LFSR generation for high test coverage and low hardware overhead

Leonel Hernández Martínez[1*], Saqib Khursheed[1], Sudhakar M. Reddy[2]

1 Dept. of Electrical Engineering & Electronics, University of Liverpool, UK.
2 College of Engineering, University of Iowa, USA.
* A.L.Hernandez-Martinez@liverpool.ac.uk

Abstract – Safety critical technology rests on optimized and effective testing techniques for every embedded system involved in the equipment. Pattern generator (PG) like linear feedback shift register (LFSR) are used for fault detection and useful for reliability and online test. This paper presents an analysis on the LFSR, using a known automatic test pattern generator (ATPG) test set. Two techniques are undertaken to target difficult-to-detect faults with their respective trade-off analysis. This is achieved using Berlekamp-Massey (BM) algorithm with optimizations to reduce area overhead. The first technique (Concatenated) combines all test sets generating a single polynomial that covers complete ATPG set (baseline-C). Improvements are found in Algorithm 1 reducing polynomial size through Xs assignment. The second technique uses non-concatenated test sets and provides a group of LFSRs using BM without including any optimization (baseline-N). This algorithm is further optimised by selecting full mapping and independent polynomial expressions. Results are generated using 32 benchmarks and 65 nm technology. The concatenated technique provides reductions on area overhead for 90.6% cases with a best case of 57% and 39% mean. The remaining 9.4% of cases, non-concatenated technique provides the best reduction of 37% with 1.4% mean, whilst achieving 100% test mapping in both cases.

1. Introduction

The statistics on automotive accidents show that on average, more than 6 million crashes are reported every year [1], which makes automobiles a major concern for road accidents. The evaluation of potential scenarios of hazards and dangers are defined by the standard ISO26262, providing safety process guidelines for each automotive component.

The safety classification is specified under the mentioned standard as Automotive Safety Integrity Level (ASIL) and categorizes safety requirements through various parameters e.g. severity, exposure and controllability. The functional safety of each automotive component has the goal to avoid any threats to the vehicle. The categorization of critical safety levels are established by the letters A, B, C and D, where D represents the highest safety level (Fig. 1). The demanding utilization of embedded systems in safety critical environment lead to a requirement of higher level of fault coverage for these systems. It is desirable to cover faults that are classified as hard to detect and are of high importance.

This research work has the goal to overcome these challenges for BIST technology [2], [3] , which consists of the integration of a test pattern generator (TPG) to perform internal testing, without requiring the presence of an external tester [4].

To achieve high reliability of electronic systems devices, they have to be tested when they are manufactured as well as in the field. Typically, test patterns are applied either by an external tester or in Built-in-test (BIST) mode by circuits internal to a device under test. For field test it is necessary to apply tests using BIST. The test patterns applied in BIST can be pseudo-random patterns generated typically by an LFSR or can be patterns which were generated by a deterministic test pattern generator. In practice, one may use a hybrid approach. The deterministic patterns will either have to be stored or generated by expanding seeds fed to an LFSR. The latter approach is popular in modern devices and is the focus of our work. In the sequel we first discuss the need for BIST in embedded systems followed by details of our proposed method to generate deterministic tests on chip using LFSRs. We demonstrate that concatenating all tests to be generated one can use a single LFSR to generate all tests leading to savings in the number of stages of LFSR compared to using one LFSR and a seed to generate each test.

The adoption level of electronics in the automotive industry has increased along with requirements to assure reliability of the embedded electronics. Nowadays the Advanced Driver Assistance systems (ADAS) is becoming a reality and the electronic systems has the ultimate control of the critical system [5].

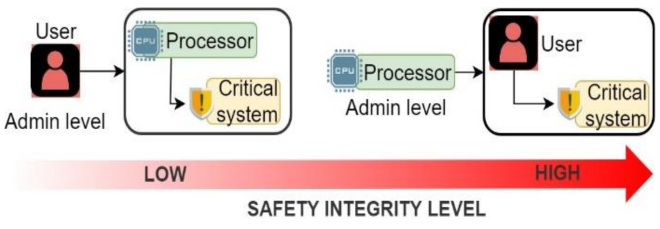


Fig. 1. *ASIL diagram representation of safety categories for electronic embedded systems, from ASILA (left) to ASILD (right) [1].*

The evolution of these automated technologies increase the complexity of each part of the system involving aid components, e.g. front facing cameras. In addition, existing safety critical systems in automotive, such as ABS (anti-lock brake systems) demands enhancements to match the pace of recent developments. These type of systems are expensive but necessary to guarantee the highest reliability levels required by automotive industry standards (ASIL-D).

Multiples techniques have been developed for this purpose, for instance the linear feedback shift register (LFSR) [6] has been used in a variety of forms e.g. as part of a fault tolerant system in space environment, where the authors provide insightful information about systems design for fault avoidance in a space environment [7]. In addition, LFSR has been applied for pseudorandom binary sequence generator, where the generation of pseudorandom-numbers with a Gaussian distribution [8] is taken as the starting point for the PG implementation. Reseeding for higher coverage [9]-[11] is another approach, where the update of the initial value of the registers in LFSR generates more specific bit sequences for fault detection, this technique has also being utilized along with encoding of patterns for the optimization of switching activities when unknown values are introduced in the test pattern [12], [13].

In continuation of PGs on chip, Smax+20, is a popular approach [14] based on reduction of the probability of linear dependencies among neighbour bits of the LFSR sequence. In recent methods the Berlekamp-Massey (BM) algorithm [15], [16] is applied as the mathematical method for accurate calculation of LFSR and the use of recurrence relation model for bit generation [17]. The use of BM provides a set of seeds for a test set, giving one seed per test pattern. The utilization of seeds, results in occupation of memory as big as the size of the LFSR multiplied by the number of test patterns. Along with improvements in fault coverage, approaches for suitable use in automotive applications has been proposed, performing various testing techniques through Stellar BIST [18], as well as further reduction of test pattern can be achieved by compressing and decompressing bit patterns [19], [20].

