Pattern Generation for a Deterministic BIST Scheme

Sybille Hellebrand*, Birgit Reeb*, Steffen Tarnick**, Hans-Joachim Wunderlich*
* University of Siegen, Germany
** Max-Planck Society, University of Potsdam, Germany

Abstract

Recently a deterministic built-in self-test scheme has been presented based on reseeding of multiple-polynomial linear feedback shift registers. This scheme encodes deterministic test sets at distinctly lower costs than previously known approaches. In this paper it is shown how this scheme can be supported during test pattern generation. The presented ATPG algorithm generates test sets which can be encoded very efficiently. Experiments show that the area required for synthesizing a BIST scheme that encodes these patterns is significantly less than the area needed for storing a compact test set. Furthermore, it is demonstrated that the proposed approach of combining ATPG and BIST synthesis leads to a considerably reduced hardware overhead compared to encoding a conventionally generated test set.

1. Introduction

The efficiency of a built-in self-test (BIST) implementation is characterized by the test length and the hardware overhead required to achieve complete or sufficiently high fault coverage. Various BIST architectures based on pseudo-exhaustive, random, weighted random and deterministic patterns offering different trade-offs between the two parameters have been developed in the past [1, 2, 4, 5, 8, 9, 14, 25, 26, 27].

This paper targets an efficient test-per-scan architecture combining pseudo-random and deterministic BIST. Such a scheme is very attractive because of the moderate hardware overhead and the simplicity of the implementation. The LFSR required for test pattern generation can be synthesized automatically together with the circuit structure, and if the synthesized circuit is completely testable by an acceptable number of patterns, a "one-pass" synthesis is sufficient. If the circuit contains random pattern resistant faults the pseudo-random BIST architecture can easily be extended to a mixed mode scheme which combines a pseudo-random sequence of limited length and deterministic patterns for the hard to detect faults [15, 18, 23]. The hardware overhead is then determined by the storage requirements for the deterministic patterns.

In [15] a mixed mode test-per-scan architecture based on multiple-polynomial LFSRs has been presented which allows a very efficient encoding of the deterministic test vectors. This approach exploits the fact that in many cases the deterministic patterns are not fully specified: a test pattern with s specified bits can be encoded into an s bit word with a very high probability of success. Further optimizations are possible for complete test sets [16, 23]. The actual storage capacity for the deterministic test set, however, strongly depends on the properties of the ATPG algorithm used to generate the patterns. With $s(t)$ denoting the number of specified bits in a test pattern t the storage amount for a test set $T = \{t_1, \ldots, t_N\}$ is determined by the maximum number of specified bits $s_{max} = \max \{s(t) \mid t \in T\}$ and the distribution of the numbers $s(t_1), \ldots, s(t_N)$.

Automatic test pattern generation has been a major concern of research for many years, and powerful and efficient algorithms have been developed [10, 11, 21, 24]. To support deterministic and mixed mode BIST a number of procedures targeting minimal test sets have been proposed [13, 17, 19, 20, 22]. Although some of these procedures try to maximize the number of unspecified bits in intermediate steps, the number and distribution of specified bits in the final test set has not been particularly addressed. In this paper a procedure for ATPG is proposed which puts additional focus on generating an "efficiently encodable" test set. To minimize the storage amount for the final encoded test set the algorithm interactively combines the ATPG and the encoding process.

Before the proposed ATPG approach is described in more detail in Section 3, the underlying BIST architecture will be sketched briefly in Section 2. Experimental results will be discussed in Section 4.

Part of the work has been supported by the DFG grant „Test und Synthese schneller eingebetteter Systeme“ and by the ESPRIT Project 7107 ARCHIMEDES.

2. The Target BIST Scheme

A “test-per-scan” architecture is assumed where the scan chain includes m flip-flops corresponding to the width of a test pattern. The BIST scheme is based on multiple-polynomial LFSRs (see Figure 1) [15].

The LFSR can operate to a limited number of different feedback polynomials, and is used for both the generation of pseudo-random patterns and the decompression of encoded deterministic patterns. A deterministic pattern is encoded as a polynomial identifier (abbreviated as “id” in Figure 1) and a seed for the respective polynomial. During test mode the pattern can be reproduced by establishing the feedback links corresponding to the polynomial identifier, loading the seed into the LFSR and performing m autonomous transitions of the LFSR. After the m-th transition the scan chain contains the desired pattern which is then applied to the CUT.

