SEED AND POLYNOMIAL SELECTION ALGORITHM FOR LFSR TEST PATTERN GENERATOR IN BUILT-IN SELF-TEST ENVIRONMENT

Bogdan Dugonik, Zmago Brezočnik

Fakulteta za elektrotehniko, računalništvo in informatiko,
Univerza v Mariboru, Slovenija

Abstract: Testing integrated circuits is of crucial importance to ensure a high level of quality of product functionality. Testing a digital circuit involves applying an appropriate set of input patterns to the circuit and checking for the correct outputs. The conventional approach is to use an external tester (automatic test equipment - ATE) to perform the test. Built-in self test (BIST) techniques have been developed in which some or all test functions are incorporated on the chip. In today's integrated circuits, BIST is becoming increasingly important as designs become more complicated and the density of VLSI circuits increases. The BIST approach offers economic benefits, reuse of logic circuits as well as some significant opportunities in hierarchical testing. On the other hand, the classical testing approach and the use of automatic test equipment is becoming a less important part in the testing process. In this paper, we give an overview of BIST methods and present some advanced new BIST solutions for testing combinational circuits. We propose an algorithm for seeding the LFSR-based test pattern generators. We achieved nearly 100% test fault coverage for a given test lengths. Our method can be used for both test-per-scan and test-per-clock BISTs. The goal of the proposed method is to minimize the test lengths through suitable selection of initial seeds, to minimize hardware overhead and achieve sufficiently high fault coverage. While semi-random generated test cubes are additionally completed by deterministically calculated test vectors, the achieved fault coverage is very high. The experiments were made on ISCAS85 (ISCAS89) benchmark circuits /7/.

Algoritem za izbiro semena in polinoma generatorja testnih vzorcev LFSR za vgrajeno samotestiranje

Izvleček: Testiranje digitalnih vezij je bistvenega pomena, da bi zagotovili visoko stopnjo funkcijske zanesljivosti ter kakovosti proizvedenega vezja. Pri postopku testiranja na vhodne priključke vezja privedemo določen nabor testnih vektorjev in preverjamo pravilnost odzivov na izhodnih priključkih. Pri klasičnem načinu testiranja v ta namen uporabljamo testne naprave (ATE). Metode, pri katerih se del testiranja izvede znotraj samega vezja, se vedno bolj uporabljajo in prevzemajo pomembno vlogo pri testiranju sodobnih vezij. Današnje tehnologije omogočajo visoko gostoto vezij in dostopnost do testnih priključkov je vedno bolj otežena. Problem dostopa in zmanjšanja odvisnosti od uporabe dragih naprav ATE je mogoče rešiti z uporabo vgrajenih testnih metod BIST. Razen ekonomskih ima BIST še vrsto drugih prednosti, kot je pouporaba testnih modulov in možnost za izvedbo hierarhičnega testiranja. V članku predstavimo osnove vgrajenega testa BIST in nov pristop k testiranju kombinacijskih logičnih vezij. Podan je algoritem za ugotavljanje najprimernejšega semena in polinoma iz določene končne množice psevdonaključnih generatorjev za generiranje naključnih testnih vektorjev. Z izbranim naborom testnih vektorjev zagotovimo visoko stopnjo pokritja napak z minimalno potrebno testno dolžino. Metodo lahko uporabimo za dva načina izvedbe testa (TPC in TPS). Cilj predlagane metode je sistematično določanje primerne začetne vrednosti (semena), kot tudi vrste polinoma za psevdonaključni generator. Naključne generatorje izberemo tako, da bi jih lahko realizirali z najmanj dodatnimi elementi. Skrajšati želimo potrebni čas za samo izvedbo testa s predpostavko, da bi dosegli čim višje pokritje napak. Naključno dobljen nabor testnih vektorjev primerjamo z deterministično izračunanimi testnimi vektorji. Eksperimenti so prikazani na standardnem naboru primerljivih vezjih iz družine ISCAS85 (iSCAS89) /7/.

1. Introduction

Testing of integrated circuits is of crucial importance in ensuring a high level of quality in product functionality. The increasing complexity of VLSI systems makes testing a challenging task. Considering that testing represents a key cost factor in the production process (a proportion of up to 70% of total product cost), an optimal test strategy can offer a substantial competitive advantage in the market of the semiconductor and electronics industry /27/. Testing effects areas of manufacturing as well as engineering and design. The time and cost are the two main considerations in IC design and production process /17/.

As the density of VLSI circuits increased, it became attractive to integrate dedicated test logic on a chip /2/. Built-in self-test (BIST) techniques enable an integrated circuit (board, system) to test itself. BIST techniques offer a great economic benefit compared to the traditional testing. When testing is built into the hardware, it has ability of being not only fast and efficient but also reusable /18/. Furthermore, BIST offers the capability for hierarchical testing where BIST circuit perform test on chips, boards, and the entire system without external, expensive automatic test equipment /20/. Another important consideration is that BIST allows an IC to be tested at its normal operating speed /2/. Figure 1 illustrates the testing process with classical and basic BIST approach. The basic BIST architecture requires the addition of three hardware blocks to the circuit under test (CUT): a pattern generator, a response analyzer, and a test controller /1/.