This paper addresses the aforementioned challenges on electronic embedded systems for safety critical domains. Improvements on the PG calculation methodology is provided, as well as hardware overhead optimization. A novel technique on the pattern reorganization for LFSR is proposed, utilizing ATPG patterns as input that makes it suitable for set-reset flip-flop (FF) implementation, putting aside the utilization of memory for seed storage. In addition, this provides a full mapping of the target patterns. This novel technique of test pattern reorganization requires just an initial binary value for the shift register. This is based on the principle that test sets can be generated following one another in uninterrupted succession, constructing a concatenated bit pattern with the length resulting from the size of number of patterns (*m*) multiplied by the number of bits (*n*).
Unlike the existing techniques, this technique utilizes only the set of an initial binary value for the full mapping of ATPG test set. Furthermore, the test bit mapping process is optimized due to avoidance of reseeding, being this is unnecessary in the concatenation approach of all the test patterns in one, this provides benefits for the speed and power due to the relation with mapping process. In consequence, the proposed technique generates ATPG test patterns in a sequential manner, optimizes the total area overhead and maintains full test coverage.

The rest of this paper is organized as follows. In Section 2, the proposed technique of concatenated ATPG set for LFSR is described along with results of LFSR size. Section 3 presents the calculation for non-concatenated patterns, improving the polynomial set required for the total mapping of the ATPG patterns. A trade-off analysis is presented in section 4, providing results on the area overhead from both methods, concatenated and non-concatenated test set for LFSR. Conclusions are drawn in section 5. In Appendix 1 we give a detailed example to illustrate the two algorithms studied in Sections 2 and 3.

2. Concatenated Technique

The integration of effective PG for embedded electronic systems has been explored from various approaches. The evolution of the external test set generator was crucial and established the starting point for on-chip pattern generators. The literature shows techniques with the goal to assure higher test coverage, utilizing test point insertion [5], [21], [22] and reseeding techniques for the LFSR [9], [10], as well as a combination of both. The strategy of applying a number of test vectors as inputs and output analysis has increased the efficiency of fault detection. The fault coverage is defined as the number of detected faults over total detectable faults.
The proposed technique prioritizes high fault coverage through a novel approach removing storage of patterns and improves the calculation process of LFSR.

2.1. Proposed Technique

As previously mentioned, many approaches utilize a test set organization where patterns remain separate and PG generates them individually [14], [16], [23]. These techniques have been developed in order to improve test coverage. On the other hand, these techniques make use of a set of seeds equal to the number of patterns, which keeps the utilization of memory for the storage of seed. In the proposed technique, the test set is reorganized into only one pattern for the calculation of LFSR. This is achieved by concatenating each test vector with the immediate next sequential pattern of the set, which generates a concatenated pattern of size *m* x *n*, where *m* is the number of patterns and *n* is the bit size of each pattern (Fig. 2).

The concatenated patterns is taken as the new goal and the next step is the calculation of LFSR that will map the full bit pattern. In the procedure of calculating the LFSR, this is represented as polynomial, therefore this proposition proceeds by finding the polynomial representation and then use the mathematical expression to generate the test pattern.

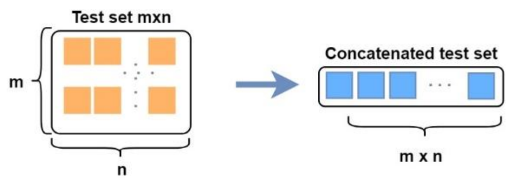


Fig. 2. *Proposed reorganization of test set.*

Since the patterns are Galois field (GF2) [24] elements, the coefficients of the polynomial are also binary numbers. In literature, a variety of approaches for calculation of polynomial can be found; many techniques take the way of trial-and-error [25], which is a very long way to find a suitable, small and full generator polynomial. In many techniques, the polynomial number of coefficients are as many or more than the number of bits present in the pattern [9], [10], [14]. In recent studies, BM algorithm [15], [16] adapted to binary field, has become a reliable calculation technique for LFSR. Many studies utilize relaxed test set in combination with BM [12], delivering polynomial representation of LFSR smaller than the test pattern bit size.

The proposed technique forms one concatenated test pattern for the input of BM algorithm and this outputs the polynomial expression that fully maps the concatenated set. The bit generation by LFSR is based on the recurrence relation model [17], where each consecutive bit results from a linear combination between previous patterns and polynomial coefficients.

$$b_j = \sum_{i=1}^{d} c_i \cdot b_{j-i} \qquad (1)$$

The linear combination (see (1)), shows that the next bit (*bj*) is the summation of each polynomial coefficient (*ci*) multiplied by the previous bits (*bj-i*). In order to produce all patterns, this expression is applied in a window-moving manner, where every time the last bit is calculated, the previous group of bits (*bj-i*) is shifted left and the last position is updated with the value of the last calculated bit (*bj*) (Fig. 3).

The linear combination among coefficients of the polynomial representation of LFSR and each bit position of the target pattern is utilized by BM algorithm for the assignment of unknown bits from relaxed test patterns [23]. Each unknown value is assigned based on the calculation of the discrepancy, which is the linear combination between the polynomial coefficients and *d* elements of the pattern (see (1)). The BM algorithm includes a decision block when *bj* is unknown (Fig. 4), assigns *bj* with the result of the immediately calculated linear combination between the previous bits and the current polynomial coefficients. In this manner the polynomial degree is updated according to the discrepancy value through the entire pattern of bits. This process of polynomial calculation guarantees a sequential and consecutive mapping of the ATPG set using LFSR.

