Using BIST Control for Pattern Generation

Gundolf Kiefer and Hans-Joachim Wunderlich

Computer Architecture Lab
University of Stuttgart, Breitwiesenstr. 20/22
D-70565 Stuttgart, Germany

Abstract

A deterministic BIST scheme is presented which requires less hardware overhead than pseudo-random BIST but obtains better or even complete fault coverage at the same time. It takes advantage of the fact that any autonomous BIST scheme needs a BIST control unit for indicating the completion of the self-test at least.

Hence, pattern counters and bit counters are always available, and they provide information to be used for deterministic pattern generation by some additional circuitry. This paper presents a systematic way for synthesizing a pattern generator which needs less area than a 32-bit LFSR for random pattern generation for all the benchmark circuits.

Keywords: *deterministic BIST, scan-based BIST*

1. Introduction

Modern design and package technologies make external testing more and more difficult, and built-in self-test (BIST) is becoming an attractive alternative. Chips integrated into MCMs are hard to access even if the design is IEEE 1149.1 boundary scan compatible, and self-testable modules are mandatory in many cases. The emerging technique of embedded coreware raises an even stronger need for BIST as the internal structure of the cores may be hidden to the designer, and accessible interfaces between cores may not exist at all. In addition to these technology-driven reasons for BIST, there are classical advantages reported in textbooks as high fault coverage, no need for automatic test equipment and support during system maintenance [ABF90, WuZo97].

If the BIST capability has to be exploited in a hierarchical system design as defined by IEEE 1149.5, for instance, the self-test must be performed in an autonomous mode which is initiated by a START signal, and whose completion is indicated by a TESTEND signal [HaWu89, HaKr95]. This leads to the basic architecture of a self-testable module as shown in figure 1.

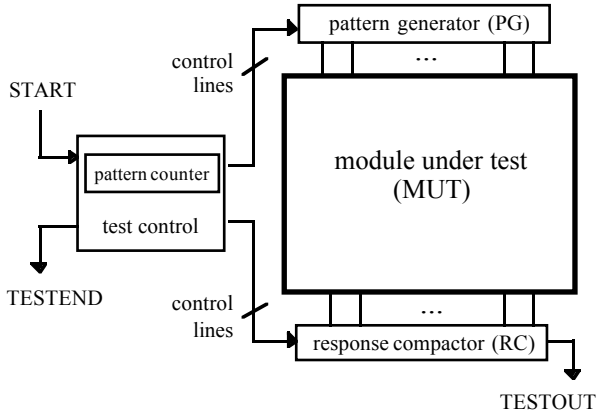


Figure 1: Basic BIST structure

If a test-per-clock scheme is applied both pattern generator and response compactor are implemented by multifunctional test registers based on linear feedback shift registers (LFSRs) like the so-called BILBO [KMZ79]. As this type of registers cannot perform response compression and pattern generation concurrently, the entire test must be scheduled and divided into sessions, and rather a complex test control unit has to be synthesized [CKS88, StWu90, StWu94].

BIST control is simplified if a test-per-scan technique is applied as first proposed in [EiLi83] and further developed under the names STUMPS and LOCST, e.g. [BaMc84, LeBl84]. In this case, the MUT is equipped with a (partial) scan path, the scan-in input is fed by an LFSR which generates a pseudo-random bit stream, and the scan-out output is observed by another LFSR for signature analysis. Testing is performed by shifting one pattern into the scan path, capturing the MUT response by one system clock, and shifting out the new scan path content, whereby a new pattern is shifted in simultaneously. The BIST control

unit does not need to distinguish between different test sessions but it must count the patterns until the test is completed. The entire structure is shown in figure 2.

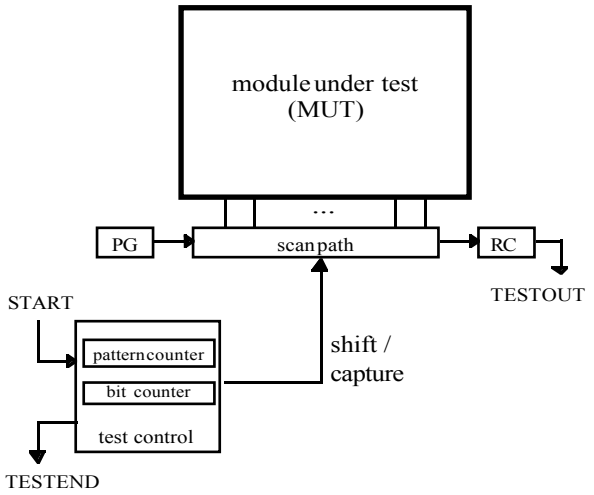


Figure 2: BIST control of a test-per-scan structure

If the pattern generator PG is implemented by an LFSR, its minimum length L depends on both the length n of the scan path and the number m of pseudo-random patterns (see fig. 3).

