VLSI Design
1994, Vol. 1, No. 4, pp. 345–357
Reprints available directly from the publisher
Photocopying permitted by license only

Integrated Test Solutions for a System Design Environment

KEVIN T. KORNEGAY† and ROBERT W. BRODERSEN
EECS Department, University of California at Berkeley

While the performance, density, and complexity of application-specific systems increase at a rapid pace, equivalent advances are not being made in making them more easily testable, diagnosable, and maintainable. Even though testability bus standards, like JTAG Boundary Scan, have been developed to help eliminate these costs, there exists a need for efficient hardware and software tools to support them. Hence, a testability design and hardware support environment for application-specific systems is described which provides a designer with a set of hardware modules and circuitry, that support these standards and software tools for automatic incorporation of testability hardware, as well as automatic test vector and test program generation.

Key Words: *Design-for-Test; Built-In-Self-Test; Boundary Scan; Device Under Test*

Recent advances in manufacturing and packaging technology have made it possible to design very large, high performance VLSI systems. The process of taking a requirement for a digital system and implementing that system in hardware begins with design and ends with test. In the past, design and test was regarded as two separate steps but today, they are thought of as two closely integrated tasks. So today, designers are faced with this unformidable challenge and they can no longer adopt the old "toss it over the wall" attitude. Some of the barriers they are faced with that compound the testing of these systems are:

- the constant demand for greater integration;
- the widespread adoption of advanced packaging technology like high density surface mount and multi-chip modules (MCMs) employed on both one-sided and two-sided printed circuit boards (PCBs);
- the increasing cost of Automatic Test Equipment (ATE) and associated test fixture;
- the growing gap in speed between the device under test (DUT) and the ATE;
- the increasing consumer demand for high quality, reliability, and maintainability.

Hence, developing a testability design and hardware support environment that helps designers overcome some of these barriers would be of great value. This paper addresses issues related to the automation of test in the SIERA design environment. In the first section, we present an overview of the SIERA system and discuss the test strategy used. The chip, board, and system level test hardware is discussed in the following three sections. We show a chip and board implementation and test software is discussed, while the system design and test procedure is presented. Finally, concluding remarks are presented.

THE SIERA DESIGN ENVIRONMENT

Advances in VLSI technology has led to the creation of chips which resulted in a complexity that created a bottleneck in the chip's overall development time. This bottleneck was alleviated by the use of silicon compilers, which produce the physical information required to fabricate chips from higher level descriptions of the design; these could be a symbolic layout, a circuit schematic, a behavioral description of a microarchitecture, an instruction set, or an algorithm for signal processing.

†Please direct all correspondence to kornegay@watson.ibm.com.

The compiler then transforms this high level description into a physical representation required by the fabrication foundry. This transformation occurs in several steps. For example, a compiler might transform a behavioral description of a design into a logic gate level representation. A major advantage of this approach is that designers can work at higher levels of abstraction, without having to know specifics about the IC design and process technology. Another and, probably the most important advantage of this approach, is the ability to rapidly produce chips. Further advances have also led to the creation of very complex systems. Even though these systems contain hundreds of components, tools that support the integration of these components to make up the system are still in a primitive state. Additionally, these same tools do not exhibit usability and rapid prototyping features. One such CAD environment that does exhibit these features is SIERA [22]. An overview of SIERA is presented in this section, along with a discussion of the test strategy employed by SIERA along with the associated testing environment.

Hardware Module Generation [22]

Embodied within the SIERA framework is a vertically integrated design methodology. This supports the development of application-specific systems at all levels, from the high level description to the board implementation and software generation. The primary objective of the hardware module generation environment was to provide an environment that could handle arbitrary hardware architectures implemented as custom boards using custom and commercial ICs, as well as the capability to explore alternative implementations.

Module generation was founded on the basis of pre-existing ASIC generators, like those in LAGER [4, 21], that automatically produce circuitry required to implement dedicated chip level macro-functions and then, tie them together to achieve the desired functionality. These same concepts also proved to be extremely useful at the board level, where one or more custom and/or commercial ICs are grouped together to implement a complex function. This environment also uses a hardware description language called SDL to describe the designs, OCT [7] to store the design information, and a design manager called DMoct to orchestrate the design flow. Additionally, chip level module generators and behavioral tools are also available to the board designers. As a result, a complex board design can be represented as a netlist of high-level modules that are maintained in a library consisting of parameterized reusable modules (adders, multipliers, etc.) or as a behavioral description (FSM controllers, decoders). The final step toward completing a board design is the layout, for this, several layout generation tools were developed.

Test Strategy Used in SIERA

Traditional approaches to system testing often employ a three level strategy. First, an engineer with very little or no test experience runs a system self test to quickly determine whether the dependency, which means that a failure is caused by environmental conditions like temperature and vibration, false alarms caused by design errors or transient faults, incompatibility of tests, which is caused by the use of inconsistent testability techniques at different levels of the system hierarchy, and faults in the test hardware. The approach described above is adequate for high volume production environments but insufficient for low to medium rapid prototyping environments like SIERA. Hence, the test strategy used in SIERA should eliminate or at least reduce the problems mentioned above. The strategy used in SIERA should be able to support testing at all levels of the system's hierarchy, support existing testability bus standards, produce test vectors, have a facility to initiate and execute tests and provide a means to integrate design for testability (DFT) techniques into the design process. This will result in a complete solution that deals with the design, as well as the testing of a system. This approach is contrary to the approaches described in [1, 2, 8, 10, 13, 28], where the authors only present partial solutions.

Embodied within SIERA is a test environment that fulfills the above requirements, where the designer, has available, hardware modules at both the chip and board level that implement Scan Path [5, 6, 24], Built-In-Self-Test (BIST) [19, 25], and the Boundary Scan architecture [9, 18], as well as, software tools for connecting these modules together, test languages that describe what DFT modules are used and how to use them during test, and a software programmable custom controller board that is used to control and access the devices (chips and boards) which contain this hardware. Specific details and issues regarding the design, implementation, and application of each component of the test environment will be discussed in the sections that follow, but a discussion on how we integrate test into SIERA is warranted.

Integrating Test into SIERA

The DFT techniques used in SIERA must tackle test problems at both the chip and board levels. Furthermore, these techniques must not significantly impact performance or area. While preserving these objectives, test is integrated into SIERA in two steps. The first step involves identifying the building blocks that comprise a chip or board. For example, most chips consist of combinational or sequential logic, register, and some memory. At the board level, devices are categorized as either Boundary Scan compliant or not. After the components have been identified, hardware that supports the DFT methodologies mentioned earlier is automatically generated and incorporated into each component during step two.