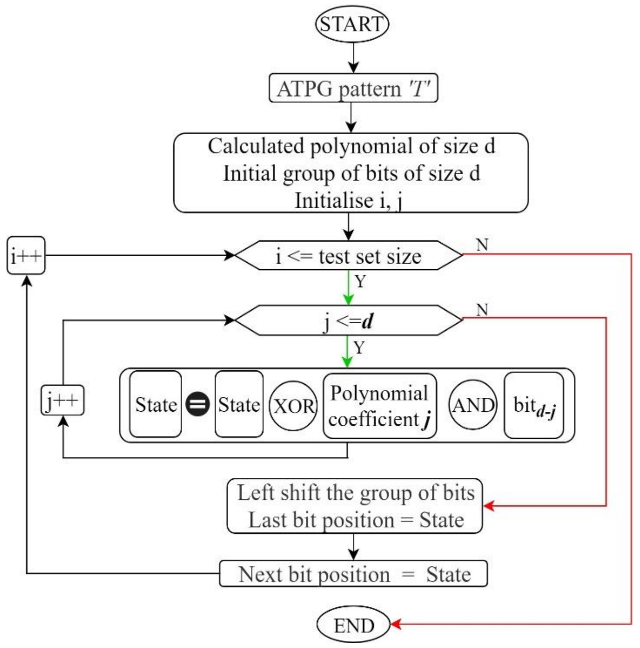

START
ATPG pattern 'T'
Calculated polynomial of size d
Initial group of bits of size d
Initialise i, j
i++
i <= test set size
N
Y
j <=d
N
Y
j++
State
State
XOR
Polynomial coefficient j
AND
bit d-j
Left shift the group of bits
Last bit position = State
Next bit position = State
END

Fig. 3. *Recurrence relation flow chart for pattern mapping, given the polynomial and initial group of bits [7], [17].*

The use of concatenated patterns and BM algorithm is addressed in this proposed technique. Results on the calculations are provided for fully specified and relaxed test sets.

2.1.1. LFSR calculation

The proposed concatenated technique utilizes concatenated ATPG set as the input of BM algorithm (Fig. 4) for the calculation of LFSR's polynomial expression. A set of 29 benchmarks are utilized [26].

As established by the analysis on the of the recurrence relation (Fig. 3), the polynomial expression guarantees the full generation of the pattern. In an initial stage the LFSR calculation is performed using fully specified bits.

In a second step, the concatenated ATPG sets are implemented with relaxed bit positions. Each benchmark has a different number of unknown values (Fig. 5). The test patterns are the input of BM, which assigns each unknown position according to the sequence shown in Fig. 4.

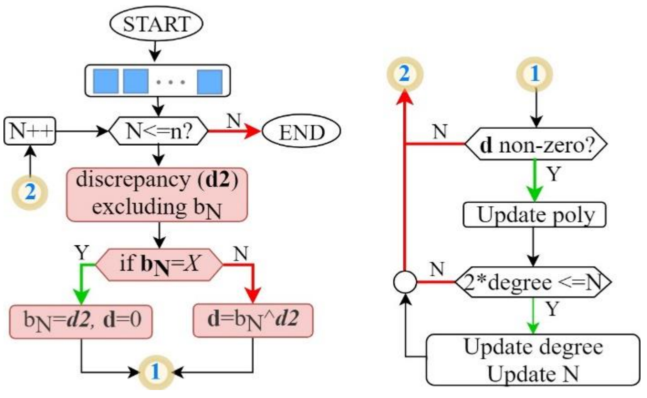


Fig. 4. *BM for the X values assignment [23].*

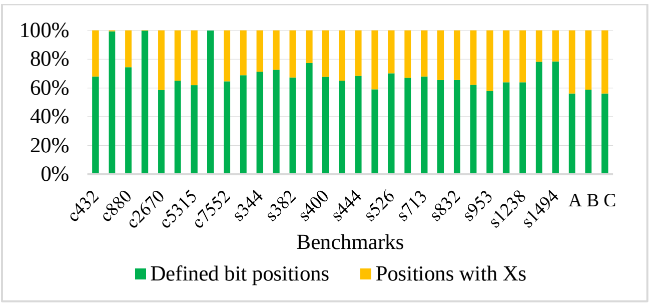


Fig. 5. *Concatenated test sets with percentage of unknown values [26].*

The stages involved in the calculation of LFSR are revised, in order to improve the result. The first stage is the initial calculation of polynomial and unknown assignment based only on BM algorithm. Furthermore, optimization on the calculation is performed by assigning the unknown bits previous to the LFSR's polynomial calculation. The polynomial degree is expected to vary slightly from the results with fully specified bits. The analysis provided in the following section delivers insightful information about the behaviour of the polynomial degree during the calculation process, the values found, vary only in small magnitude with respect to baseline numbers where the input of the algorithm was fully specified bit pattern. This small reduction is linked as well to the mathematical structure of the BM algorithm [15].

2.1.2. Polynomial degree optimization with relaxed test patterns.

The results obtained in fully assigned bits for all the benchmarks can be optimized when unknown positions are introduced. This further reduction can be found if previous assignment of unknown values is performed before the use of BM algorithm. This has a positive impact on the area overhead of LFSR and facilitates its application for microprocessors in safety critical environment. This further reduction on the polynomial expression of LFSR is achieved using a sequential bit assignment method [23]. Each assignment of one bit at a time, runs BM algorithm and check if the polynomial expression is smaller than calculated before. The starting pattern is the one calculated using only BM without previous assignment.

