anomaly population density can be obtained from historical data accumulated in the RASSP reuse library, can be generated from new estimates (based on similar systems), or can make use of a variable to take the place of a fixed value. As the process continues, actual values for transfer functions, anomaly population densities, and attributes are entered from predictions, verifications, and measurements.

The implied implementation of the TSD makes use of a three dimensional spreadsheet paradigm, since it is a tool that is available in most companies and that can run on a PC. However, it is certainly possible that a custom TSD tool could be developed in a compiled high level language (e.g., C++) and provide much faster operation, particularly for large systems, as well as provide a more friendly graphical user interface. When a TSD cell is "activated," its attribute is factored into (usually by some simple arithmetic operation) the accumulated value of that attribute through each path in the array intersecting with the cell. Thus, test time and cost factors can be summed across test means and test functions. Test cost information can be used for performing test strategy tradeoffs, manufacturing cost budgeting, or maintenance and logistics planning. Test time information can be used for input to a fault-tolerant system's availability model or for manufacturing test planning and scheduling.

Each cell in the array has one or two outputs. The quantity exiting from the bottom of the cell represents the "successful" quantity. For example, if the transfer function is fault coverage, the quantity exiting from the bottom of the top left cell in 3 represents the detected faults. The quantity exiting horizontally represents the "unsuccessful" quantity, which in the example cited above would be the undetected faults. The quantity of unsuccessfully handled faults can be used, as part of a manufacturing defect (quality) level analysis.

"Terminating cells," represent the last means to achieve detection, isolation, and correction and have only one nonzero output (successful D/l/C). In the case that a TSD lacks a terminating cell for either detection, isolation, o r correction functions then that fault quantity must be handled during testing at the next higher level of package testing. Therefore, anomaly quantities emanating from the right side of the TSD must be included in the anomaly population for the next higher level of package, along with the new fault population associated with the next higher level package. Thus, the TSD concept forces consideration of a 11 possible scenarios of anomaly detection, isolation, and correction. For example, if 100% of the faults must be detected, isolated, and corrected by some combination of BIST, test equipment, and manual procedures then the sum of all faults entering the top of the system TSD must equal the sum of the outputs exiting at the bottom of all TSDs across all packaging levels.

TSD's have many applications within the DFT Methodology, and the collection of TSDs for a system forms a hierarchy, as shown in Figure 4. A TSD would be generated to capture and store the initial system level test requirements (shown on the bottom left of the figure). The system-level, requirements TSD is used to flowdown the requirements to lower levels of the hierarchy, thus producing a requirements TSD at each packaging level. As the prediction, verification, and measurement data are generated during compliance tracking across the life cycle, a TSD is generated for each set of data (P, V, and M) at each packaging level, and automatically is compared against the requirements TSD values to flag compliance violations. A comparison can also be done between P data, V-data and M-data, to assess the integrity of the prediction, verification, and measurement methods and tools.

