Deterministic BIST with Multiple Scan Chains

Gundolf Kiefer Hans-Joachim Wunderlich

Computer Architecture Lab
University of Stuttgart

Abstract

A deterministic BIST scheme for circuits with multiple scan paths is presented. A procedure is described for synthesizing a pattern generator which stimulates all scan chains simultaneously and guarantees complete fault coverage.

The new scheme may require less chip area than a classical LFSR-based approach while better or even complete fault coverage is obtained at the same time.

Keywords: *deterministic scan-based BIST, multiple scan paths, parallel scan*

1. Introduction

Built-in self-test (BIST) is one of the most important techniques for testing large and complex circuits. The efficiency of a BIST implementation is characterized by the test application time and the hardware overhead required to achieve complete or sufficiently high fault coverage.

In a "test-per-scan" scheme, the storage elements of the circuit under test (CUT) are enhanced to a scan path [EiLi83]. A pattern generator produces one or several bit sequences that are shifted into the scan path(s) and a signature register compacts the responses of the circuit.

The test application time depends on both the number of test patterns and the number of clock cycles required to shift in a single pattern. If all flipflops of the CUT are assigned to a single scan path, the test application time may become too long. This problem can be solved by either using a partial scan path so that only a small subset of flipflops is connected to a scan chain, or by using several parallel scan chains [FSSG85, Derv89, NGB93]. If the CUT contains multiple parallel scan chains, a parallel pattern generator and a multiple input signature register (MISR) are required. Figure 1 shows the STUMPS architecture [BaMc84, BaMc86, BMS87] which is based on a pseudo-random pattern generator (PRPG).

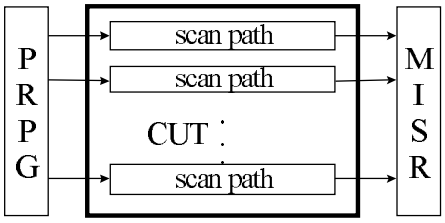

Figure 1: The STUMPS architecture

If the pattern generator is implemented by an LFSR, the bit sequences shifted into different scan chains may only differ by a small phase shift (see figure 2).

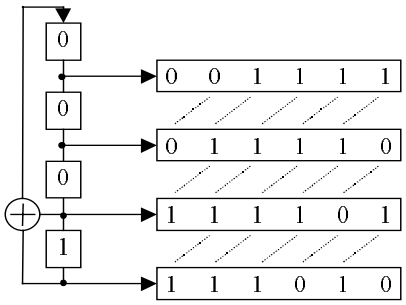

Figure 2: LFSR generating multiple bit sequences

Often, these correlations reduce the fault coverage considerably compared to an architecture with a single scan path [BMS87, Bard92]. So the PRPG of figure 1 is usually implemented by the combination of an LFSR and a phase shifting logic that transforms the LFSR outputs into several uncorrelated signals [BaMc86, Bard90, RaTy98] (see figure 3).

However, even uncorrelated random patterns cannot guarantee complete fault coverage if a circuit contains random-pattern-resistant faults. Several schemes have been proposed for detecting these faults by applying weighted random patterns [BRGL89b, Wu87, StWu91], pseudo-exhaustive patterns [Akers85,

Part of this work has been supported by the DFG under grant Wu 245/1-1.

HWH90] or deterministic patterns [Koen91, HELL92, HELL95, ToMc96, WuKi96, ZRTW96, KiWu97]. Most of the deterministic schemes are designed for single scan path architectures. Multiple scan chains are addressed in the schemes of [Koen91] and [ZRTW96] which are based on encoding deterministic patterns by seeds for an LFSR. In this paper it is shown that a pattern generator for multiple scan paths can be synthesized with a distinctly smaller hardware overhead.

