A Distributed BIST Control Scheme for Complex VLSI Devices

Yervant Zorian

AT&T Bell Laboratories
Princeton, NJ, 08540

Abstract

BIST is a viable approach to test today's digital systems. Constraints, such as power, noise, area overhead, and others, limit the possibilities of parallel BIST execution in complex VLSI devices. This paper presents a BIST scheduling process that takes into consideration such constraints, and introduces a new BIST control methodology, that implements the BIST schedule with a highly modular architecture. In fact, due to the uniformity of interface, the BIST control elements are independent of the BIST scheme used in the embedded blocks of a device. This BIST control architecture can provide block level diagnostic information.

Keywords: Built-In Self-Test, Test Scheduling

1: Introduction

With the ever increasing need for system integration, the trend today is to include in the same VLSI device a large number of functional blocks, and to package such devices, often, in Multi-Chip Modules (MCMs) that comprise complex systems. This leads to difficult testing problems in the manufacturing process and in the field [6]. An attractive approach to solve these problems is to use a multi-level integrated Built-In Self-Test (BIST) strategy [12][20][27]. This strategy assumes that BIST is used at each level of manufacturing test, and it is reused at all consecutive levels, i.e. device, MCM, board, system. It also assumes using the IEEE 1149.1 Boundary-Scan standard [11] to realize self-testing at different levels. This strategy can only be realized if the appropriate BIST features are included in devices. Devices are, in fact, the building block around which the higher level BIST approaches are built [20][27]. Therefore, a device level BIST realization must take into account the BIST requirements of all levels, device, MCM, board, etc. These requirements often affect the BIST execution schedule of a device, and hence its BIST Control Network which implements the schedule. The concentration of this paper will be on a new device level BIST scheduling process and its corresponding BIST control architecture.

The existing BIST scheduling approaches focus, mainly, on optimizing the test time for random logic data path blocks [2][7][18]. The fact that the number of embedded blocks in today's devices is in continuous growth, and also that these devices may be packaged in very high density modules (MCMs), create concerns regarding the possibility of executing BIST in parallel on all the BISTed blocks in a device, and/or module under test. The major cause of this concern is the power and noise dissipation impact during BIST execution. Because, BIST, typically, results in considerably higher circuit activity rate, compared to normal mode operation, hence, causing above normal power dissipation. This may overpass the device or MCM package limits if not planned properly. To address this problem, a new BIST scheduling process is introduced. This takes into account the power dissipation of each BISTed block, and performs global optimization considering also other factors, such as block type, device floor plan, and test time. This process, as will be described, provides a device level BIST execution schedule to be represented in a BIST sequencing profile. This profile identifies the sequence of BIST execution stages and demonstrates the BISTed blocks to be activated in parallel at each stage. The sequencing profile, which is in a matrix format, is the basis for automatically generating a BIST control network, as described later.

In our BIST control scheme, a self-contained BIST approach is adopted to simplify the BIST execution of a device especially during higher levels of test. The BIST execution is activated simply by a single RUNBIST instruction, as defined in [11][9]. By the same token, the BIST execution of each embedded block is also made autonomous, and hence, allowing a simple and uniform interface protocol to be used for communication between the device BIST controller and the embedded blocks. Another important aspect about the new device BIST controller is its distributed architecture. This is to minimize the routing cost of control signals. The control elements, that comprise the distributed architecture, will be placed close the BIST blocks their control, and be concatenated by a minimum number of control signals. These control elements, called Scheduled BIST Resource Interface Controllers (SBRICs), perform two major roles in parallel: device BIST control and test response collection [19] at each BIST execution stage. Hence, at the end of device BIST, the SBRIC network will contain a concatenated BIST signature to be shifted out via the Boundary-Scan TAP [11].

The rest of this paper is organized as follows: Section 2 introduces a device level BIST scheduling process, and a consequent BIST sequencing profile. Section 3 illustrates the new BIST control architecture and its link to individual BISTed blocks. Section 4 describes the distributed BIST control network and the SBRIC elements. Finally, Section 5 provides a summary and some concluding remarks.