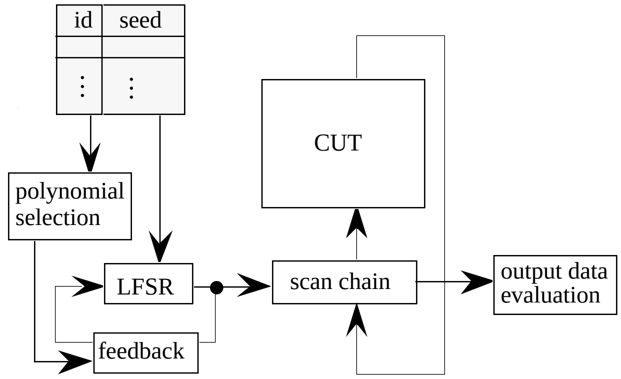


Figure 1: BIST scheme based on a multiple-polynomial LFSR.

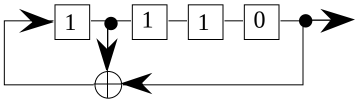
To calculate the encoding a system of linear equations has to be solved. For a fixed feedback polynomial $h(X) = X^k + h_{k-1} \cdot X^{k-1} + \ldots + h_0$ the LFSR produces an output sequence $(a_i)_{i \geq 0}$ satisfying the feedback equations $a_i = a_{i-k} \cdot h_0 + a_{i-k+1} \cdot h_1 + \ldots + a_{i-1} \cdot h_{k-1}$ for all $i \geq k$. The LFSR-sequence is compatible with a desired test pattern $t = (t_1, \ldots, t_m)$ if for all specified bits $a_i = t_i$ holds. Recursively applying the feedback equation provides a system of linear equations in the seed variables $a_0, \ldots, a_{k-1}$. If no solution can be found for the given polynomial the next available polynomial is tried, and in [15] it has been shown that already for 16 polynomials there is a very high probability of success that a deterministic pattern with s specified bits can be encoded into an s-bit seed. The identifier for the required feedback polynomial can be omitted if the seeds for specific polynomials are grouped together and a “next-bit” is used to indicate whether the feedback polynomial has to be changed [16].

Hence, for a test set $T = \{t_1, \ldots, t_N\}$ with maximum number of specified bits $s_{max} = \max \{s(t) \mid t \in T\}$ the seeds and the next bits require $(s_{max} + 1) \cdot N$ bits of storage. If P polynomials are used additional $s_{max} \cdot P$ bits are required for storing the feedback taps, such that the overall storage requirements are $S(T) := (N + P)s_{max} + N$ bits. Minimizing $S(T)$ requires both minimizing the maximal number of carebits s_{max} and the number of patterns N.

The number of patterns which have to be encoded and the number of feedback polynomials can be reduced significantly if “concatenation” of test patterns as described in [16] is supported. This technique makes use of the fact that the length of an encoded test pattern is independent of the length of the original test pattern. Thus, if a subset $T' \subset T$ of test patterns is concatenated to one long test pattern $t_{con}(T')$ whose number of care bits is not exceeding s_{max}, then encoding the pattern $t_{con}(T')$ requires the same number of bits as each of the original patterns. Furthermore, if T' consists of M patterns $t_1, \ldots, t_M$, then any permutation of these patterns can be used to build the concatenated pattern $t_{con}(T')$. Since it is sufficient to find an encoding for one of the $M!$ possible patterns representing T', the probability to find an encoding for T' as a seed of a specific polynomial is increased or, equivalently, a high probability of successful encoding can be guaranteed with a reduced number of polynomials. Figure 2 illustrates this with the help of an example.