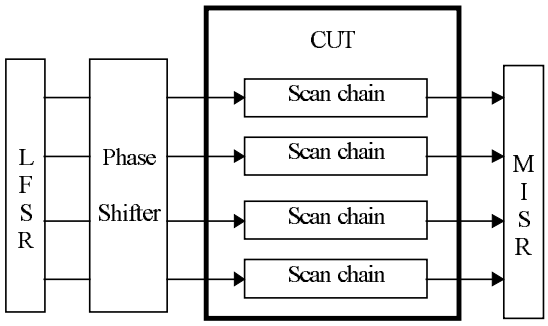

Figure 3: Generation of uncorrelated signals for multiple scan chains by phase shifting

In section 2 we introduce the structure of a pattern generator for multiple scan chains that can guarantee complete fault coverage. In section 3 an automatic procedure for synthesizing the pattern generator for a given CUT is described. Finally, results of this procedure are presented in section 4.

2. Target Structure

The construction of the pattern generator is based on the following observations:

1. Given a set of pseudo-random patterns, deterministic patterns can easily be embedded by modifying just a very small number of bits. A probabilistic analysis is given in [WuKi96].
2. In a pseudo-random test set only a very small number of patterns contribute to the fault coverage, and within these patterns only a few bits need to be specified.
3. Very often, deterministic test patterns can be clustered into a few sets such that all the patterns of a set look very similar [PaRa91].
4. Every autonomous BIST scheme must contain a test control unit containing a bit counter and a pattern counter for generating the shift/capture signal and the "testend" signal. This has been exploited in a BIST scheme for single scan chains in [KiWu97].

The target structure is shown in figure 4. It consists of an LFSR and a combinational function, the Sequence-Generating Logic (SGL). The SGL passes the bit sequences generated by the LFSR to the scan paths, and modifies these sequences at certain bit positions which are selected by both the state of the test control unit and the state of the LFSR.

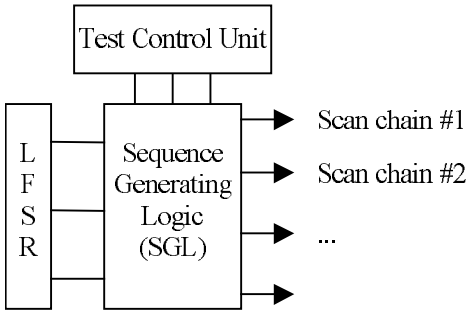

Figure 4: Target Structure

The structure of the SGL is shown in figure 5. It consists of some XOR gates and the Sequence-Modifying Logic SML. The current state of the LFSR and the bit counter and the pattern counter of the test control unit serve as inputs for the SML.

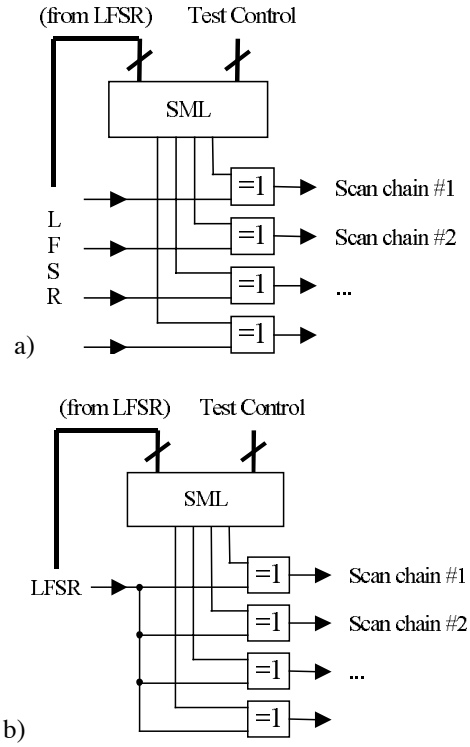

Figure 5: Examples of the Sequence-Generating Logic

As in a pseudo-random test set only a few bits are really necessary for detecting faults and as deterministic patterns can easily be embedded by modifying just a very small number of bits (observations 1 and 2), the SML can be minimized very efficiently. Furthermore, since the outputs of the pattern generator also depend on the current state of the test control unit, the LFSR may be very small. This does not only reduce the chip area of the LFSR, but also the SML will be smaller, since some LFSR patterns are repeated several time during the test, and autocorrelated deterministic patterns (observation 3) can be embedded efficiently.

