A TEST CONTROLLER BOARD FOR TSS

by

Kevin T. Kornegay

Memorandum No. UCB/ERL M91/4

16 January 1991

A TEST CONTROLLER BOARD FOR TSS

by

Kevin T. Kornegay

Memorandum No. UCB/ERL M91/4

16 January 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A TEST CONTROLLER BOARD FOR TSS

by

Kevin T. Kornegay

Memorandum No. UCB/ERL M91/4

16 January 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Testing, in general, is a method of exercising a digital system and examining the response to ascertain whether it behaves correctly. If an incorrect response is observed, a second goal of testing is to diagnose why the system behaved incorrectly. Testing is becoming an important part of the design process since designers have become more concerned with reducing design cycle times and cost. As technologies become more advanced, system designers are faced with the challenging testing problems shown in Figure 1.1. Although advances in fabrication, packaging, and surface mount technologies have paved the way for faster, denser, and more complex digital systems, they have made it more difficult to test. For example, using surface mount interconnection technology or multi-chip modules at the board level limits access to internal connections which consequently causes technical difficulties in probing smaller, more closely packed contact points. At the integrated circuit (IC) level, the difficulties of complexity and restricted access have been recognized for some time and have led to the development of structured design techniques for testing such as Scan-Path [1], Built-In-Self-Test [3, 9] (BIST), and the JTAG Boundary-Scan [7, 8] standard to alleviate these difficulties.

A rapid-prototyping computer-aided-design (CAD) system is currently under development, called System Integration For Embedded Real-Time Applications (SIERA), that manages the synthesis of a real-time system. The goal of this project is to provide test support for SIERA at all levels of the system design hierarchy. This support comes in the form of software and hardware that performs the duties of automatic test hardware

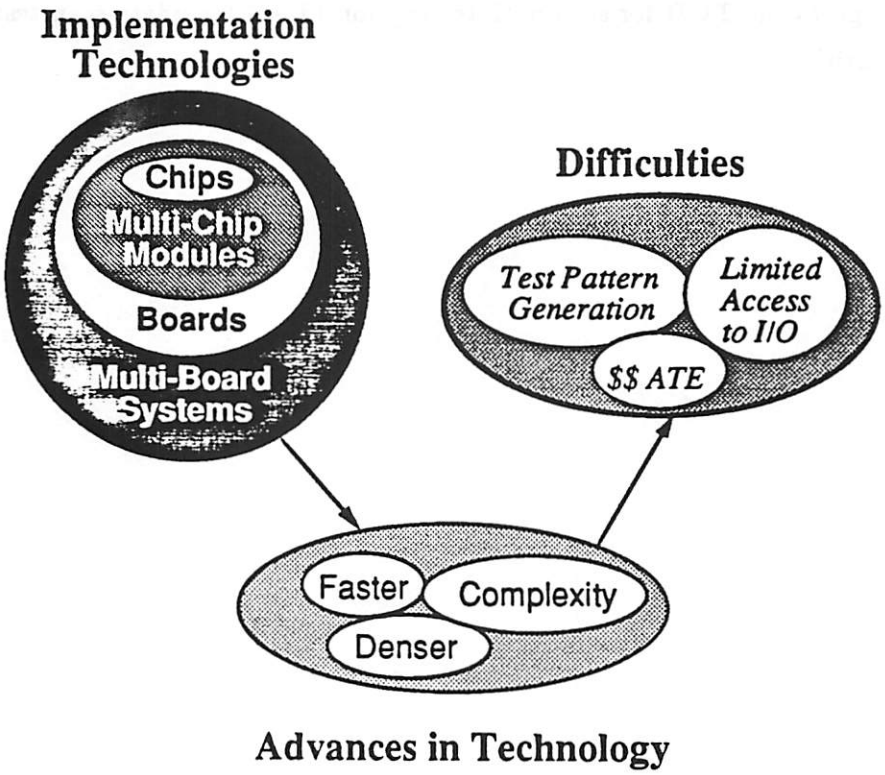


Figure 1.1: Problems in testing.

generation, test pattern generation, test management and diagnosis. To accomplish this, a sub-system of **SIERA**, called the Test Support System (**TSS**), is being developed that will provide the test hardware and software support for **SIERA**. This report describes the features, design, architecture, and implementation of the Test Controller Board (**TCB**), which is used for test hardware support at the system-level, the board-level, and the chip-level.

The **TCB**'s primary functions are to apply deterministic test vectors, which are provided by the designer, to the device under test (**DUT**), to configure the **DUT** for a test by providing the appropriate control signals, to execute a test, to acquire and to store the test response for comparison with a good unit. The **TCB** contains an asynchronous and a synchronous interface for communication with the host and **DUT**, a test data memory for storing the input and output test data, and a controller which implements the scan-path and boundary-scan bus master protocols. The software for **TCB**, called SCANTEST, is written in the C-programming language and is executed from the VXWORKS software environment. The SCANTEST software allows us to control the **TCB** from a **SUN** workstation. It

configures the **TCB** for scan-path testing, for **JTAG** boundary-scan testing, and for data acquisition.

Chapter 2

Test Strategies

2.1 Overview

For a system to have a high degree of testability, the system must be testable at every level of integration. In order to achieve a high degree of testability, we decided to partition a system into four levels, (1) chips, (2) multi-chip modules, (3) boards, and (4) multi-board systems, where each level utilizes a specific structured design-for-test (DFT) technique. Two important attributes related to DFT are controllability and observability. Controllability is the ability to establish a specific signal value at each node in a circuit by setting values on the circuit's inputs. Observability is the ability to determine the signal value at any node in a circuit by controlling the circuit's inputs and observing its outputs. There are two major advantages to using structured DFT techniques, first they are easy to incorporate into our existing design environment, and second, they simplify test generation. By using a scan-path register, a sequential circuit can be treated like a combinational circuit and standard test generation algorithms such as PODEM [2] can can be used.

Figure 2.1 illustrates how these techniques may be used in a board design. The Scan-Path technique discussed in Section 2.2 is employed at the chip-level and is used to test combinational and sequential logic. The BIST discussed in Section 2.3 is also employed at the chip-level and is used to test RAMs, ROMs, and other embedded memory structures structures. Boundary-Scan which supports board-level testing will be discussed in Section 2.4. A dedicated chip-set, called the Boundary-Scan Buffer Chip-Set (BSBC), is used for testing of non-boundary-scan components at the board-level. Finally, the TCB which is used for testing multi-board systems, is employed at the system-level.

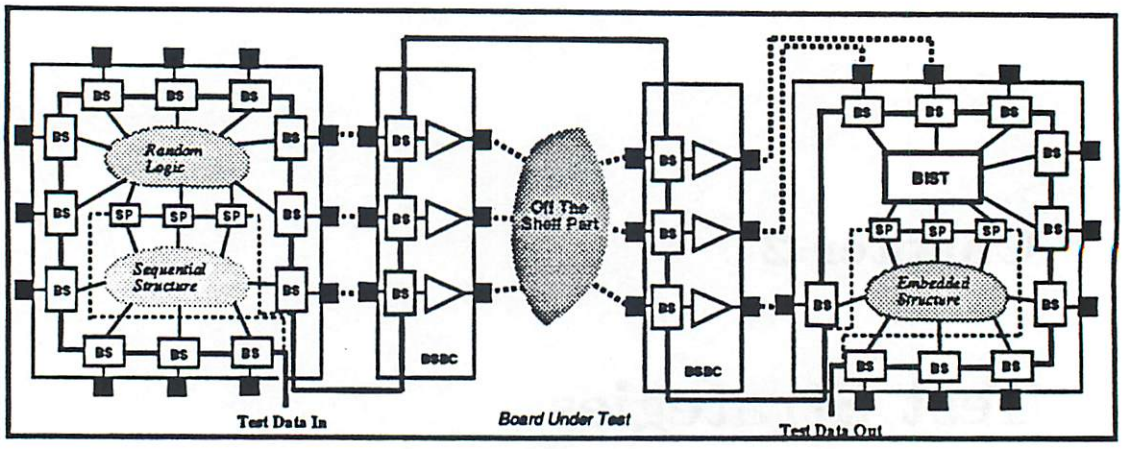


Figure 2.1: Test strategies implemented at the chip and board levels.

2.2 Scan-Path

Scan-Path is probably the most widely used **DFT** technique. It is based on converting the system memory elements, *D*-type flip-flops or latches, into a scan-path register. With the circuit in test mode, it is possible to shift an arbitrary test pattern into the scan-path register. By returning the circuit to normal mode for one clock period, the contents of the scan register and primary input signals act as inputs to the combinational circuitry, and new values are stored in the scan-path register. If the circuit is then placed into test mode, it is possible to shift out the contents of the scan register for comparison with the correct response. It is assumed that the circuit is constructed of *D*-type flip-flops interconnected by combinational circuits and that all the flip-flops are controlled by the system clock.

2.3 BIST

The ultimate in testable design is to make the design test itself. **BIST** is a structured **DFT** technique that accomplishes this by adding test hardware to circuits so that they can activate testing. A **BIST** test requires a mechanism for supplying test patterns to the **DUT** and a means for comparing the unit responses to the known good responses as suggested in Figure 2.2, and both mechanism and means must be compact enough to reasonably be built into the **DUT**. Building test into the design, as might be expected, consumes added circuit I/O overhead, but at the same time results in visible reductions to

the costs of testing when compared with an external test using automatic test equipment. BIST testing achieves these savings by

1. eliminating the costs of test pattern generation and fault simulation,
2. shortening the time duration of tests (by running tests at circuit speeds),
3. simplifying the external test equipment, and
4. easily adopting to design changes.