$T = \{t_1, t_2, t_3\}$, $m = 5$, $s_{max} = 4$, $h(X) = X^4 + X^3 + 1$

$t_1 = (x, x, 1, 1, x)$,
$t_2 = (x, 1, x, x, 0)$,
$t_3 = (1, 0, x, 0, 1)$



$T' = \{t_1, t_2\}$

1st possibility to concatenate t_1 and t_2:

$t_{con}(T') = (x, x, 1, 1, x, x, 1, x, x, 0)$

Equations: $a_0 = 0$, $a_3 = 1$,
$a_6 = a_0 + a_1 + a_2 + a_3 = 1$
$a_7 = a_0 + a_1 + a_2 = 1$
$\Rightarrow$ no solution

2nd possibility to concatenate t_1 and t_2:

$t_{con}(T') = (x, 1, x, x, 0, x, x, 1, 1, x)$

Equations: $a_1 = 1$, $a_2 = 1$,
$a_5 = a_0 + a_1 + a_3 = 0$
$a_8 = a_1 + a_2 + a_3 = 1$
$\Rightarrow$ $a_3 = 1$, $a_0 = 0$

Sequence generated by the seed (1, 1, 1, 0):

$$t_{seq} = (\underbrace{1, 1, 0, 1, 0}_{t_2'}, \underbrace{1, 1, 1, 1, 0}_{t_1'})$$

Figure 2: Concatenation of test patterns.

If the patterns of the test set T in Figure 2 are encoded separately, three seeds for the polynomial $h(X)$ have to be stored. If concatenation is used, two seeds are sufficient: one for $T' = \{t_1, t_2\}$ and one for $T'' = \{t_3\}$.

To implement a test set grouped into concatenated patterns a slight variation of the BIST scheme shown in Figure 1 is used. Assume that T is grouped into G subsets T_1, …, T_G consisting of at most M patterns each, then once the seed for a pattern $t_{con}(T_i)$ is loaded, the LFSR works in autonomous mode for $M \cdot m$ clock cycles. After each m cycles a test pattern is completely loaded into the scan chain and can be applied to the circuit under test. For subsets containing less than M patterns this implies the application of some additional random patterns. The storage requirements for T are reduced to $S(T) = (G + P)s_{max} + G$ bits. The number of carebits accepted in a concatenated pattern can be increased to a parameter $s_B \geq s_{max}$ to allow even better compaction. In [16] an algorithm has been proposed to minimize G for a given test set T. In the next section it will be shown how the possibilities for concatenation can be exploited during ATPG.

3. Automatic Test Pattern Generation

In this section the ATPG procedure SCARLETT (**S**elf-Test **C**odable **A**nd **R**educed **LE**ngth **T**est **T**ool) is presented which is specifically tailored for the BIST scheme of Section 2.

The proposed procedure is applied to the hard-to-detect faults remaining after a pseudo-random sequence of length R and minimizes the storage requirements $S(T)$ for the resulting deterministic test set T. The proposed algorithm supports the concatenation of patterns and requires as input parameters the maximum number of patterns M and the maximum number s_B of specified bits accepted in a group of patterns to be concatenated. The basic idea to achieve an efficient encoding is to alternate test pattern generation and the encoding of test patterns as sketched in Figure 3.

Test patterns are generated until the limits M or s_B for a group of patterns to be concatenated are reached. All patterns in this group are concatenated to one pattern t_{con} which is encoded as a seed for one of the available feedback polynomials, and the first $M \cdot m$ bits of the LFSR sequence resulting from this seed are determined. Since this subsequence corresponds to the fully specified patterns fed into the scan chain during test mode, it is fault-simulated against the remaining fault set. Faults which are detected in addition to the original target faults can be dropped immediately, and the process is repeated. The main procedures of the complete algorithm are explained in more detail in the sequel.