As the SML can overcome dependencies between different LFSR outputs, no phase shifting is required and the same pseudo-random sequence (generated by a single output of the LFSR) can be used for all the scan chains (see figure 5b). In some cases, using a single pseudo-random sequence reduces the random pattern fault coverage so that a large SML is required. In that case different LFSR outputs and - if necessary - linear combinations of them are used to generate the basic pseudo-random sequences for different scan chains (see figure 5a and figure 6).

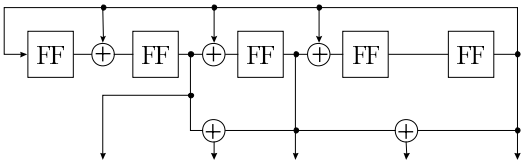

Figure 6: Generating multiple pseudo-random sequences

The signals are selected as follows: It is required that the LFSR is implemented in a modular way, so that its XOR gates are inserted between the flipflops. Then among the flipflop outputs a set of primary signals is selected, so that there is always at least one XOR gate between two primary signals. These signals are connected to scan paths. If there are more scan chains than primary signals, the other scan chains are fed with linear combinations of primary signals. The linear combinations are chosen in a way that the required number of additional XOR gates is minimized.

3. Synthesizing the Sequence-Modifying Logic

The Sequence-Modifying Logic SML is constructed iteratively. The main loop of the algorithm is sketched in figure 7. The algorithm starts with an empty SML and terminates when sufficient fault coverage is achieved. In each iteration, the SML is enhanced, so that new deterministic patterns are produced while certain essential bits remain unchanged. In order to achieve an efficient implementation of the SML, the ESPRESSO-like logic minimization procedures "Expand" and "Reduce" [BRAY84] are integrated into the main loop of the algorithm.

The following subsections describe the main steps of the algorithm in detail.

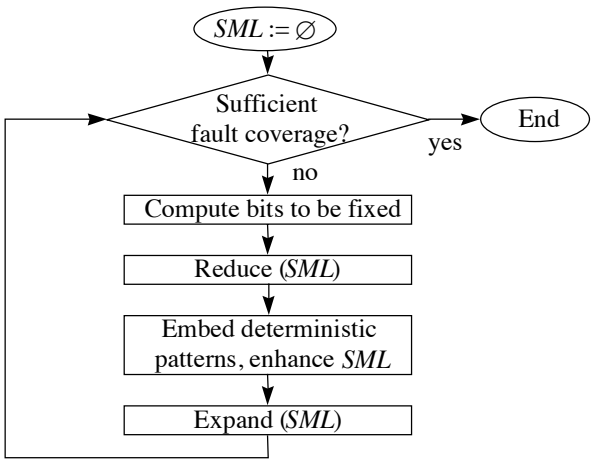

Figure 7: Synthesis of an SML

3.1. Computing bits to be fixed

In order to improve the SML, it is necessary to protect patterns which detected some hard-to-detect 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 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 of the LFSR and the test control unit. Let n be the number of scan chains. For every chain i, $1 \le i \le n$, the set of states corresponding to the essential bits of the i-th scan chain is called the i-th fix-set FIX_i.

For example, figure 8 illustrates how a deterministic pattern is shifted into 4 parallel scan chains by a 3-bit-LFSR. Each output bit is tagged with the corresponding state of the LFSR and the test control unit. For the sake of simplicity, the test control bits are omitted in figure 8.

