A MIXED MODE BIST SCHEME BASED ON RESEEDING OF FOLDING COUNTERS

Sybille Hellebrand
University of Innsbruck
Austria

Hua-Guo Liang*), Hans-Joachim Wunderlich
University of Stuttgart
Germany

Abstract

In this paper a new scheme for deterministic and mixed mode scan-based BIST is presented. It relies on a new type of test pattern generator which resembles a programmable Johnson counter and is called folding counter. Both the theoretical background and practical algorithms are presented to characterize a set of deterministic test cubes by a reasonably small number of seeds for a folding counter. Combined with classical approaches for test width compression and with pseudo-random pattern generation these new techniques provide an efficient and flexible solution for scan-based BIST. Experimental results show that the proposed scheme outperforms previously published approaches based on the reseeding of LFSRs or Johnson counters.

1 Introduction

Serial or scan-based built-in self-test (BIST) offers an excellent solution for the challenges of today's integrated circuit testing. The built-in capabilities of test pattern generation and test response evaluation allow an efficient test even for externally inaccessible components in complex systems-on-a-chip (SOCs). The classical architecture with an LFSR feeding pseudo-random patterns into the scan path is easy to implement and minimizes both hardware overhead and the impact on the system performance [1]. To overcome the problem of random pattern resistant faults a number of advanced techniques have been proposed in the literature. These approaches range from test point insertion to weighted random pattern and mixed mode testing and offer different trade-offs between fault coverage, hardware overhead, performance degradation and test length.

Mixed mode schemes use a limited number of pseudo-random patterns to eliminate the easy to detect faults and deterministic patterns to cover the remaining random pattern resistant faults [7, 9 - 11, 14, 16, 17]. The deterministic patterns are either stored on chip in a compressed format and expanded during BIST ("store and generate") or directly embedded into an LFSR sequence by "bit-fixing" or "bit-flipping" techniques. It has been demonstrated that bit-fixing and bit-flipping can provide high quality test patterns at low hardware overhead [13, 16, 17]. However, the BIST architecture is extremely tailored to the specific circuit, and a change in the test set requires a resynthesis of the complete BIST hardware. For applications which demand a more flexible BIST architecture the store and generate approaches provide an alternative at comparable cost. Here, deterministic patterns (or groups of patterns) are encoded as seeds of simple test pattern generators as for example LFSRs, multiple polynomial LFSRs, counters or twisted ring (Johnson) counters [4, 9 - 12, 14].

In this paper we present a novel store and generate architecture based on the reseeding of a new and simple type of generator called folding counter. It works with a dynamically changing, but still very regular, state transition function. The implementation for serial BIST resembles a programmable Johnson counter and can easily be extended to a generator which is able to switch between pseudo-random pattern generation and the generation of deterministic patterns from folding seeds. To minimize the storage requirements the basic scheme is combined with classical techniques for test width compression.

The rest of this paper is organized as follows. Section 2 introduces the new generator and develops the target architecture for the presented work. The theoretical background for the characterization of deterministic test sets by folding seeds is given in Section 3. Subsequently, Section 4 describes the complete synthesis procedure for the BIST hardware, and Section 5 reports on the experimental results.

2 Folding counters and deterministic BIST

In contrast to commonly used generators a folding counter works with a dynamically changing state transition function depending on both the state of the counter and the "index" of the transition. It is defined as follows:

Starting from an initial state $s \in \{0, 1\}^n$ a sequence of $n + 1$ states $s = F(0, s), F(1, s), \ldots, F(n, s)$ is produced, such that the transition from $F(i, s)$ to $F(i+1, s)$ retains

*) Hua-Guo Liang is with Hefei University of Technology, China. Currently he is a guest research fellow at the University of Stuttgart, Germany.

the first i bits and inverts the remaining ones. A simple example sequence is shown in Figure 1.

state	register contents	index
F(0, s)	0110	0
F(1, s)	1001	1
F(2, s)	1110	2
F(3, s)	1101	3
F(4, s)	1100	4

Figure 1: Sequence produced by a 4-bit folding counter.

