TANDEMCOMPUTERS

Built-In-Self-Test For The Tandem NonStop CLX Processor

David J. Garcia

Technical Report 87.6
November 1987
Part Number 11641

Built-In-Self-Test For The Tandem NonStop CLX Processor

David J. Garcia

Technical Report 87.6
November 1987
Part Number 11641

Table of Contents

Abstract 1

CLX Introduction 1

CLX Built In Self Test 1

Pattern Generator/Signature Register 2

MDP Software Control of BIST 2

Initialization for BIST 3

Memory System Operation 3

Integration with BIST 3

Deterministic Pseudo-Random Memory Operations 4

Initialization of the Memory System 4

Test Coverage 4

BIST Signature Management 5

BUILT-IN-SELF-TEST FOR THE TANDEM NONSTOP CLX PROCESSOR

David J. Garcia

Tandem Computers Inc.
19333 Vallco Parkway
Cupertino, CA 95014

Abstract

A built-in self-test (BIST) method is presented that uses pseudo-random test vectors and scan path design. The BIST method followed on the Tandem NonStop CLX™ processor is shown as an example. The pseudo-random test covers several custom ICs, commercial MSI logic, a static RAM array and their interconnects. Also, the BIST does a functional test of the dynamic RAM main memory and its control logic. Requirements for the BIST were that it be low cost and require minimal overhead to support the test function. Control of the test is handled by maintenance processor software, simplifying the hardware dedicated to BIST.

CLX Introduction

The Tandem NonStop CLX is a low-cost multiprocessor computer for online transaction processing. A CLX system can have from 2 to 6 single-board processors. The processor design is based on 4 custom CMOS chips, a 32K x 60 cache/control store static RAM array, and a 4 MB main memory array [1]. An optional expansion memory board of 2, 4, or 8 MB may be added to each processor.

Apart from the traditional Tandem design requirements of fault tolerance, data integrity and reliability, the CLX is required to be inexpensive and user serviceable. Furthermore, its use in an office environment requires such operations as diagnostics to be speedy and require no user intervention. A BIST function to test the CPU and memory met these requirements.

CLX Built In Self Test

The BIST on the NonStop CLX processor is based on scan, in which the majority of state elements within the processor are made of shiftable latches and registers. The BIST repetitively shifts pseudo-random patterns into the registers and latches, single steps the machine, and then shifts the results out, incorporating the results into a signature [2]. This is repeated until 2^{16} test vectors have been shifted in. The signature is then compared against that of a known good board, giving a pass/fail result. The system responds deterministically to the pseudo-random stimulus, responding with the same signature for good boards but a different signature for boards containing a detectable failure.

The test is completely self contained; it does not require control or data from other boards. The processor board and its external memory board are covered by BIST. The total test time is less than one minute. The test is run at power-on and on request by a maintenance process running in a separate CPU.

The BIST covers failures throughout the processor and memory boards, as well as within the core of the custom chips. The control logic for large RAM arrays is well tested, but the arrays themselves (cache static RAMs and main memory dynamic RAMs) are tested to a lesser extent.

The test does not cover the Tandem IO Bus interface, nor does it cover the Tandem Inter-Processor Bus interface. Testing these areas would involve different processors, and BIST is only intended to verify the one processor. Running in a single processor, BIST has no way to control and observe these external interfaces without affecting the shared bus.

The BIST test of memory changes the memory contents. Because of this the BIST can be done in two ways -- either cold BIST or warm BIST. The two techniques differ in the way they affect memory. In cold BIST the contents of the main memory are written to and tested. This type of test is suitable for use before a cold start of the processor when the contents of main memory are not important. In a warm BIST the contents of main memory are not disturbed. Of course this version of the test is not as thorough. A warm BIST is suitable for testing the processor after a power fail when battery backup preserved the contents of main memory, allowing a resumption of existing jobs. Unless specified, for the rest of this paper BIST refers to cold BIST.

While the scan-based design necessary for BIST is expensive, it allows for many other attractive features. The NonStop CLX uses the scan feature within development engineering, manufacturing, and in the field. For test and debug, scan gives excellent observability and controllability of registers and latches [3]. A stand-alone tester to screen custom chips uses a pseudo-random scan/step/collect signature method. Another tester does board-level repair using scan techniques. The board level tester does pseudo-random pattern testing, and when a failure is found, it masks off scan strings until the problem is isolated. Finally, scan is used in the field to determine the state of failing boards. This information allows diagnostic software to determine the cause of failure and prescribe an appropriate fix [4].