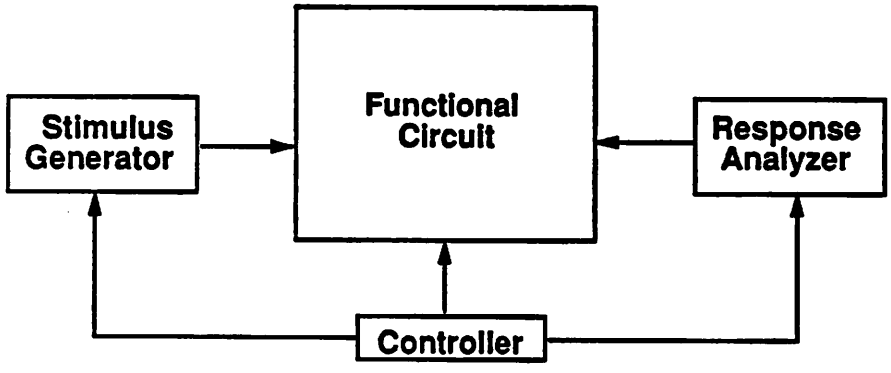


Figure 2.2: General form of a BIST structure.

2.4 Boundary-Scan

To better address problems of board-level testing, several DFT standards have been developed. The primary goal of these proposed standards is to ensure that chips of very-large-scale-integration (VLSI) complexity contain a common denominator of DFT circuitry that will make the test development and testing of boards containing these chips significantly more effective and less costly. One of these standards is the IEEE JTAG Testability Bus Standard [8]. This standard deals primarily with the use of a test bus which will reside on a board, the protocol associated with this bus, I/O ports that tie a chip to the bus, and some control logic that must reside on a chip to interface the test bus ports to the DFT hardware residing on the application portion of the chip. The primary reason for using boundary-scan are to allow for efficient testing of board interconnect and to facilitate isolation and testing of chips either via the test bus or by BIST hardware.

With boundary-scan, chip level tests can be reused at the board-level. of logic required to support JTAG. This circuitry may include BIST or scan-path hardware. If so, the scan paths are connected via the test-bus circuitry to the chip's scan-in and scan-out ports. The normal I/O terminals of the application logic are connected hardware.

There are two major components associated with JTAG, namely (1) boundary-scan registers and (2) the test access port (TAP) controller which is a finite-state machine. If so, the scan paths are connected via the test-bus circuitry to the chip's scan-in and

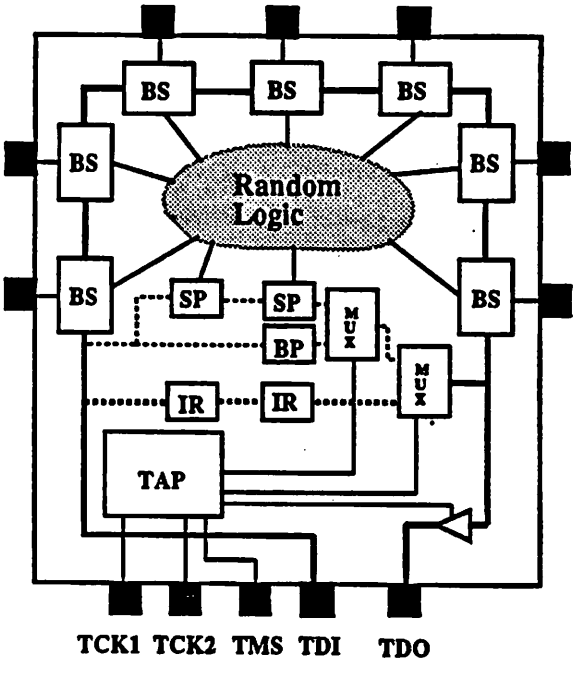


Figure 2.3: Chip architecture for JTAG Standard.

scan-out ports. The normal I/O terminals of the application logic are connected through boundary-scan cells to the chips I/O pads. The rest of the test circuitry consists of a 1-bit bypass register, an instruction register, and the TAP. The TAP bus consists of four lines, namely a test clock *TCK*, a test mode select signal *TMS*, the test data in *TDI* and the test data out *TDO* line. Figure 2.3 also illustrates how the boundary-scan cells are interconnected into a single scan-path, where the *TDO* of one chip is tied to the *TDI* of another chip. Using this configuration various tests can be carried out, including (1) interconnect test, (2) snapshot observation of normal system data, and (3) test of each chip. To implement these tests, three test modes exist, namely internal test, external test, and

sample test. While in the internal test mode, the internal scan-paths and BIST operations can be activated to test a chip. During this test mode, the inputs to the application logic are driven by the input boundary-scan register cells, and the response can be captured in the output boundary-scan cells. By using appropriate boundary-scan register cells and test data in the external test mode, interconnect tests can be carried out for tri-state logic and bi-directional pads. In the sample test mode, I/O data associated with a chip can be sampled during normal system operation.

Chapter 3

Test Support System (TSS) Hardware

3.1 Overview

The TSS hardware is comprised of scan-path register cells, boundary-scan register cells, LFSR cells, and the TCB. A block diagram of the TSS hardware and how it is configured in a system development environment is shown in Figure 3.1. in this environment, the TCB, the CPU, and the DUT, which may contain scan-path register cells, boundary-scan register cells, and BSBC chips, reside in a VME card cage which communicates with a SUN workstation over the ethernet.

In order to minimize the overall cost of implementing the test hardware, a set of design objectives were defined. These objectives are listed as follows:

1. test hardware must be implemented with very little overhead,
2. it must be easy to incorporate into the system design hierarchy,
3. it must support synchronous designs, and
4. it must not severely degrade the system performance.

Furthermore, by imposing the rules given below, the structured DFT techniques described in Chapter 2 can be easily incorporated into a design.

- All registers are scanable.

- All chip inputs and outputs are scanable.
- Use Boundary-Scan Buffer Chips to enhance testing of off-the-shelf parts.

The specifics of the TSS hardware used at the system-level, board-level, and chip-level are discussed in Section 3.2, Section 3.3, and Section 3.4 respectively.

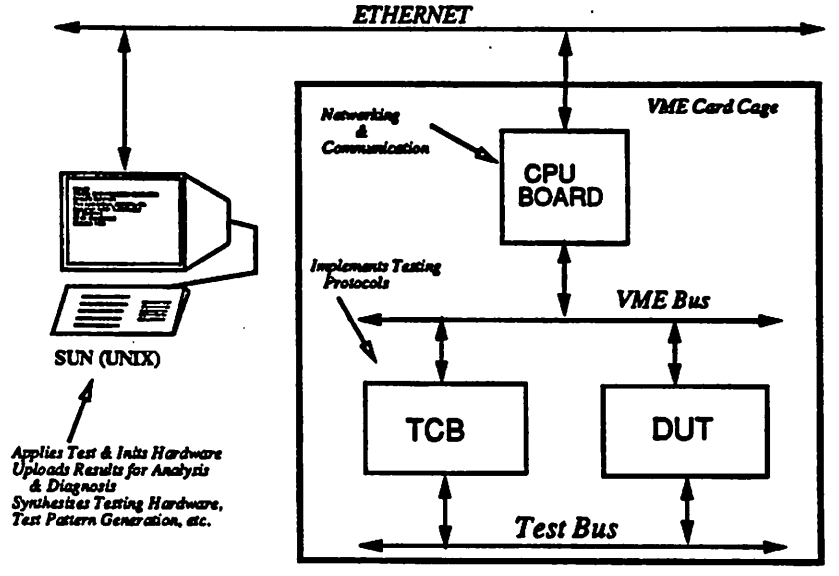


Figure 3.1: Using TSS hardware in a development environment.

3.2 Multi-Board System Level Test Hardware

In designing multi-board systems, it is often useful for purposes of testing to be able to isolate one printed-circuit board (PCB) from the other. This can be done using the TCB which implements the boundary-scan bus master protocol [8]. Assuming all the ICs on a PCB are designed using the boundary-scan architecture, we can exercise the internal scan-path, BIST hardware, and the interconnect of every PCB in the system. The details and features of the TCB will be discussed in the following subsections.

3.2.1 TCB Design And Specifics

The TCB was designed using the DMoct [5] tool. This tool manages the design of an application-specific IC or an application-specific board. A netlist representing the

structure of the TCB is written to a textual file in structure description language (sdl) format (See Appendix A). DMoct parses the sdl file and stores the information contained in the file in the OCT [4] database. We used the Racal-Redac PCB tool, a commercial board design package, after verifying the functionality of the design using the Thor behavioral simulator. The Racal-Redac PCB tool was used to place and route the TCB components. Before using this tool, the information stored in the OCT database was used to generate a rinf file by the Oct2rinf CAD tool. The rinf file was loaded into PCB tool and each part was manually placed.

After the design was placed, we used the router tool to route the 8-layer TCB. Once the routing was completed, Gerber files were generated and films were produced using these files. We shipped the films off to a PCB fabrication facility after checking them for obvious shorts. After the board returned from the fabrication facility, the IC sockets were inserted and wave soldered. The TCB was fabricated on a 6U (double-height) board and resides in a 21 slot VME card cage. It functions as a slave module and is controlled by the Heurikon CPU board which handles all the communication and arbitration on the VMEbus. The TCB occupies addresses 01FC0000 thru 01FFFFFF of the VMEbus standard address space. In order to reduce the local noise on the TCB, bypass capacitors were inserted between the power and ground pins of each IC. It contains 48 IC's and 9 connectors which are listed below:

- 40 TTL components (tri-state buffers, registers, ands, ors),
- 1 VME 2000 interface controller chip,
- 3 Electrically Programmable Logic Devices (2 Altera EPLD 1810's, 1 EP910),
- 2 Cypress 256K x 1 Static Rams,
- 1 10 MHz crystal,
- 1 50ns delay-line,
- 2 VME 96-pin connectors,
- 4 14-pin right angle connectors,
- 1 10-pin right angle connector,
- 2 2-pin right angle power connectors (for testing Analog parts).

