IEEE 1149.1 Testability

Introduction

Design for Test (DFT), also known as design for *testability*, is a process that incorporates rules and techniques in the design of a product to make testing easier.

Structured design for test is a system methodology rather than a collection of discrete techniques. This methodology impacts all phases of product′s life, from device circuit design through field service. Design for test is used to manage complexity, minimize development time, and reduce manufacturing cost.

Testing has two major aspects: control and observation. In order to test any system it is necessary to put the system into a known state, supply known input data (test data), and observe the system to see if it performs as designed and manufactured. If control and observation cannot be carried out, there is no way to know empirically if the system performs in the proper manner.

During the normal product development flow, testing (it may be known by different names) takes place at many points during the process. If testing is considered at the chip design level, its benefits can be used at all levels of electronic assembly; from chip through system-level testing. See Figure 1-1.

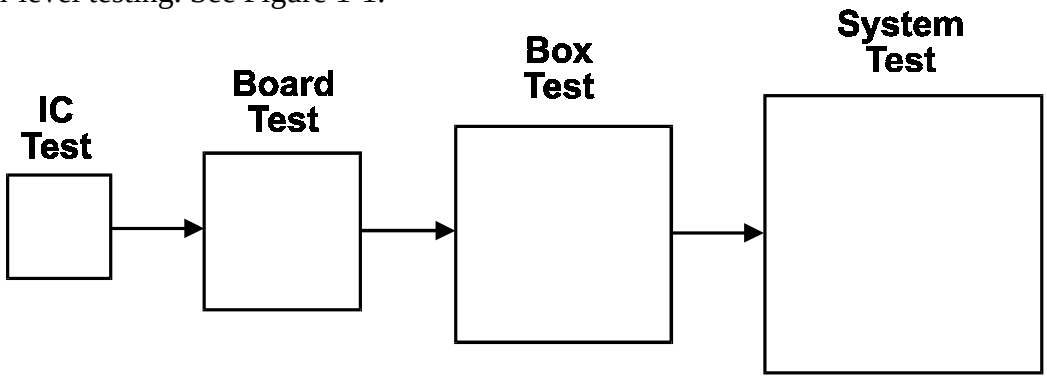

Figure 1-1. Chip Through System-Level Test

Designers usually test various functions to validate their design. Manufacturing and customer groups subject the design to an assortment of unique criteria to see if the concept works in practice. Is it manufacturable? Will it stand up to real-world operating conditions? Will repair be cost efficient? In addition to direct testability considerations, production managers want features designed into the product to help them minimize scrap and manufacturing costs. Good system-testability methodology provides an integrative function throughout the product development cycle and allows materials created during an early phase of development to be reused in later phases. Various chip designers have used this integration feature as a tool to help manage the development of complex products.

Testability provides companies with a firmer grasp on the economic and market-window constraints due to product development. One major workstation manufacturer claimed :

- Test program development would have been nearly impossible without scan techniques.
- Chip-level test development time fell from 1 man-year to about 20 hours.
- Board-level test development time fell from multiple man-years to about a week.
- Three months were cut off development time.

Overall Rationale for Design for Test

Manufacturers of state-of-the-art electronic product face a unique set of problems. Although modem circuit density, high device speed, surface-mount packaging, and complex board-interconnected technology have a positive influence on state-of-the-art electronic systems, these factors can adversely affect ability to verify correct design and operation. Increased complexity and lack of physical access to circuitry makes for costly and time-consuming testing using traditional test techniques.

Reduced Cost and Higher Quality

Reacting to this complexity with an eye on the bottom line, manufacturers may opt to perform less rigorous testing. Manufacturers who choose the less rigorous techniques as an expeditious alternative to the expense of full testing gamble their technical credibility in the market place and expose themselves to the high cost of product returns. In today's global electronics marketplace, a manufacturer who delivers poorly tested products does not remain competitive. The cost for detecting and identifying faults using traditional test methods increases by an order of magnitude as a circuit's level of complexity increases. These increased costs and development time reduce profit margins, delay product introduction, and reduce time-to-market windows. An increasing number of companies have simultaneously improved their product quality and profit margins by adopting system-level (integrative) design methods. Design for test is one such system-level approach.

Benefits Over Standard Test Methods

Time to market is more important than ever before in the high technology marketplace. Companies that can produce quality products with a short product development cycle-time have a competitive advantage. Designing testability into a system can play an important role in introducing a new high-technology product with an expected five-year live cycle to market on time. Table 1-1 shows various product development time/budget scenarios and the resulting project profitability.

	Product A	**Product B**	**Product C**
To Market:	on time	on time	6 month later
Budget:	on	50% over	on
Profit Over 5 Years:	100%	96%	66%

Table 1-1. High-Technology Products Scenarios

Adding testability to a product increases design time and costs while reducing cost of design validation, manufacturing test, and system maintenance.

The system design phase of product development represents only 15% of product's total life-cycle costs. However, the system design phase has a 70% impact on product's operation and support costs over the product's total life. (Source: Mitre Corporation, 1987 Government Microcircuit Applications Conference)

The majority of faults found on boards, such a solder joints (shorts and opens), components (wrong device, missing device, wrong orientation, wire bond failure, and stuck pins), etch integrity, and connector faults, make up over 95% of failures found. Structured technique such as boundary-scan technique allows for pins-out testing to detect this failures easily. (Source: Teradyne)

The additional costs of designing testability into a system during the system design phase can be more than made up over the product's total life.

Design cycle times have shortened significantly over the years while test program development time has increased, necessitating that companies adopt structured or repeatable methodologies. Table 1-2 documents the increase in test program development in time as a test requirements increase.

1977 - 1980	**3 - 6 months**
1981 - 1983	**6 - 12 months**
1984 - 1986	**9 - 18 months**
1987 - 1990	**12 - 24 months**

Table 1-2. Time of Develop Test Programs (in Man Months)

Standard Test Solutions Versus Proprietary Solutions

Embedded test, emulation, and maintenance circuitry are well defined and understood within the test community. Previously, the lack of standards caused these structures to be implemented in an ad hoc and proprietary manner. Since proprietary solutions are usually more expensive and labor intensive, the added costs further limited the use of these test circuits. Boundary-scan testing combined with the common test bus interface and test protocol has these benefits:

- Provides a standard and cost effective solution to traditional test problems
- Opens up new applications

The ability of reuse previously developed test data and to use less costly test equipment means that this approach yields products that are less expensive to manufacture.

An Industry Standard - IEEE 1149.1 - 1990 (JTAG)

In 1985, an ad hoc group composed of key electronic manufacturers formed a group called JTAG (Joined Test Action Group). JTAG had over 200 members around the world including major electronics and semiconductors manufacturers. This group met to establish solution to the problems of board test and to promote a solution as an industry standard. The solution, which became IEEE Std. 1149.1-1990. IEEE Standard Test Access Port and Boundary-Scan Architecture, is the basis for TI testability products. The IEEE Std.1149.1 allows test instructions and data to be serially loaded into device and enables the subsequent test results to be serially read out.

Every IEEE 1149.1 - compatible device has four additional pins - two for control an one each for serial test data input and output. To be compatible, a component must have certain basic test features, but the IEEE 1149.1 specification allows designers to add test features to meet their own unique requirements. The specification was adopted as an IEEE standard in February of 1990.

Benefits of Testability

This chapter explains how designing testability into devices eliminates problems associated with traditional testing and improves quality and efficiency.

Traditional Testing

Traditional board-level and device-level testing consumes a great deal of time and requires special hardware and complex Automated Test Equipment (ATE) for each type of board or device. This results in increased costs and development time. In addition, extensive testing is necessary for the heightening reliability standards and performance standards in the defence, aerospace, automotive, computer, and communications industries. These extensive tests can delay the market introduction of products, disrupt Just-In-Time (JIT) manufacturing flows, and limit the productivity of standard ATE operations. This creates numerous problems because time to market is more important than ever before in the high-technology marketplace. Companies that produce quality products with a short product-development cycle time have a competitive advantage.

Efficient Testing

An innovative approach to the problems inherent with traditional testing is to incorporate design-for-test techniques that allow embedded testing to be performed. For example, data can be scanned in to stimulate internal system nodes while the component or circuit is embedded within the system. During the same scan, the previous condition of each node is scanned out. This saves test time and reduces the number of test vectors needed.

Lower Cost for Testing

The additional costs of designing testability into a system during the design phase is more than made up over the products total life. This is accomplished by reducing the test program development time, minimizing fixture complexity, and allowing for the use of lower cost ATE solutions. Another cost benefit is the economy of scale gained by having a standard test approach that spans design, test, manufacture, field repairs, and maintenance.

Production Time Savings

Board-level boundary-scan testing is easily implemented using TI's line of IEEE 1149.1 testability devices such as:

- Octal and Widebus™ buffers
- ASICs and FPGAs
- DSPs, microprocessors, and controllers

These IEEE1149.1-compliant devices are included in board design with little modification to existing circuitry. Embedded testability greatly reduces the need for other test points on the board and offers these advantages:

- Greatly simplified test fixtures
- Reduced fixture construction time
- Sophisticated build-in test and debug operations

Many ICs on boards may be tested together using the serial IEEE 1149.1test bus under the control of ASSET software.

Easier Board-Level Isolation

Fault isolation on a printed circuit board can be greatly improved by electronically isolating suspect areas using boundary-scannable devices. The IEEE1149.1 test bus controls boundary-scannable devices to place them in *EXTEST* for pins-out testing. This effectively partitions offer isolates circuitry for separate testing. Partitioning the system using IEEE 1149.1-compliant devices reduces the number of patterns required for testing each circuit area. See Figure 2-1 for an example of a design than can be partitioned.

Simple Access to Circuits

Highly integrated, modern, multi-layer systems or ICs with fine-pitch pins are virtually impossible to access using manual probes or ATE. Some boards require extensive fixturing or even some redesign before they can be tested effectively. TI's testability devices with boundary-scan architecture eliminate physical access problems. These parts provide the designer with testability for the most complex and hard-to-access circuits, and add controllability of test circuits. In addition, a designer can easily observe and control internal device functions.

Figure 2-1. Boundary-Scan Testing Using the IEEE 1149.1 Bus

Boundary-Scan Architecture and IEEE Std 1149.1

Boundary scan is a special type of scan path with a register added at every I/0 pin on a device. Although this requires the addition of a special test latch on some pins, the technique offers several important benefits. The most obvious benefit offered by the boundary-scan technique is allowing fault isolation at the component level. Such an isolation requirement is common in telecommunications switching environments where prompt field repair is critical.

A major problem driving the development of the IEEE 1149.1 boundary-scan standard is the adverse effect of surface-mount technology. The inclusion of a boundary-scan path in surface-mount components, in many cases, affords the only way to perform continuity tests between devices. By placing a known value on an output buffer of one device and observing the input buffer of another interconnected device, it is easy to see if the Printed Wiring Boards (PWB) net is electrically connected. Failure of this simple test indicates broken circuit traces, cold solder joints, solder bridges, or Electrostatic Discharge (ESD) induced failures in an IC buffer-all common problems on PWBs.

A less obvious advantage of the boundary-scan methodology is the ability to apply predeveloped functional pattern sets to the I/0 pins of the IC by way of the scan path. IC manufacturers and ASIC developers create functional pattern sets for DC test purposes. Subsets of these patterns can be reused for in-circuit functional IC testing. Reusing existing patterns in the development of system diagnostics can save large amounts of development resources, especially if many of the ICs in a system have embedded boundary-scan paths.

The IEEE 1149.1 standard is a common protocol and boundary-scan architecture developed into an industrial standard after thousands of man hours of cooperative development by approximately 200 major international electronics firms. Early contributors in the development of the IEEE 1149.1 standard were AT&T, DEC, Ericsson, IBM, Nixdorf, Philips, Siemens, and Texas Instruments. These companies recognized that only a nonproprietary architecture would encourage companies to offer the compatible integrated circuits, test equipment, and CAD software needed to bring product development, manufacturing, and test costs under control in today's competitive electronics marketplace. Many people believe that boundary-scan architecture will do for development, manufacturing, and test what the RS-232C standard did for computer peripherals.

Boundary-Scan Overview

Boundary scan is the application of a scan path at the boundary (I/O) of ICs to provide controllability and observability access via scan operations. In Figure 3-1, an IC is shown with an application-logic section and related input and output, and a boundary-scan path consisting of a series of boundary-scan cells (BSC), in this case one BSC per IC function pin.

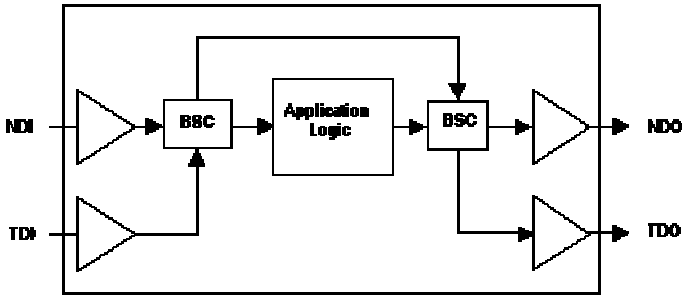

Figure 3-1. Boundary-Scan Example

The BSCs are interconnected to form a scan path between the host IC's Test Data Input (TDI) pin and Test Data Output (TDO) pin. During normal IC operation, input and output signals pass freely through each BSC, from the Normal Data Input (NDI), to the Normal Data Output (NDO). However, when the boundary-test mode is entered, the IC's boundary is controlled in such a way that test stimulus can be shifted in and applied from each BSC output (NDO), and test response can be captured at each BSC input (NDI) and shifted out for inspection. External testing of wiring interconnects and neighboring ICs on a board assembly is accomplished by applying test stimulus from the output BSCs and capturing test response at the input BSCs. As an option, internal testing of the application logic can be accomplished by applying test stimulus from the input BSCs and capturing test response at the output BSCs. The implementation of a scan path at the boundary of IC designs provides an embedded testing capability that can overcome the physical access problems in current and future board designs.

Test Interface and Boundary-Scan Architecture

The IEEE 1149.1 architecture is shown in Figure 3-2. The architecture consists of an Instruction Register, a Bypass Register, a Boundary-Scan Register (highlighted), an optional User Data Register, and a test interface referred to as a Test Access Port (TAP). In Figure 3-2, the Boundary-Scan Register (BSR), a serially accessed Data Register comprised of a series of boundary-scan cells (BSCs), is shown at the input and output boundary of the IC.

The Instruction Register and Data Registers are separate scan paths arranged between the primary Test Data Input (TDI) pin and primary Test Data Output (TDO) pin. This architecture allows the TAP to select and shift data through one of the two types of scan paths, instruction or data, without accessing the other scan path.

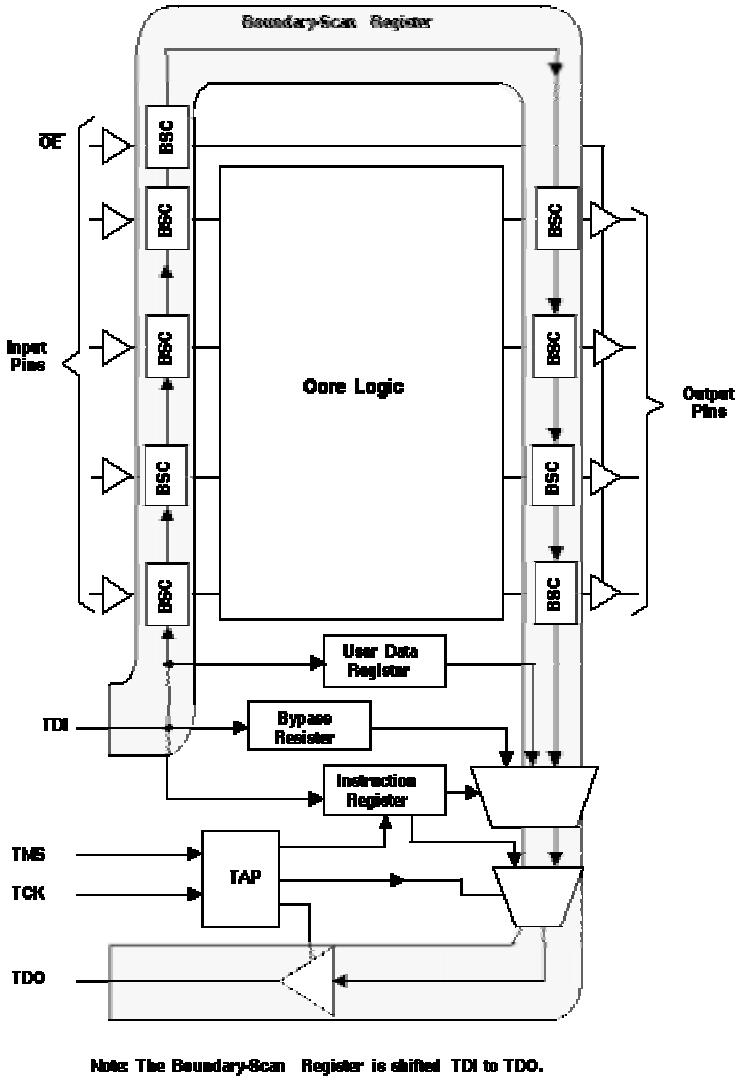

Figure 3-2. Boundary-Scan Architecture

Test Access Port and Operation

The TAP is controlled by the Test Clock (TCK) and Test Mode Select (TMS) inputs. These two inputs determine whether an Instruction Register scan or Data Register scan is performed. The TAP consists of a small controller design, driven by the TCK input, which responds to the TMS input as shown in the state diagram in Figure 3-3. The IEEE 1149.1 test

bus uses both clock edges of TCK. TMS and TDI are sampled on the rising edge of TCK, while TDO changes on the falling edge of TCK.

