Proceedings IEEE International Test Conference, Baltimore, MD, October 30 – November 1, 2001

TWO-DIMENSIONAL TEST DATA COMPRESSION FOR SCAN-BASED DETERMINISTIC BIST

Hua-Guo Liang[†], Sybille Hellebrand[‡], Hans-Joachim Wunderlich[†]
[†] University of Stuttgart, Germany
[‡] University of Innsbruck, Austria

Abstract

In this paper a novel architecture for scan-based mixed mode BIST is presented. To reduce the storage requirements for the deterministic patterns it relies on a two-dimensional compression scheme, which combines the advantages of known vertical and horizontal compression techniques. To reduce both the number of patterns to be stored and the number of bits to be stored for each pattern, deterministic test cubes are encoded as seeds of an LFSR (horizontal compression), and the seeds are again compressed into seeds of a folding counter sequence (vertical compression). The proposed BIST architecture is fully compatible with standard scan design, simple and flexible, so that sharing between several logic cores is possible. Experimental results show that the proposed scheme requires less test data storage than previously published approaches providing the same flexibility and scan compatibility.

1 Introduction

State-of-the-art systems-on-a-chip (SoCs) typically consist of some user defined logic and various embedded cores, such as memory and processor cores as well as mixed-signal, analogue or RF cores. This core-based design style greatly increases the design productivity and speeds up the time to market, but on the other hand, it also turns testing into a more and more challenging task. The increased system complexities and growing test data volumes, the inaccessibility of internal blocks, the need for testers specifically tuned to certain types of cores, the increasing impact of timing faults, and the growing need for periodic maintenance and on-line testing capabilities discard traditional external testing with automatic test equipment (ATE) as a feasible solution. Built-in self-test (BIST) not only circumvents these problems, but, in many cases, also offers the cheaper and more efficient alternative. Therefore BIST has become a topic of major interest in recent years, and pseudo-random BIST using an LFSR for test pattern generation has meanwhile been widely accepted as the standard BIST approach [1].

For circuits containing pseudo-random pattern resistant faults basically two different strategies have been proposed: Techniques for test point insertion modify the mission logic to improve the random pattern testability and to guarantee a high fault coverage with a feasible number of test patterns [8, 20, 22]. To minimize performance degradations, the second strategy avoids any changes in the mission logic and relies on more sophisticated test pattern generators instead. This category comprises weighted random pattern and pseudo-exhaustive testing as well as mixed mode approaches [4, 5, 9, 12-14, 16-18, 21, 23]. The latter typically use an LFSR to generate a limited number of pseudo-random patterns detecting most of the faults. For the remaining hard to detect faults deterministic patterns are provided, which are either directly embedded into the LFSR sequence by some extra control logic ("bit-fixing", "bit-flipping") or stored in compressed form and regenerated during BIST ("store-and-generate"). The bit-flipping approach has been proven to be very successful even for large industrial circuits [16, 17, 23]. However, the BIST architecture is specifically tailored to a given deterministic test set, and a small change in the test requirements and the test set may make it necessary to resynthesize the BIST hardware. In contrast to that, store-and-generate schemes can provide a more flexible solution. Here, a powerful technique for compressing the deterministic data is the key to an efficient BIST implementation. As sketched in Figure 1, there are two natural ways of reducing the amount of test data to be stored: Vertical compression schemes reduce the number of patterns to be stored, whereas horizontal compression schemes reduce the number of bits to be stored for each test pattern.

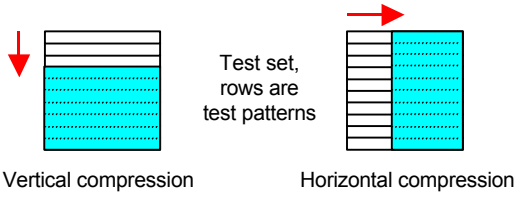

Figure 1: Vertical and horizontal test data compression.

* Part of this work has been supported by the DFG grant WU 245/1-4 "Test and Synthesis of Fast Embedded Systems". Hua-Guo Liang is also associated with Hefei University, China.

A classical method for horizontal compression (also referred to as "test width compression") is to store identical columns of the test set only once [6, 7, 19]. In general this approach allows a high compression rate, but in a scan-based BIST environment it may require a reorganization of the scan chain and disturb the design flow. On the other hand, techniques based on the reseeding of LFSRs efficiently compress the width of the information to be stored while leaving the scan chain untouched [12, 18]. As sketched in Figure 2 deterministic test cubes are encoded as seeds of an LFSR, and during BIST the LFSR expands the seeds to the desired scan patterns. The length of the LFSR is determined only by the maximum number of specified bits s_{max} in the given set of test cubes. If only one polynomial is used, an LFSR of length s_{max}+20 guarantees a high probability of successful encoding; for the reseeding of multiple-polynomial LFSRs feedback polynomials of degree s_{max} are sufficient [12, 18].