3.2.2 TCB Architecture

A photograph, block diagram, and schematics of the The TCB are shown in Appendix B, Figure 3.2, and Appendix D respectively. It is comprised of the VME interface, which connects it to the J1 and J2 backplanes, the VME interface logic (in the form of a single chip VME 2000), which implements protocols that meet the VMEbus specification IEEE 1014 timing requirements, the control, status, and data registers whose sizes and functional meaning are shown below in Table 3.1, the test controller, test data memory (SCANIN and SCANOUT memories), the address counter, which generates the addresses for the test data memory, and the analog, scan-path, and TAP[7] interfaces, which connect the TCB to the DUT's. The VME standard address spaces occupied by the control

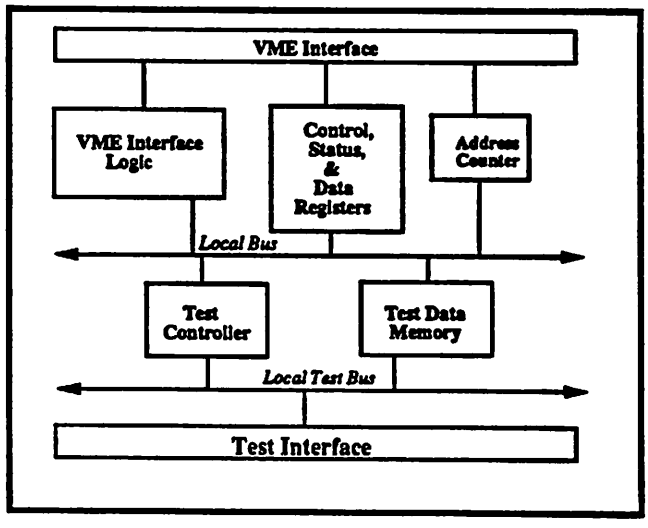


Figure 3.2: TCB architecture.

register, the status register, and the test data memory are shown above in Table 3.2. The TCB to VME connector pin mappings are given below in Table 3.3. The state transition diagram of the test controller which implements the JTAG boundary-scan bus master protocol and a scan-path protocol is shown in Figure 3.3. After initializing the appropriate set of registers, enabling the *BEGIN TEST* signal will initiate the operation of the controller. Depending on the *SCANP* signal, the controller follows one of the major branches as shown in Figure 3.3. When the *SCANP TEST* signal is a logic 0, the boundary-scan branch is selected.The Data Scan or the Instruction Scan branches are accessible from this branch. The Data Scan branch is selected when the *BSDS TEST* signal is a logic one. While in this

branch, the controller loads the address counter, which decrements on every clock period, with a starting address, toggles the Test Mode Select (*TMS*) signal, reads the test data memory, polls for the *CARRY* to indicate the end of a scanning operation, runs the DUT for one clock period, and writes the results into the test data memory. The Instruction Scan branch is followed when transmitting instructions to the DUT. In this branch, the operation of the TCB is similar to that for Data Scan operations. When the Scan Path branch is chosen, the TCB loads the address counter with a starting address, stuffs the DUT's local scan-path with test data, enables the *SCANTEST* signal to configure the DUT for testing, polls for a *CARRY* signal, executes the DUT for one clock period, reloads the address counter with the same starting address, and writes the contents of the scan-path register into the test data memory until the *CARRY* signal is detected.

The test controller was synthesized from its boolean equation representation in bds format (See Appendix A). The controller is housed in an Altera EP910 EPLD and was programmed using the A+PLUS software package on an IBM PC. The controller configures the TCB for a particular operation based on the control words given in above Table 3.4. The address counter which generates the addresses for the test data memory was implemented on an Altera EP1810 EPLD using the same software package. A second EP1810 was used to generate the module select and local decode signals. The pinouts of all of the EPLD's are shown in Appendix C.

3.2.3 Functional Testing Of The TCB

The board was tested in a top-down manner with the aid of a Tektronix DAS 9100 Series Digital Analysis system. We started testing from the VME interface and worked our

TCB Signal Mnemonic	Functional Meaning
Control_reg[0]	Undefined
Control_reg[1]	Resets TCB Controller
Control_reg[2]	Configres TCB Instruction/Data Scan Mode
Control_reg[3-4]	Selects appropriate *TMS* connector
Control_reg[5]	Sets SCANOUT Memory R/W Control
Control_reg[6]	Sets SCANIN Memory R/W Control
Control_reg[7]	Configures TCB for Boundary-Scan/Scan-Path Test
Status_reg[0]	Test completion signal
Status_reg[1-4]	TCB controller state

Table 3.1: Control and status register pin functions.

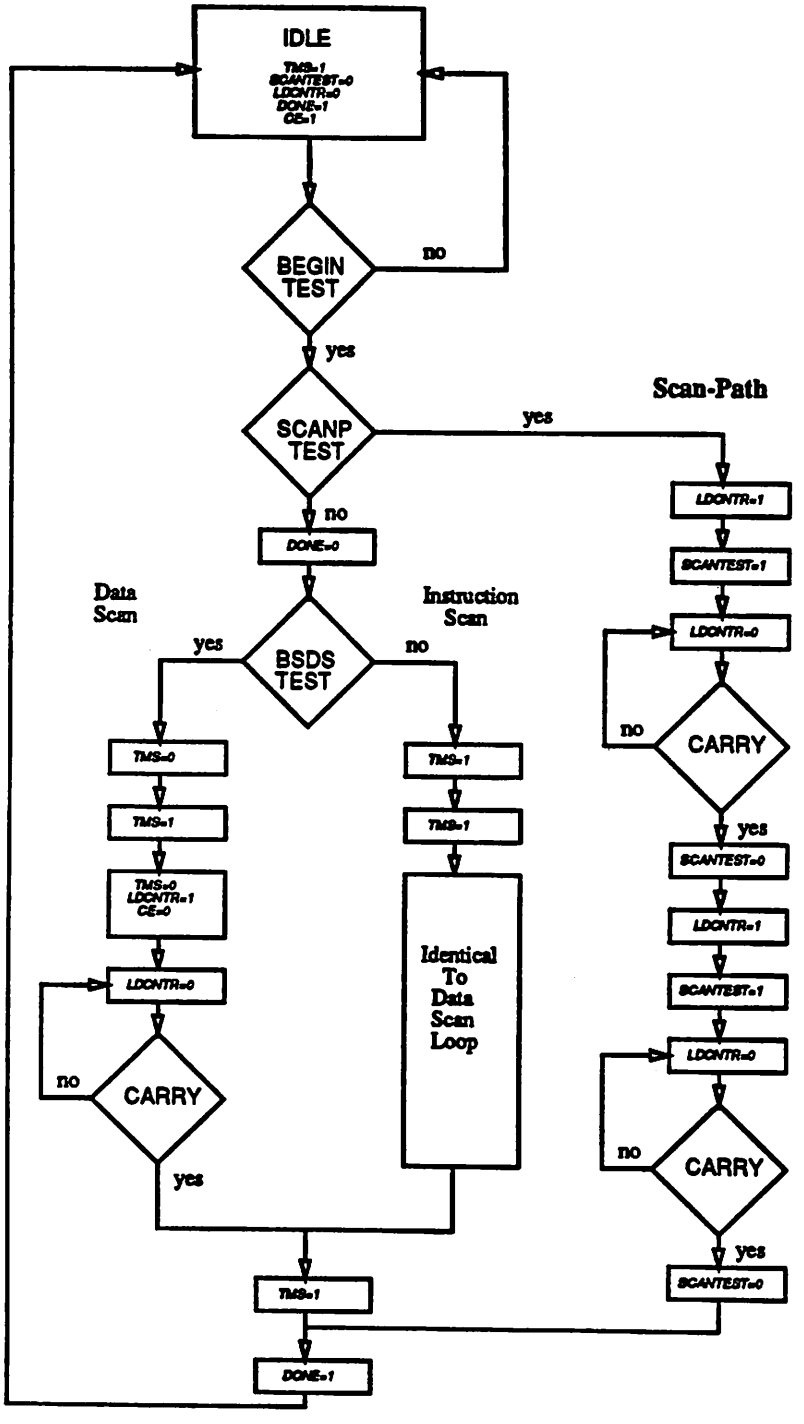


Figure 3.3: State transition diagram for TCB controller.

VME Address (Hexadecimal)	TCB Component
01FC0000-01FFFFEC	SCAN IN/OUT Data Memory
01FFFFF0	Control Register
01FFFFC0	Status Register

Table 3.2: VME standard address space occupied.

VME Signal Mnemonic	TCB Components
D0	SCANIN & SCANOUT Data Memorys
D1-D18	18-Bit Loadable Counter
D19-D26	8-Bit Control Register
D27-D31	5-Bit Status Register

Table 3.3: Control, data, and address registers pin mappings.

way to the **DUT** connectors. A VME extension card was used to allow easy access to the component pins of the **TCB**. Simple C programs were used to verify the integrity of the VME interface, that is, to determine whether on not we can read and write the **TCB**. Each time a component was added to the **TCB**, a new program was created to test the new module. This procedure was repeated until the **TCB** was completely functional.

3.3 Board-Level Test Hardware

In designing multi-chip modules (**MCM**) or printed circuit boards (**PCBs**), it is often useful for purposes of testing to be able to isolate one module from the others. This can be done using the concept of boundary-scan. If we assume all the chips and MCM's on a **PCB** are designed using the boundary-scan architecture described in Chapter 2, then all board interconnects can be tested. Through the use of a boundary-scan buffer chip set, we can enhance the testing of non-boundary-scan components. The **BSBC** set and the boundary-scan register cell are discussed in the following subsections.

3.3.1 Boundary-Scan Buffer Chip Set

The **BSBC** chip set shown in Figure 3.4 is used for testing of non-boundary-scan parts. This chip set will support a variety of bus structures and wires with large fan-outs. There are two advantages to having a **BSBC** set namely, (1) we can conserve precious **PCB** area by placing all the buffers on a single chip and (2) we can completely isolate the non-boundary-scan parts with strategic placement of these chips. Each chip of the

Control Word (Hexadecimal)	Functional Meaning
03000000	TCB Reset
07D00000	Run Scan Path Test
03100000	Run Boundary Data Scan Test
03300000	Run Boundary Instruction Scan Test
02C00000	Read Test Data Memory
01C00000	Write Test Data Memory

Table 3.4: **TCB control words.**

BSBC contains boundary-scan cells and non-inverting buffers. In addition to the normal data inputs and outputs, the chips have *TDI* and *TDO* pins for scanning test data, and control signals for tri-stating and signal direction. This chip set enhances testability at the board-level.