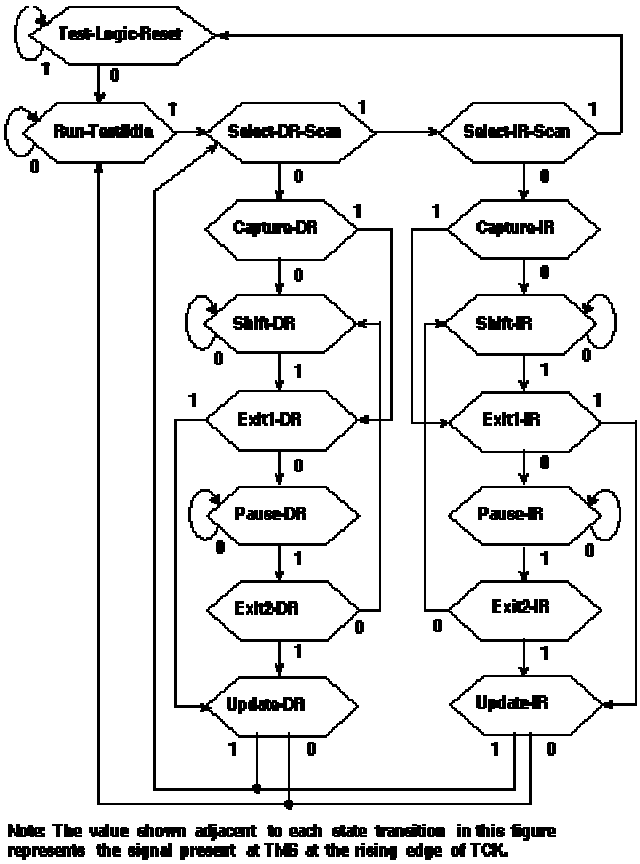

Figure 3-3. TAP State Diagram

The main state diagram consists of six steady states; Test-Logic-Reset, Run-Test/Idle, Shift-DR, Pause-DR, Shift-IR, and Pause-IR. A unique feature of this protocol is that only one steady state exists for the condition when TMS is set high: the Test-Logic-Reset state. This means that a reset of the test logic can be achieved within five TCKs or less by setting the TMS input high.

At power up or during normal operation of the host IC, the TAP is forced into the Test-Logic-Reset state by driving TMS high and applying five or more TCKs. In this state, the TAP issues a reset signal that places all test logic in a condition that does not impede normal operation of the host IC. When test access is required, a protocol is applied via the TMS and TCK inputs, causing the TAP to exit the Test-Logic-Reset state and move through the appropriate states. From the Run-Test/Idle state, an Instruction Register scan or a Data Register scan can be issued to transition the TAP through the appropriate states shown in Figure 3-3.

The states of the Data and Instruction Register scan blocks are mirror images of each other adding symmetry to the protocol sequences. The first action that occurs when either block is entered is a capture operation. For the Data Registers, the Capture-DR state is used to capture (or parallel load) the data into the selected serial data path. If the BSR is the selected Data Register, the normal data inputs (NDI) is captured during this state. In the Instruction Register, the Capture-IR state is used to capture status information into the Instruction Register.

From the Capture state, the TAP transitions to either the Shift or Exit1 state. Normally the Shift state follows the Capture state so that test data or status information can be shifted out for inspection and new data shifted in. Following the Shift state, the TAP either returns to the Run-Test/Idle state via the Exit1 and Update states or enters the Pause state via Exit1. The reason for entering the Pause state is to temporarily suspend the shifting of data through either the Data or Instruction Register while a required operation, such as refilling a tester memory buffer, is performed. From the Pause state shifting can resume by re-entering the Shift state via the Exit2 state or terminated by entering the Run-Test/Idle state via the Exit2 and Update states.

Upon entering the Data or Instruction Register scan blocks, shadow latches in the selected scan path are forced to hold their present state during the Capture and Shift operations. The data being shifted into the selected scan path is not output through the shadow latch until the TAP enters the Update-DR or Update-IR state. The Update state causes the shadow latches to update (or parallel load) with the new data that has been shifted into the selected scan path.

In Figure 3-4, the TAP control output signals are shown along with the Instruction and Data Register interconnects.

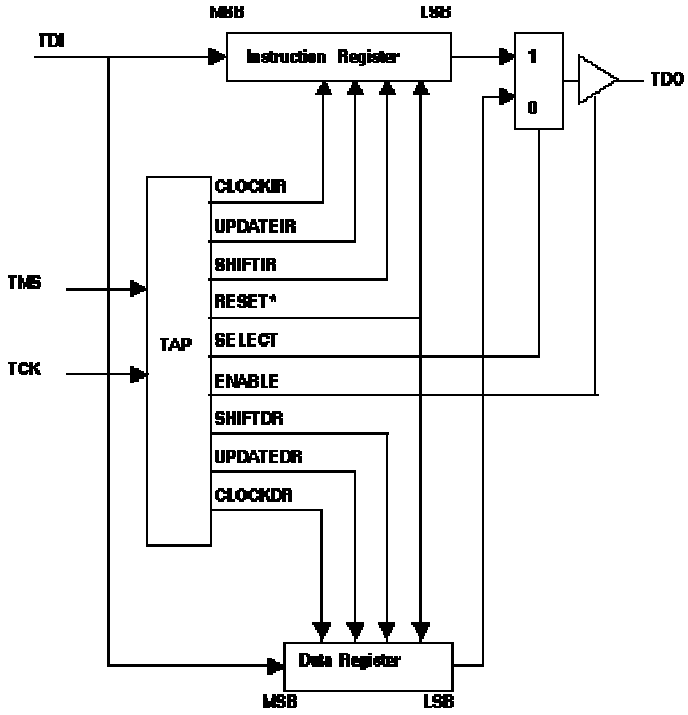

Figure 3-4. TAP Output Control Interconnect Diagram

IEEE 1149.1 Registers

This section contains descriptions of the required and optional registers specified in IEEE Std 1149.1-1990.

Instruction Register (Required)

The Instruction Register is responsible for providing the address and control signals required to access a particular Data Register in the scan path. The Instruction Register is accessed when the TAP receives an Instruction Register scan protocol. During an Instruction Register scan operation the SELECT output from the TAP (Figure 3-4) selects the output of

the Instruction Register to drive the TDO pin. A general Instruction Register architecture is shown in Figure 3-5.

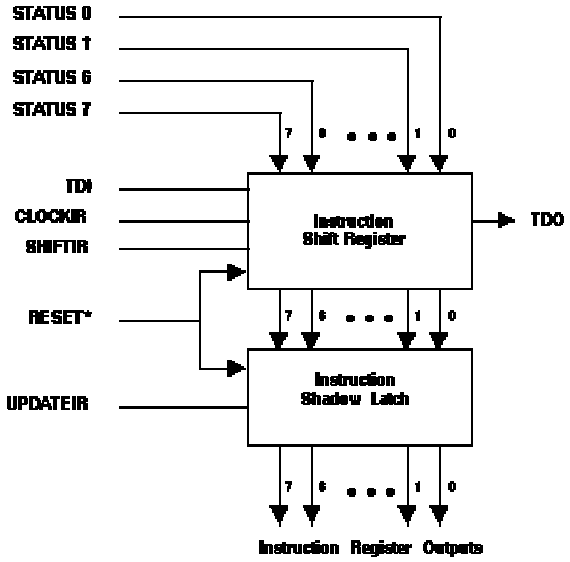

Figure 3-5. General Instruction Register Architecture

The Instruction Register consists of an instruction shift register and an instruction shadow latch. The instruction shift register (Figure 3-5) consists of a series of shift register bits arranged to form a single scan path between the TDI and TDO pins of the host IC. During Instruction Register scan operations, the TAP exerts control via the Instruction Register shift enable (SHIFTIR) and Instruction Register Clock (CLOCKIR) signals to cause the instruction shift register to preload status information and shift data from TDI to TDO. Both the preload and shift operations occur on the rising edge of TCK; however, the data shifted out from the host IC from TDO occurs on the falling edge of TCK. The status inputs are user-defined observability inputs, except for the two least significant bits, which are always 01 for scan-path testing purposes. (The Instruction Register has a minimum length of two bits.) When activated, the RESET* input sets the instruction shift register to be set to all ones. This forces the device into the functional mode and selects the Bypass Register (or the Device Identification Register if one is present).

The instruction shadow register (Figure 3-5) consists of a series of latches, one latch for each instruction shift register bit. During an Instruction Register scan operation, the latches remain in their present state. At the end of the Instruction Register scan operation, the Instruction Register update (UPDATEIR) input updates the latches with the new instruction installed in the instruction shift register. When activated, the RESET* input sets the latches to all ones.

Data Registers

The IEEE 1149.1 standard requires two Data Registers; Boundary-Scan Register and Bypass Register, with a third, optional, Device Identification Register. Additional user-defined Data Registers may be included. The Data Registers are arranged in parallel from the primary TDI input to the primary TDO output. The Instruction Register supplies the address that allows one of the Data Registers to be accessed during a Data Register scan operation. During a Data Register scan operation, the addressed scan register receives TAP control via the Data Register shift enable (SHIFTDR) and Data Register clock (CLOCKDR) inputs to preload test response and shift data from TDI to TDO. During a Data Register scan operation, the SELECT output from the TAP (Figure 3-4) selects the output of the Data Register to drive

the TDO pin. When one scan path in the Data Register is being accessed, all other scan paths remain in their present state.

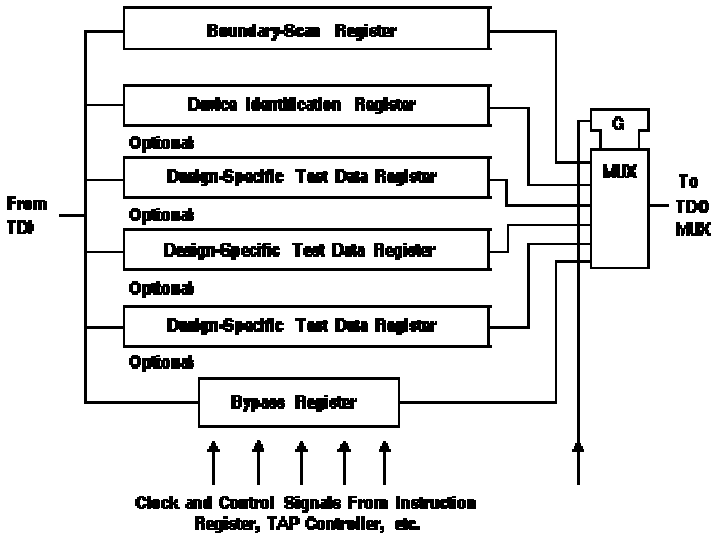

Figure 3-6. Test Data Register Architecture

Boundary-Scan Register - The Boundary-Scan Register (BSR) consists of a series of boundary-scan cells (BSCs) arranged to form a scan path around the boundary of the host IC. The BSCs provide the controllability and observability features required to perform boundary-scan testing as described in the Boundary-Scan Overview section of this chapter. Shadow latches in the BSCs, driving the NDO outputs remain in their present state during a Data Register scan operation. At the end of a Data Register scan operation, the Data Register update (UPDATEDR) input updates the shadow latches with the new boundary test pattern to be applied from the NDO outputs of the BSCs. Figure 3-7 shows a conceptual view of a Control-and-Observe BSC.

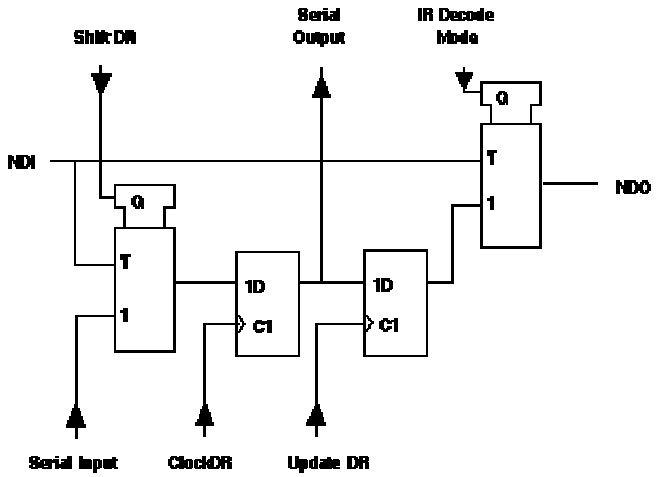

Figure 3-7. Conceptual View of a Control-and-Observe BSC

Bypass Register (Required) - The Bypass Register consists of a single scan register bit. When selected, the Bypass Register provides a single bit scan path between TDI and TDO. The Bypass Register allows abbreviating the scan path through devices that are not involved in the test. The Bypass Register is selected when the Instruction Register is loaded with a pattern of all ones to satisfy the IEEE 1149.1 Bypass instruction requirement.

Device Identification Register (Optional) - The Device Identification Register is an optional register defined by IEEE 1149.1, to identify the device's manufacturer, part number, revision, and other device-specific information. Figure 3-8 shows the bit assignments defined for the Device Identification Register. These bits can be scanned out of the Identification Register after being selected.

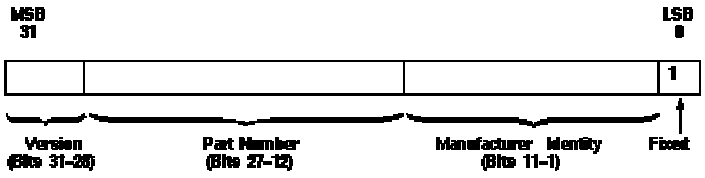

Figure 3-8. Structure of the Device Identification Register

Although the Device Identification Register is optional, IEEE 1149.1 specification has dedicated an instruction to select this register. The Device Identification Register is selected when the Instruction Register is loaded with the IDCODE instruction, which is defined by the vendor.

Manufacturer's identification codes (Bit1 through Bit11) are assigned, maintained, and updated by the EIA/JEDEC office. Any company can be added to the JEDEC Standard Manufacturer's Identification Code (Publication JEP106) by request to the JEDEC office at 202-457-4973.

IEEE 1149.1 Required Instructions

The IEEE 1149.1 standard defines nine test instructions. Of the nine instructions, three are required and six are optional. The following subsections contain brief descriptions of each required test instruction.

BYPASS Instruction

The required BYPASS instruction allows the IC to remain in a functional mode and selects the Bypass Register to be connected between TDI and TDO. The BYPASS instruction allows serial data to be transferred through the IC from TDI to TDO without affecting the operation of the IC. The bit code of this instruction is defined to be all ones by the IEEE 1149.1 standard.

SAMPLE/PRELOAD Instruction

The required SAMPLE/PRELOAD instruction allows the IC to remain in its functional mode and selects the Boundary-Scan Register to be connected between TDI and TDO. During this instruction, the Boundary-Scan Register can be accessed via a data scan operation, to take a sample of the functional data entering and leaving the IC. This instruction is also used to preload test data into the Boundary-Scan Register prior to loading an EXTEST instruction. The bit code for this instruction is defined by the user.

EXTEST Instruction

The required EXTEST instruction places the IC into an external boundary test mode and selects the Boundary-Scan Register to be connected between TDI and TDO. During this instruction, the Boundary-Scan Register is accessed to drive test data off-chip via the boundary outputs and receive test data off-chip via the boundary inputs. The bit code of this instruction is defined by the 1149.1 standard to be all zeros.