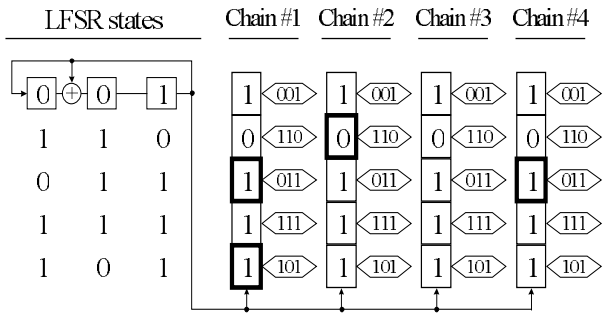

Figure 8: Pattern bits and corresponding states

Now assume the marked bits in figure 8 are essential. Then the fix-sets are:

$FIX_1 = \{ 011, 101 \}$
$FIX_2 = \{ 110 \}$
$FIX_3 = \{ \}$
$FIX_4 = \{ 011 \}$

3.2. Embedding deterministic patterns

An automatic test pattern generator (ATPG) is used to generate deterministic patterns for all the currently undetected faults, and one or several of these patterns are selected for embedding. In order to obtain small on- and off-sets, the ATPG tool should minimize the number of specified bits [HELL95].

Let n be the number of scan chains, and let m be the length of the longest scan chain. A mapping of a deterministic pattern $d \in (\{0,1,-\}^m)^n$ to a pseudo-random pattern $r \in (\{0,1\}^m)^n$ is characterized by $2n$ state sets. For each $1 \le i \le n$, the set $ON_i(d, r)$ corresponds to the bits of scan chain i that have to be changed, and $OFF_i(d, r)$ corresponds to the bits that must not be changed. For example, figure 9 shows a pseudo-random pattern, a deterministic pattern and the resulting on- and off-sets.

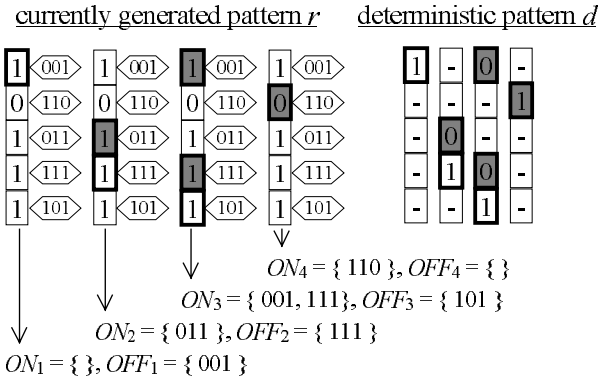

Figure 9: Embedding a deterministic pattern

Deterministic patterns with only a few specified bits correspond to faults that are comparably "easy to detect". Due to the integrated logic minimization they might be detected by random patterns in some later iteration of the algorithm. So a deterministic pattern d with a maximum number of specified bits is selected for embedding.

Given a deterministic pattern d, a currently generated pattern r is selected, such that the fix-sets and the on-sets are disjoint for each $1 \le i \le n$ and the expected increase of hardware overhead is minimized. The increase of hardware overhead can be approximated by the total number of states in all on-sets. Better results are obtained if a more accurate cost function is used which tries to predict the effectiveness of the integrated logic minimization procedures (see section 3.3).

Finally, the SML is modified, so that its function changes for the inputs represented by the on-sets, and for each scan path i, the i-th on-set and the i-th off-set are added to the i-th fix-set.

3.3. Logic minimization

In order to achieve an efficient implementation of the SML, the ESPRESSO-like procedures "Reduce" and "Expand" [BRAY84] are integrated into the algorithm (see figure 7). Given a set of product terms representing a single-output function F and a fix-set FIX, "Reduce" removes as many don't cares as possible in each of the product terms of F. Furthermore, terms that do not cover any element of the fix-set are removed completely.

"Expand" adds as many don't cares as possible so that FIX is still retained. The positions of the don't cares are selected so that terms are merged if this is possible, e. g. Expand ({100, 110}) = { 1-0 }.

Usually, FIX only contains a few minterms and its complement is a very large don't-care set which can be exploited effectively. As the SML has several outputs it is necessary to implement a bundle minimization that is also able to exploit such a large don't care set.