Compared to classical generators the sequence produced from a single initial state (seed) is rather short, and it is very unlikely that a complete set of deterministic vectors can be embedded into a single folding counter sequence. However, in general it is possible to find a reasonably small number of seeds, such that the union of all the resulting sequences covers a given deterministic test set. This supports an efficient deterministic BIST, and in particular for scan-based BIST the basic architecture is very simple. As illustrated in Figure 2, in this case the folding counter can be realized by a Johnson counter with programmable feedback.

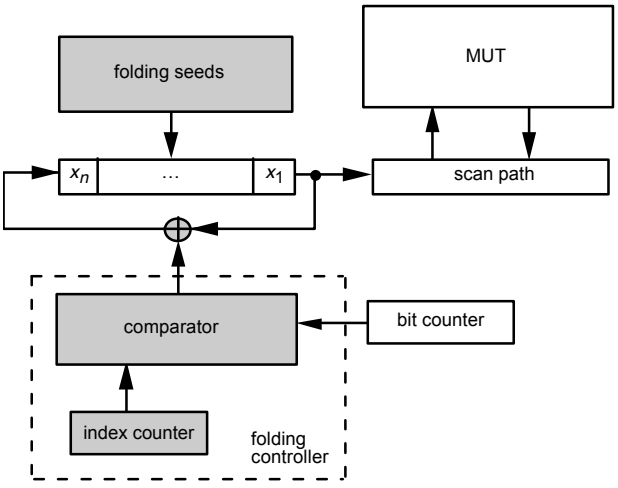


Figure 2: Basic deterministic BIST scheme using a folding counter.

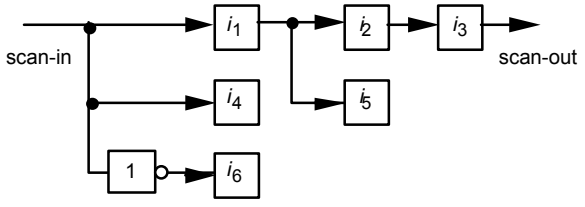
To apply a folding sequence to the MUT a seed is loaded into the shift register, and the index counter and bit counter are initialized. While the first pattern is loaded into the scan chain the Johnson counter serially generates the next state of the folding counter. For each bit the state of the index counter is compared to the state of the bit counter which controls the loading of the scan path. Since for the first pattern the index counter is zero, the feedback function inverts all bits (in the general case the first i bits remain unchanged according to the definition of the folding counter). As soon as the pattern is completely loaded, it is applied to the MUT, the bit counter is reset and the index counter is activated. This procedure is repeated until the index counter has cycled through all states and the next seed can be processed.

It should be noted that the bit counter is required anyway for scan-based BIST and the additional hardware for deterministic pattern generation merely consists of the Johnson counter with programmable feedback and the "folding controller" with index counter and comparator. Despite its simple and regular structure, the basic architecture of Figure 2 has one serious drawback: the shift register must have the same length as the scan chain, which may be unacceptable for larger circuits. To circumvent this problem the technique of (pseudo) input reduction provides a good means [5, 6, 15]. As proposed in [6] the circuit function is analyzed, and sets of "compatible" and "inversely compatible" inputs are identified. Inputs in each set can share a single signal in test mode without sacrificing fault coverage. For deterministic BIST it is then sufficient to remember the value of one test signal for each set of inputs. To expand the patterns to their original format during test application as simply as possible, we propose to implement a slightly modified scan chain.

An analysis of the test set in Figure 3a shows that the columns corresponding to inputs i_1 and i_4 are identical and the column for i_6 is the inverse of the column for i_4. Also the columns for i_2 and i_5 are identical. Therefore the sets of compatible inputs are $\{i_1, i_4, i_6\}$, $\{i_2, i_5\}$ and $\{i_3\}$. The compressed test set consisting of only three columns stores the complete test information, if the scan path is implemented as shown in Figure 3b.

original test set						compressed test set		
i_1	i_2	i_3	i_4	i_5	i_6	i_1	i_2	i_3
1	0	0	1	0	0	1	0	0
1	0	1	1	0	0	1	0	1
1	1	0	1	1	0	1	1	0

a) Input reduction



b) Scan structure

Figure 3: Test set compression by input reduction in scan-based BIST.