IEEE 1149.1 Optional Instructions

The following subsections contain brief descriptions of the optional IEEE 1149.1 instructions.

INTEST Instruction

The optional INTEST instruction places the IC in an internal boundary test mode and selects the Boundary-Scan Register to be connected between TDI and TDO. During this instruction, the Boundary-Scan Register is accessed to drive test data on-chip via the boundary inputs and receive test data on-chip via the boundary outputs. The bit code of this instruction is defined by the user.

RUNBIST Instruction

The optional RUNBIST instruction places the IC into a self-test mode, enables a comprehensive self-test of the IC's core logic, and selects a user-specified Data Register to be connected between TDI and TDO. During this instruction, the boundary outputs are controlled so that they cannot interfere with neighboring ICs during the RUNBIST operation. Also, the boundary inputs are controlled so that external signals cannot interfere with the RUNBIST operation. The bit code of this instruction is defined by the user.

CLAMP Instruction

The optional CLAMP instruction sets the outputs of an IC to logic levels determined by the contents of the Boundary-Scan Register and selects the Bypass Register to be connected between TDI and TDO. Before you load this instruction, you can preset the contents of the Boundary-Scan Register with a SAMPLE/PRELOAD instruction. During this instruction, data can be shifted through the Bypass Register from TDI to TDO without affecting the condition of the outputs. The bit code of this instruction is defined by the IC designer.

HIGHZ Instruction

The optional HIGHZ instruction sets the three-state outputs of an IC to a disabled state and selects the Bypass Register to be connected between TDI and TDO. During this instruction, data can be shifted through the Bypass Register from TDI to TDO without affecting the condition of the IC outputs. The bit code of this instruction is defined by the IC designer.

IDCODE Instruction

The optional IDCODE instruction allows the IC to remain in its functional mode and selects an optional Device Identification Register to be connected between TDI and TDO. The Identification Register (see Figure 3-8) is a 32-bit shift register containing information regarding the IC manufacturer, device type, and version code. Accessing the Identification Register does not interfere with the operation of the IC. Also, access to the Identification Register should be immediately available, via a TAP data scan operation, after power-up of the IC or after the TAP has been reset using the optional TRST* pin or by issuing a Test-Logic-Reset instruction. The bit code of this instruction is defined by the IC device designer.

USERCODE Instruction

The optional USERCODE instruction allows the IC to remain in its functional mode and selects a User Data Register to be connected between TDI and TDO. The User Data

Register is an optional 32-bit shift register containing user-defined information about the IC. Accessing the User Data Register does not interfere with the operation of the IC. The bit code of this instruction is defined by the IC designer.

Obtaining IEEE Std 1149.1-1990

To learn more about the IEEE 1149.1 standard, please refer to the publication, Standard Test Access Port and Boundary-Scan Architecture, IEEE Std 1149.1-1990 (includes IEEE Std 1149.1a-1993). This document is available through IEEE (1-800-678-IEEE, 908-981-1393), and is available on CD-ROM.

Using DFT in ASICs

The concern most often voiced by Application Specific Integrated Circuit (ASIC) users is that of testability. This chapter is intended to provide an understanding of ASIC testability that can be used for developing test strategies when designs are being initiated.

Design-For-Test Considerations

Designing testability into any circuit affects the hardware to some degree. Additional logic will probably have to be added. This additional logic increases the amount of silicon required to implement the design. The savings from enhanced testability do not usually show up until the cycle time and testing cost of the circuit and its end system are analyzed.

Fault simulation is an important part of designing for testability. This technique enables you to evaluate your test patterns to determine whether these patterns detect faults. Faults may occur during either the design-tooling stage or the circuit-fabrication stage. A fault simulator uses fault models, such as a node shorted to power (stuck-at-one) or a node shorted to ground (stuck-at-zero), and compares the response of a fault-free circuit with the response of a faulty circuit. If the response of the fault-free circuit is different than the response of the faulty circuit, then the test patterns have detected the fault.

By faulting all the nodes in the circuit, the fault simulator produces the test pattern fault grade or fault coverage. The fault coverage is the percentage of faults detected among the total faults. The higher the fault coverage, the better the test pattern set separates a faulty circuit from a fault-free circuit. After determining which faults have not been detected by the current set of test patterns, you can generate additional test patterns to detect those faults.

Adoption of design-for-test principles early in the design process ensures the maximum testability for the minimum effort. These guidelines emphasize that test is a part of the design flow, not a process that is done at the end of the design cycle.

Three basic elements must come together to make a successful ASIC circuit:

- Logic design including schematic capture, library selection, synthesis, and simulation
- Logic testability including fault-detection and test-application criteria within predefined cost and time scale budgets
- Vendor's manufacturing capability including processing and packaging

The Need for Testability

Most engineers involved in the design of ASIC devices are familiar with the trade-offs between gate arrays, standard cells, and full custom devices. They are also familiar with the vendor selection process. This aspect of test capability and testability is often overlooked.

Testability could be ignored when typical designs were only a few thousand gates. These designs were implemented first and then turned over to a test engineer or to a vendor to

create a test program for production. As design complexities increased, this approach to testing became futile. Successful high-density ASIC design and manufacturing demand that circuits be put together with testability incorporated into the design process.

Although testability imposes additional constraints in the design phase, design verification and test can be unmanageable if ignored until the design is completed and testability is handled as a post-design insertion. In fact, the design constraints are overwhelmingly balanced by improved testability, which adds value to the device throughout manufacturing and system life.

Test-Time Cost

Test cost, as it relates to time, is a simple calculation. Most commercial testers cost between $2 million and $3 million. Under normal circumstances, the tester depreciation, plant, operator, and support personnel costs are between 10 and 20 cents per test second.

Brute-force test approaches often generate a large number of test patterns. Since test patterns are run at multiple power supply values and possibly at multiple temperatures, inefficient pattern sets can severely impact the test costs of a complex ASIC device.

Time to Market

Surveys indicate that 40 percent of the development cycle time for an ASIC device is required for test insertion and test pattern generation. This figure is expected to increase as device complexity increases. The intent of a design-for-test strategy is to achieve high fault detection test programs in reduced time (Figure 4-1). The obvious cycle-time reductions result from designed-in testability (elimination of iterative redesigns resulting from poor design practices), and from automatic test pattern generation (ATPG).

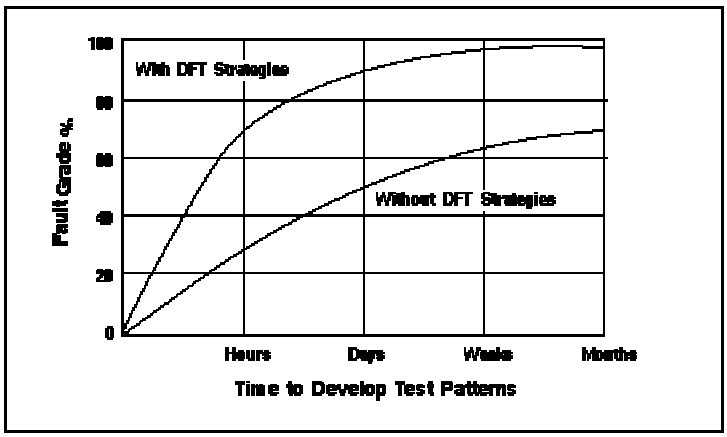

Figure 4-1. Fault Grade Versus Development Time

Figure 4-2 shows the economic relationship between time-to-market and system manufacturing and field maintenance costs. Point 1 represents the case where market entry timing forces a constraint on the development time. Since 40 percent of this time is expended in inserting testability, the temptation is to rush to market with devices that are not completely testable or tested. The result is a higher than desirable manufacturing and field-maintenance cost. Point 2 represents the case where DFT and ATPG techniques are employed to develop devices that are completely tested. This situation allows an economic optimum that is more favorable to long-term manufacturing and maintenance costs.

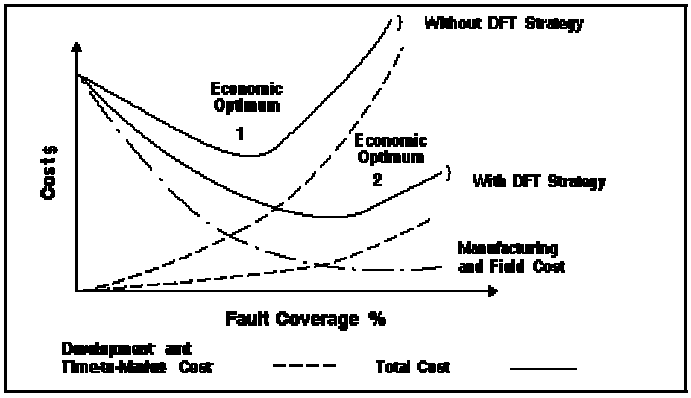

Figure 4-2. Economic Trade Off for a Testable Design

A less obvious result of a DFT strategy is the reduction of debug time. You, as designer, must make certain assumptions about system requirements. Often a new device does not work in the system environment and requires debugging. If the device is designed for controllability and observability access, the debugging process is enhanced. Conversely, if these two features are overlooked, debugging and manufacturing can be significantly harder to accomplish, if not impossible. Oscilloscopes and logic analyzers are not very effective in debugging systems utilizing complex ASIC devices in a surface-mount environment.

Fault Coverage and Cost of Ownership

Figure 4-3 shows the trade-off between time-to-market and manufacturing and field-maintenance costs. The horizontal factor on this figure is fault coverage. The relationship between fault coverage and device defect level is well documented.

Figure 4-3 is a plot of the relationship modeled by T. W. Williams for fault coverage of 90 percent or greater.

The Williams model is:

$$D = [1 - Y^{(1-T)}] \times 100$$

Where:

- D = Defect level in percent
- Y = Theoretical functional process yield
- T = Fault coverage of the test program used

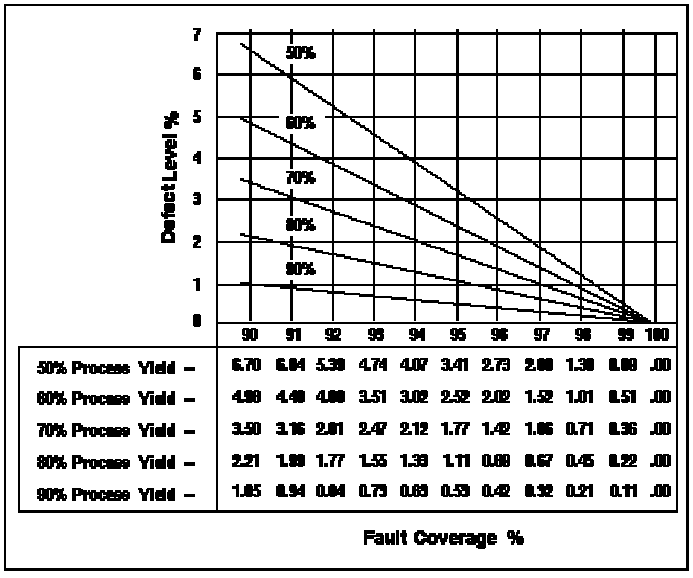

Figure 4-3. Defect Level Versus Fault Coverage

To explore the Williams model briefly, assume that the ASIC vendor has a silicon and assembly process yield that is 70 percent. If the fault grade of the test program is also 70 percent, the defect level is projected to be 10.1 percent or 101,000 ppm (This is outside the limits of the chart and was calculated.). At a fault grade of 90 percent, the defect level is projected to be 3.5 percent or 35,000 ppm.

A study of the model shows that the process yield becomes an insignificant term when the fault coverage of the test program is very close to 100 percent.

Motorola and Delco performed a study in 1980 that supports the Williams model. Their experimental results are shown in Figure 4-4. A fault coverage of 99.9 percent was required to obtain defect levels in the range of 100 ppm.

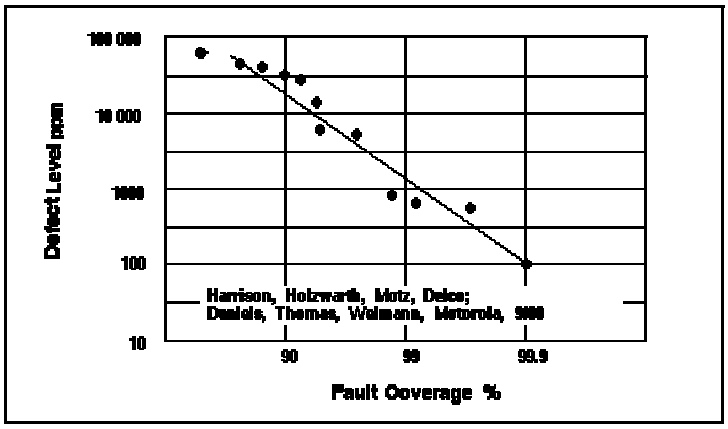

Figure 4-4. Motorola and Delco Study Results

Figure 4-5 shows the maximum allowable ASIC defect rate to achieve a goal PCB defect rate as a function of the number of ASIC devices per board assembly. Note that for multiple-device PCB designs, a goal of 500 ppm requires ASIC defect levels in the range of 100 to 200 ppm.

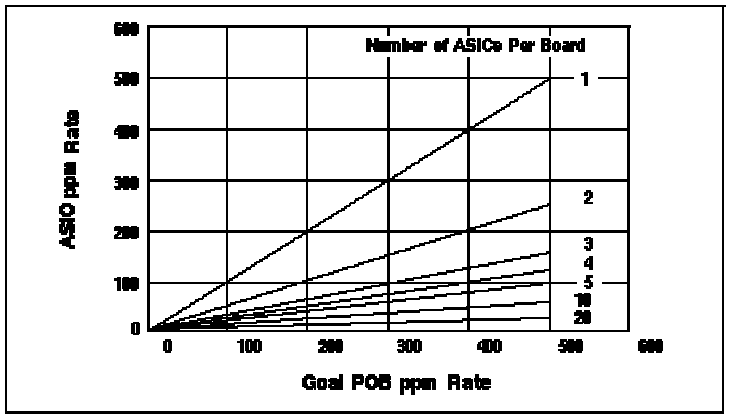

Figure 4-5. ASIC ppm Versus PCB ppm Rate

Theoretical and experimental studies conclude that a high-fault-grade test-pattern set is required for low defect level ASIC devices. This type of pattern set is nearly impossible to obtain through brute force. The requirements for a high-fault-grade pattern set are:

- ATPG tool
- Fault grader
- A testable design meeting the constraints of the ATPG tool

As stated earlier, a design-for-test strategy has performance and area costs. Now the cost of new tools has been added. Benefits such as lower test costs and reduced time to market have been mentioned. These benefits are real, but often hard to quantify. Reduced cost of ownership is another major benefit and is easy to quantify.

Figure 4-6 shows what is commonly referred to as the "cost-of-ownership, order-of-magnitude relationship". It shows that each company has a cost associated with finding a defect in a packaged device before it has entered the assembly process. This cost can be calculated easily. The cost of finding a defective device after assembly onto a PCB is an order of magnitude more than before assembly. This continues until the cost to discover a defective device in a system at a customer's site is three orders of magnitude higher than that of discovery before assembly on to a PCB. The lowest cost of ownership is to find defective units before they are shipped from the vendor.

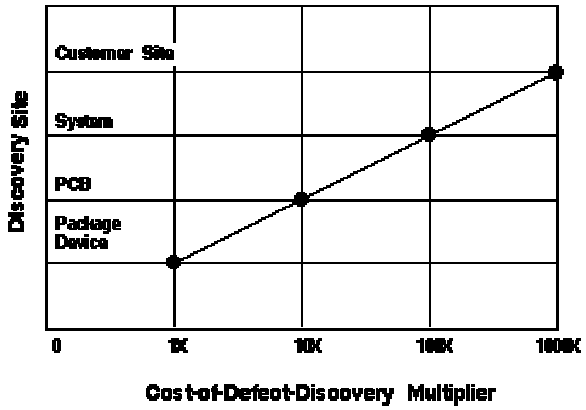

Figure 4-6. Cost of Ownership

The previous discussions lead to the conclusion that the lowest cost of ownership can be obtained by providing the ASIC vendor with an efficient, high-fault-detection set of test vectors. These DFT methodologies provide lower cost of ownership with the added benefit of reduced time to market.

Developing Testability Strategies

The following strategies step you through the process of design for test.

1. Select a technology.

   When selecting a technology or vendor, make sure there is enough performance and gate-count margin to allow the insertion of testability.