In addition to the minimization procedures, a cost function is required to estimate the number of new product terms corresponding to a vector of on-sets. The cost function controls the embedding step of the algorithm and the selection of an XOR slot if several XOR gates per scan chain are required as described in section 3.4.

For this reason the following model is used: Each state word is concatenated with the 1-out-of-n code of the scan path number (see figure 10). The vectors of fix-, on- and off-sets described in the previous

subsections can now be represented by one set of cubes each, FIX^*, OFF^* and ON^*, where each cube consists of a basic part representing the SML inputs and an extension representing the scan path number.

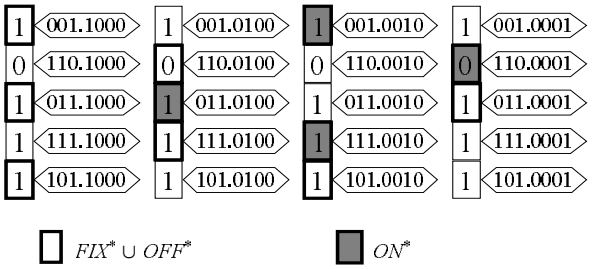

Figure 10: Cubes for bundle minimization

The SML is also represented by a set SML^* of cubes in which each literal of the extension can be either '0' or '-'. Let $c \in SML^*$ be a cube and let $m \in FIX^* \cup OFF^* \cup ON^*$ be a minterm originating from $FIX_i \cup OFF_i \cup ON_i$, $1 \le i \le n$. The cube c covers m if and only if the basic part of c covers the basic part of m and the i-th literal of the extension of c is '-'.

In return, for each scan chain i, the subset of cubes that are contained in its function SML_i is given by

$$SML_i = \{\, c \in SML^* \mid \text{literal } i \text{ of the ext. of } c \text{ is '-'} \,\}.$$

Now the single-output implementations of "Expand" and "Reduce" can be used without modifications and they minimize the total number of product terms. Furthermore, a new minterm $c \in ON^*$ can be merged with an existing cube $c_0 \in SML^*$ by "Expand" if

$$\left(FIX^* - \{c, c_0\}\right) \cap Expand(c, c_0) = \varnothing ,$$

where $Expand(c, c_0)$ denotes the smallest boolean subspace covering both c and c_0. This condition can easily be checked and used to estimate the cost for a pattern mapping as described in section 3.2.

For the example of figure 10, "Expand" will return

$$SML^* = \{11\text{-}.\text{-}0\text{--}, 0\text{--}.0\text{--}0\}$$

as a minimal cover of the set ON^*. The implementation of SML^* and the resulting pattern are shown in figure 11.

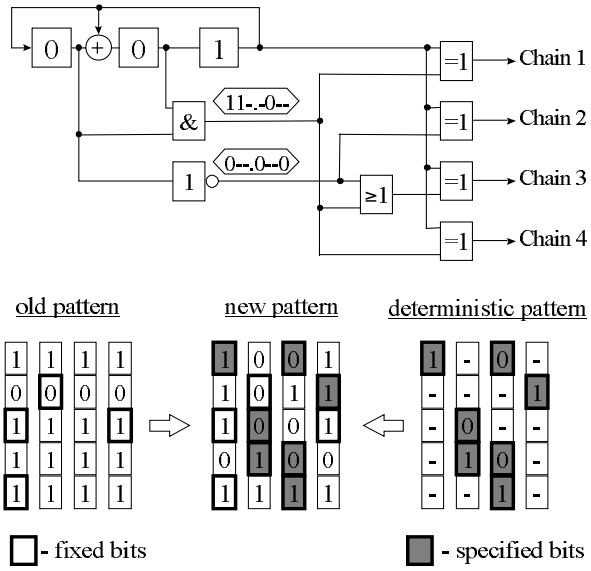

Figure 11: Example for a pattern generator and the resulting pattern

Each of the terms 0-- and 11- is used to change two bits in different scan chains. Furthermore, 11- changes two bits in chain 1 without any extra cost. This may cause detecting more previously undetected faults.

