Fault Tolerance and Testing with Boundary Scan

by

Alan Ptak

A thesis
presented to the
University of Manitoba
in partial fulfillment of the requirements
for the degree of

Master of Science

Department of
Electrical and Computer Engineering
University of Manitoba
Winnipeg, Canada 1990

ISBN 0-315-63258-5

FAULT TOLERANCE AND TESTING WITH BOUNDARY SCAN

BY

ALAN PTAK

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This thesis concerns the use of boundary scan to improve fault tolerance and testability in digital systems. A detailed overview of boundary scan is provided, and an implementation using University of Manitoba 3 micron CMOS standard cells is presented. A major design feature is cellular automata built-in self-test (CA-BIST) incorporated into the boundary scan register. Estimates for area overhead are presented based on data from an actual layout. A hierarchical test methodology is outlined for WSI systems incorporating boundary scan. An end-of-production and operational testing strategy and a concurrent fault detection and isolation method are also described.

Acknowledgements

The assistance and guidance of Prof. Bob McLeod throughout my graduate program was invaluable, and is gratefully acknowledged. Dave Blight, Roland Schneider and the other wizards in the graduate program provided valuable assistance with the design tools and software packages in the VLSI lab. I would also like to thank the management of Transport Canada, Central Region for supporting my efforts and my interest in postgraduate studies.

This work was supported through funding from Transport Canada and the Natural Sciences and Engineering Research Council of Canada, and through the equipment loan program of the Canadian Microelectronics Corporation.

Dedication

To Myra,
whose encouragement and patience made this work possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The demand for dependable electronic hardware has skyrocketed with the widespread use of computers and digital systems. Advanced fault tolerance techniques are needed to meet the challenges of critical systems requiring high reliability, high availability, low life cycle costs, and long operational life.

Dependable systems must able to determine the operational status of its components. Fault tolerant systems use built-in testing and system diagnosis to autonomously locate and isolate faults, and to reconfigure the remaining components for continued operation. Conventional (non-fault tolerant) systems also employ testability features and effective system diagnosis methods to allow rapid fault detection and location. In either case, the complexity of test equipment and the time required to effect repairs by maintenance personnel is significantly reduced.

Fault detection and fault location are accomplished through *testing*. **Implicit testing** detects errors which occur during normal system operation, while the component is *on-line*. Parity, arithmetic, and error detection/correction codes are examples of techniques used to accomplish implicit testing. Additional hardware and communications bandwidth is used

to construct fault tolerant, self-checking and fail-safe components. In contrast, **explicit testing** is performed when the component is *off-line*. Deterministic or random test vectors are applied to the circuit's inputs, and the outputs produced are compared with the expected results. Explicit testing is used at all stages of a component's life cycle, from fabrication to maintenance and repair.

The *testability* of a logic circuit refers to the degree of difficulty involved in applying test data and analyzing the test results. **Controllability** and **observability** are two important measures of how easily a logic signal can be controlled or observed from the circuit's external connections. Circuit nodes can be directly probed in systems consisting of simple (SSI or MSI) components on printed wiring boards. However, since direct access to internal circuit nodes is not possible for VLSI and WSI technologies, indirect techniques are needed.

The term *design for testability* (DFT) describes adhoc guidelines and structured techniques to make circuit testing easier and therefore more economical. Provisions for testing are incorporated in the circuit at the design stage to improve the test coverage, fault detection and fault location. Systems constructed with such components can be made more reliable and can continue to operate in the presence of faults.

Structured DFT includes the use of scan techniques and built-in self-test. Scan-based techniques improve the controllability and observability of a circuit. Scan paths are formed from modified register elements (flip flops or latches) that can be configured as a shift register chain for serial access to internal circuit nodes. The *boundary scan* technique involves placing a boundary scan cell adjacent to each physical device pin to observe and control the component's signals at its boundaries. Built-in self-test (BIST) reduces the cost of test-

ing by building dedicated test logic into the circuit to generate test data and to analyze the outputs produced.

A number of problems encountered in testing large circuits can be resolved with scan techniques. A potentially unlimited number of internal nodes can be accessed through a small number of dedicated external connections. Large circuits can be partitioned into several smaller ones, reducing the number of test vectors needed for a given test coverage.[1] Sequential circuits are reduced to easily-tested blocks of combinational logic by using scan-type devices in place of conventional flip flops or latches. Boundary scan allows a circuit to be isolated from the rest of the system, so that it may be tested in place (*in-circuit testing*), and so that interconnections between components can be tested.

A serious limitation of scan-based testing can be overcome by incorporating BIST circuitry into the scan cells. Without BIST, each test vector must be shifted serially into the scan register one bit per clock cycle. The cost of testing increases in direct proportion to the length of the scan path, typically by at least an order of magnitude. With BIST, test vectors are calculated within the circuit, at the rate of one per clock cycle, and test results are compacted at the circuit outputs. BIST reduces the cost of external test equipment by addressing the problems of test generation, storing large test vector sets, and analyzing voluminous test results. The main disadvantage of BIST is that more test vectors are needed as compared to the scan path technique with deterministic precomputed test data. However, the use of BIST with the scan path approach does not preclude using deterministic test data as well.

The objectives of this work are to investigate boundary scan as a technique for built-in

[1] *Test coverage* is the percentage of faults that can be detected with a set of test vectors, for a particular fault model. *Test coverage* and *fault coverage* are used synonymously.

self-test in large digital systems, critical applications, and parallel processing systems, and to develop a systems approach to integrate testability features provided at the circuit level. Wafer scale integration (WSI) is envisioned as the implementation technology for parallel processor applications.

The boundary scan technique with BIST is the subject of Chapter 2. A detailed overview of the JTAG/IEEE Test Access Port and Boundary Scan architecture is provided, and the design of a boundary scan register incorporating cellular automata BIST (CA-BIST) is discussed. An implementation using University of Manitoba 3-micron CMOS standard cells is presented. The area overhead is calculated from data obtained from an actual layout, and the effect of process scaling is estimated. This implementation has been submitted for fabrication through resources provided by the Canadian Microelectronics Corporation.

Chapter 3 discusses the application of boundary scan to large digital systems and to Wafer Scale Integration (WSI) systems in particular. An overview is presented for a hierarchical test methodology to integrate control of testing activities at the system level. A strategy for end-of-production and operational testing is described, and a fault detection and location technique is presented.

The final chapter presents a brief summary, conclusions and recommendations for future work.

Chapter 2

A Component-level Test Controller

This chapter describes the architecture and the operation of the JTAG[1]/IEEE Test Access Port (TAP). The TAP is used as a model for the component-level test and maintenance controller (CMC) in the hierarchical test and maintenance system described in the following chapter. A design example is presented and the area overhead is analyzed for 3, 1.2 and 0.8 micron CMOS implementations. Appendix A provides specific information for the logic designer wishing to use this implementation.

2.1 Functional Requirements of the CMC

The purpose of the CMC is to integrate the various built-in testing (BIT) resources of individual components[2] at the module level. The CMC simplifies access to testing resources by providing a uniform protocol for all components in the module. To perform in-circuit testing concurrently with normal system operation, the CMC must support the following

[1]Joint Test Action Group

[2]The meaning of the term *components* depends somewhat on the scale of integration and the fabrication technology. For WSI, *component* refers to a distinct processing element or computational unit with its own integral CMC. For VLSI a *component* is likely to be a separately packaged IC.

functions:

1. Provide a slave interface for communication with the test controller[3] (MMC), to allow test vectors and seed values to be written into test data registers, and to allow test results to be read out.

2. Allow read and write access to various control, status and data registers used to interact with the component's normal operating mode.

3. Generate the necessary clock and control signals for the built-in test logic.

4. Isolate the circuit from the rest of the system during testing.

The Test Access Port and the Element Test and Maintenance (ETM) interface[34] are proposed for the CMC. The TAP and the ETM-bus interface are similar serial bus interfaces requiring four and six lines respectively to communicate with the test bus master (MMC). The TAP is chosen in this work over the ETM-bus interface for several reasons. The TAP provides all the necessary functions to operate as the CMC, and it requires 2 fewer signal lines than for the ETM-bus. Also, the TAP is a more recent proposal, and there is a wide and growing base of support for the TAP within the testing industry and among the manufacturers of electronic components.

[3]**MMC** refers to the dedicated test bus controller described in a later section dealing with the hierarchical test control system. **MMC** will be used when referring to the test bus controller, be it a dedicated test controller or some other type of automated test equipment (ATE).

2.2 The Test Access Port

The IEEE proposed standard P1149[10] describes a system-level interface between automatic test equipment and the test hardware incorporated into electronic systems. The goal of this standard is to provide in-circuit test capability for complete systems at the module level, and to reduce the cost of testing. The standard defines the signals and their function for 4 subsets of the interface bus:

1. Minimum serial digital system (MSDS)
2. Extended serial digital system (ESDS)
3. Real-time digital system (RTDS)
4. Real-time analog system (RTAS)

The MSDS subsystem is used at the component, chip or circuit board levels as an interface to built-in testing resources. The proposed standard P1149.1 defines the protocol and implementation for a testing interface which conforms to the MSDS subset[11]. This interface, referred to as the Test Access Port or TAP, supports three essential activities for in-circuit testing:

1. Testing the connections between components.
2. Testing the component internally.
3. Observing the activity of the component's circuitry during normal operation.

The basic parts of this interface are presented in this section. The interested reader is referred to [11] for the complete specification of the TAP and the boundary scan architecture.