2: BIST Scheduling Process

BIST scheduling is an analysis process, during which no hardware modification is performed to a device. This section addresses the scheduling process. Two other analysis processes are performed prior to scheduling. They are device partitioning into its basic building blocks and BIST scheme selection. These two processes are out of the scope of this paper. Hence, they will only be mentioned briefly in an example. In order to help illustrate these and other processes a sample device, called ASIC Z, is used throughout this paper. As shown in Figure (1), ASIC Z, which is a 132 pin standard cell device, is partitioned into ten blocks. The partitioning takes into consideration different criterion [3]: the structural types of blocks, such as in cases of four RAMs, two ROMs, and Register File; circuit hierarchy, as in the case of the two random logic blocks RL1 and RL2, and clock domains, as in the case of the high frequency block RL3. Each block requires a Test Pattern Generator (TPG), an Output Data Evaluator (ODE) and a dedicated controller to coordinate the BIST operation of that block, namely a BIST Resource Controller (BRC). In addition to these, there is a need for isolation between BISTed blocks during BIST mode. This can be realized by adding some type of isolation nodes [3][4][9].

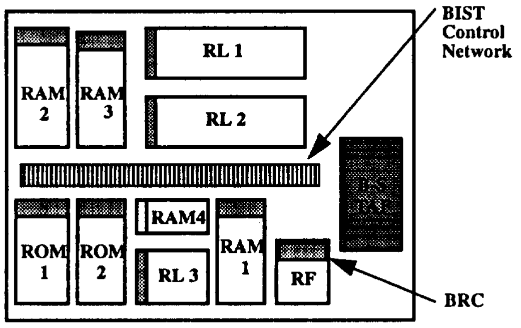

Figure (1) Partitioning and BIST Control of ASIC Z

Example: In ASIC Z, a pseudo-random based TPG (ex. LFSR) and a polynomial division based ODE (ex. MISR) is selected for both random logic blocks, RL1 and RL2. In the case of the Register File and the four RAMs deterministic TPGs are selected with 100% fault coverage of extended fault models [1][23]. The selected ODE for the Register File performs Programmable Space Compaction (PSC) [25] to provide zero loss of fault coverage information, whereas the RAM ODE performs direct output data comparison followed by space compaction [1]. The TPGs selected for the two ROMs provide exhaustive test sets and their ODEs perform an output data specific compaction as in [26]. In ASIC Z, the selected BIST facilities for the RAMs, Register File, and ROMs are automatically synthesized with their corresponding blocks using [10].

BIST scheduling has been a topic of interest for a while. However, the proposed techniques address block level, and not overall device level scheduling issues, such as scheduling for test time optimization by using parallelism [18], or scheduling for area overhead optimization by sharing BIST resources (ex. BILBOs) in data path blocks [2] [7]. These techniques are certainly valid on the block level, and can be implemented in BRCs where appropriate. But due to the power and noise dissipation impact during BIST execution, one can not assume operating BIST in parallel in all the BISTed blocks of a device, as it is often assumed [22] if a device size is not large. The device level BIST scheduling, presented in this section, analyzes a number of attributes to come up with a BIST sequencing profile. The attributes that have an impact on device level BIST scheduling are: power dissipation of each block; adjacency of blocks; type of each block; test time. In the following a BIST scheduling process that considers these attributes is described.

2.1 Power Dissipation Analysis:

If the BIST execution of a block is performed at a reduced clock rate, and not system speed, then the higher activity rate during BIST mode may not cause excess of power dissipation. But, in order to obtain its full benefit, BIST is usually executed at system clock rate. Power dissipation in random logic blocks is a function of frequency F, block size (number of grids) N, number of watts per active grid PG, and the activity rate [10], more specifically P=F.N.PG. The difference between the BIST and normal modes, given a random logic block, is in the activity rate, which is, for instance, assumed 0.5 in case of pseudo-random based BIST schemes, and often less for normal mode operation. In order to obtain a realistic P for a random logic block, a predetermined percentage (ex. 10%) needs to be added to the block size N, corresponding to the average area overhead for the selected BIST scheme. On the other hand, the power dissipation values for different regular structure blocks are generally given in parameterized formulas [10]. The power value in such blocks is a function of the block parameters (i.e. number of words, and number of bits per word) [10]. There are, usually, separate formulas for normal, BIST active, and BIST idle modes. The BIST active formula provides the power value during the BIST execution of a given block. Whereas the BIST idle formula provides the power dissipation value of that same block during the periods when it waits for other blocks to execute BIST. The total power dissipation of a device during normal or BIST mode consists of the sum of the power dissipation values of its individual blocks, added to the power values of its input and output buffers where applicable. If the total power dissipation of

BIST mode is higher then the one for normal mode, then naturally the BIST execution needs to be performed in multiple stages.

