TESTING CONVENTIONAL LOGIC AND MEMORY CLUSTERS USING BOUNDARY SCAN DEVICES AS VIRTUAL ATE CHANNELS

Peter Hansen

Teradyne, Inc.
321 Harrison Avenue
Boston, MA 02118

ABSTRACT

While Boundary Scan simplifies board-level testing, most boards in the next several years will contain conventional logic as well as Boundary Scan components. In cases where a tester cannot contact the non-scan circuitry via bed-of-nails fixture or special test points, the inputs and outputs of onboard Boundary Scan devices may be used as virtual ATE channels to test clusters of conventional logic functionally.

INTRODUCTION

Test technologists have heartily embraced Boundary Scan as a solution to the testability problems presented by complex VLSI/ASIC circuit boards and surface-mounted device packaging. It is certain that the most popular implementation of Boundary Scan will be the one developed originally by the Joint Action Test Group (JTAG) and now described by the proposed IEEE Standard P1149.1.

Boundary Scan simplifies the problem of testing and diagnosing structural defects like shorts, opens, and stuck-at pin faults, which account for the great majority of board faults. Because structural faults are easier to detect and diagnose than timing-related performance problems, it makes good economic sense to do the best possible job of screening out structural defects prior to performance testing of a board. Boundary Scan accomplishes this goal — with shorter programming times, higher fault coverage, and better diagnosis than either in-circuit or functional testing for structural faults.

Boundary Scan equips a device with shift register elements that accompany each I/O pin. These elements are connected to form a shift-register path around the periphery of the IC. If a board were constructed entirely from components built using Boundary Scan, the shift-register paths of all devices could be connected to form a single data path, through which a test system could control and observe all device pins and associated interconnects.

All testing for stuck-at faults could as a result take place from the edge connector. This testing would include verification of both device logic (using the Boundary Scan internal test mode) and the interconnects between devices (using the external test mode)[1,2].

While it will be practical to build most gate array and standard cell ASICs with Boundary Scan within the next year, and while a few commercial devices such as Texas Instruments' SCOPE™ OCTAL parts [3] now are becoming available with Boundary Scan, it will take many more years for most commercial logic to be offered with this capability. In fact, the economics involved in designing and producing small logic devices and memories may preclude these parts from ever incorporating Boundary Scan. As a result, we believe that the most common board design approach by far will be one which mixes Boundary Scan components with clusters of conventional logic ICs and memory devices.

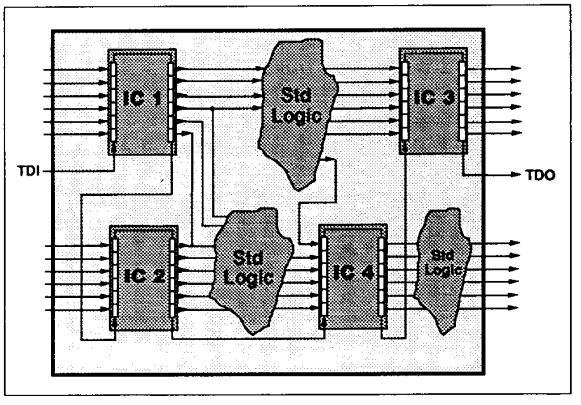

Figure 1.

MIXED-TECHNOLOGY BOARDS

This mixed-technology approach may require a hybrid test strategy. Boundary Scan components and their interconnections will be tested using serial test patterns clocked along the Boundary Scan path on the board, while the board's conventional logic (as well as analog and mixed-signal circuitry) will be tested using current in-circuit or functional-cluster test techniques.

From a programming, testing, and diagnostic point of view, the simplest way to deal with conventional-logic clusters on a mixed-technology board is to surround each cluster with ATE channels. The channels would contact individual or clustered components through either a bed-of-nails fixture or special test points [4].

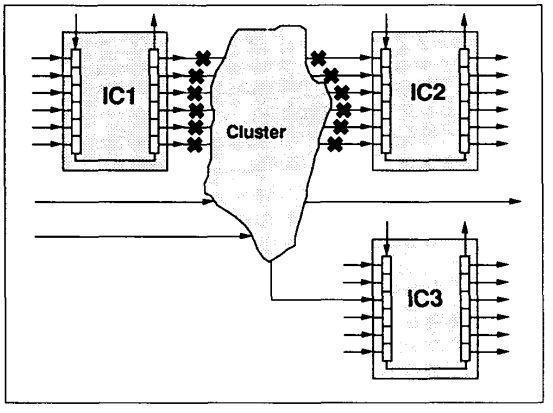

Figure 2.