For simpler applications the deterministic test patterns are stored in an on-chip ROM instead of using a pattern generator. Since this method requires high storage overhead it leads into a non-practical solution /4/. Instead of pre-

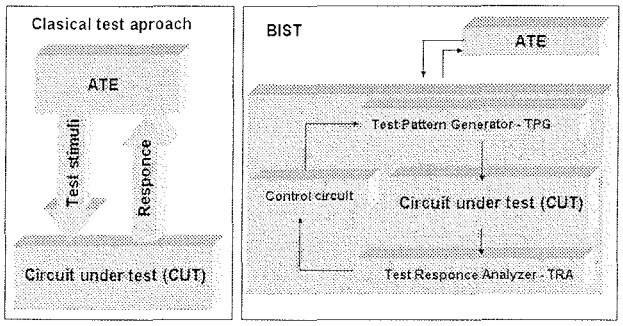


Figure 1: *Classical and BIST testing scheme.*

calculated test vectors a linear feedback shift registers (LFSR) is generally preferred for pattern generation /1/, /2/,/13/. It is able to generate a pseudo-random test patterns with low hardware overhead (easy and cheap to generate).

However, several circuits contain random pattern resistant (r.p.r.) faults which limit the performance of BIST /19/. For background, the r.p.r. faults are faults with low detectability, and therefore require a very long testing time to achieve the appropriate test coverage. While r.p.r. faults are known to be implicated in insufficient fault coverage levels, the selection of the appropriate pseudo-random test pattern generator has been extensively investigated for enhancing fault coverage levels /31/,/22/. A pure pseudo-random test could be very long, hence too expensive in terms of time, to obtain the highest possible fault coverage. Moreover, it is not always guaranteed that the highest possible fault coverage will be achieved even with extremely long pseudo-random test pattern sequence.

A variety of solutions have been suggested to solve this problem /2/. A common method is to modify the circuit by test point insertion (control, observation point), or to redesign it to make it testable via random patterns /25/. Modifying the CUT is rarely a desirable option because of performance degradation and for intellectual property reasons /18/. Another technique is to test with weighted pseudo-random sequences, where the random patterns are biased using additional logic to increase the probability of detecting the r.p.r. faults /13/.

Another widely investigated approach in BIST is the so called mixed-mode testing method /11/. In this BIST technique the circuit is tested in two phases. In the first phase, pseudo-random patterns are applied to cover easy to detect faults /11/. In the second phase, deterministic test patterns are applied to cover the remaining faults. The mixed-mode BIST can be performed in several variations. One is to apply the deterministic patterns from a tester while another way is to store the deterministic patterns in an on-chip ROM. Storing, however, requires a large amount of hardware overhead. As an alternative with lower memory requirements seeds can be stored on the external tester.

Next significant innovation in mixed-mode testing came with the introduction of mapping logic /22/. The idea is to identify patterns in the original generated set that don't detect any new faults and map them by hardware into deterministic patterns.

The other mixed-mode approach is presented as "bit-fixing" method /24/. The deterministic test cubes are embedded in the pseudo-random sequence of bits /8/. Logic is added at the serial output of the LFSR to alter the pseudo-random bit sequence, so that it contains deterministic patterns that detect the r.p.r. faults. This is accomplished by "fixing" certain bits in the pseudo-random test sequence. The method provides 100% fault coverage, but with high hardware overhead.

Wunderlich et.al proposed another mixed-mode technique called the "bit-flipping method" /29/. To avoid storing the deterministic patterns on the chip, the authors show that it is sufficient to alter just a few bits of the general pseudo-random sequence. The efficiency of this scheme relies on the fact that relatively small amounts of the generated pseudo-random patterns are useful for fault detection while others are candidates for altering. Moreover, as a deterministic test pattern usually contains many unspecified bits, there is a very high possibility that one of the ineffective patterns can be modified at just a few bit positions so that it becomes compatible with a previously computed deterministic pattern. Results are provided to show that this scheme represents the most area-efficient solution to-date /29/.

Another possible way to enhance low detectability of the r.p.r. faults is to select the seed very carefully. Several procedures to select seeds have already been studied /5/,/6/,/10/,/14/,/16/. Bayraktaroglu et al. examined the pseudo-random pattern generator structure and selection approaches /5/,/6/. Lempel et al. proposed an LFSR seed-selecting algorithm that used the theory of discrete logarithms /16/. In Fagot et al.'s study /10/, fault simulation computes an efficient LFSR seed which outputs the test sequence including a test cube. To reduce test application time, Ichino et al. used a reseeding method and reverse order simulation /16/.

In this paper we propose a BIST technique that selects the type of LFSR test pattern generator (TPG) and appropriate seed (initial state) to improve random-pattern test quality. The idea is to find a minimal possible set(s) of pseudo-random generated tests to achieve the highest possible test coverage within specified time constraints. An additional effort was invested to embed the test and keep the hardware overhead as low as possible. We consider that the size of the LFSR, its primitive feedback polynomial, and the length of generated test are a priori known. The proposed method is intended to produce a one-seed test sequence of a given length that achieves the highest possible stuck-at fault coverage. The main feature of this technique is a minimal requirement for the additional hardware to cover as many r.p.r. faults as possible. The goal of the presented method is an efficient test pattern generator which guarantees nearly 100% test fault coverage. Furthermore, the proposed technique minimizes the test ap-

plication time, test generation time, and hardware requirements. For most tested circuits we reached a reasonable fault coverage. When these techniques are applied to a mixed-mode BIST such as reseeding, bit-flipping or bit-fixing, the number of additional circuits is generally reduced. With a minor modification of the proposed software the seed-selection methods can be applied not only to typical LFSRs but also to other types of test pattern generators such as Cellular Automata. /29/.

The paper is organized as follows. In next section, we describe the construction methods of LFSR test pattern generator. Two basic BIST techniques are then described. In section 3, we briefly present some necessary definitions. In section 4, we introduce the proposed method for generating an efficient one-seed LFSR test sequences. Experiments performed on ISCAS 85 benchmark circuits are presented and discussed in section 5. Concluding remarks are given in section 6.