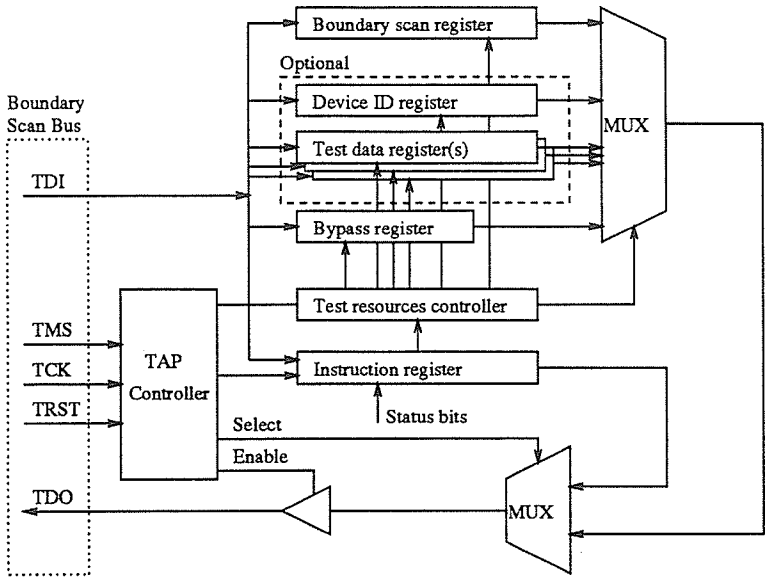

Figure 2.1: The Test Access Port Architecture

2.2.1 Architectural Description

Figure 2.1 shows a block diagram of the TAP architecture. The major functional units are the TAP controller, instruction register, boundary scan register, and optional test data registers.

External access to built-in test resources is accomplished through the dedicated test bus, consisting of the **test data input** (TDI), **test data output** (TDO), **test mode select** (TMS) and the dedicated **test clock** (TCK). The **test logic reset** (TRST) input is an optional control signal which provides an asynchronous reset for the test logic. These signal lines are connected to the test bus controlled by the MMC. Several CMC's are assumed to be connected to the same test bus. TDI on the first CMC on the bus is connected to the MMC, as is TDO on the last. The TDI and TDO connections on the remaining CMC's are chained together to form a ringed bus structure.

The TAP controller configures the component for either normal operation or for one of the built-in testing modes. All clock and control signals for the internal test circuitry are generated by the TAP controller. The instruction, bypass and boundary scan registers are mandatory. Optional registers include the device identification register, internal scan path registers, and design-dependent test data registers. The contents of these registers are accessed serially through TDI and TDO. A connection from TDI to TDO is provided through one of the internal registers at all times.

The instruction register is selected by applying a specific bit sequence to the TMS input. The instructions shifted into the register from the TDI input are decoded to select the data register to be connected to the TDI and TDO pins, and the hardware test to be performed. The instruction register is a serial shift register with parallel inputs and outputs. The register size is design-dependent, but it must be at least two bits in length, since there are three mandatory instructions.

The bypass register provides a direct serial connection from TDI to TDO through a single-bit shift register. The operation of the test logic is not affected by data passing through this register. The MMC can use the bypass register to reduce the logical length of the test path serial data path which passes through many components in a module. This allows faster access to a given component than would be possible if any of the other data registers were selected.

The boundary scan register consists of the serial connection of single-bit register cells located adjacent to each I/O connection to the component. A separate register cell is required for each input, output, bi-directional and three-state connection, including clock signals. The design and operation of this register is described in detail in a following sec-

tion.

Optional test data registers include the device identification register and design-specific registers associated with BIST, scan path, and other testability features. Scan paths through operational registers can be provided to allow the component's state (represented by the contents of these registers) to be examined and set. This capability is necessary to implement certain system-level diagnostic techniques, such as the roving spares method described in the following chapter.

The identification register allows the MMC to automatically determine the manufacturer, part number and version number for all components connected to the test bus. The component identification register described in the IEEE TAP standard consists of 32 bits arranged to conform to industry standards for manufacturer and part numbering schemes.

2.2.2 The TAP Controller

The TAP controller is a synchronous state machine which generates all clock and control signals for the test logic. The state of the controller is determined by the TMS input signal on the rising edge of TCK. All outputs from the controller depend only on its present state.

Figure 2.2 shows the state diagram for the controller. The state space consists of the Test Logic Reset state, scanning cycles for the instruction register (IR Scan cycle) and the data registers (DR Scan cycle), and the Run Test/Idle state.

The Test Logic Reset state disables the test logic to allow the component to operate in its normal mode, and causes a synchronous reset for the test logic. With the TMS input held at logic 1, the TAP controller will reach the Test Logic Reset state from any other state within at most five TCK cycles. The IDCODE instruction is automatically loaded into

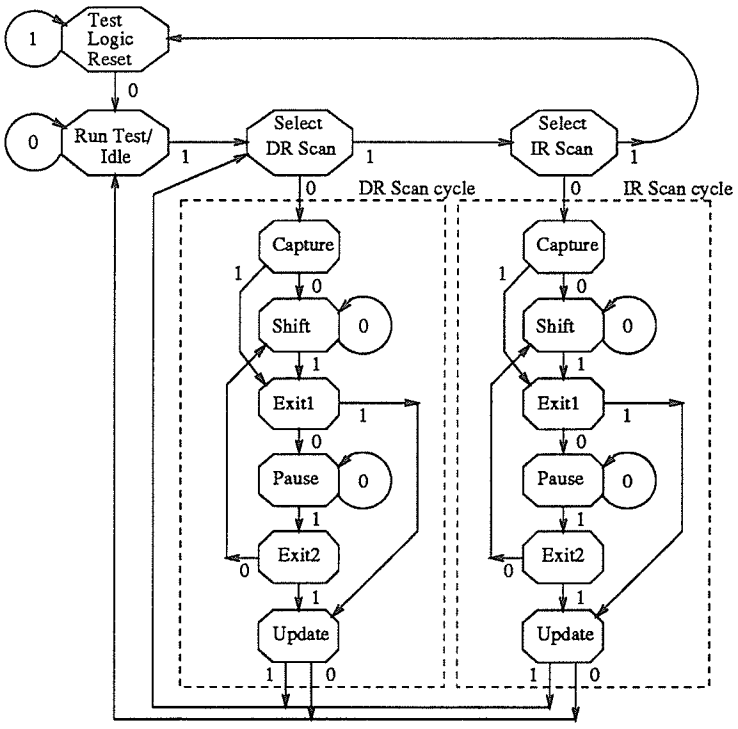

Figure 2.2: The TAP controller state diagram

the instruction register when the controller enters this state, connecting the identification register to TDI and TDO. This allows the MMC to identify all components on the test bus without interrogating each one individually.

The IR Scan cycle is used to load a new instruction into the instruction register. During the IR-scan cycle, the TAP controller connects the instruction register to TDI and TDO. The Capture-IR state loads data from the parallel inputs onto the serial path connecting the register stages. The two least significant bits of the instruction register are always loaded with 01 (binary). The value(s) loaded in the remaining bit position(s) are design-dependent. Using these "free" parallel inputs, the MMC can examine status or data bits in the compo-

nent by reading the contents of the instruction register. The Shift-IR state shifts data in the instruction register one bit position towards TDO on each rising edge of TCK, and shifts data on the TDI pin into the most significant bit position. The Update-IR state loads the data from the serial shift register path to the instruction decoding logic connected to the register's parallel output connections.

The DR Scan cycle allows the external test controller to access the contents of test data registers in the component. The instruction latched in the instruction register selects the test data register to be connected to TDI and TDO. Test data registers are serial shift registers with parallel inputs and/or parallel outputs. The Capture-DR state loads data into the test register from its parallel inputs if the register selected has read (*capture*) capability. The meaning of the data loaded into the register is completely design-dependent. The Shift-DR state shifts data in the selected register one bit position towards TDO on each rising edge of TCK, and shifts data on the TDI pin into the bit position nearest the TDI connection. The Update-DR loads the data from the test register's serial data path to the parallel outputs if the selected register has write (*update*) capability.

The Exit1 and Exit2 states in the IR Scan and DR Scan cycles are temporary controller states only. The Pause state causes all test logic to hold its present state of operation, allowing the testing operation to be suspended indefinitely. The test bus controller selects this state to temporarily halt test programs while loading or storing test data in its local memory.

The Run Test/Idle state allows self-testing hardware to operate when the appropriate instruction is present, such as the RUNBIST instruction. The test activity and the test data register selected depends on the instruction present in the instruction register. The Run Test/Idle state can also be used when pausing between register scan operations.

Mandatory Instructions			
Instruction	Code	Register Selected	Test Logic Response
EXTEST	$00\cdots0$	Boundary scan	Set/capture logic values on all circuit interconnects
BYPASS	$11\cdots1$	Bypass	Pass data from TDI to TDO
SAMPLE	—	Boundary scan	Capture logic values on all pins during normal operation
Optional Instructions			
Instruction	Code	Register Selected	Test Logic Response
RUNBIST	—	Boundary scan and test data registers	Execute self-contained self-test of component
INTEST	—	Boundary scan	Apply scanned-in test vector and capture response
IDCODE	—	ID register	Scan out mfgr-pgm'd ID code
USERCODE	—	ID register	Scan out user-pgm'd ID code

Table 2.1: Instruction Summary

2.2.3 The Instruction Set

The behavior of the test logic is defined for three mandatory and four optional instructions. These seven instructions are *public* instructions, meaning their operation must comply with the standard and that they must be fully documented for the user. Public and private extensions to the instruction set are allowed. Table 2.1 summarizes the instructions defined by the standard. The EXTEST and BYPASS instructions must use the specified binary codes to support global instructions broadcast to all components on to the test bus. The binary codes used for all other instructions are design-dependent. Undefined instructions select the bypass register and cause all testing features to remain inactive.

The three mandatory instructions are EXTEST, BYPASS and SAMPLE. The EXTEST instruction allows the interconnections between components to be tested. In this test, test data is shifted into the boundary scan register. Output cells apply the test stimuli while the input cells capture the logic values at the component's inputs. The BYPASS instruction connects the single-bit bypass register to TDI and TDO. The SAMPLE instruction is used to load the logic values at each system pin into the boundary scan register without affecting the normal operation of the component. The component's operation can be verified while it is in operation, by analyzing its outputs externally. The sampling rate is limited by the time required to shift the contents of the scan register out of the all components on the test bus.

The RUNBIST instruction enables the operation of the BIST logic. Test data registers with BIST capability, including the boundary scan register and BILBO-type internal scan register are converted to pseudorandom pattern generators (PRPG's), to generate test data and compress test results. The TAP controller must remain in the Run Test/Idle state for the required number of TCK clock cycles for the test to complete. The MMC holds the TMS input at logic 0 for the duration of the test. After the test is complete, the test signature can be read from the test data register with a pass through the DR Scan cycle.