However, the depicted scan structure is not suitable for test response evaluation, as only the contents of one scan element for each set of compatible inputs can be shifted out. There are a number of well known solutions for this problem:

1) The scan flip-flops may be appropriately distributed to multiple scan chains as suggested in [6].
2) If a balanced design with multiple scan chains is not possible, the contents of all scan cells for a set of compatible inputs may be collected by an EXOR-tree.
3) If the second alternative results in an unacceptable loss of fault coverage, we propose the extended scan structure sketched in Figure 4. The scan flip-flops are able to distinguish between a regular and a compressed shift mode. The regular mode, where the scan chain behaves as a conventional scan chain, can be used to shift in uncompressed patterns and to shift out test responses. In compressed shift mode the dotted connections are active and compressed patterns are automatically expanded without any additional control logic. The proposed modifications have no impact on the critical path in system mode and require only moderate extra cost compared to a standard scan design.

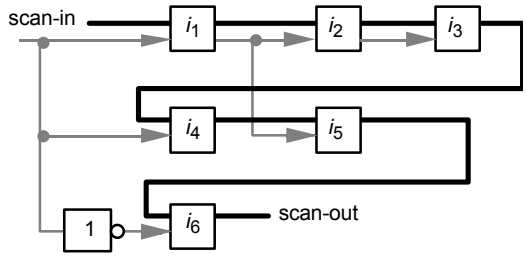


Figure 4: Scan design with regular and compressed shift mode.

Combining this technique with the architecture of Figure 2, the length of the Johnson counter and, accordingly, the length of the folding seeds can be shortened considerably. A further reduction of the storage amount is obtained by extending the purely deterministic scheme to a mixed mode scheme which uses pseudo-random patterns for the easy to detect faults. The resulting architecture, which is the target architecture for the remainder of this paper, is shown in Figure 5.

The generator can switch between an LFSR for pseudo-random pattern generation and the programmable Johnson counter producing the folding counter sequences. With some minor changes the length of the LFSR and the Johnson counter can even be different. The crucial step in synthesizing the architecture of Figure 5 for a particular circuit is to identify a minimal number of seeds for the folding counter, such that the resulting folding counter sequences contain a width-compressed deterministic test set for the hard faults.

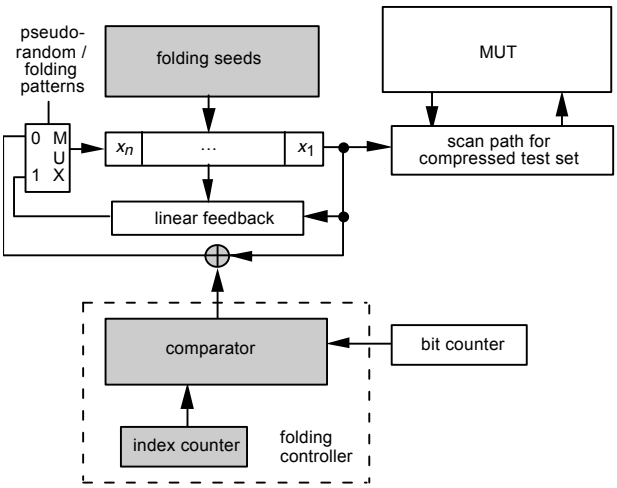


Figure 5: Architecture for mixed mode BIST based on reseeding of folding counters.

For the solution of this problem a more thorough understanding of the nature of folding counters sequences is required. Therefore the next section first provides the necessary theoretical background before the complete synthesis procedure and experimental results are presented in the subsequent sections.

3 Folding Counters – Theoretical Background

To generate deterministic test sets by a folding counter it is crucial to have a criterion to decide whether two or more patterns can be part of the same sequence. Furthermore, it must be possible to derive the appropriate seeds for the folding counter. Both problems can be solved by computationally simple procedures, which are motivated and explained below by a more detailed analysis of the simple example sequence of Figure 1 (see Figure 6).

	state	index
register contents # inversions	0110 0000	0
register contents # inversions	1001 1111	1
register contents # inversions	1110 1222	2
register contents # inversions	1101 1233	3
register contents # inversions	1100 1234	4

Figure 6: Analysis of the example sequence of Figure 1.