2. Test pattern generator implementations

The main disadvantage of the classical testing approach is that the purchase price of test equipment (ATE) is very high /31/. For built-in self testing one must be able to generate and apply test patterns internally. By applying BIST, we can eliminate the need for ATE /1/,/2/,/13/. Figure 2 illustrates a basic BIST implementation. The method relies on embedding some extra elements to the circuits like test pattern generator and test response analyzer. In this paper we will primarily focus on the problem of built-in generation of test patterns.

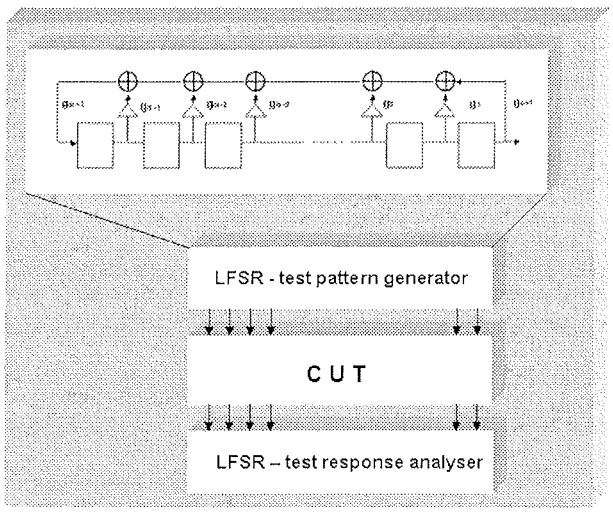


Figure 2: Basic BIST implementation to CUT.

The test sets can be stored in chip ROM but in the case of larger circuits this is an impractical and costly solution. LFSRs are commonly used as test pattern generators that generate pseudo-random patterns. These circuits are autonomous in the sense that they have no inputs except for clocks. Each cell is assumed to be a clocked D flip-flop. The LSFRs are cyclic as they go through a fixed sequence of states. The output sequences (patterns) generated by such circuits are also cyclic. The patterns could be also encoded into seeds by solving a linear system of equations, which is an algebraic representation of the linear expansion of the LFSR /15/.

The behaviour of a linear feedback register is completely determined by the feedback coefficients $g_0, g_1, .., g_m$ where g_i is a binary constant, and $g_i = 1$ implies that a connection exists, while $g_i = 0$ implies that there is no connection. A sequence of numbers $g_0, g_1, .., g_m$ can be associated with a polynomial called the generator function G(x), defined in equation (1).

$$G(x) = g_0 + g_1 x^1 + g_2 x^2 + \ldots + x^m \qquad (1)$$

Since the feedback coefficients determine the polynomial of this LFSR, they need to be set to certain combinations to realize a specific desired polynomial. The LFSR goes through a cyclic i.e. periodic sequence of states and the output sequence is also periodic. The maximum length of this period is 2^m-1, where m is the number of stages. The characteristic polynomial associated with a maximum length sequence is called a primitive polynomial. If the feedback polynomial is primitive then the output sequence has the same random properties and is called pseudo-random. In many cases pseudo-random patterns perform well for testing but may also lead to reduced fault coverage due to linear dependencies /1/.

There are two established realizations of characteristic polynomials in Figure 3. The Fibonacci implementation consists of a simple shift register in which a binary-weighted modulo-2 sum of the taps is fed back to the input. On the other hand, the Galois implementation consists of a shift register the contents of which are modified at every stage by a binary-weighted value of the output stage. The choice of LFSR generators, configuration of feedback taps and applied seeds (initial values) can make an enormous difference in the efficiency of testing and in achieving the required test coverage for a particular CUT. When implemented in hardware, modulo-2 additions are performed with XOR gates. The Galois form is generally faster than the Fibonacci one due to the reduced number of gates in the feedback loop, thus making it the favoured form /13/.

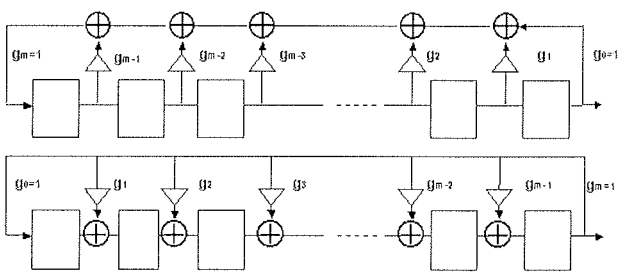


Figure 3: Fibonacci and Galois implementation of LSFR.

Figure 4 illustrates pseudo-random patterns generated with Fibonacci and Galois type of polynomial. For experimental purposes we implemented both types of generators in the software. The example shows an LFSR of size m = 4 with feedback connections at g_4 and g_3. The feedback taps are specified as lfsr "/4,3/g" for the Galois form, and lfsr "/4,3/f" for the Fibonacci form. The left side of Figure 4 shows the output sequence behavior of the Fibonacci implementation with a primitive Fibonacci field. The non primitive cyclic form in our example are obtained when the feedback taps of the polynomial are specified as lfsr "/4,2,1/f" for the Fibonacci or lfsr "/4,2,1/g" Galois form.