The INTEST instruction is used for testing with deterministic test data. Individual test vectors are shifted into the component and applied to the circuit inputs. The response vector is captured at the circuit outputs and shifted out for comparison with expected results. The next test vector is simultaneously loaded into the component while the results from the previous test are read out.

The IDCODE and the USERCODE access the component identification register. The

IDCODE instruction selects an identification register programmed during the fabrication of the component, while the USERCODE selects a user-programmable identification register, such as that found in field-programmable devices.

2.3 The Boundary Scan Register

The boundary scan register is formed by concatenating the serial data connections of single-bit register cells placed adjacent to each I/O connection. A register cell is required for each input, output, bidirectional and three-state connection. Each cell is a single shift register stage with parallel input and/or parallel output connections. During the normal operation, system data passes through the cells from the parallel input to the parallel output. The serial mode is used to shift data into and out of the cell in the test mode. The cell's ability to transfer data values between the parallel and serial connections is determined by the design of the logic within the cell. Thus, capability can be traded for reduced area or propagation delay to suit application-specific design criteria. Boundary scan cells for skew-sensitive signals such as clock inputs may be designed to have read-only capability. A read-only cell can be constructed with just one flip flop and one multiplexer and has less impact on the transmission of the signal into the component, as compared to a full-function input cell. The IEEE TAP standard [11] contains design examples for various types of boundary scan cells.

Figures 2.3 and 2.4 show example designs for input and output cells with parallel input and parallel output connections. Positive edge triggered D flip flops are used rather than latches. The terminals labelled **p_in** and **p_out** are the parallel input and output connections respectively. Similarly, **s_in** and **s_out** are the serial input and output connections. The TAP controller generates the global signals shiftDR, clockDR and updateDR. The instruction decoder generates the imode signal, which is also global. For normal operation, shiftDR, updateDR and imode are low, and clockDR is high.

Referring to Fig. 2.3, **p_in** is connected to the external input and **p_out** is connected to an

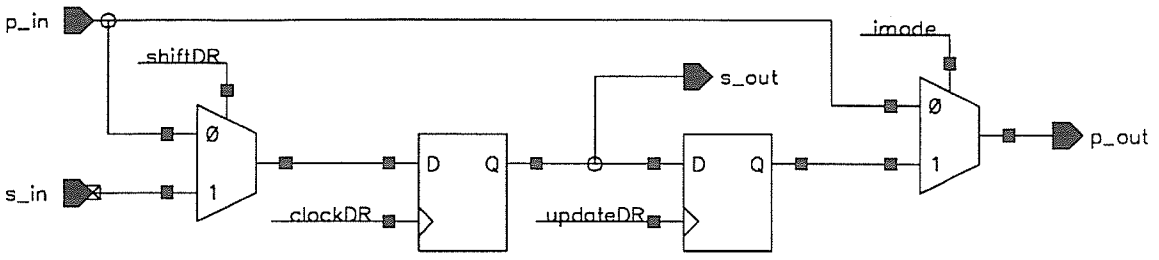

Figure 2.3: Input Boundary scan cell with Capture and Update capability

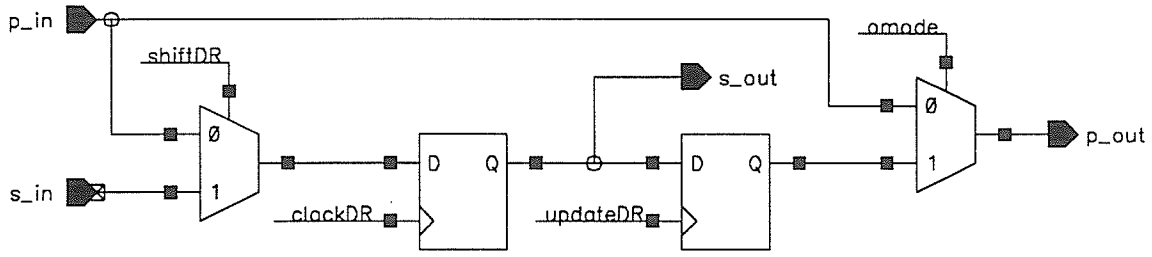

Figure 2.4: Output Boundary scan cell with Capture and Update capability

internal net. The serial connections **s_in** and **s_out** are connected to neighbouring boundary scan cells, forming the serial path through the boundary scan register. The shiftDR and the imode signals are low, and input data passes from **p_in** through the multiplexer to **p_out**.

Data is transferred to the serial data path from the parallel input when the TAP controller is in the captureDR state. The shiftDR signal remains low, and the clockDR signal goes through a single clock cycle to latch the logic state on **p_in** to **s_out**. This sequence of events occurs when the TAP controller is in the Capture-DR state.

Data is shifted along the boundary scan register when the TAP controller moves the shiftDR state. The shiftDR signal is set high so that data is transferred from **s_in** to **s_out** on the rising edge of the clockDR signal. The contents of the boundary scan register are shifted one bit position towards TDO for each TCK cycle that the TAP controller remains

in the shift-DR state.

The final mode of operation occurs when the TAP controller is in the updateDR state. The imode signal is set high and updateDR is active for one clock cycle to latch the logic state on **s_out** to **p_out**.

Modifications to incorporate BIST into these cells are discussed in detail in the next section.

2.3.1 Built-in Self Test

In the absence of BIST, external testing equipment generates and applies test data to the logic inputs for the circuit under test, and collects the test results at the logic outputs. Large and complex circuits require many test vectors for thorough testing. Test equipment that can store large test vector sets and run extensive tests in a reasonable amount of time is expensive. The primary motivations for developing alternatives to this method of testing include the desire to reduce or eliminate large test data sets, and to simplify test equipment for testing large and complex components.

With BIST, test data is generated, applied and compacted within the component. The test controller now needs only to load seed values into BIST registers prior to testing, scan out the test signature at the end of the test, and possibly apply a small set of deterministic test vectors to improve the test coverage. It then becomes feasible to implement these functions as a dedicated test controller which forms a permanent part of the operational system.

The linear feedback shift register (LFSR) is widely used to implement the PRPG's needed for producing test data and compacting test results. Recently, one-dimensional cellular automata have been proposed as an alternative to LFSR's. The requirements for the

design and layout of the boundary scan register make CA-based designs more attractive than the LFSR for several reasons:

1. Boundary scan registers are easier to construct with CA's, since CA's require only nearest-neighbour communications. LFSR's require several feedback lines, including one which spans the complete length of the PRPG register.

2. CA-based pseudorandom pattern generators can operate faster, due to locality in communications between individual cells in the PRPG register.

3. Changes to the design of the boundary scan register can be easily accommodated, since the construction of CA-based PRPG's follows a structured, modular approach.

4. The aliasing properties of CA-based data compactors are better than for LFSR's, which increases the probability that a fault will be detected by the comparison of test signatures. [12]

5. Maximal length random sequences can be produced using combinations of just two type of cells, referred to as rule 90 and rule 150 CA's. This further simplifies the design of the boundary scan register, whether it is manually crafted or produced using computer-aided design (CAD) tools.

Equations 2.1 and 2.2 describe the behavior of the rule 90 and rule 150 CA's respectively:

$$a_i(t+1) = a_{i-1}(t) \oplus a_{i+1}(t) \tag{2.1}$$

$$a_i(t+1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t) \tag{2.2}$$

where $a_i(t)$ is the output of the i^{th} CA at time t. The implementation of rule 150 cells requires one extra 2-input exclusive-or gate.

Zhang and Miller have calculated the construction rules for maximal length random sequences using a minimum number of rule 150 CA's. At most two rule 150 CA's are needed to construct a maximal length pseudorandom sequence, independent of the number of bits in the register [13]. Table 2.2 shows the construction rules for registers up to 16 bits in length. The position of the 1's in the construction rule column of the table indicates the position of the rule 150 cell in the register. For example, a five-bit register can be constructed from one rule 150 and four rule 90 CA's, arranged as follows:

$$GND \rightharpoonup 90 \rightleftharpoons 90 \rightleftharpoons 90 \rightleftharpoons 90 \rightleftharpoons 150 \leftharpoonup GND$$

GND denotes the null boundary conditions for the register. Construction rules for registers up to 150 bits in length can be found in [13].

2.3.2 Integrating BIST with Boundary Scan

Figures 2.5 and 2.6 show input and output boundary scan cells modified to include CA-based BIST. The new inputs **ca-** and **ca+** are connected to the **p_out** nets of the adjacent right-hand and left-hand boundary scan register cells[4] respectively. BISTclk and BIST-mode are global signals from the instruction decoding logic, and are low during the normal operation of the component.

BISTmode is set high when RUNBIST is loaded into the instruction register. BISTclk, a dedicated clock signal for the BIST test mode, is active when BISTmode is high and the TAP controller is in the Run Test/Idle state.

[4]The least significant bit of the register is the one closest to TDO, and is assumed to be located on right.

Register Length	Construction Rule	Rule 150 Cell Position(s)
1	1	0
2	01	0
3	001	0
4	0101	2 0
5	00001	0
6	000001	0
7	0000100	2
8	00000110	2 1
9	000000001	0
10	0001000010	6 1
11	00000000001	0
12	000001000100	6 2
13	0000000010000	4
14	00000000000001	0
15	000000000000100	2
16	0100000000000001	14 0

Table 2.2: Pseudorandom pattern generator construction rules for hybrid CA's. The position of the 1's in the construction rule are the locations of the rule 150 CA's. The remaining cells are rule 90 CA's. The second column explicitly states the positions of the rule 150 cells, referenced to the right hand side of the PRPG.

The BIST clock is combined with the normal flip flop control signals in the register cells. In the input boundary scan cells, BISTclk is gated with the updateDR signal. Since both BISTclk and updateDR are normally low, and they may not be active simultaneously, the logical OR of these two signals provides the needed control signal in the normal and BIST modes. In the output register cells, BISTclk is gated with the clockDR signal. Since clockDR is high when inactive, BISTclk and clockDR are combined with an exclusive-OR gate to produce the proper signal. Table 2.3 summarizes the control signals needed for the operation of three types of boundary scan cells.