At the chip level, registers are configured as a Scan Path and used to test combinational logic, circuitry required to implement the Boundary Scan architecture is added, and BIST circuitry is added to the memory. At the board level, components that have a Boundary Scan counterpart are replaced and all Boundary Scan devices are chained together and connected to the board's Boundary Scan test port. Figure 1 illustrates how these features are integrated into SIERA. Embedded with SIERA is a Test Environment which consists of four major parts: Test Hardware Generator, Test Vector Generator, Test Program Generator, and the Test and Diagnosis System [14, 15]. A major advantage of this approach is that it shields the designer from having to worry about the implementation of specific details associated with each DFT methodology and the mundane tasks of test vector and test program development. More importantly, all this is achieved in an automated fashion.

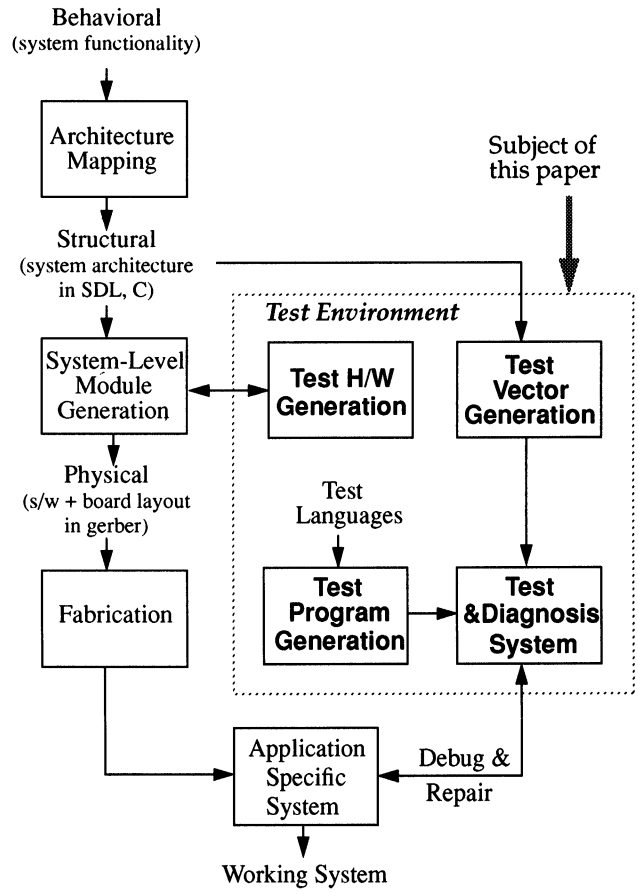

FIGURE 1 Structure of SIERA including test environment.

TEST HARDWARE—CHIP LEVEL

This section describes the chip level test hardware required to implement the Scan Path, BIST, and Boundary Scan test methodologies. Implementation issues and design versus test cost trade-offs are also discussed.

The Boundary Scan Standard—An Overview

Two continuing trends are increasingly making it more difficult to test printed circuit boards:

1. Increasing complexity—As chips become more complex, so does the task of generating tests for boards that use them. For functional testing, test generation times are significantly longer, due to the need to propagate test data through some chips while tests are applied to others. Test lengths also increase as complexity increases.
2. Greater miniaturization—The use of multi-chip modules as well as surface mount packaging technology, particularly when coupled with double-sided component mounting reduces board geometries making boards more difficult or impossible to probe using traditional bed-of-nails access.

The purpose of the Boundary Scan [9] standard is to provide the basis for solutions for these problems. Boundary Scan solved these problems by eliminating the need to physically probe a component's I/O pins by implementing an electronic probe inside the component's I/O pins. This section describes the prin-

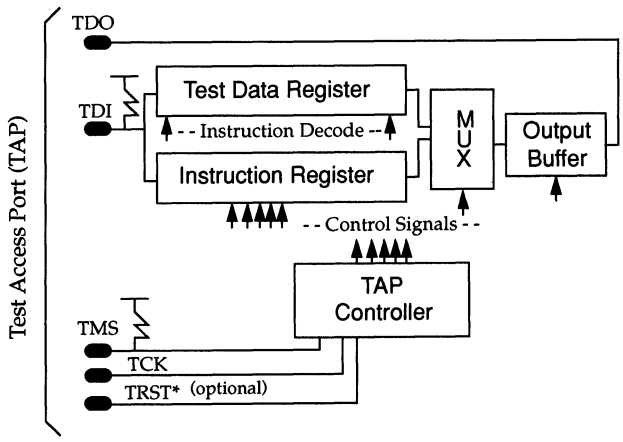

FIGURE 2 Top level view of Boundary Scan test logic.

ciple features of the Boundary Scan Macrocell. The standard architecture requires three major circuit blocks as shown in Figure 2 and described below:

TAP Controller—a finite state machine that responds to control signals supplies through the test access port (TAP) and generates the clocks and control signals required for correct operation.

Instruction Register—an n-bit shift register whose contents determine which test is to be executed

Test Data Register—an n-bit shift register that applies the test stimuli or conditioning values required by a test. At the end of a test, the results in the test data register can be shifted out for examination. This register, for example can be implemented as a Scan Path register.

Designing Chips with Boundary Scan

When incorporating the Boundary Scan architecture into a chip, there are several issues that must be addressed. The decision to incorporate this architecture in a chip first should be considered from a board or system level point of view. If Boundary Scan is to be used as a system requirement, then it is very important to define the instructions at the system or board level first, and then implement the appropriate instructions on the chip. As a minimum, the standard requires that a chip must be able to execute three instructions: BYPASS, SAMPLE, and EXTEST.

To implement the logic required by the standard, the designer must either design the Boundary Scan test circuitry manually or use an existing library module. The module must consist of a minimum of four basic building blocks; a Boundary Scan register, a TAP controller, an instruction register, and a bypass register. The standard also contains a number of design requirements that must be followed to ensure that the chip works properly with other Boundary Scan chips.

Boundary scan macrocell

The design of the Boundary Scan architecture should be a structured, parallel effort that complements the natural top-down design style associated with each chip. SIERA achieves this using a hardware generator called JTAG_MOD that automatically implements the Boundary Scan architecture has been developed. It is written in the SDL language and requires that the designer provide parameters such as boundary scan register length. The designer provides parameters to determine the length, size, and shape of the module. To accommodated designs where the designer is constrained to a limited chip core area, a library of input and output pads have been developed that contain boundary scan registers inside them. The serial scan path is constructed by simple abutment of pad cells that form a frame around the border of the chip.