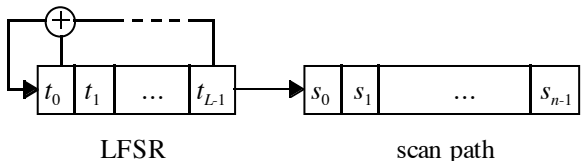


Figure 3: LFSR length L and scan length n

An LFSR with a primitive feedback polynomial cycles through 2^L-1 different states (for details see [PeWe72, LiNi97]), and each state corresponds to a certain bit s_j in the scan chain. If for all m patterns the first bit s_0 corresponds to a different state, the minimum LFSR length is determined by $2^L - 1 \geq m$. A necessary and sufficient condition for reaching the lower limit is that the scan path length n and the cycle length 2^L-1 do not have a common divisor, gcd $(2^L - 1, n) = 1$. But in this case we have strong linear dependencies between the generated patterns since some of them differ only by a certain number of shifts. Linear dependencies are reduced if all bits generated correspond to a different LFSR state, whereby the minimum length is bounded by $2^L - 1 \geq m \cdot n$.

For this LFSR length, linear dependencies within the scan chain may also reduce fault coverage, for details see [AvMc93, BMS87], e.g. . As an example, assume, we have to generate all 8 possible assignments to the three scan elements s_i, s_j, s_k. If $k - i > L$ this may be impossible for an LFSR of length L even if a primitive polynomial is used [BCR83]. In consequence, L should be sufficiently large to cover all the necessary bit distances. If, for instance, the scan chain consists of several 32-bit data registers, the LFSR should be of length $L = 32$ bits, too. This prevents linear dependencies even if far less than 2^{32} patterns are applied.

The pattern generator PG of figure 2 can also be implemented to produce weighted random patterns [BRGL89b, Wu87, StWu91], pseudo-exhaustive patterns [Akers85, HWH90], or deterministic patterns [HELL92, ToMc96, WuKi96]. All these different types of pattern generators are able to obtain complete coverage of all detectable faults, and the whole structure is easily implemented by some CAD tool which supports scan design. There is no need for reordering or modifying the scan chain, and test points in the mission logic are superfluous, otherwise they would have an impact on the system behavior and may require a redesign. The main drawback of these types of pattern generators is the hardware overhead which may be larger than the LFSR area by several orders of magnitude.

In this paper it is shown that the area of a deterministic pattern generator PG can be reduced significantly if PG is not working in an autonomous mode like an LFSR but exploits information from the pattern counter and bit counter of the BIST control unit as shown in figure 4.

The rest of this paper is organized as follows: in the next section we introduce the structure of PG in detail, in section 3 a procedure is presented which synthesizes PG for a given module under test (MUT). The procedure is illustrated by a small example in section 4, and results of this procedure are reported in section 5, they confirm that the hardware overhead of this BIST scheme is less than the area required for the usual LFSR-based pseudo-random BIST.

2. The target structure

The pattern generator to be developed will be an extension of the bit-flipping BIST scheme presented in [WuKi96]. This scheme is summarized next, the extension exploits the autocorrelation of test patterns which is explained in the second part of this section.

2.1. Bit-flipping BIST

The bit-flipping BIST scheme is based on the observation that many patterns of an LFSR sequence do not increase the fault coverage and can be mapped

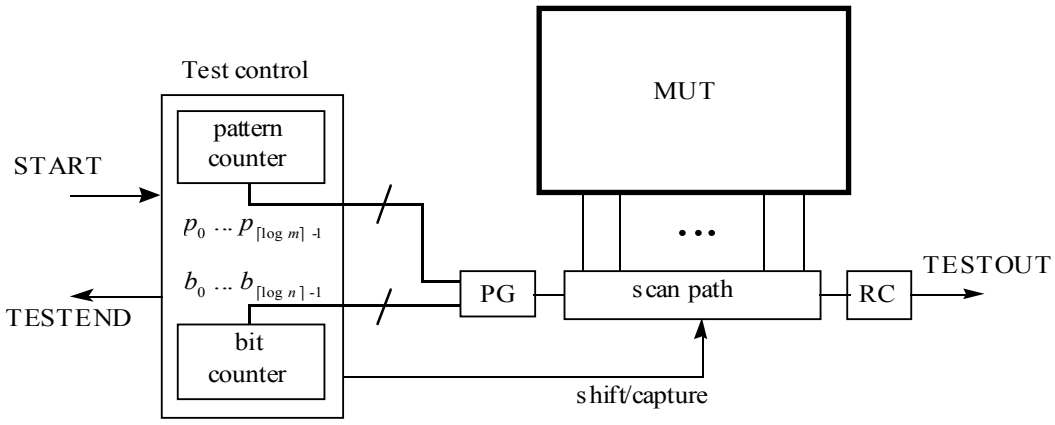


Figure 4: Exploiting BIST control information during pattern generation

to some precomputed deterministic test patterns by changing some bits. Since one can choose from a large space of random patterns, there is a high chance to find a pair of a deterministic and a random pattern, so that only very few bits have to be changed. As a bit corresponds to a state of the LFSR, bit-flipping is implemented by a combinational function as shown in figure 5.