3.4. Reflipping

Due to the logic minimization, many random bits that were neither fixed nor subject of a mapping, may change in a random way. Sometimes these incidental changes have to be reverted. The best way to do so is to allow several XOR gates to be inserted between the LFSR and the scan paths. The general form of the sequence generating logic is shown in figure 12.

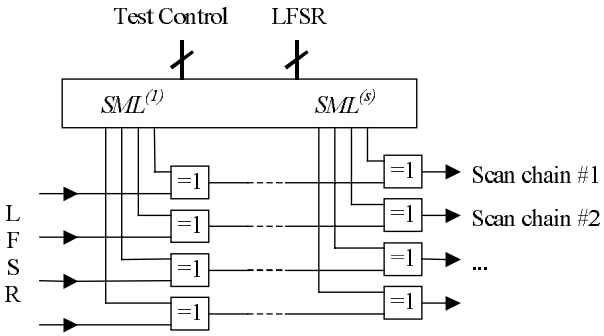

Figure 12: General form of the Sequence Generating Logic

Instead of a single set of cubes SML^*, the logic is specified by a vector of cube sets ($SML^{(1)}$, ..., $SML^{(s)}$) where each set represents one of the s XOR slots.

Let m be a minterm corresponding to a bit to be modified. For m, an XOR slot i is selected as follows:

1. Select $i \in \{ 1, ..., s \}$, such that $SML^{(i)}$ does not cover m, but there is a $c \in SML^{(i)}$ such that "Expand" can merge m and c to a single cube.
2. If 1 is not possible, select an $SML^{(i)}$ which does not cover m.
3. If 2 is not possible, introduce a new XOR slot s+1 and set $SML^{(s+1)} := \{ m \}$.

Experiments have shown that usually not more than three XOR slots are required. Furthermore, not all the XOR gates sketched in figure 12 have to be implemented as reflipping may only be required for some scan chains.

4. Experimental Results

A series of experiments was performed with benchmark circuits from ISCAS-85 and ISCAS-89 [BRGL85, BRGL89a]. Only those circuits were analyzed which have more than 50 flipflops and primary inputs and still have undetected non-redundant faults after applying 10,000 random patterns.

The chip area of the pattern generator is estimated by using a 1 µm standard cell library, and the SML is implemented by one or several PLA macro cells.

Into each circuit, several parallel scan chains of equal length with at most 51 flipflops have been inserted randomly. Table 1 shows the number of pseudo-primary inputs PPIs (= number of primary inputs + number of flipflops), the number of scan chains inserted and the number of clock cycles needed to apply a single pattern. The rightmost column of table 1 shows the test application time in relation to the time required for a test based on a single scan chain.

Name	PPIs	Chains	Clocks per pattern	Test time / time for single scan
c2670	233	5	47	21%
c7552	207	5	44	22%
s641	54	2	28	53%
s713	54	2	27	51%
s838	66	2	37	57%
s5378	214	5	43	20%
s9234	247	5	50	21%
s13207	700	14	50	7%
s15850	611	13	47	8%
s38417	1664	32	52	3%
s38584	1464	30	49	3%

Table 1: Inserting multiple scan chains

The first series of experiments investigated the size of the sequence generating logic required to achieve complete fault coverage. We varied the LFSR size, the LFSR polynomial and the way the pseudo-random sequences are generated (single sequence / multiple sequences, see figure 5). Furthermore, in order to minimize the width of the PLA, we restricted the inputs of the SML to a subset of all LFSR and test control signals. If a multi-level synthesis tool is used, this restriction is not necessary.

Table 2 describes the smallest SGL we obtained for each benchmark circuit. The column "LFSR" denotes the length of the LFSR. The next column shows if the sequence modification is based on a single bit sequence or on multiple sequences. Then the total number of XOR gates ("XORs"), the number of inputs ("ins"), the number of outputs ("outs") and the number of product terms ("terms") of the sequence modifying logic follow. The column "XORs" includes all XOR gates for the LFSR, for the sequence decorrelation (if different pseudo-random sequences are generated) and for attaching the sequence-modifying logic. For comparison, the number of product terms for a single scan path architecture is shown in the rightmost column. These numbers are based on the same number of input lines for the SML (except for s38584).