Integrating scan path and BIST with boundary scan

The Scan Path test methodology can be easily supported by using a private SCANTEST instruction. When this instruction is selected, the Test Mode Select (TMS) signal, which controls movement between controller states, acts like the test mode control for a traditional Scan register, which causes movement between shift and load. Test data can be loaded and shifted by toggling the TMS signal. For chips that employ a single scan path, the Test Data Input and Test Data Output pins become the SCANIN input and SCANOUT output. Multiple scan paths can be supported by multiplexing the serial inputs and outputs onto normal package pins.

In [13], the idea of integrating BIST with Boundary Scan was introduced. Similar to the Scan Path case, the Boundary Scan circuitry can be supplemented to provide test support for BIST applications. By selecting the RUNBIST instruction, which is one of the public instructions used to support BIST applications, and placing the TAP controller in the appropriate state for as many clock cycles as is required to execute the self-test a, BIST circuitry can be easily controlled through the Boundary Scan test bus interface.

Embedded memory testing is one of many applications that is ideally suited for BIST. In embedded memories, the address, data, and control inputs may not be directly controllable and the data output may not be directly observable at a chip's input and output pins. Further, the test patterns for memories are required to detect a wide variety of complex faults. These faults are different from classical stuck-at faults. Since, the use of Scan based design techniques do not simplify the problem of testing embedded RAMs, it becomes cost-effective to incorporate BIST features in embedded memories to avoid time consuming test generation. The BIST feature of embedded memories provides vertical testability of the RAM, not only at the board level, but at the system level as well.

SIERA uses a module generator called RAMBIST to produce the BIST hardware required for RAM testing. RAMBIST consists of a counter, exclusive-OR gates, several multiplexers, and a Linear Feedback Shift Register (LFSR). An LFSR is formed by ex-oring the outputs of two or more of the flip-flops together and feeding them back into the input of one of the flip-flops. The counter is used to supply the test patterns to the embedded RAM and to control the testing sequence. Unlike typical BIST techniques which use an LFSR as a pseudo-random pattern generator, the counter provides a deterministic set of patterns necessary for thorough testing of the RAM circuitry. The LFSR in this BIST approach is used to compress the output data from the RAM using signature analysis techniques. The designer also provides the parameters to determine the size of the RAM and the length of the counter and LFSR.

Trade-Offs: Designs Costs vs. Test Costs

The impact of Boundary Scan as seen at the chip level is much more profound that at both the board and system levels. This is because Boundary Scan at the chip level affects the I/O pins, which consequently affects the package size, the overall gate count, and the performance due to the additional delay seen at the I/O pins. The Boundary Scan standard requires a minimum of four extra I/O pins. This overhead is always required, but is less obvious in larger package devices. The number of gates required to implement Boundary Scan is primarily driven by the number of chip I/O pins. The reason for this is because the standard requires each functional I/O pin to have a boundary scan register cell. Naturally, chips with low gate counts and a high number of I/O pins will have proportionally more gates to implement the Boundary Scan architecture. The standard also requires a 2-bit instruction Register as a minimum, but longer Instruction Registers are allowed to implement additional application specific instructions. The following formula can be used to estimate the gate count overhead (functional I/O pins are only counted):

$$\text{Gate Count} = (\text{TAP}) + [(\text{IR}) * (\text{IR bit width})] + (\text{BP}) + [(\#\ \text{I/O pins}) * (\#\ \text{gates/pin})].$$

$$\text{TAP} = \#\ \text{gates in TAP Controller}$$

$$\text{IR} = \#\ \text{gates in Instruction Register}$$

$$\text{BP} = \#\ \text{gates in Bypass Register}$$

$$\text{IO} = \#\ \text{of unidirectional I/O pins}$$

Boundary scan register cells introduce a propagation delay in the data path that is equivalent to that of a 2-1 multiplexer. A typical value for the cells in SIERA in 2um CMOS technology is on the order of 3.0 nanoseconds.

TEST HARDWARE—BOARD LEVEL

Testing at the system or subsystem level is not always accomplished by simply using a set of testable chips unless they are properly integrated at the board level. Traditional board level testing consumes a great deal of time and requires special hardware and complex Automated Test Equipment for each type of board or device. This ultimately results in increased development time. One approach to the problems associated with traditional board level testing is to incorporate DFT techniques that allow embedded testing to be performed. For example, scanned in values can initialize states before testing, and testing can be done while the component is embedded within a board. Board level testing can be made easy with Boundary Scan components. These components can be used to effectively partition and isolate sections of a board for quicker fault isolation.

Furthermore, these components can eliminate physical access problems and provide the designer with access to and control of hard to access nodes on the board. Not only do these components perform functions such as buffers, transceivers, latches, and flip-flops, but they also include components that perform dedicated low level test functions. These low

level test functions can be easily combined to create high level test functions. Some of these high level test functions are used in a prototype design called the Test Master Controller board which is used to control and access the Boundary Scan components of a target board.

This section deals with the requirements, design, and implementation of the hardware that is used to support board level testing, which includes a Boundary Scan components library, dedicated board level test modules, a custom Test Master Controller board, and a discussion of board level trade-off issues.

Boundary Scan Component Library

Due to widespread adoption of the Boundary Scan standard by the commercial ASIC industry, many Boundary Scan components are beginning to appear on the market. Since most board designs include octal devices such as buffers, latches, transceivers and flip-flops—for bus operations, manufactures have provided families of octal chips that support the Boundary Scan standard. These octals can replace their standard IC counterparts to enhance the testability at the board level. When used in their test mode, these octals offer designers a number of useful test features such as pseudorandom pattern generation and parallel signature analysis.

A variety of Boundary Scan components which are manufactured by a number of ASIC vendors are organized into a test component library. As an example, a partial listing of an SDL file, described in Section 2.1, for an octal buffer is shown in Figure 3. The file provides a black-box footprint of a generic TTL LS244 component where the I/O terminals names are the same but the pin mappings are different for a given package type. In this example the JTAG contains all of the Boundary Scan implementation specific information for this particular device. This information includes the size of the Boundary Scan (BSR), Instruction (IR), and Bypass registers (BPR) and a variable indicating whether the chip is a Boundary Scan slave (MASTER 0) or master (MASTER 1) device.