Example: The sizes of blocks in ASIC Z, and their power dissipation values, in mW's, are shown in Table (1). The parameterized BIST active and BIST idle formulas from [10] are used for the cases of RAMs, ROMs and Register Files (RF). The package limit of ASIC Z is 900mW. and based on the numbers shown in Table (1) the device power dissipation, if parallel BIST execution is performed, is 2.332 W. Hence, BIST scheduling for power optimization is necessary.

Table 1:

	Size	$P_{B/active}$	$P_{B/idle}$
RL1	13400g	295	
RL2	16000g	352	
RF	64X17	95	19
RAM1	768X9	282	20
RAM2	768X8	241	17
RAM3	768X5	213	11
RAM4	768X3	96	7
ROM1	1024X10	279	23
ROM2	1024X10	279	23

2.2 Grouping and Ordering:

If the BIST execution of a device is to be performed in multiple stages, then creating groups of blocks to be executed in each stage and proper sequencing of such stages is necessary. A criteria that contributes to such grouping and sequencing is the physical distribution of the blocks on a floor plan of a device. Leveraging such a distribution optimizes the hardware realization of the BIST Control Network.

A group of blocks to execute BIST in a single stage, i.e. in parallel, is created based on two conditions: The the BIST power dissipation of a group has to be less then the device limit; And the blocks in a group have to be physically adjacent on a floor plan. Grouping such blocks results in reducing the distribution of control lines, that connect these blocks.

Example: ASIC Z is divided to four groups, each of which meeting the above two conditions. Based on the physical distribution of blocks, shown in Figure (1), the four groups are: RL1, RL2; RAM1, RAM4, RF; RAM2, RAM3; and ROM1, ROM2.

The Sequence of BIST execution of multiple groups is also based on the floor plan information. The aim, in this case, is to minimizes the distances between consecutive group level control elements, as shown in section 4. As a result to this grouping and sequencing step a device level BIST Sequencing Profile, S, is produced. This Profile can be represented in a matrix format:

$$S = \begin{bmatrix} B_{1,1}, B_{1,2}, \ldots\ B_{1,m} \\ \ldots\ldots\ldots\ldots\ B_{i,j} \ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ B_{n,1}, B_{n,2}, \ldots\ldots \end{bmatrix}$$

where each matrix element $B_{i,j}$ represents a block in the device; each row i represents a single BIST execution stage; j represents the number of a block in a given stage; and finally, the order of the rows represents the sequence of BIST execution stages.

Example: Based on the physical distribution shown in Figure (1), the BIST Sequencing Profile of ASIC Z will be:

$$S = \begin{bmatrix} \text{RAM1, RAM4, RF} \\ \text{RL1, RL2} \\ \text{RAM2, RAM3} \\ \text{ROM1, ROM2} \end{bmatrix}$$

If the floor plan of a device is not final, then the sharing analysis, as shown next, can be performed prior to grouping and ordering. The consequent group distribution can be used in determining the final device floor plan.

2.3 Sharing and Test Time Optimization:

The objective of sharing is to further optimize the BIST hardware area. This is achieved by reducing redundancies of BIST facilities. In general, sharing BIST facilities is possible in groups of blocks and even between groups. The simplest possible sharing is between blocks of the same type, e.g. RAMs, random logic blocks, since their BRCs are usually identical, and their TPGs and ODEs are only different in their parameters. Another cost effective sharing can be obtained by merging the Boundary-Scan Register [11] with pseudo-random TPGs (ex. LFSR [13] or CAR [8]) on one hand; and with polynomial division based ODEs (ex. MISR [13] or CAR [8] on the other.