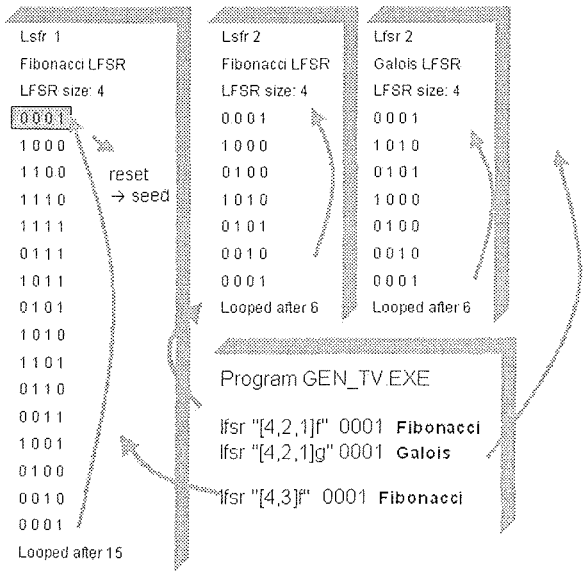
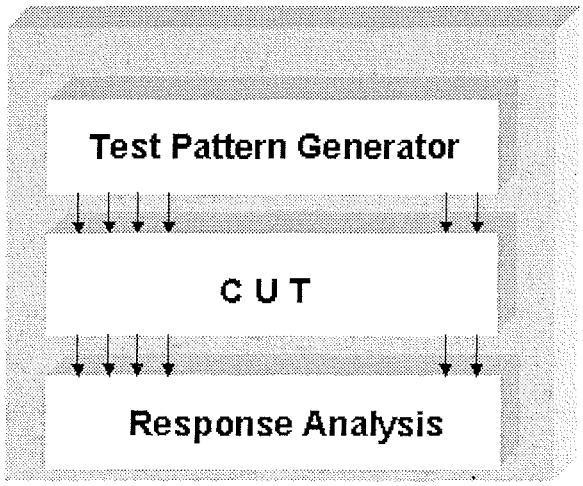


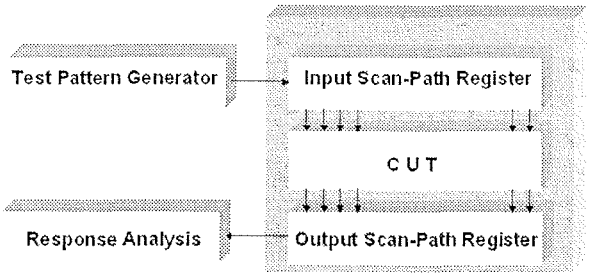
Figure 4: *Examples of primitive and non primitive sequence generation.*

BIST techniques, as shown in Figure 5, are classified into two categories: test-per-clock and test-per-scan /2/. In the test-per-clock BIST (Figure 5(a)), a test vector is applied and its response is captured every clock cycle. In the test-per-scan BIST (Figure 5(b)) a test vector is applied and its response is captured into the scan chains only after the test is scanned into the scan chains. A scan cycle is defined as the number of clock cycles required to shift the vector into a serial register of the serial scan paths (whichever is larger) plus one or more normal mode clocks. The test-per-scan approach is generally much slower than the test-per-clock approach. The two approaches involve distinct hardware structures and trade-offs.

We propose an algorithm for selecting a seed that can achieve target fault coverage with minimum test length for a given set of polynomials. Additionally, we propose an algorithm for seed selection that can achieve maximum fault coverage for a given test length and for a given set of polynomials. In other words, the number of undetected faults is reduced to a minimum. The proposed algorithm considerably improves pseudo-random testing without the need to change circuits in the CUT.



(a) Test-per-clock



(b)Test-per-scan

Figure 5: *Basic BIST techniques*

3. Preliminary details and definitions

We define the notation where F is the set of modeled faults in the CUT. In this paper we consider single stuck-at faults. Let |F| represent the number of elements in a set. Then a fault coverage of C % by a test pattern sequence means that |F| x C /100 faults are detected by this sequence. F_T denotes the set of faults that are detected by test pattern T.

Let T be an input test pattern generated from an m-stage LFSR, where m is the number of primary inputs to the CUT. When the LFSR outputs T_0 as the first test vector, then the sequence of outputs proceeds as follows: T_0, T_1, T_2, ..., T_{m-1}, T_0... Similarly, for seed T_1 the LFSR output sequence is: T_2, T_3, ..., T_{m-1}, T_0, T_1, T_2... In general, when the seed is T_i, the *j-th* test pattern is T_{i+j} (mod m).

To model the required test length we define a function $L(T_i, N_g)$ where N_g is a total number of faults detected for a selected seed T_i. N_g denotes the target number of detected faults, and $N_g \leq |F|$. Function $L(T_i, N_g)$ can be estimated or computed by fault simulations. For a given CUT there is at least one seed that detects N_g faults with a minimum test length. We search for this as the minimum test length seed $T_{i\,min}$. It is always possible to find at least one such seed.

4. The proposed approach

When performing pseudo-random testing using an LFSRs as test pattern generators, the fault coverage that can be obtained is limited by the presence of random resistant faults in the CUT. To evaluate the detection hardness of these faults, we need to use a measure that represents the difficulty level for a random pattern to detect a given fault. Computing the fault detection probability is a hard problem. Generally, the higher the number of patterns detecting a given fault, the easier this fault is detected. In our method, we measure the quality of the generated test after each test vector is applied to the CUT.

A pattern generator based on an LFSR generates a test pattern sequence after a seed has been set. We first applied the method to the test-per-clock BIST structure, where the CUT is a combinational circuit. Once an LFSR seed is selected, the succeeding output test sequence is uniquely determined. A binary vector from the *m*-stage LFSR is regarded as an element T_i ($0<i<2^m-1$) from Galois Field (GF) (2^m), where T is a primitive element from GF (2^m) and i denotes the index.