For each state of the folding counter the contents of the state register is shown as well as the number of inversions of each bit position with respect to the seed. It may be easily verified that the number of inversions is deter-

mined only by the bit position j and the index i of the state. For $1 \le j \le n$ and $0 \le i \le n$ it can be computed as

$$(*) \qquad inv(j,i) = \begin{cases} j & if \quad j < i \\ i & else \end{cases}.$$

This implies that the seed can easily be reconstructed once the index of a state is known. Although it is not possible to determine the index of a state only from the register contents, the bitwise EXOR of two state vectors reflects the index constellation of the two vectors. Consider for example the three state vectors $x_1 = (1,0,0,1)$, $x_2 = (1,1,1,0)$ and $y = (1,1,0,1)$. As illustrated in Figure 7 the bitwise EXOR-operations $x_1 \oplus y$ and $x_2 \oplus y$ provide vectors z_1 and z_2 consisting both of the following parts: a sequence of zeros in the first bit positions, a sequence of zeros or ones in the last bit positions, and a middle part in the case of z_1.

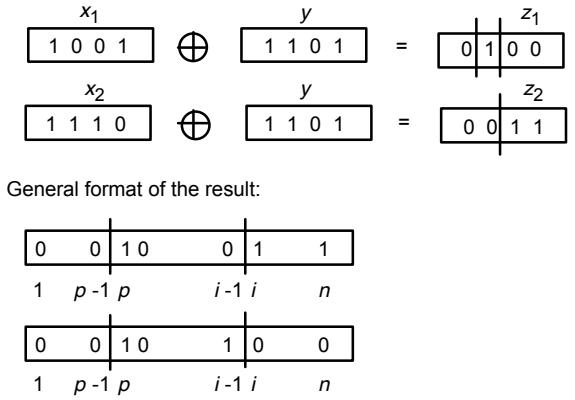


Figure 7: Comparing state vectors within a folding sequence.

In general, it can be observed that for a state with index i the last n-i+1 bits either coincide with the last n-i+1 bits of all states with lower index or are exactly the logic complement of them. Comparing the lower bit positions to states of lower index some of the first bits may coincide and the remaining bit positions have an alternating odd and even difference in the number of inversions. This leads to the general format of the result when a state of lower index is compared to a state of higher index. Then the bit position p-1 denoting the end of the all zero sequence exactly corresponds to the index of the first vector, whereas the position i identifying the tail exactly corresponds to index of the second state vector.

Definition 1 (*folding relation*): Two vectors $x, y \in \{0,1\}^n$ fulfill the folding relation $x \; F \; y$, if and only if the bitwise EXOR operation of both vectors results in a vector $z = x \oplus y$ of the form

$$(z_1, \ldots, z_{p-1}, z_p, \ldots, z_{i-1}, z_i, \ldots, z_n),$$

where $1 \le i \le n$, $1 \le p < i$ and

i) $(z_1, \ldots, z_{p-1}) = (0, \ldots, 0)$,

ii) $(z_i, \ldots, z_n) = (0, \ldots, 0)$ or $(z_i, \ldots, z_n) = (1, \ldots, 1)$, and

iii) $(z_p, \ldots, z_{i-1})$ is an alternating sequence of zeros and ones with $z_{p-1} \neq z_p$ and $z_{i-1} \neq z_i$.

The bit position i is called the ***folding index*** of x and y.

The following theorem shows that the folding index exactly corresponds to the maximal index of x and y in a folding counter sequence.

Theorem 1: Let $x, y \in \{0,1\}^n$ be two vectors with $x \; F \; y$ and folding index i. With $\neg^k y_j$ denoting the operation of inverting y_j k times the seed

$$s = (s_1, \ldots, s_n) = (\neg^{inv(1,i)} y_1, \ldots, \neg^{inv(n,i)} y_n),$$

provides a folding counter sequence such that the vectors x and y appear as states of index p-1 and i, respectively.