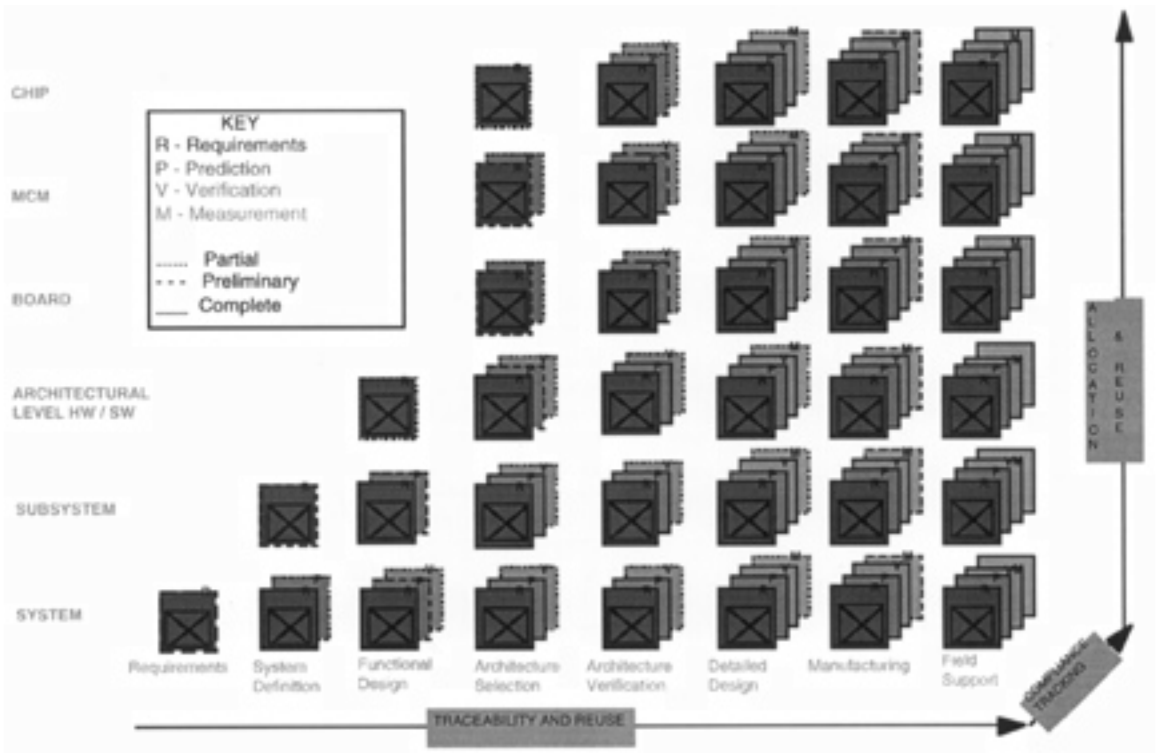

Figure 4 - The Hierarchy of Test Strategy Diagrams

The application of the TSD along the life cycle dimension in Figure 4 can be thought of as providing traceability and reuse enforcement. This is accomplished by prohibiting the "collapsing" of the TSD spreadsheet as the life cycle steps are traversed. For example, the establishment of boundary-scan based PC testing during the design phase would automatically be carried forward into the TSDs for manufacturing and the field to force at least an analysis of the potential reuse of the test means in those stages.

The application of the TSD along the vertical (packaging hierarchy) dimension can be considered as providing requirements allocation and reuse enforcement as well. In regard to the latter application, the reuse enforcement is accomplished similar to the life cycle reuse enforcement: by prohibiting the "collapsing" of TSD's when evaluating TSD values up the hierarchy. It is required that a test means be carried up to at least one higher level of packaging for analysis. For example, the use of BIST and boundary-scan in a chip as a chip-level test-means would appear not only in the chip-level TSD, but also reappear in the TSDs of at least the MCM or board in which the chip resides. If the chip becomes the basis of a board-level BIST capability, that board-level BIST capability must be considered as a test means for the system-level TSD. This philosophy forces the evaluation of the contribution of low-level test-means (especially BIST) at the higher packaging levels.

The application of the TSDs in the diagonal dimension in Figure 4 (R, P, V, M) represents the means for managing compliance tracking information. TSDs with dashed or dotted lines represent, respectively, those that are preliminary or for which only partial information would be available in that life cycle phase.

3.5.1 Example of Using TSD's for Integrated Diagnostics Analysis

Figure 5 provides an example of a TSD for trading off, analyzing, and finally capturing field test requirements for a system having both BIST and fault tolerance requirements. This example was developed under a study of the Integrated Diagnostics for a fault tolerant avionics system for an advanced tactical fighter program. The test requirements for the system assumed a fault model that consisted of single stuck-at faults, as well as single transient or intermittent faults each fault affecting a single node.

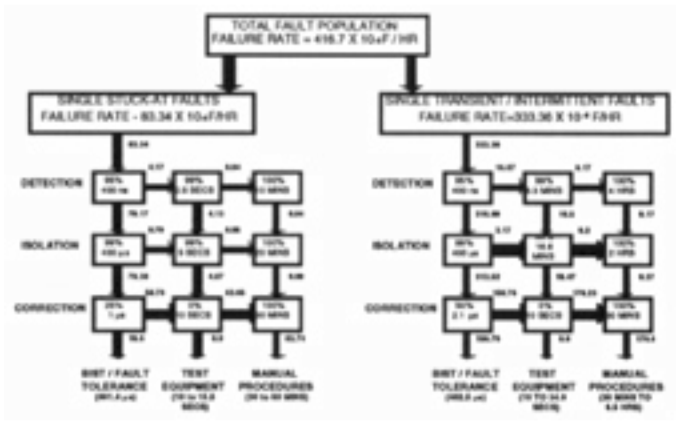

Figure 5 - TSD's developed to study Integrated Diagnostics for fault tolerant avionics.