2. Commit to testability design practices.

   Testability design practices work. Commit to use them, and review them with the design team before the design begins. Add a testability commitment to the design requirements document developed for the design project. Make testability audits part of the design review process.

3. Establish a fault-grade requirement.

   The fault-grade requirement can usually be provided by the manufacturing or quality organization. Establish this requirement before the first design review. Add the fault-grade requirement to the design requirements document. This requirement drives many of the decisions that follow in the development of the test strategy.

   Many companies consider the fault-grade requirement to be an index of device cost of ownership. Failure to achieve it costs profits throughout the lifetime of the device.

4. Decide if IEEE Std 1149.1 will be a system requirement.
   When implemented in an ASIC device, IEEE Std 1149.1 allows test of the interconnect between devices on a PCB through a four-pin test bus. If IEEE 1149.1 is selected, the four dedicated test pins can also be used to control core logic test techniques such as Built-In Self-Test (BIST), internal scan, on-chip emulation, and boundary-scan through the Test Access Port (TAP) to the Instruction Register.
5. Select an ASIC testability approach based on gate density.
   - Designs with fewer than 10K gates
     Designs with fewer than 10K gates are not generally complex enough to require structured test approaches. The overhead impact is usually too high to justify them. Nonstructured, good design practices are usually sufficient.
   - Designs with more than 10K gates but less than 20K gates
     Structured techniques should be considered for designs in this density. Nonstructured, good design practices are probably sufficient for highly combinatorial circuits without memory. Structured approaches should be considered as complexity is increased by the addition of sequential circuits, feedback, and memory. Consider boundary-scan testing for reduced cycle times and high fault grades.
   - Designs with more than 20K gates
     The complexity of circuits this dense usually requires structured approaches to achieve high fault grades. At this density, it is often hard to control or observe deeply embedded circuits. The overhead associated with structured testability approaches is acceptable at this density.
6. Choose structured tools.
   Scan testing is typically the preferred structured approach for sequential logic. The available scan choices are:
   - Clocked scan
   - Multiplexed flip-flop scan
   - Level-sensitive scan-design (LSD)
   - Parallel scan chains
   - Partial scan
7. Establish a diagnostic functional-pattern set to expedite debug.
   This is an important step in decreasing the time to market for an ASIC device. The purpose of this pattern set is to isolate circuitry for analysis.
8. Generate high fault-grade test patterns.
   The fault grade of a test pattern set determines the best possible quality level attainable with that set of patterns.
9. Simulate test patterns and timing.
   Two types of simulation are required during development. Logic simulation verifies both functionality and performance of the device. Test pattern simulation produces the information needed to verify the test patterns in a tester environment.

Figure 4-7 contains a flow chart that steps through the design-for-test process.

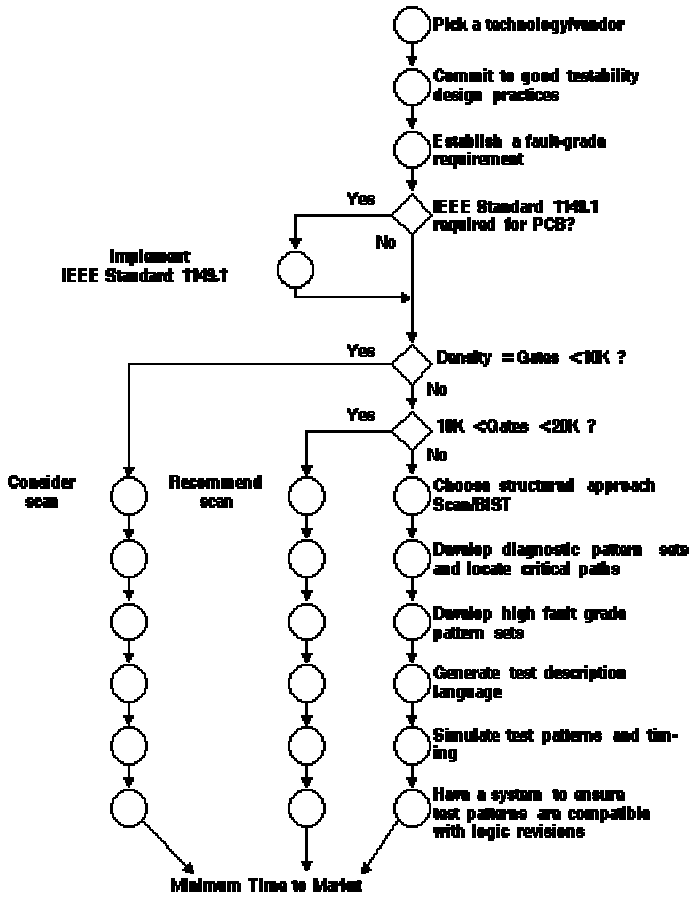

Figure 4-7. Testability Development Flow

Data Formats

Several data formats have emerged to make IEEE 1149.1 successful and well-supported by tools. This section discusses the most widely accepted data formats that support IEEE 1149.1 -- BSDL, HSDL, and SVF.

Boundary-Scan Description Language (BSDL)

In 1990, the IEEE 1149.1 standard was approved and implementation of the standard accelerated. As more people became aware of and used the standard, the need for a standard method for describing IEEE 1149.1-compatible devices was recognized. The IEEE 1149.1 working group established a subcommittee to develop a device description language to address this need.

The subcommittee has since developed and approved an industry standard language called Boundary-Scan Description Language (BSDL). BSDL is a subset of VHDL (VHSIC Hardware Description Language) that describes how IEEE 1149.1 is implemented in a device and how it operates. BSDL captures the essential features of any IEEE 1149.1 implementation. BSDL was approved in 1994 as IEEE Std. 1149.1b.

The IEEE 1149.1 is a structured design-for-test approach well suited for tools and automation. Tools developed to support the standard can control the TAP (Test Access Port) if they know how the boundary-scan architecture was implemented in the device. Tools can also control the I/O pins of the device. BSDL provides a standard machine and human-readable data format for describing how IEEE 1149.1 is implemented in a device.

How BSDL is Used

Many IEEE 1149.1 tools already on the market support BSDL as a data input format. These tools offer different capabilities to customers implementing IEEE 1149.1 into their designs including board interconnect Automatic Test Pattern Generation (ATPG) and Automatic Test Equipment (ATE).

When you use tools that support BSDL you can often obtain BSDL from your semiconductor vendor. This can result in significant time and cost savings.

Teradyne estimates that to create in-circuit test patterns for a leading microprocessor normally may require as much as seven weeks time:

- One week to study the device
- Four weeks to develop in-circuit test patterns
- Two weeks to verify the patterns on ATE

The development cost estimate for this approach is $14,000.

If the microprocessor supports IEEE 1149.1, and the BSDL is supplied by the vendor, the time to develop in-circuit test patterns is less than two hours (less than $100) using today's tools.

Elements of BSDL

A BSDL description for a device consists of the following elements:

- Entity descriptions
- Generic parameter
- Logical Port description
- Use statements
- Pin Mapping(s)
- Scan Port identification
- Instruction Register description
- Register Access description
- Boundary Register description

Entity Descriptions -- The entity statement names the entity, such as the device name (e.g., SN74ABT8245). An entity description begins with an entity statement and terminates with an end statement.

```
entity XYZ is
  {statements to describe the entity go here}
end XYZ
```

Generic Parameter -- A generic parameter is a parameter that may come from outside the entity, or it may be defaulted, such as a package type (e.g., "DW").

```
generic (PHYSICAL_PIN_MAP : string := "DW");
```

Logical Port Description -- The port description gives logical names to the I/O pins (system and TAP pins), and denotes their nature such as input, output, bidirectional, and so on.

```
port (OE:in bit;
 Y:out bit_vector(1 to 3);
 A:in bit_vector(1 to 3);
 GND, VCC, NC:linkage bit;
 TDO:out bit;
 TMS, TDI, TCK:in bit);
```

Use Statements -- The use statement refers to external definitions found in packages and package bodies.

```
use STD_1149_1_1994.all;
```

Pin Mapping(s) -- The pin mapping provides a mapping of logical signals onto the physical pins of a particular device package.

```
attribute PIN_MAP of XYZ : entity is
PHYSICAL_PIN_MAP;
constant DW:PIN_MAP_STRING:=
  "OE:1, Y:(2,3,4), A:(5,6,7), GND:8, VCC:9, "&
       "TDO:10, TDI:11, TMS:12, TCK:13, NC:14";
```

Scan Port Identification -- The scan port identification statements define the device's TAP.

```
attribute TAP_SCAN_IN of TDI : signal is TRUE;
attribute TAP_SCAN_OUT of TDO : signal is TRUE;
attribute TAP_SCAN_MODE of TMS : signal is TRUE;
attribute TAP_SCAN_CLOCK of TCK : signal is (50.0e6,
          BOTH);
```

Instruction Register Description -- The Instruction Register description identifies the device-dependent characteristics of the Instruction Register.

```
attribute INSTRUCTION_LENGTH of XYZ : entity is 2;
attribute INSTRUCTION_OPCODE of XYZ : entity is
    "BYPASS (11), "&
    "EXTEST (00), "&
    "SAMPLE (10) ";
attribute INSTRUCTION_CAPTURE of XYZ : entity is
    "01";
```

Register Access Description -- The register access defines which register is placed between TDI and TDO for each instruction.

```
attribute REGISTER_ACCESS of XYZ : entity is
  "BOUNDARY (EXTEST, SAMPLE), "&
  "BYPASS (BYPASS) ";
```

Boundary Register Description -- The Boundary Register description contains a list of boundary-scan cells, along with information regarding the cell type and associated control.

```
attribute BOUNDARY_LENGTH of XYZ : entity is 7;
attribute BOUNDARY_REGISTER of XYZ : entity is
  "0 (BC_1, Y(1), output3, X, 6, 0, Z), "&
  "1 (BC_1, Y(2), output3, X, 6, 0, Z), "&
  "2 (BC_1, Y(3), output3, X, 6, 0, Z), "&
  "3 (BC_1, A(1), input, X), "&
  "4 (BC_1, A(2), input, X), "&
  "5 (BC_1, A(3), input, X), "&
  "6 (BC_1, OE, input, X), "&
  "6 (BC_1, *, control, 0)";
```

Verifying BSDL Accuracy

Creating a BSDL file that is syntactically and semantically correct is only the beginnig. A syntactically and semantically correct BSDL file can still contain descriptive errors and result in time-consuming debugging of board-level tests and diagnostics. A BSDL file must be valiadated (compared) against the silicon.

Potential BSDL Errors

As with any hand-entered data, typographical errors potentially exist. BSDL contains many commas and semicolons that contribute error possibilities. Fortunately, syntax and semantics errors can easily be identified and corrected using syntax and semantics checking tools during the authoring process.

Other common errors are:

- Wrong pinout
- Wrong cell types
- Wrong Boundary-Scan Register order
- Wrong Boundary-Scan Register length
- Wrong Instruction Register opcodes
- Wrong control cell locations
- Wrong control cell disable value
- Wrong I/O pin control cell
- Wrong ID Code value
- Wrong Capture-IR value

Typographical errors or device documentation errors can result in implementation errors.

How to Receive the BSDL Specification

Contact the IEEE Standards Department (1-800-678-IEEE) and refer to IEEE P1149.1b to receive a copy of the Boundary-Scan Description Language (BSDL) specification.

Obtaining BSDL for TI Catalog Devices

Texas Instruments makes catalog device BSDL files available via ftp download. To download a BSDL file, go to TI's IEEE 1149.1 silicon product availability listing and click on "download BSDL now" to start the ftp download process.

Hierarchical Scan Description Language (HSDL)

Texas Instruments developed the Hierarchical Scan Description Language (HSDL) to complement BSDL, using the same subset of VHDL statements as BSDL. HSDL picks up where BSDL stops to describe additional attributes of IEEE 1149.1 devices and how IEEE 1149.1 devices are connected at the board and system level.

HSDL uses the BSDL entity and package in new ways. Entities in HSDL are used to describe modules as well as devices. A module is any level of architecture above the device level, including boards, multichip modules, backplanes, subsystems, and systems. In addition, HSDL provides two new packages used to indicate that an entity is an HSDL device or module.

BSDL is great for describing how IEEE 1149.1 is implemented in a device, but stops there. HSDL provides a method for describing how IEEE 1149.1 devices are connected at the board, module, and system levels. HSDL serves two needs not addressed by BSDL:

- Describes the test bus interconnections of IEEE 1149.1 at the board or module level
- Improves ease-of-use and reduces risk during interactive design debug and verification
- Allows descriptions of boards with dynamic and reconfigurable architectures

HSDL Module Statements

HSDL module statements use much of the same syntax as BSDL. New statements have been added to describe the members and scan paths of the module and to simplify interactive use.

- Entity descriptions

- Generic parameter
- Logical Port description
- Use statements
- [Optional Module descriptions]
- [Optional Port description(s)]
- Pin Mapping(s)
- Scan Port identification
- [Optional Member description(s)]
- [Optional Bus description(s)]
- Path description
- [Optional Member Connections]
- [Optional Constraint Description(s)]
- [Optional Design Warning]

Entity Descriptions -- The entity statement names the entity, such as the module name (e.g., BOARD). An entity description begins with an entity statement and terminates with an end statement.

```
entity BOARD is
  {statements to describe the entity go here}
end BOARD;
```

Generic Parameter -- A generic parameter may come from outside the entity or it may be defaulted, such as a package type (e.g., "UNDEFINED").

```
generic (PHYSICAL_PIN_MAP : string := ("UNDEFINED")
```

Logical Port Description -- The port description gives logical names to the I/O pins (system and TAP pins), and denotes their nature such as input, output, bidirectional, and so on.

```
port (TDI:in bit;
  TDO:out bit;
  TMS:in bit;
  TCK:in bit);
```

Use Statements -- The use statement refers to external definitions found in packages and package bodies.

```
use STD_1149_1_1994.all;
use HSDL_module.all;
```

Pin Mapping(s) -- The pin mapping provides a mapping of logical signals onto the physical pins of a particular entity.

```
attribute PIN_MAP of BOARD : entity is
PHYSICAL_PIN_MAP;
    constant PINOUT1 : PIN_MAP_STRING :=
    "TDI:1, TDO:2, TMS:3, TCK:4, GND:5";
```

Scan port Identification -- The scan port identification statements define the entity's TAP.

```
attribute TAP_SCAN_IN of TDI : signal is TRUE;
attribute TAP_SCAN_OUT of TDO : signal is TRUE;
attribute TAP_SCAN_MODE of TMS : signal is TRUE;
attribute TAP_SCAN_CLOCK of TCK : signal is (5.0e6,
        LOW);
```

Members Description (Optional) -- Members represent devices or other modules that are on the module. Usually members represent components, but some boards may contain scannable daughtercards, card slots, etc. that require modules to describe them.

```
attribute MEMBERS of BOARD : entity is
  "U1 (XYZ1, DW),"&
  "U2 (XYZ2, DW), ";
```

Bus Composition (Optional) -- Buses in an HSDL module can be built of module buses, member module buses, member device buses, and member device test registers.

```
attribute BUS_COMPOSITION of BOARD : entity is
  "bus1[4] (U1.Boundary[3,0]), "&
  "bus2[4] (U2.Boundary[3,0]), ";
```

Path Description -- Module paths are intended to describe the netlist of TAP signals (scan paths) on the board.

```
constant boardpath1 : STATIC_PATH :=
   "U1, U2";
end BOARD;
```

How to Receive the HSDL Specification

Texas Instruments makes the Hierarchical Scan Description Language (HSDL) specification available through an electronic bulletin board (214-575-6480).

Serial Vector Format (SVF)

Serial Vector format, commonly referred to as SVF, was jointly developed by Texas Instruments and Teradyne in 1991. SVF is a standard ASCII format for expressing test patterns that represent the stimulus, expected response, and mask data for IEEE 1149.1-based tests. The need for SVF arose from the desire to have vendor-independent IEEE 1149.1 test patterns that are transportable across a wide selection of simulation software and test equipment -- from design verification through field diagnostics.

Boundary-scan test execution is controlled by the sequencing of TAP signals on the pins of the devices. Each device's behavior is determined solely by the states of its TAP pins. Boundary-scan tools must maintain knowledge of the sequences required to exert certain behaviors within a device and where that device is located down the serial scan path.

SVF controls the IEEE 1149.1 test bus using commands that transition the TAP from one steady state to another. Rather than describe the explicit state of the IEEE 1149.1 bus on every TCK cycle, SVF describes it in terms of transactions conducted between stable states. for instance, the process of scanning in an instruction is described merely in terms of the data involved and the desired stable state to enter after the scan has been completed. The Capture, Update, Pause, etc. states are inferred rather than explicitly represented. the data to be scanned in, expected data out, and compare mask are all grouped in an easily understandable manner. A command is provided to support deterministic navigation of TAP states where required.