The test time needed to execute the BIST operation of a device may be limited by the system level test requirements, where a limited amount of time may be dedicated per device. Such requirements influence the BIST scheduling process. On the other hand, the BIST execution, if run at system speed, is, usually, only a minor fraction of the total test time during device manufacturing test. Because, other tests, such as parametric [3] and read/write memory hold time tests, are far more time consuming than the BIST operation. Hence, optimizing for BIST execution time may not be very crucial in general.

3: The BIST Control Architecture

The BIST control operation of a device, generally, consists of four major functions: External access to device level BIST; Control of BIST facilities of each block; Control of device BIST, which also contains the sequencing operation; and collection and transfer of BIST response data. In our BIST control architecture, these functions are realized in three types of hierarchical controllers. These controllers, as shown in the shaded areas of Figure (1), are composed of:

3.1: External BIST Access Port

Various control schemes, including the one presented in this paper, use the Boundary Scan TAP [11] as the external access port for device BIST [8][13][9][16][22]. Some of these control schemes tend to combine the above control functions in one controller [16] [22], and suggest using the TAP finite state machine [11] and a number of Boundary-Scan instructions to execute device BIST [16][22]. This may be possible in simple test operations. However, for more complex test procedures separate controllers are required for each one of the BIST control functions. Since, a multi-level integrated BIST approach, as described in section 1, requires minimum external involvement in the details of device level BIST execution, a single RUNBIST is used [11][9]. This results in an autonomous, i.e. self-contained, device level BIST operation. Boundary-Scan [11] requires a predefined and fixed duration of the RUNBIST instruction, which can be obtained from adding test lengths of the BIST stages. The BIST execution as well as the BIST response data transfer are performed via the Boundary-Scan TAP. There will be no additional pins dedicated for external BIST access.

3.2 BIST Resource Controllers (BRCs)

Since the number of blocks per device is in a continuous growth, and since the BIST schemes of each type of block are often different, then individual BRCs have to be allocated to each block [4] [9], such that the BRC performs the BIST operation autonomously. BRCs are generally customized finite state machines for a given TPG and ODE. The complexity of the state machine is a function of the BIST scheme used for a given block. If a BRC controls a multiple clock block, it often performs the clock unification function.

Techniques to merge BRCs for different random logic schemes are reported [5][4][7], and ones to merge BRCs with functional controllers [16]. Sharing of non-identical random logic BRCs is possible, because of their low complexity. A similar sharing for non-identical BRCs in regular structure blocks may not be cost effective. Because these BRCs usually contain much larger and hence complex finite state machines to generate deterministic BIST algorithms [1] [24] [23].

3.3 BIST Control Network

The third control block, namely the BIST Control Network, device BIST coordination, is performed by a dedicated control block called the. This network is basically represents the BIST Sequencing Profile obtained in section 2. Hence, it executes the device BIST operation based on a predetermined schedule. The BIST Control Network receives the device BIST execution order via TAP, and provides the proper activation signals to the sets of BRCs in controls, based on a predetermined schedule.

Contrary to the existing control schemes [3][9][16], the BIST Control Network performs an additional function. That is the collection and transfer of BIST response data via TAP, upon device BIST completion. In most of the existing schemes a separate network is routed throughout the device to collect the response data.

One of the advantages of this control architecture is the structured division of control functions, where each function contains an autonomous capability to control subsequent BIST operation. This results in a uniformity and simplicity of interfaces between the three control levels. For instance, a BRC executes BIST in a given block, and hence, has a simple interface with the BIST Control Network. This interface is uniform for all BRCs irrespective to the type of block it controls, i.e. RAM, ROM, random logic. Similarly, the BIST Control Network is autonomous, since it contains all the scheduling information and also has a simple interface with the TAP.

The uniform interface protocol, between the BIST Control Network and the BRCs connected to, is uniquely specified and consists of four signals [19]. Two input signals, namely BIST and BFC, and two output signals, BC and BF. The BIST signal, sent by a BIST Control Network to a BRC, initiates the BIST session of that block. The BRC in its turn starts executing the BIST operation. When the BIST operation is completed, BC (BIST Complete) will inform the BIST Control Network. BF (BIST Flag) might have gone high by this time if any faults are detected. BF low indicates a fault free block. If BF is low, the BIST Control Network sends BFC (BIST Function Control) signal, which will cause BF to toggle, and hence to check for stuck-at fault BF line.