In the project, failure rate was used as an estimate of relative fault population densities (shown at the top of the diagram). It is important to note that the failure rate values were not derived from the traditional MIL-STD 217 MTBF predictions. They were derived from a reliability model which assumed that only the specific target faults defined in the fault model occurred. Hence, consistency between the reliability predictions and the testability analysis was assured.

The total failure rate for the system was allocated between two types of faults (stuck-at and transient/intermittent). The allocation between the tow fault types was based upon statistics gathered during previous field failure studies. The studies showed that similar systems in the same type of environment tended to experience four times as many transient or intermittent failures as solid (stuck-at) failures.

Following the allocation described above, the total failure rate for each fault population, single stuck at and single transient/intermittent, was entered into the top of the respective cell arrays. For both types of fault populations, the detection, isolation, and correction requirements were entered into the cells for each of the three allowed test means: BlST/fault tolerance, test equipment, and manual procedures. The calculations were performed using test coverage and an attribute of test time. Fault coverage numbers in each cell are multiplied times the failure rate number coming into the cell to determine what quantity of the fault population is successfully detected, isolated, and corrected by each of the test means.

The test time attribute is "activated" when the transfer function is applied to the fault population entering a given TSD cell. The TSD allows for the analysis of each scenario of handling faults by any combination of the three allowed test means, providing the possible values of cumulative test time through each scenario path (shown at the bottom of each array column). A range of aggregate test times is shown for all but the first column, because there are several scenarios or mixes of detection, isolation, and correction means that could lead to the same end point for the second and subsequent columns. For example, within the single stuck-at fault array, the fault could be detected (400 ns) and isolated (400 us) by BlST/fault tolerance, but corrected manually, yielding the low end of the range at the bottom of the manual procedure column, roughly 30 minutes. On the high end, the fault could be detected (10 minutes), isolated (20 minutes), and corrected (30 minutes) solely by manual procedures, yielding the upper limit, 60 minutes. Note that a range of cumulative test time is indicated for the test equipment column even though there are no scenarios that end there (due to the correction capability for test equipment being zero). This range is shown simply for completeness and comparison purposes.

Later a cost figure was associated with the test time, and a "cost of test" analysis was then performed to provide another dimension of tradeoffs to further refine the requirements. Eventually, after many "what-if studies" using the TSD to swap test means and to evaluate test times and costs for various test strategies, the requirements as shown in figure 5 were finalized and 'frozen' in the TSD. Subsequent prediction, verification, and measurement TSDs were then compared against the requirements TSD to flag discrepancies during compliance tracking .

3.6 The Prescribed Testability Architecture

The prescribed RASSP testability architecture relies heavily on the use of BIST and IEEE 1149.1 boundary-scan incorporated at the IC level. It also relies on the reuse of BIST and boundary-scan tests at all packaging levels from the MCM-level to the system-level. COTS components and/or designs are accommodated by use of the "lead, follow, or get out of the way" philosophy. The "lead" concept suggests insertion or use of a COTS item or reuse item that incorporates BIST or DFT features that can form the basis for a test. The "follow" concept applies COTS items that may not incorporate BIST or DFT features, but which at least do not interfere with the test (e.g., simply pass the test vectors through to the next stage). Finally, the "get out of the way" concept applies to COTS items which lack testability and BIST, and which must be bypassed during the primary test process and dealt with separately for their own test.

Communication of test data across the packaging hierarchy is accomplished through a system of hierarchical test and maintenance controllers and busses such as IEEE 1149.5.

An overview of the RASSP DFT Methodology and its relationship to the overall RASSP Methodology is illustrated in Figure 6. It has some important features, as outlined below:

a. It is a methodology embedded in the RASSP methodology and proceeds concurrently. It is deliberately not depicted as a separate process, but rather as an embedded subset of the overall RASSP process.
b. It is driven by test requirements that span the entire life cycle of the system from design to manufacturing to field support.
c. It promotes continuous tracking of compliance to requirements, rather than periodic tracking.
d It forces attention to reuse in multiple dimensions

Details of the methodology are presented in the sections that follow, all of which make use of Figure 6.