In addition to supporting higher level depiction of scan operations, SVF also supports combined serial and parallel operations. This allows SVF to accommodate ATE environments where some stimulus/response is handled via parallel I/O, and serial signals are accessed via an IEEE 1149.1-control environment.

SVF also supports the concept of scan offsets. Offsets allow a test to be applied to a component or cluster of logic embedded in the middle of a scan path. For example, assume a device exists in multiple instances on a board. Serially applied tests were generated by the designer that are available in SVF format. to reuse this test, it is necessary to put all other devices on the scan path into the bypass mode. The IEEE 1149.1 test controller must therefore comprehend the number of Instruction Register bits before and after the target device. Once in bypass, the devices introduce Data Register bits before and after the target device.

SVF allows a header and trailer to be defined once which maintains the Instruction Register and Data Registers of the non-targeted devices in the desired bypass state. No modifications are required to the SVF for the device. If the same test were targeted towards

another device downstream in the scan path, this would be accommodated merely by changing the headers and trailers.

The offset approach is capable of installing any Instruction and Data Register stimulus, provided these values are constant for the entire process of applying the SVF device sequence.

SVF Structure

The SVF file is defined as an ASCII file that consists of a set of SVF statements. Statements are terminated by a semicolon (;) and may continue for more than one line. The maximum number of ASCII characters per line is 256. SVF is not case sensitive, and comments can be inserted into an SVF file after an exclamation point (!) or a pair of slashes (//).

Each statement consists of a command and parameters associated with that specific command. Commands can be grouped into three types: state commands, offset commands, and parallel commands.

State commands

State commands are used to specify how the test sequences will traverse the IEEE 1149.1 TAP state machine. The following state commands are supported:

SDR	Scan Data Register
SIR	Scan Instruction Register
ENDDR	Define end state of DR scan
ENDIR	Define end state of IR scan
RUNTEST	Enter Run-Test/Idle state
STATE	Go to specified stable state
TRST	Drive the TRST line to the designated level

SDR performs an IEEE 1149.1 Data Register scan. *SIR* performs an IEEE 1149.1 Instruction Register scan. *ENDDR* and *ENDIR* establish a default state for the bus following any Data Register scan or Instruction Register scan, respectively. *RUNTEST* goes to Run-test/Idle state for a specific number of TCKs. for each of the above commands, a default path through the state machine is used. Each of these commands also terminates in a stable, nonscannable state.

STATE places the bus in a designated IEEE 1149.1 stable state. *TRST* activates or deactivates the optional test reset signal of the IEEE 1149.1 bus.

Offset Commands

Offset commands allow a series of SVF commands to be targeted towards a contiguous series of points in the scan path. Examples would be a sequence for executing self-test on a device, or a cluster test where al devices involved in the cluster test are grouped together. The following offset commands are supported:

HDR	Header data for data bits
HIR	Header data for instruction bits
TDR	Trailer data for data bits
TIR	Trailer data for instruction bits

HDR specifies a particular pattern of data bits to be padded onto the front of every data scan. *HIR* specifies the same for the front of every Instruction Register scan. These patterns

need only be specified once and are included on each scan unless changed by a subsequent *HDR*, *HIR*, *TDR* or *TIR* command.

Parallel Commands

Parallel commands are used to map and apply the following commands:

PIO	Specifies a parallel test pattern
PIOMAP	Designates the mapping of bits in the PIO command to logical pin names

Parallel commands allow SVF to combine serial and parallel sequences. *PIOMAP* commands are used by parallel I/O controllers to map data bits in the command into parallel I/O channels using the ASCII logical pin name as a reference. The *PIO* command specifies the execution of a parallel pattern application/sample. SVF does not specify any other properties of parallel I/O such as drive, levels, or skew.

Default State Transitions

SVF uses names for the TAP states that are similar to the IEEE 1149.1 TAP state names. Below is a list of SVF equivalent names for the TAP states.

Test-logic-Reset	[RESET]
Run-Test/Idle	[IDLE]
Select-DR-Scan	[DRSELECT]
Capture-DR	[DRCAPTURE]
Shift-DR	[DRSHIFT]
Pause-DR	[DRPAUSE]
Exit1-DR	[DREXIT1]
Exit2-DR	[DREXIT2]
Update-DR	[DRUPDATE]
Select-IR-Scan	[IRSELECT]
Capture-IR	[IRCAPTURE]
Shift-IR	[IRSHIFT]
Pause-IR	[IRPAUSE]
Exit1-IR	[IREXIT1]
Exit2-IR	[IREXIT2]
Update-IR	[IRUPDATE]

Below is a listing to identify sample default paths taken when transitioning from one state to a specified new state. For example, if the current state is RESET and you select DRPAUSE as the end state, the TAP moves from RESET through IDLE, DRSELECT, DRCAPTURE, DREXIT1 to DRPAUSE. You only have to specify the current and end states and not each intermediate step.

Current State	End State	State Path
RESET	RESET	RESET
RESET	IDLE	RESET
		IDLE
RESET	DRPAUSE	RESET
		IDLE

		DRSELECT
		DRCAPTURE
		DREXIT1
		DRPAUSE
RESET	IRPAUSE	RESET
		IDLE
		DRSELECT
		IRSELECT
		IRCAPTURE
		IREXIT1
		IRPAUSE

SVF Example

The following is an example SVF file:

```
! Begin Test Program
! Disable Test Reset line
  TRST OFF;
! Initialize UUT
  STATE RESET;
! End IR scans in DRPAUSE
  ENDIR DRPAUSE;
! 24 bit IR header
  HIR 24 TDI (FFFFFF);
! 3 bit DR header
  HDR 3 TDI (7) TDO (7) MASK (0);
! 16 bit IR trailer
  TIR 16 TDI (FFFF);
! 2 bit DR trailer
  TDR 2 TDI (3);
! 8 bit IR scan, load BIST seed
  SDR 16 TDI (ABCD);
! RUNBIST for 95 TCK Clocks
  RUNTEST 95 TCK ENDSTATE IRPAUSE
! 16 bit DR scan, check BIST status
  SDR 16 TDI (0000) TDO (1234) MASK (FFFF);
! Enter Test-Logic-Reset
  STATE RESET;
! End Test Program
```

The test begins by deasserting TRST. The DRPAUSE state is established as the default end state for instruction scans and data scans. Twenty four bits of header and sixteen bits of trailer data are specified for Instruction Register scans. No status bits are checked. three bits of header data and two bits of trailer data are specified for Data Register scans.

In the example above, a single device in the middle of the scan is targeted. Notice from the 24-bit IR header (3x8-bit IR) and the 3-bit DR header (3x1-bit DR) that the targeted device has three devices before it in the scan path. From the 16-bit IR trailer (2x8-bit IR) and the 2-bit DR trailer (2x2-bit DR), the targeted device has 2 devices following it in the scan path. After the header and trailer offsets are established all subsequent scans are the concatenation of the header, scan data, and trailer bits. the targeted device supports BIST, which is initialized by scanning a hex ABCD into the selected Data Register. the BIST in the targeted device is then executed by entering the Run-Test/Idle state for 95 clock cycles. Next,

the BIST result is scanned out and the status bits compared against a deterministic value to determine pass/fail.

How to Receive the SVF Specification

Texas Instruments makes the Serial Vector Format (SVF) specification available through an electronic bulletin board (214-575-6480).

Suggested Design Flow

The designer of any new product must plan for testing at any time in the life cycle of the product. This process is called design for test (DFT). The test methodology, defined by the IEEE 1149.1 standard, is used to ease problems associated with both product development and all levels of testing. Once boundary-scan architecture is built into a device, tests developed for that device can be reused after it is in a product. Figure 6-1 summarizes the concerns applicable to any new product design.

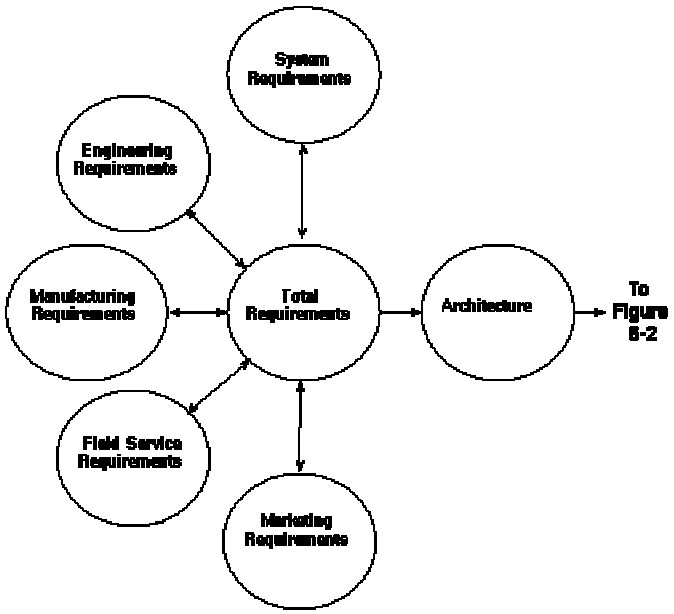

Figure 6-1. Initial DFT Concerns

Test Requirements

Before implementing an boundary-scan test scheme, the designer must define test requirements for the following phases of the product life cycle:

- Design verification
- Manufacturing test
- Field test (can also be used for incoming inspection)

These studies must ensure that the necessary controls are designed into the hardware to support the following requirements:

- During design verification, an engineer requires controllability of the circuitry, both internal to the IC and at board and system level. Design engineers need controllability and observability of every logic node in the system. If this proves to be unrealistic, reasonable coverage is achievable with partial scan in a boundary-scan design.

- Manufacturing test may include interconnect testing and functional logic validation. •Field or embedded testing typically may only include the ability to report faults and to isolate a fault to a field repairable unit (board or subsystem).
- Field or embedded testing typically may only include the ability to report faults and to isolate a fault to a field repairable unit (board or subsystem)

For both manufacturing test and field test, IEEE 1149.1 logic and test vectors can be used to test clusters or blocks of logic.

Internal IC testing at the board or system level requires a subset of the complete IC logic verification patterns to validate basic IC logic functions. Each level of hierarchical testing has a different set of fault characteristics. IEEE 1149.1 logic can be used throughout the testing life cycle of the product.

Built-In Self-Test (BIST) Methodology

BIST usually consists of special circuitry built as part of an IC's internal design. BIST tests typically perform these functions:

- Pseudo-random Pattern Generation (PRPG)
- Parallel Signature Analysis (PSA)
- Serial Signature Analysis (SSA)
- State machine-based generation/comparison

IEEE 1149.1 commands can initiate BIST tests and return the test results.

Internal Scan Test Methodology

Many high-density microprocessors and ASICs now include internal IEEE 1149.1-compliant scan circuitry. High pin count and fine-pitch packages limit physical access to package pins and devices with high gate count limit observation of the internal logic states. Internal scan, either partial or full, can do the following:

- Provide access to internal registers
- Partition the design
- Reduce the design to combinatorial networks surrounded by scannable registers
- Provide fast, efficient testing of internal logic

Internal scan also allows on-chip emulation control and circuit observability within a device to provide design verification. The emulation capability can also aid in software development to control highly complex functions designed into a device.

Design Effort

Figure 6-2 summarizes the various steps involved in ASIC and board design.

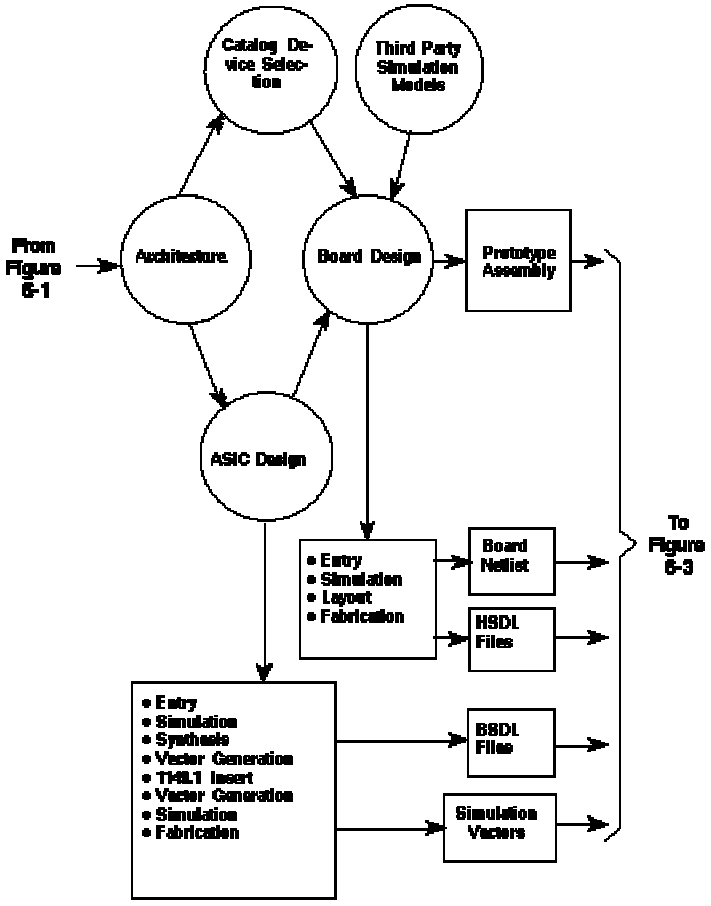

Figure 6-2. Designing Testability for ASICs and Boards

IC Design Implementation

Synthesis of ASICs has become a popular approach to ASIC design. ASIC designers faced with DFT for the first time are concerned about lack of knowledge and the additional design time involved. Today, there are several vendors supporting IEEE 1149.1 synthesis into ASICs. For example, the Synopsys Test Compiler family offers IEEE 1149.1 synthesis to generate these functions:

- Test Access Port (TAP) controller
- Control circuitry
- Instruction Register
- Instruction decode logic
- Bypass Register
- Boundary-Scan Register

The compiler also automatically generates test vectors for IEEE 1149.1-compliant designs. This means that a designer can quickly and correctly implement basic IEEE 1149.1 test circuitry in an ASIC.

If synthesis is not available, most ASIC vendors offer a library of macros that the designer can use to implement IEEE 1149.1 into their designs. Using vendor macros requires considerably more effort and knowledge to implement IEEE 1149.1 into an ASIC compared to using synthesis. However, using vendor macros is significantly easier than implementing

boundary-scan architecture using a library that has no such macros. Texas Instruments offers a family of IEEE 1149.1-compliant ASIC macros for gate array and Field Programmable Gate Arrays (FPGA).

The ASIC designer is required to provide a Boundary-Scan Description Language (BSDL) file that describes the finished ASIC. This information becomes the basis for tools that create board interconnect test vectors automatically. These vectors can be used by the board designer for debug and validation and by manufacturing test departments for quality assurance.

Several synthesis tool vendors plan to implement automatic BSDL creation that is available once the ASIC synthesis has been completed. The creation of a BSDL file by the ASIC designer is straight forward if the design tool used does not create it automatically.

IC Simulation

During IC simulation, the designer can use the IEEE 1149.1 logic to simplify the simulation patterns required to validate the design. First, patterns must be developed to verify that the IEEE 1149.1 interface is operating properly. Simple IR and DR scan patterns can be developed to perform this check. If internal scan is implemented, then simulation pattern sets can be written to test internal logic clusters using the internal scan registers. This partitioned approach simplifies design validation.

Using SVF for IC Design Validation

Many IEEE 1149.1-compliant tools use SVF (serial vector format) as the primary test pattern data format and interfaces between simulators and ATE testers also support SVF.

A collection of manually-generated SVF patterns, simulation patterns translated into SVF and ATPG-generated SVF patterns can be used to validate the IEEE 1149.1 logic. Initial tests must first verify that serial data can be scanned into and out of the IC. Proper use of internal scan registers to partition the internal functional logic can simplify fault detection and isolation.

IC testing usually is broken into three parts; static functional tests at 1 MHz, at-speed functional tests, and parametric tests. Because IEEE 1149.1 architecture relies on serially scanning of control and test data, only static functional testing can be performed via the IEEE 1149.1 architecture.

INTEST patterns generated by a simulator can be used to validate the internal logic of the IC. By comparing these patterns with simulation results and by observing internal scan register contents, the designer can isolate faulty internal logic. Figure 6-3 summarizes the design validation flow for all products, from IC through system-level products.