Table 2.4 compares the hardware needed for a boundary scan cell with that required

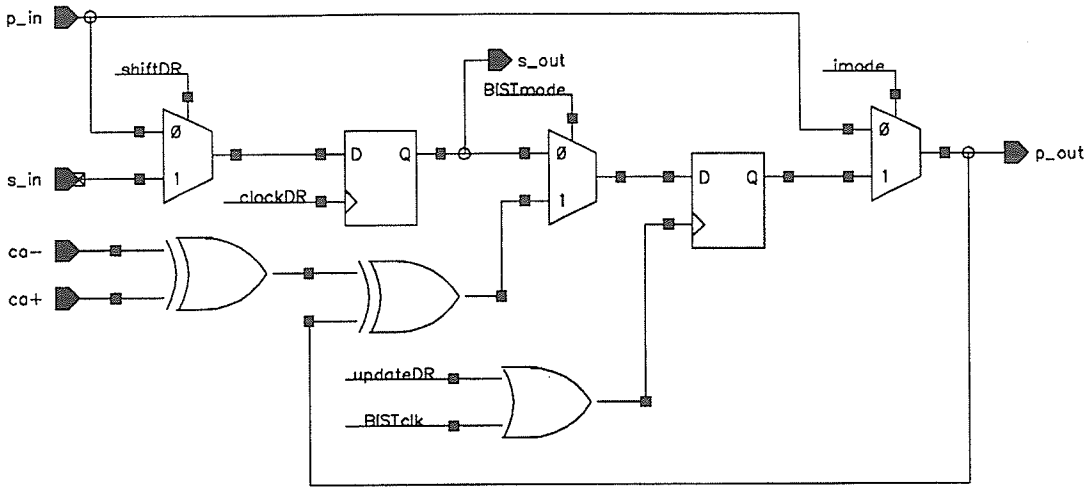

Figure 2.5: Boundary scan cell with BIST for input pins

for the rule 90 and rule 150 BIST scan cells. The multiplexer used in this implementation was constructed with 2-input nand gates and inverters from a standard cell library. A more efficient layout is possible by using a multiplexer constructed with pass transistors. Pass transistor designs require less area and have lower propagation delays, but they are susceptible to latch-up problems and therefore are not considered further in this work.

The preferred physical location for the boundary scan cells is adjacent to the I/O pad,

Signal Name	Boundary scan cell type			
	Input Scan	Output Scan	Input BIST	Output BIST
shiftDR	●	●	●	●
clockDR	●	●	●	●
updateDR	●	●	●	●
imode	●		●	
omode		●		●
BISTmode			●	●
BISTclk			●	●

Table 2.3: Control signals for boundary scan cells

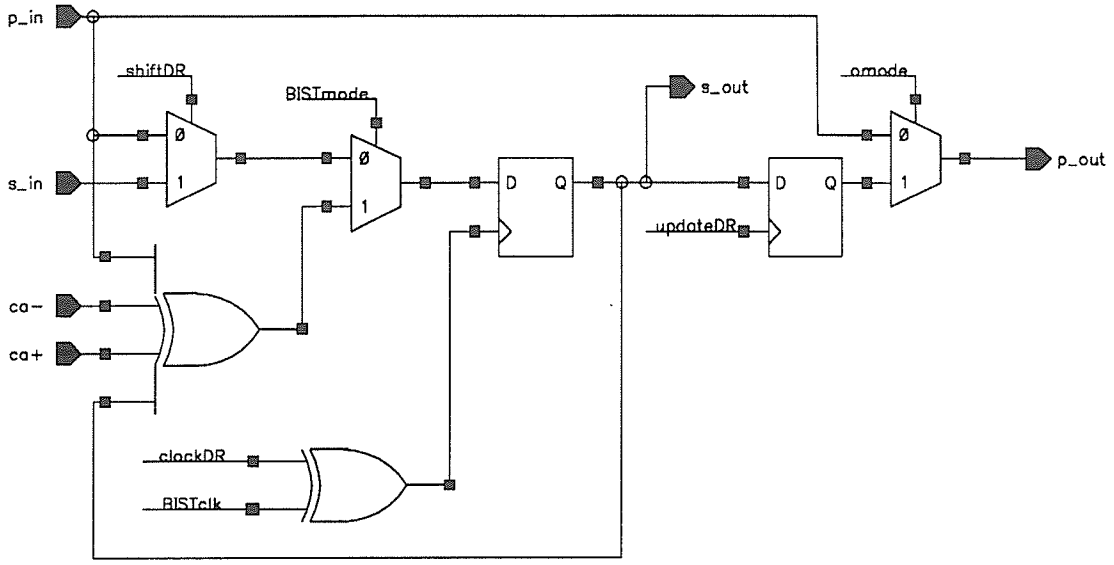

Figure 2.6: Boundary scan cell with BIST for output pins

as indicated in Figure 2.7. The boundary scan circuitry is ideally the same width as the I/O cell to obtain the best use of silicon area. The routing of nets for the boundary scan register cells is simplified by aligning connections for signals between adjacent cells.

Standard cell type	Unit Area (mm^2)	Boundary scan output cell type		
		w/o BIST	Rule 90	Rule 150
D flip flop	0.013	2	2	2
Multiplexer	0.012	2	3	3
2-input XOR	0.003	0	3	4
Total Area (mm^2)		0.050	0.073	0.076
BIST Overhead (%)		—	44	50

Table 2.4: A comparison of the hardware required for boundary scan cells with and without BIST. This table shows the number of logic components for a standard cell implementation for output cells without BIST, and with rule 90 and 150 CA. The total area and the additional overhead for BIST, relative to the boundary scan cell without BIST, is shown. Areas are stated in square millimeters, for a 3 micron fabrication process.

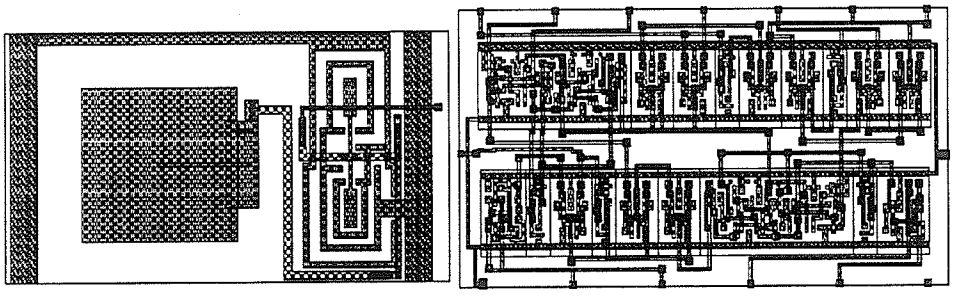

Figure 2.7: A conventional inpad (left) consists of a metal bonding pad, clamping diodes to Vcc and Vdd, and a polysilicon series resistor for protection from electrostatic discharge (ESD). For comparison, a 3.0 micron standard cell implementation of the rule 90 boundary scan/BIST register cell is shown on the right. The area of the inpad cell is 0.094 mm^2, compared to 0.109mm^2 for the boundary scan cell.

2.4 An Analysis of Area Overhead

The TAP controller developed in this work requires approximately 4500 transistors to construct. Table 2.5 summarizes the silicon area required for the TAP controller and boundary scan cells, based on a standard cell implementation using the Northern Telecom 3 micron CMOS technology. In comparison to the total area required for boundary scan, the additional area cost for CA-BIST is insignificant.

Component	Area Required (mm^2)
TAP controller	3.936
Boundary scan cells:	
w/o BIST	0.050
Rule 90 BIST	0.073
Rule 150 BIST	0.076

Table 2.5: Area summary for the TAP controller and boundary scan cells. Areas are stated in square millimeters for a 3 micron process.

Table 2.6 shows the overhead in area for three chip dice sizes available from Northern Telecom through the Canadian Microelectronics Center. The area required for the identification register is not included. The overhead for boundary scan alone reported by Brglez in [14] is 25 to 50% of that shown here, however those figures were based on an earlier

Chip Size	Maximum I/O	Net Working Area		Overhead (%)
		W/O Bdy. Scan	With Bdy Scan	
Oversize dice	120	156 mm^2	140 mm^2	10
A pad frame	40	53 mm^2	45 mm^2	15
B pad frame	32	25 mm^2	19 mm^2	26

Table 2.6: Area summary for boundary scan. The area overhead for the boundary scan circuitry is shown as a percentage of the area available inside the I/O cells. Areas are stated in square millimeters for a 3 micron process.

proposal for the test hardware which was considerably simpler. The design described here is expected to be larger than that found in industry, since our standard cells are not optimal area designs. These calculations assume inputs and outputs are grouped into two registers and that all available I/O are used. The four I/O connections which are required for the test access port itself are not part of the boundary scan register.

As the minimum feature size is further decreased, the area required for boundary scan diminishes to an insignificant level. Figure 2.8 illustrates the effect of technology scaling on the area overhead.

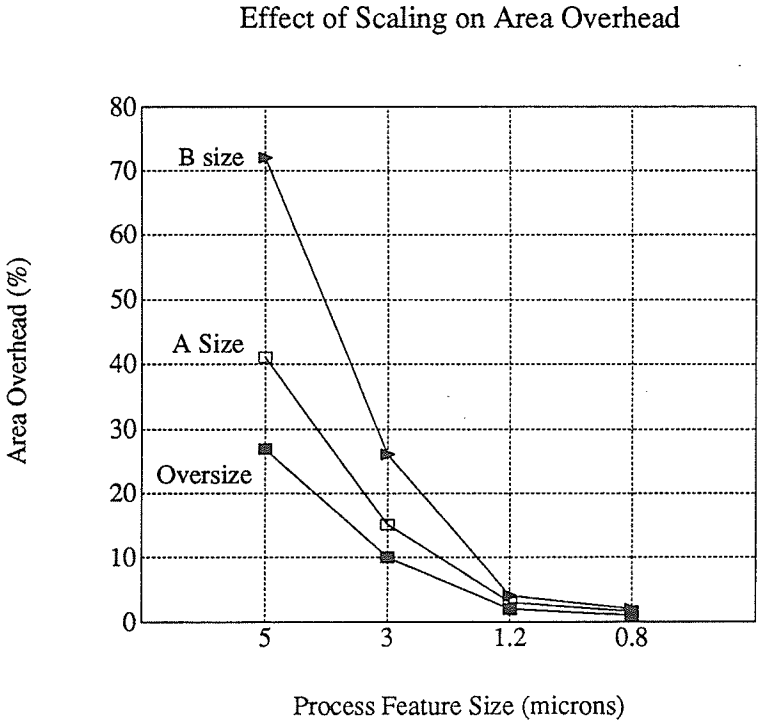

Figure 2.8: Effect of process scaling on the area overhead, for three die sizes and reduced feature size.