Figure 6 shows a three pillar example of the seed selection procedure. First we select one of LFSR generators with a starting seed T_i. We first consider that the length L of the generated test sequences is set. Then we start to generate a fixed number of pseudo-random test vectors. With a fault simulator /21/ we measure the success of generated test vectors after each new generated and applied test vector. The first pillar represents the result for test set with seed T_i. The bright frames in the pillar contain the *useful test vectors*. Those cover one or more faults in the CUT. The dark frames contain *unuseful* ones (no new faults were found by a specific test vector). By comparing those three pillars, we suppose a seed that generates a test cube that covers more faults then any other. Unfortunately, it is not only the number of successful test vectors in a cube that is important, but also how many faults are covered by them. Figure 7 describes this situation, where the cubes were rearranged by a fault coverage C (%). The seed T_{i+n} is the most successful selection with the highest achieved fault coverage. However, the generated test vectors for seed T_{i+m} contains more *useful* test vectors, but test vectors for seed T_{i+n} in a set are more effective.

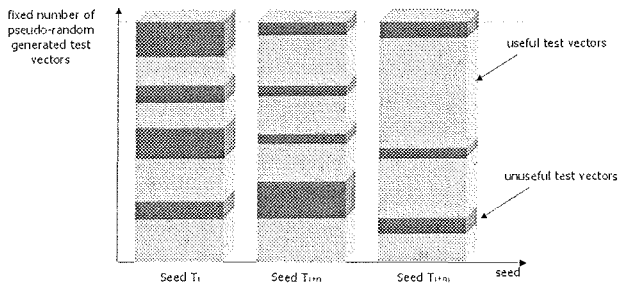


Figure 6: *Fixed generated number of test sets with different seeds.*

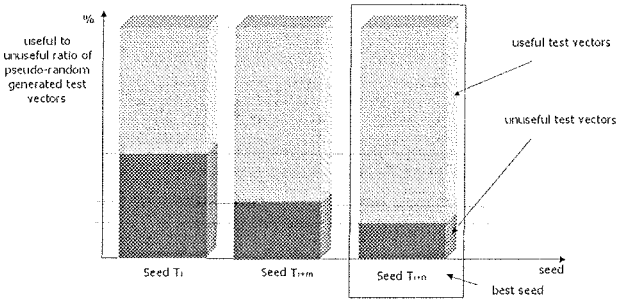


Figure 7: *Ordered pillars by effectiveness.*

Next, we assume that the test-per-clock BIST has been used in the C17 circuit, which is one of the ICSAS´85 benchmark circuits. This circuit has five primary inputs and two outputs. An example of the best seed calculation procedure is given in figure 8. For a given polynomial

Name of the circuit	c17
Number of primary inputs	5
Number of primary outputs	2
Number of gates	6
Level of the circuit	3

Polinomial	Seed	Test patter	Fault cover	Collapsed	Detected f	Undetecte	Memory	CPU
[5,1]	10000	17	100.000	22	22	0	168112	0.000
[5,1]	01000	16	100.000	22	22	0	168112	0.010
[5,1]	00100	15	100.000	22	22	0	168116	0.010
[5,1]	00010	14	100.000	22	22	0	168116	0.016
[5,1]	10001	13	100.000	22	22	0	168116	0.000
[5,1]	11000	12	100.000	22	22	0	168116	0.000
[5,1]	01100	11	100.000	22	22	0	168120	0.001
[5,1]	00110	10	100.000	22	22	0	168120	0.010

Figure 8: *The best seed calculation procedure for minimal required number of test patterns according to complete fault coverage of C17 with polynomial /5,1/.*

$P(X)=x^5+x+1$ and the starting seed T_0 (10000) the fault simulation returns 100% fault coverage of C17 after 17 generated test vectors. If we calculate the required test length to achieve the target coverage (in our case 100%), we next increment the starting seed by one and a function $L(T_i, N_g)$ returns 16 test vectors for all 22 faults. Starting with a seed 00110, we achieve the same (100%) fault coverage with shorter test length of only L=10 pseudo-random generated test vectors, which is the minimum function for the proposed polynomial and the seed $T_{i\ min}$ is 00100.

Figure 9 shows the differences in fault coverage when we implement different polynomials. We found that the implementation of specific polynomial as pseudo-random pattern generator can be very successful for some circuits while another type of polynomial returns weak results. Therefore, it is important to try more than one polynomial.

For a large circuit a seed calculation through a complete space of all possible test vectors is an NP-hard problem, therefore we have to be satisfied with approximations. It is important for the first step to make a realistic estimation of realistic test length. In the second step we can then begin with the estimation of the shortest test length $L_{min}(Ti, Ng)$. Figure 10 shows an example where the fault coverage of L=100 pseudo-random generated tests were simulated. For a given seed T_j the possible fault coverage (FC) of 88% after applied 58 pseudo-random generated test vectors was achieved. The rest (number) of test vectors are only unusable ones in window of 100 pseudo-random test vectors (PRTV). The great benefit of our technique is the possibility for exact calculation of the lower bound of those testing windows which are able to reach the target fault coverage for a specific circuit.