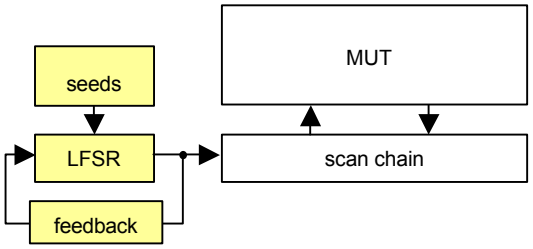

Figure 2: Test width compression based on the reseeding of LFSRs.

To achieve a vertical compression the test data are usually encoded as seeds of a standard pattern generators or of an arbitrary finite state machine [4, 5, 10, 15]. Recently, a vertical compression scheme based on the reseeding of folding counters has been proposed [11]. As depicted in Figure 3, a folding counter can be realized as a Johnson counter with programmable feedback, and for each seed it produces a sequence of patterns to be shifted into the scan chain.

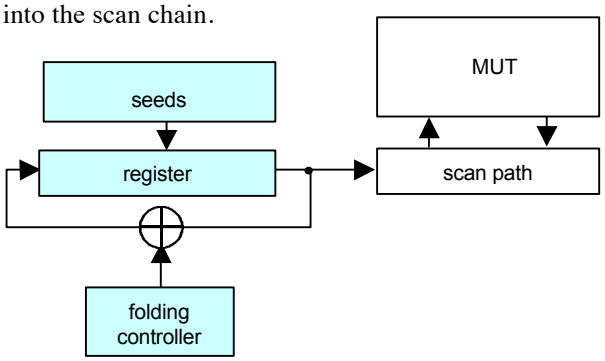

Figure 3: Vertical test data compression based on the reseeding of folding counters.

It has been shown in [11] that folding counter sequences well reflect the typical properties of deterministic patterns, and that in general a reasonably small number of seeds is sufficient to characterize the complete test data. In this paper a two-dimensional compression technique will be presented which efficiently combines the advantages of the reseeding of folding counters for vertical compression and the reseeding of LFSRs for horizontal compression (see Figure 4).

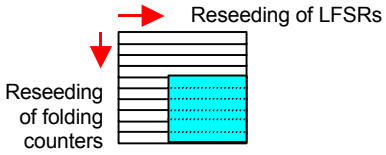

Figure 4: Two-dimensional test data compression.

The target BIST architecture shown in Figure 5 integrates an LFSR and a folding controller to achieve the proposed two-dimensional data compression. It is fully compatible with a standard design flow, since it does not require any modifications of the scan chain. Moreover, as the LFSR can also be used for pseudo-random pattern generation, it supports a mixed mode BIST without any extra cost.

However, in their basic forms the reseeding of LFSRs and the reseeding of folding counters are not compatible, such that a simple merging of the original BIST architectures is not possible. To provide the necessary background for the detailed description of the proposed scheme in Section 3, the next section briefly reviews the basic concepts of folding counter sequences and derives some new important properties. Section 4 describes the complete synthesis procedure for the proposed architecture, and finally Section 5 summarizes the experimental results obtained so far. They clearly show that the presented approach requires less storage requirements than other BIST schemes providing the same flexibility and scan compatibility.

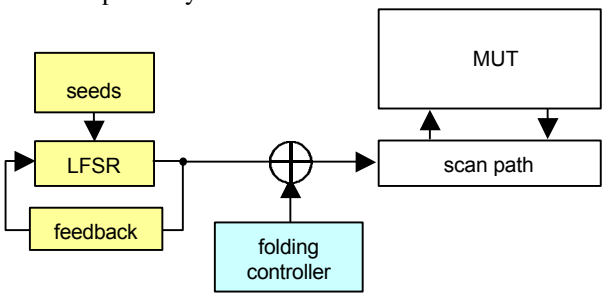

Figure 5: Target BIST architecture for two-dimensional test data compression.

2 Folding Counters Revisited

This section briefly summarizes some of the basic concepts and properties of folding counters as introduced in [11] and develops the necessary background for the two-

dimensional reseeding approach proposed in this paper. Unlike most other BIST pattern generators folding counters work with a dynamically changing state transition function depending on both the state of the counter and the "index" of the transition. Starting from an initial state $s \in \{0, 1\}^n$ a sequence of $n+1$ states s = $F(0, s)$, $F(1, s)$, …, $F(n, s)$ is produced, such that the transition from $F(i, s)$ to $F(i+1, s)$ retains the first i bits and inverts the remaining ones. A simple example is shown in Figure 6.

state			index
F(0, *s*)	=	0 1 1 0	0
F(1, *s*)	=	1 0 0 1	1
F(2, *s*)	=	1 1 1 0	2
F(3, *s*)	=	1 1 0 1	3
F(4, *s*)	=	1 1 0 0	4

Figure 6: Sequence produced by a 4-bit folding counter.