Chapter 3

Fault Tolerance and Testing for WSI Systems

Wafer-scale integration (WSI) technology allows fabrication of complete electronic systems or subsystems on a single semiconductor wafer. This technology is attractive because of its potential to provide fast, reliable and compact computing systems. Opportunities exist for WSI in high performance real-time systems and critical applications, including image and signal processing, avionics, real-time reasoning systems and robotics.

The step-and-repeat fabrication process lends itself to system architectures based on arrays of identical processing elements (PE's) or *cells*. An interconnecting network of wires and switches provides a communications path between PE's. An external general purpose computer (the *host*) manages the flow of data into and out of the processor array, and initializes the processors prior to operation. Normally, the host also controls the switches in the interconnection network.

Fault tolerance techniques are mandatory for WSI systems, to improve their yield, reliability and availability, and to extend their operational life. This chapter discusses several fault tolerance techniques using boundary scan. The first section describes a hierarchy of

test controllers which integrates the control of testing activities for the system and supports fault diagnosis and system reconfiguration tasks. A strategy for testing the wafer requiring only the operation of the test bus is then presented. The *roving spares* fault detection and isolation technique is also described. This technique uses the test controllers to support off-line testing concurrently with normal operation of the system. The boundary scan DFT technique is shown to be an essential requirement for testing and fault tolerance.

3.1 A Hierarchical Test Controller

Several recent papers have proposed a hierarchical arrangement of test controllers to integrate the component-level testing resources for an entire system. The Hierarchical test and Maintenance System (HTM) proposed by Breuer and Lien[22] and the multilevel self-test scheme of Haedtke and Olsen[25] are based on the concept of embedded test controllers in the target system. Each level of hardware integration in the target system (chip, module, subsystem and system) has a corresponding hierarchical test program. The test controller at each level translates high level test instructions into lower-level test activities.

The four-level design methodology proposed by Lien and Breuer is shown in Fig. 3.1. At the bottom level, each chip contains a Component test and Maintenance Controller (CMC) in addition to the application circuitry. Modules, printed circuit boards consisting of many chips each, contain a Module test and Maintenance Controller (MMC) to control testing activities via the CMC's. Each subsystem, a complex unit consisting of several modules, has a Subsystem test and Maintenance Processor (SuMP) to control the MMC's within the modules. The System Maintenance Processor (SMP) controls the top level test and maintenance functions by directing the testing activities at the subsystem (SuMP), mod-

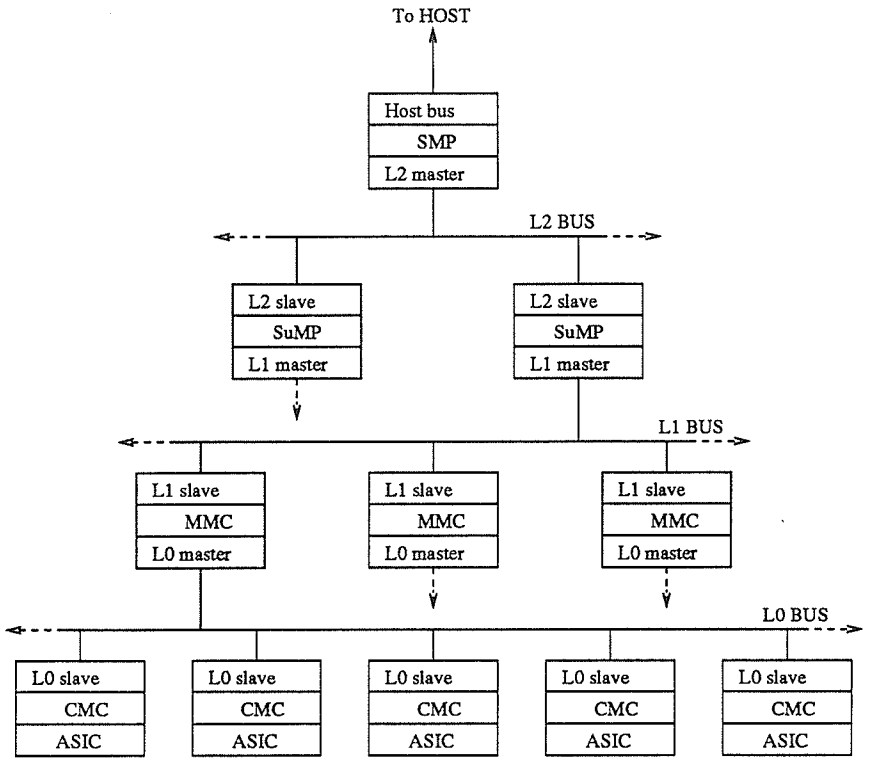

Figure 3.1: The four level testing hierarchy

ule (MMC) and component (CMC) levels.

Dedicated test busses carry instructions, data and test results between the various levels of test controllers. Each MMC is connected to the CMC's within a module by an L0 bus, referred to as the *test bus*. The L0 bus carries instructions, data and hardware control signals from an MMC to the CMC's on the test bus, and returns test results to the MMC. The L1 bus is used for sending commands from the SuMP to the MMC's and for returning test results to the SuMP. The L2 bus carries high-level commands, and status and activity reports between

the SMP and the SuMP's.

Serial busses minimize the the area required and the complexity of the communication interfaces. The IEEE Minimum Serial Digital System (MSDS) bus[11] and the VHSIC Element Test and Maintenance (ETM) bus[34] are proposed for the L0 bus. The MSDS bus, described in detail in the previous chapter, consists of the **test data input** (TDI), **test data output** (TDO), **test mode select** (TMS) and the dedicated **test clock** (TCK) signals. The corresponding ETM bus signals are DATA IN, DATA OUT, MODE and CLOCK. The two additional signal lines for the ETM bus are the SELECT line, and the INTERRUPT signal which allows the CMC to request service from the MMC. The use of the MSDS bus is assumed for the L0 bus in the discussions which follow. Packet-oriented communications protocols are more likely to be used for the L1 and L2 busses, which carry data, instructions and commands in a high-level form.

The usual partioning problem exists for the test controllers, especially at the interface between the CMC and the MMC. The area required for the CMC is greatest when all the test hardware needed for the component self-test is provided on-chip, however this arrangement allows components to be tested simultaneously and in the shortest possible time. High-level instructions initiate tests, and pass/fail results are returned to the MMC. The area of the on-chip CMC can be reduced by shifting some or all of the test hardware to the MMC, in which case the MMC and the test bus are involved in testing to a greater extent. This increases the time required for testing and reduces the number of components which can be tested in parallel. With all the test hardware in the MMC, the CMC is reduced to a simple L0-bus slave, and only one component on each test bus can be tested at a time. A further consideration is the reduction of yield which occurs with increases in the size (complexity) of integrated circuits. This problem is especially acute for wafer scale integration, since

faulty parts cannot simply be cut from the wafer and replaced with working ones. In WSI applications, a primary objective of partitioning may be to minimize the on-chip area for the CMC.

3.1.1 The CMC

The CMC provides a slave interface to the L0 test bus and includes some or all of the test control hardware. In response to instructions from the MMC, the CMC configures the component for testing, and controls the built-in test hardware for logic testing. Depending on the degree of self test provided, the CMC may generate and apply test data, and collect, compress and analyze test results. The CMC may also receive status and error signals from on-chip implicit testing hardware. Access to internal registers and memory arrays may be possible through the CMC. The Test Access Port is used as a model for the CMC.

3.1.2 The MMC

The MMC described by Breuer and Lien[26] is a "universal" test controller in the sense that it supports all forms of testing and test methodologies on an application chip. The MMC provides fault detection, isolation and diagnosis based on a stored system diagnosis algorithm and test results. Translating high-level test instructions into complete test programs, the MMC supplies deterministic and random test data, receives responses to tests, and controls all aspects of the test process. Interconnections between components are also tested by the MMC, using the components' boundary scan registers.

The MMC consists of a microprocessor, RAM, ROM, DMA controller, an L1-bus slave, and one or more *test channels*. Test programs and information regarding the physical and

logical arrangement of the target system are stored in ROM. Test programs consist of stimulus and response vectors, instruction codes for BIST and other on-chip BIT hardware, and codes for the use of test hardware provided by the test channel. Test results from the CMC's are stored in RAM, as are alternate test programs[1] loaded "on the fly" by the SuMP. DMA provides rapid transfer of data between the test channel buffers and memory to allow testing to proceed at the full clock rate.

The test channel consists of a L0 bus master and provides all functions needed for testing. The test channel generates the detailed bus timing signals needed to operate the CMC, freeing the processor to perform other tasks. The processor initiates the DMA controller and test channel by loading instruction codes into appropriate command registers. The test channel and DMA controller then operate independently according to the test program for the component(s) being tested.

The test channel gives the MMC a modular design. A single MMC can simultaneously control several testing channels, since it needs only to initiate tests and collect test results. With this approach, a high degree of testing parallelism can be achieved.

3.1.3 The SuMP

The primary function of the SuMP is to execute test programs in response to commands received from the SMP, and to report back the status of components, modules and subsystems. The SuMP selects tests from a stored set for the particular hardware installed, and instructs the appropriate MMC's to execute them. Some tests may involve several components connected to separate test channels, possibly controlled by different MMC's. Test

[1] Alternate test programs may be required in some cases to improve the fault coverage over that provided by the default test stored in the MMC, or to resolve ambiguity in diagnosis of test results.

results are returned as a pass/fail status, a test signature, or as raw circuit responses, as determined by the SuMP. The SuMP may retest a component, test the interconnections between components, test an alternate set of components or test the MMC itself. The SuMP uses test results to determine the system status, based on a system-level failure diagnosis algorithm. Where redundancy is provided, the SuMP may reconfigure the subsystem to avoid faulty units.

The design of the SuMP is similar to that of the MMC, except for the type of bus interfaces provided and the absence of test channels. The SuMP consists of a microprocessor, RAM, ROM, a L2 bus slave and a L1 bus master.