The proof of Theorem 1 is given in the appendix. As an immediate consequence Theorem 2 is obtained.

Theorem 2: Let $X \subset \{0,1\}^n$ be a set of vectors, and let $x^* \in X$ be a vector with $x \; F \; x^*$ for all $x \in X \setminus \{x^*\}$ and a single folding index i. Then there exists a seed $s \in \{0,1\}^n$, such that the folding counter sequence starting from s contains the complete set X.

Example: Consider the set $X = \{x_1 = (0,0,1,1), x_2 = (1,0,0,0), x_3 = (1,0,0,1)\}$. The vector $x^* = x_3 = (1,0,0,1)$ yields $x^* \oplus x_1 = (1,0,1,0)$ and $x^* \oplus x_2 = (0,0,0,1)$. Therefore x^* is in folding relation with x_1 and x_2, and the folding index is 4 in both cases. According to Theorem 1 the index of x_1 is 0, and the index of x_2 is 3. Therefore (0,0,1,1) is a suitable seed. The complete folding counter sequence starting from (0,0,1,1) is (0,0,1,1), (1,1,0,0), (1,0,1,1), (1,0,0,0), (1,0,0,1).

4 The Complete Synthesis Procedure

The core of the synthesis procedure is an algorithm to determine the minimal number of folding seeds, such that the resulting folding counter sequences cover a given test set T. As a consequence of Theorem 2 a „folding graph“ $G_F(T)$ can be associated with T as shown in Figure 8.

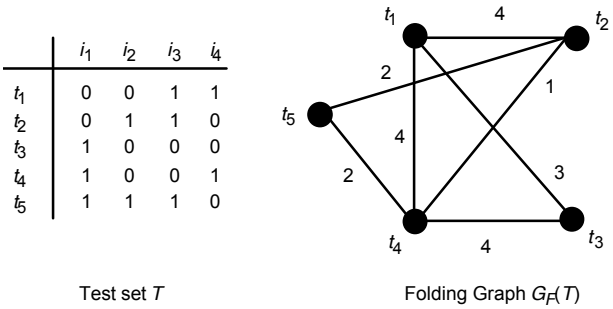


Figure 8: Deterministic test set and folding graph.

The vertices of the graph correspond to test vectors, and there is an edge between two vertices, if and only if the vectors are in folding relation. The edges are labeled with the respective folding indices. The set of vertices must be covered by a minimal number of subsets, such

that in each subset C there is a vertex v_C* which is connected to all other vertices in C by edges carrying a unique label. For the example of Figure 8 it can be easily verified that $C = \{t_1, t_3, t_4\}$ and $D = \{t_2, t_4, t_5\}$ form a minimal cover with the desired properties. To solve the covering problem in the general case a heuristic is used. As long as the set of vertices is not completely covered by suitable subsets, for each of the uncovered vertices a maximum set of vertices with the required properties is determined. The best set is selected as new subset in the cover. Finally for all subsets the corresponding seeds are constructed according to Theorem 1.

So far only completely specified test sets T have been considered for the sake of clarity. However, to fully exploit the available optimization potential, the synthesis procedure must be able to deal with test cubes efficiently. In this case a folding relation may hold only if certain don't cares are fixed appropriately. Also two cubes do not uniquely define a folding index. Instead, a set of possible folding indices can be determined for each pair of cubes.

Consider for example the cubes x = (1,1,-,-,0) and y = (-,1,-1,1). Bitwise EXOR yields a cube z = (-,0,-,-,1). To guarantee a folding relation between x and y the first bit of z must be zero, therefore the first bit of y must be fixed to one. The resulting folding index may be 3, 4 or 5 depending on the way the remaining don't cares are fixed.