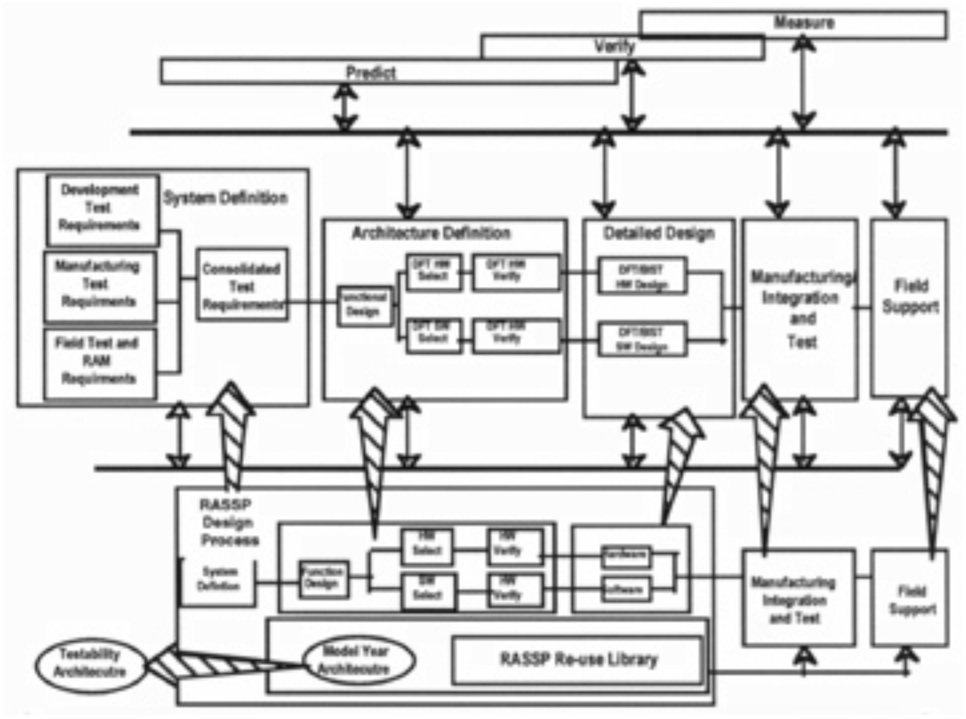

Figure 6 - The RASSP "DFT Methodology" embedded in the overall methodology.

4.1 System Definition

As depicted in Figure 6, the first step, System Definition, involves test requirements specification. The test requirements come from an integration of customer and derived requirements for the three phases of testing -design, manufacturing, and field support. Specification involves a preliminary, life-cycle cost-of-test analysis (if such an economics model is available), a test-technology assessment, and a design-impact analysis to determine the realizability, consistency, and validity of the requirements. Subsequently, during the same step, the three sets of requirements are consolidated into a consolidated requirements specification and captured in the first test strategy diagram. The purpose of consolidating the life cycle requirements is to establish a single source of test requirements and, more importantly, to encourage all three test organizations (design, manufacturing, and field) be explore the possibility of a singular test philosophy that can be used throughout the entire life cycle of the system.

4.2 Architecture Definition

The Architecture Definition phase, as shown in Figure 6, consists of three steps: functional-design, architecture selection, and architecture-verification

In the architecture selection step, the top level test strategy is used to develop and evaluate various candidate, top level test architectures, by determining which architecture(s) best supports the top level test strategy with the least impact on the candidate "functional" architectures. Thus, the impact of each test architecture on the candidate "functional" architectures is assessed and incorporated into the tradeoff and selection process for them. Test requirements, test strategy, and test architecture are then allocated to BIST/DFT hardware and software for one or more of the selected architectures. In addition, candidate

DFT and BIST techniques are identified for later implementation, based on the specified requirements. Also in this step, any top level BIST supervisory software development will begin, as will on-line BIST code, since it may have an impact on functional performance and throughput. Prediction and verification processes begin in this stage as appropriate for compliance tracking.

In the Architecture Verification step, the next level of detail of the selected test architecture(s) are generated and additional details are provided regarding the test architecture impact on the selected functional architecture(s). For example, behavioral and performance simulations will be modified to include effects of DFT/BIST techniques, such as the estimated performance degradation due to hardware concurrent fault detection circuits or due to periodic execution of on-line BIST software diagnostics. Prediction and verification processes continue during this stage for compliance tracking.