3.1.4 The SMP

The SMP coordinates the operation of the test system with that of the host, and controls all top-level test and maintenance functions. The self test of the test control system is controlled by the SMP, which is assumed to be fault free. A massive redundancy approach such as triplication with voting (TMR) may be required to ensure the SMP operates without errors. The SMP provides system-level fault diagnosis for both the test control system and the target systems, differentiating between transient, intermittent and permanent faults. The functions of the SMP include the following tasks:

1. Receive status reports from all subsystems.
2. Diagnose the system status based on the test results.
3. Initiate off-line functional tests on the system.
4. Record data in log files for later analysis.

5. Control the system reconfiguration where possible.

The architecture of the SMP is similar to that of the SuMP except for the bus interfaces provided. The SMP consists of a microprocessor, RAM, ROM, an L2 bus master and a bus interface for communications with the host of the target system.

3.1.5 Application to a WSI-based System

Consider a multiprocessor system consisting of a host computer, one or more WSI processor subsystems, and a hierarchical test control system. Individual processing elements (PE's) are assumed to be moderately complex, consisting of approximately 50,000 transistors or more, to justify the overhead for the CMC and built-in test resources. Each PE incorporates a boundary scan register, CMC, and sufficient self-test hardware to exercise the majority of the PE logic. Figure 3.2 shows the WSI array and the details of the test bus connections to processing elements on the wafer.

The L0 test bus and the interconnection network are located in the channels formed by the rows and columns of PE's. The test bus is connected to the CMC in each PE through a simple two state switch, as shown in Figure 3.3. Faulty CMC's are bypassed to maintain the continuity of the serial test bus. The MMC controls these bypass switches through a dedicated output channel. Switches in the interconnection network are also controlled by the MMC, rather than by the host. The host computer manages data input and output for the WSI processor array, and starts and stops computations.

The test control system is responsible for the following functions related to the WSI subsystem:

1. Initial (power-up) testing of all processors on the array.

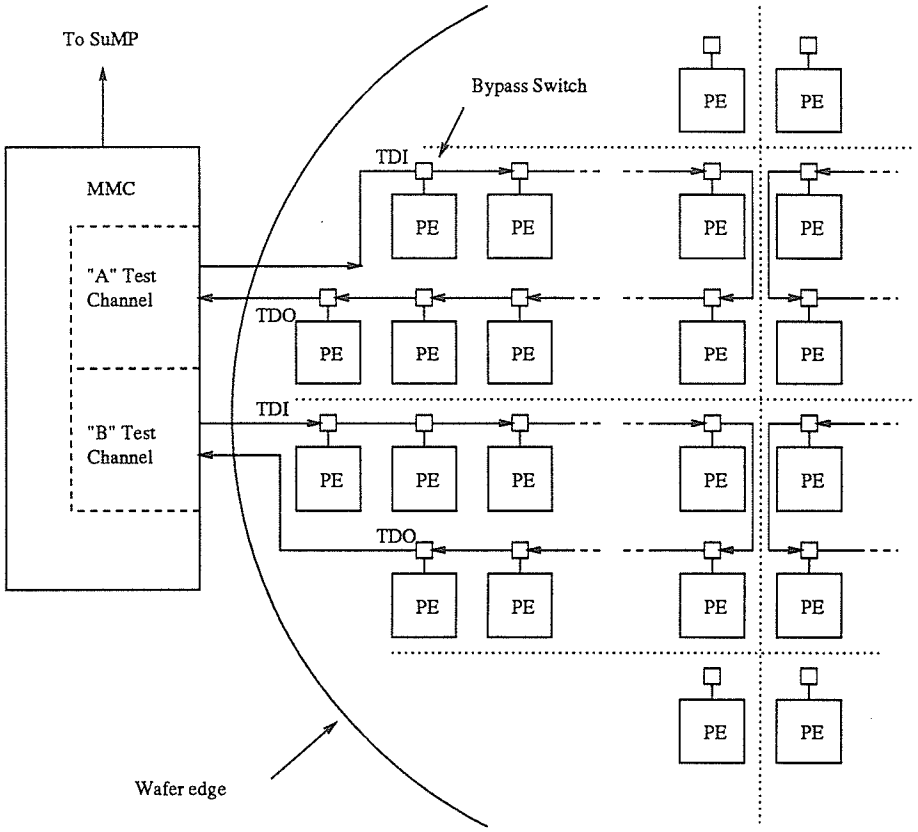

Figure 3.2: A WSI processor array

2. Assignment of resources to the current processing task, in conjunction with the host computer.

3. Control of switch settings to configure the interconnection network.

4. Supervision of fault tolerant techniques to maintain the operational status of individual subsystems and of the system as a whole.

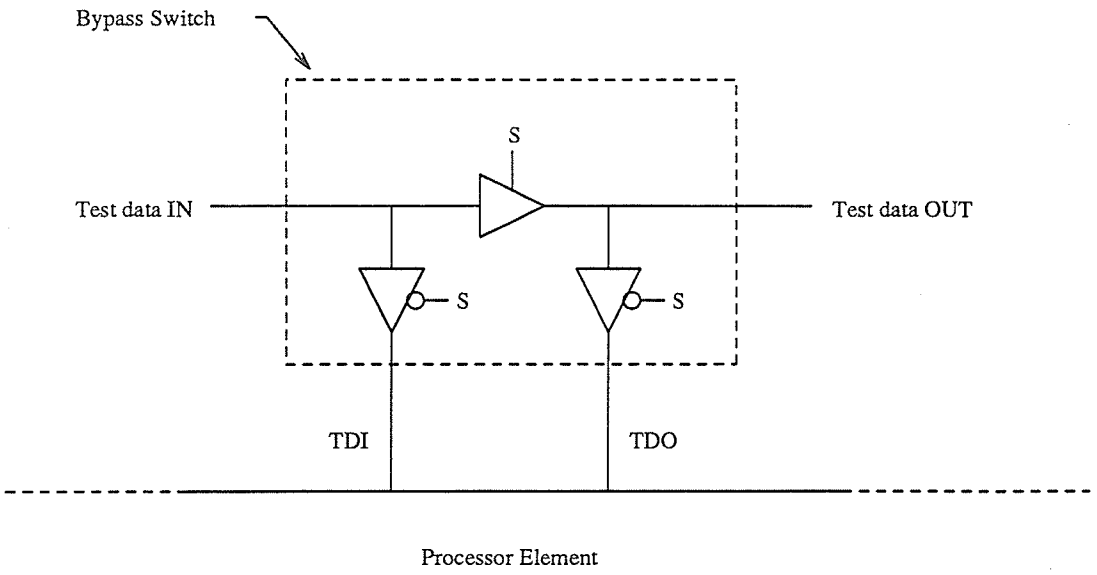

Figure 3.3: The PE bypass switch on the test bus. The control bit S is set by the MMC to either connect the TDI and TDO lines to the test bus, or to bypass the PE. The **test data IN** input to the switch is connected to the previous PE or to the MMC test channel for the first PE on the bus. The **test data OUT** output from the switch is connected to the next PE or to the MMC test channel for the last PE on the bus.

As a lower priority task, PE's previously identified as faulty are retested to flush out intermittent and transient faults. Those PE's which subsequently pass an exhaustive test program can be returned to the set of spares. Normally, testing activity and results are logged for later examination and analysis. An excessive number of transient and intermittent faults may indicate problems with the fabrication process parameters, excessive clock rates or signal skew, or environmental conditions such as temperature variations or power supply fluctuations.

3.2 A Hierarchical Test Strategy

A four step wafer test strategy for end-of-production and power-up testing of the WSI array can be devised using the hierarchical test controller, and the testability features built into the processors. Boundary scan is an essential component for this procedure. References are made to Figures 2.1, 3.2 and 3.3 in the discussion which follows.

The first step is to ensure the continuity of the test bus connected to each test channel. All PE bypass switches are set to bypass mode, reducing the test bus to a string of buffers. A simple test vector is shifted through the test bus. The response vector will be identical when the test bus and the bypass switches are free of faults. This test verifies the operation of the first level of the test hierarchy. When this test is used for end-of-production inspection and repairs, redundant switches or repair techniques can be used to correct for defects. Probe pads adjacent to the bypass switches may be required to locate the fault.

The next step directly tests the continuity of the boundary scan register and indirectly tests the operation of the CMC in each PE.[2] All PE's on a test bus are tested sequentially, with just one connected to the test bus at a time. The CMC is instructed to select the boundary scan register, and a test vector is serially shifted through it. The expected response is simply a delayed version of the stimulus vector. Once complete, this testing stage verifies the operation of the CMC and the shift capability of boundary scan register.

The wires and switches forming the interconnection network are tested by loading the EXTEST instruction into the CMC. This test is similar to that used on interconnects for a printed wiring board, except that it must be repeated for each possible (or required) switch setting on the interconnection network. Output boundary scan cells set the logic value for

[2]Although no explicit test is provided for the CMC itself, its operation is indirectly tested through its use.

individual nets, and input cells observe the logic value on the nets. This test requires the simultaneous control of several PE's, possibly controlled by separate MMC's. The I/O connections to off-wafer data busses are also tested by this means. This testing stage establishes the functionality of the interconnection network and the ability of the boundary scan register to control and observe external logic values. Hard configuration may be applied to repair faults identified in the interconnection network wires and switches.

The final testing stage determines the operational status of all PE's to the extent of the test program provided. Testing the operation of the PE's is accomplished by exercising the application-specific test program. Built-in self test, where provided, will reduce the time required for testing. Sufficient self-test is assumed to maintain a reasonable time for completion of the test program. Testing of all PE's can proceed in parallel for all testing modes including scan testing with deterministic test data, pseudorandom testing using CA-BIST, and self-contained self-test. The test bus and BIST provisions do not exclude the use of other testing approaches which use the normal (operational) data paths for test data.

The subsystem-level test processor (the SuMP) executes a system diagnosis program which takes the results of each testing stage and creates a map representing the functional hardware in the system. System diagnosis algorithms such as *t-diagnosis*[28] can determine the status of hardware external to the PE's, mainly the interconnection network. Diagnosis of test results for the PE's themselves is simpler, mainly based on direct comparison of response vectors or test signatures to expected values.