Folding counters provide a good means for vertical test set compression, since for deterministic BIST a folding counter may be run with a number of different seeds, such that the resulting folding counter sequences contain a given deterministic test set. To store the deterministic test set, it is then sufficient to store the folding seeds which can be used to regenerate the complete test set during BIST. To find an appropriate selection of seeds in [11] "folding relations" between pairs of vectors have been exploited. In this paper a completely different approach is followed which is based on the observations stated below.

Observation 1: For a state of index i the value at bit position j can be derived from the initial state s by performing a number of inversions which is determined only by the bit position j and the index i of the state. For $0 \le j < n$ and $0 \le i \le n$ it is given by

$$(*) \qquad inv(j,i) = \begin{cases} j+1 & if \quad j < i \\ i & else \end{cases}.$$

Figure 7 illustrates this important property for the example sequence of Figure 6.

	j = 0 1 2 3	i
state	0 1 1 0	0
inv(j, i)	0 0 0 0	
state	1 0 0 1	1
inv(j, i)	1 1 1 1	
state	1 1 1 0	2
inv(j, i)	1 2 2 2	
state	1 1 0 1	3
inv(j, i)	1 2 3 3	
state	1 1 0 0	4
inv(j, i)	1 2 3 4	

Figure 7: State transitions and inversions.

As a consequence, for any given state in a folding counter sequence the seed can easily be reconstructed, once the index of the state is known. Conversely, for an arbitrary vector $x \in \{0,1\}^n$ and a given index i a seed vector can be constructed, such that the resulting folding counter sequence contains x as a state of index i.

Observation 2: Let $x \in \{0,1\}^n$ be an arbitrary vector, and let i be a desired index, $0 \le i \le n$. The seed

$$s = (s_0, \ldots, s_{n-1}) = (\neg^{inv(0,i)} x_0, \ldots, \neg^{inv(n-1,i)} x_{n-1}),$$

provides a folding counter sequence such that x appears as state of index i ($\neg^k x_j$ denotes the operation of inverting x_j k times).

Since a given vector $x \in \{0,1\}^n$ can be part of a folding counter sequence with an index varying from 0 to n, overall there are $n+1$ possible seed vectors which lead to folding counter sequences containing x. Figure 8 shows an example for $x = (0,1,1,0,1)$. It can easily be verified that the seed vectors $s^i = \left(s_0^i, \ldots, s_{n-1}^i\right)$ for $x = (x_0, \ldots, x_{n-1})$ as state of index i and $s^{i+2} = \left(s_0^{i+2}, \ldots, s_{n-1}^{i+2}\right)$ for x as state of $i+2$ differ only in bit position i for $0 \le i \le n-2$. Furthermore, s^{n-1} and s^n differ only in bit position $n-1$. In general, due to the inherent regularity of folding counter sequences, the following theorem is true.

Theorem: Let $x = (x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ be an arbitrary vector, and let $s^i = \left(s_0^i, \ldots, s_{n-1}^i\right)$ denote the seed, such that x appears as state of index i in an n-bit folding counter sequence, $0 \le i \le n$. Then for $0 \le i \le n-2$ the seeds $s^i = \left(s_0^i, \ldots, s_{n-1}^i\right)$ and $s^{i+2} = \left(s_0^{i+2}, \ldots, s_{n-1}^{i+2}\right)$ differ only in bit position i, and the seeds s^{n-1} and s^n differ only in bit position $n-1$.

Proof: According to Observation 2 the seed s^i is obtained as $s^i = (\neg^{inv(0,i)} x_0, \ldots, \neg^{inv(n-1,i)} x_{n-1})$, and in the same way $s^{i+2} = (\neg^{inv(0,i+2)} x_0, \ldots, \neg^{inv(n-1,i+2)} x_{n-1})$. If $0 \le i \le n-2$, four cases must be distinguished for the inversion numbers $inv(j, i)$ and $inv(j, i+2)$:

(i) If $j < i$ holds, then also $j < i+2$ and $inv(j, i) = inv(j, i+2)$.

(ii) If $j = i$ holds, then $inv(j, i) = i$, but still $j < i+2$ and $inv(j, i+2) = j+1 = i+1$.

(iii) If $j = i+1$ holds, then $inv(j, i) = i$, but still $j < i+2$ and $inv(j, i+2) = j+1 = i+2$. However, $\neg^i x_j = \neg^{i+2} x_j$.

(iv) If $j \ge i+2$ then $inv(j, i) = inv(j, i+2) = i$.

Only in case (ii) for $j = i$ different results are obtained for $\neg^{inv(j,i)} x_j$ and $\neg^{inv(j,i)} x_j$, and thus s^i and s^{i+2} differ only in bit position i. In the same way it can be shown that s^{n-1} and s^n only in bit position $n-1$, which concludes the proof. q.e.d.

As a consequence of the Theorem, the seeds s^i and s^{i+2} can be merged to a seed cube with a don't care in bit position i. For the example of Figure 8 the set of possible seeds for $x = (0,1,1,0,1)$ can be represented by $\{(-,1,1,0,1), (1,-,0,1,0), (1,1,0,0,-)\}$.