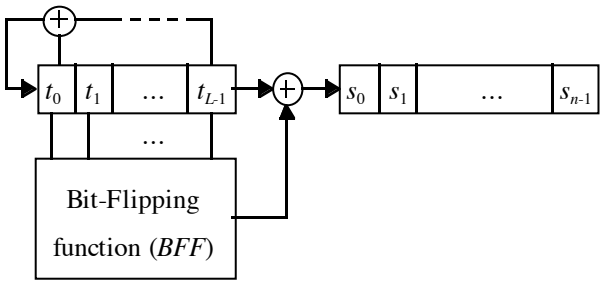


Figure 5: Bit-flipping BIST

The bit-flipping BIST scheme is the most area-efficient mixed-mode scan-based BIST technique published so far due to the following reasons:

1. *BFF* has a very small on-set and off-set, but a very large don't-care-set. In general this leads to low area requirements even for a two-level implementation.
2. *BFF* is also used to overcome linear dependencies so that the LFSR can be shortened.

In many cases, it is possible to cut the LFSR down to a length L with $2^L - 1 \geq m$.

The length may be further reduced if the autocorrelation of deterministic test patterns is exploited so that the size of the combinational logic is reduced, too. This will be described in the next paragraphs.

2.2. Autocorrelated test patterns

Very often, deterministic test patterns can be clustered into a few sets in such a way that all the patterns of a set look very similar. This effect was studied and used for a BIST scheme in [PaRa91], e. g. .

An example is given in figure 6, where the combination of an AND/OR-gate is tested by clusters. Each cluster contains test patterns which are variations of a master pattern at a few bit positions.

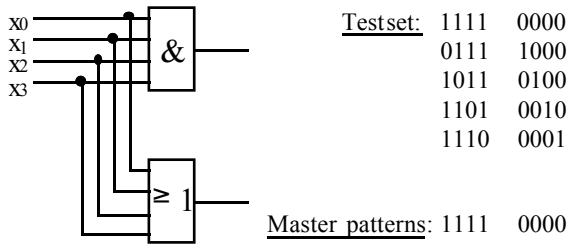


Figure 6: Autocorrelated test patterns

This test set can be generated if the master patterns are repeated several times with different modifications. Obviously, the bit-flipping scheme of figure 5 cannot be used as patterns are not repeated.

2.3. Sequence modifying function

Repeated patterns can be generated if the LFSR is shortened, so that $2^L - 1 < m$. This leads to a scheme where rather a small LFSR generates pseudo-random patterns, some of these patterns are chosen as master and the modification of the master is controlled by the bits $b_0, \ldots, b_{\lceil \log n \rceil - 1}$ of the bit counter and by the bits $p_0, \ldots, p_{\lceil \log m \rceil - 1}$ of the pattern counter. The entire structure is shown in figure 7.

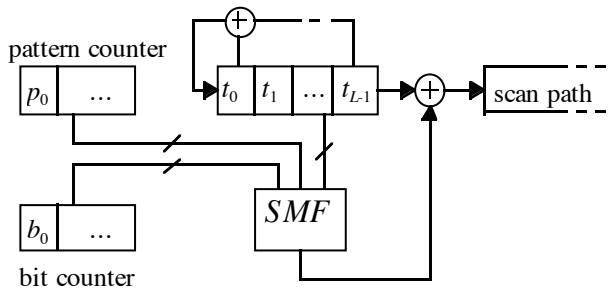


Figure 7: Target Structure

The sequence modifying function *SMF* is completely combinational. The advantages of the target structure in comparison with bit-flipping are twofold:

1. The LFSR is smaller.
2. In general, the sequence modifying function *SMF* is smaller than the bit-flipping function *BFF* as autocorrelation and repetition are exploited.

For the synthesis of *SMF* many degrees of freedom may be used:

1. Selection of the length L of the LFSR.
2. Selection of the feedback polynomial.
3. Selection of LFSR bits t_i, bit counter bits b_i and pattern counter bits p_i as inputs for *SMF*.

In the next section a synthesis procedure is presented, which minimizes the total area of the pattern generator by selecting a short LFSR and a small number of inputs for the SMF so that complete fault coverage is still guaranteed.

3. Synthesis of the pattern generator

The entire synthesis procedure consists of several nested loops. The outermost loop has the LFSR length L as an iteration variable. With increasing L some polynomials are selected, and for each LFSR an SMF is generated by the inner loop.