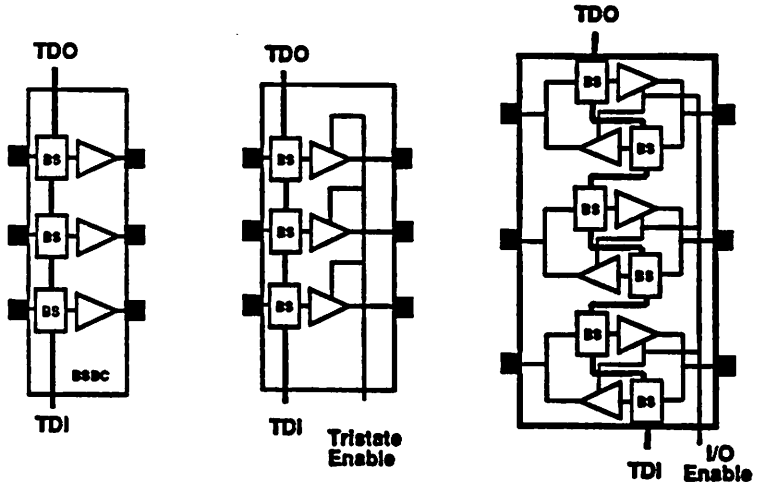


Figure 3.4: **BSBC chip set.**

3.3.2 Boundary-Scan Register Cell

The boundary-scan register allows testing of circuitry external to the IC package (primarily the board interconnect). It also permits the signals flowing through the system pins to be sampled and examined without impacting the operation ot the system logic. The boundary-scan register is a single shift-register-based path containing cells connected to all system inputs and outputs of the circuit and the system logic. Boundary-Scan register cells

must allow execution of at least three types of tests as determined by the control signals supplied to each cell. A schematic of an output boundary-scan cell is shown in Figure 3.5. During normal operation, the *MODE* signal is disabled and the cell becomes transparent

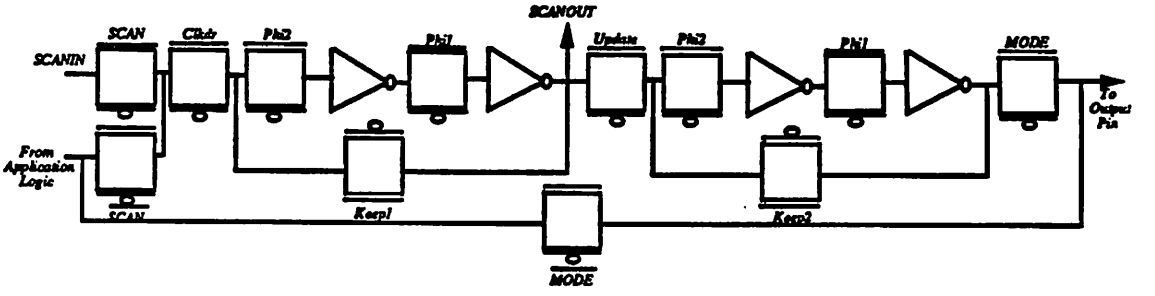


Figure 3.5: Boundary-Scan cell.

and data is passed directly to the output pin. When a test session is envoked, the *SCAN* and *Clkdr* signals are asserted until all the test data is loaded into the boundary-scan chain, after which, the *Update* signal is asserted and the test data is applied to the output pins. The test results are then captured at the input pin of an adjacent chip and shifted out for comparison with a good circuit while a new test vector is shifted in. The *Keep* signals are used to retain the state of the cell. The *MODE* signal is asserted during the entire test session. Two cells, namely the input boundary-scann cell and the output boundary- scan cell are required to implement the JTAG Boundary-Scan standard. Figure 3.6 demonstrates how these cells are used in relation to the system I/O pins. For example, the output boundary-scan cell is placed in the signal path before the output buffer. These cells support the three basic tests outlined in the JTAG standard and were designed using the Magic graphics editor in 2μ and 1.6μ CMOS technologies. The cells occupy an area of 181μ x 64μ and 145μ x 52μ for the 2μ and 1.6μ boundary-scan cells respectively. The size of the input and output cells are idendical but only differ in their orientation and topologies. A layout of an input boundary-scan cell is shown in Appendix E. The cost of implementing boundary-scan in a design is summarized in Table 3.5 below. Implementing boundary-scan in the 2μ CMOS technology increases the delay in the signal path by an additional 3 nSecs and increases the overall chip area by approximately 14%.

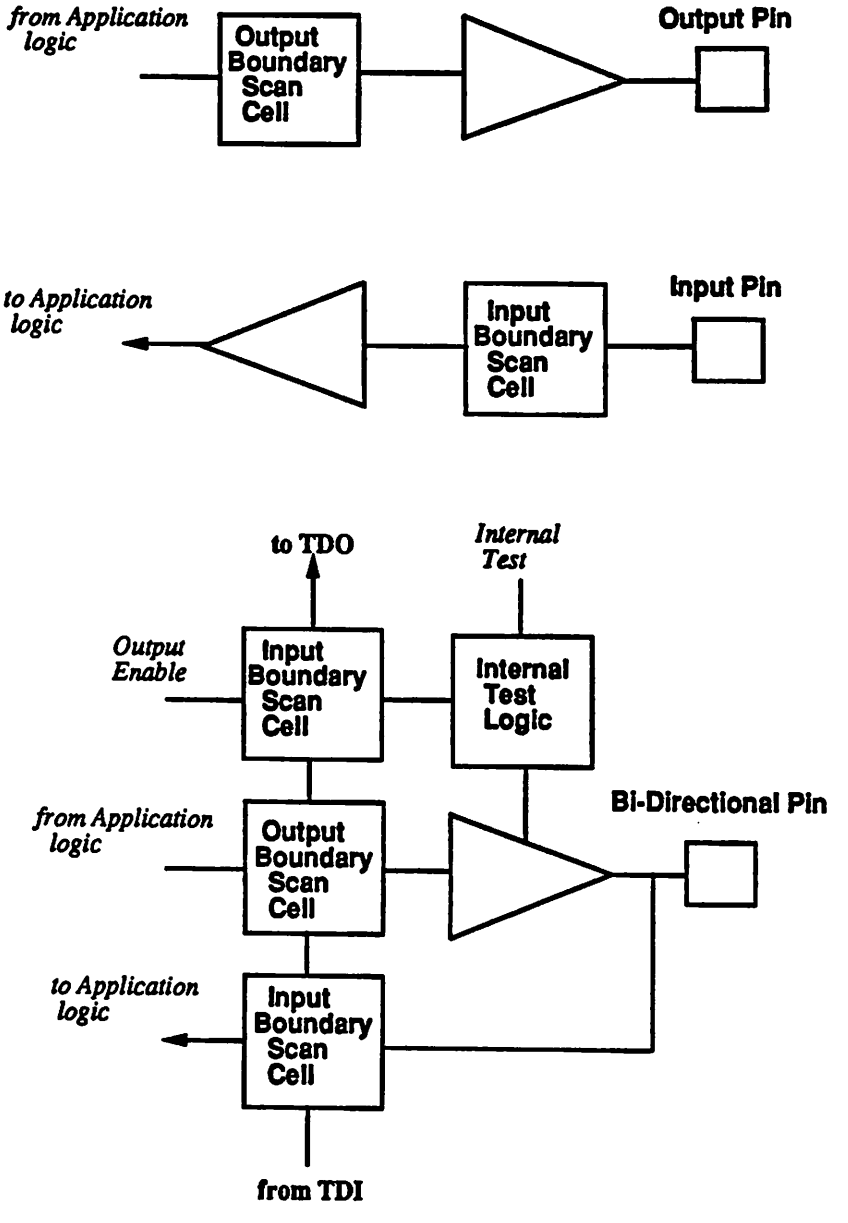


Figure 3.6: Pin configurations.

Technology	Chip Area (With/Without)	Performance (With)
2μ CMOS	12.5 mm x 12.5 mm/11 mm x 11 mm	+3 nSecs
1.6μ CMOS	10 mm x 10 mm/9 mm x 9 mm	+2 nSecs

Table 3.5: Area and performance penalties per technology.

3.4 Chip-Level Test Hardware

Most chip-level structured **DFT** techniques are built upon the concept that if the values in all the latches can be controlled to any specific value, and if they can be observed with a very straightforward operation then the test generation can be reduced to that of doing test generation for a combinational circuit. A control signal can switch the memory element from their normal mode of operation to a mode that makes them controllable and observable. The two structured **DFT** techniques used at the chip-level are scan-path and **BIST**. Our implementation of these **DFT** techniques will be discussed in the following subsections.

3.4.1 Scan-Path Register Cell

By using test points one can easily enhance the observability and controllability of a circuit. One way to accomplish this is by using a scan-path register. A scan-path register is a cascade of scan-path register cells that are connected to the internal logic of an **IC**. A schematic of our scan-path cell is shown in Figure 3.7. Operation of the cell requires two non-overlapping clocks namely *Phi1* and *Phi2*. During normal operation, the *LOAD* signal is asserted and the logic value at the *IN* input reaches the output (*OUT*) after one clock cycle. When the *SCAN* signal is asserted during a test, the logic value at the *SCANIN* input arrives at the *SCANOUT* output one clock cycle later. The *KEEP* signal is used to refresh the value stored in the cell. This cell was designed using the Magic graphics editor and was implemented in a 2μ CMOS technology. It occupies an area of 143μ x 52μ.

3.4.2 Memory Testing With BIST

Embedded memory structures such as RAMs and ROMs are not easily testable by techniques such as scan-path. These structures usually require deterministic test sequences to insure fault-free status for both classical and non-classical faults. **BIST** architectures at the chip-level consist of several key elements, namely

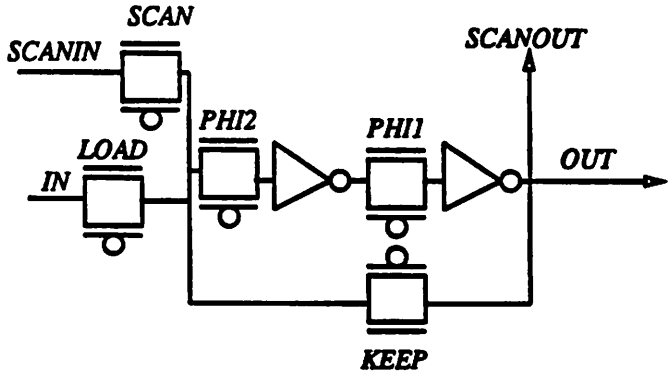


Figure 3.7: Scan-Path cell.