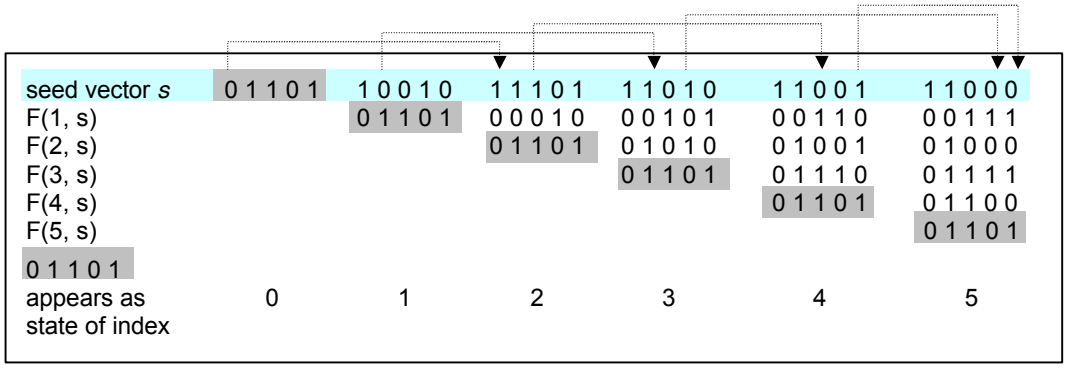

Figure 8: Possible seeds for $x = (0,1,1,0,1)$.

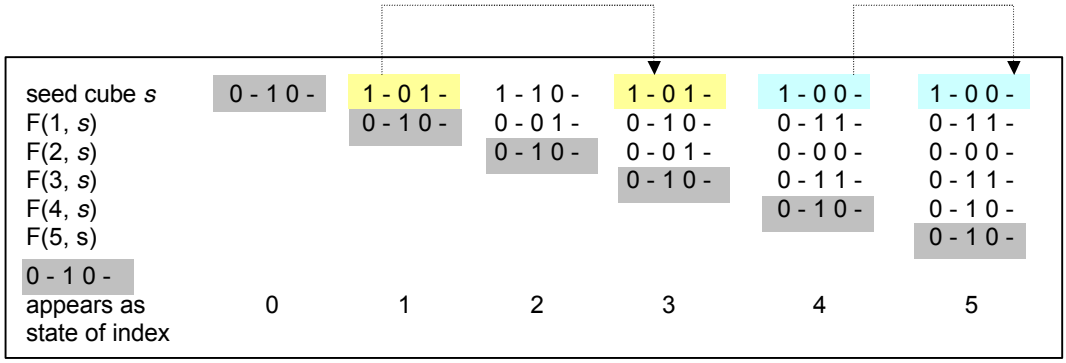

Figure 9: Seed construction for test cubes.

When dealing with test cubes $t \in \{0,1,-\}^n$, a similar procedure for extracting possible seed cubes is applied as described in the following. For a test cube $t = (t_0, \ldots, t_{n-1}) \in \{0,1,-\}^n$ the seed cubes $s^i = (s_0, \ldots, s_{n-1}) = (\neg^{inv(0,\,i)}\, t_0, \ldots, \neg^{\,inv(n-1,\,i)}\, t_{n-1})$ are constructed with the additional rule $\neg t_j = t_j$ for $t_j =$ '-' (see Figure 9).

As highlighted in Figure 9, even without further merging some of the seed cubes are identical. In general, if the construction rule defined above is applied to a test cube $t = (t_0, \ldots, t_{n-1}) \in \{0,1,-\}^n$, then according to the Theorem for each don't care $t_i =$ '-' the seed cubes s^i and s^{i+2} are identical. Consequently, if the number of specified bits is equal to k, at most k+1 different seed cubes are obtained. In the example of Figure 9 the 5-bit cube $t = (0,-,1,0,-)$ has 3 specified bits and 2 don't cares, and therefore there are 6-2 = 4 possible seed cubes. Of course, in the same way as in the fully specified case, further merging is possible, and it can be easily verified that the two cubes (-,-,1,0,-) and (1,-,0,-,-) represent all possible folding seeds for $t = (0,-,1,0,-)$.

Observation 3: Let $t \in \{0,1,-\}^n$ be a test cube with k specified bits. Then the number of possible seed cubes is at most k +1, and each seed cube contains at most k specified bits.

Thus for a set of test cubes $T \subset \{0,1,-\}^n$ with a maximum number of specified bits s_{max} for each cube at most s_{max}+1 possible seed cubes can be extracted, and a covering heuristic may be used to derive the best selection of seed cubes representing T. The actual BIST architecture depends on the realization of the folding counter. In [11] a Johnson counter with programmable feedback as shown in Figure 10 was proposed.