The sequence modifying function has to map some pseudo-random patterns to the deterministic test set. The function is implemented in two level logic which is represented by a set of cubes. In each step of the inner loop shown in figure 8 the SMF is enhanced so that new deterministic patterns are produced while certain old patterns remain unchanged. The synthesis procedure differs from the synthesis of the bit-flipping logic described in [WuKi96] since the SMF is not generated on the basis of the LFSR states but on a selection of state variables of the LFSR and the test control hardware. For a brief overview of the algo-

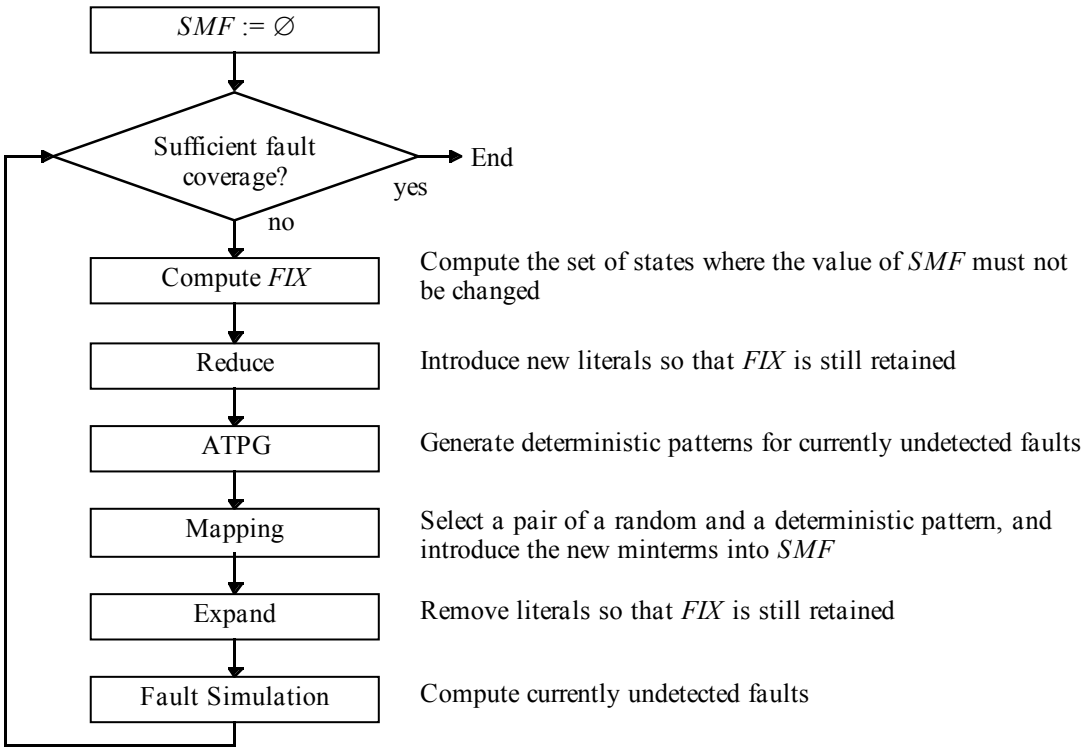


Figure 8: Synthesis of an SMF

rithm, let l denote the number of LFSR bits involved, b the number of bit counter bits, and p the number of pattern counter bits.

In order to improve the SMF, it is necessary to protect patterns which detected some hard faults in former iteration steps. These patterns are called essential, and their number is minimized by fault simulation in several permuted orders. Using three-valued fault-simulation, it is possible to decide which bits of the essential patterns have to be specified. For each output bit of the pattern generator there is a corresponding state in the set $\{0,1\}^l \times \{0,1\}^b \times \{0,1\}^p$. The set of states corresponding to the essential bits is called the fix-set *FIX* and represents the set of inputs for which the output of the SMF must not be changed.

After determining the fix-set, deterministic patterns for the undetected faults are computed so that the number of specified bits is minimized [HRTW95]. In each iteration of the inner loop one ore more of these deterministic patterns are mapped to pseudo-random patterns.

A mapping of a deterministic pattern d to a pseudo-random pattern r is characterized by two state sets. The set *on* (d, r) corresponds to bits that have to be modified, and the set *off* (d, r) corresponds to bits that must not be modified in order to make r compatible with d.

A mapping (d,r) is allowed only if *on* $(d, r) \cap FIX = \varnothing$ holds. Otherwise, a bit that has to remain unchanged would be modified. Among all possible mappings, d is selected so that the number of specified bits is maximum, and r is selected so that the Hamming distance between d and r is minimum.

The fix-set contains the states where the SMF, which has been constructed so far, must not be changed, and its complement is a large don't-care set. The don't-cares are exploited by the ESPRESSO-like procedures "Expand" and "Reduce" [BRAY84] which are executed in each iteration. Furthermore, many random patterns which were neither fixed nor subject of a mapping, may change in a random way. This can cause more previously undetected faults to be detected without requiring any extra hardware.

But sometimes these incidental changes have to be reverted. The best way is to inserting another XOR gate, and the general form of the SMF is shown in figure 9.