The current version of the proposed synthesis procedure does not attempt to find an optimal fixing of unspecified bits. Instead, a simple iterative procedure is used to build the folding graph. The main algorithm described before then operates on this graph. Whenever a new subset is selected with a specific folding index, the corresponding bits are fixed to guarantee the consistency of the procedure. With these extensions the complete synthesis procedure for the proposed BIST architecture basically consists of the following three steps:

1) The feedback polynomial for the LFSR and the desired number N of random patterns are selected. The first N patterns of the LFSR sequence are fault simulated to determine the set F_{hard} of random pattern resistant faults [8, 11].
2) A set of deterministic test cubes $T \subset \{0,1,-\}^n$ is computed for F_{hard}, and input reduction is performed applying the algorithms proposed in [6]. As a result a width compressed test set $T' \subset \{0,1,-\}^k$ is obtained.
3) The algorithm described above is applied to solve the folding cover problem for T'.

Since the overall hardware overhead for the proposed BIST scheme may depend on both the choice of the LFSR and the number of random patterns, the complete procedure may be iterated with a different choice of these parameters.

5 Experimental Results

A series of experiments has been performed with the ISCAS-85 and the combinational parts of the ISCAS-89 circuits [2, 3]. Only circuits which still had undetected faults after 10000 random patterns were analyzed in further detail. To generate the deterministic test cubes for the hard faults a proprietary ATPG tool was used with the option to minimize the number of specified bits.

Table 1 shows the results for the advanced scheme of Figure 5. Columns one and two contain the names of the circuits and the number of pseudo-primary inputs. The next two columns list the lengths of the LFSRs used in the advanced architecture as well as the number of inputs remaining after input reduction. Columns five and six contain the number of folding seeds and the total number of bits to be stored ("ROM(A)"). To estimate the impact of input reduction the last column shows the ratio of the storage requirements for the advanced scheme and for the basic scheme of Figure 2 ("ROM(B)"). It can be observed that, particularly for the larger circuits with relatively long scan chains, input reduction results in an enormous reduction of the overall storage amount. For the largest two circuits, s38417 and s38584, the number of bits to be stored for the advanced scheme is only 5 % and 2 %, respectively.

Circuit	PPI	Advanced Scheme with Input Reduction				$\frac{\text{ROM(A)}}{\text{ROM(B)}}$
		LFSR	Circuit	Seeds	ROM(A)	
s420	34	13	12	11	132	0,65
s641	54	11	10	5	50	0,46
s713	54	11	9	4	36	0,33
s838	66	28	28	25	700	0,42
s953	45	13	6	2	12	0,13
s1196	32	13	5	2	10	0,01
s1238	32	13	8	3	24	0,03
s5378	214	13	11	12	132	0,06
s9234	247	26	33	70	2310	0,20
s13207	700	13	13	19	247	0,01
s15850	611	27	27	89	2403	0,13
s38417	1664	20	38	179	6802	0,05
s38584	1464	21	20	33	660	0,02
c2670	233	26	40	27	1080	0,21
c7552	207	18	42	64	2688	0,30

Table 1: Comparison of basic and advanced schemes.

These results also outperform the results achieved by competitive approaches relying on the reseeding of multiple-polynomial LFSRs and of "twisted ring" (Johnson)

counters [4, 10]. For our comparison we referred to published results for a mixed-mode BIST using 10000 pseudo-random patterns to eliminate the easy to detect faults. In all cases the proposed scheme had the lowest number of bits to be stored. The reduction of storage amount with respect to both previously published approaches is summarized in Table 2, which lists the ratios of the storage requirements for the respective schemes.

Circuit	$\frac{\text{ROM(Folding)}}{\text{ROM(LFSR)}}$	$\frac{\text{ROM(Folding)}}{\text{ROM(TRC)}}$
s420	0,53	0,47
s641	0,27	0,10
s713	0,20	0,08
s838	0,43	0,36
s953	0,09	0,04
s1196	0,04	0,03
s1238	0,10	0,08
s5378	0,18	0,62
s9234	0,33	-
s13207	0,07	0,01
s15850	0,37	-
s38417	0,28	-
s38584	0,19	-
c2670	0,32	0,07
c7552	0,51	0,12

Table 2: Storage amount for the proposed scheme as percentage of the storage requirements for [10] and [4].

The proposed scheme only requires between 7% and 53 % of the storage for reseeding of multiple-polynomial LFSRs and between 3 % and 67 % of the storage amount for the method based on twisted ring counters (TRC).