4.3 Detailed Design

In the Detailed Design phase, test strategies, architecture, and requirements are flowed down to the detailed design of BIST/DFT hardware and software. The detailed design of the BIST/DFT hardware is performed concurrently and interactively with functional design using automatic (e.g., front-end or back-end synthesis) or manual insertion and then is reflected into behavioral and structural simulation models, whenever possible. The BIST/DFT detailed design is performed interactively with the functional design, since each impacts the other. Any remaining BIST software (e.g., for power-up or other off-line BIST functions) is implemented. Test vector sets are developed and verified for each packaging level for physical prototype test, production test, and field test. All test vector sets are documented using WAVES. As the detailed design is verified and debugged, design flaw models are updated to provide more accurate models for the TSDs. Prediction, verification, and measurement processes are used in this stage for requirements compliance tracking .

4.4 Manufacturing

In the Manufacturing phase, functional and performance testing of the overall prototype is performed to verify compliance to functional and performance requirements, with DFT and BIST hardware and software included. Ongoing production test is performed. Verification and measurement processes are used in this stage for compliance tracking to test requirements. Measurement data i s acquired through such techniques as automatic, system, fault-history logging and ATE-based data collection. BIST and tester-based test cost and performance data are captured and encapsulated for the reuse library. Manufacturing defect analysis profiles and distributions are used to update the manufacturing fault model for use in the TSDs.

4.5 Field Support

In the field phase, BIST and DFT capabilities are in use. BIST and DFT functions are also used for lower level (e.g., organizational and depot level) testing. Verification and measurement processes are used in this stage for compliance tracking. As in the manufacturing phase, measurement data is acquired through such techniques as automatic, system, fault-history logging and ATE-based data collection. BIST and tester-based test cost and performance data are captured and encapsulated for the reuse library. Field defect analysis profiles and distributions are used to update the field fault model for use in the TSDs.

4.6 Additional Activities within the DFT Methodology

Throughout the entire DFT methodology, interfacing to the RASSP reuse library is performed to access existing candidates and to add to the library when appropriate. In addition, feedback is being provided continuously from the compliance tracking process back to the responsible persons to assure corrective action is taken. Finally, it is recognized that iterations may be necessary because of the inherent nature of the RASSP methodology.

5.0 Conclusions

An overview of the RASSP DFT Methodology has been presented. It is viewed as an evolving methodology, which can be upgraded and extended effectively. Yet it is tightly integrated with the overall RASSP methodology and directly supports the overall goal of 4X reduction in cycle time and cost.

The definition of testing is expanded to encompass all phases of the life cycle and to include as fundamental activities: detection, isolation and correction. Process steps are included to check requirements for consistency, validity and realizability. Test requirements are consolidated for design verification testing, manufacturing testing, and field support testing to ensure a singular test solution. Process steps are included tD predict, verify and measure solutions be identify problems early and/or to provide feedback for subsequent model years.

Test strategies and architecture are shared and flowed down to ensure consistency and to minimize cost. A structured testability architecture based upon boundary scan and BIST is promoted. Tools are used to control and verify test reuse library elements. Test strategy diagrams are used to control, enforce and verify reuse of test resources across and within levels of the system hierarchy. The TSDs also provide a basis for management (predict, verify and measure) of the requirements as they are flowed down.

Acknowledgment

The authors would like to acknowledge the significant contribution of Pat McHugh of the Army Research Lab, as well as the other members of the RASSP DFT Working Group, all of whom have reviewed the DFT Methodology Document, upon which this paper i s based and have provided very helpful suggestions for improvement of the document.

References

RASSP Documents

RASSP Methodology, Version 1.O, December 1994.

RASSP Model Year Architecture Working Document, Version 1.0, October 28, 1994.

RASSP DFT Methodology, Version 1.O, July, 1995.

Non-RASSP Documents

The Documentation of Military Electronic Computers with the VHDL Handbook, Preliminary Draft Manuscript, Section Vlll, Modeling Testability with VHDL Models, April 16, 1993.

Flynn, et al., "Using WAVES in a TopDown Design Methodology", Proceedings of VIUF Fall 1994 Conf., Component Modeling, paper 12.1.

Sanchez, "Concurrent Engineering with DFT in the Digital System: A Parallel Process", Proceedings of the IEEE International Test Conference 1994, paper 36.1.