For all large circuits, the number of product terms for multiple scan chains is considerably smaller than in the case of a single scan chain. If a multi-level synthesis tool is used, a smaller number of product terms usually results in smaller chip area. Thus in consequence the pattern generator is smaller if the circuit under test contains multiple scan chains instead of a single one.

	Multiple Scan						Single Scan
Circuit	LFSR	PR seq.	XORs	ins	outs	terms	terms
c2670	9	single	17	14	10	58	81
c7552	14	single	23	14	12	249	323
s641	10	multiple	10	13	3	8	9
s713	9	single	9	13	2	11	7
s838	8	multiple	9	14	4	56	48
s5378	12	multiple	17	14	8	44	18
s9234	14	single	21	14	10	187	397
s13207	14	multiple	34	18	18	56	77
s15850	14	single	34	14	22	199	214
s38417	14	multiple	102	34	68	509	590
s38584	14	multiple	80	14	46	132	361*

* 32 PLA inputs

Table 2: Results of the SGL BIST for complete fault coverage

However, in order not to be biased by the quality of logic synthesis tools, we measured the chip area of

the pattern generator (PG) based on a PLA implementation of the SML. Table 3 shows the chip area for the multiple scan pattern generator and compares it to the area of a single scan pattern generator. The columns "LFSR", "PLAs" and "XORs" denote the area required for the LFSR flipflops, the PLAs and the XOR gates. The sum of these numbers is given in the next column. The rightmost column shows the total area for a single scan PG.

	Multiple Scan				Single Scan
Circuit	LFSR	PLAs	XORs	Total	Total
c2670	0.024	0.103	0.022	0.149	0.145
c7552	0.038	0.345	0.030	0.413	0.428
s641	0.027	0.029	0.013	0.069	0.063
s713	0.024	0.031	0.012	0.067	0.067
s838	0.021	0.087	0.012	0.120	0.101
s5378	0.032	0.080	0.022	0.134	0.096
s9234	0.038	0.266	0.027	0.331	0.494
s13207	0.038	0.134	0.044	0.216	0.195
s15850	0.038	0.332	0.044	0.414	0.308
s38417	0.038	1.685	0.133	1.856	1.370
s38584	0.038	0.313	0.105	0.456	0.851

Table 3: Area in μm^2 required for complete fault coverage (1 µm technology)

For some benchmarks circuits the multiple scan PG requires more area than the single scan PG even if the number of product terms is smaller. This is mainly due to the large number of outputs causing a large OR-array if a PLA implementation is used. However, for the circuit s38417 e. g., a multiple scan PG is approximately 35% larger than a single scan PG but the test application time is reduced by about 97%.

Table 4 shows how the area of the pattern generator depends on the basic pseudo-random sequences. For some circuits a single random sequence for all scan chains yielded much better results (e. g. c2670, c7552, s9234), for some circuits it is favorable to generate multiple pseudo-random sequences (e. g. s5378, s13207).

Circuit	Single sequence	Multiple sequences
c2670	0.149	0.191
c7552	0.413	0.447
s641	0.072	0.069
s713	0.067	0.073
s838	0.121	0.120
s5378	0.233	0.134
s9234	0.331	0.476
s13207	0.373	0.216
s15850	0.414	0.426

Table 4: Results with / without decorrelation

We also compared the efficiency of the SGL scheme with a pseudo-random scheme based on a 32-Bit LFSR and a sequence decorrelation as described in section 2 and shown in figure 6. For this reason we stopped the SML synthesis after reaching the fault coverage of the pseudo-random BIST. The results are shown in table 5.