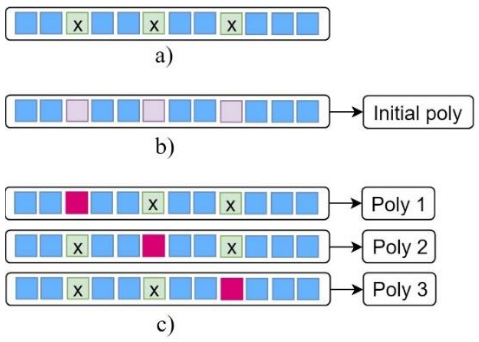


Fig. 6. *Sequential diagram of flipping technique, a) Original bit pattern, b) Baseline assignment of Xs using only on BM along with the baseline polynomial and c) Flipping of one position at a time of the initially assigned and re-calculation of the polynomial [23].*

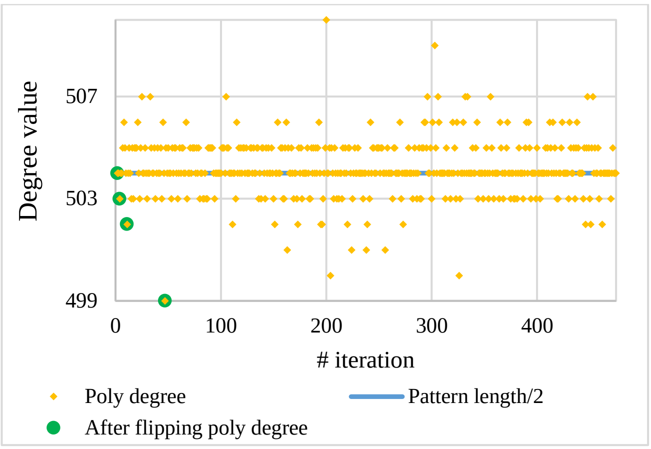


Fig. 7. *Exploration of minimal polynomial degree for benchmark c432 [26] with unknown values using flipping technique.*

As show in Fig. 6, step *b)* is used as the reference for the flipping and in each time all the other bits goes back to unknown values and a re-calculation of the polynomial is performed. During each flipping, the minimal polynomial degree is updated and preserved.

A small reduction is found due to the mathematical composition of BM algorithm where the limit value of the polynomial degree is half of the pattern. Due to this process of polynomial calculation, the value of the degree does not move far from the half of the total bit size of the pattern. An example on this polynomial degree behaviour is shown in Fig. 7, where the values remain close to half the pattern's length.

Algorithm 1 Proposed polynomial calculation for concatenated ATPG set

Input: ATPG set *'T'*
Output: 1 polynomial for LFSR, its seed and the assigned unknown values of the concatenated ATPG set '*Tc*'

```
%ATPG concatenation
1.  While i <= T size do
2.  Tc=concatenate(Tc,Ti)
3.  End While
% Polynomial calculation of concatenated set
4.  Baseline polynomial calculation with BM
5.  If Xs number in Tc > 0
6.  While i <= Tc size do
7.  If bit position is X
8.  Flip bit position
9.  If polynomial size < previously found
10. Update minimum polynomial
11. End if
12. End if
13. End while
14. End if
15. Return minimal polynomial and final bit
    assignment
```

Fig. 8. *Algorithm 1 Polynomial calculation for concatenated ATPG set*

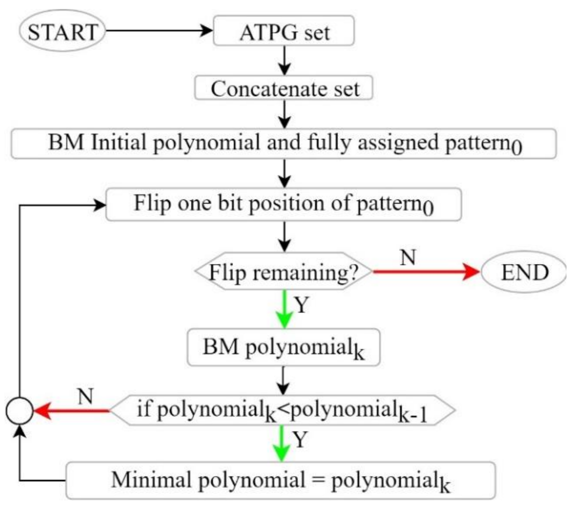

START
ATPG set
Concatenate set
BM Initial polynomial and fully assigned pattern0
Flip one bit position of pattern0
Flip remaining?
N
END
Y
BM polynomialk
N
if polynomialk<polynomialk-1
Y
Minimal polynomial = polynomialk

Fig. 9. *Top level diagram of the steps involved in the concatenated technique.*

The behaviour of the polynomial degree shown previously is verified for all benchmarks. Improvements were found in all cases with a maximum reduction found in benchmarks s349 with a polynomial 3.2% smaller after optimization through different assignment of Xs bit positions.

The exploration in the assignments of unknown values present in the test patterns has been improved slightly and ***Table 1*** (see Appendix 1) shows sizes of LFSR obtained using Algorithm 1 (Fig. 8) for fully specified bits and for when Xs are present in each benchmark. The latter is shown after being optimized with the flipping technique for concatenated ATPG test set.
The optimized way for the implementation of this LFSR is by using set-reset flip-flops, avoiding any memory for seed storage. As summary, the concatenated ATPG technique for LFSR calculation is shown in Fig. 9. This presents the sequence followed in this proposed technique for its implementation.

2.1.3. *Big O complexity analysis on time and space for Algorithm 1.*

The O complexity of Algorithm 1 is depicted as follows, considering each of the steps involved. The concatenation of the patterns has a linear and direct proportion to the size of the test pattern, this has a maximum iteration of O(T). Where T is the size of the test pattern. Berlekamp-Massey (BM) algorithm complexity has a quadratic relation with the length of the pattern $O(T^2)$, as mentioned in [16], [27].
Algorithm 1, performs additional reduction on the calculated LFSR and this is achieved by flipping segments of the full test pattern. The number of segments to flip are selected a priori, based on a maximum calculation time established as a constant. The complexity contribution of this task is the product of number of unknown positions to flip ($X_{to\text{-}flip}$) by the number of segments ($n_{segment}$), thus the complexity result is $O(X_{to\text{-}flip}*n_{segment})$. The overall complexity is $O(T + X_{to\text{-}flip}*n_{segment} * T^2)$.

