An Automated BIST Approach for General Sequential Logic Synthesis

C. E. Stroud

AT&T Bell Laboratories
1200 East Warrenville Road
Naperville, IL 60566

ABSTRACT

An automated Built-In Self-Test (BIST) technique for general sequential logic is described. This BIST approach has been incorporated in a behavioral model synthesis system providing automated implementation of BIST in Very Large Scale Integration (VLSI) devices as well as Programmable Logic Device (PLD) based circuit packs. The BIST technique can be directly used at all levels of testing from device testing through system diagnostics. The BIST approach is based on selective replacement of existing system memory elements with BIST flip-flop cells that are connected to form a circular chain, performing data compaction and test pattern generation simultaneously. Two production VLSI devices have been implemented with this automated BIST approach. In each case, the total fault coverage was in excess of 96 percent and the logic overhead incurred was between 9.7 and 18.9 percent.

INTRODUCTION

Built-In Self-Test (BIST) techniques have gained popularity in recent years as a solution to growing testing complexity in Very Large Scale Integration (VLSI) devices. This growing testing complexity has traditionally been attributed to shrinking design rules in VLSI technologies, allowing significant increases in circuit density and complexity in VLSI devices. Another major factor that adds to the testing complexity issue is recent advances in behavioral modeling synthesis systems. Design processes using automated design synthesis techniques tend to take designers to higher levels of abstraction, removing them from an intimate gate level knowledge of the circuitry. On the other hand, these design processes significantly reduce the design interval required to implement the system function [1]. Creating an effective BIST approach for a given system function requires designing a complete BIST function (including pattern generation, response compaction, system input isolation, and proper control of the BIST sequence) in addition to the required system function. The manual incorporation of a BIST feature can undermine the advantages gained when using automated design synthesis techniques by increasing design intervals and adding the additional risk of improper operation of either the system function or the BIST function once the VLSI device or circuit pack has been implemented. Therefore, the automation of BIST techniques is imperative in order to maintain pace with automated design synthesis capabilities. Although BIST approaches for regular structures such as Random Access Memories (RAMs) have been developed and automated [2,3], the lack of a complete and automated BIST capability at the device and circuit pack level is primarily due to the lack of an automated BIST approach for the general sequential logic common to all VLSI devices and circuit packs.

Krasniewski and Pilarski [4,5] recently proposed a BIST technique, referred to as the Circular Self-Test Path (CSTP) technique [4], that promises to have potential for design automation. The CSTP technique is based on a feedback shift register approach with the output of the last flip-flop in the shift register feeding back to the first flip-flop. The construction of the feedback shift register creates a data compaction capability with the results of the data compaction used as the input stimulus for the circuitry under test during the subsequent clock cycle(s). The effectiveness of the patterns supplied to the circuitry under test were shown to be comparable to that of an ideal pseudo-random test generator [5]. Although the CSTP technique appears to be an excellent BIST approach for device level testing, certain assumptions (that will be discussed in this paper) were made which limit its effective use at the system level. Improvements to system level testing due to reduced diagnostic run time, improvements to Mean Time To Repair (MTTR), and reductions in diagnostic code development are but a few of the many benefits that can be obtained when BIST capabilities can be used at the system level [2]. Considering the total number of executions of the BIST sequence over the lifetime of the product, a conservative estimate of the percentage of executions of BIST by system diagnostics is 99 percent. Since the major use of BIST is by system diagnostics, the effort made in any BIST design to make access by system diagnostics feasible is usually worth the investment in design time and logic overhead.

This paper describes a BIST approach which was developed during the same period of time as the CSTP technique and uses some of the same underlying principles. However, this BIST approach can be used directly at all levels of testing through system diagnostics, and a partial scan capability is provided with no extra logic overhead. In addition, this BIST approach has been automated and incorporated into a behavioral model synthesis system providing automated implementation of BIST in VLSI devices as well as Programmable Logic Device (PLD) based circuit packs. Fault coverage, random pattern properties, and logic overhead will be reported for two production VLSI devices that have been implemented with this automated BIST approach. One of the VLSI device implementations demonstrates that this approach can easily be integrated with automated BIST techniques for embedded RAMs to obtain a complete application of BIST at the device level.