The above interface protocol uses BF as a single bit BIST response. Hence, the ODE of each block presents its final signature in a single bit. BIST schemes with single bit ODEs have been reported for various types of blocks, example RAM [1], ROM [26], CAM [24], FIFO [21] and Register File [23]. For BIST schemes using polynomial division based ODEs (ex.MISRs), such as the ones often used with random logic blocks [17], it is possible to obtain a single bit signature by using a prediction algorithm [26]. This calculates the initial seed of a polynomial divider to provide a predetermined final seed, such as an all zero string [15]. The seed gets accumulated simply in a BIST flag as in [26]. **Example:** All BRCs in ASIC Z use the uniform interface protocol to communicate with the BIST Control Network. The selected BIST schemes for each block [10] use BRCs with the above four control lines.

4: A Distributed BIST Control Network

Most existing control schemes propose using centralized controllers to perform device level BIST control [4] [9]. But, the problem with a centralized controller is the routing area of the control lines [16] that connects the device controller to local BRCs. In general, the number of such control lines increases linearly with the number of BRCs. For devices with small number of blocks, a centralized BIST controller is possible. However, for complex devices where the size of the device and/or the number of embedded blocks is large a distributed one is suggested.

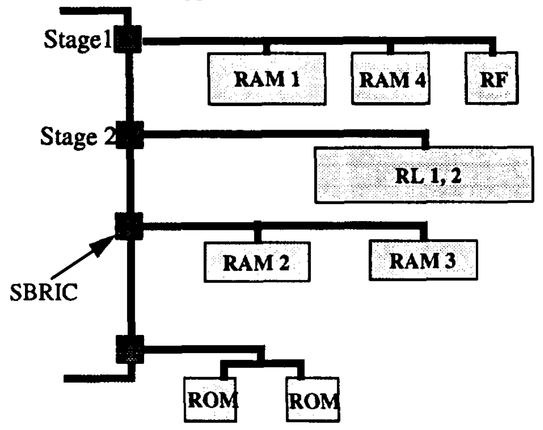

Figure (2) Distributed BIST Control Network ASIC Z

The BIST Control Network, introduced here, proposes using a distributed control architecture, which optimizes the routing cost of such control lines. Centralized controllers can often be cost effective for local control functions [3], but higher level controllers in a hierarchy, usually, require distributed control.

Example: Figure (2) shows the distributed BIST Control Network of ASIC Z. This is a hardware realization of the BIST sequencing profile developed in Section 2. Each BIST execution stage is controlled by one control element.

The distributed control architecture will be composed of a set of control elements, to be called Scheduled BIST Resource Interface Controllers (SBRICs). These elements will be concatenated in a modular manner.