Based on the time complexity, the space complexity is determined as follows. For each of the benchmarks utilized, the ATPG test pattern varies and the first step is to load the whole test set in memory to perform the calculation of the corresponding LFSR. Therefore this step contributes with a space complexity of O(T), where T is the size of the concatenated pattern. The contribution of BM algorithm in the space complexity is defined by O(4T).
The additional optimization on the calculated LFSR can be considered as an auxiliary space, which is taken each time a new segment of the pattern is flipped. Therefore the contribution of this step is O(T) for the new pattern and an additional calculation of BM multiplied by a factor of the final polynomial (p_{concat}). The overall space complexity of Algorithm 1 is $O(6T+ p_{concat}*4T)$.

The computer utilized to perform the tests for both algorithms has an intel core i3 processor with a speed of 3.30 GHz and RAM of 8 GB and 500 GB of storage.

3. Non-Concatenated Patterns and Polynomial Calculation

Other techniques for the calculation of LFSR uses non-concatenated test sets [16]. In this section, an analysis of this process is provided, showing the steps involved, along with reduction techniques utilized on the polynomial set size. In the beginning, a polynomial set is calculated according to the individual mapping, where corresponding patterns are removed every time a polynomial is found. This delivers a progressive reduction on the test set, which shrinks the target set for next polynomial calculation.

Due to the initial exclusion of patterns, a further exploration on the baseline selection of non-concatenated test set is performed. The progressive selection of the initial set of expressions, provides each polynomial with ATPG mapping subset, excluded in the consecutive next search, this sequence delivers each subset in an isolated manner. The optimization step exploits this fact and re-checks the coverage of polynomials considering the full test set, providing a more precise selection of polynomial expressions.

In the bar graph of Fig. 11, the calculation of maximum and independent mapping polynomials for benchmark c432 is shown. Fig. 11 shows the total number of polynomials that remain after all the iterations, each previously selected polynomial expression, reveals the total mapping capacity when using as goal the full test set, and in following checks the remaining set of patterns are tested against the reduced set of polynomials. The method also shows that in each step, the mapping of some polynomials are included by others, this excludes some polynomials since its mapping subset is already included by another. Therefore, the polynomials are optimized according to the mapping sets. In Fig. 11 the maximum mapping of a polynomial is 29% of the test set.

```
Algorithm 2 Polynomial calculation for non-concatenated ATPG set
Input: ATPG set 'T'
Output: polynomial set for LFSR and its seeds

%initial polynomial calculation (baseline)
1.  While T size > 0 do
2.  min_map=10%;
3.  select randomly a pattern from T
4.  Initial polynomial calculation with BM for selected pattern
5.  Generate polynomial set based on initial polynomial
6.  While polynomial set size > 0 do
7.  Select one polynomial from set
8.  if polynomial mapping>=10%
9.  Save polynomial, seeds and mapping set
10. Remove polynomial and the correspondent mapping set
11. end if
12. end while
13. end while
% optimization of polynomial set
14. Restore original T
15. While T size>0 do
16. Calculate the maximum mapping polynomial
17. Save the maximum mapping polynomial and its seeds
18. Remove the polynomial from set and its mapping sub set
19. end while
20. Return the final group of polynomial for LFSR and its seeds
```

Fig. 10. *Algorithm for optimum number of polynomial using non-concatenated ATPG set.*

In Algorithm 2 (Fig. 10) the description of each step for the polynomial calculation for non-concatenated ATPG set is provided. The calculation sequence of LFSR for non-concatenated ATPG set is shown in Fig. 13. This sequential diagram shows the optimization of polynomial set according to its maximum and independent mapping capacity. Fig. 12, shows improvement on the polynomial set for c432 benchmark, where the total reduction on the polynomial set size is 33%. For each of the benchmarks presented in Table 2 (see Appendix 1), Algorithm 2 achieves reductions on all polynomial sets, and the results can be seen in table 2, where improvements are found in all cases with a mean of 27% reduction on the polynomial number.

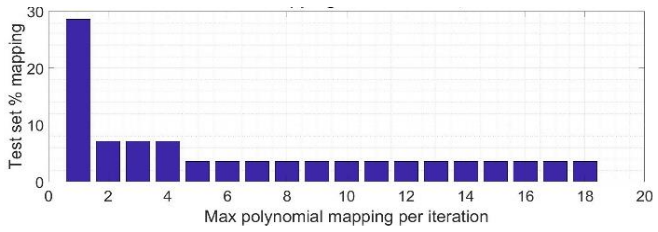


Fig. 11. *Individual maximum mapping for full test set, c432 benchmark.*

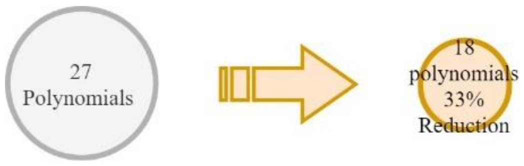
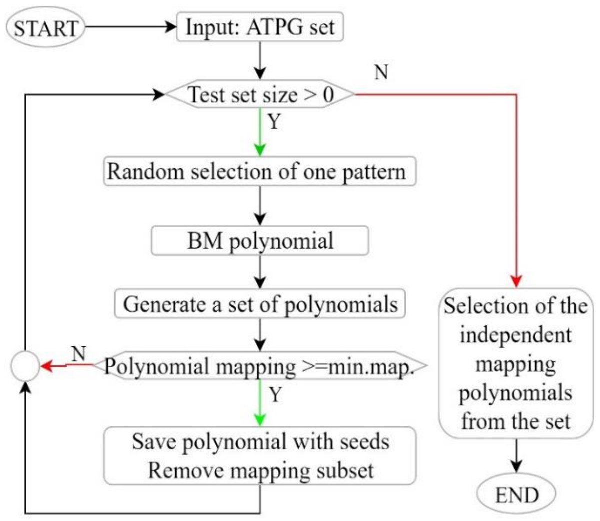


Fig. 12. *Total polynomial set size reduction of Benchmark c432.*



START
Input: ATPG set
Test set size > 0
N
Y
Random selection of one pattern
BM polynomial
Generate a set of polynomials
Polynomial mapping >=min.map.
N
Y
Save polynomial with seeds
Remove mapping subset
Selection of the independent mapping polynomials from the set
END

Fig. 13. *Top level diagram of the steps involved in the non-concatenated technique.*

As show in Table 2 (see Appendix 1) a reduction in most of polynomial sets of each benchmarks was achieved. The highest reduction in the polynomial set can be seen in benchmark s510 with 78% of reduction. More high quality results were also found in benchmarks: s386 with 59%, s820 with 50% and s832 with 66% reduction.