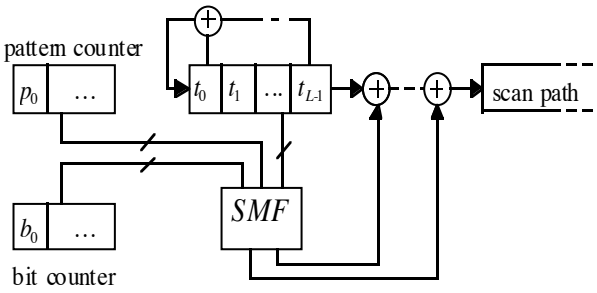
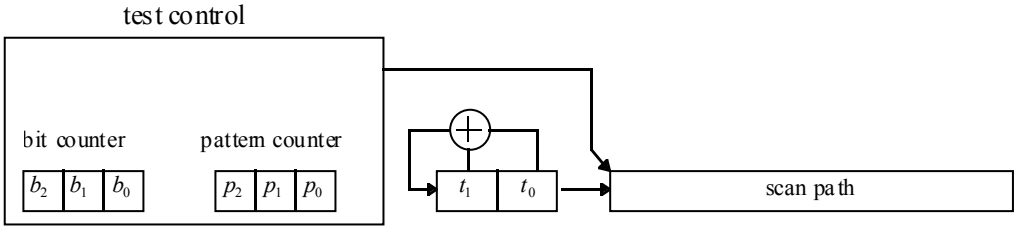


Figure 9: General form of the target structure

4. Example

The synthesis procedure will be illustrated by a small example. Assume that the outermost loop defines the LFSR length $L = 2$ and selects the feedback polynomial $x^2 + x + 1$ with the highest fault coverage for a scan design with scan path length $n = 5$. Let the test length be $m = 6$ patterns. Figure 10 shows the LFSR, the resulting patterns and the corresponding states in the format $b_2b_1b_0.p_2p_1p_0.t_1t_0$.



#	pattern	state sequence				
1	1 0 1 1 0	000.000.01	001.000.10	010.000.11	011.000.01	100.000.10
2	1 1 0 1 1	000.001.11	001.001.01	010.001.10	011.001.11	100.001.01
3	0 1 1 0 1	000.010.10	001.010.11	010.010.01	011.010.10	100.010.11
4	1 0 1 1 0	000.011.01	001.011.10	010.011.11	011.011.01	100.011.10
5	1 1 0 1 1	000.100.11	001.100.01	010.100.10	011.100.11	100.100.01
6	0 1 1 0 1	000.101.10	001.101.11	010.101.01	011.101.10	100.101.11

Figure 10: Example of an LFSR, resulting patterns and state sequences

The LFSR periodically goes through three states (01, 10, 11), and as n = 5 and gcd (3,5) = 1, three different patterns are generated. At a test length of m = 6, each pattern is repeated twice.

Initially, let the fix-set be empty ($FIX = \varnothing$), and assume that the deterministic patterns d_1 = 00010 and d_2 = 00011 have to be generated. Considering the initial pseudo-random pattern sequence, for both d_1 and d_2 at least 2 bits have to be modified. However, as d_1 and d_2 look very similar, there is a solution with only two product terms which is constructed now.

First, d_1 is mapped to pattern #1 which requires the lowest number of bits to be modified. The on- and off-set can be derived from figure 10:

$$on_1 = \{\ 000.000.01,\ 010.000.11\ \}$$

$$off_1 = \{\ 001.000.10,\ 011.000.01,\ 100.000.10\ \}$$

After logic minimization the sequence modifying function $SMF_1 = \{$ --0.---.-1$\}$ covers all minterms of on_1 but none of off_1. The resulting pattern set is shown in table 1. The states for which SMF_1 is active are printed in bold type.

As SMF_1 does not depend on the pattern counter, only three different patterns are generated again, and d_1 occurs twice (#1 and #4). For the next iteration pattern #1 is fixed:

$$FIX_1 = on_1 \cup off_1$$
$$= \{\,000.000.01,\ 010.000.11,\ 001.000.10,\ 011.000.01,\ 100.000.10\ \}$$

Now d_2 = 00011 is mapped. Pattern #1 cannot be mapped to d_2 without being in conflict with FIX_1, so that pattern #4 is selected as only one bit has to be modified. The on- and off-sets can be derived from table 1.

$$on_2 = \{\,100.011.10\ \}$$

$$off_2 = \{\,000.011.01,\ 001.011.10,\ 010.011.11,\ 011.011.01\ \}$$

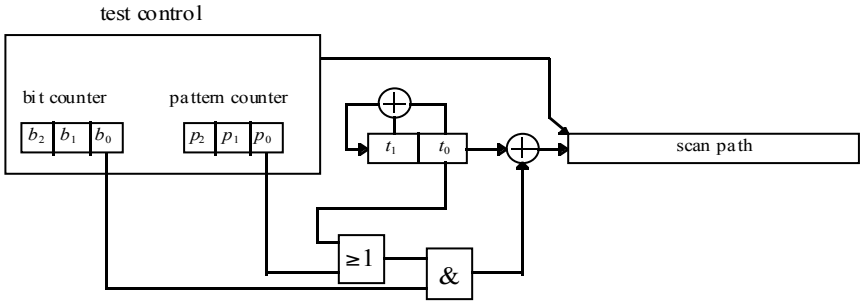
With respect to FIX_1 and off_2, the function represented by on_2 can be minimized to $\{$ --0.--1.-- $\}$ and the final sequence-modifying function is given by

$$SMF_2 = \{\ \text{--0.---.-1},\ \text{--0.--1.--}\ \} = b_0 \cdot (p_0 + t_0)$$