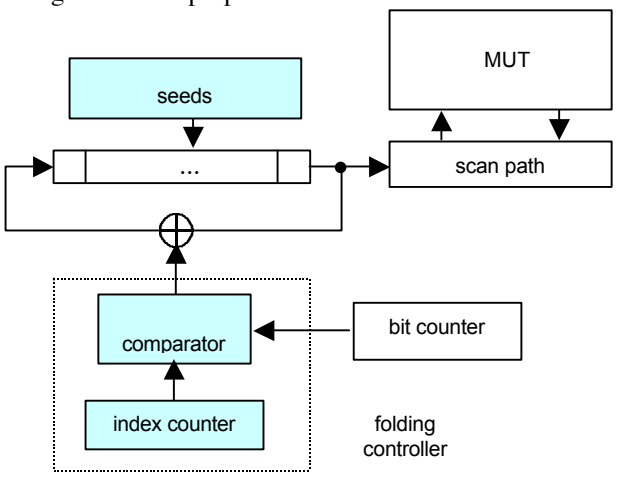

Figure 10: Basic deterministic BIST scheme using a folding counter.

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. 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.

As mentioned above, embedding deterministic test cubes in a folding counter sequence results in a vertical test set compression, i.e. in a reduction of the number of vectors to be stored. However, the basic scheme of Figure 10 does not allow any horizontal test set compression, and the Johnson counter must be of the same length as the scan path. To circumvent this problem in [11] classical techniques for test width compression were combined with the basic architecture of Figure 3 [6, 7, 19]. The experimental results in [11] show that this way high compression rates for the test data to be stored can be achieved, but a major drawback of the applied techniques for test width compression is that they require a reorganization of the scan chain, and that they are not compatible with common design flows. The next section will demonstrate that the combination of appropriate techniques for both horizontal and vertical test set compression allows both a fully scan compatible implementation of mixed mode BIST and high compression rates.

3 A BIST Architecture for Two-Dimensional Reseeding

As already pointed out classical techniques for test width compression require a reorganization of the scan chain and may not be acceptable within a typical design flow. On the other hand, reseeding of LFSRs offers a technique for horizontal test data compression which does not require any modifications of the scan chain. The purpose of this section is to show that a horizontal compression based on the reseeding of LFSRs can efficiently be combined with a vertical compression based on the reseeding of folding counters. The first problem to be solved is that the respective hardware schemes of Figures 2 and 10 cannot simply be merged, because the feedback functions of the LFSR and the folding counter cannot be activated simultaneously without interfering with each other. This problem is eliminated by a more flexible implementation of the folding compression scheme as shown below.

The folding controller in Figure 10 is replaced by a controller which is able to produce a state of any given index in the folding counter sequence from a given seed. Applying the same seed several times while increasing the index of the produced states then provides a complete folding counter sequence. This way the Johnson counter of Figure 10 is no longer necessary for implementing the state transitions of the folding counter, and the seed can be provided serially by any source (see Figure 11). In particular, the folding seed can be the output of an LFSR decompressing LFSR seeds into test cubes.

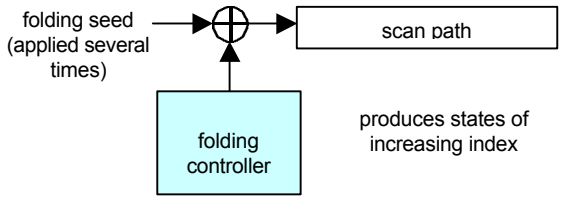

Figure 11: Alternative approach to generate scan patterns as folding counter sequences.

Before the resulting BIST architecture is discussed, the new folding controller is explained in some more detail. According to Observation 1 each bit position j in a state of index i is determined by the number of inversions $inv(i, j)$ to be performed starting from the seed. As shown in Figure 12 the value of $inv(j, i)$ is either i or j+1 depending on the bit position relative to the index.

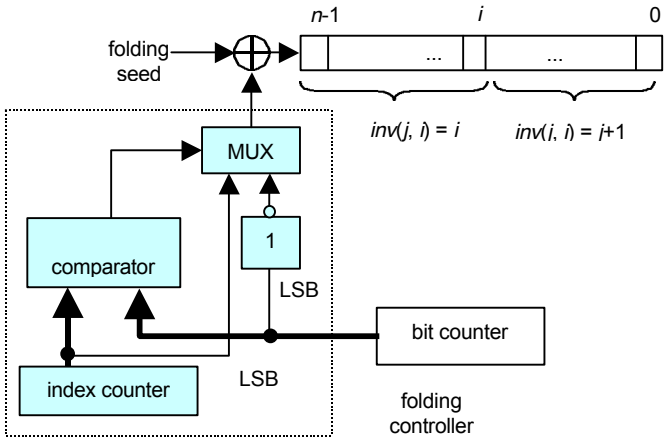

Figure 12: Structure of the proposed folding controller.

Since an inversion has to be performed, if and only if the value of $inv(i, j)$ is odd, for $0 \le j < i$ the output of the folding controller can be provided by the inverted least significant bit of the bit counter, and for $j \ge i$ the least significant bit of the index counter may be used. The comparator controls the multiplexer accordingly.