Many of today's board test applications, however, preclude this solution. Conformal coatings on boards — or contract restrictions — may rule out bed-of-nails probing, for example. Test pads may require unacceptable amounts of board real estate.

In cases like these, an alternative means must be found to gain access to non-scan logic clusters for testing and diagnosis. A promising new approach treats the leads of Boundary Scan devices surrounding a non-scan cluster as virtual ATE channels. The advantage of doing this — rather than resorting to structural testing from the edge connectors — is that the cluster is considerably smaller than a full board and therefore allows far easier generation of high fault-coverage programs and faster, more accurate fault diagnosis.

In this approach, a tester loads test stimulus via the Boundary Scan path into Boundary Scan device outputs. These outputs then drive the data to the inputs of the non-scan logic cluster. The cluster's response is captured by the input pins of other Boundary Scan devices, loaded into the pins' associated shift-register elements, and clocked out along the Boundary Scan path for analysis.

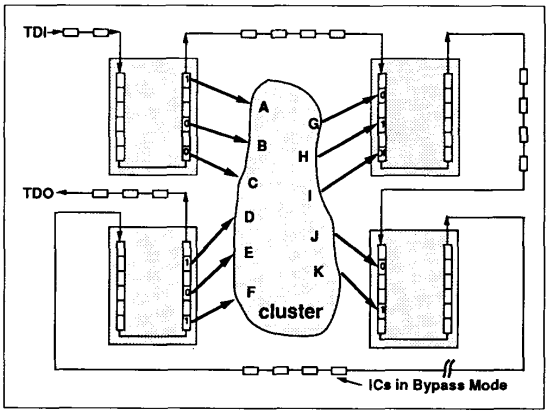

Figure 3.

BOARD DESIGN CONSIDERATIONS

A board designer can contribute significantly to the success of this test strategy by taking into account the special requirements of testing through the Boundary Scan I/O.

For one thing, care should be taken to minimize the sequential depth of the board's conventional-logic clusters — which, at the present rate of technological development, can be expected to rival a VLSI board of today in complexity. Using TI's Boundary Scan SCOPE™ OCTAL components in an otherwise conventional logic cluster, for example, would provide improved controllability and observability, and could allow that cluster to be partitioned into one or more simpler circuits which could be tested separately via Boundary Scan virtual channels.

It also should be remembered during design that the shifting operations involved in clocking test data along a Boundary Scan path limit the speed of test vector application and present keep-alive problems for dynamic parts. A designer might resolve this issue by providing test points through which dynamic ATE channels in a tester could provide the necessary keep-alive signals.

TEST PATTERN GENERATION

The approach used to generate test patterns for testing conventional-logic clusters functionally via onboard Boundary Scan I/O depends on the composition of the cluster. Clusters composed exclusively of memory devices or analog/mixed-signal ICs, for example, will probably require manual generation of functional test patterns.

Other clusters may be populated by a mix of diverse components. These could include TTL parts, various types of programmable-logic devices, RAM and ROM chips, and other off-the-shelf commercial VLSI devices.

A random-logic cluster like this should be treated as if it were a discrete circuit board for the purpose of test pattern generation, and the cluster test should be developed using a simulator. To generate the simulation model of the cluster, an engineer uses netlist-editing tools to extract device and device-interconnect data from the full-board netlist.

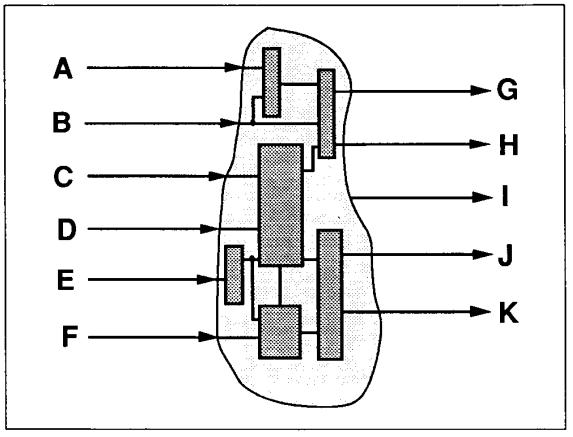

Figure 4.