The local variable PKGLIST is used to define the various package types that are available for the given part. PKGCODE is a Lisp function that checks to see if the package type value, which in this case can be a dual-in-line or small outline or leadless chip carrier package, assigned to PKGTYPE by the user is a valid one. The other user parameter, JTAG, is used to distinguish between a non-Boundary Scan

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;NAME:xx244.sdl ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 (parent-cell xx244 (PACKAGECLASS PCB)
(bag JTAG (BSR 18) (IR 8) (BPR 1) (MASTER 0)))

 (parameters ;
;pre-defined local variables
(PKGLIST '("DIP" "SOIC" "PLCC") (local))

 ;user parameters
   (PKGTYPE "SOIC" (assert (memql PKGTYPE PKGLIST)))
   (JTAG 1 (assert (or (= JTAG 0) (= JTAG 1))))

 ;local variables determined from parameters
(PKGCODE (list-index PKGTYPE PKGLIST) (local))

 ;oct2rinf variables
(PARTNAME (sel_jtag "74BCT8244" "74LS244")) (PHYSNUMBER (sel_jtag (sel_pkg
"L-GEN24" "L-GENSO24WB" "L-PLCC28SA") (sel_pkg "L-GEN20" "L-GEN-
SO20WB" "L-PLCC28SA")))

 ;info for interconnect test pattern generation
(PARTTYPE "DIGITAL"))
(layout-generator NONE) ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;NETS AND
TERMINALS ;;;;;;;;;