3.1. Big O complexity analysis on time and memory for Algorithm 2.

For Algorithm 2, the generation of the pool of polynomials is undertaken through Sidorenko-Borset algorithm that considers the uniqueness of LFSRs and utilizes BM algorithm to generate a pool based on a chosen pattern ($t_{pattern}$). This section of the algorithm contributes with a time complexity of $O(t_{pattern}^2)$ [27]. The pool of polynomials are tested with all the remaining patterns of the original test set and has a maximum complexity of O(T). This previous tasks are undertaken as long as test patterns remains in the test set, which contributes with a factor of T for both previous tasks. As a result, this section of the algorithm has a complexity of $O(T*(t_{pattern}^2+T))$. In the last section, Algorithm 2 performs a selection of the highest generator polynomials ($P_{generator}$) performing the test for the whole set of patterns, this contributes with a complexity of $O(P_{generator}*T)$. Overall the complexity is $O(T*(t_{pattern}^2+T)+P_{generator}*T)$.
For the space complexity of Algorithm 2, the initial load of whole ATPG test patterns contribute with O(T). The generation of the pool can be considered as auxiliary space and contributes with O (4T) multiplied by a pool factor (f_{pool}). The selection of highest polynomial generator can contribute as much as many patterns in the test set, therefore the complexity is O(T). The final polynomial set contribute with a factor of $p_{non-concat}$. Overall the space complexity is $O(T+ f_{pool}*4T+p_{non-concat}* T)$.

4. Trade-Off Analysis

In this section, results are provided on the technique for concatenated and non-concatenated ATPG set for LFSR (see Appendix 1, Table 3). Each technique is formed by the necessary logic to perform the mapping of bit patterns. In the case of concatenated patterns, only the LFSR is required while in the non-concatenated technique the LFSR and all seeds per benchmark are necessary (Fig. 14). For the following comparison, the results are provided separately in order to show the number of polynomials in one graph (Fig. 15 (c)) and the comparison of total area overhead in the other graph (Fig. 16 (c)).

The results on polynomial trade-off analysis compares the total size of each technique, the concatenated (Fig. 15 (a)) requires only one polynomial and this is the value plotted, however for the non-concatenated method (Fig. 15 (b)), a group of polynomials are required, the summation of these polynomials size is plotted. The results show that in 56% cases the concatenated method improves the size of the polynomial, with a best case reduction of 39% and with a mean of 12%. For the remaining 44% cases, non-concatenated technique provides a best case of 93% reduction on the polynomial set, with a mean of 23%.

As an example on results of each algorithm, benchmark s349 can be seen in Appendix 1, Table 4 and 5. Both of the ATPG test patterns have the same unknown positions (X) and the patterns shown in these results represent the final bit assignment chosen by each algorithm in the optimization process by assigning a binary value (1 or 0) for LFSR reduction. Table 4 shows the non-concatenated technique results, where a set of 13 seeds has been calculated, this is due to the fact that s349 circuit has an ATPG test set of 13 patterns with a size of 24 bits each. Due to the uniqueness of the test patterns, the expected number of seeds is the same as number of patterns in most of the cases with a reduction found in the polynomial number. In this case a reduction of one unit can be seen in the set of polynomials, this happen because polynomial 7 can generate two test patterns and for this generation it utilizes two seeds. Table 5 shows the concatenated technique results with only one seed and the corresponding LFSR. This shows that the concatenated technique delivers an LFSR which is 26% smaller than the set of LFSRs together, delivered by non-concatenated technique (Table 4). Note that in our calculations, seeds (Table 5) are stored in LFSRs using set-reset flip-flops for further area saving. Whereas non-concatenated technique stores all seeds in memory as in literature.