	SGL		LFSR-32	
Name	Fault efficiency	Area	Fault efficiency	Area
c2670	90.4	0.061	88.2	0.125
c7552	96.6	0.101	96.3	0.125
s838	68.7	0.049	61.7	0.125
s5378	99.0	0.119	92.8	0.125
s9234	90.6	0.102	90.5	0.125
s13207	93.3	0.055	92.3	0.125
s15850	94.4	0.124	92.2	0.125
s38417	93.8	0.154	93.5	0.129
s38584	98.4	0.133	98.4	0.126

Table 5: Efficiency of the SML BIST and the LFSR BIST

In most cases, the area of the SGL scheme is considerably smaller than the area of the 32-bit LFSR. In two cases (s38417 and s38584) the area is slightly larger due to the PLA implementation which is less efficient than standard cell implementations especially for small SMLs. In general, the SGL scheme allows higher fault coverage than a pseudo-random BIST but requires less chip area at the same time.

5. Conclusions

A synthesis procedure for a deterministic BIST scheme has been presented for supporting multiple scan chains.

The scheme is scalable with respect to hardware overhead and fault coverage. On the one hand complete fault coverage can be guaranteed if requested. On the other hand for obtaining better fault coverage than the classical LFSR-based STUMPS architecture less hardware is required.

Furthermore, it is not necessary to modify the mission logic or to reconfigure the scan chains, so there is practically no additional effect on the system performance besides the scan chain.

References

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

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

[BaMc86] P. H. Bardell, W. H. McAnney: "Pseudo-random arrays for built-in tests", IEEE Trans. Comp., vol. C-35, No. 7, 1986, pp. 653-658

[Bard90] P. H. Bardell: "Design Considerations for parallel pseudo-random pattern generators", Journal of Electronic Testing: Theory and Applications (JETTA), vol. 1, No. 1, 1990, pp. 73-87

[Bard92] P. H. Bardell: "Calculating the Effects of Linear Dependencies in m-Sequences Used as Test Stimuli", IEEE Trans. on CAD, Jan. 1992, pp. 83-86

[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 (ISCAS), 1985, pp. 663-698

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

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

[Derv89] B. I. Dervisoglu: "Scan-Path Architecture for Pseudorandom Testing", IEEE Des. & Test, Aug. 1989, pp. 32-48

[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

[FSSG85] P. P. Fasang, J. P. Shen, M. A. Schuette, W. A. Gwaltney: "Automated design for testability of semicustom integrated circuits", Proc. Int. Test Conf. (ITC), 1985, pp. 558-564

[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. (ITC), 1992, pp. 120-129

[HELL95] S. Hellebrand, B. Reeb, S. Tarnick, H.-J. Wunderlich: "Pattern Generation for a Deterministic BIST Scheme", Proc. Int. Conf. on Computer-Aided Design (ICCAD), 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. (ITC), 1990, pp. 670-679

[KiWu97] G. Kiefer, H.-J. Wunderlich: "Using BIST Control for Pattern Generation", Proc. IEEE Int. Test Conf. (ITC), 1997, pp. 347-355

[Koen91] B. Koenemann: "LFSR-Coded Test Patterns for Scan Design", Proc. IEEE Int. Test Conf. (ITC), 1991, pp. 237-242

[NGB93] S. Narayanan, R. Gupta, M. A. Breuer: "Optimal Configuring of Multiple Scan Chains", IEEE Trans. on Comp., Sep. 1993, pp.1121-1131

[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. (DAC), 1991, pp. 347-352

[RaTy98] J. Rajski, J. Tyszer: "Design of phase shifters for BIST applications", accepted for the VLIS Test Symp. '98

[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

[ToMc96] N. A. Touba, E. J. McCluskey: "Altering a pseudo-random bit sequence for scan-based BIST", Proc. Int. Test Conf. (ITC), 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 (ICCAD), 1996, pp. 337-343

[ZRTW96] N. Zacharia, J. Rajski, J. Tyszer, J. A. Waicukauski: "Two-Dimensional Test Data Decompressor for Multiple Scan Designs", Proc. Int. Test Conf. (ITC), 1996, pp.186-194