Compared to the folding controller used in the original scheme of Figure 10 there is one extra multiplexer and one extra inverter, but this extra cost is overcompensated by the elimination of the Johnson counter and the scan compatibility of the complete BIST architecture shown in Figure 13. During BIST an LFSR seed is loaded into the LFSR and decompressed into a folding seed by the LFSR

operation. While the folding seed is shifted into the scan path the folding controller transforms it into a state of index i in the folding counter sequence depending on the state of the index counter. When the scan path is completely loaded, the pattern is applied to the module under test, the bit counter is reset, and the index counter is activated. If the index counter has not yet cycled through all possible states, the same LFSR seed is reloaded, otherwise the index counter is reset and the next LFSR seed is processed.

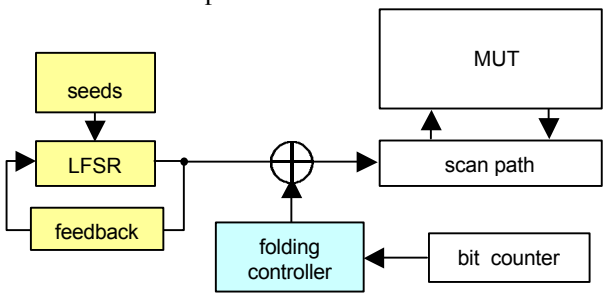

Figure 13: BIST scheme for two-dimensional reseeding.

4 Synthesis Considerations and Encoding Procedures

The scheme of Figure 13 is particularly suitable for a mixed mode BIST, since the LFSR can be used for pseudo-random pattern generation without any extra cost. Folding seeds and the corresponding LFSR seeds therefore only have to be determined for the remaining random pattern resistant faults F_{hard} after a given number N of pseudo-random patterns. Overall, the synthesis of the BIST hardware consists of the following four steps, which may be iterated several times.

1. The feedback polynomial of the LFSR and the desired number N of pseudo-random patterns must be selected. Using one polynomial for both pseudo-random pattern generation and for encoding may lead to sub-optimal solutions for both tasks. In these cases the scheme can be applied with two polynomials selected independently.
2. N pseudo-random patterns are fault simulated to determine F_{hard}.
3. ATPG for F_{hard} provides a deterministic test set $T \subset \{0,1,-\}^n$ for the pseudo-random pattern resistant faults.
4. An optimal set of LFSR seeds must be determined, such that during BIST the scheme of Figure 13 generates a sequence of scan patterns detecting all faults in F_{hard} .

In the fourth step the objective is to achieve minimum storage requirements, i. e. both the number and the width of the seeds should be minimized. The straightforward approach of determining an optimal set of folding seeds and encoding the folding seeds into LFSR seeds is not promising, because in an optimal set of folding seeds the maximum number of specified bits will be much higher than in the original test set. An increased number of specified bits results in a need for larger LFSRs for the encoding step and implies a decreased overall compression rate. Instead, a more careful combination of the synthesis techniques for folding schemes and the encoding techniques for the reseeding of LFSRs is required.

The proposed encoding scheme proceeds in three phases:

- For each test cube in T all possible seed cubes are constructed as explained in Section 2.
- The seed cubes are encoded into LFSR seeds by solving the corresponding systems of linear equations as described in [12, 18].
- A covering algorithm selects a minimal number of LFSR seeds, such that during BIST the patterns applied to the MUT detect all hard faults.

This strategy provides excellent possibilities to optimize the LFSR encoding. If the test set T contains a maximum number of specified bits s_{max}, then according to Observation 3 the maximum number of specified bits for all possible seeds cubes is at most s_{max}. A successful encoding is possible with a probability of $1\text{-}10^{-6}$ using a single-polynomial LFSR of degree s_{max}+20 or a multiple-polynomial LFSR of degree s_{max} [12, 18]. In general, the degree of the polynomial can be further decreased because of two reasons: Firstly, as shown in Section 2, merging of seed cubes reduces the number of specified bits, and, secondly, for a successful BIST it is sufficient to encode one of the folding seeds. In our experimental work in most cases a polynomial of degree s_{max}-5 turned out to be sufficient.

The covering problem to be solved after the encoding phase is illustrated in Figure 14.

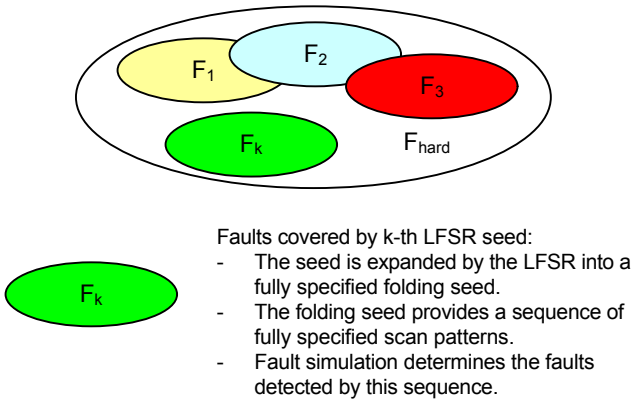

Figure 14: Seed Covering Problem.