Scan Implementation

Most of the state elements on the CPU board are scannable. That is, they operate in one of three modes: normal, frozen, or scanning. When frozen, registers or latches hold their state. When scanning, the registers and latches are configured into a large shift register, allowing their entire contents to be shifted in and out.

Among the numerous scan methods [5] a single-clock multiplexed-data scheme was chosen. That is, a free-running clock is distributed to all registers, and the normal, frozen, or scan modes are determined by control lines (fig. 1).

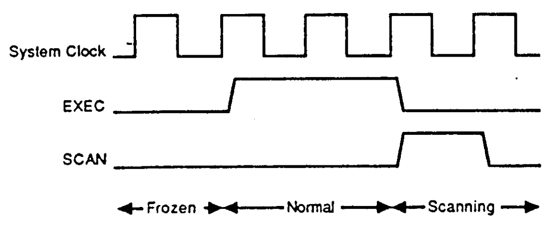

Figure 1, Scan Control Signals

This scan method allows state elements from different technologies and clocking regimes to coexist in one scan string and use the same control signals. The custom CMOS chips have shifting latches which use two-phase non-overlapping clocks [6], while the scannable MSI FAST parts and PALs use a single, edge-triggered clock. The same free-running system clock is distributed to all state elements (the two-phase clocks for the custom chips are generated internally).

Pattern Generator/Signature Register

The BIST uses strings of pseudo-random bits generated by a 16 bit linear feedback shift register (LFSR) [7] called the PPG (pseudo-random pattern generator). The BIST signature is collected by another 16 bit LFSR called the SR (signature register). The characteristic polynomial of the two LFSRs is $x^{16} + x^{12} + x^9 + x^7 + 1$. Figure 2 shows the PPG and SR designs.

The PPG and SR must be writeable for initialization and the SR must be readable to determine the results of the test. Both registers reside in a custom chip, occupying less than 5% of the chip's logic area. To minimize IO connections to the chip, the PPG and SR are connected to a scan string. When not doing BIST the registers may be examined and set using scan. During the BIST, the two registers are removed from the scan string and operate as LFSRs.

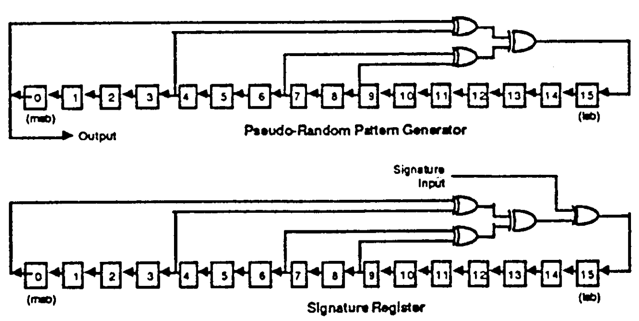

Figure 2, PPG and SR Description

MDP Software Control of BIST

The decision to include a BIST feature on CLX was made after the decision to include scan in the processor design. To achieve the CLX design requirement of low-cost, control of the BIST function had to added with minimum hardware overhead. The requirement was met by maximizing the use of MDP software to control the BIST function.

The main function of the MDP, the Maintenance and Diagnostic Processor, is to manage microcode loading, fault analysis, and scan functions. The MDP, based on the 6803 8-bit microprocessor, is central to the control of the BIST function. The MDP executes the proper sequence of steps for initialization and directs the scanning and stepping of the processor. During the BIST function, the MDP software is completely deterministic. There are no real-time interrupts or data-dependent steps allowed.

The MDP can control the scan strings to shift one bit at a time into or out of scannable registers. To transfer a bit between the MDP and a scan string (one of 13), the MDP writes a certain address, causing the corresponding scan string to shift in the high order bit of the microprocessor data bus. A read to the same address gives the current scan-data-out of the string on the low order data bus bit. This allows direct observability and controllability of all scannable registers. This mode of scanning could theoretically be used for BIST, i.e. one bit could be read from the PPG and then shifted into a scan string. This method would be slow, since at microprocessor speeds, several microseconds per bit would be needed.

The BIST on the CLX uses a faster way to load the scan strings with pseudo-random data (fig. 3). When in BIST mode, the decoders that assert the individual scan control signals operate differently. Normally, the SCAN<nn> control line is asserted for one system clock when the appropriate MDP address is written to. In BIST mode, a write to this address causes the SCAN<nn> signal to assert continuously, and all other SCAN signals to deassert. The input to the scan strings, SCAN_IN, is driven by the PPG output. With this method, the scan strings can be loaded at a rate of one bit per system clock.