The Boundary Scan cells that surround the cluster are now treated as if they were simple ATE channels that can be thought of as Virtual Primary Inputs and Outputs. The engineer then manually generates test stimuli, which are applied against the cluster model in logic and fault simulation. When fault simulation indicates that the desired level of fault coverage has been reached, the stimulus and response data from the simulation is postprocessed as if the cluster were to be tested directly by ATE channels[5].

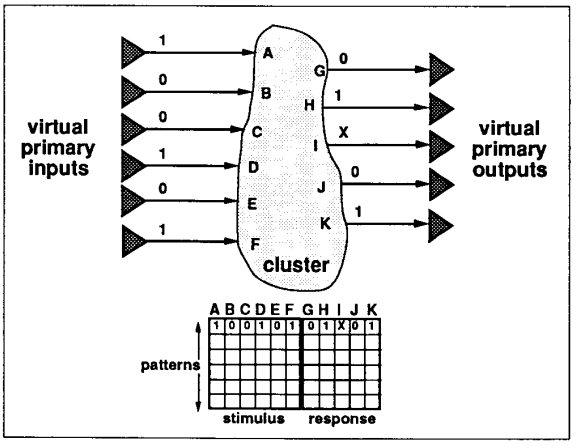

Figure 5.

High fault coverage is essential because VLSI/ASIC/SMT boards typically contain many shorts and opens, and because the structural cluster testing being described here will usually represent the very first loaded-board test for this class of fault. Test programs must in consequence provide higher fault coverage than ever before — at least 98% or 99% on stuck-at and open-input faults, as measured by the fault simulator — to prevent excessive numbers of faulty boards from passing undetected to later production stages. Achieving these levels of fault coverage generally entails the generation of large numbers of test patterns.

THE BOUNDARY SCAN ENVIRONMENT

The need for high fault coverage isn't the only factor that inflates the amount of patterns needed to test conventional logic on a board. The static nature of the cluster test, which is dictated by the speed limitations inherent in testing through the Boundary Scan path, also demands much larger pattern sets than does conventional testing.

Unlike real ATE channels, Boundary Scan virtual channels — which are only device I/O pins, remember — have none of the data formatting (i.e., return-to-zero and return-to-one) or complex timing capabilities that an ATE channel employs to convey large amounts of information with each pattern[6]. So information such as an entire bus cycle that a sophisticated board tester can pack into a single pattern for normal testing now requires several patterns.

Whether generated by hand or by a simulator, the cluster-test patterns are best produced in a conventional format — that is, one which assumes that the tester will be applying the patterns in parallel to a number of primary inputs, and detecting circuit responses in parallel from a number of primary outputs. This is the most convenient way to express the patterns from a test engineer's point of view; it also is efficient in terms of data storage.

But these patterns must be serialized for use in the Boundary Scan test, by any of several techniques which will be discussed in the next section. Software or hardware that performs this function will be referred to as a serializer. Serialization, however, further increases the amount of test data which a test system must cope with.

Another factor contributing to the heavy data load involved in testing a conventional-logic cluster through the Boundary Scan path is the fact that the shift-register cells that act as virtual channels typically account for a very small portion of the total Boundary Scan path. Many more shift-register elements in the path don't actively supply data for the test — nor do they receive response data from the cluster.

In any case, every shift-register element on the path must carry some logic value for the test. Sometimes, this logic value may be assigned an arbitrary background pattern, and be considerated as completely neutral and without function for the test of the cluster. Alternatively, the serializer may need to assign a specific background pattern to quiet individual devices' internal logic, and to isolate the cluster during testing by breaking onboard feedback loops or disabling circuitry surrounding the cluster. This background pattern can be the same for each test pattern applied to the cluster.

So, while as little as two or three percent of the hundreds or thousands of bit positions in the Boundary Scan path may contain bits which are relevant for the test being performed, any method used to serialize test data still must fill the entire path. This means supplying not only the test stimuli to the virtual channels, but also the background pattern carried by the other shift-register cells on the path. And analysis of the cluster's response must distinguish which bits — out of all the bits clocked out of the path — are meaningful for the test results.