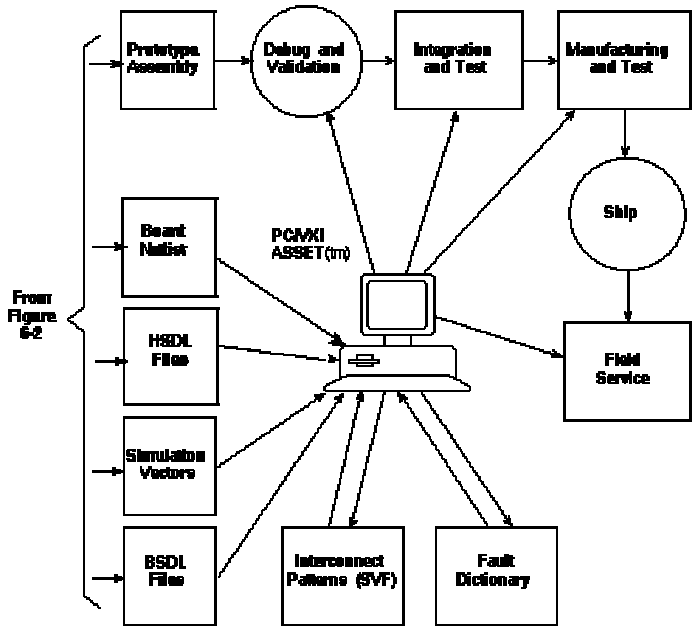

Figure 6-3. Debug and Verification of a Boundary-Scan Design

Data Passed to Board Designer

The board designer needs test data to use during the design debug and validation phase. This data includes:

- BSDL for the ASICs as developed by the ASIC designer
- BSDL for catalog devices from the device vendor

The BSDL that the designer creates for the ASIC must be validated to ensure that it is correct since it will be used by tools that develop more tests for board debug and validation. The new tests are then applicable for manufacturing test and field service test vector generation. BSDL created by a synthesis tool should accurately reflect a device's construction.

Along with the BSDL file for the device, the ASIC designer may also be asked to provide a subset of the ASIC simulation patterns if the ASIC supports the optional INTEST instruction. A subset of the simulation patterns may consist of the static (1 MHz) functional patterns. The reason only a subset of patterns may be needed is because it may be impractical to reapply the entire static functional test pattern set at the board level. Many companies have found that they can achieve an acceptable level of confidence that the ASIC is functioning correctly using only a portion of the entire static functional pattern set.

Board Design

Most of the concepts described for IC design also apply to board and system integration. BSDL is used to describe the IEEE 1149.1 scan architecture of an IC, and Hierarchical System Description Language (HSDL) is used to describe subsystems and systems. HSDL is used to describe the scan architecture, group discrete signals into logical groups (e.g., ADDR0 -15, DATA 0-7) and to define constraints.

Constraints

A constraint defines a logic condition that can apply potentially harmful signals to a circuit. IEEE 1149.1 gives direct control of all scannable signals, so it is possible to create

logic states that functionally would not be entered. For example, bus-arbitration logic can be overridden and dangerous hardware conditions or conflict can occur. An example of a constraint is to inhibit any test pattern that would cause two scannable output nodes to drive a common bus net at the same time.

Partitioned Scan Path

At the board level, the designer must consider how to partition the IEEE 1149.1 scan path. Commercial devices are available (such as TI's SN74ACT8997(1) SPL) to switch a single primary scan path to one or more multiple secondary scan paths. The designer should consider the benefits and penalties associated with partitioned scan paths.

Penalties associated with partitioned scan paths include:

- Scan overhead required to switch from one partition to another
- Timing delays in routing TMS (affecting scan frequency)
- Added complexity of the IEEE 1149.1 test program

Benefits associated with partitioned scan paths include:

- Shorter scan paths
- Additional fault tolerance
- Somewhat simplified test software

Board Validation/Manufacturing Test

The IEEE 1149.1 architecture has the ability to detect, and in many cases, isolate a scan path fault. The Instruction Register definition requires that the two (2) least significant bits (those closest to TDO) of a device be a logic 1 followed by a logic 0 during an IR scan. This rule allows the IEEE 1149.1-compliant bus controller to perform a simple IR scan to prove all devices on the bus can drive a logic 1 and logic 0 on the TDI-TDO path. There are ATPG tools available that automatically generate SVF bus validation patterns that can be applied by the bus controller.

The primary objective of the IEEE 1149.1 standard is to allow control and observation of device input and output pins for the purpose of interconnect testing. The reduced size of IC packages has made interconnect testing of PWBs difficult and expensive when using conventional interconnect testing. Commercial tools such as Teradyne’s VICTORY VIT (Virtual Interconnect Test) product develop SVF patterns for interconnect testing. VIT inputs the netlist and the BSDL of the scannable components and outputs an SVF pattern set. This pattern set along with a fault dictionary lookup table allows an IEEE 1149.1 bus controller to apply the interconnect patterns to the board and detect/isolate faulty interconnections.

After proving correct operation of the bus controller, the designer can control and observe all boundary-scan nodes. Tools such as ASSET allow interactive, macro, or C programming control for the designer. The board designer can also import and reapply the SVF patterns developed by the IC designer. This reuse of pattern sets provides these benefits:

- Knowledge is passed from one designer to another.
- Overall product development cycle is reduced.
- Using common SVF patterns reduces ambiguity between hierarchical test levels.

The SVF pattern set developed by the IC designer tests an individual IC. The board designer has to be able to relocate the relative scan position of the IC SVF test to match its scan position on the board dynamically. Boundary-Scan software tools such as ASSET(tm) (Advanced Support System for Emulation and Test) allow dynamic test relocation.

Summary

As designers become better acquainted with IEEE 1149.1, they will no doubt find benefits well beyond those discussed in this chapter. The standard became available just in time to offer a solution and benefits to customers experiencing complexity problems introduced by new technology.

The benefits that customers can realize from IEEE 1149.1 architecture include:

- Reuse test data throughout the product life cycle
- Apply off-the-shelf components, tools, and ATE solutions from multiple vendors
- Maintain common data base formats for these benefits:
    - ♦ Shared common databases between IC, board and system designers (BSDL, HSDL)
    - ♦ Reusable pattern sets (SVF)
    - ♦ Reusable Test Pattern Set (TPS) code

Applications

This article presents a number of testing problems and shows how boundary-scan testing and TI products can be used to solve them.

Board-Etch and Solder-Joint Testing

The current approach to detecting board-etch and solder-joint faults in today's electronics industry uses two forms of Automatic Test Equipment (ATE): bare-board testers and digital in-circuit testers. Bare-board test is done after board fabrication. Tests for open-etch circuits between nodes force a voltage onto one circuit node and sense the voltage at another circuit node. This technique is used only on a raw, unassembled board.

An in-circuit test is done after board assembly and checks for:

- Correct orientation of parts
- Correct value and type of parts
- Correct solder joints between all parts and the board

Software models of each part on a board are used to generate the test patterns used by the in-circuit tester. The main assumption is that both board and in-circuit testers require physical access to nodes or points on the board in order to perform the necessary testing. Both testers require the ability to inject a voltage or current onto a nodal network and sense the voltage or current at another point on the network.

Higher-density packaging technologies, finer-pitch board etch, and more complex digital integrated circuits have made bare-board and in-circuit testing much more expensive.

Solution

Several testability products from Texas Instruments lend themselves to support IC-to-IC continuity testing. The techniques outlined allow detection and in most cases, isolation of board-etch and solder-joint faults, without forcing any additional requirements on the electrical design engineer. ASIC devices with boundary-scan cells (BSC) and bus interface

components allow controlled insertion of digital logic values onto a nodal network by a driver-integrated circuit. This provides the means to sample the digital logic values on all receiver integrated circuits connected to a particular network. Broken-etch and solder-joint faults can be detected and isolated in this manner.

Detailed Description

In the simple circuit shown in Figure 7-1, one driver (IC #1) on a network drives digital logic values while the other receivers (ICs #2 and 3) sense changing values on the net. If multiple driver ICs exist on a network, each driver is allowed to drive in turn while all receivers sense to detect and isolate a fault to a specific driver or portion of the net.

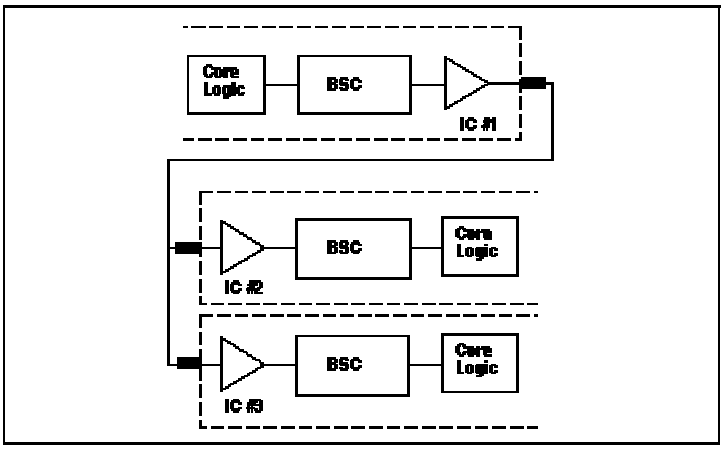

Figure 7-1. Etch/Interconnect Testing

If an etch is broken on a network or an IC is not soldered to the board properly, then nodes of some ICs are not connected to nodes on others as shown in Figure 7-2 and Figure 7-3. When driver IC #1 drives, only IC #2 senses the correct logic value, IC #3 does not.

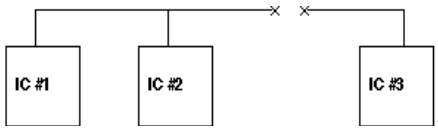

Figure 7-2. Open-Etch Condition

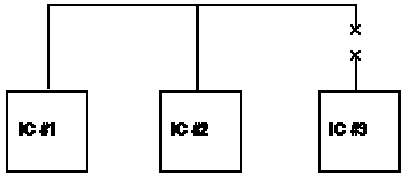

Figure 7-3. Open-Solder-Joint Condition

If an etch is shorted to ground or power due to a solder fault, then none of the ICs are able to drive or sense the correct logic values. See Figure 7-4.

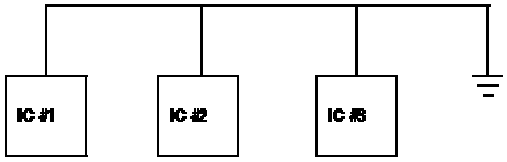

Figure 7-4. Short-to-Ground Condition

The overall coverage of etch- and solder-fault detection and isolation depends on the percentage of a board's circuitry that has boundary-scan architecture. Only networks driven by BSCs or IEEE 1149.1-compliant ICs can be tested in this way; non-scan controlled networks require conventional approaches.

Because the driver and receiver BSCs are in silicon within the ASIC, this technique can be used to test silicon bonding to package pins in the integrated circuit foundry. A simple bond-out tester can drive a logic value to an input pin of a IC and the sensing function can occur on the BSC buffering the input to the core logic of the ASIC. The reverse procedure can be used on an output pin of an IC, the BSC driving the output of the core logic forces a logic value that the bond-out tester senses and records. See Figure 7-5.

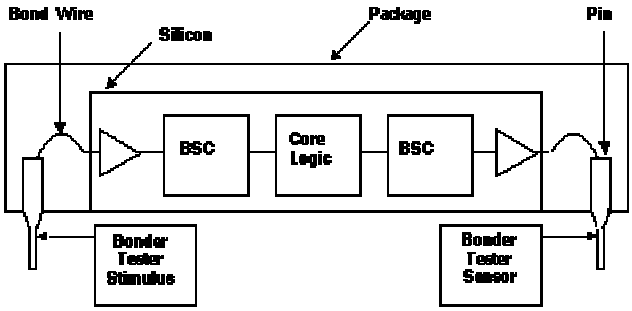

Figure 7-5. Bond Wire Testing

Summary

The current technique used to detect and isolate board-etch and solder-joint faults remains unaltered. With TI's suite of testability products, board testing can be performed without external digital in-circuit testers. Boundary-scan architecture eliminates the need for physical probe points on printed-circuit boards that contain advanced-packaged, surface-mount integrated circuits. The need to develop part models for in-circuit test generation is also eliminated.

Cluster Testing

Complex digital designs use large-scale integrated circuits and ASICs to accomplish many functions on compact circuit boards. Not all logic functions are integrated, so discrete devices are still used for some functions. Discrete logic seldom includes boundary-scan test architecture due to the expense and complexity; these parts must be tested another way.

Solution

Testing discrete logic functions on a board has to be done from the boundary of the logic cluster. Combinatorial logic is normally driven by and into sequential elements such as a register or a clocked ASIC. By replacing the register with a IEEE 1149.1-compliant device, or by using Texas Instruments IEEE 1149.1-compliant macro cells at the boundary of the ASIC, the cluster logic may be easily tested. By using members of TI's family of testability products,

a higher percentage of circuitry may be tested while reducing the vector count necessary to perform a functional test.

Detailed Description

Figure 7-6 shows a simple circuit incorporating combination logic sandwiched between sequential logic.

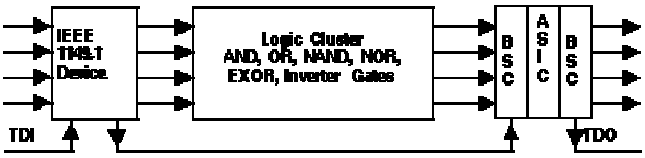

Figure 7-6. Cluster Testing

During normal circuit operation, the IEEE 1149.1-compliant device drives the glue logic through to the BSCs in the ASIC. Two methods may be used to perform a test of the logic cluster using the TI family of testability products. There are benefits and drawbacks to both methods.

1. Method one is to scan a fixed-order set of test patterns called deterministic test vectors into the IEEE 1149.1-compliant device and sample the resultant data at the ASIC boundary.
2. Method two is to command the IEEE 1149.1-compliant device to output a pseudo-random pattern of data while the ASIC performs a parallel signature analysis of its inputs.

Deterministic Test Vectors

Deterministic test vectors scanned into the UUT can pinpoint a problem more easily than signature analysis. Since the exact input stimulus is known, the failing response can be detected. Deterministic test vectors perform a complete and meaningful test, but consume a lot of time and enough memory to store all vector data for both stimulus and response.

Signature Analysis

Signature analysis can be performed quickly and virtually automatically. After valid pattern generation and signature analysis seed values are obtained, the resultant signature can be checked against the correct value. This test can be performed at the scan clock speed and requires very little memory for data storage. However, the fault cannot be located without using an extensive fault dictionary.

Signature analysis requires that the IEEE 1149.1-compliant device be able to generate stimulus patterns and the ASIC be able to capture a signature of its inputs. This means that appropriate macro cells must be used as outputs of the driving device and as inputs to the boundary of the ASIC. These cells require additional space and additional circuitry to be incorporated into each BSC.

Summary

Using the IEEE 1149.1-compliant family of testability products from Texas Instruments enables designers to add features previously unachievable in their designs. It is beneficial to test glue logic clusters using existing boundary-scan components that surround the logic cluster. This provides added value to the boundary-scan overhead built into a design's larger devices.

Board-Edge Connector Testing

Board-to-board fault detection and isolation within a digital system have historically depended primarily on Built-In Test (BIT) software. This approach is adequate for detecting functional faults on boards assuming that the processor executing BIT software can access the boards installed in the system. Backplane faults typically manifest themselves in these ways:

- Broken or bent pins on connectors
- Broken etch on a backplane motherboard
- Termination-network fault
- Bus contention on a bused network

Many bus faults in today's digital processor environments are so catastrophic in nature that the processor executing BIT can not access the boards installed within the system to isolate the fault.

Solution

In much the same way a boundary scan within ASICs allows detection and isolation of etch and solder joint faults on a PWB; surrounding the bus interface of a board with IEEE 1149.1-compliant devices allows detection and isolation of bus and connector faults. By placing IEEE 1149.1-compliant devices in a controlled test mode, each board in turn can drive the backplane bus while the other boards receive the driven logic values.

Detailed Description

Broken etch on a motherboard backplane presents similar problems to that of broken etch on a PWB connecting integrated circuits; some boards cannot communicate to others past the etch break (see Figure 7-7). By placing the testability devices on each board into a controlled test mode, one board at a time can drive logic data onto the bus while the others are receiving logic data. This method provides enough conclusive data to detect and isolate the continuity break.