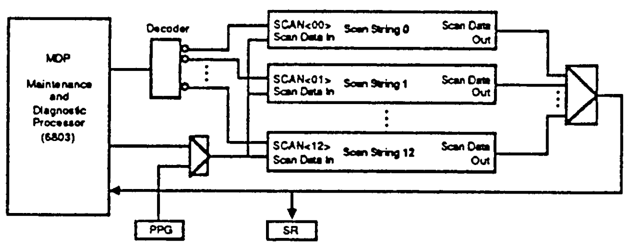

Figure 3, Block Diagram of BIST Structure

To load the scan strings with pseudo-random data, the MDP does a write to the address which asserts SCAN<00>. The software then enters a wait loop until enough bits have been shifted in. The MDP then writes to the address which asserts SCAN<01>, causing SCAN<00> to deassert. This is repeated for the other scan strings until all are loaded. The MDP software need not shift in exactly the right number of bits for the scan string. All that is required is that the minimum number of bits for each scan string be shifted in. A string is at most 255 bits (a constraint imposed by our board-level tester), so MDP software shifts a fixed number of bits into each string.

The BIST loads a test vector into all scannable registers/latches before single stepping the machine. The length of the test vector generated by the PPG is determined by the total number of bits shifted into the scan strings, not simply the number of bits in the scan string (since MDP software shifts in a number greater than 255). Since the pseudo-random pattern is generated by an LFSR, the test vector length determines the number of unique test vectors generated.

Initialization for BIST

For the BIST to return a stable signature, all data incorporated into the signature must be deterministic. This requires that all state bits affecting the signature be initialized. Scannable registers/latches are initialized through scan, non-scannable state elements through different processes.

Small RAM arrays and register files (non-scannable) are initialized by repeating the BIST, and only using the signature from the second pass. That is, the 2^{16} iterations of scanning in test vectors and single stepping the machine are repeated twice. The PPG and SR are reinitialized between the two passes. Since the RAMs are small, the first BIST pass will write all those locations that the second BIST pass will read.

A different method is required for large RAM arrays, as there is a distinct probability that BIST will read a location that it never writes to. For these RAMs, initialization is also done in a two pass approach. During the first pass, a control signal is asserted which forces all read operations to writes. Then, during the second pass, the control signal is deasserted, and the same pseudo-random test vectors are applied. Thus, all the locations that the BIST second pass reads contain initialized, pseudo-random data.

Further initialization issues are covered in the section on initializing a memory system.

Memory System Operation

The CLX memory system includes an instruction/data cache and page table cache in static RAM and between 4 and 12 megabytes of main memory dynamic RAM [1]. 4 MB of memory are on the CPU board and the rest are on the optional memory expansion board. The main memory interface to the CPU supports transfers of single words at random addresses, and higher speed transfers of words from contiguous addresses. The memory system also provides support for memory to memory block moves and data alignment.

Integration With BIST

The main memory system is functionally tested by BIST. BIST gives good coverage of the control logic, buffers, and interconnect but only does a partial test of actual DRAM cells in the main memory. The memory system is tested functional. That is, BIST applies pseudo-random operations to the memory system, along with pseudo-random address and data, and incorporates the results of those operations into the BIST signature. The memory system is not scanned and no attempt is made to systematically test every location in memory.

Even without exhaustively testing all DRAMs, the BIST test of main memory is quite adequate for our needs. First, it tests at least a few memory locations in each dynamic RAM along with all the memory support circuitry external to the DRAMs. Total failure of a dynamic RAM, such as a missing component or a floating pin, is detected. Failures undetected by BIST would be confined to a single DRAM and would be easily detected by a microcode diagnostic.

The BIST must test the memory system differently from the rest of the processor. The processor is tested by scanning in pseudo-random data, single stepping the machine, and scanning out the data. The memory system cannot be tested in this fashion because of the DRAMs. The dynamic RAMs' periodic refresh would be interrupted if pseudo-random data were to be scanned into the control logic. Also, DRAM operations take longer than one CPU system clock, e.g. a memory read takes 4 cycles to complete. The normal pseudo-random test expects all operations to complete in a single cycle. Thus, the CPU is operated for one system clock and then results are scanned out and incorporated into the signature.