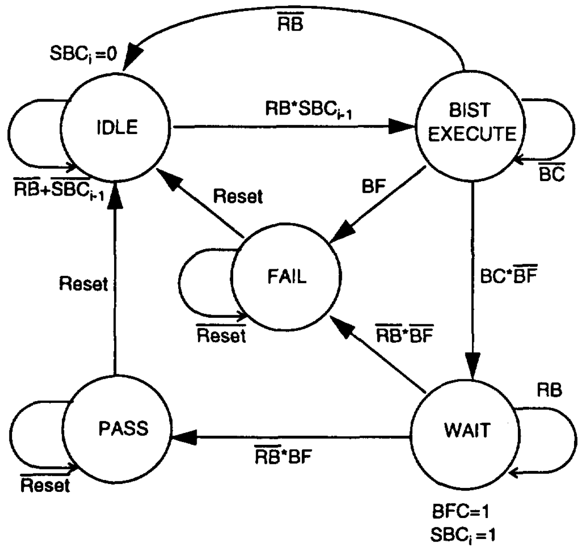

Figure (3) SBRIC Finite State Machine

An SBRIC is a customized finite state machine which contains five states, as shown in Figure (3). It provides BRC coordination along with test response collection and transfer capabilities. An SBRIC communicates with its associated BRCs using the uniform interface protocol. An $SBRIC_i$ is set to IDLE state by the reset input. Upon receiving the RB (run BIST) signal and the SBC_{i-1} (previous SBRIC BIST execution completed) signal, as shown in Figure (3), the BIST output goes high to all BRCs under its control, and $SBRIC_i$ proceeds to BIST Execution state. During this state if any one of the BFs goes high the $SBRIC_i$ moves to FAIL state. However, if no faults are detected and all BC signals go high (i.e. all blocks of that stage have BIST completed), then $SBRIC_i$ proceeds to WAIT state. At this state $SBRIC_i$ generates the BFC signal to check the stuck-at faults of all BFs, and also generates the SBC_i signal to allow the next one, $SBRIC_{i+1}$, to start its BIST execution. $SBRIC_i$ stays in WAIT state until the run BIST of the device is over, and then it moves to PASS state if all its BFs are high, and to FAIL state if not. The final state of $SBRIC_i$ is its signature, which is shifted out on TDO via the TAP, as in other schemes [20][22]. In fact, the device signature obtained contain the states of all SBRICs, because, as shown in Figure (2), all SBRICs are concatenated, hence the BIST status of each stage can be identified.

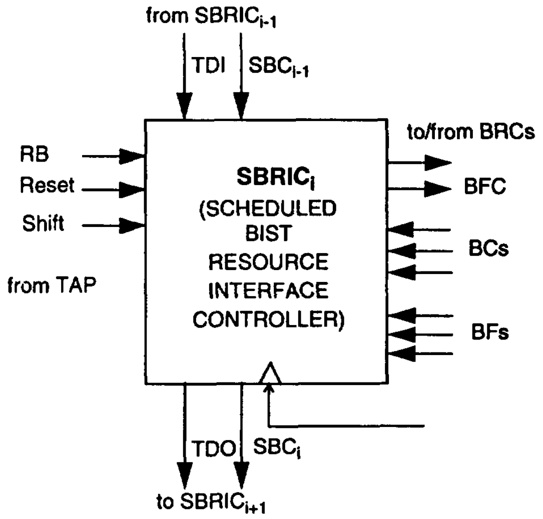

Figure (4) SBRIC Architecture

Some BIST control approaches propose accumulating the signatures of each BISTed block in a common output data compactor, as in the signature analyzer of [22]. This causes loss of diagnostic information obtained through the individual signatures of BISTed blocks.

To obtain additional diagnostic capability, one can use an enhanced SBRIC design that does not merge the BFs and hence provides up to block, and not only group, level diagnostic information.

The generation of an SBRIC element needs to be parameterizable, since it communicates with different number of BRCs. SBRICs are predesigned library elements of [10]. Their automatic generation ability makes their use very attractive and almost transparent to designers. The SBRIC network can simply be connected to the TAP, with no additional pin-out. However, this architecture is a generic one and can be used without Boundary-Scan if needed.

5: Conclusions

This paper presented a generic BIST scheduling process and an effective BIST control architecture. The scheduling process provides optimization for power dissipation in addition to other common constraints. The control architecture, on the other hand, provides an autonomous BIST activation and a diagnostic capability to identify failed blocks. Future extensions to the above BIST control scheme may include the development of scheduling processes that take into account multi-level system test restrictions; and the development of new BIST control networks that have additional diagnostic capabilities to be used for reconfiguration and repair option.

Acknowledgments:

The author would like to thank I. Kim, K. Kinney, P. Rutkowski, H. N. Scholz, R.E. Tullos, S. Wu, and C.W. Yau, for their fruitful discussions.

References

[1] Aadsen, D.R., and Jain, S.K., "Automation of BIST for Embedded RAM", Proc. IEEE Custom Integrated Circuits Conference (CICC), pp. 66-69, May 1987.

[2] Abadir, M.S., and Breuer, M., "Test Schedules for VLSI Circuits Having Built-In Test Hardware", IEEE Trans. on Computers, pp. 361-368, April 1986.

[3] Beenker, F.P.M., Eerdewijk, K.J.E., Gerritsen, R.B.W., Peacock, F.N. and van der Star, M., "Macro Testing: Unifying IC and Board Test", IEEE Design & Test of Computers, pp. 26-32, December 1986.

[4] Beenker, F., Dekker, R. and Stans, R., "Implementing Macro Test In Silicon Compiler Design", IEEE Design & Test of Computers, pp. 41-51, April 1990.