3.3 The Roving Spares Technique

Several methods have been proposed for system-level fault diagnosis in which computations and diagnostics are performed concurrently. The *roving spares* method described by Shombert and Siewiorek[27] uses spare processors to provide concurrent fault detection and isolation in systolic arrays. Off-line tests on spare PE's are performed concurrently with the on-line operation of the array. Active PE's are systematically replaced by fully-tested spares so that all PE's are eventually tested. When tracked over time, the spare PE's appear to rove through the array. This technique can be useful for WSI arrays, using the boundary scan register and built-in testing resources for fault detection and isolation. The subsystem-level test processor schedules tests and reconfigures the network to accommodate the relocation of active PE's.

Two issues which must be addressed are the control strategy used to determine when and where spare PE's are allowed to move, and the manner in which spare PE's are initialized prior to taking over for an active PE. Active processors have a property of *state*, represented by state variables stored in registers within the PE. The state of the spare PE must be the same as that of the active PE it is about to replace. This can be accomplished by copying the contents of the state register(s) of the active PE into those of the spare using the test bus. Since the operation of the processing array is halted during PE initialization, the principle objectives are (in order of importance) to minimize the effect on system performance, to minimize the activities performed by the external controller, and to minimize the area overhead for each PE.

3.3.1 Roving Control

The control strategy must first determine when a spare PE is operating properly, since only fully functional PE's are allowed to rove. Boundary scan makes in-circuit testing possible, so that the status of a PE is simply determined by exercising the test program.

The control strategy must also decide which active PE will be replaced by the next available spare, based on the principle objectives stated above. The duration of the interruption and the complexity of the initialization task is shortest when the spare and the active PE are connected to the same test bus, as shown in Fig. 3.4. This factor increases as the logical distance between them spans higher levels on the test control system hierarchy. Connectivity limitations in the interconnection network will also restrict the choice to the neighbourhood of the spare, even though the active PE least recently tested should be the one next selected for testing.

3.3.2 PE Initialization

Spare PE's are initialized by duplicating the state variables of the active PE they are about to replace. The state variable registers are assumed to be accessible through the on-chip CMC as a single serial shift register. Three situations are possible when transferring state variables from the active PE **A** to spare **B**.

1. In the simplest case, the spare and the active PE's are connected to the same test channel, as shown in Fig. 3.4. The MMC sets up the transfer by instructing the CMC's in both PE's to connect the state register to the test bus. The CMC's for the other PE's on the test bus are instructed to select the bypass register. The MMC determines the

number of shifts required and loads the value to test channel *shift count register* and loads the test channel shift count register with the number of bits to be shifted. The test channel then serially shifts the state variables from **A** to **B**, in this case through the test channel registers. The operation of the processor array continues after the interconnection network is changed to reflect the new location of the active PE.

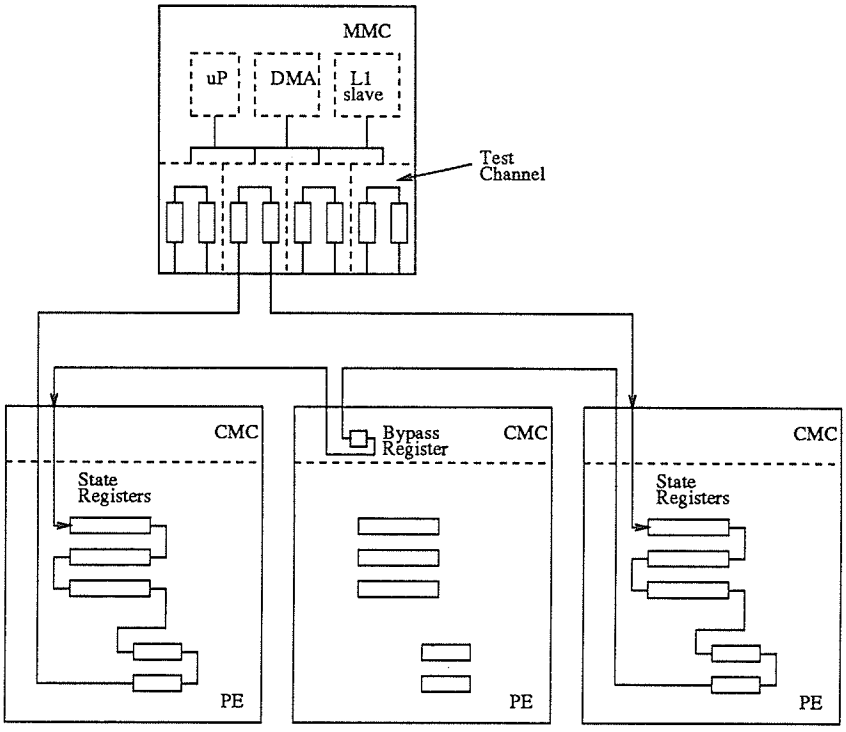

Figure 3.4: Initialization of processor elements is accomplished by transferring state variables from the spare to the active PE over the test bus. The state variables are accessed as a single serial register in each PE. Intervening PE's are bypassed. The discussion in the text assumes the active PE **A** is on the left, and the spare PE **B** is on the right.

2. The procedure is slightly more complicated when the spare and active PE are connected to separate test channels controlled by the same MMC. Either the microprocessor or the DMA controller is used to transfer data from one serial test bus to the

other.

Assuming the test channels and the MMC operate at the same clock rate, the time required to transfer state data from A to B using DMA is the same as for the previous case, provided $m > r + s$, where m is the size of the test channel transmit and receive buffers in bits, r is the number of clock cycles for each transfer using DMA, and s is the number of clock cycles needed to set up the DMA controller. For the design described in [22] using commercial microprocessor peripherals, the test channel buffers contain 16 bits, 4 clock cycles are needed for each transfer, and the setup time is 1 clock cycle.

Conventional (CISC) processors require some form of high-speed data transfer such as DMA to satisfy this condition, although a RISC processor design may not.

3. The most involved situation occurs when the spare and the active PE's are on test channels controlled by different MMC's. Since the MMC's are connected only by the L1 bus through the SuMP, an extra layer of the test controller hierarchy must be traversed to transfer the state data. State data is transferred between the test channel buffers and the L1 bus slave interface in the MMC using the microprocessor or DMA. Since the MMC's are only slaves on the L1 test bus, the SuMP transfers each packet of data between MMC's. Message traffic on the L1 bus will affect the effective data transfer rate.

3.4 Implementation Considerations

A number of semiconductor processing and repair techniques have been developed to accommodate defects in the finished wafer. **Hard configuration** is an irreversible process

applied to the wafer in the final stages of manufacture to bypass faulty units, using discretionary wires or fusible links. **Soft configuration** uses programmable switches to bypass the defective units prior to use. Switches are externally controlled and may or may not be volatile. **Dynamic configuration** includes all techniques applied to the circuit while it is in operation, and are of special interest, since they can cope with permanent, intermittent and transient faults.

Hard configuration is needed to ensure the operation of the test bus and bypass switches only. Since this hardcore circuitry consists of simple buffers and wires, the probability of failure can be significantly reduced by using conservative design rules and large wire widths. Laser direct-write techniques are useful to repair defects and configure redundant hardware as an end-of-production procedure. Laser pantography[33] can be used to fabricate transistors and wires on an essentially bare wafer, so that it is conceivable to consider constructing the test busses directly on the wafer after the PE's. This process is relatively slow as yet, and will likely be useful only for repairs until the technology matures. In general, hard configuration techniques are costly and can introduce additional problems through contamination, metallization defects, and changes to device characteristics such as the gate threshold voltage.

A considerable amount of research has been done into the basic problem of constructing arrays on wafers containing faulty cells. Leighton and Lieserson[32] present several algorithms for linear and 2-D arrays. The *divide-and-conquer* algorithm works very well on average, with a high probability of connecting all healthy cells on a N-cell array. Redundancy links between PE's in the order of $O(\log \log N)$ are required in the interconnection network to bypass faulty cells.[3] For example, an 8 by 8 processor array would require an

[3] $O(f(N))$ denotes a function bounded above by $cf(N)$ for a constant c and all sufficiently large N.

interconnection network with triple redundancy to provide a high probability of reconfiguration for an arbitrary distribution of faults.

Chapter 4

Summary and Conclusions

Boundary scan is a powerful and versatile DFT technique. The cost to incorporate boundary scan and associated control logic is reasonable for most current VLSI applications, and will diminish to an insignificant level as integrated circuit fabrication technology continues to improve. Cellular automata are ideal for pseudorandom pattern generation with boundary scan, and can be integrated into the boundary scan register with a small incremental cost.

Boundary scan with built-in self-test is an essential component of fault tolerance and testing techniques for WSI processor arrays and large digital systems. A hierarchical test control system is necessary to integrate testing resources in large digital systems. A simple yet complete test strategy for end-of-production and operational testing can be devised with these resources. Similarly, concurrent fault detection and isolation schemes and dynamic system reconfiguration are possible.

In conclusion, the objectives identified at the outset of this thesis have been achieved.

4.1 Future Work

There are several areas where more work is required. Many of the items mentioned here are suitable as undergraduate-level topics for course work and thesis projects.

Additional components, such as cells for interior scan path, LSSD and BILBO techniques, should be developed to extend the versatility of the test access port as an on-chip test controller.

In anticipation of the trend towards hardware description languages for design entry, a VHDL implementation of the boundary scan component library is needed. An interesting related problem is design of the boundary scan register automatically from the table of CA construction rules.

Software tools are needed to convert conventional test vector files to a format suitable for use with scan techniques and boundary scan in particular.

The operation of the test control system for fault tolerance and testing in WSI arrays requires further development and analysis. The effect of redundancy in the interconnection network on the performance of concurrent techniques needs to be explored, possibly through simulation of appropriate models.

A quantitative assessment of the improvement to system reliability with these techniques is another challenging and interesting problem that has not been addressed here.

Bibliography

[1] Williams, Thomas W. and Kenneth P. Parker, **Design for Testability - A Survey,** *Proc. IEEE*, Vol. 71, No. 1, 1983, pp. 98-112.

[2] Dahbura, Anton T., M. Ümit Uyar and Chi W. Yau, **An Optimal Test Sequence for the JTAG/IEEE P1149.1 Boundary-Scan Controller,** AT&T Bell Laboratories, Murray Hill, New Jersey, 1989.