1. test-pattern generators;

2. output-response analyzers;

3. the **DUT**;

4. a **BIST** controller for controlling the **BIST** circuitry during self-test.

Testing a RAM using **BIST** requires a counter and a **LFSR**, which is a linear circuit constructed from *D*-type flip-flops and modulo-2 adders.

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 a **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 example is used to compress the output data from the RAM using signature analysis techniques.

For purposes of illustration, a RAM with 16 x 4 RAM will be considered as shown in Figure 3.8. The basic idea is to use the counter to supply the address data, and control to the RAM during a test. A 6-bit counter is used in this example. The four lower bits of the counter are used to supply data and addresses. The fifth bit is used to control writing and reading of the RAM. The sixth bit is used to invert the data going into the RAM during the test. By using a counter, we can easily implement the classical Walking One and Walking Zero testing protocols.

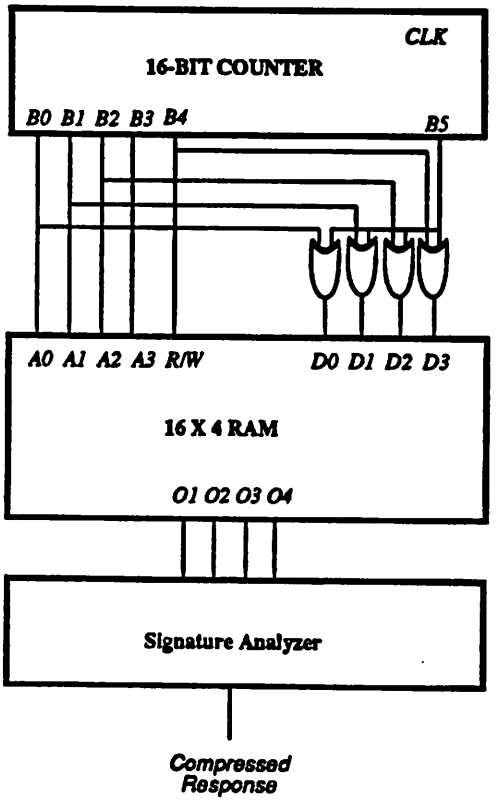


Figure 3.8: RAM test using **BIST**.

Chapter 4

SCANTEST Software

4.1 Overview

The SCANTEST software controls an entire testing sequence by providing a user on a host system with the means to apply test data to a DUT, to execute a particular test, and to capture the response from the DUT for analysis. This software essentially controls the TCB and is capable of executing three types of test namely scan-path, boundary-scan, and data acquisition. The software was written in the C-programming language and it must be compiled along with special VXWORKS header files (See Appendix A). SCANTEST is comprised of four main functions whose features are listed below.

SCANTEST ()

{
Prompt user for input file name and test type. format (binary or hexadecimal).
Open the input file and create the output file.
Call fileToBoard, runTest, and boardToFile routines.
Close files.
}

fileToBoard ()

{
Initialize TCB.
Check initialization status.
Download test vectors form the host to test data memory.
Parse input file.
Compute length of scan-path.
}

```
runTest ()
    {
    Fill scan-path with contents of test data memory.
    Execute test scan-path, boundary-scan, or data-acquisition tests.
    Poll for completion signal.
    }
boardToFile ()
    {
    Read the contents of the scan-path.
    Upload results to host for analysis.
    }
```

SCANTEST requires the input file to contain the register names, the register sizes, the register type, which is used to determine the what direction the data is shifted into the scan-register, and the register vaules. It then keeps track of all the registers and the order in which they appear in the **DUT**. We will demonstrate how to use the SCANTEST software to test a device in the following sections.

4.2 Scan-Path And Boundary-Scan Testing

During a scan-path test, the **DUT** shifts in the test data, runs for one clock cycle, and then shifts out the results. The functionality of the **DUT** can then be verified once the results are obtained. A SCANTEST user can easily perform a scan-path or a boundary-scan test by following the five steps given below. This, of course, is assuming that the user has a configuration identical to that shown in Figure 3.1 and that **DUT** contains the appropriate **TSS** hardware which is correctly connected to the **TCB**.

1. Open an X-window.
2. Remotely login to the development system.
3. Change to a working directory.
4. Load the SCANTEST executable.
5. Type SCANTEST to execute program.

We can execute a boundary-scan test by following the same steps outlined above and by selecting the boundary-scan test option when prompted for the test type. During a boundary-scan test, we can access the **DUT**'s scan-register, the boundary-scan register, and any **BIST**

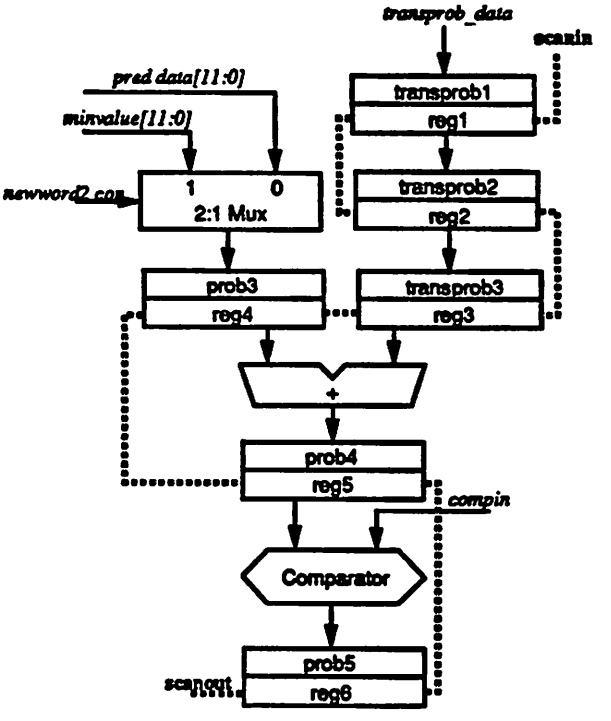


Figure 4.1: Data path (*dp1*) of Viterbi Processor with Scan-Path.

hardware through its TAP [7] interface. The Viterbi processor [6], which has scan registers in place of its pipeline registers, contains several smaller scanpaths and three additional pins (2 data, 1 control) namely *SCANIN*, *SCANOUT*, and *SCANTEST*, thus rendering it testable. The processor has been successfully tested with the TCB and SCANTEST software. A block diagram of one of the smaller scanpaths is shown in Figure 4.1.

4.3 Data Acquisition

The TCB was successfully used to test the front-end of a delta-sigma modulator which encodes an analog signal into a serial bit-stream of zeroes and ones. To acquire this binary bit-stream, the modulator output and a master clock are connected to the TCB. A scan-path test is performed and the bit-stream is collected by the test data memory. The bit-stream is then uploaded to a host computer where the it can be filtered using software filters and analyzed for signal-to-noise-ratio or harmonic distortion properties.

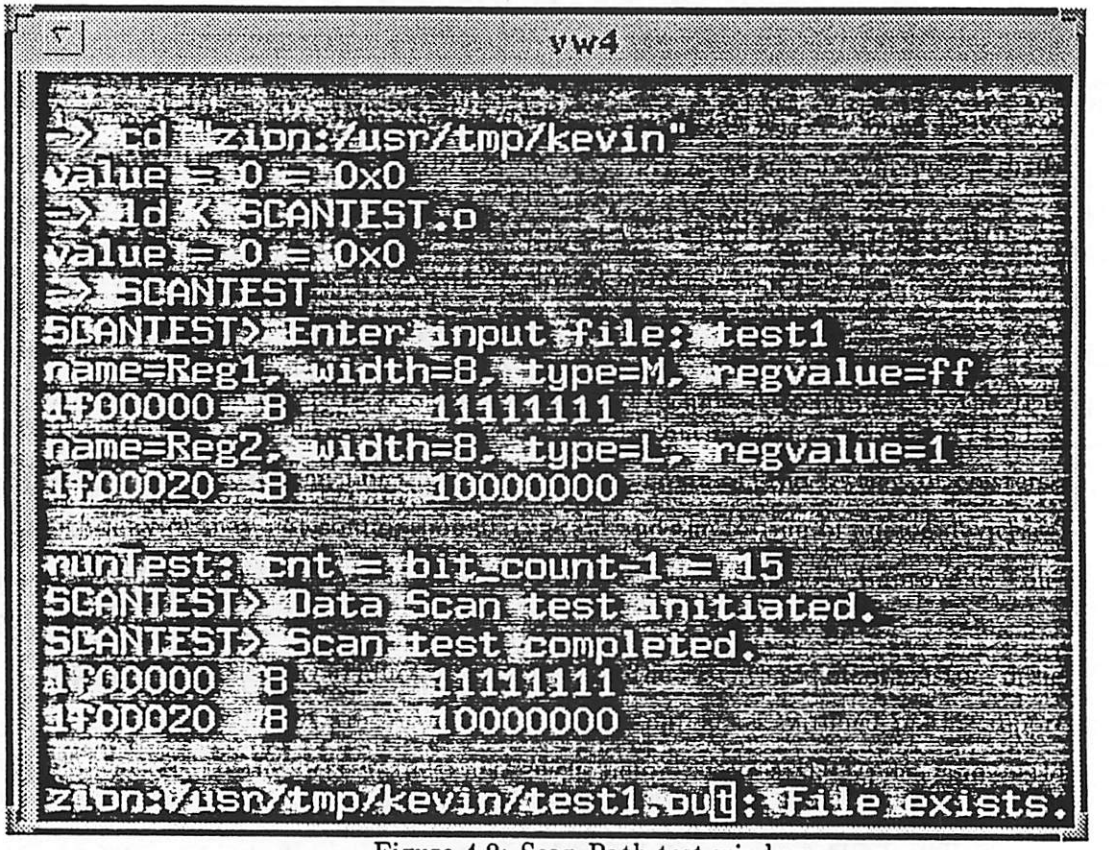


Figure 4.2: Scan-Path test window.