 (net A ((parent (term A (PINNUMBER (pin (sel_jtag
  (sel_pkg (23 22 21 20 19 17 16 15)
  (23 22 21 20 19 17 16 15) (6 5 4 3 2 27 26 25))
```

FIGURE 3 Partial listing of SDL file for a TTL LS244.

and a Boundary Scan component. By default, values for both user parameters are chosen for the user to bias designs to use Boundary Scan components housed in surface mount packages and most importantly, this is the first step toward automating the process.

The PARTNAME and PHYSNUMBER variables contain information that is required by the board layout generation tool, while the Lisp functions sel_jtag and sel_pkg are used to select the correct part name and part number based on the chosen package type. For example, if the PLCC package type chosen and JTAG is 1, the sel_pkg and sel_jtag will select part name "74BCT8244", part number "L-PLCC28SA" and generate the correct pin mappings for this package. The CONDITIONAL property is used to create the additional nets and terminals for a Boundary Scan component. The TERMTYPE and DIRECTION properties to facilitate netlist checking.

Board Level Test Modules

Three dedicated test modules which perform specific testing functions. These board level test modules are

intended to be used as building blocks for higher level testing functions are described in this section. Therefore, a designer can enhance the testability of their board design by simply adding one or more of these modules to their design. The modules can also be accessed and controlled via Boundary Scan test bus. The functionality of each of the modules is determined by groupings of one or more of the test components contained in the Test Component Library.

These modules have been created using the hardware module generation method described in the section on hardware module generation. A library of these reusable board level modules has been created and is listed in Table 1 along with their corresponding functions. Although the number of library elements is small, it will continue to grow as the demand for modules with more sophisticated test functions increases. The modules are usually specified in a hierarchical SDL file describing the structural interface between its primitive test components. The SDL file also contains floorplanning information. These modules do not require any parameters or placement information because their primitive components have already been pre-placed. Other useful modules include the data acquisition and the clock generation both of whose layouts are shown in Figure 4.

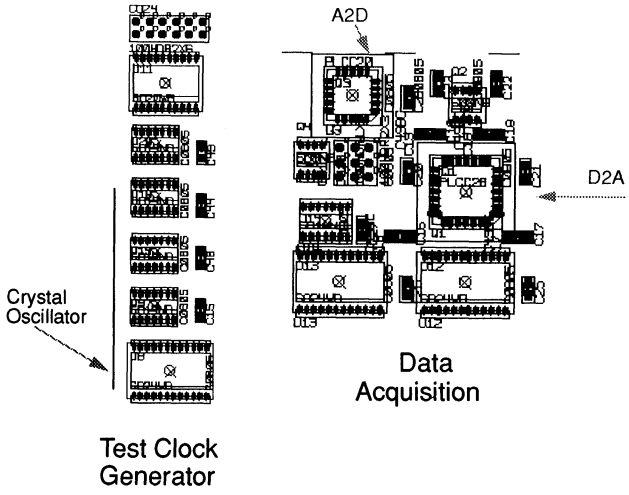

FIGURE 4 Layout of data acquisition and clock generator modules.

TEST HARDWARE—SYSTEM LEVEL

When the application boards are finally assembled to form a complete system, they must be tested while it operates in the environment for which it was developed in order to verify its correct operation and diagnose any failures. A high level view of the Test and Diagnosis System is shown in Figure 5. It consists of a VME card cage for housing the application boards, a single-board CPU that runs a real-time customizable operating system kernel, an Ethernet board (not shown in the figure) for communicating with a Unix workstation, which is used for software development and debugging, and a Test Master Controller (TMC) board [14, 15] to access and control the test hardware residing on system components. To support system level testing, the following requirements were imposed:

1. be capable of accessing and executing chip level test structures;
2. provide control sequences to enable proper execution of chip level test structures;
3. apply test data and collect and analyze test results;
4. provide a facility for analyzing test results;
5. be able to test the board interconnection between various components on a board via Boundary Scan registers;

TABLE I
Functional definitions of board level test modules.

Module Name	Module Function
1149.n Master Controller	A software programmable test master controller that can be configured to implement any one of the IEEE 1149 or custom serial test protocols. This feature is achieved by using a Xilinx FPGA.
Local Boundary Scan Master	A Boundary Scan test bus controller module that supports efficient transfer of serial data and control to and from target devices on the local serial test bus.
Real-Time Monitor	Provides a facility for monitoring embedded digital signal paths between components on a board. Can be used to reveal timing-sensitive and/or intermittent failures that are otherwise undetectable without the use of external equipment.

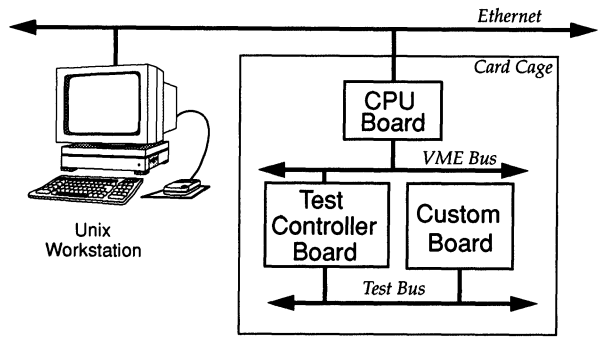

FIGURE 5 Test and Diagnosis system.

6. be compatible with the existing hardware development;
7. operate at system clock rates;
8. be flexible enough to support a variety of testability bus standards such as Boundary Scan;
9. provide access and control of non-Boundary Scan, as well as, analog components for mixed signal applications.

Test and Diagnosis System

To fulfill the requirements mentioned above, the Test and Diagnosis System uses the hierarchy of test buses which are embedded into the system's physical hierarchy as shown in Figure 6. In this hierarchy, each chip contains a Boundary Scan interface; all Boundary Scan devices on each board are serially cascaded forming a single scan path where all of the control and test data are applied through a centralized Boundary Scan slave interface; all Boundary Scan slave interfaces on every board are tied to the Boundary Scan master interface on the Test Master Controller board, where test programs direct the execution of all test functions for the entire system; at the next level, the CPU board is used to initialize the Test Master Controller board, which is described in the next section; and finally, at the top-most level, the UNIX workstation provides the user interface for the Test and Diagnosis system where test vectors are automatically generated and test results are analyzed.

DESIGN EXAMPLES

The chip and board level test hardware described in the previous sections has been implemented on a prototype chip and printed circuit board. Both designs were facilitated by the SIERA design system which reduced the design effort from several months to several weeks.

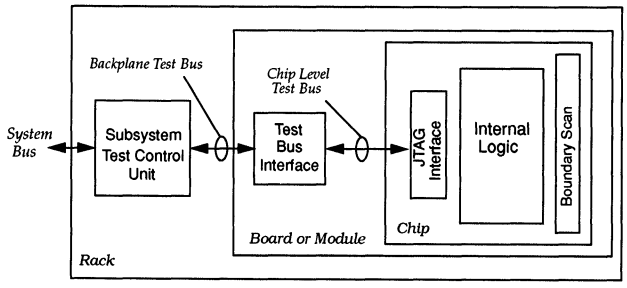

FIGURE 6 Hierarchy of test buses.

A Prototype Boundary Scan Chip

A prototype chip called TEST_CHIP1 has been implemented in 2u CMOS technology using the MOSIS fabrication facility. Its layout is shown in Figure 7. In this chip, the boundary scan register sits at the periphery of the chip's internal core circuitry and its architecture consists of a data path, a JTAG_MOD module, and a Scan register. The overhead is large in this case since a minimum amount of non-test hardware was included.

Test Master Controller Board

The Test Master Controller board is used to control the test process of a target board by accessing each components' DFT structures via Boundary Scan bus. The TMC transmits test data to and from every component under test in the system. It also receives instructions and data, which are provided by the user, from the Unix workstation. These instructions determine which tests are to be executed for the targeted chips on the application board. After a test has been executed, the results are gathered and uploaded to the UNIX workstation where they can be analyzed. Further, it can access a chip's DFT structures through a Boundary Scan interface. Finally, it can be dynamically reconfigured to suport other testability bus standards that use a 5 wire serial test access port.