[3] Abraham, Jacob A. and Vinod K. Agarwal, **Test Generation for Digital Systems,** in *Fault Tolerant Computing - Theory and Techniques,* D.K. Pradhan (ed.), Prentice Hall, Englewood Cliffs, New Jersey, 1986.

[4] McKluskey, E. J., **Design for Testability,** in *Fault Tolerant Computing - Theory and Techniques,* D.K. Pradhan (ed.), Prentice Hall, Englewood Cliffs, New Jersey, 1986.

[5] Fujiwara, Hideo, **Logic Testing and Design for Testability,** *The MIT Press,* Cambridge, Massachusetts, 1985.

[6] Negrini, R., M.G. Sami and R. Stefanelli, **Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays,** *The MIT Press,* Cambridge, Massachusetts, 1989.

[7] Johnson, Barry W., **Design and Analysis of Fault Tolerant Digital Systems,** Addison Wesley, Reading, Massachusetts, 1989.

[8] Bennets, R.G, **Design of Testable Logic Circuits,** Addison Wesley, Reading, Massachusetts, 1984.

[9] Bateson, John, **In-circuit Testing** Van Nostrand Reinhold Company, New York, 1985.

[10] **Testability Bus Specification P1149/D7,** Institute of Electrical and Electronics Engineers, Inc., New York, 1988. 1989.

[11] **Standard Test Access Port and Boundary-Scan Architecture P1149.1/D5,** *IEEE Computer Society Press,* Washinton, D.C., 1989.

[12] Miller, D.M. and S. Zhang, **Aliasing in Multiple-Input Data Compactors,** *Proc. 1989 Canadian Conference on Electrical and Computer Engineering,* pp. 347-51.

[13] Zhang, S. and D.M. Miller, **The Determination of Minimal Cost Linear One-Dimensional Cellular Automata,** Submitted to *Electronics Letters* (1990), awaiting publication.

[14] Gloster, Clay S. and Franc Brglez, **Boundary Scan with Cellular-Based Built-In Self Test,** *Proc. 1988 IEEE International Test Conference,* Paper 7.2.

[15] Parker, Kenneth P., **The Impact of Boundary Scan on Board Test,** *IEEE Design and Test of Computers,* August 1989, pp. 18-30.

[16] Dervisoglu, Bulent I., **Scan Path Architecture for Pseudorandom Testing,** *IEEE Design and Test of Computers,* August 1989, pp. 32-48.

[17] Canadian Microelectronics Corporation, **Guide to the Integrated Circuit Implementation Services of the Canadian Microelectronics Corporation (GICIS Version 4.0),** Queen's University, Kingston, Canada, March, 1989.

[18] Hortensius, Peter D., **Parallel Computation of Non-Deterministic Algorithms in VLSI,** Ph.D. thesis, University of Manitoba, 1987.

[19] Weste, Neil and Kamran Eshraghian, **Principles of CMOS VLSI Design,** Addison Wesley, Reading Massachusetts, 1985.

[20] Hortensius, Peter D., R.D. McLeod and H.C. Card, **Parallel Random Number Generation for VLSI Systems Using Cellular Automata,** *IEEE Transactions on Computers,* Vol. 38, No. 10, October, 1989.

[21] Budde, W.O., **Modular Testprocessor for VLSI Chips and High-Density PC Boards,** *IEEE Transactions on Computer-Aided Design,* Vol. 7, No. 10, October 1988, pp. 1118-24.

[22] Breuer, M.A. and J.C. Lien, **A Methodology for the Design of Hierarchically Testable and Maintainable Digital Systems,** *Proc. 8th Digital Avionics Systems Conf.,* 1988, pp. 40-47.

[23] Breuer, M.A. and J.C. Lien, **A Universal Test and Maintenance Controller for Modules and Boards,** *IEEE Transactions on Computers,* Vol. 36, No. 2, May, 1989, pp. 231-40.

[24] Breuer, M.A. and J.C. Lien, **A Test and Maintenance Controller for a Module Containing Testable Chips,** *Proc. 1988 International Test Conference,* Paper 27.1, pp. 502-13.

[25] Haedtke, J.E. and W.R. Olsen, **Multilevel Self-Test for the Factory and Field,** *Proc. 1987 Annual Reliability and Maintainability Symposium,* pp. 274-79.

[26] Breuer, Melvin A, Rajiv Gupta and Jung-Cheun Lien, **Concurrent Control of Multiple BIT Structures,** *Proc. 1988 International Test Conference,* Paper 23.2, pp. 431-42.

[27] Shombert, Lee A. and Daniel P. Siewiorek, **Using Redundancy for Concurrent Testing and Repairing of Systolic Arrays,** *Proc. 1987 IEEE Fault Tolerant Computing Symposium,* pp. 244-9.

[28] Friedman, Arthur A. and Luca Simoncini, **System-level Fault Diagnosis,** *IEEE Computer,* March 1980, pp. 47-53.

[29] Westmore, R.J., **A Scaleable Waferscale Architecture for Real 3-D Image Generation,** *Proc. 1989 IEEE International Conference on Wafer Scale Integration,* pp. 95-110.

[30] Stewart, A.K.J., **The Development of a Fault-Tolerant ULSI Signal Processor,** *Proc. 1989 IEEE International Conference on Wafer Scale Integration,* pp. 245-55.

[31] Saucier, Gabrièle and Jacques Trilhe, **Wafer Scale Integration,** *Proc. 1986 IFIP WG 10.5 Workshop on Wafer Scale Integration.*

[32] Leighton, F.T. and C.E. Lieserson, **Wafer-scale Integration of Systolic Arrays,** *IEEE Transactions on Computers,* Vol. C-34, No. 5, May 1985, pp. 448-61.

[33] Burns, L.L. **Laser Pantography,** *Proc. 1986 IFIP WG 10.5 Workshop on Wafer Scale Integration,* pp. 281-90.

[34] IBM, Honeywell, and TRW, **VHSIC phase 2 INTEROPERABILITY STANDARDS, ETM-BUS Specifications,** December, 1986.

Appendix A

The TAP Component Library

The components needed to include boundary scan in an application circuit are available in schematic form from within the CĀDENCE design environment. The system administrator should be consulted to determine the pathname to the component library. Table A.1 lists the basic components needed. **TAP** and **bist_reg** are required for all designs; **uofmid_reg** is the optional identification register. Any component in the library may be modified by first copying it to a working directory. The library also includes a complete design example for a 4-bit ALU. This design has been submitted for fabrication as **MB085**. Table 2.2 should be used when placing the boundary scan register cells to ensure a maximal length sequence is obtained.

Table A.2 summarizes the instruction set. Undefined instructions select the bypass register and cause all testing features to remain inactive. **inst_decode** can be modified to change the function of these instructions or to define new ones. Eight instructions are available since **inst_reg** is a three bit register. This component can be modified to add more instructions.

The **uofmid** component is a customized identification register which corresponds to the

Component	Description
bsc_i150	Rule 150 input boundary scan cell
bsc_i90	Rule 90 input boundary scan cell
bsc_o150	Rule 150 output boundary scan cell
bsc_o90	Rule 90 output boundary scan cell
TAP	Test Access Port
uofmid_reg	U of M identification register
bist_reg	CA-BIST control register

Table A.1: Boundary Scan Component Library

Instruction	Code	Register Selected	Test Logic Response
EXTEST	000	Boundary scan	Set/capture logic values on all circuit interconnects
SAMPLE	001	Boundary scan	Capture logic values on all pins during normal operation
UOFMID	010	CMC ID register	Scan out CMC-format ID code
RUNBIST	011	Boundary scan and test data registers	Execute self-contained self-test of component
INTEST	100	Boundary scan	Apply scanned-in test vector and capture response
BYPASS	111	Bypass	Pass data from TDI to TDO

Table A.2: Instruction Summary

format of the Canadian Microelectronics Corporation design reference name. This designator consists of a two-character institution code and a three-character design label. The format for the UOFMID register is "MBxxx", where "xxx" is an arbitrary yet unique combination of three numeric characters chosen by the designer[1]. The register has three 4-bit binary coded decimal inputs which are programmed to the design identification number in the design. The UOFMID instruction selects this register to be connected to TDI and TDO. The IDCODE instruction defined by the standard is not implemented.

[1]"MB" is the unique identifier for all submissions originating at the University of Manitoba.

Appendix B

Glossary of Terms

BILBO	Built-in Logic Block Observer
BIST	Built-in Self Test
CAD	Computer-Aided Design
CISC	Complex Instruction Set Computer
CMC	Component Test and Maintenance Controller
CMOS	Complementary Metal Oxide Silicon
DFT	Design for Testability
DMA	Direct Memory Access
ETM	Element Test and Maintenance (VHSIC bus standard)
HTM	Hierarchical Test and Maintenance system
IC	Integrated Circuit
JTAG	Joint Test Action Group
LFSR	Linear Feedback Shift Register
LSSD	Level Sensitive Scan Design
MMC	Module Test and Maintenance Controller
MSDS	Minimum Serial Digital System (IEEE TAP bus standard)
PE	Processing Element
PRPG	Pseudorandom Pattern Generator
RISC	Reduced Instruction Set Computer
SMP	System Test and Maintenance Processor
SuMP	Subsystem Test and Maintenance Processor
TAP	Test Access Port
TMR	Triple Modular Redundancy
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
WSI	Wafer Scale Integration

Boundary Scan Control Signals

BISTclk	BIST clock signal
BISTmode	BIST mode control (active HIGH)
captureDR	Boundary scan *read* control signal (clock pulse)
clockDR	Boundary scan *shift* clock
imode	Input boundary scan multiplexer control signal
omode	Output boundary scan multiplexer control signal
shiftDR	Boundary scan multiplexer control signal
updateDR	Boundary scan *write* control signal

Boundary Scan Instructions

BYPASS	Pass data through component
EXTEST	Test interconnections between components
IDCODE	Access programmed identification register
INTEST	Apply scanned-in test data
SAMPLE	Sample data on all boundary scan cells
RUNBIST	Perform built-in self-test
USERCODE	Access user-programmed identification register
UFOMID	Access custom identification register

MSDS Test Bus Signals

TCK	Test clock
TDI	Test Data Input
TDO	Test Data Output
TMS	Test Mode Select
TRST	Test Logic Reset (optional)