For test purposes, the memory system operates separately from the CPU. While the CPU is frozen, scanned and stepped by BIST, the memory system continues to run normally, doing periodic refreshes of the dynamic RAMs. The BIST test of the memory system occurs when the CPU interacts with the memory system. During BIST the CPU pseudo-randomly does reads and writes of pseudo-random addresses with pseudo-random data. Results of these memory operations are loaded by the CPU and then incorporated into the BIST signature. In effect, the BIST does a functional test of the memory system.

This type of testing places several requirements on the memory system. Special memory system features to support BIST are listed below:

- The memory system must be able to accept single stepped operations from the CPU.
- The memory system interface to the CPU must ignore pseudo-random bits scanning past the interface.
- The memory system must respond deterministically to any command from the CPU presented in any sequence.
- The memory system must give a deterministic response to the CPU in the presence of events such as refresh.
- There must be a mechanism for initializing all state elements observable by BIST.
- The memory refresh must occur deterministically relative to the start of BIST.

During the BIST, the CPU is scanned and single-stepped while the memory system continues executing every cycle, maintaining the dynamic RAM refresh operation. When the CPU is single-stepped, an operation may be requested of the memory system. Possible operations are writes to memory, initiation of a read from memory, transfer of data from a read FIFO to the CPU, or read or write of status registers.

The memory system only acts on requests from the CPU when it is executing normally or when it is being single stepped (as it is during BIST). When the CPU is frozen or being scanned (see fig. 1) the EXEC signal is deasserted and the memory system ignores the bits being scanned past its interface with the processor.

When the BIST single steps the CPU, the memory system takes the pseudo-random bits at its interface with the CPU and executes the command. The command is executed in successive clocks. That is, the memory system is not single stepped like the CPU.

In the case of a read or write to a memory system status register the operation is completed in one cycle and the results are loaded by the CPU and are incorporated into the scan string.

In the case of a multi-word read , the operation is completed in 7 cycles (for a 4 word read) and the read data is buffered in a memory system FIFO. The data waits in the FIFO until a single stepped pseudo-random FIFO-to-CPU transfer operation occurs. Once transferred to the CPU, the data is incorporated into the scan string and hence influences the BIST signature. The read data cannot bypass the FIFO and go directly to the CPU because data is being read from the DRAM in a stream of back-to-back clocks. The CPU is unable to accept data on back-to-back clocks while it is being scanned for the next BIST iteration.

In normal operation, a CPU microinstruction advances down a multistage pipeline. For example, Figure 4 diagrams a line of microcode to do a memory write. The interface signals between the CPU and memory system are derived from both R1 and R2 pipeline stages.

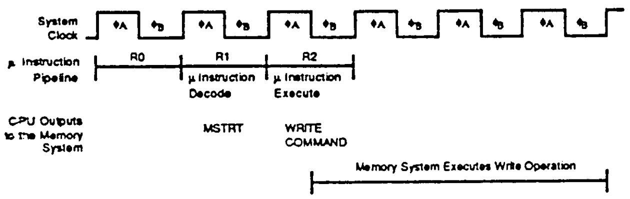

Figure 4, Microinstruction Pipeline for Memory Operation

In the case of memory reads and writes, the R1 signal MSTRT always precedes the R2 encoded signal to do a read or write. When a line of microcode is single-stepped, the pipeline advances one stage for each single step. There may be many cycles between R0 and R1 or between R1 and R2 where the processor is frozen. When a memory operation is single-stepped in R2 (i.e. EXEC is asserted by the falling edge of ϕ_A) the operation is said to be committed. The committed operation will complete in the next 4 system clocks, even if EXEC is freezing the processor.

During BIST, pseudo-random data is scanned into all pipeline stages of the processor (but not into the memory system pipeline stages). For example, in order for the memory system to do a read operation, the memory system expects to see an MSTRT in the R1 stage and in the following single-step see the appropriately encoded read command in R2. For this sequence to happen, pseudo random data has to, by chance, load the pipeline correctly on successive BIST iterations. If the BIST does not load the pipeline correctly, the memory operation will not complete. Instead of accessing the memory, there is some other deterministic response (such as not doing anything).

Deterministic Pseudo-Random Memory Operations

The memory system's response to any request during BIST must be deterministic. This is also true of invalid requests such as would occur if the previously described pipeline stages were set to an invalid state. Not only should the response of the memory system be deterministic, the control logic should remain in a state where it can service further memory requests and service the DRAM refresh. That is, it is not acceptable for a normally illegal sequence of requests to cause the memory control state machines to halt or go to undefined states. While this would most likely be deterministic, it would prevent the memory system from participating in any further functional tests.