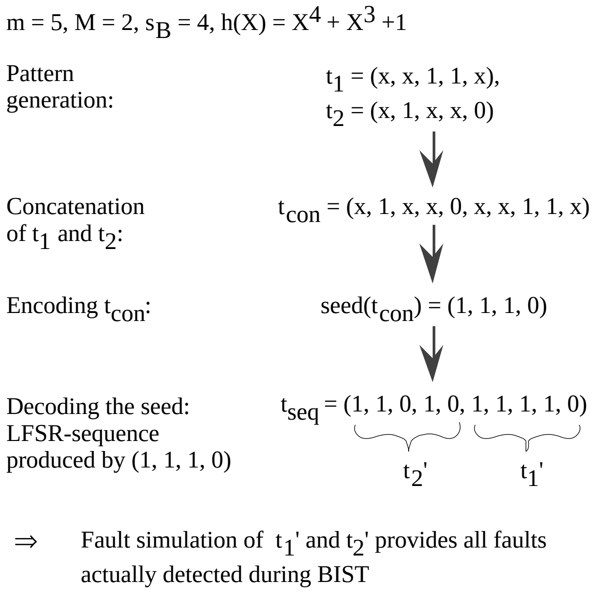


Figure 3: Example for alternating test pattern generation and encoding.

3.1 *Preprocessing*

Since the supported BIST approach uses the same LFSR to generate pseudo-random and deterministic test vectors, the degree of the feedback polynomial should be fixed before simulating the pseudo-random sequence. To guarantee that the deterministic patterns for the hard-to-detect faults can be encoded with a high probability of success the degree should be selected equal to the parameter $s_B \geq s_{max} = \max\{s(t) \mid t \in T\}$, where T is the deterministic test set for the hard-to-detect faults. In the preprocessing phase ATPG is performed for all faults in the circuit to get an approximation for s_{max} and thus a guideline how to select s_B. Also, redundant faults are eliminated during preprocessing.

Next, for a number of different primitive polynomials of degree s_B a pseudo-random sequence of length R is generated and fault simulated. The polynomial corresponding to the pseudo-random sequence with the highest fault coverage is selected for pseudo-random pattern generation.

3.2 *Generation of a test pattern group*

A subset T' of test patterns can be grouped together for concatenation if

$$|T'| \leq M \text{ and } \sum_{t \in T'} s(t) \leq s_B$$

hold. Assume that k patterns t_1, …, t_k have already been generated and the group is not yet complete, i. e.

$$k \leq M \text{ and } \sum_{i=1}^{k} s(t_i) < s_B.$$

Then there are two chances to detect an additional fault from the list of remaining faults F:

1) By increasing the number of specified bits in one of the patterns $t_1, \dots, t_k$, i.e. one pattern t_j is replaced by a pattern t_j* which is covered by t_j. This approach is also known as "dynamic compaction", but in contrast to classical applications here only a restricted number of bits can be specified additionally [12].
2) If $k < M$, then an additional fault can also be detected by a new pattern t_{k+1}.

If $k = 0$ or all of the patterns $t_1, \dots, t_k$ are already fully specified, dynamic compaction is not possible. Also, if the number of specified bits in each of the patterns $t_1, \dots, t_k$ exceeds a user-defined limit, dynamic compaction is not expected to be successful and not considered therefore. To keep the final test set small, in this case a target fault $f \in F$ is selected, such that the number of undetected faults on a path from the fault location to a primary output is maximal. A new pattern t_{k+1} is generated for f and the resulting number of carebits

$$s = \sum_{i=1}^{k} s(t_i) + s(t_{k+1})$$

is determined. If $s > s_B$ the fault f cannot be detected within the current test group. It is postponed for the next test group, and a new target fault f' is selected from F.

In all other cases, both possibilities for additional fault detection are investigated as follows. From the list of patterns $t_1, \dots, t_k$ the pattern with the least number of specified bits is selected and for all circuit nodes the observabilities corresponding to this partial assignment are computed by critical path tracing [3]. The first observable fault $f \in F$ is selected as new target fault. All patterns $t_1, \dots, t_k$ are checked if they cover a pattern for f, and a pattern t_j* is determined such that the resulting number of carebits

$$s_1 = \sum_{i=1, i \neq j}^{k} s(t_i) + s(t_j*)$$

is minimal. Additionally, a new pattern t_{k+1} is generated for f and

$$s_2 = \sum_{i=1}^{k} s(t_i) + s(t_{k+1})$$

is computed. If both $s_1 > s_B$ and $s_2 > s_B$ the fault f cannot be detected within the current test group, and a new target fault f' is selected from F. Otherwise, if $s_1 \le s_B$ or $s_2 \le s_B$ or both $s_1, s_2 \le s_B$, the possibility resulting in a minimal number of carebits is chosen. The computational effort for this procedure can be reduced by checking only those pattern in $\{t_1, \dots, t_k\}$ where the number of specified bits does not exceed a user-defined limit.