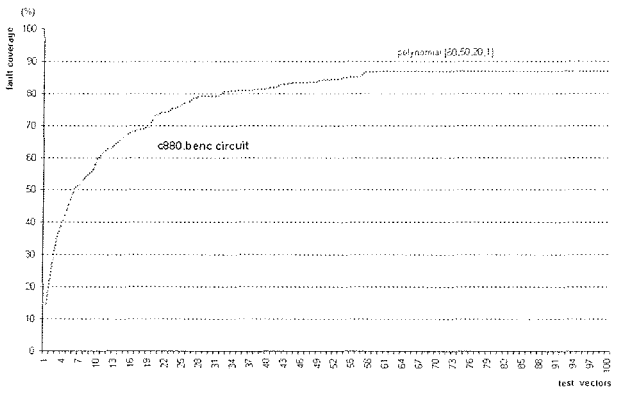


Figure 10: Coverage/number of generated test vector ratio for c880.bench

5. Experimental results

The method to reduce the pseudo-random test length described in this paper was applied to ISCAS 85 benchmark circuits. We considered the test-per-clock BIST structure. The size of each LFSR is equal to the number of primary inputs in each circuit. The characteristic polynomial of each LFSR is a primitive feedback polynomial. Fault simulation in each circuit was performed using FSIM simulation tool

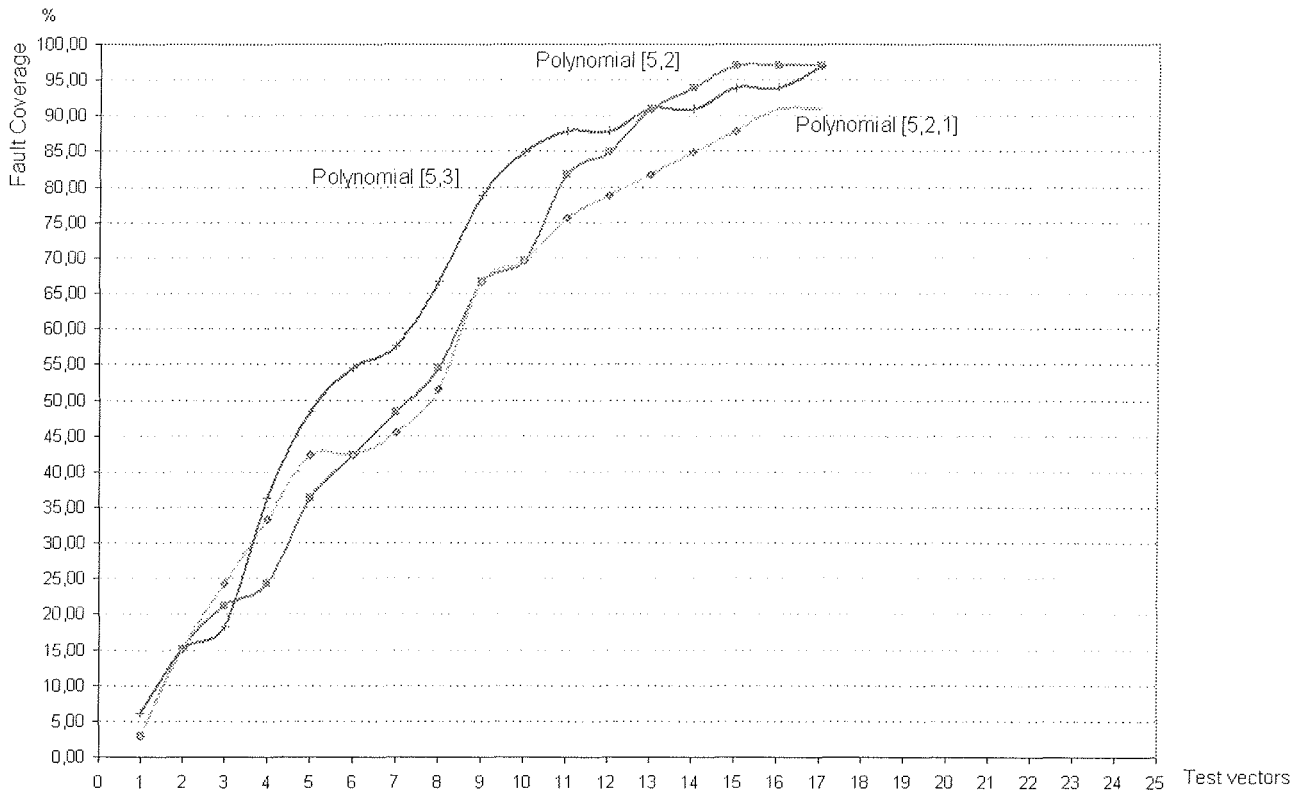


Figure 9: Polynomial discrepancy

/21/. The computer we used is PC based platform with 1 GHz Pentium Processor.

First, we select a seed that could achieve the minimum test length to cover the highest possible number of detectable faults in a particular benchmark circuit. Next, we search for suitable polynomial for a circuit. First polynomial in the database is a polynomial with the lowest possible number of feedbacks. Then we progressively increase the number of feedbacks for each additional polynomial in the base. The number of selected polynomials differs and depends on the size and complexity of a circuit. For example, we used 20 different polynomials for C880 circuit and only three polynomials for C7552 circuit. In the next step we randomly select the first seed and determine the fixed test length TSL for each test sequence. Further on, we determine the distance to the next seed and the total number of generated test sequences for each individual polynomial in database. The number of selected seeds, i.e. the number of test sequences, depends on the number of inputs in the tested circuit and ranges between a few hundred and few thousand for each polynomial. With the simulator, we calculate the achieved fault coverage for each generated test sequence, for each polynomial and seed. Next step is a determination of the average (AVR_FC %) and maximum fault coverage (M_FC %) for the specific test set. The results give us the information, which polynomial, seed and test length are the most suitable for a tested circuit.