Chapter 5

Conclusion

The design of the **TCB** has been described. A fully functional 8-layer prototype of the **PCB** resides in a VME card-cage and is controlled by the SCANTEST software. The **TCB** supports the functional testing of a modules, the testing of clusters of chips which may or may not contain boundary-scan, and the testing of analog devices. The cost of implementing the **JTAG** boundary-scan in a given technology improve as you move to smaller technologies.

Most **DFT** techniques deal with either the resynthesis of an existing design or the addition of extra hardware to the design. The approaches outlined in this report require modifications to the design and affect such factors as area, I/O pins, and performance. The values of these attributes usually increase when these techniques are employed. Hence, a critical balance must exist between the amount of **DFT** used and gain achieved when employing these techniques. **DFT** techniquies are also used to reduce test generation costs. Without **DFT**, tests may have to be generated manually; with **DFT** they can be generated automatically. This project has provided us with some insight into ways of integrating test with the design process. It has also provided us with a platform for which we can investigate how to intelligently and automatically determine test hardware versus performance/area trade-off.

A cell library containing the boundary-scan register cells and scan-path register cells described in Chapter 3 has also been developed. Test chips containing boundary-scan and scan-path cells have been designed, fabricated in 2μ CMOS technology, and succesfully tested. The **BSBC** set is in the early stages of the design. Critical performance issues such as I/O fan-out and speed are being addressed.

Bibliography

[1] S.N. Funatsu, N. Wakatsuki, T. Arima, "Test Generation Systems in Japan" *12th Design Automation Symposium*, June 1975, pp. 112-114.

[2] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", *IEEE Trans. Comput.*, **C-30(3)**, 1981, pp. 215-222.

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

[4] P. Moore, D.S. Harrison, R.L. Spickelmier, A.R. Newton, "OCT, VEM and RPC", *University of California, Berkeley*, August 25, 1987

[5] C.B. Shung, R. Jain, K. Rimey, R.W. Brodersen, E. Wang, M.B. Srivstava, B. Richards, E. Lettang, S.K. Azim, P.N. Hilfinger, J. Rabaey, "An Integrated CAD System for Algorithmic-Specific IC Design", To be published.

[6] A. Stolzle, "A VLSI Wordprocessing Subsystem For A Real-Time Large Vocabulary Continuous Speech Recognition System", *Masters Report*, December 14 1989.

[7] J. Turino, "IEEE P1149 PROPOSED STANDARD TESTABILITY BUS - AN UPDATE WITH CASE HISTORIES", *IEEE International Conference on Computer Design*, January 1988, pp. 334-337.

[8] Sponsored by Test Technology Technology Technical Committee of the IEEE Computer Society, "Standard Test Access Port and Boundary-Scan Architecture", *Document P1149.1/D5 (Draft)*, June 20, 1989.

[9] T.W. Williams, K.P. Parker, "Design for Testability–A Survey", *Proceedings of the IEEE*, January 1983, pp. 98-112.

Appendix A

File Reference

The actual C-code and header files for the **SCANTEST** software are located in the:

1. /home/zabl/kornegay/masters/TEX/appendix/cfile directory.
2. /home/zabl/kornegay/masters/TEX/appendix/hfile directory.

The **SDL, BDS,** and **Magic** files are located in the:

1. /home/zabl/kornegay/masters/TEX/appendix/sdl.
2. /home/zabl/kornegay/masters/TEX/appendix/bds.
3. /home/zabl/kornegay/bscan/bsr/leafcells.