ARCHITECTURE AND OPERATION OF THE BIST APPROACH

The basic architecture of a device or circuit pack using this automated implementation for BIST is shown in Figure 1 in a configuration that would be seen in the BIST mode of operation. The underlying concept of this automated BIST technique is to selectively replace memory elements in the system function with a special BIST flip-flop cell and to interconnect these flip-flops such that a circular chain of BIST flip-flops is obtained. A Signature Analysis Register (SAR) is included in this BIST chain to allow effective use of the BIST capability during system diagnostics. The SAR is strategically located to facilitate easy access during device and system testing, but it can be located anywhere in the BIST flip-flop chain. Multiplexers are used at the inputs to the circuit under test to isolate the system data from the BIST circuitry so that reproducible results can be obtained during system diagnostics from one execution of the BIST sequence to the next. In addition to input isolation, the multiplexers provide the ability to apply patterns generated by the BIST chain to circuitry normally driven by the system inputs. Finally, a control circuit is needed to control the BIST sequence and to hold the resultant signature in the SAR until it can be read by the test machine at device

and circuit pack level testing or by diagnostic software during system diagnostics.

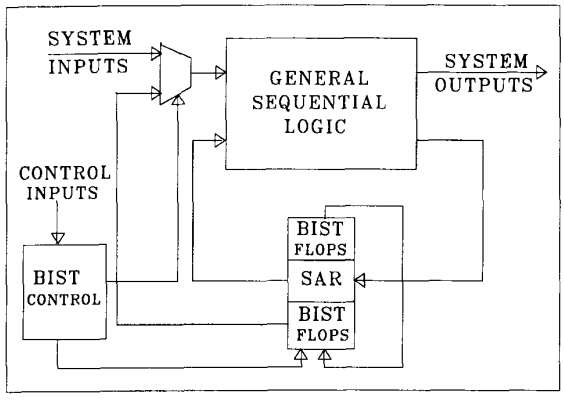

FIGURE 1

BIST Circuitry Architecture

The BIST flip-flop used for selective replacement of the system D-type flip-flops is shown in Figure 2. The BIST flip-flop for this BIST approach uses an exclusive-or gate with each input driven by a NAND gate. This provides four modes of operation using the two BIST control signals "B0" and "B1". The four modes of operation, given in Table 1, include an initialization mode, a shift mode, a system mode, and a BIST mode similar to that of some applications of a basic Built-In Logic Block Observer (BILBO) cell [6]. The initialization mode sets the flip-flop output to a logic zero and is used for initialization of the BIST chain, and the general sequential logic being tested, at the beginning of the BIST sequence. The shift mode configures the BIST chain into a partial scan chain that can be used to obtain additional fault coverage and/or testing orthogonality to that obtained by BIST. The system mode enables the flip-flops to behave as normal D-type flip-flops and is used for performing the system function. Finally, the BIST mode performs an exclusive-or function of the system data input and the output data from the previous flip-flop in the BIST chain.

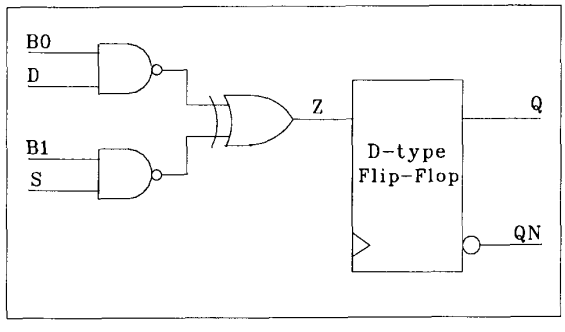

FIGURE 2

BIST Flip-Flop

TABLE 1

BIST Flip-Flop Modes of Operation

B0	B1	INPUT TO FLIP-FLOP	MODE
0	0	Logic Zero	Initialization
0	1	Value at Input S	Shift Mode
1	0	Value at Input D	System Mode
1	1	D exclusive-or S	BIST Mode