The pattern generator including the modifying logic and the corresponding pattern set, which now contains more than three different patterns, is shown in figure 11.

#	pattern	state sequence				
1	**0** 0 **0** 1 0	**000.000.01**	001.000.10	**010.000.11**	011.000.01	100.000.10
2	**0** 1 0 1 **0**	**000.001.11**	001.001.01	010.001.10	011.001.11	**100.001.01**
3	0 1 **0** 0 **0**	000.010.10	001.010.11	**010.010.01**	011.010.10	**100.010.11**
4	**0** 0 **0** 1 0	**000.011.01**	001.011.10	**010.011.11**	011.011.01	100.011.10
5	**0** 1 0 1 **0**	**000.100.11**	001.100.01	010.100.10	011.100.11	**100.100.01**
6	0 1 **0** 0 **0**	000.101.10	001.101.11	**010.101.01**	011.101.10	**100.101.11**

Table 1: Patterns generated by SMF_1



#	pattern	state sequence				
1	**0** 0 **0** 1 0	**000.000.01**	001.000.10	**010.000.11**	011.000.01	100.000.10
2	**0** 1 **1** 1 **0**	**000.001.11**	001.001.01	**010.001.10**	011.001.11	**100.001.01**
3	0 1 **0** 0 **0**	000.010.10	001.010.11	**010.010.01**	011.010.10	**100.010.11**
4	**0** 0 **0** 1 **1**	**000.011.01**	001.011.10	**010.011.11**	011.011.01	**100.011.10**
5	**0** 1 0 1 **0**	**000.100.11**	001.100.01	010.100.10	011.100.11	**100.100.01**
6	**1** 1 **0** 0 **0**	**000.101.10**	001.101.11	**010.101.01**	011.101.10	**100.101.11**

Figure 11: Pattern generation with SMF_2

5. Experiments

A series of experiments has been performed with benchmark circuits from ISCAS-85 and the combinational versions from ISCAS-89 [BRGL85, BRGL89a]. Only those circuits which still have undetected non-redundant stuck-at faults after applying 10,000 random patterns were analyzed. The area of the LFSR and the PLA implementation of the SMF was determined by using a 1 micron standard cell library and a PLA generator.

The first experiments investigated how the synthesis procedure takes advantage of the autocorrelation of test patterns and shortens the LFSR. Figure 12 draws the entire area of the PG in mm^2 versus the LFSR length if 100% fault coverage is required. Fault coverage is always computed with respect to all non-redundant faults.

The best results are always obtained for LFSR lengths which are much smaller than those required for pseudo-random testing, and obviously, the algorithm takes advantage of pattern repetition.

The second experiments investigated the area required for complete fault detection. The first two columns of table 2 are the circuit name and the length n of the scan path. As comparisons we include the area for the reseeding approach [HRTW95] and for bit-flipping [WuKi96] in the next two columns.

Then the results for the new approach are listed, first the LFSR length L, followed by the number of XORs for sequence reverting, the number of product terms and the total area for these devices. This area is compared to the area of a 32-bit LFSR. For nearly half