The underlying ATPG-procedure is based on the FAN-algorithm [10]. Both static and dynamic global implications and unique sensitization techniques are applied to accelerate the process of test pattern generation and the identification of redundancies [21]. Decisions are guided by a number of heuristics which particularly aim at generating test patterns with a large number of unspecified bits, and keeping the overall test set small. As mentioned above, the heuristics use observability values which are determined by critical path tracing and which are updated dynamically for each partial assignment. Controllability values are used as follows: the i-controllability of a node corresponds to the minimal number of primary inputs in order to put the value i on that node.

For the propagation of fault effects a node on the D-frontier is selected which is as close as possible to the primary outputs and is located on a path with a maximum number of undetected faults. To minimize the number of specified bits, the number of primary inputs which must be set to propagate the fault effect, is used as a second criterion.

For line justification the user can chose between "rotating backtrace" as introduced in [19] or a mechanism based on observability values as follows: When there is a choice of a gate input line to be set to a controlling value, two cases are distinguished:

a) If the gate output is observable a gate input line is selected such that the number of undetected faults preceding this line is maximum.
b) If the gate output is not observable a gate input line is selected such that the number of primary inputs to be set is minimal.

In addition to that, the "maximal compaction" technique suggested in [19] is used to minimize the number of specified bits in a test pattern. For each specified bit also the complementary logic value is simulated. If the target fault is still detected the bit position is considered as don't care. Since only single bits are flipped while keeping the original values for the other specified bits, the resulting pattern need no longer be a test for the target fault. Experience shows that this is not very likely to happen, but if the fault simulation step in Figure 3 reveals such a problem the fault is processed again without maximal compaction.

3.3 Encoding and fault simulation

To encode a group T' of test patterns the following steps are performed:

1) If $|T'| < M$ is true, then $M - |T'|$ unspecified "dummy" patterns are added to T'.
2) A feedback polynomial $h(X)$ of degree s_B is selected from the table of primitive polynomials. To reduce the overall number P of feedback functions to be implemented, the polynomials required for pseudo-random pattern generation or for other test groups already encoded are tried first.
3) A permutation of the patterns $t_1, \dots, t_M$ in T' is generated, and the system of linear equations correspond-

ing to the concatenated pattern t_{con} and the polynomial $h(X)$ is derived as described in Section 2. Standard techniques for solving linear equations are applied to this system. If a solution exists the seed value and the polynomial identifier are stored and the pattern t_{seq} consisting of the $M \cdot m$ first bits of the corresponding LFSR-sequence is calculated. If there is no solution another permutation of $t_1, \ldots, t_M$ is generated and analyzed.

4) If no encoding can be found in step 3, the process is repeated for another feedback polynomial.

The encoding procedure provides a seed for a polynomial $h(X)$, for which during test mode the first $M \cdot m$ autonomous cycles of the LFSR produce a pattern t_{seq}. This process is simulated and the resulting pattern t_{seq} is split into M single patterns (the i-th pattern consisting of the i-th m bits of t_{seq}) corresponding to the patterns generated for the current test group. In contrast to the original patterns in the test group, the patterns obtained from t_{seq} are fully specified and will be actually applied to the circuit under test. Fault simulation performed for these patterns is thus less complex and allows to drop faults immediately which are additionally detected by the LFSR-sequence.

4. Experimental Results

A series of experiments has been performed to determine the trade-offs between the length of the pseudo-random sequence and the storage requirements for the deterministic patterns. For the first experiment a complete deterministic BIST has been assumed. Allowing $M = 8$ patterns to be concatenated, the storage requirements $S(T_{enc})$ for an encoded test set T_{enc} generated by the presented tool SCARLETT have been compared to the number of bits $S(T_{comp})$ in the minimum deterministic test set reported in any of the papers [13, 17, 20, 22]. Tables 1 and 2 show the results for the ISCAS-85 and ISCAS-89 circuits [6, 7]. For the circuits s35932 and s38584 a compact test set has been generated by an own ATPG implementation similar to the one described in [19].