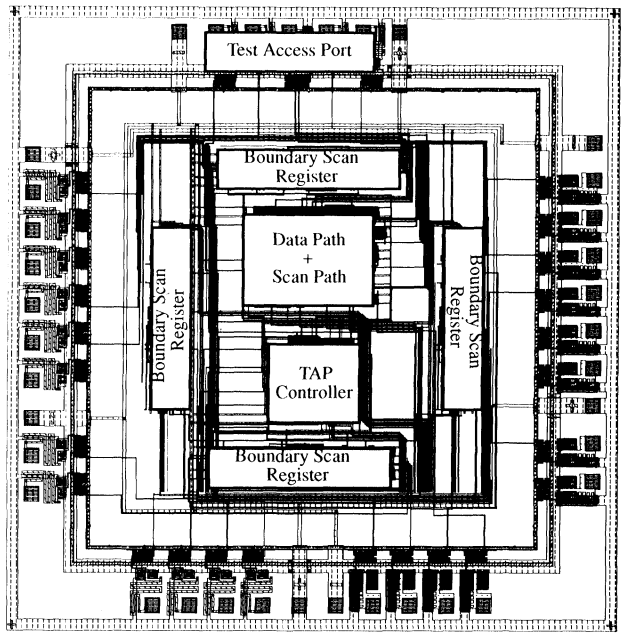

FIGURE 7 Layout of TEST_CHIP1.

The TMC board architecture, shown in Figure 8, was designed with modularity and reusability in mind. Breaking the architecture into smaller, more manageable parts simplifies TMC board testing and reuseability allows us to re-use these modules in other designs. A brief description of the modules is given below:

VME Bus Interface Logic—implements the VME bus protocol which orchestrates communication between the TMC and the CPU boards.

Control Register—contains execution specific control information required to configure the TMC to operate in one of its test modes.

Status Register—contains all of the system specific status information such as test completion signals, boundary scan path integrity checking information, and other information pertinent to proper system operation.

Clock Generator—a jumper programmable clock generator that produces a two phase non-overlapping clock that operates at clock rates up to 50MHz.

1149.n Test Module—a software programmable test controller module that consists of three smaller modules; a data acquisition, memory, and 1149.n controller.

The data acquisition module contains one 12-bit user programmable Analog-to-Digital Converter that operates up to 100KHz, and one 8-bit Digital-to-Analog Converter that also operates up to 100KHz. This module is provided for testing mixed-signal systems. The memory module which consists of 2 256k × 1 bit SRAM, one for storing the test data to be applied to the device under test, and the other for storing the results that are captured at the end of a test. The 1149.n controller module is the heart of the system is also made up of several smaller dedicated test modules which will be described in the next section.

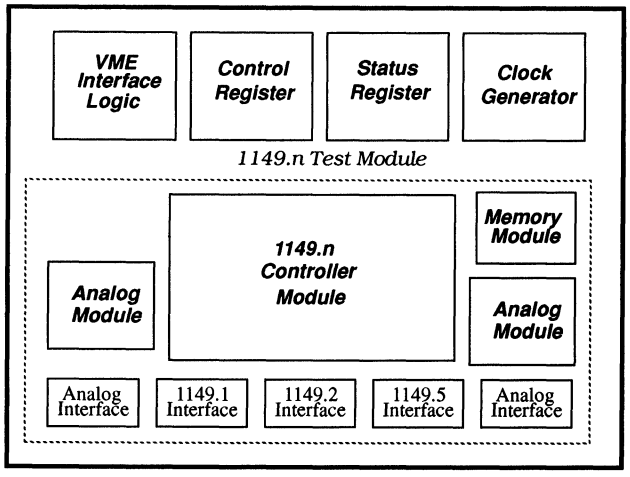

FIGURE 8 Test Master Controller board architecture.

It can be reconfigured using software to implement any one or all of the IEEE 1149 [9, 10, 11] standard bus protocols. In fact, the controller can be configured to perform any custom test protocol provided it uses a five wire serial port. A simplified version of the state diagram for the controller is shown in Figure 16. After initialization, the controller begins in the idle state, from which point it can traverse any one of the branches depending on the value of the test mode (TM) signal. For example, when TM = 0, the controller executes the Boundary Scan bus master protocol. Likewise, the controller will execute any of the other IEEE 1149 bus protocols exercise any BIST features of the devices, or apply/capture analog data when TM = 1, 2, or 3. Physical ports exist for the IEEE 1149 buses and the analog module.

The TMC board was implemented on a 6 layer 6 inch × 9 inch printed circuit board that contains over 200 surface mount components (chips, capacitors, switches, etc.). Component placement and routing was done using the Racal-Redac PCB design system [20]. The actual use of the board will depend on whether a centralized or distributed control strategy adopted. In the distributed approach, most of the test functionality is implemented in dedicated hardware that resides on the target boards, whereas, the centralized approach uses the TMC board to implement all board level test functions. The benefits of both of these approaches are outlined below:

Centralized Approach:

- Cost—By centralizing all test functions to a single controller, test sequencing capabilities are not required for each of the target application boards. This can reduce the cost of the test interface and control hardware on each board in a system.
- Simplicity—As the test interface on each board does not contain any board-specific test information, a common test interface can be used on each of the application boards in a system.

Distributed Approach:

- Software—Because distributed test hardware and software allow higher level test functions, less software is required for the TMC board for each of the target application boards. Distributed software can reduce the software development time.

- Bus Traffic—Since self-test routines and test patterns are contained on the individual application boards, test bus traffic is restricted to instructions and compressed test results.
- Test Application Speed—The distributed approach facilitates concurrent testing of individual boards, allowing substantial reduction in overall system test time.

TEST SOFTWARE: TOOLS AND LANGUAGES

As described in the first section, the objective of our test strategy is to integrate test into our system design environment. To realize this strategy requires dedicated software tools. These tools should automate the addition of the test hardware required to implement a DFT methodology, while sparing the designer having to know about any implementation specific details. After adding the test hardware, test patterns can then be applied to test the target chip or board interconnect via local test buses. Producing these tests manually is an arduous and tedious task that is very prone to error, especially for large designs. On the other hand, generating test patterns automatically eliminates these problems while producing them efficiently and error free. Furthermore, the test pattern generation task is ideal for automation because many algorithms exist for both combinational circuits and printed circuit board interconnect.

The widespread adoption of the Boundary Scan standard has necessitated the need for a way to simply and effectively describe its implementation specific details in a manner suitable for software to utilize. The Boundary Scan Description Language (BSDL) [14] was developed for this purpose. Two other languages called the Chip Test Language (CTL) [17] and the Module† Test language (MTL) [17] which describe how to use the test features implemented on a chip and board for testing have been developed at the University of Southern California. These two languages have been produced to allow the designer to write high level test procedures which are later compiled to automatically produce test programs (written in C) which control the operation of Test Master Controller board. These languages and translators were integrated into the test system to provide a high level user interface.

†In the context of their work, the term module and board are used interchangeably.

Testability Hardware Design Tools

To ensure design for testability, the system designer must follow a methodology that addresses testability issues as part of the design process. Much published work on CAD tools, that are now available, support a testability design methodology at the chip level only. However, there also exists a need for tools that support testability design methodologies at the board level. One such tool called JTAGtool described here has been developed to ensure that every Boundary Scan chip, in a board design, is correctly connected to the scan chain.