The results are given in Table 1. The first column indicates the circuit name. The second and the third column give the number of primary inputs (# PI) and primary outputs (# PO), respectively. The fourth column gives the number of detectable faults (# F) in a circuit. The fifth column reports the chosen length of the generated test sequences (TSL). Columns six and seven give the average fault coverage (AVR_FC in %) and maximum achieved fault coverage (M_FC in %) according to the selected test length. The CPU time (in seconds) needed for the complete process of generating a one-seed LFSR test sequence is in the last column. For each circuit, an average of 15 different primitive feedback polynomials were evaluated. The results in table 1 show the performances obtained with the proposed method to compute the most successful LFSR seed and the best polynomial. By comparing our results to previously published results /23/,/10/, we conclude that for most tested circuits our results were comparable or even better. Computational effort is much more difficult to compare because of different equipment used in former reports.

In the second step of the algorithm, the goal was to find the shortest test sequence according to selected polynomial and starting seed. We used the outcome of first step of the algorithm, i.e. the best suited polynomial and the best seed. The seed is then used as initial seed in the second step of procedure. For each new generated test vector a fault coverage is computed by the fault simulator.

Table 1: Test results according to selected test lengths for ISCAS 85 circuits

Circuit	# PI	# PO	# F	TSL (10^3)	AVR_FC (%)	M_FC (%)	CPU (s)
c432	36	7	524	5	98,696	99,237	322,69
				10	98,993	99,237	5398,50
				50	99,135	99,237	29073,51
c499	41	32	758	1	98,697	98,945	519,56
				5	98,725	98,945	12702,11
				100	98,945	98,945	19800,47
C880	60	26	942	1	94,286	99,894	529,02
				5	99,5	100	1022,55
				10	99,753	100	1495,23
C1355	41	32	1574	1	96,215	99,238	1180,11
				5	97,343	99,942	3100,21
C1908	33	25	1879	1	95,268	97,765	141624,00
				10	99,487	99,521	384374,12
				50	99,519	99,521	15500,98
C2670	233	140	2747	5	84,526	88,278	191052,19
				10	84,748	90,462	358931,22
C3540	50	22	3428	1	94,278	95,362	7906,16
				5	95,829	96,004	13000,01
C5315	178	123	5350	5	69,907	98,897	6 h
				10	98,897	98,897	4008
C6288	32	32	7744	500	99,561	99,561	1008,09
				1	99,561	99,561	1579,15
C7552	207	108	7550	10	94,188	96,768	7 h
				50	96,467	96,927	13 h

Table 2: Average test results according to selected test lengths for ISCAS 85

Circuit	# PI	# PO	# F	# AVR_TL	AVR_FC (%)	MIN_TS	M_FC (%)
C17	5	2	22	11	100,00	5	100
C432	36	7	524	611	98,413	353	99,24
C499	41	32	758	793	98,820	512	98,95
C880	60	26	942	14353	98,272	1024	100,00
C1355	41	32	1574	2060	97,059	1760	99,49
C1908	33	25	1879	3502	99,071	3296	99,31
C2670	233	140	2747	> 100000	92,111	99999	95,413
C3540	50	22	3428	9949	95,920	4160	96,004
C5315	178	123	5350	512	98,061	483	98,897
C6288	32	32	7744	62	99,330	64	99,561
C7552	207	108	7550	> 100000	96,026	14432	97,007

When the desired fault coverage is achieved, the procedure ends and the last generated test vector determines the intermediate achieved test length. For each polynomial, we randomly select 1.000, 5.000, 10.000 and 100.000 test sets on average. Then, we calculate minimal test length (MIN_TS) for the highest fault coverage (FC %), average test length (# AVR_TL) and average fault coverage (AVR_FC). We found out that the majority of tested circuits the average and minimal number of test vectors are very close. For some circuit, i.e. C880, we calculate the minimal test length of 1024 generated test vectors to achieve 100% of fault coverage. To achieve the average of 98,272 % fault coverage we need 14.353 test vectors in average.

In Table 2, we compare the fault coverage provided by our method and the best fault coverage of random selected seed test sequence. The first column has the circuit name. The second and the third columns give the number of primary inputs (# PI) and the number of primary outputs (# PO). The fourth column gives the number of detectable faults (# F) in each circuit. Columns five and six give the average test length (# AVR_TL) and the average achieved fault coverage (AVR_FC in %) according to the selected set of seeds. In column seven we report the shortest achieved test length. The highest possible fault coverage achieved according to minimal test length is in the last column.

6. Conclusion

In this paper, we proposed a technique for selecting the LFSR TPG seed in order to achieve the highest fault coverage with a given set of characteristic polynomials. In our research we can conclude that different characteristic polynomials can return a different best possible seed to reach the expected fault coverage. The test length (total number of generated pseudo-random test patterns) differs more than 1 against 10 for some circuits being tested. To compute the best seed for the largest circuits in the test (C7552) we need 13 h of computational time. Note that for large circuits the applicability of the method depends on two user constraints. The first one is the computational effort one is willing to spend. The second one is the test length of the test sequence one is willing to allow. Comparing to the results of other authors we can conclude that our technique is fast enough to deal with combinational circuits of great size and with a large number of primary inputs. Furthermore, the proposed seed/polynomial selection can be applied not only as conventional BIST but also in mixed mode BIST such as those with reseeding, mapping, and bit-fixing.

7. References

/1/ M. Abramovici, M. A. Breuer, A. D. Friedman, "Digital Systems Testing and Testable Design", New York: Computer Science, 1990.

/2/ V. D. Agrawal, C. R. Kime, K. K. Saluja, "A Tutorial on Built-in Self-Test, Part 1: Principles ", IEEE Design&Test, pp. 69-77, 1993.

/3/ V. D. Agrawal, C. R. Kime, K. K. Saluja, "A Tutorial on Built-in Self-Test, Part 2: Applications", Design & Test of Computers, pp. 73-82, 1993.