It should be noted that the method based on twisted ring counters published in [4] can also be combined with width compression techniques. Unfortunately, results for this extension were only available for a pure deterministic test without an initial pseudo-random pattern generation and only for few of the hard circuits. To allow a fair comparison we reran the experiments for the proposed scheme also without initial pseudo-random pattern generation (see Table 3). The results show that for the small circuits the method based on twisted ring counters is superior. This is largely due to the fact that the results presented in [CHAK99] are obtained using a more sophisticated technique of width compression (as can be seen by comparing the widths listed in columns two and five). However, for the larger circuits the proposed scheme outperforms the twisted ring approach and also provides good results for the circuits which could not be processed by the latter.

Circuit	TRC with width compression			Folding		
	Width	Seeds	ROM	Width	Seeds	ROM
s420	16	8	128	21	28	588
s641	8	5	40	18	16	288
s713	9	5	45	18	15	270
s838	9	5	45	17	29	493
s953	14	18	252	19	57	1083
s1196	15	44	660	19	62	1178
s1238	15	43	645	37	50	1850
s5378	-	-	-	30	106	3180
s9234	-	-	-	43	194	8342
s13207	27	177	4779	39	89	3471
s15850	32	314	10048	46	163	7498
s38417	-	-	-	71	500	35500
s38584	-	-	-	41	178	7298
c2670	33	193	6369	55	90	4950
c7552	64	649	41536	73	434	31682

Table 3: Comparing the proposed technique and the technique based on twisted ring counters (TRC) combined with test width compaction (results without pseudo-random pattern generation).

6 Conclusions

A new and efficient scheme for scan-based BIST has been presented. The core of this scheme is a new type of generator called folding counter, which is enhanced with features for pseudo-random pattern generation. With the scan design being adapted to process width compressed test patterns the proposed target architecture can produce pseudo-random patterns for the easy to detect faults and characterize width compressed deterministic test sets for the hard faults as a small number of seeds for the folding counter. An experimental analysis shows that the storage amount for the seeds is considerably lower than for competitive approaches. Furthermore, the simple and regular structure of the folding counter allows an efficient hardware implementation, such that the overall scheme provides a flexible low cost solution for high quality BIST.

7 References

1 M. Abramovici, M. Breuer, A. Friedman: Digital Systems Testing and Testable Design; New York: Computer Science Press (W. H. Freeman and Co.), 1990

2 F. Brglez et al.: Accelerated ATPG and fault grading via testability analysis; Proc. IEEE Int. Symp. on Circuits and Systems, Kyoto, 1985

3 F. Brglez, D. Bryan and K. Kozminski: Combinational Profiles of Sequential Benchmark Circuits; Proc. IEEE Int. Symp. on Circuits and Systems, 1989, pp. 1929-1934

4 K. Chakrabarty, B. T. Murray, and V. Iyengar: Built-In Test Pattern Generation for High-Performance Circuits Using Twisted-Ring Counters; Proc. 17th IEEE VLSI Test Symp., Dana Point, CA, 1999, pp. 22-27

5 C.-A. Chen, S. K. Gupta: A Methodology to Design Efficient BIST Test Pattern Generators; Proc. IEEE Int. Test Conf., Washington, DC, 1995, pp. 814-823

6 C.-A. Chen, S. K. Gupta: Efficient BIST TPG Design and Test Set Compaction via Input Reduction; IEEE Trans. on CAD, Vol. 17, No. 8, August 1998, pp. 692-705

7 C. Dufaza, G. Cambon: LFSR based Deterministic and Pseudo-Random Test Pattern Generator Structures; Proc. Eur. Test Conf., Munich, 1991, pp. 27-34

8 C. Fagot, O. Gascuel, P. Girard, C. Landrault: On calculating efficient LFSR seeds for built-in self test; Proc. IEEE Eur. Test Workshop 1999 (ETW'99), Constance, Germany, 1999, pp. 7-14

9 S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois: Built-in Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers; IEEE Trans. on Comp., Vol. 44, No.2, February 1995, pp. 223-233