In operation, the BIST controller, upon activation of the BIST sequence, puts the BIST flip-flops and SAR into the initialization mode via the "B0" and "B1" control leads for a number of clock cycles. This first initializes the BIST flip-flops themselves (only one clock cycle needed for this) and then initializes the general sequential logic in the circuitry under test (a clock cycle is needed for each level of memory elements in the circuitry under test). The circuit is then put into the BIST mode and the initialized value in the BIST chain is applied to the general sequential logic being tested. The resulting logic values at the inputs of the BIST chain are then compacted with the initialized value in the chain and shifted by one memory element in the BIST chain (via the exclusive-or function). This value represents the signature resulting from the application of the first test pattern and is used as the next test pattern to be supplied to the general sequential logic. This process continues with the signature produced by each clock cycle of operation being used as the next input test pattern to the circuitry under test. As the process continues, each bit, which represents fault information from the general sequential logic, propagates to the SAR and under goes the normal data compaction associated with signature analysis. After the specified number of clock cycles (determined by the BIST controller design) has completed in the BIST mode, the SAR is disabled from further data compaction until the contents of the SAR are read by the test machine or system diagnostics and the BIST sequence is deactivated.

ADDITIONAL ASPECTS OF THE BIST APPROACH

The use of the results of data compaction as the input stimulus for the general sequential logic under test is also the basic concept used by the CSTP approach [4,5]. Similar approaches have been suggested and investigated by several other researchers [7,8,9]. Patterns generated by these techniques have been shown to provide high fault coverage and close correlation to patterns generated by ideal random and pseudo-random pattern generators. The major differences in this BIST approach and the CSTP technique are the type of BIST flip-flop used, the associated capabilities the BIST flip-flop provides, and the use of an SAR in the BIST chain.

The BIST flip-flop used in the CSTP approach consisted of an exclusive-or gate and a multiplexer providing a system mode and BIST mode of operation [4,5]. In the CSTP technique, the assumption was made that there existed a global signal for initializing the circuit to predetermined states to obtain reproducible results from one execution of the BIST sequence to the next [4,5]. This is not a valid assumption for many VLSI and circuit pack implementations. Therefore, in this approach, an initialization capability is implemented in the BIST circuitry and regarded as BIST logic overhead. In the CSTP approach, when a global reset capability was not available, the recommendation was made to include an additional multiplexer in the basic CSTP cell and shift in a set of initialization values [4,5]. Though a partial scan mode is an inherent part of this BIST capability, system diagnostics rarely have access to, or control of, scan chains in actual practice so that the partial scan approach to initialization is not a valid alternative in most systems. Therefore, an initialization mode is also necessary to facilitate the use of the BIST capability at the system level.

In the CSTP approach, the pass/fail determination of the BIST sequence was made by monitoring a stream of bits exiting a selected output of the circular self-test path within a specified interval at the end of the test sequence [4,5]. The number of flip-flops in the circular self-test path determined the length of this signature to minimize the potential for fault masking. This technique is sufficient during device level testing when per clock cycle control of the device is obtained via the test machine. However, system diagnostic code typically runs on a microprocessor running asynchronously with respect to the device or circuit pack being diagnosed in the system such that per clock cycle control or comparisons are impossible. By including an SAR in the BIST chain, the fault information cycling through the chain is held and compacted within the SAR. The resultant signature can then be held at the end of the BIST sequence until system diagnostics can read the signature to determine the pass/fail status. In addition, only the signature within the SAR is needed to determine the pass/fail status regardless of the number of flip-flops in the BIST chain.