Each LFSR seed constructed in the first phase corresponds to a subset of F_{hard}, and a minimum number of seeds has to be determined, such that F_{hard} is completely covered.

To implement the covering algorithm a simple greedy algorithm has been developed. For any new LFSR seed added to a partial solution fault simulation is performed for the expanded folding seed and the complete folding counter sequence produced by this seed. The algorithm stops, when all faults are detected. Since a folding seed expanded from an LFSR seed is a fully specified pattern and the resulting folding counter sequence also consists of fully specified patterns only, the fault simulation process is much more efficient than the fault simulation performed during the ATPG phase (during ATPG test cubes with a minimum number of specified bits are fault simulated). This way, the number of seeds can be kept reasonably small. The covering heuristic, of course has a higher optimization potential, if more than one seed cube can be encoded per test cube, and there is still a trade-off between minimizing the degree of the feedback polynomial and minimizing the number of required LFSR seeds.

5 Experimental Results

To evaluate the proposed BIST scheme, 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 pseudo-random patterns have been analyzed in further detail. Deterministic test cubes for the hard faults have been generated to guarantee the detection of all irredundant faults, i. e. to provide 100% fault efficiency. A proprietary ATPG tool has been used with the option to minimize the number of specified bits. Since preliminary experiments showed, that using only one polynomial for both pseudo-random pattern generation and LFSR encoding in all cases provided worse results than independently chosen feedback polynomials, the synthesis procedure described above has been applied with independently chosen feedback polynomials in all further experiments. The degrees of the polynomials have been varied from s_{max}-5 to s_{max}+2, and the best results have been selected.

Table 1 shows the results. The names of the circuits and the number of pseudo-primary inputs are given in columns one and two. The subsequent columns show the degree of the LFSR for pseudo-random pattern generation, the degree of the LFSR for encoding, the number of seeds, and the overall number of bits to be stored (ROM). The main advantage of the presented scheme is that both vertical and horizontal compression leave the scan chain untouched. The next two experiments therefore analyze the trade-off between the efficiency of test data compression and scan compatibility.

Circuit	# PPI	LFSR PRPG	LFSR Encoding	# Seeds	ROM
s420	34	15	16	10	160
s641	54	13	16	4	64
s713	54	13	16	4	64
s838	66	25	33	26	858
s953	45	15	10	2	20
s1196	32	17	10	3	30
s1238	32	13	14	3	42
s5378	214	14	14	14	196
s9234	247	41	40	95	3800
s13207	700	21	18	58	1044
s15850	611	22	30	112	3360
s38417	1664	49	42	267	11214
s38584	1464	46	49	59	2891
c2670	233	47	37	28	1036
c7552	207	127	133	36	4788

Table 1: Storage requirements for the two-dimensional reseeding scheme of Figure 13.

Table 2 compares the new scheme with the folding counter scheme in [11], which achieves high data compression rates but requires a reorganization of the scan chain.

Circuit	# PPI	ROM 2D	ROM [11]	$\frac{\text{ROM 2D}}{\text{ROM [11]}}$
s420	34	160	132	1,21
s641	54	64	50	1,28
s713	54	64	36	1,78
s838	66	858	700	1,23
s953	45	20	12	1,67
s1196	32	30	10	3
s1238	32	42	24	1,75
s5378	214	196	132	1,48
s9234	247	3800	2310	1,65
s13207	700	1044	247	4,23
s15850	611	3360	2403	1,4
s38417	1664	11214	6802	1,65
s38584	1464	2891	660	4,38
c2670	233	1036	1080	0,96
c7552	207	4788	2688	1,78

Table 2: Two-dimensional reseeding versus the folding counter scheme combined with scan path reorganization in [11].

Here it can be observed, that the folding counter scheme in [11] requires less test data storage than the proposed two-dimensional reseeding. However, the lower storage requirements are paid by a non standard scan organization.

Next we compare the proposed approach to the advanced twisted ring counter approach recently published in [5], which also is not scan compatible and requires to configure the circuit inputs into a twisted ring counter. Since the experiments reported in [5] do not rely on pseudo-random patterns to eliminate the easy to detect faults, the synthesis algorithm has been modified to target all faults (in general this requires more seeds than targeting F_{hard} only). The results in Table 3 show that in most cases the twisted ring counter approach is characterized by a smaller number of seeds, but the proposed approach is superior with respect to the number of bits to be stored. In the best case, the proposed approach requires only 26% of the test data volume for the twisted ring counter scheme, on the average approximately 77% are sufficient. Thus, the two-dimensional reseeding scheme is superior both with respect to the efficiency of test data compression and the scan compatibility.