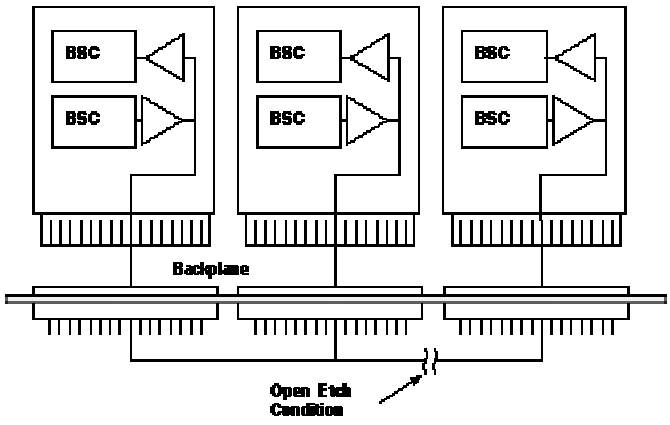

Figure 7-7. Backplane Open-Circuit Fault

A broken connector pin fault can be detected and isolated in the same manner as etch breaks; the board with a bad pin is not able to drive and receive data onto a particular channel on the backplane. See Figure 7-8.

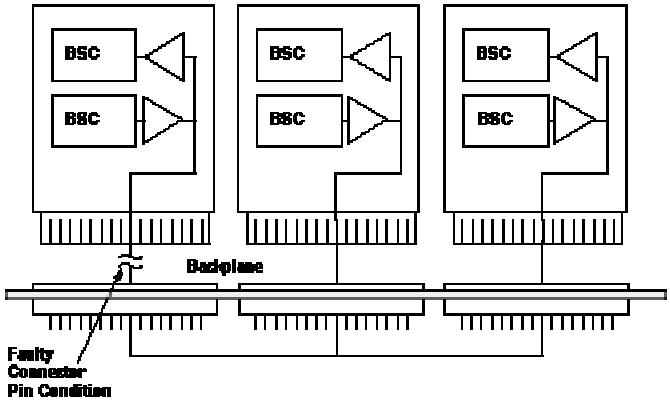

Figure 7-8. PWB Connector Fault

Termination network open or short to VCC /Ground shows up as a channel on the backplane that can not be driven to a certain logic state. The characteristic of this fault could be confused with a bus contention fault. See Figure 7-9.

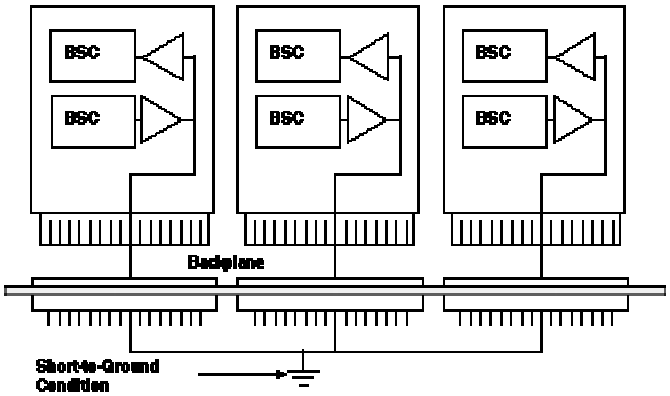

Figure 7-9. Backplane Short-To-Ground Fault

Bus contention faults occur when a three-state bus driver is stuck in a functional-enable mode. The BSCs in a controlled mode can be disabled by the test circuitry, thus bypassing the functional-enable control. All of the tests described can be executed off-line in a test mode or by Built-in Test (BIT) software interfacing to a IEEE 1149.1-compliant serial scan controller in the system.

Summary

Board-to-board fault detection and isolation in a controlled off-line test mode and concurrent background BIT execution can be dramatically improved by using boundary-scan testability devices or ASICs with boundary-scan cells within the bus interface circuitry of each board installed in a system. Boundary-scan architecture at the backplane-interface can increase fault isolation by masking some interface logic from the circuitry under test.

ASIC Verification

Very high density ASIC devices pose severe barriers to all testing, beginning with design verification and continuing with test and verification of the manufactured part.

Solution

IEEE 1149.1-based testing can be used for testing and verifying ASIC devices, provided that the design process includes scan-based testing beginning with design specifications. This example cites the development of a vector processor ASIC with 165,000 gates.

Detailed Description

The test logic in the vector processor includes internal scan and built-in emulation logic in addition to boundary-scan and the optional INTEST capability. The emulation capability includes control of the processor, implementing RUN, STOP, SINGLE-STEP, and EXAMINE/MODIFY internal registers, and real-time breakpoints.

Test Development

Design verification, test hardware and test software are developed concurrently with the ASIC. Verification code consists of two types of tests:

- Scan-based tests and diagnostics
  - ♦ Developed in C++
  - ♦ Developed using ASSET
- Embedded test code

ASSET(tm) (Advanced Support System for Emulation and Test) software is used to execute and monitor boundary-scan tests. The tests are used to verify test bus operation, scan path integrity, internal-register access, core-logic operation, and IC-to-IC interconnects. ASSET is used on an IBM PC(tm)-compatible computer and consists of a PC-AT(tm) controller card and controller software. An expanding set of scan-based utilities and special-purpose test programs are written to perform specific tests or trouble-shoot hardware problems. These utilities are developed from the time hardware integration is begun.

In addition to the scan-based tests, embedded test code is developed in microcode to verify instruction set execution, register and memory decode, memory access, and input/output operations. The embedded tests target functional logic and verify at-speed operations.

Design Verification

The vector processor ASIC in the design incorporates extensive internal scan paths and breakpoint logic controlled via the IEEE 1149.1 test bus. Key registers that are accessible via scan include the address and data register files, ALU and multiplier pipeline registers, and the program instruction register.

Initial checkout of the vector processor includes scan path integrity and internal register access using internal scan and boundary scan. The first devices contain nonfunctional microcode RAM therefore an alternate plan is devised to use the IEEE 1149.1 to validate all other circuitry.

The vector processor checkout uses the boundary-scan architecture to perform register reads and writes and simple operations to verify microcode execution. Simple microcode operations are tested by scanning an opcode into the instruction register and clocking the processor using boundary scan.

A test routine is developed in C++ that automates the microcode execution steps. The program reads microcode-object files, scans the instruction into the instruction register, clocks

the processor, and reads the program counter to locate the next instruction. As each instruction is executed, the program dumps the accumulator and status registers to allow monitoring of each instruction. This process is repeated for each microcode routine until all microcode is executed and nearly all processor functions are verified using the "bad" vector processors. Many weeks are saved while waiting for new vector processors.

Summary

Design verification, hardware test, and software integration represent significant challenges. Anticipating this, many design for test and integration capabilities need to be incorporated into the device to ensure controllability and observability of embedded functions. The IEEE 1149.1 standard provides the primary method for design verification, hardware/software integration, and hardware test.

Test and interaction issues were exaggerated in this example due to the lack of physical access caused by small geometry of the module. Boundary-scan testing saved a significant portion of the design verification effort that would have been required for the less effective conventional physical access methods.

Memory-Testing Techniques

The steady increase in memory density has led to the increased use of high-capacity memory boards in today's electronic systems. It has also encouraged the use of embedded memory within microprocessors, digital signal processors (DSP), and ASICs. As a result, testing of these boards and devices has become more complex. Large-capacity memory boards require very long test execution times for adequate fault detection. Large bus sizes (32 bits) and high bus fanout further complicate the problem of isolating component faults. Embedded memories, which often make up a significant portion of the device area, create additional difficulties due to a lack of test access.

Solution

The use of scan-path technologies can simplify the problem of testing large memory arrays within a system or embedded memories within a device. Scan-path devices such as TI's bus interface components can be used to partition large memory device arrays into smaller, more easily testable arrays to increase component fault isolation. In addition, the use of TI's IEEE 1149.1-compliant macro cells within ASIC devices provide controllability and observability of embedded memory that is normally inaccessible. Memory test times can be reduced through the efficient partitioning and testing of several memory arrays (on a board or within a device) in parallel.

Detailed Description

Figure 7-10 shows a 256 x 8 memory array, associated bus interface, and control logic configured to test various memory-pattern-generation techniques. This configuration allows for explicit read/write operations using IEEE 1149.1 EXTEST and SAMPLE instructions. In addition, the components have a BIST capability (controlled by IEEE 1149.1 instructions) used to perform memory read/write operations. The test results shown in Table 7-1 compare explicit scanning in the RAM array address, data, and strobe signal to using an IEEE 1149.1-instruction-controlled BIST to generate address, data, and strobe signals automatically at the Test Clock (TCK) rate of 6.25 MHz.

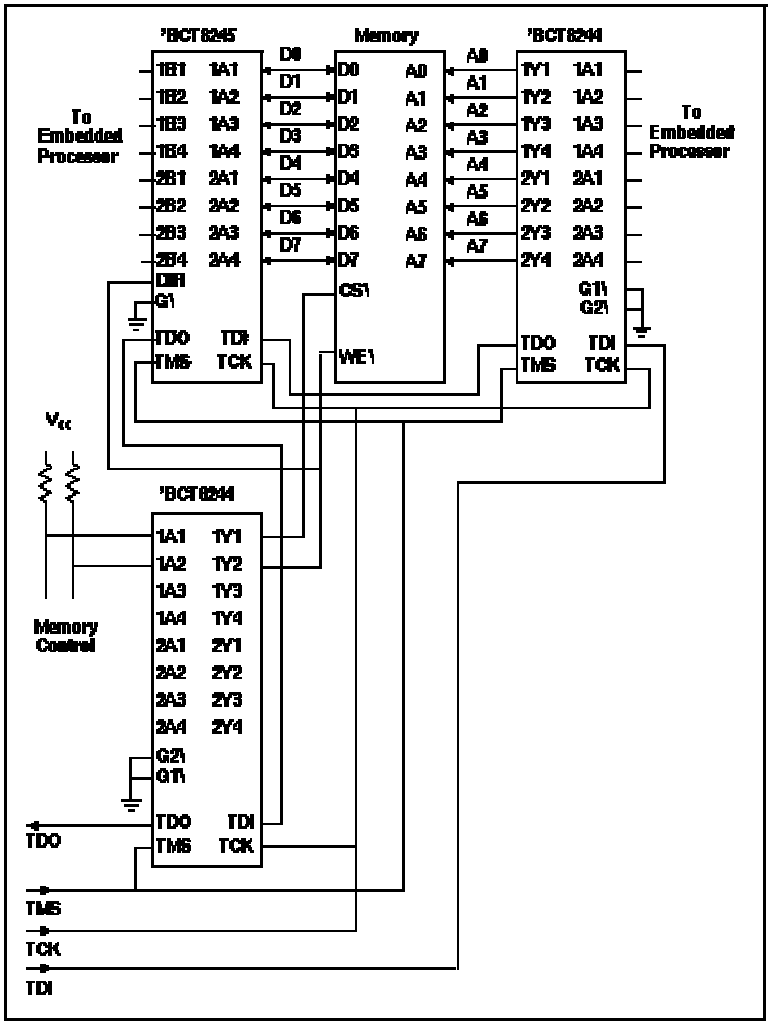

Figure 7-10. Block Diagram of Simplified Boundary-Scannable RAM Interface

Table 7-1. Access Rates of IEEE 1149.1 Devices With and Without BIST

Mode		256 Accesses	1,000,000 Accesses
IEEE 1149.1 (With BIST Capabilities)			
	Time to Apply	0.00041 seconds	<0.01 minutes
	Number of Scans	7	28,000
	Number of Patterns	512	2,000,000
IEEE 1149.1 (Using *EXTEST* and *SAMPLE*)			

Table 7-1 shows that the BIST controlled by IEEE 1149.1 instructions overcomes the limitations of conventional memory testing. Using boundary-scan EXTEST and SAMPLE/PRELOAD solves the direct physical access problem, but it is time consuming. By using BIST circuitry embedded within the functional logic surrounding large memory arrays, IEEE 1149.1 tests can be designed to emulate closely the characteristics and timing speeds of the memory being accessed. TI's bus interface components offer the IEEE 1149.1-controlled BIST functionality required to test memory while providing the electrical signal conditioning and buffering a design engineer typically designs around a microprocessor.

In a similar fashion, TI's family of IEEE 1149.1-compatible cells can be used within an ASIC device to provide access to embedded memory normally inaccessible through external pins. The IEEE 1149.1-compatible cells support the same self-test features as the bus interface devices (PRPG, PSA, Toggle 1/0 modes), which further simplify memory testing. Additional features such as programmable PRPG and PSA taps, built-in comparison logic, maskable PSA/comparison inputs, and bidirectional cells provide even more flexibility in self-test design. These features can be used with a built-in test controller to create an autonomous BIST for the memory array and/or other functions within the device.

The use of IEEE 1149.1 cells in performing read/write tests on RAM is shown in Figure 7-11. Address sequencing is performed using the PRPG mode. The PRPG and toggle modes automatically generate data patterns to be written to the memory cells under test while the memory write enable is controlled by direct scan. When reading from memory, data patterns are verified using the comparison logic in the IEEE 1149.1-compatible cells. This is possible since each IEEE 1149.1-compatible cell can latch and remember the last data pattern that was written when configured correctly. During memory reads, the latched pattern is fed back for comparison with the data being read. A pass/fail signal for the test can be generated by combining the comparison outputs of each IEEE 1149.1-compatible cell.

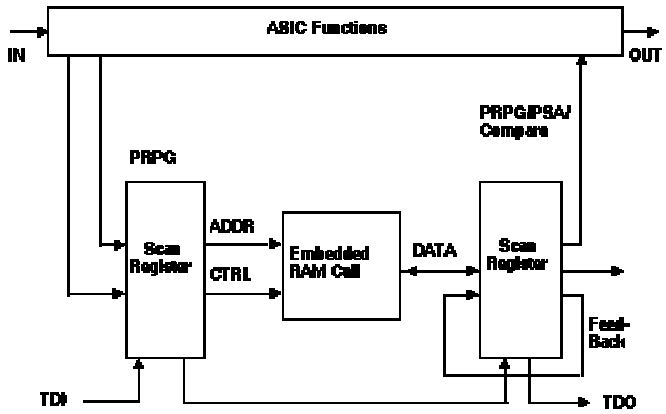

Figure 7-11. Testing Embedded RAM

A similar testing method can be used when testing Read-Only Memory (ROM) embedded within a device, as shown in Figure 7-12. During ROM tests, address sequencing and read enables are controlled as in Figure 7-11, and the PSA mode is used to compress the data output into a representative signature. If desired, comparison logic can be used to compare the calculated signature with a known good signature to generate a memory pass/fail signal.

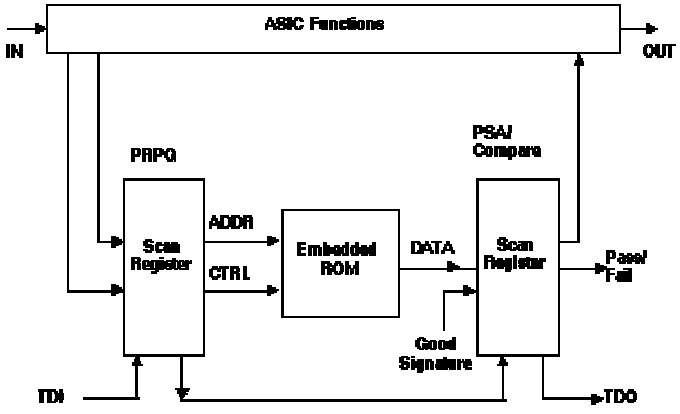

Figure 7-12. Testing Embedded ROM

Summary

The use of TI's bus interface components and boundary-scan cells provides increased fault detection and isolation capabilities when used to partition large-capacity memory boards and devices with embedded memory. The increased partitioning allows the use of parallel testing and autonomous built-in self test, greatly reducing the overall test execution times normally attributed to memory testing. IEEE 1149.1 bus interface components provide simple replacements for buffers and transceivers that are normally used in memory system design while creating little impact on system performance. TI's boundary cell library is a completely configurable set of test cells that allow the designer to create sophisticated test structures (such as built-in self test for memories) while requiring only a minimal knowledge of test engineering principles.

Backplane Multidrop Environment

The backplane environment, which carries many signals in parallel to all the boards, contains certain conditions not found in other test situations. A typical backplane with a complement of boards (such as a computer that uses different accessory boards) may be configured differently for different purposes. In other cases, certain boards may be capable of interacting independent of normal backplane activity, and some boards may include BIST. For maximum usefulness and short test times, a test program must accommodate a partially populated backplane while allowing more than one board to be active at a time.