Selective replacement of existing system memory elements with BIST flip-flops helps to reduce logic overhead incurred with the BIST implementation. The primary idea behind selective replacement of memory elements can be seen in Figure 3. Since few patterns are required to test the flip-flops and their interconnections, in the simple case of a flip-flop to flip-flop data transfer, the patterns produced by the first BIST flip-flop in the chain will be sufficient to test the subsequent flip-flops. This idea can be extended to memory elements that are driven by small logic cones (a two input NAND gate in this example). Sufficient patterns will be generated to test the NAND gate and its associated flip-flop, yet, the NAND gate will not significantly degrade the patterns so that the circuitry being driven by the NAND gate's flip-flop will be sufficiently tested. Thus, by not replacing memory elements with few inputs to their driving cones (typically two inputs or less), logic overhead due to incorporation of BIST is minimized without appreciable impact on the fault coverage. In addition, selective replacement provides the opportunity to prevent replacement of a memory element that may reside in a critical timing path avoiding the introduction of additional delays.

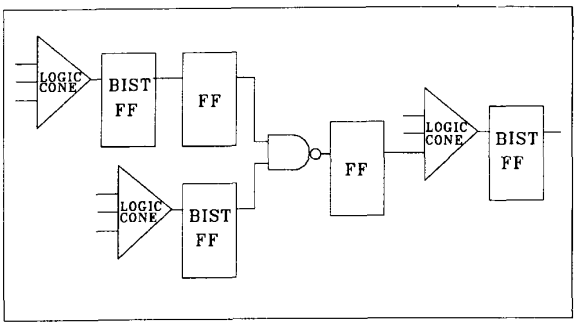

FIGURE 3

BIST Flip-Flop Selective Replacement

The addition of the Partial Scan Design (PSD) capability has no effect on the logic overhead of the BIST flip-flop. However, an additional pin to the device or circuit pack is required to control the PSD mode. The PSD mode can be used to augment the fault coverage of the BIST sequence at device and circuit pack level testing. This is particularly useful in detecting random pattern resistant faults or faults remaining due to "limit cycling" during the BIST sequence [10]. In addition, the PSD capability can be used to provide orthogonality in testing at the device and circuit pack levels of testing from that at system level testing using system diagnostics. This helps to insure that device faults not detected by the BIST capability (used at all levels of testing, including system diagnostics) can be detected at device testing for new product and at circuit pack testing for product returned from the field. The PSD test patterns for additional and/or orthogonal testing can be generated automatically [14,15].

AUTOMATION OF THE BIST APPROACH

The BIST approach has been automated and incorporated in a behavioral model synthesis system called CONES [11]. CONES generates general sequential logic from behavioral models written in "C" and implements the resultant logic in the form of standard cell VLSI devices or PLD based circuit packs. This BIST approach can be incorporated in either type of implementation. For VLSI implementations, the automated BIST approach is applied as a post-process to the standard cell synthesis. For PLD based circuit packs, the BIST approach is applied as a preprocess to PLD partitioning [12] and synthesis. In both cases, the application of the BIST approach is the same.

The first step in the BIST implementation process is the determination of the memory elements to be replaced with BIST flip-flops. This selection is made automatically by the BIST software in CONES with the designer given the ability to select the number of inputs to logic cones that will be used to determine selective replacement. All memory elements driven by logic cones with greater than the number of inputs specified by the designer will be selected for replacement with the BIST flip-flop. Since selective replacement helps to reduce logic overhead, specifying a larger number of inputs will result in the selection of fewer flip-flops to be replaced. However, specifying too large a number can reduce the number of BIST flip-flops to the point of being detrimental to fault coverage. In addition, this can destroy the ability of the BIST sequence to obtain reproducible results by preventing proper initialization of some of the general sequential logic. Therefore, selective replacement of memory elements with logic cones having three or more inputs is recommended.

The result of the selection of memory elements for replacement with BIST flip-flops is a textual file generated by the software. The designer is given the chance to review and edit the list of memory elements to be replaced. The designer can remove any flip-flops in the list which reside in critical timing paths of the circuit. This enables the designer to prevent the application of BIST from interfering with the required system performance specifications. Similarly, the designer can add memory elements to the list that were not selected for replacement by the software if so desired. This textual file also contains the information determining which system inputs will be isolated.