Appendix B

Photograph

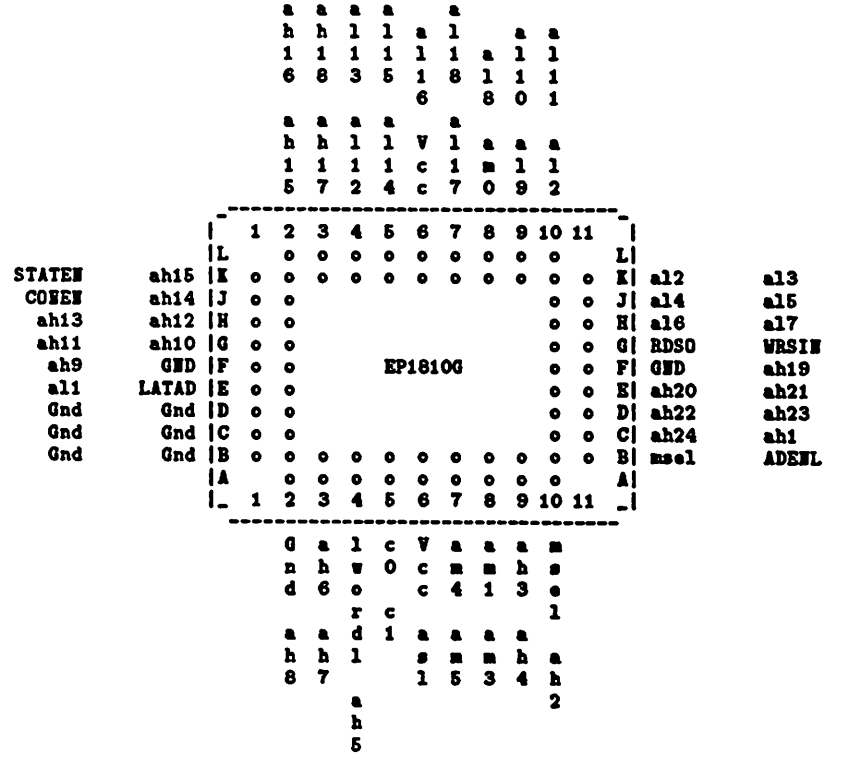
Appendix C

PLD Pinouts

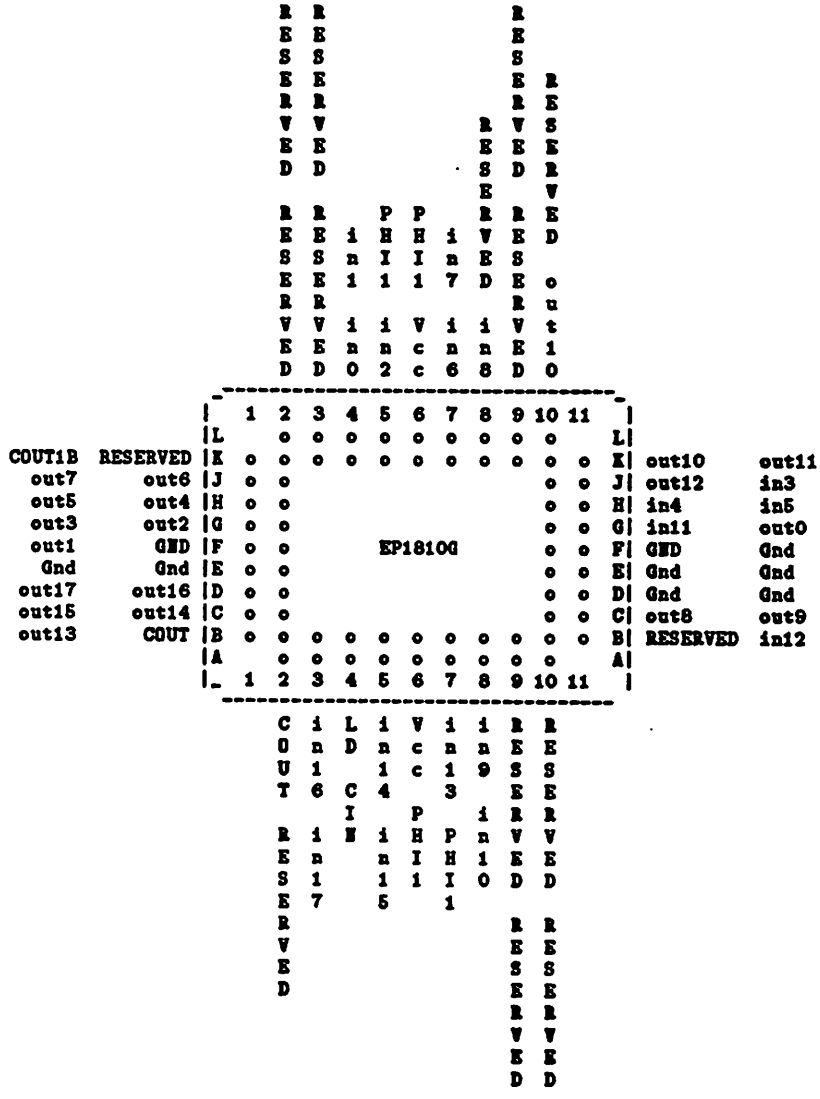
Pld2

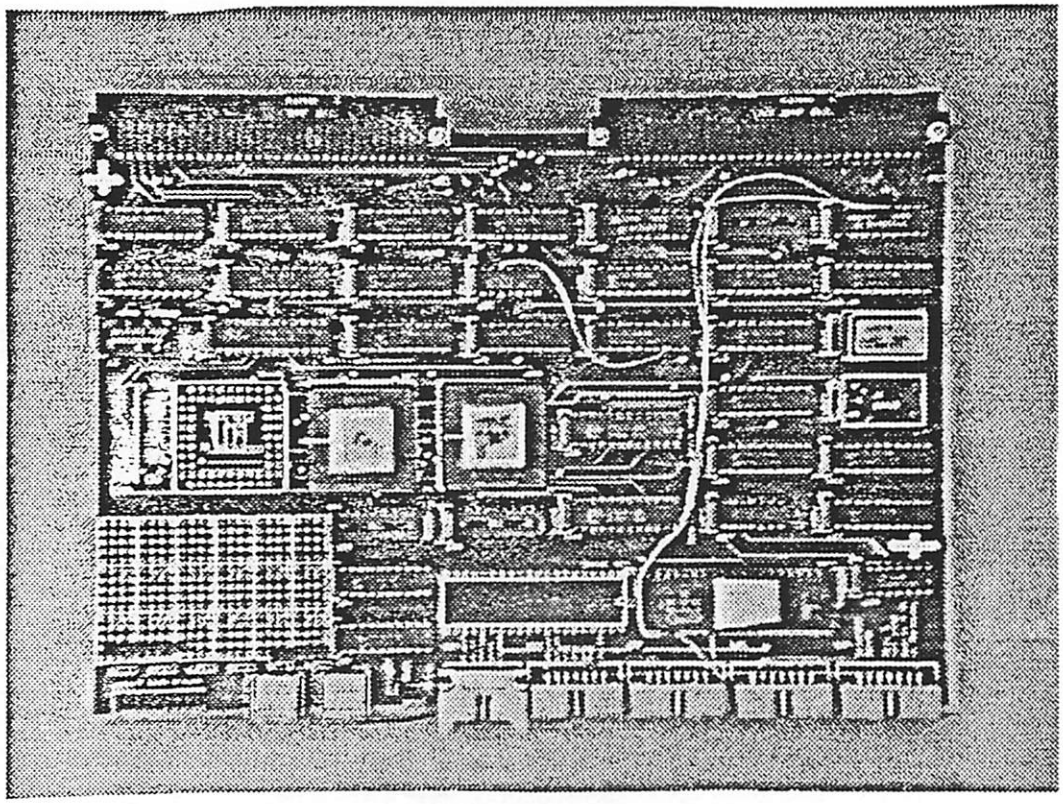
```
          EP900
        - - - - -
   PHI1 -|1      40|- Vcc
testmod -|2      39|- Gnd
   test -|3      38|- Gnd
  start -|4      37|- Gnd
 state3 -|5      36|- Gnd
 state2 -|6      35|- Gnd
 state1 -|7      34|- Gnd
 state0 -|8      33|- Gnd
  WRITE -|9      32|- Gnd
    TMS -|10     31|- Gnd
   sel3 -|11     30|- Gnd
   sel2 -|12     29|- Gnd
   sel1 -|13     28|- LATCH
   sel0 -|14     27|- CE
SCANTEST -|15    26|- DONE
   READ -|16     25|- LDCNTR
   cout -|17     24|- ack
    bs1 -|18     23|- c0
    bs0 -|19     22|- c1
    GND -|20     21|- PHI1
        - - - - -
```

Pld



Pld3

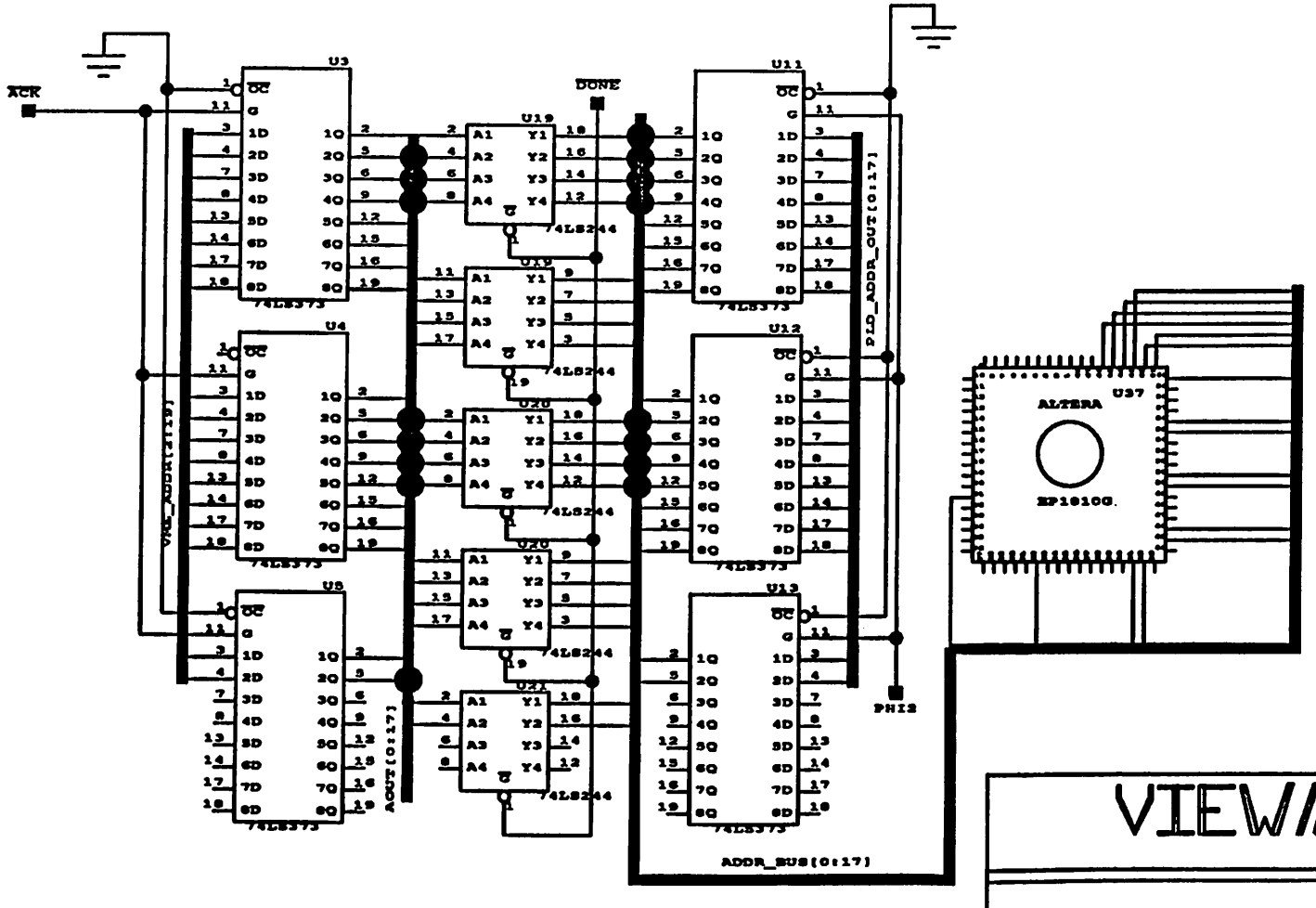

RESERVED
RESERVED
in1
PHI1
PHI1
in7
RESERVED
RESERVED
RESERVED
RESERVED
out10
COUT1B
RESERVED
out11
out7
out6
out12
in3
out5
out4
in4
in5
out3
out2
in11
out0
out1
GND
GND
Gnd
Gnd
Gnd
Gnd
Gnd
out17
out16
Gnd
Gnd
out15
out14
out8
out9
out13
COUT
RESERVED
in12
EP1810G
COUT
in16
LD
CI
in15
in14
Vcc
in13
in9
in10
RESERVED
RESERVED
RESERVED
RESERVED