In addition to the problems associated with serialization of the cluster patterns for go/no-go testing, automated diagnosis of cluster failures is important. The techniques of guided probing in a scan environment have been used for some time on boards with devices that employ internal scan techniques (such as LSSD) [7], and this approach is applicable to cluster testing with Boundary Scan. Additionally, fault dictionary diagnostic techniques have the advantage of eliminating manual probing [8] — or complimenting it by minimizing probing[9]. In either case, virtual outputs are the basis for diagnostic input instead of physical channels associated with conventional test.

SERIALIZING TEST PATTERNS

Serializing and delivering functional cluster test patterns into the Boundary Scan path require two scanning operations: one for stimulus and one for response. This sequence represents one "scan vector," which corresponds to one pattern of the cluster test. The process of applying stimulus and observing response is actually overlayed; as the stimulus for vector n is shifted into the Boundary Scan path, the response for vector n-1 is shifted out.

The serializer therefore must consider stimulus and response as two separate — but concurrently executed — operations. First, the stimulus portion of a cluster pattern must be combined with data about the topology of the Boundary Scan path. This identifies which shift-register cells on the path serve as Virtual Primary Inputs. While the inputs to the cluster can be thought of as being quite logical and orderly in nature, they also can be expected to be somewhat random in their location in the scan path. The serializer must further provide the background pattern for the remaining positions on the Boundary Scan path.

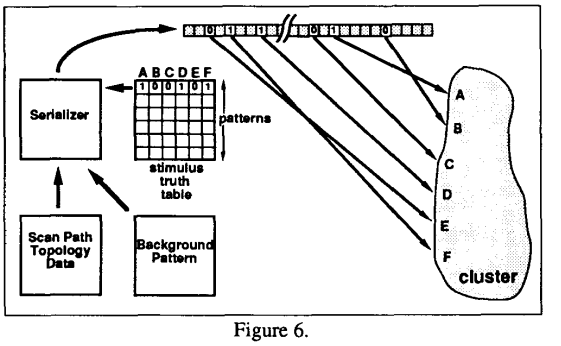

Figure 6.

The resulting (lengthy) bit stream will be fed into the Test Data Input (TDI) pin of the first IC in the Boundary Scan path. The serializer must also specify the state of the Test Mode Select (TMS) pin for each cycle of the Test Clock (TCK).

On the response side, the serializer can ignore all of the bits shifted out of the Test Data Output (TDO) except for the Virtual Primary Ouput bit positions determined from the Boundary Scan path topology data. Since a Virtual Primary Output can be in an unknown state, more than one bit of data is required as input to the test result logic in the ATE system (allowing 1, 0, and X states).

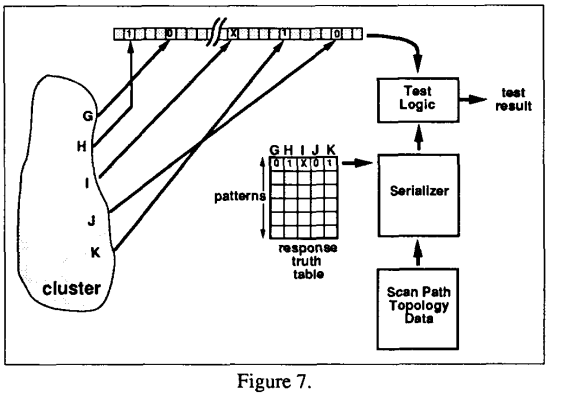

Figure 7.

For the combination of stimulus and response, then, the serializer must provide — on each clock cycle — the following bits of data:

- one bit of stimulus data for TDI
- one bit of stimulus data for TMS
- two bits of data for the test hardware to specify 1, 0, or X at TDO

Prior to serialization the cluster vectors, topological data, and background pattern can be stored in an efficient, compact manner. The Boundary Scan path topology data is fairly small, and is applied to each scan vector used to test the cluster. The same is true of background patterns. The cluster-test patterns can be stored just as they would be to drive standard ATE channels, which might be in the form of a truth table or as incremental state changes (delta encoding) [7] if that affords more efficient use of storage space. After serialization the lengthy bit stream that results is very inefficient because so few of the bit positions represent virtual channels for the cluster test.