Once the designer has made any desired modifications to this text file, the BIST software checks for and minimizes occurrences of register adjacency [13]. A very simple example of register adjacency that could occur in this BIST approach is shown in Figure 4. Consider a flip-flop to flip-flop data transfer in which the chain of BIST flip-flops is constructed such that the flip-flops are connected in the same sequence in the BIST chain. During the BIST mode, the output of the XOR gate will always be a logic zero. However, the data from the first BIST flip-flop contains information regarding faults that have been detected earlier in the BIST chain. This fault information could be lost by the occurrence of register adjacency and lower fault coverage will result in the BIST sequence. This example never occurs under normal selective memory element replacement described in the previous section. However, the occurrence of register adjacency can be found in more subtle situations such as counters and complex shift registers. Since each bit of data in the BIST chain contains fault information, the loss of a single bit through register adjacency may lead to fault masking. Register adjacency can be minimized by preventing the connection of two consecutive flip-flops in the chain when the first flip-flop also serves as an input to the logic cone driving the second flip-flop. This register adjacency check and ordering of the chain to minimize register adjacency is performed automatically by the BIST software in CONES.

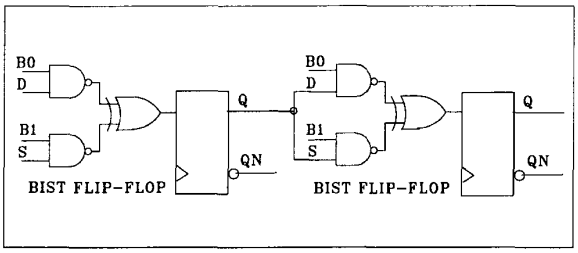

FIGURE 4

Register Adjacency Example

Finally, the replacement of the memory elements with BIST flip-flops, the appropriate connection of the BIST flip-flops to create the BIST chain, and the addition of input isolation multiplexers is performed by the BIST software. For VLSI implementations, this final step is performed by modifying the "netlist" that specifies the interconnection of standard cells in the device. In the case of PLD based circuit packs, the input files to the PLD partitioning program in CONES [12] are modified to reflect the appropriate BIST circuitry implementation and the PLD synthesis process proceeds as usual [11].

The BIST controller and the SAR for this BIST approach are provided for the designer in the form of behavioral models written in "C" and synthesizable by CONES. These models may be used by designers for:

- direct use (as is) to be synthesized by CONES for controlling the BIST circuitry applied by CONES and providing the appropriate data compaction via signature analysis,
- modification to provide additional or modified control to and data compaction for the BIST circuitry applied by CONES, or other BIST or system functions, or
- use as a basis for understanding the control and data compaction needed by the BIST circuitry applied by CONES if the control and/or data compaction is to be supplied as part of the system function to be synthesized.

APPLICATIONS OF THE BIST APPROACH

Two production VLSI devices have been implemented using this automated BIST approach. The first device has been in the field for approximately 2 years, and the second device is currently in product manufacturing. In both cases, the BIST capability is used for all levels of testing with primary use during system diagnostics. In this section, the logic overhead incurred due to incorporation of this BIST approach is given for each device. Detailed fault simulations were performed along with investigations of the random pattern properties of the test patterns generated by their BIST chains.