Circuit	# PPI	Proposed Approach		TRC [5]		ROM 2D / ROM [5]
		# Seeds	ROM 2D	# Seeds	ROM TRC	
s420	34	16	304	9	315	0,97
s641	54	12	228	6	324	0,7
s713	54	12	216	8	432	0,5
s838	66	41	1435	31	2077	0,69
s953	45	30	330	8	360	0,92
s1196	32	46	644	15	480	1,34
s1238	32	57	684	25	800	0,86
s5378	214	78	1326	18	3852	0,34
s9234	247	127	5080	50	12350	0,41
s13207	700	140	2660	2	1400	1,9
s15850	611	151	4379	11	6721	0,65
s38417	1664	327	13734	19	31616	0,43
s38584	1464	169	8281	22	32208	0,26

Table 3: Comparison of the proposed scheme and advanced twisted ring counter approach in [5].

Finally, the proposed two-dimensional reseeding is compared to the pure LFSR encoding reported in [13], which is comparable with respect to flexibility and scan compatibility. The results after 10,000 pseudo-random patterns for both schemes are shown in Table 4.

It can be observed that for all circuits the proposed scheme achieves a higher compression rate. In the best case the number of bits to be stored is reduced down to 11 %, and on the average a reduction of the test data storage by 57 % is obtained. Furthermore, in all cases the proposed scheme requires less polynomials than the conventional reseeding scheme described in [13].

Circuit	#PPIs	Two-dimensional Reseeding		Reseeding of LFSRs [13]		ROM 2D / ROM [13]
		# Polynomials	ROM 2D	# Polynomials	ROM [13]	
s420	34	2	160	3	250	0.64
s641	54	2	64	2	183	0.35
s713	54	2	64	2	183	0.35
s838	66	2	858	6	1623	0.53
s953	45	2	20	4	141	0.14
s1196	32	2	30	4	267	0.11
s1238	32	2	42	4	249	0.17
s5378	214	2	196	3	726	0.27
s9234	247	2	3800	8	6923	0.55
s13207	700	2	1044	6	3570	0.29
s15850	611	2	3360	6	6528	0.51
s38417	1664	2	11214	6	24283	0.46
s38584	1464	2	2891	3	3406	0.85
c2670	233	2	1036	5	3412	0,30
c7552	207	2	4788	12	5241	0,91

Table 4: Comparison of the proposed technique to the reseeding of multiple polynomial LFSRs [13].

Overall, the experimental experience shows that there is a trade-off between scan compatibility and the memory size required to implement a store-and-generate approach. Compared to previously published approaches providing the same flexibility and scan compatibility the proposed two-dimensional reseeding approach clearly offers the most efficient solution.

6 Conclusions

A new and efficient scheme for scan-based BIST has been presented. This scheme applies a two-dimensional reseeding technique which encodes deterministic test cubes as seeds of a folding counter sequence and again compresses the folding seeds into LFSR seeds. This way the amount of the test data storage is reduced considerably. Moreover, the simple and regular structure of the pattern generator is completely compatible with standard scan design, allows an efficient hardware implementation and 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 International Symposium on Circuits and Systems, Kyoto, 1985

3 F. Brglez, D. Bryan and K. Kozminski: Combinational Profiles of Sequential Benchmark Circuits; Proc. IEEE International Symposium 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 Symposium, Dana Point, CA, 1999, pp. 22-27

5 K. Chakrabarty and S. Swaminathan: Built-in self testing of high-performance circuits using twisted-ring counters; Proceedings 2000 IEEE International Symposium on Circuits and Systems, 2000, pp. I-72 - I-76

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

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

8 K.-T. Chen, C.-J. Lin: Timing Driven Test Point Insertion for Full-Scan and Partial-Scan BIST, Proceedings IEEE International Test Conference, Washington, DC, 1995, pp. 506-514

9 C. Dufaza, G. Cambon: LFSR based Deterministic and Pseudo-Random Test Pattern Generator Structures; Proceedings European Test Conference, Munich, 1991, pp. 27-34

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

11 S. Hellebrand, H.-G. Liang, H.-J. Wunderlich: A Mixed Mode BIST Scheme Based on the Reseeding of Folding Counters; Proceedings IEEE International Test Conference, Atlantic City, NJ, October 2000, pp. 778-784

12 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

13 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

14 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

15 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

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

17 G. Kiefer, H. Vranken, E. J. Marinissen, H.-J. Wunderlich: Application of Deterministic Logic BIST on Industrial Circuits; Proc. IEEE International Test Conference, ITC 2000, Atlantic City, NJ, October 3 - 5, 2000

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

19 E. J. McCluskey: Verification Testing - A Pseudoexhaustive Test Technique; IEEE Transactions on Computers, Vol. C-33, No.6, June 1984, pp. 541–546

20 Y. Savaria, M. Yousef, B. Kaminska, M. Koudil: Automatic Test Point Insertion for Pseudo-Random Testing; Proceedings International Symposium on Circuits and Systems, 1991, pp. 1960-1963

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

22 M. J. Y. Williams, J. B. Angell: Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic; IEEE Transactions on Computers, Vol. C-22, No. 1, January 1973, pp. 46-60

23 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