Several strategies may be used to implement pattern serialization:

- run-time software serialization
- one-time serialization with run-time DMA test
- one-time serialization with large scan buffer
- run-time hardware serialization

Run-Time Software Serialization

The serialization process could take place in software as each board is tested, an approach that has the advantage of minimizing data-storage requirements.

Using a truth table to store the cluster data requires one bit (either 1 or 0) per pattern for each Virtual Primary Input, and two bits per pattern (to specify 1, 0, or X) for each Virtual Primary Output. If there were 40 Virtual Primary Inputs and 30 Virtual Primary Outputs, the storage requirements for 10,000 patterns would be:

(# inputs • 1 bit/input + # outputs • 2 bits/output) • (cluster patterns)

$= (40 \cdot 1 + 30 \cdot 2) \cdot (10{,}000)$

$= 10^6$ bits

The size of the background pattern would be an inconsequential 10^3 bits and the topology data for this example can be stored in such a way as to be likewise negligible. Thus, the storage requirements are much the same as they would be on ATE that had direct access to the cluster with conventional channels. The problem with this software approach is that run time on a general-purpose computer is prohibitively long, typically being 2-4 orders of magnitude slower than the bandwidth of the Boundary Scan path.

One-Time Serialization with Run-Time DMA

An alternative solution would be to do a one-time serialization, then apply the serial stream via special DMA hardware to each new board placed on the tester; this approach, however, has none of the storage efficiencies of the previous software method. Assume the Boundary Scan path is 1,000 clocks long and that four bits of data are required for each clock cycle:

(4 bits/clock) • (1,000 clocks) • (10,000 patterns)

$= 4 \times 10^7$ bits

This is 40 times as many bits as the run-time serialization approach. Even with a fast DMA controller, loading this amount of data from disk to the ATE hardware is quite time-consuming, and inefficient data storage is expensive.

One-Time Serialization with Large Scan Buffer

A common tactic is to store the data in a large scan buffer directly accessible to the ATE channels servicing the pins which control the Boundary Scan test [10]. Once loaded, many boards can be tested at scan rates of up to the limitations of the ATE or of the Boundary Scan path. However, this approach depends on two things: (a) that the buffer can hold an entire board program; and (b) that many boards will be tested.

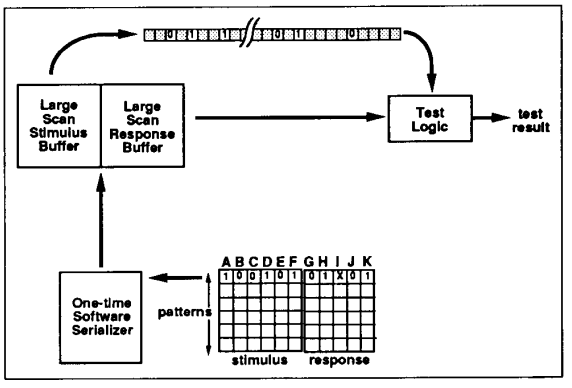

Figure 8.

If either of these assumptions proves false, the large scan buffer offers no speed benefits over direct DMA — and it suffers from the disadvantage of inefficient data storage. If a board had 10 clusters of conventional logic like the one in our example above, the large scan buffer would require one million locations, four bits wide.

In addition, the application example that follows shows that algorithmic generation of test patterns could easily produce even longer serial data sequences than the ones we've been considering. Clearly, it is difficult to predict the size needed for a large scan buffer — and creating one large enough to accommodate all contingencies is usually impractical.

Moreover, just-in-time (JIT) manufacturing lines tend to send very small lot sizes of any given board type to a tester. This increasingly common situation eliminates the second requirement underlying the feasibility of using large scan buffers for test pattern serialization.

Run-Time Hardware Serialization

The solution to both the test time and data-storage issues is a hardware serializer. With this special hardware, or serial pattern processor, all the mixing of cluster patterns, background patterns, and topological data takes place for both stimulus and response in real time as each board is tested.

Additionally, to maximize throughput, it is important that this hardware serializer be able to keep up with Boundary Scan shift rates that will typically be in the 10-20 MHz range. This serial pattern processor approach has the advantage of data compression via Topological Data Compression, and high throughput via serialization in hardware.

Data Storage and Test Time Summary

The following table summarizes the amount of data required and test times for the four types of serializers described above to test random logic clusters.