Logic Overhead

The first device implemented with this BIST approach consisted of approximately 2500 gates and an 8K RAM embedded in the device. The RAM contained an automated BIST capability [2,3] which was used in conjunction with this BIST approach, for the general sequential logic, to obtain a completely self-testing device (with the exception of the input and output buffers and portions of the BIST input isolation multiplexers). During the first half of the BIST sequence, the RAM BIST sequence was executed in parallel to the BIST sequence for the general sequential logic. The results of the RAM BIST sequence, in terms of the pass/fail indication outputs of the RAM and the RAM data outputs, were allowed to enter the BIST chain for data compaction through the normal system logic. During the second half of the BIST sequence, the RAM was put into the normal mode of operation such that random accesses of the RAM were executed via the patterns generated by the BIST chain. In this way, the interface between the RAM and the general sequential logic was tested. Upon completion of the BIST sequence, the signature in the SAR was held until it could be read by the test machine or system diagnostics. The resultant signature was used to determine the pass/fail status of the entire device (with the exception of input/output buffers and portions of the input isolation multiplexers) since the results of the RAM BIST sequence had been compacted along with the results of the general sequential logic BIST sequence. The input and output buffers were tested, along with the untested portions of the input isolation multiplexers, with a small set of functional test patterns at device and circuit pack level testing and with functional diagnostics in the system.

The logic overhead due to the incorporation of the automated BIST approach is shown in Table 2 for the first device. The BIST circuitry (for the RAM BIST control and for the general sequential logic BIST) accounted for 27.9 percent of the total general sequential logic but only 9.7 percent of the active area of the device when the area of the RAM was taken into consideration. Since the total device area was limited by the number of input/output buffers (also referred to as "pad limited"), the incorporation of the BIST circuitry had no impact on the total device area. A full fault simulation was performed on the device executing the BIST sequence in the same manner as it would be executed during system diagnostics and device testing. The fault coverage obtained with the BIST sequence alone was 93.4 percent. A short set of functional test patterns to test the input/output buffers and the input isolation multiplexers provided an additional 4.8 percent fault coverage to give a total fault coverage of 98.2 percent. Of the total 152 memory elements in the device, 76 were selected by the BIST software for replacement with BIST flip-flops, creating a BIST chain of 92 flip-flops when the SAR was included. In this device, control for the BIST and the SAR were provided by using existing system circuitry as BIST control and SAR functions in the BIST mode. The additional logic required to turn the existing system functions into control and SAR functions is shown in Table 2 under the heading of "BIST Integration".

TABLE 2

Logic Overhead Incurred With BIST
For Device Number 1

FUNCTION	NO. OF GATES	% TOTAL LOGIC	% ACTIVE AREA
System Function	1820	72.1	90.3
RAM BIST Control	51	2.0	0.7
BIST For Logic	460	18.3	6.5
Input Isolation	66	2.6	0.8
BIST Integration	127	5.0	1.7
Total for BIST	704	27.9	9.7
TOTAL	2524		

The second VLSI device consisted of approximately 20,700 gates with no internal RAM. The BIST approach was applied to only 85.1 percent of the device. This was done in order to insure access for activating/deactivating the BIST sequence and reading the resultant signature during system diagnostics. The untested portions of the device were the most controllable and observable parts of the device affording easy testing from a functional diagnostic standpoint. During device and circuit pack level testing, 97.3 percent of the device is tested with the BIST sequence. The logic overhead due to the incorporation of the automated BIST approach is given in Table 3 and accounts for 18.9 percent of the logic in the device. This device was also pad limited so that the incorporation of the BIST circuitry did not

increase the total device area. A full fault simulation was performed on the device executing the BIST sequence in the manner it would be executed during system diagnostics. The fault coverage obtained from this simulation was 77.8 percent which corresponds to a fault coverage of 91.4 percent for the 85.1 percent of the circuitry tested by BIST during system diagnostics. Similarly, a fault simulation was performed on the device executing the BIST sequence in the manner it would be executed during device testing. The fault coverage obtained from this simulation was 89.2 percent which corresponds to a fault coverage of 91.6 percent for the 97.3 percent of the circuitry tested by BIST during device level testing. A short set of functional tests brought the fault coverage to 95.3 percent and the PSD mode was used to obtain a total fault coverage of 96.8 percent. Of the total 920 memory elements in the device, 723 were selected for replacement with BIST flip-flops by the BIST software, creating a BIST chain of 739 flip-flops when the SAR was included. In this device, the BIST control and SAR functions were obtained by synthesizing the standard BIST controller and SAR models as described in the previous section.