Some test applications may require the use of a custom test protocol or some new standard comes along requiring a new test protocol, in either case, the architecture of the Test Master Controller (TMC) board is flexible enough to support them. As described in the preceding section, the TMC uses a programmable device for this purpose. A tool called PLDS [29] is used to map a behavioral representation of the test protocol to the target programmable device, which in this case is a Xilinx Field Programmable Gate Array [27]. The JTAGtool and the procedure for using PLDS to reconfigure the 1149.n Controller module for the TMC board will be discussed in the sections that follow.

JTAGtool: boundary scan path routing tool

The role of JTAGtool is twofold: one, it threads all of the Boundary Scan chips in the design as they appear in the design hierarchy, and two, it generates a file containing the design netlist, which is used in the Module Test Description [17]. This tool also eliminates any errors that may otherwise occur when the designer has to manually configure the Boundary Scan path. A block diagram of JTAGtool, which consists of three modules, is shown in Figure 9. In the Process-Facet module, the structure_instance view [4, 21] that contains all of the structural information of the design that has been created from a hierarchy of SDL files is flattened down to the PACKAGECLASS property. This will allow JTAGtool to preserve the order of the Boundary Scan chips during creation of the Boundary Scan path. The MakeBScanPath module performs the following tasks:

1. Identifies all of the Boundary Scan master and slave chips present in the design:
2. Cascade all Boundary Scan slave chips in the order in which they appear in the design with

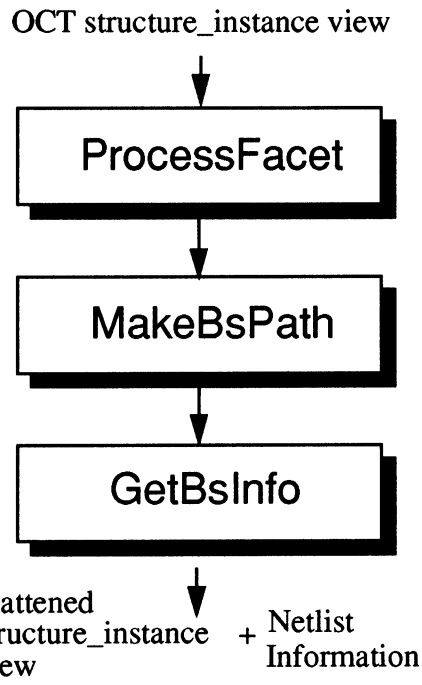

FIGURE 9 Block level diagram of JTAGoal.

the TDI of the first chip connected to a global TDO net and the TDO of the last chip connected to a global TDI net.;

3. The TMS and TCK pins of every slave chip are connected to global TMS and TCK nets;
4. The global nets TDI, TDO, TMS, and TCK are all connected to a local Boundary Scan master controller chip that is either added automatically by a test module or added manually by the board designer.
5. Finally, the TDI, TDO, TMS, and TCK nets are connected to a 10 pin right angle connector which is to be placed manually by the board designer.

Lastly, the GetBSinfo module extracts all of the pertinent information required for board interconnect test generation such as the board net list.

Test Controller Configuration Tool

One of the most important features of the Test Master Controller board architecture is its reconfigurability. By reconfigurability, we mean hardware that can be changed dynamically or hardware that must be adapted to different user applications. Commercial devices such as Field-Programmable Gate Arrays (FPGAs), in particular, the Xilinx XC4000 Logic Cell Array family, exhibit this feature. These devices can be dynamically reconfigured an unlimited number of times. Xilinx FPGAs comprise three major configurable elements: configurable logic blocks (CLBs), input/output blocks (IOBS), and interconnections. CLBs provide the functional elements for implementing the user's logic. IOBs provide the interface between the package pins and internal signal lines. The programmable interconnect resources provide routing paths to connect the inputs and outputs of the CLBs and IOBs. Reconfiguration is established by programming internal static memory cells that determine the logic functions and their interconnect.

Figure 10 illustrates the reconfiguration procedure. The procedure is partitioned into two steps. In the first step, a file describing the behavior of the 1149.n device to be implemented is used as an input to PLDS whose objective is to provide a solution to efficiently map a high-level description of a design into a set of one or more programmable devices. It provides an interface between the Oct database and commercially available tools supplied by the manufacturer which in this particular case is Xilinx. PLDS produces output files, in Xilinx Netlist Format, which are then used by the Xilinx XACT Development

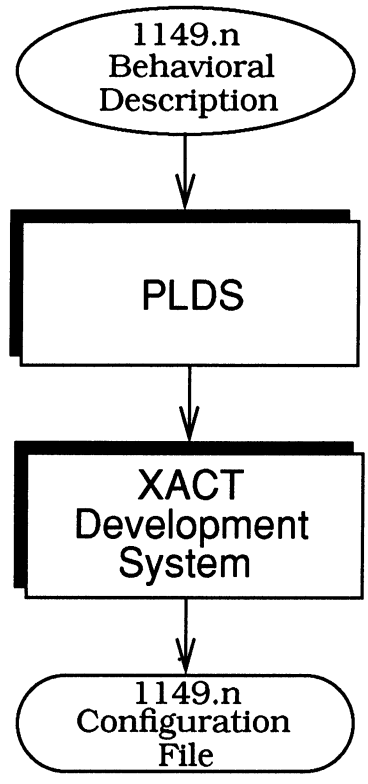

FIGURE 10 Configuration file generation process.

System [26]. Finally, after running the design through the XACT software, an 1149.n configuration file is generated which must be down-loaded to the Test Master Controller board to configure a XC4005 FPGA during system initialization.

TGS—a test vector generation tool [23]

The Test Generation System (TGS) from the University of Southern California was integrated to provide a method for generation of test vectors for combinational circuits described at the gate level. The gate types supported by the system include AND, OR, NAND, NOR, INV (inverter), BUF (buffer), and INPT (input gate). The system provides fault collapsing, test vector generation, and fault simulation. This system is also capable of executing a complete test generation procedure without any user intervention, once the required input parameters are set up.

Before combining the combinational logic blocks of a chip with its other functional blocks such as ALUs or RAMs, the combinational blocks must be processed using oct2tgs [3] which is a conversion utility developed at Mississippi State University that generates a TGS circuit description from a flattened OCT structure_instance view (default view) or symbolic view of the chip. It decomposes macro cells, such as multiplexers or decoders into primitive logic descriptions which are given in the technology file. Further, it ignores latches, like the scanlatch in the stdcell library, and treats the logic between the latches as an independent logic block and later combines them to form a top level TGS input file. After running this input file through the TGS system to produce test vectors, they can be used to test the combinational logic blocks via Scan Path.

SYSTEM DESIGN AND TEST PROCEDURE

The use of the Te11st and Diagnosis System along with the test software provide the designer with a structured procedure for verifying, debugging, and testing systems designed using SIERA. This procedure can be divided into four steps:

1. Hardware Design;
2. Generate Test Vectors;
3. Generate Test Programs;
4. Compile Programs and Execute.

In the first step, the system designer selects the test hardware modules that satisfy system test requirements and adds them to the SDL files describing the design. OCT2TGS and TGS are used in the second step to generate test vectors for the system's combinational blocks. Next CTL and MTL files are written and compiled into C code which is eventually downloaded to the CPU board, and executed.

CONCLUSION

The work reported in this paper not only automates the process of incorporating testability into the SIERA design system, it also provides dedicated hardware and software for controlling the test circuitry that has been added to each level of the system's hierarchy.