Assumptions:

- Board has ten clusters of the size presented in the previous exampl
- Total serialized data size is (10) • ($4x10^7$ bits) = 4×10^8 bits = 50 Mbytes
- Pre-serialized data size (compacted) = (10) • (10^6 bits) = 10^7 bits = 1.25 Mbytes
- Overall DMA rate from disk to ATE (for direct test or reload of large scan buffer) = 10 Mbits/second
- Maximum scan clock rate = 20 MHz
- Software serializer rate = 100 KHz

Serialization Method	Data Storage Requirements	Test Time
Run-Time Software	1.25 Mbytes	1000 sec.
1 x Serialization with Run-Time DMA	50 Mbytes	40 sec.
1 x Serialization with Large Scan Buffer	50 Mbytes	5 sec.* or 45 sec.
Run-Time Hardware (Serial Pattern Processor)	1.25 Mbytes	5 sec.

*Only if Large Scan Buffer can hold entire board test program of 400 Mbits

SERIALIZATION OF ALGORITHMIC PATTERNS

The hardware serializer described above receives the tests for the cluster as a linear set of patterns, or a truth table, which typically come from logic simulation for random logic clusters. Memory devices or arrays are more efficiently tested using algorithmically generated patterns.

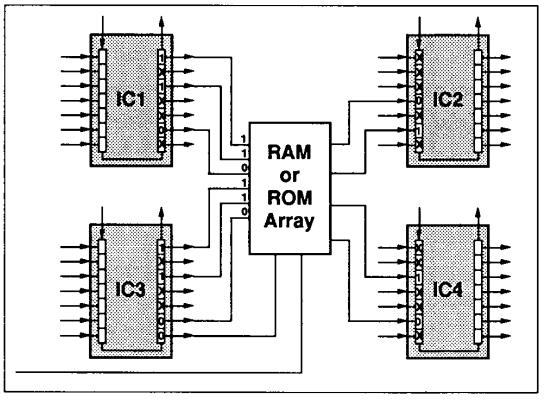

Figure 9.

The algorithms applied to the memory cluster can be described in a terse manner, rather than by very long truth tables. Algorithmically created values are employed by the serializer in place of the truth table values used for testing random-logic clusters.

At the board level it is usually sufficient to test each address to check that each contains the prover contents (ROM) or that it can hold both 1 and 0 states (RAM) independently from all other addresses.

ROM testing is achieved by sequencing through every address and applying the outputs to a Multiple Input Shift Register (MISR). In this mode, the serializer works in reverse from the logic-cluster example. Instead of comparing the cluster outputs to expected responses, the actual response values at the various Virtual Primary Outputs in the Boundary Scan path are applied to the MISR after being extracted from the output bit stream.

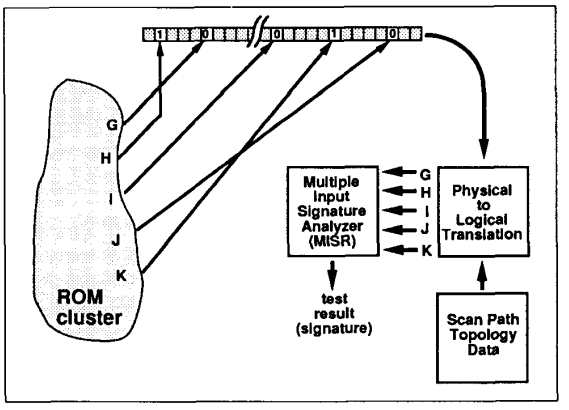

Figure 10.

RAM tests are typically performed by a MARCH routine which actually accesses each location six times, in a sequence that alternates between ascending and descending address sequences, and alternately writing (and then reading) a data pattern or its complement. Here the serializer creates all stimulus and response values, and testing takes place in hardware as in the logic-cluster mode.