TABLE 3

Logic Overhead Incurred with BIST
For Device Number 2

FUNCTION	NUMBER OF GATES	PERCENT OF TOTAL LOGIC
System Function	16820	81.1
BIST Flip-Flops	3109	15.0
Input Isolation	345	1.7
BIST Control	270	1.3
BIST SAR	185	0.9
Total for BIST	3909	18.9
TOTAL	20729	

The BIST flip-flops have the greatest impact on the logic overhead incurred in the BIST implementation, as can be seen in the data for the two VLSI implementations. Reductions in the number of system memory elements replaced by BIST flip-flops could improve the logic overhead penalty. Since the selective replacement algorithm described in the previous section is rather simple, improvements to this method of selection appear to be a good area for future research. Investigation of algorithms for selection of flip-flops for partial scan chains [14,15] may provide insight to improve the selective replacement algorithm for this BIST approach.

Random Pattern Properties

In the applications described above, the BIST sequence was 8176 clock cycles in duration (not counting a 16 clock cycle initialization sequence). To study the random properties of the patterns generated by the BIST chain, the number of logic ones generated by each flip-flop in the BIST chain was determined during the BIST sequence. In the case of an ideal pseudo-random pattern generator, the number of ones generated by each element would be 4088. The mean and variance of the number of ones generated by flip-flops in the BIST chains of the first two VLSI devices as compared to an ideal pseudo-random pattern generator are given in Table 4.

TABLE 4

Number of Logic Ones Generated
by BIST Chains

DEVICE	MEAN	VARIANCE
IDEAL	4088	0
DEVICE #1	4088	1466
DEVICE #2	4084	12460

Another method used to study the random-pattern properties of this BIST approach was to group BIST flip-flops into groups of four and observe the distribution of the 16 possible patterns that could be obtained with each group of four bits. In a pseudo-random pattern generator each of the 16 possible patterns should occur an equal number of times with each pattern occurring 511 times during the 8176 clock cycle BIST sequence. The mean and variance of patterns generated by the groups of four BIST flip-flops in the BIST chains are given in Table 5 for both devices.

TABLE 5

Distribution of Patterns Generated
by Groups of Four Flip-Flops in
BIST Chains

	DEVICE NUMBER 1		DEVICE NUMBER 2	
PATTERN	MEAN	VARIANCE	MEAN	VARIANCE
0x0	508	389	517	1179
0x1	504	454	511	1075
0x2	517	616	516	5786
0x3	511	143	514	3629
0x4	511	278	507	840
0x5	514	384	509	797
0x6	507	453	509	802
0x7	512	287	507	714
0x8	514	261	510	625
0x9	507	598	510	786
0xA	503	334	515	1035
0xB	512	285	509	524
0xC	516	285	511	1255
0xD	512	571	512	1196
0xE	506	597	512	1169
0xF	513	666	512	1104
All PATs	511	411	511	1408

The first device demonstrates pattern generations properties for both the number of ones produced by each BIST flip-flop and the patterns produced by groups of four BIST flip-flops that are quite close to that of an ideal random or pseudo-random pattern generator. The larger variances associated with the second device can be attributed to some extent to the length of the BIST chain and the settling time of the circuit [4]. The variances for both the number of logic ones generated by each BIST flip-flop and the patterns generated by groups of four BIST flip-flops were reduced by an average of 17.5 percent when measured after the first 740 clock cycles which corresponds to one complete cycle through the BIST chain. However, the primary reason for the larger variances in the second device are due to occurrences of register adjacency caused by problems in the register adjacency minimization software at the time the second device was synthesized. This accounts for the large variances for patterns '0x2' and '0x3'. The effect of register adjacency on the resultant fault coverage for this device is currently under study.