The text hardware incorporation was automated by hardware module generators. These generators relieve the designer of having to know how to implement a specific DFT methodology and they guarantee correct implementation. Test circuitry is added to an already existing design supports the JTAG Boundary Scan standard, as well as traditional test methodologies such as Scan Path and Built-In-Self-Test. Implementation issues associated with each of the methods were easily dealt with using our hierarchical test strategy. Through integration of previously developed languages and translators, the mundane tasks of writing test programs for a target system are automated by the test program generation software. This software extracts the necessary information required to generate these program from several high level testability description languages. Various issues related to the incorporation of test into the system design flow, test vector generation, and testability hardware description languages are also addressed in this work. With this system, prototypes designed with SIERA can be functionally verified in a timely fashion. This work is one step closer to the ultimate goal of total automation of all the tasks associated with system verification, debugging, and testing. The hierarchical structure of this system makes it easier to incorporate new test methodologies and testability bus standards. Two design examples are shown, a Boundary Scan chip and the Test Master Controller board. The software reconfigurability feature of the TMC board make it extremely flexible and useful.

Future enhancements to this work may include: hardware and software support for AC parametric tests like delay fault testing, investigation of algorithms that produce more efficient board wiring test

vector sets, and use of a hierarchical testability hardware description language like the Hierarchical Scan Design Language (HDSL) developed by engineers at Texas Instruments.

ACKNOWLEDGMENTS

This project was funded by DARPA. We would like to thank John Andrews and Jay Brown of National Semi-conductor and Texas Instruments for providing us with free JTAG samples and XILINX for donating software.

REFERENCES

[1] L. Avra, "A VHSIC ETM—Bus Compatible Test and Maintenance Interface", *Proc. Int'l Test Conf.*, pp. 964–971, September 1987

[2] D. Bhavsar, "An Architecture for Extending the IEEE Standard 1149.1 Test Access Port to System Backplanes," *Proc Int'l Test Conf.*

[3] A. Bomdica, "oct2tgs—Users Manual," Mississippi State University, 1990.

[4] R.W. Brodersen, (ed.), "Anatomy of a Silicon Compiler," *Kluwer Academic Publishers,* Boston 1992.

[5] E. Eichelberger, T. Williams, "A Logic Design Structure for LSI Testing," *Proc. 14th Design Automation Conf.*, pp. 462–468, June 1977.

[6] S. Funatsu, N. Swkatsuki, T. Ar4ima, "Test Generation Systems in Japan," *12th Design Automation Conf.*, pp. 112–114, June 1975.

[7] D.S. Harrison, et. al., "Data Management and Graphics Editing in the Berkeley Design Environment," *Proc. Int'l Conf. Computer-Aided Design,* pp. 24–27, November 1986.

[8] J. Hallenbeck, et. al., "The Test Engineer's Assistant: A support Environment for Hardware Design for Testability," *IEEE Computer,* pp. 59–68, April 1989.

[9] IEEE Std. 1149.1-1990, "IEEE Standard Test Access Port and Boundary Scan Architecture," February 1990.

[10] IEEE Std. 1149.2/D0.2, "Extended Digital Serial Subset," February 1991.

[11] IEEE Std. P1149.5/D0.2, "Standard Backplane Module Test and Maintenance Bus Protocol," April 1990.

[12] IEEE Std. P1149.lb/D1, "Supplement (B) to Standard Test Access Port and Boundary Scan Architecture," October 1991.

[13] J. Jarwala, et. al., "A Framework for Boundary-Scan Based System Test Diagnosis," *Proc. Int'l Test Conf.*, pp.993–999, September 1992.

[14] K.T. Kornegay, "Automated Testing in an Integrated System Design Environment," *Ph.D. Dissertation,* Memorandum No. UCB/ERL M92/104, September 1992.

[15] K.T. Kornegay, R.W. Brodersen, "An Architecture for a Reconfigurable 1149.n Master Controller Board," *Proc. Int'l Test Conf.*

[16] J. LeBlanc, "LOCST: A Built-In-Self-Test Technique," IEEE Design and Test, November 1984, pp. 45–52.

[17] J. Lien, "Design of Hierarchically Testable and Maintainable Systems," *Ph.D. Thesis,* July 1991, Univeristy of Southern California, CEng Technical Report 91–19.

[18] C. Maunder, R. Tullosss, "The Test Access Port and boundary Scan Architecture," *IEEE Computer Society Press,* 1990.

[19] E.J. McCluskey, "Built-In-Self-Test Techniques," *IEEE Design and Test,* pp. 21–28, April 1985.

[20] Racal-Redac Inc., "VISULA-PLUS User's Guide."

[21] C.S. Shung, et. al., "An Integrated CAD System for Algorithm-Specific IC Design," *IEEE Trans. on CAD,* April 1991.

[22] M. Srivastava, "Rapid-Prototyping of Hardware and Software in a Unified Framework," *Ph.D. Thesis,* June 1992.

[23] USC Test Group, "Test Generation System (TGS) User's Manual—Version 1.0", USC, Dept. of Elec. Eng.-Sys., Technical Report No. CENG 89–03.

[24] M. Williams, J. Angell, "Enhanced Testability of Large Scale Integrated Circuits via Test Points and Additional Logic," *IEEE Trans. on Computers,* January 1973.

[25] T. Williams, K. Parker, "Design for Testability—A Survey," *proc. of the IEEE,* pp. 98–112, January 1983.

[26] Xilinx Inc., "User's Guide and Tutorials," 1991.

[27] Xilinx Inc., "The XC4000 Data Book," 1992.

[28] C. Yau, "The Boundary Scan Master: Target Applications and Functional Requirements," *Proc. Int'l Test Conf.*, pp. 311–315, September 1990.

[29] R. Yu., "PLDS: Prototyping in LAGER using Decomposition and Synthesis," *M.S. Report,* U.C. Berkeley, May 1991, ER Memo. No. UCB/ERL M91/53.

Biographies

KEVIN T. KORNEGAY received the B.EE. degree from Pratt Institute (Honors) in 1985. He received a M.S. and Ph.D. degrees in Electrical Engineering and Computer Sciences from the University of California, Berkeley, in 1990 and 1992, respectively.

He is currently a Research Staff Member in the Manufacturing Research Department at the IBM Thomas J. Watson Research Center. He was the recipient of an AT&T Cooperative Research Fellowship from 1986–1992. His research interests include design for test, VLSI systems design, and CAD for VLSI. He is a member of Eta Kappa Nu, Tau Beta Pi, and the IEEE Computer Society.

ROBERT W. BRODERSEN received B.S. degrees in Electrical Engineering and in Mathematics at California State Polytechnic University, Pomona, California in 1966. From the Massachusetts Institute of Technology, Cambridge, he received the Engineers and M.S. degrees in 1968, and a Ph.D. in 1972.

From 1972–1976 Dr. Brodersen was with the Central Research Laboratory at Texas Instruments, Inc., Dallas, where he was engaged in the study of operation and applications of charge coupled devices. In 1976, he joined the Electrical Engineering and Computer Science faculty of the University of California at Berkeley, where he is now a professor. In addition to teaching, he is involved in research involving new applications of integrated circuits that is focused in the area of signal processing and the CAD tools necessary to support this activity. He has won best paper awards at a number of conferences, and in 1979 he received the W.G. Baker award. In 1982 he became a Fellow of the IEEE and in 1983, he was co-recipient of the IEEE Morris Liebmann award. In 1986 and 1991, he received the Technical Achievement awards in the IEEE Circuits and Systems Society and the Signal Processing Society. In 1988 he was elected to be a member of the National Academy of Engineering.