The columns of Table 1 list the number of primary inputs pi, the maximal number s_{max} of specified bits after preprocessing the number of specified bits s_B accepted in M patterns to be concatenated, the number of testgroups G, the required number of feedback polynomials P and the overall storage requirements $S(T_{enc}) = (G + P) \cdot s_B + G$ for the presented approach. For comparison, in Table 2 the number of primary inputs, the size of the compact test set $|T_{comp}|$ and the number of bits necessary to store the compact test set $S(T_{comp}) = |T_{comp}| \cdot pi$ are shown. The last column of Table 2 reports the ratio $S(T_{enc})/S(T_{comp})$. In cases with s_{max} being very large also values $s_B < s_{max}$ have been tried successfully.

Circuit	pi	s_{max}	s_B	G	P	$S(T_{enc})$
c432	36	20	30	16	2	556
c499	41	41	41	23	5	1171
c880	60	26	27	19	5	667
c1355	41	41	41	41	1	1763
c1908	33	31	33	67	5	2443
c2670	157	48	60	59	6	3959
c3540	50	28	30	48	5	1638
c5315	178	47	80	30	3	2670
c6288	32	32	32	6	2	262
c7552	206	130	100	74	18	9218
s208	19	12	12	16	2	232
s298	17	7	10	10	2	130
s344	24	9	9	7	2	88
s349	24	9	9	8	2	98
s382	24	9	9	13	3	157
s386	13	11	12	34	2	466
s420	35	20	20	29	4	689
s444	24	11	9	12	2	138
s510	25	9	9	20	2	218
s526	24	14	14	36	2	568
s526n	24	14	14	32	2	508
s641	54	22	24	24	3	672
s713	54	22	22	27	4	686
s820	23	13	13	65	3	949
s832	23	14	13	62	5	933
s838	67	36	36	51	4	2013
s953	45	15	15	46	4	796
s1196	32	17	17	82	6	1578
s1238	32	17	17	84	7	1614
s1423	91	41	42	26	8	1328
s1488	14	12	12	56	3	764
s1494	14	12	12	55	3	751
s5378	214	29	27	129	3	3973
s9234	247	63	60	166	5	10426
s13207	700	24	30	244	4	7684
s15850	611	44	40	222	5	9302
s35932	1763	8	11	16	3	225
s38417	1664	84	91	403	6	37622
s38584	1464	55	70	182	2	13062

Table 1: Number of bits to be stored for encodable test sets $S(T_{enc})$ for $M = 8$.

For most of the circuits, the proposed approach reduces the storage requirements down to around 25% - 50%. For the larger circuits with a large number of primary inputs the gain is still significantly higher. For the circuits s13207 to s38584 the necessary memory for test data is reduced down to 1% - 25%.

This experiment has been repeated for pseudo-random sequences of 1000 and 10000 patterns preceding the deterministic test pattern generation. For the remaining faults

Circuit	pi	$\|T_{comp}\|$	$S(T_{comp})$	$\frac{S(T_{enc})}{S(T_{comp})}$
c432	36	29	1044	0.53
c499	41	52	2132	0.55
c880	60	21	1260	0.53
c1355	41	84	3444	0.51
c1908	33	108	3564	0.69
c2670	157	51	8007	0.49
c3540	50	97	4850	0.34
c5315	178	49	8722	0.31
c6288	32	16	512	0.51
c7552	206	84	17304	0.53
s208	19	27	513	0.45
s298	17	23	391	0.33
s344	24	15	360	0.24
s349	24	13	312	0.31
s382	24	25	600	0.26
s386	13	64	832	0.56
s420	35	43	1505	0.46
s444	24	24	576	0.24
s510	25	55	1375	0.16
s526	24	50	1200	0.47
s526n	24	51	1224	0.42
s641	54	24	1296	0.52
s713	54	23	1242	0.55
s820	23	95	2185	0.43
s832	23	96	2208	0.42
s838	67	75	5025	0.40
s953	45	77	3465	0.23
s1196	32	117	3744	0.42
s1238	32	129	4128	0.39
s1423	91	29	2639	0.50
s1488	14	102	1428	0.54
s1494	14	101	1414	0.53
s5378	214	104	22256	0.18
s9234	247	116	28652	0.36
s13207	700	235	164500	0.05
s15850	611	113	69043	0.13
s35932	1763	18	31734	0.01
s38417	1664	91	151424	0.25
s38584	1464	141	206424	0.06

Table 2: Number of bits to be stored for compact test sets $S(T_{comp})$ and ratio $S(T_{enc})/S(T_{comp})$.

an encodable test set provided by SCARLETT has been compared to a compact test set generated by the own ATPG implementation mentioned before.