Appendix D

Schematics

Address & Data Regs, PLD Decoder

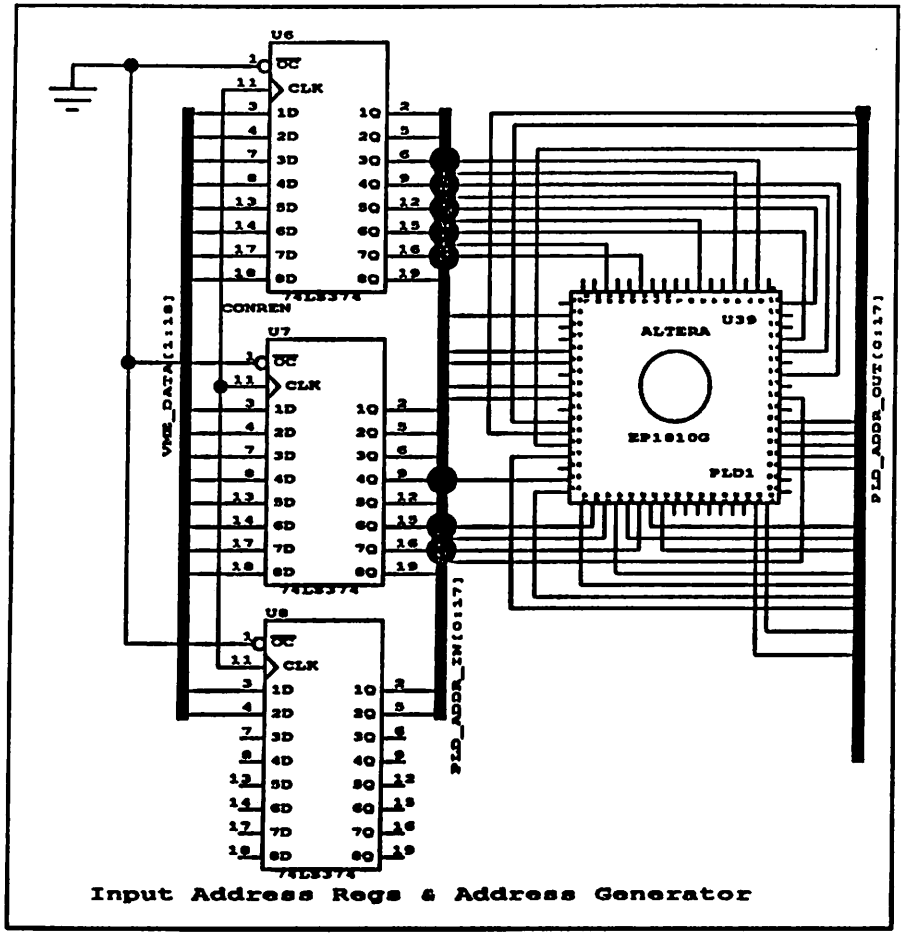


VIEWlogic

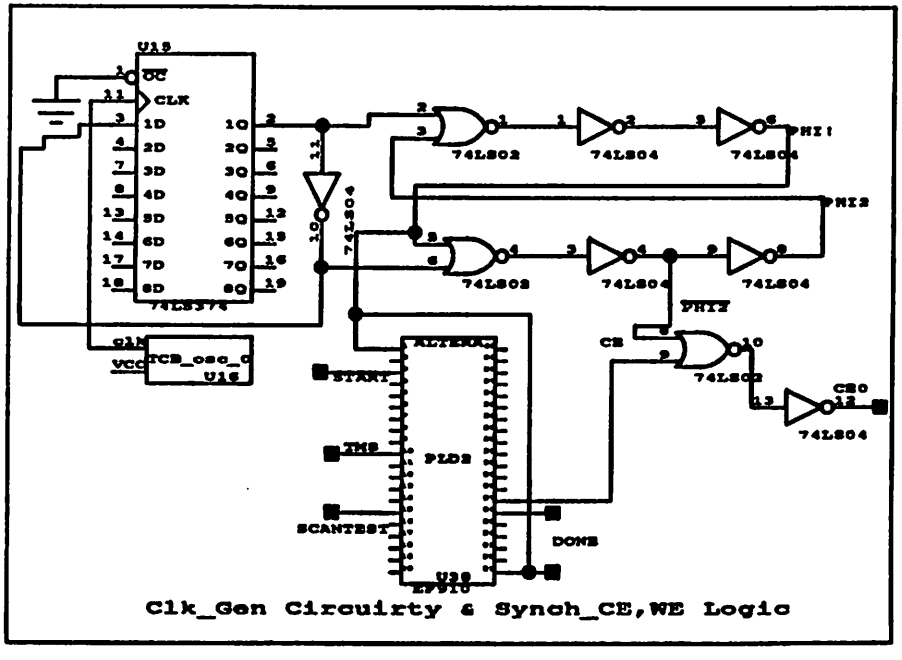
Test Controller Board

Part #1

SHEET: No. 1	DATE:	DRAWN BY: ktk



Input Address Regs & Address Generator



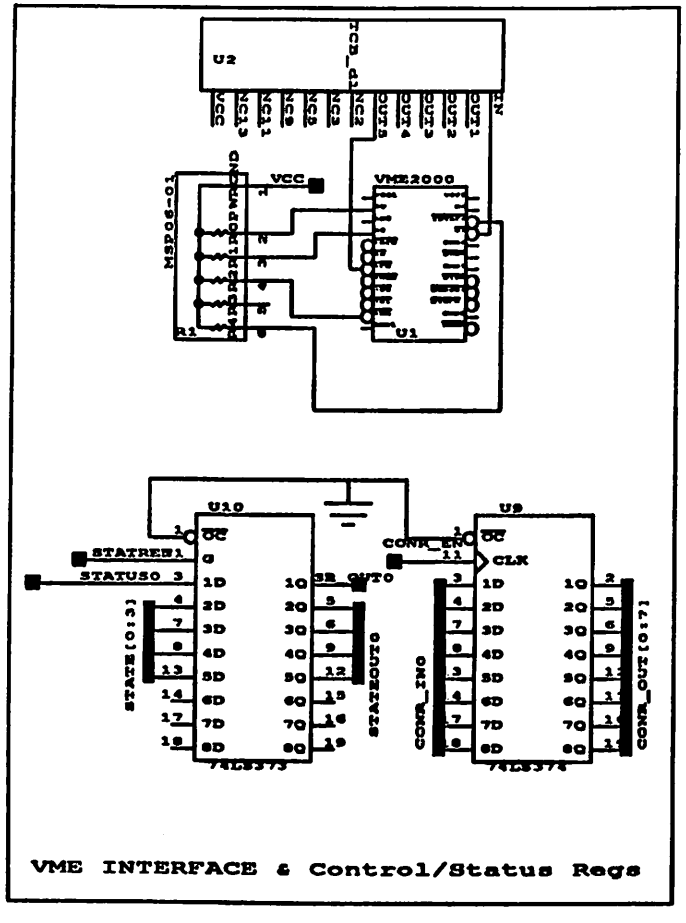
Clk_Gen Circuirty & Synch_CE,WE Logic

VIEWlogic

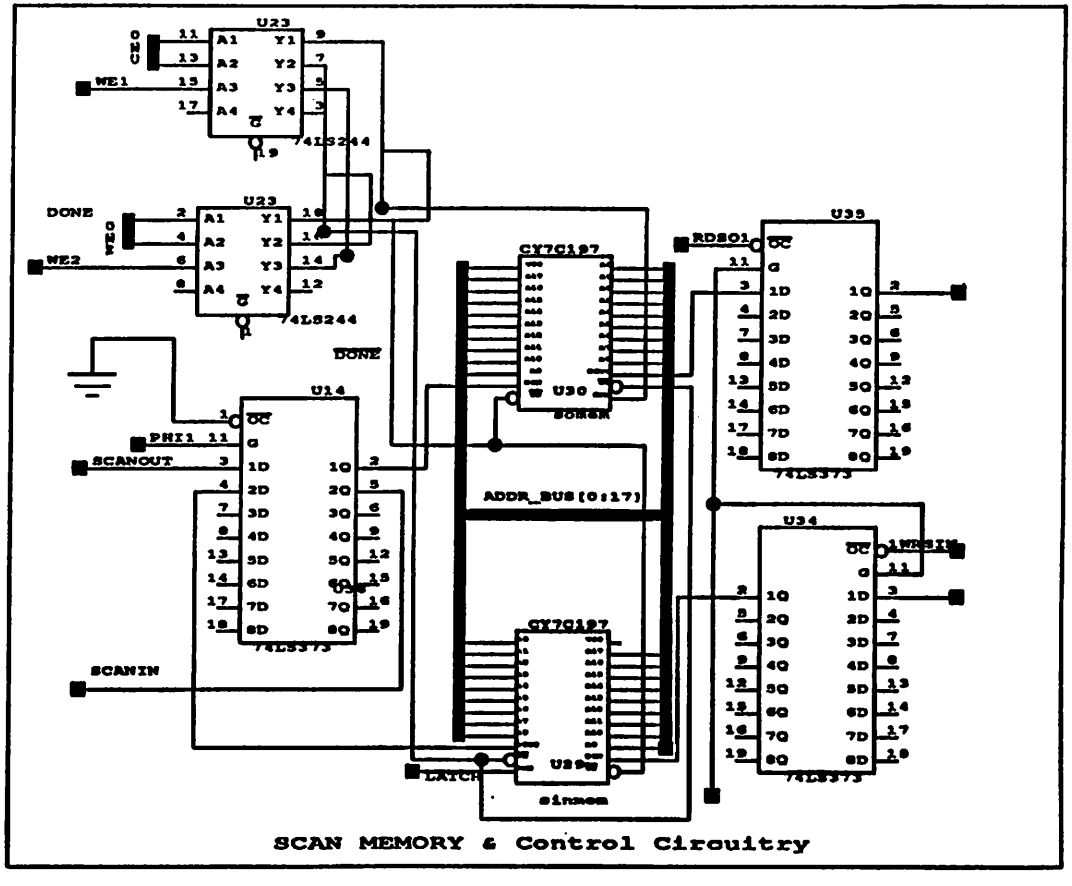
Test Controller Board

Part #2

SHEET:	DATE:	DRAWN BY:
NO. 2		ktk



VME INTERFACE & Control/Status Regs



SCAN MEMORY & Control Circuitry

VIEWlogic

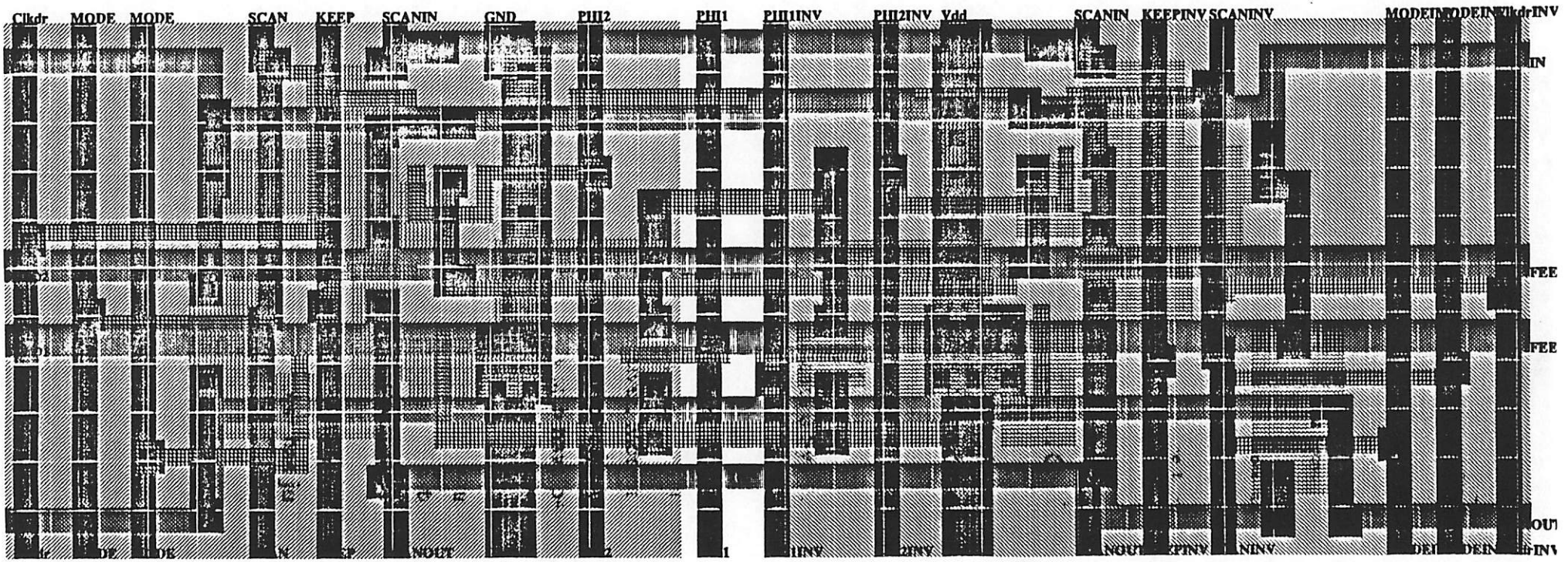
Test Controller Board

Part #3

SHEET: No. 3	DATE:	DRAWN BY: ktk

Appendix E

Layout


Clkdr
MODE
MODE
SCAN
KEEP
SCANIN
GND
PHI2
PHI1
PHI1INV
PHI2INV
Vdd
SCANIN
KEEPINV
SCANINV
MODEIN
MODEIN
ClkdrINV
IN
FEE
FEE
OUT
ClkdrINV