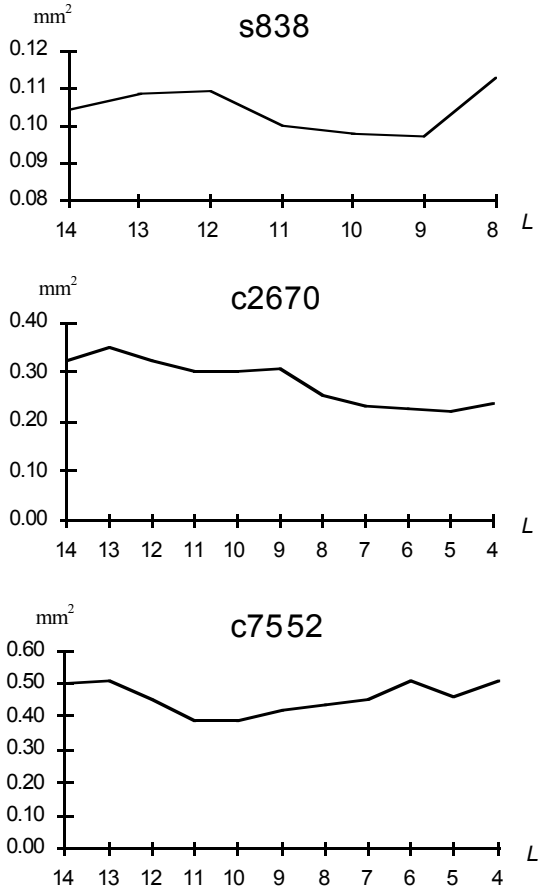


Figure 12: Efficiency of the scheme depending on the LFSR size L

Circuit	n	Reseeding [mm^2]	Bit-Flipping [mm^2]	LFSR length	XORs	Product terms	Area [mm^2]	% of LFSR-32	% of scan path
s420	34	0.344	0.063	13	2	10	0.057	64.8%	43.5%
s641	54	0.344	0.063	9	2	10	0.052	59.7%	25.2%
s713	54	0.344	0.063	11	1	9	0.051	58.2%	24.6%
s838	66	0.533	0.100	11	2	35	0.090	102.0%	35.3%
s953	45	0.308	0.063	13	1	4	0.050	57.5%	29.2%
s1196	32	0.335	0.067	14	2	6	0.057	64.8%	46.2%
s1238	32	0.332	0.063	13	2	8	0.057	64.6%	46.1%
s5378	214	0.423	0.081	14	2	17	0.078	88.6%	9.5%
s9234	247	0.944	0.544	14	3	193	0.448	510.0%	47.2%
s13207	700	0.730	0.193	14	3	60	0.158	179.7%	5.9%
s15850	611	0.918	0.331	14	3	237	0.327	371.8%	13.9%
s38417	1664	1.896	1.733	14	3	829	1.492	1698.2%	23.3%
s38584	1464	0.770	0.577	14	3	102	0.294	334.1%	5.2%
c2670	157	0.734	0.279	5	3	141	0.220	250.3%	24.5%
c7552	206	0.987	0.517	11	3	295	0.384	437.0%	48.2%

Table 2: Area required for complete fault coverage

of the circuits it is possible to obtain complete fault coverage at less costs than required for a pseudo-random BIST by a 32-bit LFSR which still leaves faults undetected. The total area is also compared with the area of the scan path. In any case only a fraction of the scan path area has to be used for implementing the PG. Especially for large circuits, the additional overhead to the scan design is very low. It should be noted that this design method does not require reordering of the scan chain or additional test points.

The next two experiments compare the efficiency of a pseudo-random BIST [HWH96] and the presented approach. The first two columns of table 3 show the pseudo-random fault-coverage and the fault coverage of the new approach if the size of the SMF is limited by the area of a 32-Bit LFSR and the synthesis procedure for the SMF is stopped before complete fault coverage is reached. The last column "Area" gives the area of the PG as a percentage of a 32-Bit LFSR area if the synthesis is stopped after reaching the fault coverage of the pseudo-random BIST.

Circuit	random FC	SMF FC	Area [% of LFSR-32]
s838	66.92%	95.92%	41.6%
s9234	90.63%	91.51%	66.4%
s13207	93.83%	96.38%	43.8%
s15850	94.58%	97.25%	43.8%
s38417	93.41%	93.62%	46.9%
s38584	98.71%	98.93%	43.8%
c2670	88.26%	89.19%	76.0%
c7552	96.29%	97.05%	78.2%

Table 3: Efficiency of the LFSR BIST and the SMF BIST

In any case the PG including an SMF needs less hardware than a 32-bit LFSR and obtains better or even complete fault coverage. Hence, we have the surprising result that a deterministic BIST scheme requires less hardware than an LFSR but reaches higher fault coverage for all the benchmark circuits.

6. Conclusions

A deterministic scan-based BIST scheme may use the information from BIST control so that the overall hardware required for BIST implementation is less than the area of a 32-bit LFSR.

The presented BIST scheme does not touch the mission logic, does not need test point insertion, and does not require reordering the scan chain.

For all benchmark circuits complete fault coverage can be obtained at a cost which is only a fraction of the cost of the scan path.

References

[ABF90] M. Abramovici, M. A. Breuer, A. D. Friedman: "Digital Systems Testing and Testable Design", Computer Science Press, 1990

[Akers85] S. B. Akers: "On the use of Linear Sums in Exhaustive Testing", Proc. Of the 15th Int. Symp. On Fault-Tolerant Computing, 1985, pp. 148-153

[AvMc93] L. Avra, E. J. McCluskey: "Synthesizing for scan dependence in Built-in Self-Testable Designs", Proc. Int. Test Conf., 1993, pp. 734-743

[BCR83] Z. Barzilai, D. Coppersmith, A. L. Rosenberg: "Exhaustive Generation of Bit Patterns with Applications to VLSI Self-Testing", IEEE Transactions on Computers, Vol. C-32, No. 2, Feb. 1983, pp. 190-194

[BaMc84] P. H. Bardell, W. H. McAnney: "Parallel Pseudo-random Sequences for Built-In Test", Proc. Int. Test Conf., 1984, pp. 302-308