[5] Breuer, M., Gupta, R., and Lien, J-C, "Concurrent Control of Multiple BIT Structures", Proc. International Test Conference (ITC), pp. 431-442, 1988.

[6] Campbell, R.L., "Creating Wealth - Through Testing?", chapter in Economics of Design and Test, edited by Ambler et al., pp.28-35, Ellis Horwood, 1992.

[7] Eschermann, B. and Wundrelich, H-J., "Parallel Self-Test and the Synthesis of Control Units", Proc. of IEEE European Test Conference (ETC), pp. 73-82, Munich, April 1991.

[8] Gloster, C.S., and Brglez, F., "Boundary-Scan with Cellular-based Built-In Self-Test", Proc. International Test Conference (ITC), pp. 138-145, 1988.

[9] Haberl, O.F., Kropf, T., "HIST: A Methodology for the Automatic Insertion of a Hierarchical Self Test", Proc. of IEEE Int'l Test Conference (ITC), pp. 732-741, 1992.

[10] "High-Speed HS900C CMOS Standard-Cell Library", AT&T Microelectronics, January 1992.

[11] IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE Std. 1149.1-1990, IEEE Standards Office, NJ, May 1990.

[12] Jarwala, N. et al., "A Framework for Boundary-Scan Based System Test and Diagnosis" IEEE Int'l Test Conference, pp. 993-998, 1992.

[13] Maierhofer, J. "Hierarchical Self-Test Concept based on the JTAG Standard", Proc. of IEEE Int'l Test Conference (ITC), pp. 127-134, 1990.

[14] Marinissen, E.J., and Dekker, R., "Minimization of Test Control Blocks", Proc. European Test Conference, pp. 427-436, 1991.

[15] McAnney, W.H., and Savir, J., "Built-In Checking of the Correct Self-Test Signature", IEEE Trans. on Computers, Vol. 37, No. 9, pp. 1142-1145, Sept. 1988.

[16] Mukherjee, D., Njinda, C., and Breuer, M., "A Partially Distributed Control Scheme for DFT/BIST Hardware", Proc. of WESCON, Nov. 1992.

[17] Rutkowski, R.W. and Lin, C.J., "Two CAD Tools for Random Logic BIST", Porc. GOMAC, Nov. 1990.

[18] Sayah, J.Y., and Kime, C.R., "Test Scheduling in High Performance VLSI System Implementations", IEEE Trans. on Computers, Vol. 41, No. 1, January 1992.

[19] Scholz, H.N., R.E. Tulloss, C.W. Yau and W. Wach, "ASIC Implementations of Boundary-Scan and BIST", 8th Int'l Custom Microelectronics Conference, pp. 43.0-43.9, London, U.K., November 1988.

[20] Tulloss, R.E. and Yau, C.W., "BIST & Boundary-Scan for Board Level Test: Test Program Pseudocode", Proc. of IEEE European Test Conference (ETC), pp. 106-111, Paris, April 1989.

[21] van de Goor, A.J., and Zorian, Y., "Functional Tests for Arbitration SRAM-Type FIFOs", Proc. IEEE 1st Asian Test Symp., pp. 96-101, Nov. 1992.

[22] van Riessen, R.P., and Kerkhoff, H.G., "Automatic Test-Specification Generation for Macro-Level BIST Based on the Boundary-Scan Standard", Proc. European Test Conference, pp. 447-453, April 1991.

[23] Zorian, Y., "A Structured Approach to Macrocell Testing Using Built-In Self-Test", Proc. IEEE Custom Integrated Circuits Conference, pp. 28.3.1-28.3.4, Boston, 1990.

[24] Zorian, Y., "An Effective BIST Scheme for CAMs", IEEE BIST Workshop, March, 1991.

[25] Zorian, Y., "On Output Data Specific Compaction", IEEE Int'l Symp. on Circuits and Systems, May 1991.

[26] Zorian, Y., and Ivanov, A., "An Effective BIST Scheme for ROMs", IEEE Trans. on Computers, Vol. 41, No. 5, pp. 646-653, May 1992.

[27] Zorian, Y., "A Universal Testability Strategy for Multi-Chip Modules Based on BIST and Boundary-Scan, IEEE Int'l Conf. on Computer Design, pp. 59-66, Oct. 1992.