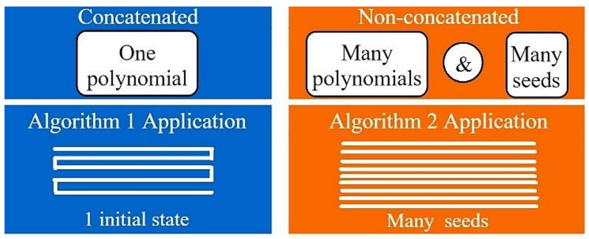
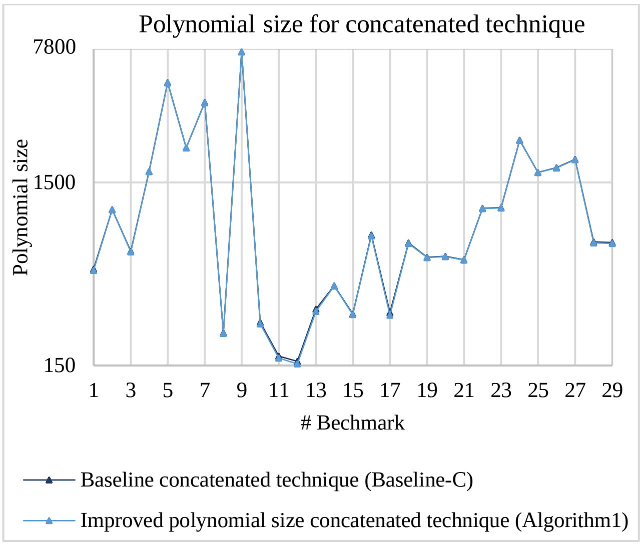
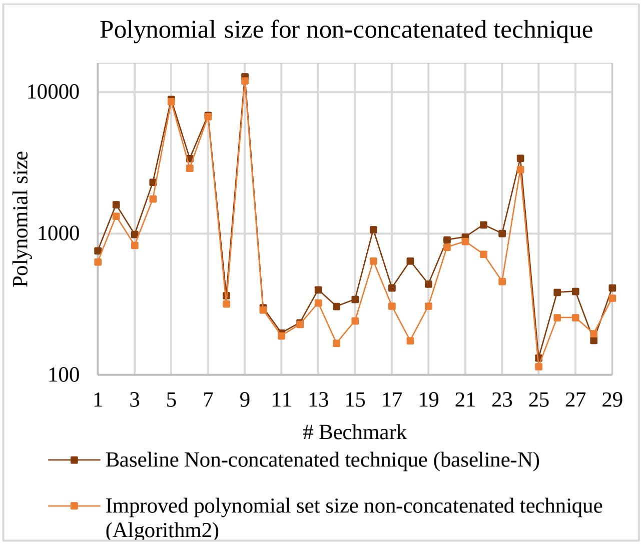


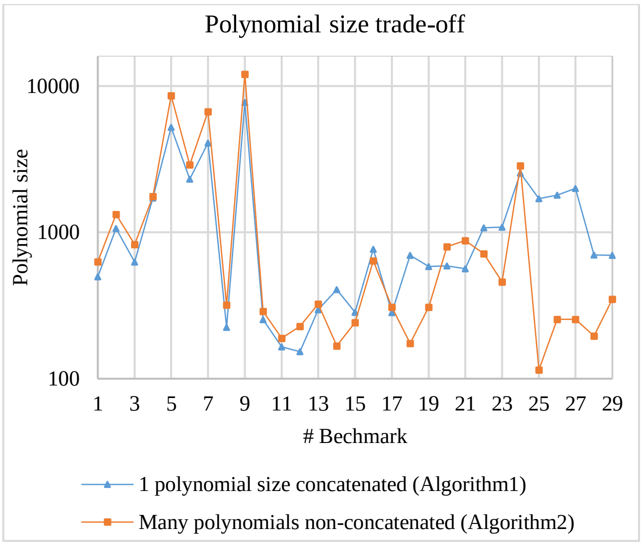
Fig. 14*. Results of each algorithm. Algorithm 1 delivers one polynomial with a reduced application time due to the utilization of only an initial state of the LFSR. Algorithm 2 delivers many polynomials with many seeds, this result in larger application time, due to reseedings.*



(a)



(b)



(c)

Fig. 15. *Polynomial size for each technique. (a) Concatenated technique baseline and optimized numbers, (b) Non-concatenated technique baseline and optimized numbers, (c) Trade-off analysis for both methods.*

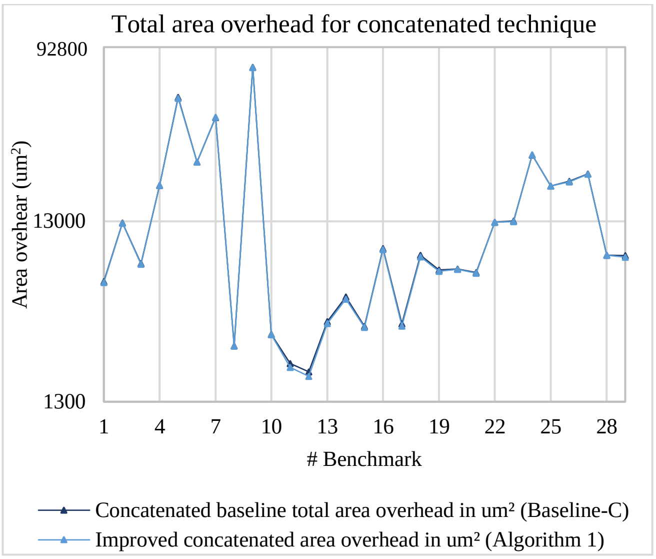
A trade-off analysis of total area overhead for each technique is provided. The calculation of the area is based on the constitution of LFSR as a set of flip-flops and XOR gates, therefore the summation of both elements forms the area for each method. For non-concatenated technique (Fig. 16 (b)), seeds are included in the total area. Seeds are not necessary in the concatenated technique (Fig. 16 (a)) because FFs are set with the necessary initial binary value and can be reset to re-start the mapping of the bit pattern, nevertheless this flip-flop configuration increases the area for concatenated and this is included in the total area overhead trade-off (Fig. 16 (c)). For the area overhead numbers, the technology size of 65 nm is used.

In the results of total area overhead (Fig. 16), the concatenated technique improves the area by setting and resetting the FFs to initial binary value, which puts aside the use of seed storage in a separate memory. The values shown in Fig. 16, has similar shape as the values presented for the polynomial size (Fig. 15), this is due to the strong link of both methods to the size of the LFSR. These results show that concatenated technique of ATPG set, provides an improvement on the area overhead for 90.6% of cases with a best case of 57% smaller area overhead and a mean of 39%. The remaining 9.4% of cases, non-concatenated technique provides a best case of 37% with a mean of 1.4% whilst achieving 100% test mapping in both cases using a known ATPG test set.