10 S. Hellebrand, B. Reeb, S. Tarnick, H.-J. Wunderlich: Pattern Generation for a Deterministic BIST Scheme; Proc. IEEE/ACM Int. Conf. on CAD-95, San Jose, CA, November 1995, pp. 88-94

11 S. Hellebrand, H.-J. Wunderlich, A. Hertwig: Mixed-Mode BIST Using Embedded Processors; Journal of Electronic Testing Theory and Applications (JETTA), Vol. 12, Nos. 1/2, February/April 1998, pp. 127-138

12 D. Kagaris, S. Tragoudas, A. Majumdar: On the Use of Counters for Reproducing Deterministic Test Sets; IEEE Trans. on Comp., Vol. 45, No. 12, Dec. 1996, pp.1405-1419

13 G. Kiefer, H.-J. Wunderlich: Using BIST Control for Pattern Generation; Proc. IEEE Int. Test Conf., Washington, DC, November 1997, pp. 347-355

14 B. Koenemann: LFSR-Coded Test Patterns for Scan Designs; Proc. Eur. Test Conf., Munich 1991, pp. 237-242

15 E. J. McCluskey: Verification Testing - A Pseudoexhaustive Test Technique; IEEE Trans. on Comp., Vol. C-33, No.6, June 1984, pp. 541 - 546

16 N. A. Touba and E. J. McCluskey: Altering a Pseudo-Random Bit Sequence for Scan-Based BIST; Proc. IEEE Int. Test Conf., Washington, DC, 1996, pp. 167-175

17 H.-J. Wunderlich, G. Kiefer: Bit-Flipping BIST; Proc. ACM/IEEE Int. Conf. on CAD-96 (ICCAD96), San Jose, CA, November 1996, pp. 337-343

Appendix: Proof of Theorem 1

Since $x\ F\ y$ with folding index i, the vector $z = x \oplus y$ is of the form $(z_1, \ldots, z_{p-1}, z_p, \ldots, z_{i-1}, z_i, \ldots, z_n)$ with the properties listed in Definition 1. Consequently $(z_1, \ldots, z_{p-1}) = (0, \ldots, 0)$ and $z_{p-1} \neq z_p$ hold, and therefore $z_p = 1$ is true. If the difference $i - p$ is odd, then the tail is given by $(z_i, \ldots, z_n) = (0, \ldots, 0)$, else $(z_i, \ldots, z_n) = (1, \ldots, 1)$ holds.

Obviously

$$s = (s_1, \ldots, s_n) = (\neg^{inv(1,i)} y_1, \ldots, \neg^{inv(n,i)} y_n),$$

implies

$$y = (\neg^{inv(1,i)} s_1, \ldots, \neg^{inv(n,i)} s_n)$$

and y appears as state of index i in a folding counter sequence starting from s.

To show that that p-1 indicates the index of x in the sequence define

$$x^* = (\neg^{inv(1,p-1)} s_1, \ldots, \neg^{inv(n,p-1)} s_n).$$

Then x^* appears as state of index p-1 in the folding counter sequence and the bitwise EXOR operation $z^* = x^* \oplus y$ yields a vector of the form $(z_1^*, \ldots, z_{p-1}^*, z_p^*, \ldots, z_{i-1}^*, z_i^*, \ldots, z_n^*)$. Since $inv(j, p\text{-}1) = inv(j, i)$ for $j \leq p\text{-}1 < i$, the EXOR operation

$$z_j^* = x_j^* \oplus y_j = \neg^{inv(j,p-1)} s_j \oplus \neg^{inv(j,i)} s_j$$

provides 0 for $j \leq p\text{-}1 < i$. For $p \leq j < i$ the value of $inv(j, p\text{-}1)$ is constant while $inv(j, i)$ increases with j. Therefore $x_j^* \oplus y_j$ produces an alternating sequence of zeros and ones. For j ≥ i both $inv(j, p\text{-}1)$ and $inv(j, i)$ are constant, and we have $(z_i^*, \ldots, z_n^*) = (0, \ldots, 0)$, if the difference $i - p$ is odd, else $(z_i^*, \ldots, z_n^*) = (1, \ldots, 1)$. Thus $z = z^*$ and $x = x^*$. q.e.d.