Both experiments showed the same trends as observed for the first experiment. The results for 10000 random patterns are listed in Table 3 and 4; examples where only a few or no patterns remain are not reported. Here s_{max} denotes the maximal number of specified bits in a test set generated for the remaining faults.

Circuit	pi	s_{max}	s_B	G	P	$S(T_{enc})$
c2670	157	48	60	52	4	3412
c7552	206	100	100	41	11	5241
s420	35	20	20	10	2	250
s641	54	22	22	7	1	183
s713	54	22	22	7	1	183
s838	67	36	36	39	5	1623
s953	45	15	15	6	3	141
s1196	32	17	17	12	3	267
s1238	32	17	17	11	3	249
s5378	214	19	27	24	2	726
s9234	247	66	61	103	7	6923
s13207	700	24	24	138	5	3570
s15850	611	45	46	134	5	6528
s35932	1763	9	11	6	2	83
s38417	1664	72	91	259	5	24283
s38584	1464	55	70	46	2	3406

Table 3: $S(T_{enc})$ after a pseudo-random sequence of 10 000 patterns, $M = 8$.

Circuit	pi	$\|T_{comp}\|$	$S(T_{comp})$	$\frac{S(T_{enc})}{S(T_{comp})}$
c2670	157	44	6908	0.49
c7552	206	35	7210	0.73
s420	35	10	350	0.71
s641	54	7	378	0.48
s713	54	7	378	0.48
s838	67	42	2814	0.58
s953	45	6	270	0.52
s1196	32	12	384	0.70
s1238	32	11	352	0.71
s5378	214	26	5564	0.13
s9234	247	95	23465	0.30
s13207	700	78	54600	0.07
s15850	611	33	20163	0.32
s35932	1763	5	8815	0.01
s38417	1664	85	141440	0.17
s38584	1464	41	60024	0.06

Table 4: $S(T_{comp})$ and the ratio $S(T_{enc})/S(T_{comp})$ after a pseudo-random sequence of 10 000 patterns, $M = 8$.

Tables 3 and 4 show that for the smaller circuits the presented approach reduces the test data storage down to 30% - 70%, but for the larger circuits a reduction down to 1% - 30% is achieved. In all experiments, the number of bits required for easily encodable test sets is significantly smaller than the number of bits of a compact test set.

Comparing the presented approach of combining ATPG and encoding of patterns to the encoding of complete test sets also a considerable gain in efficiency can be observed. In [23, 16] results for the circuits s5378, s9234 and s13207 after 1000 and 10000 random patterns are reported, where patterns generated by SOCRATES were encoded. Table 5 compares the storage requirements in bits to the proposed approach and to storing compact test sets.

Circuit	# random patterns	$S(T_{enc})$	[23]	$S(T_{comp})$
s5378	1000	2629	11008	13696
	10000	726	4096	5564
s9234	1000	9329	19152	28405
	10000	6923	13482	23465
s13207	1000	6720	59175	119700
	10000	3570	6615	54600

Table 5: $S(T_{enc})$ compared to $S(T_{comp})$ and the storage requirements after encoding a complete test set.

5. Conclusions

A deterministic BIST scheme is feasible if already during ATPG additional requirements are taken into account. A compact deterministic test set is often not the best choice as both the number of carebits and the number of patterns determine the size of the BIST memory. By combining pattern generation and pattern encoding the amount of bits to be stored is significantly reduced compared approaches known before.