/4/ S. B. Akers, W. Jansz, "Test Set Embedding in Built-in Self-Test Environment", IEEE 1989 International Test Conference, pp. 257-263, 1989.

/5/ I. Bayraktaroglu, K. Udawatta, A. Orayloglu, "An Examination of PRPG Selection Approaches for Large, Industrial Design", Proc. Asia Test Symp., pp. 440-444, 1998.

/6/ I. Bayraktaroglu, A. Orailoglu, "Selecting a PRPG: Randomness, Primitiveness, or Sheer Luck", 10th Asian Test Symposium (ATS'01), 2001.

/7/ F. Brglez, H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN", IEEE International Symposium on Circuits and Systems, 1985.

/8/ K. Chakrabarty, S. R. Das, "Test-Set Embedding Based on Width Compression for Mixed Mode BIST", IEEE Transactions on Instrumentation and Measurement, vol. 49, no. 3, pp. 671-678, 2000.

/9/ B. Dugonik, T. Kapus, Z. Brezocnik, "Design error diagnosis and test in logic circuits using error simulation models", V: HAMZA, M.H. (ur.). IASTED International conference Software Engineering, Anaheim; San Francisco, Calgary; Zürich: IASTED/Acta Press, Software engineering: proceedings of the IASTED International conference, pp. 154-158, 1997.

/10/ C. Fagot, O. Gascuel, P. Girard and C. Landrault, "On Calculating Efficient LFSR Seeds for Built-in Self Test", Proc. European Test Conf., pp . 4-14. 1999.

/11/ S. Hellebrand, H. G. Liang, H. J. Wunderlich, "A mixed-mode BIST scheme based on reseeding of folding counters", Proc. Int. Test Conf., pp. 778-784, 2000.

/12/ S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," Proc. of IEEE Int'l Test Conf., pp.120-129, 1992.

/13/ N. K. Jha, S. Gupta, "Testing of Digital Systems", Cambridge University Press, 2003.

/14/ K. Ichino, K. Watanabe, M. Arai, S. Fukumoto, K. Iwasaki, "A Seed Selection Procedure for LFSR-Based Random Pattern Generators", Proceedings of the ASP-DAC 2003. Asia and South Pacific Design Automation Conference 2003, pp. 869-74, 2003.

/15/ B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs," Proc. of European Test Conference, pp. 237-242, 1991.

/16/ M. Lempel, S. K. Gupta, and M. A. Breuer, "Test Embedding with Discrete Logarithms," IEEE Transactions on Comput.-Aided Design, Vol. 14, pp. 554-566, 1995.

/17/ R. Meolic, T. Kapus, B. Dugonik , Z. Brezocnik, "Formal verification of distributed mutual-exclusion circuits", Inf. MIDEM, nr. 3(107), pp. 157-169, 2003.

/18/ B. Murray, J. Hayes, "Testing ICs: Getting to the Core of the Problem", IEEE Computer, vol. 29, no. 11, pp. 32-38, 1996.

/19/ J. Savir, W. McAnney, "A Multiple Seed Linear Feedback Shift Register", IEEE transactions on Computers, vol. 41, no. 2, pp. 250-252, 1992.

/20/ A. Steininger, "Testing and Built-in Self Test - A Survey", Journal of Systems Architecture, Elsevier Science Publishers B.V., North Holland, 2003.

/21/ H. K. Lee and D. S. Ha, "An Efficient Forward Fault Simulation Algorithm Based on the Parallel Pattern Single Fault Propagation", Proc. of the 1991 International Test Conference, pp. 946-955, 1991.

/22/ N. A. Touba, E. J. McCluskey, "Synthesis of Mapping Logic for Generating Transformed Pseudo-Random Patterns for BIST," Proc. of International Test Conference, pp. 674-682, 1995.

/23/ N. A. Touba, E. J. McCluskey, "Transformed Pseudo-Random Patterns for BIST," Proc. of International Test Conference, pp. 674-682, 1994.

/24/ N. A. Touba, E. J. McCluskey, "Bit-Fixing in Pseudo-random Sequences for Scan BIST", IEEE Transaction on Computer-Aided Design of Integrated Circuit and Systems, Vol 20, pp. 71-82, 2001.

/25/ N. A. Touba, , E. J. McCluskey, "Test Point Insertion Based on Path Tracing", Proc. of VLSI Test Symposium, pp. 2-8, 1996.

/26/ S. Venkataraman, J. Rajski, S. Hellebrand, and S. Tarnick, "An Efficient BIST Scheme Based On Reseeding of Multiple Polynomial Linear Feedback Shift Registers," Proc. of IEEE Int'l Conf. on Computer-Aided Design, pp. 572-577, 1993.

/27/ R. Williams, "IBM Perspectives on the Electrical Design Automation Industry", Keywords to IEEE Design Automation Conference, 1986.

/28/ H. J. Wunderlich, "Multiple Distributions for Biased Random Test Patterns", IEEE Transactions on Computer-Aided Design, Vol. 9, No. 6, pp. 584-593, 1990.

/29/ H. J. Wunderlich, G. Kiefer, "Bit-Flipping BIST", ACM/IEEE International Conference on CAD-96 (ICCAD96), San Jose, California, pp. 337-343, 1996.

/30/ H. J. Wunderlich,"BIST of Systems-on-a Chip", The VLSI journal, pp. 55-78, 1998.

/31/ H. J. Wunderlich, "Test and Testable Design", Springer Verlag, pp. 141 - 190, 1998.

Bogdan Dugonik, Zmago Brezočnik
Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, Slovenija
Smetanova 17, 2000 Maribor

Prispelo (Arrived): 01.09.2004 *Sprejeto (Accepted): 15.09.2004*