SUMMARY

Advances in design automation, particularly in the area of behavioral model synthesis, significantly reduce design intervals and remove designers from gate and register level details of device and circuit pack implementations. This increases the need for automated BIST techniques to keep pace with the rest of design automation. Since device and circuit pack level testing account for only a small percentage of the potential use of BIST capabilities, system diagnostic access to BIST should be a high priority in any BIST implementation. The BIST approach described in this paper has been automated and is designed to be used at all levels of testing through system diagnostics. This BIST approach has been applied to two production VLSI devices with a logic overhead penalty of less than 19 percent in each case. The fault coverage obtained with the BIST sequence alone was in excess of 90 percent and the patterns generated by the BIST chain for application to the general sequential logic correlated closely with that of an ideal random pattern generator. The PSD mode provided by this BIST approach can be used to detect random pattern resistant faults that might not be detected by the BIST sequence. The two primary areas for future research are improvements to the selective replacement algorithm which would improve the logic overhead penality and improvements to the register adjacency minimization algorithm which could improve fault coverage. Although this BIST approach has also been automated for PLD based circuit pack synthesis, no applications of this capability have yet been implemented.

REFERENCES

[1] D. A. Pierce and C. E. Stroud, "Impact of Behavioral Modeling and Synthesis on the Design Process", Proc. of National Communications Forum, September 1986, Vol. 40 No. 2, pp. 1058-1061.

[2] S. K. Jain and C. E. Stroud, "Built-In Self-Testing of Embedded Memories", IEEE Design and Test of Computers, Vol. 3, No. 5, October 1986, pp. 27-37.

[3] D. R. Aadsen and S. K. Jain, "Automation of BIST for Embedded Memories", Proc. of IEEE Custom Integrated Circuits Conference, 1987, pp. 66-67.

[4] A. Krasniewski and S. Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique", Proc. of ACM/IEEE Design Automation Conference, July 1987, pp. 407-415.

[5] A. Krasniewski and S. Pilarski, "Circular Self-Test Path", Technical Report No. 128, Technical University of Warsaw, Institute of Telecommunications, 1986.

[6] B. Konemann, T. Mucha, and G. Zwiehoff, "Built In Logic Block Observation Techniques", Proc. of IEEE International Test Conference, October 1979, pp. 37-41.

[7] F. B. Beucler and M. J. Manner, "HILDO: The Highly Integrated Logic Device Observer", VLSI Design, June 1984, pp. 88-96.

[8] B. I. Dervisoglu, "VLSI Self-Testing Using Exhaustive Bit Patterns", Proc. of IEEE International Conference on Computer Design, 1985, pp. 558-561.

[9] T. W. Williams, R. G. Walther, P. S. Bottorff, and S. Das Gupta, "Experiment to Investigate Self-Testing Techniques in VLSI", IEEE Proceedings, Vol. 132, Pt. G, No. 3, June 1985, pp. 105-107.

[10] M. J. Ohletz, T. W. Williams, and J. P. Mucha, "Overhead in Scan and Self-Testing Designs", Proc. of IEEE International Test Conference, September 1987, pp. 460-470.

[11] C. E. Stroud, R. R. Munoz, and D. A. Pierce, "CONES: A System for Automated Synthesis of VLSI and Programmable Logic from Behavioral Models", Proc. of IEEE International Conference on Computer-Aided Design, November 1986, pp. 428-431.

[12] R. R. Munoz and C. E. Stroud, "Automatic Partitioning of Programmable Logic Devices", VLSI Systems Design, Vol 3. No 11, October 1987, pp. 74-79.

[13] C. L. Hudson and G. D. Peterson, "Parallel Self-Test with Pseudo-Random Test Patterns", Proc. of IEEE International Test Conference, September 1987, pp. 954-963.

[14] V. D. Agrawal, K. T. Cheng, D. D. Johnson, and T. Lin, "A Complete Solution to the Partial Scan Problem", Proc. of IEEE International Test Conference, September 1987, pp. 44-51.

[15] E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology", Proc. of IEEE International Test Conference, November 1980, pp. 153-162.