References

1 V.K. Agarwal, E. Cerny: “Store and Generate Built-In Testing Approach”, Proc. 11th Int. Symp. Fault-Tolerant Comput., 1981, pp. 35-40

2 S.B. Akers and W. Jansz: “Test Set Embedding in Built-In Self-Test Environment”, Proc. IEEE Int. Test Conf., Washington D.C., 1989, pp. 257-263

3 M. Abramovici, P.R. Melon, D.T. Miller: “Critical Path Tracing - An Alternative to Fault Simulation”, Proc. of 20th Design Automation Conf., 1983, pp. 214-220

4 P. Bardell, W.H. McAnney and J. Savir: “Built-In Test for VLSI”, New York: Wiley-Interscience, 1987

5 Z. Barzilai, D. Coppersmith, A.L. Rosenberg: “Exhaustive Generation of Bit Patterns with Applications to VLSI Self-Testing” IEEE Trans. on Comp., Vol. C-32, No. 2, Feb. 1983, pp. 190-194

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

7 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

8 F. Brglez et al: “Hardware-Based Weighted Random Pattern Generation for Boundary-Scan” Proc. IEEE Int. Test Conf., Washington D.C., 1989, pp. 264-274

9 W. Daehn, J. Mucha: “Hardware Test Pattern Generators for Built-In Test”, Proc. IEEE Int. Test Conf., 1981, pp. 110-113

10 H. Fujiwara and T. Shimono: “On the Acceleration of Test Generation Algorithms”, IEEE Trans. on Comp., Vol. C-32, No. 12, December 1983, pp. 1137-1144

11 P. Goel: “An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits”, IEEE Trans. on Comp., Vol. C-30, No. 3, March 1981, pp. 215-222

12 P. Goel and B. C. Rosales: “Test Generation and Dynamic Compaction of Tests”, Proc. IEEE Test Conf., Cherry Hill, N. J., 1979, pp. 189-192

13 H. Higuchi, N. Ishiura, and S. Yajima: “Compaction of Test Sets Based on Symbolic Fault Simulation”, Synthesis and Simulation Meeting and Int. Interchange, pp. 253-262, 1992

14 S. Hellebrand, H.-J. Wunderlich, O. F. Haberl: “Generating Pseudo-Exhaustive Vectors for External Testing”, Proc. IEEE Int. Test Conf., Washington DC, 1990, pp. 670-679

15 S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois: “Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers”, Proc. IEEE Int. Test Conf., Baltimore 1992, pp. 120-129

16 S. Hellebrand et al.: “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, Feb. 1995, pp. 223-233

17 S. Kajihara, I. Pomeranz, K. Kinoshita, S. M. Reddy: “Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits”, Proc. 30th ACM/IEEE Design Automation Conf., 1993, pp. 102-106

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

19 I. Pomeranz, L.N. Reddy and S.M. Reddy: “COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits”, Proc. Int. Test Conf., 1991, pp. 194-203

20 L.N. Reddy, I. Pomeranz, and S.M. Reddy: “ROTCO: A Reverse Order Test Compaction Technique”, Proc. IEEE EURO-ASIC Conf., September 1992, pp. 189-194

21 M. Schulz and E. Auth: “Advanced Automatic Test Generation and Redundancy Identification Techniques”, Proc. 18th Int. Symp. Fault-Tolerant Comput., Tokyo 1988, pp. 30-35

22 G. Tromp: “Minimal Test Sets for Combinational Circuits”, Proc. IEEE Int. Test Conf., 1991, pp. 204-209

23 S. Venkataraman, J. Rajski, S. Hellebrand and S. Tarnick: “An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers”, Proc. IEEE/ACM Int. Conf. on Computer-Aided Design, Santa Clara 1993, pp. 572-577

24 J. A. Waicukauski, P. A. Shupe, D. J. Giramma, and A. Matin: “ATPG for Ultra-Large Structured Designs”, Proc. IEEE Int. Test Conf., Washingtion, D.C., 1990, pp. 44 - 51

25 L.T. Wang and E.J. McCluskey: “Circuits for Pseudo-Exhaustive Test Pattern Generation” Proc. IEEE Int. Test Conf., 1986, pp. 25-37

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

27 H.-J. Wunderlich: “Multiple Distributions for Biased Random Test Patterns”, Proc. IEEE Int. Test Conf., Washington D.C. 1988, pp. 236-244