[BMS87] P. Bardell, W. H. McAnney, J. Savir: "Built-in Test for VLSI", Wiley-Interscience, New York, 1987

[BRAY84] R. K. Brayton, G. D. Hachtel, C. McMullen, A. Sangiovanni-Vincentelli: "Logic Minimization Algorithms for VLSI Synthesis", Boston: Kluwer Academic Publishers, 1984

[BRGL85] F. Brglez, H. Fujiwara: "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", Proc. Int. Symp. On Circuits and Systems, 1985, pp. 663-698

[BRGL89a] F. Brglez, D. Bryan, K. Komzminski: "Combinational Profiles of Sequential Benchmark Circuits", Proc. Int. Symp. On Circuits and Systems, 1989, pp. 1929-1934

[BRGL89b] F. Brglez et al.: "Hardware-Based Weighted Random Pattern Generation for Boundary-Scan", Proc. Int. Test Conf., 1989, pp. 264-274

[CKS88] G. L. Craig, C. R. Kime, K. K. Saluja: "Test Scheduling and Control for VLSI Built-In Self-Test", IEEE Transactions on Computers, Sep. 88, pp. 1099-1109

[EiLi83] E. B. Eichelberger, E. Lindbloom: "Random Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Tet", IBM Journal of Research and Development, Vol. 27, No. 3, May 1983, pp. 265-272

[HaKr95] O. F. Haberl, T. Kropf: "HIST: A Hierarchical Self Test Methodology for Chips, Boards and Systems", Journal of Electronic Testing: Theory and Applications, 6/1995, pp. 85-106

[HaWu89] O. Haberl, H.-J. Wunderlich: "The Synthesis of Self-Test Control logic", Proc. COMPEURO, May 8-12, 1989, Hamburg

[HELL92] S. Hellebrand, S. Tarnick, J. Rajski, B. Courtois: "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", Proc. Int. Test Conf., 1992, pp. 120-129

[HRTW95] S. Hellebrand, B. Reeb, S. Tarnick, H.-J. Wunderlich: "Pattern Generation for a Deterministic BIST Scheme", Proc. Int. Conf. on Computer-Aided Design, 1995, pp. 88-94

[HWH90] S. Hellebrand, H.-J. Wunderlich, O. F. Haberl: "Generating Pseudo-Exhaustive Vectors for External Testing", Proc. IEEE Int. Test Conf., 1990, pp. 670-679

[HWH96] S. Hellebrand, H.-J. Wunderlich, A. Hertwig: "Mixed-Mode BIST Using Embedded Processors", Proc. IEEE Int. Test Conf., 1996, pp. 195-204

[KMZ79] B. Koenemann, J. Mucha, G. Zwiehoff: “Built-In Logic Block Observation Techniques”, Proc. of International Test Conference, 1979

[LeBl84] J. LeBlanc: "LOCST: A Built-In Self-Test Technique", IEEE Design & Test of Computers, Vol. 1, No. 4, 1984, pp. 42-52

[LiNi97] R. Lidl, H. Niederreiter: "Finite fields", 2nd ed. , New York: Cambridge University Press, 1997

[PaRa91] S. Pateras, J. Rajski: "Generation of Correlated Random Patterns for the Complete Testing of Synthesized Multi-level Circuits", Proc. 28th ACM/IEEE Design Autom. Conf., 1991, pp. 347-352

[PeWe72] W. W. Peterson, E. J. Weldon: "Error-Correcting Codes", MIT Press, Cambridge 1972

[StWu90] A. Ströle, H.-J. Wunderlich: "Error Masking in Self-Testable Circuits", Proc. Int. Test Conf., 1990, pp. 544-552

[StWu91] A. Ströle, H.-J. Wunderlich: "TESTCHIP: A chip for weighted random pattern generation, evaluation, and test control", IEEE Journal of Solid State Circuits, July 1991, Vol. 26, No. 7, pp. 1056-1063

[StWu94] A. Ströle, H.-J. Wunderlich: "Configuring Flipflops to BIST Registers", Proc. Int. Test Conf., 1994, pp. 939-948

[ToMc96] N. A. Touba, E. J. McCluskey: "Altering a pseudo-random bit sequence for scan-based BIST", Proc. Int. Test Conf., 1996, pp.167-175

[Wu87] H.-J. Wunderlich: “Self Test Using Unequiprobable Random Patterns”, Proc. 17th In. Symp. Fault-Tolerant Comput., Pittsburgh 1987, pp. 258-263

[WuKi96] H.-J. Wunderlich, G. Kiefer: "Bit-Flipping BIST", Proc. Int. Conf. On Computer-Aided Design, 1996, pp. 337-343

[WuZo97] H.-J. Wunderlich, Y. Zorian: "Built-In Self Test (BIST): Synthesis of Self-Testable Systems", Kluwer Academic Publishers, 1997