The memory system must respond to any sequence of operations without producing nondeterministic results. For example, a series of read FIFO-to-CPU transfers could empty the FIFO. The data value transferred from the empty FIFO must be some deterministic constant, and not be dependent on a random power-on state for example.

Initialization of the Memory System

Unlike most of the processor, the memory system is not initialized by scan, as the memory system control logic must preserve the dynamic RAM refresh function. The refresh interval is initialized by synchronizing its count to the start of BIST. During a refresh, the memory system asserts a control signal to the CPU indicating that it is busy and cannot respond to any requests. Failure to synchronize BIST with refresh would result in nondeterministic results being sent back to the CPU and cause an unstable BIST signature.

The refresh counter and other control logic is synchronized by the MDP software asserting a signal at the start of the BIST. This ensures that refresh interactions will always be consistent across every run of the BIST. Note that the refresh interval may change during the course of BIST due to the CPU pseudo-randomly writing to the memory system's refresh interval register.

Test Coverage

The exact test coverage of the BIST on CLX is unknown as we were unable to run a complete fault simulation. The large number of test vectors with over 350 chips, including commercial and custom VLSI and large static and dynamic RAM arrays, are an obstacle to simulation.

The BIST does not cover the IO channel interface or interprocessor bus interface. Both of these connect to external sources which cannot be controlled or observed by the BIST. Of the remaining logic, very high fan in circuits, such as the comparators for the cross-coupled chips, are known not to be covered. The logic known not to be covered by BIST is 5-10% of all the logic on the board.

Even though the DRAMs are not exhaustively tested, the BIST coverage of the memory system is quite good for control logic, board wiring, and address/data paths.

In the first 160 processor boards built, there were no cases of failures (outside of the areas known not to be covered by BIST) that were undetectable by BIST. In some cases BIST indicated a board to be bad even though the board passed all other tests and seemed to function normally. In these cases it is assumed that BIST detected a failure in a path not sensitized by normal operation.

BIST Signature Management

After the BIST is completed, MDP software determines if the test passed or failed. The generated signature is compared against known good signatures kept in EEPROM. There is more than one valid signature possible. The signature is dependent on the type of BIST, either warm or cold, and the size of the external memory board installed.

A simple solution would be to require that the correct warm and cold BIST signatures be updated any time a processor or memory board is installed. This is undesirable for a system intended to be user serviceable, as it would add a manual step to the installation process.

Keeping a list of known good signatures only on the processor or memory board presents a version control problem. A new size or version of memory board unknown to the processor would cause a signature miscompare and erroneously indicate a failing board.

The solution is a list of signatures in non-volatile memory kept on both the processor and memory boards. On the processor board, a list is kept of all good signatures possible when that processor board is paired with any memory board. On the memory board, a list is kept of all good signatures possible when that memory board is paired with any processor board.

After completing the BIST, the maintenance processor scans the list on both the processor board and the memory board. Failure to find a match on either list indicates that the board does not pass the BIST. This approach allows engineering changes or new versions of either board to change the correct BIST signature without updating the signature list previously stored on the other board.

Conclusion

The BIST design presented here does a good test of the board without high hardware overhead. Keeping control of the test in maintenance processor software minimized hardware overhead. The test of the processor board covers much of the dynamic RAM memory system as well as the scannable logic.

References

[1] Lenoski, D.E., "A Highly Integrated, Fault Tolerant Minicomputer: The NonStop CLX", *Digest of Papers, Compcon Spring 1988*, San Francisco, CA 1988.

[2] McCluskey, E.J., *Logic Design Principles: With Emphasis on Testable Semicustom Circuits*, Prentice-Hall, Englewood Cliffs, NJ 1986.

[3] Staas, G., "TDL: A Hardware/Microcode Test Language Interpreter", *17th Annual Microprogramming Workshop*, New Orleans, LA 1984.

[4] Tandem Computers Inc., *Tandem Maintenance and Diagnostics Manual*", Tandem Part No. 82387, 1986.

[5] McCluskey, E.J., "A Survey of Design for Testability Scan Techniques", *VLSI Semicustom Design Guide*, Summer 1986.

[6] Mead, C.A. and Conway, L.A., *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA 1980.

[7] Peterson, W.W. and Weldon E.J., *Error Correcting Codes*, MIT Press, Cambridge, MA 1972.

™ Tandem, NonStop, and CLX are trademarks of Tandem Computers.

Distributed by
TANDEMCOMPUTERS
Corporate Information Center
19333 Vallco Parkway MS3-07
Cupertino, CA 95014-2599