The IEEE 1149.1 standard was developed to access ICs on a board serially, but it can also be used to access boards installed in a backplane. By making simple hardware modifications to the backplane circuitry, ring or star backplane configurations can be accessed using normal IEEE 1149.1 commands.

In a backplane configured as a ring (see Figure 7-13), all boards directly receive the TCK and TMS control outputs from a primary test bus controller (TBC). The scan path is daisy chained between the TBC's TDO output and TDI input. During a test, the TBC drives TMS and TCK to scan data through all boards in the backplane, through the TDO/TDI bus connections. A serial path works only if all the boards are present and can scan data from their TDI input to TDO output. A missing or faulty board breaks the path so the TBC cannot scan data through the backplane.

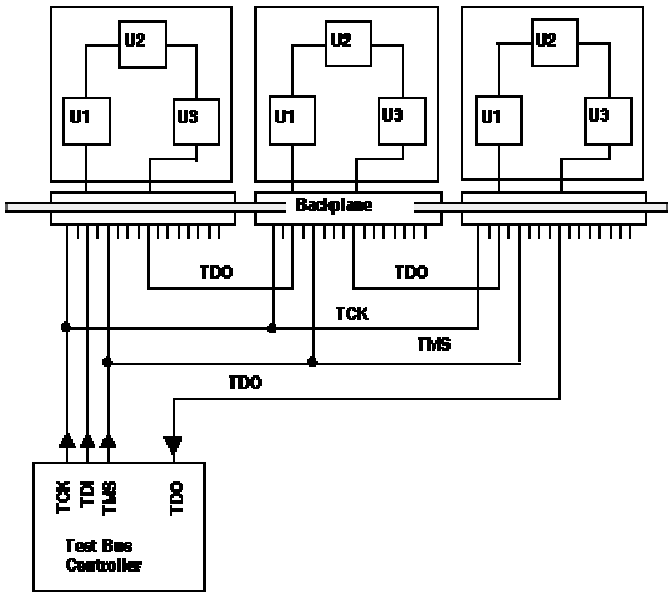

Figure 7-13. Backplane Ring Configuration

In a backplane star configuration (see Figure 7-14), the TBC drives TCK and TDI directly to all boards and each board outputs a TDO signal to the TBC. Each board requires a unique TMS signal from the TBC and a separate circuit trace for each signal. During testing, only one board is enabled at a time and only the TMS signal for that board is active. In a backplane with 50 boards, the TBC needs 50 individual TMS signals, with circuit traces for each. Another problem is that only one board may be scanned at a time.

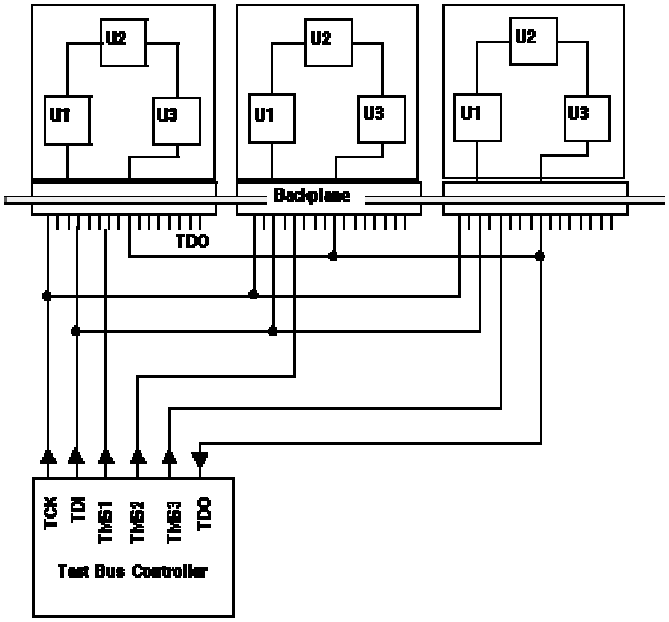

Figure 7-14. Backplane Star Configuration

Solution

A new serial bus protocol has been developed by Texas Instruments that is interpreted by a TI device called the Addressable Scan Port (ASP) (SN74ABT8996). This device provides linkage between IEEE 1149.1-compatible boards and a backplane. The ASP becomes the IEEE 1149.1-input bus for each board in the backplane. It intercepts the protocol on the normal incoming IEEE 1149.1 signal wires, allowing it to connect or disconnect its board with the IEEE 1149.1 bus on the backplane. Once connected, the board can communicate using the standard 1149.1-bus protocol.

The ASP protocol is invoked when the IEEE 1149.1 bus is in one of the following four TAP states: Test-Logic-Reset, Run-Test/Idle, Pause-IR, or Pause-DR.

The protocol can address and select any ASP-equipped board in the backplane. After the board is connected to the IEEE 1149.1 bus, the ASP simply passes data through to the bus so normal operation can resume.

Detailed Description

Figure 7-15 shows a backplane with ASP devices on each board, allowing individual boards to be configured and active simultaneously.

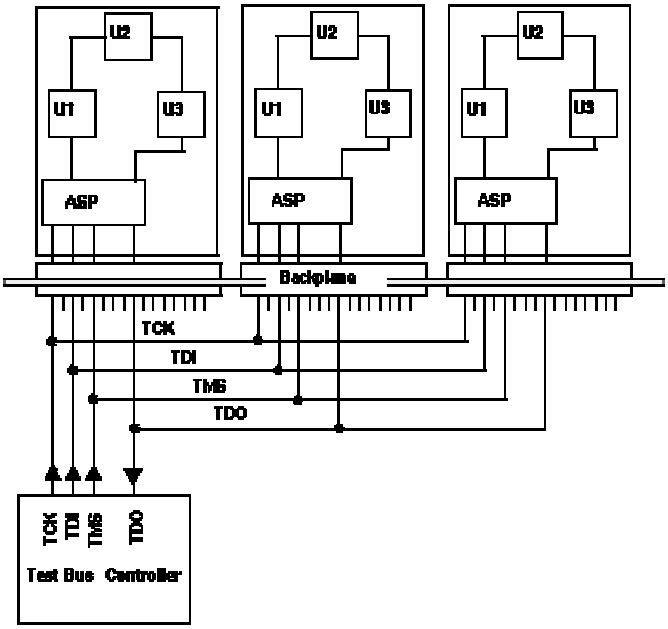

Figure 7-15. Backplane With ASP-Equipped Boards

For example, if the interconnects between two boards are to be tested, it is necessary to select and scan the first board to output a test pattern, then select the other board to receive the test pattern. The ASP makes it possible to select and scan the second board without resetting the first board. During operation, each pattern of the interconnect test is scanned out of the second board before a new pattern is scanned into the driving board.

Another test may be to select and initiate self-tests in a selected group of backplane boards. The ASP allows BIST to be started on each board and left running while other board tests are initialized. Each board is then polled to read the test results, one board at a time.

Summary

The special problems associated with backplane testing can be accommodated when the SN74ABT8996 Addressable Scan Port is added to individual boards.

Embedded Applications

Field testing can be simplified if equipment is configured to take advantage of embedded IEEE 1149.1 functions.

Solution

TI's SN74ACT8990 Test Bus Controller (TBC) can be embedded in system circuitry to provide autonomous IEEE 1149.1-based testing. A licensable, embeddable "C" source code product called Scan Engine, which includes test vector translators, is available to simplify embedded boundary-scan tests.

Detailed Description

The host interface on the TBC consists of a 16-bit data bus (DATA15-0), 5-bit address bus (ADRS4-0), a read strobe (/RD) and a write strobe (/WR). Twenty-four 16-bit internal registers are accessible by the host processor.

Both port 0 and port 1 share a common TDO and TCK. The 'ACT8990 has separate TDI pins for ports 0 and 1, so that the return serial data from each scan path remains isolated.

IEEE-1149.1 Interface

The 'ACT8990 has two separate IEEE 1149.1-compliant interface ports, only one of which Scan Engine software can access at a given time. Port 0 consists of three signals: TMS0, TDI0, TDO. Port 1 signals are: TMS1, TDI1, TDO.

This allows an active buffer/driver to be used between TDO of the last scannable device and the TDI input pins. TDO is common for both ports 0 and 1. Port selection is controlled by which TMS signal is activated.

Isolating the 'ACT8990 From the IEEE-1149.1 Bus

The 'ACT8990 has a pin designated TOFF*. This pin controls all output pins of the 'ACT8990. When asserted (logic low), all 'ACT8990 I/O pins and output pins are forced to the high-impedance state.

By placing the outputs of the 'ACT8990 in high-impedance state, the IEEE 1149.1-compliant interface is disabled. This allows an external IEEE 1149.1-compliant controller (such as the ASSET Interface Pod) to take over the scan bus.

Host Memory Requirements

A host processor executing embedded Scan Engine software needs three primary hardware resources:

- Memory-mapped parallel access to the embedded 'ACT8990
- ROM/EPROM space for the Scan Engine object code
- RAM or ROM space for the test vectors. These test vectors are broken into two primary blocks, SBSF for stimulus data and SBRF for response data.

It should be noted that the SBSF stimulus data could either be downloaded into RAM or be permanently programmed into ROM/EPROM.

Summary

The SN74ACT8990 Test Bus Controller offers the designer of high-reliability equipment a way to embed automatic IEEE 1149.1-compliant test circuitry into products. Scan Engine provides an easy way to control the SN74ACT8990 in an embedded system. The solution discussed here maintains IEEE 1149.1-compatibility and access to test vectors developed during design verification. The result is continuous quality assurance with direct traceability to development data.

Boundary-Scan Test Flow

When new ASICs or boards come from the manufacturing line, the first concern is that the devices work as designed. High-density ASICs and multi-layer PWBs with surface mount devices can't be adequately tested without boundary-scan testing.

Solution

Use boundary-scan techniques and tools on products designed with 1149.1-1990-compatible components.

Detailed Description

Figure 7-16 summarizes an efficient verification process using BSDL supplied by manufacturers of IEEE 1149.1-compatible ICs and HSDL developed for the ASICs and PWB.

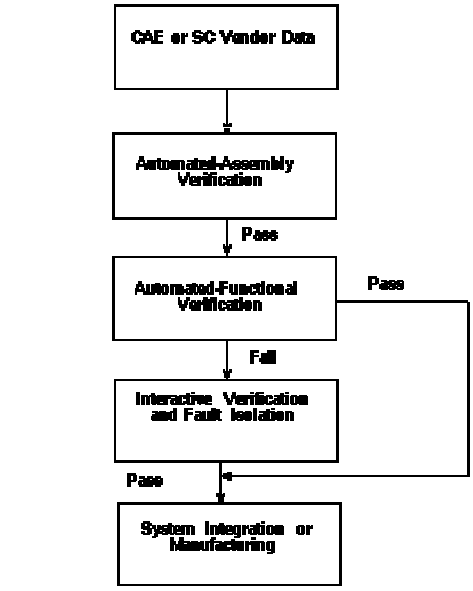

Figure 7-16. General Boundary-Scan Test

Automated-Assembly Verification

Automated-assembly verification depends on getting BSDL or HSDL descriptions for IEEE 1149.1-compatible devices from the manufacturer and developing corresponding HSDL for modules, boards, and systems. Figure 7-17 shows the flow that verifies a correct scan path and interconnections between boundary-scan cells.

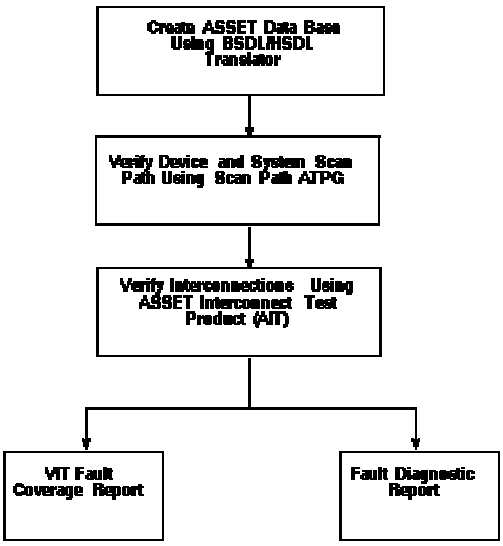

Figure 7-17. Assembly-Verification Flow

Devices that pass assembly verification must then be tested for functionality.

Automated Functional Verification and Fault Detection

Functional verification begins with IC testing followed by virtual component or cluster testing and system verification. These flows and the tools involved are shown in Figure 7-18.

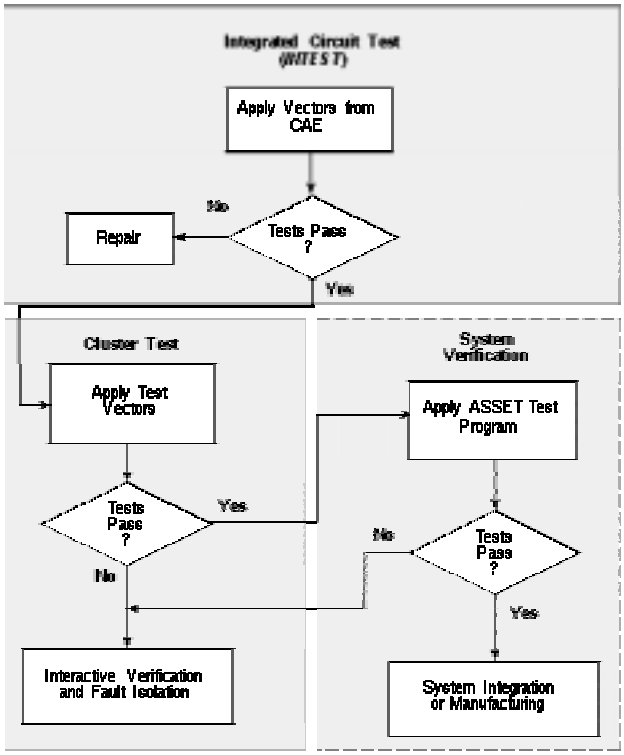

Figure 7-18. Automated Functional Verification and Fault Detection

No matter which flow is used, the goal is to reuse data whenever possible. For functional verification we want to reuse any serial or parallel vectors developed in the CAE system; those vectors become a pass/fail test. When CAE vectors are not available, ASSET supports test stimulus development approaches such as vector recording, macros, and C++ programs.

All functional tests highlight failures down to the vector level. Knowing which vector failed can reduce the time spent in interactive testing to isolate the problem to a device, register, or pin.

Interactive Verification and Fault Isolation

Figure 7-19 shows how faults in units failing previous tests can be isolated using specialized software tools, the ASSET Debugger and Scan Analyzer.

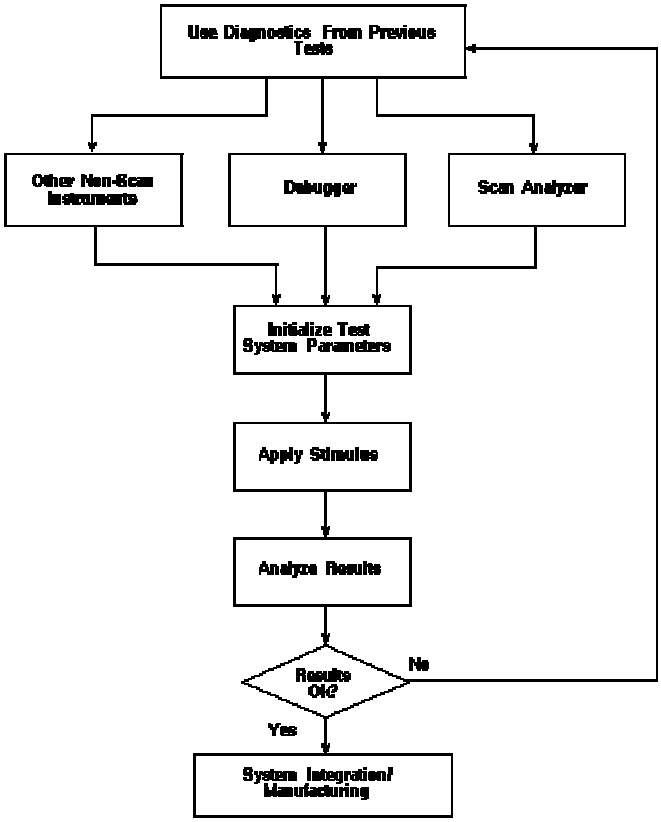

Figure 7-19. Interactive Verification and Fault Isolation

Summary

Design verification and fault isolation processes are greatly enhanced when a full suite of boundary-scan tools and techniques are used.