APPLICATIONS EXAMPLES

1. Typical random logic cluster

   Test Source for cluster is simulation

   Components:
   - 1-2 LSI
   - 10-20 SSI/MSI
   - 3-5 sequential PLDs
   - 40 input pins
   - 30 output pins
   - >99% fault coverage for device pins SA0, SA1, open inputs
   - 2,500 patterns with data formatting if cluster could have been accessed with ATE channels, 10,000 patterns with unformatted "virtual channels"

   A) Approximate Data Storage Requirements (compacted)

   Topological data 1K bytes
   Background Pattern Data 1,000 bits (125 bytes)
   Cluster patterns:
   (# inputs • 1 bit/cluster pattern +
   # outputs • 2 bits/cluster pattern)
   • cluster patterns)
   = (40 • 1 + 30 • 2) • 10,000
   = 100 • 10,000
   = 10^6 bits = 125,000 bytes

   B) Total Shift Cycles

   = scan path length • cluster patterns
   = 1,000 • 10,000
   = 10^7 shift cycles

   C) Test Time for Hardware Serializer

   = total shift cycles/max. shift rate
   = 10^7/20 MHz = 0.5 seconds

   D) Size of serialized output data (size required for a large scan buffer)

   = test bits/cycle • total shift cycles
   = 4 x 10^7 bits = 5 Mbytes

   E) Compression Factor

   $$= \frac{\text{serializer output data size}}{\text{serializer input data size}}$$

   $$= \frac{5\text{ Mbytes}}{\approx 125\text{K bytes}} = 40$$

2. ROM Cluster Test

   Test source for cluster is CAE database or learning by the serializer

   256K x 16 bit ROM array
   18 address inputs
   1 output enable input
   16 data outputs
   100% test of contents
   256K patterns with exhaustive pseudo-random or counting pattern on address inputs and multiple input signature analysis on data outputs
   boundary scan path length = 250

   A) Approximate Data Storage Requirements

   Topological Data 1K bytes
   Background Data Pattern 250 bits (32 bytes)
   Cluster Patterns
   (algorithmic description language) ≈100 bytes

   B) Serialized bit stream length

   = scan path length • cluster patterns
   = 250 • 256 x 10^3
   = 6.4 x 10^7

   C) Test Time

   = serialized bit stream length/max. shift rate
   = 6.4 x 10^7/20 x 10^6
   = 3.2 seconds

D) Size of serialized output data
(size required for a large scan buffer)

= serialized bit stream length • 4 bits
= 6.4 x 10^7 • 4
= 256 x 10^6 bits = 32 Mbytes

E) Compression Factor

= 32 Mbytes/ ≈ 1000 bites ≈ 32,000

CONCLUSION

The test of embedded clusters of conventional logic via Boundary Scan virtual channels is a practical way to test and diagnose structural faults where these clusters are inaccessible to standard ATE channels. Huge quantities of data can be created by this technique, which could result in prohibitive storage requirements and poor throughput. Using Topological Data Compression and special hardware, the storage requirement can be small and test times limited only by speed of the Boundary Scan path.

REFERENCES

[1] F.P.M. Beenker, "Systematic and Structured Methods for Digital Board Testing," *IEEE International Test Conference Proceedings*, paper 11.1, p. 380, 1985.

[2] P.T. Wegner, "Interconnect Testing with Boundary Scan," *IEEE International Test Conference Proceedings*, paper 2.2, p. 52, 1987.

[3] Texas Instruments Inc., *Product Preview*, SN54BCT8244, SN74BCT8244 Scan Test Device with Octal Buffer, 1989.

[4] P. Hansen, "The Impact of Boundary Scan on Board Test Strategies," *ATE & Instrumentation Conference East Proceedings*, paper DT-2, June, 1989.

[5] S. Caplow, "Cluster Testing Overcomes Many Testability Problems," *EDN*, October 1987.

[6] P. Hansen, "Converting Device Test Vectors to an In-Circuit Board Test Environment," *IEEE International Test Conference Proceedings*, paper 25.2B, p. 972, 1985.

[7] P. Hansen, "New Techniques for Manufacturing Test and Diagnosis of LSSD Boards," *IEEE International Test Conference Proceedings*, paper 3.1, p. 40, 1983.

[8] J. Richman and K.R. Bowden, "The Modern Fault Dictionary," *IEEE International Test Conference Proceedings*, paper 19.2, p. 696, 1985.

[9] V. Ratford and P. Keating, "Integrating Guided Probe and Fault Dictionary: An Enhanced Diagnostic Approach," *IEEE International Test Conference Proceedings*, paper 10.3, p. 304, 1986.

[10] M.L. Fichtenbaum, "A Scan-Device Test Approach for the Digital Board Tester," *ATE Instrumentation Conference East Proceedings*, paper MT-7.1, p. 441, 1988.