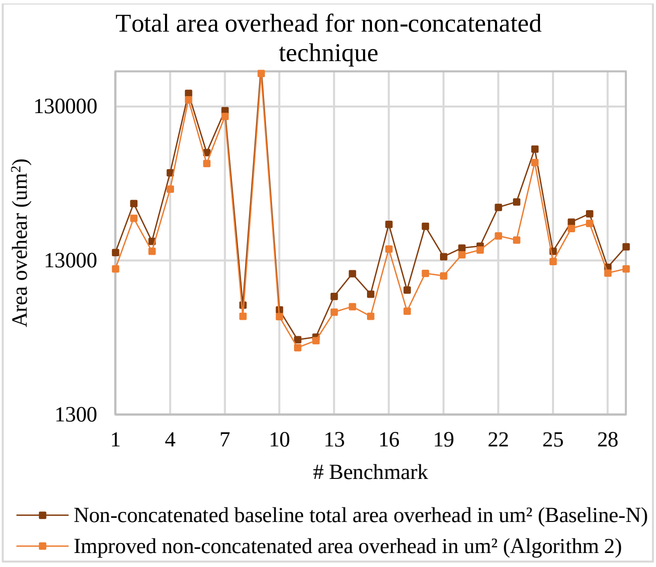
5. Conclusions

In safety critical environment the utilization of embedded electronic systems is increasing constantly and therefore methods for more reliable electronic components are needed. In this paper, BIST technology is used in testing technique, since the fault coverage has a crucial impact on the reliability of the electronic circuit. An analysis on the LFSR is provided in this paper using ATPG test sets. Two techniques are used to target difficult to detect faults with their respective trade-off analysis. This is based on Berlekamp-Massey (BM) algorithm with additional optimizations to reduce area overhead. The first technique is referred as concatenated, and combines all test sets generating a single polynomial that covers complete ATPG set. In this stage the results are referred as baseline-C for this concatenated technique. Further improvement are found using Algorithm 1, reducing polynomial size through Xs assignment. The second technique is referred as non-concatenated and in the initial stage provides a group of LFSRs using BM without including any optimization, referred as the baseline-N. Further optimization is performed through Algorithm 2 that selects full mapping and independent polynomial expressions.

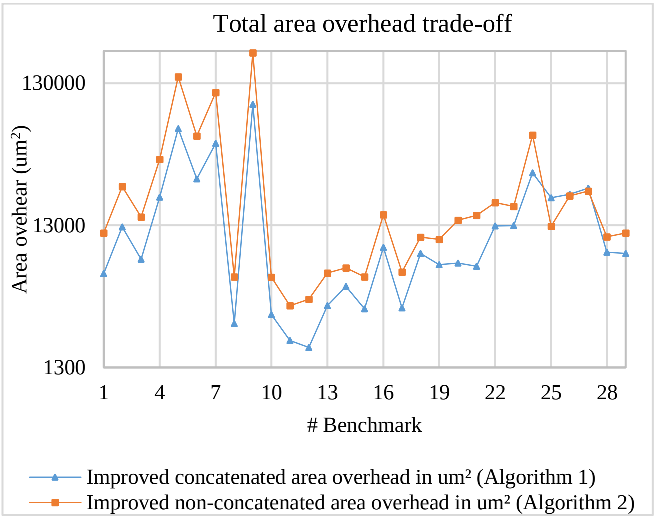
Using 32 benchmarks designs and 65 nm technology library, overall results show that concatenated technique, provides a reduction of area overhead in 90.6% cases with a best case of 57% compared with non-concatenated technique with the left 9.4% with a best case of 37% area overhead reduction. The mean numbers of these techniques show 39% for concatenated compared with a mean area overhead reduction of 1.4% by non-concatenated technique. In both cases, full test coverage is preserved.



(a)



(b)



(c)

Fig. 16. *Total area overhead for each method. For concatenated technique only FFs + XORs. And for non-concatenated additional ROM is included. (a) Concatenated technique baseline and optimized numbers, (b) Non-concatenated technique baseline and optimized numbers, (c) Trade-off analysis for both methods.*

6. Acknowledgements

The authors would like to thank the Consejo Nacional de Ciencia y Tecnología (Mexican Government) and the Department of Electrical Engineering & Electronics, University of Liverpool, UK, for their financial support.

7. References

[1] BSI Standards Limited, "Road vehicles — functional safety, part 10: Guideline on ISO 26262," in *BSI Standards Publication,* BSI Standards Limited ed., BSI Standards Limited, Ed. UK: BSI Standards Limited, 2012, .

[2] V. D. Agrawal, C. R. Kime and K. K. Saluja, "A tutorial on built-in self-test. I. Principles," *IEEE Design & Test of Computers,* vol. 10, *(1),* pp. 73-82, 1993.

[3] L. Wang, C. Wu and X. Wen, *VLSI Test Principles and Architectures: Design for Testability.* Elsevier, 2006.

[4] W. F. Lee and Glaser, *Learning from VLSI Design Experience.* Springer, 2019.

[5] T. McLaurin, *"Periodic Online LBIST Considerations for a Multicore Processor," 2018 IEEE International Test Conference in Asia (ITC-Asia),* pp. 37-42, 2018.

[6] Wang, *System-on-Chip Test Architectures: Nanometer Design for Testability.* Morgan Kaufmann Publishers Inc., 2008.

[7] P. P. Shirvani and E. J. McCluskey, "Fault-tolerant systems in a space environment: The CRC ARGOS project," 1998.

[8] C. Condo and W. J. Gross, "Pseudo-random Gaussian distribution through optimised LFSR permutations," *Electronics Letters,* vol. 51, *(25),* pp. 2098-2100, 2015.

[9] W. C. Lien *et al*, "A new LFSR reseeding scheme via internal response feedback," in *2013 22nd Asian Test Symposium,* 2013, pp. 97-102.

[10] R. Chakraborty and D. R. Chowdhury, "A novel seed selection algorithm for test time reduction in BIST," in 2009, pp. 15-20.

[11] C. V. Krishna, A. Jas and N. A. Touba, "Test vector encoding using partial LFSR reseeding," in *Proceedings International Test Conference 2001 (Cat. no.01CH37260),* 2001, pp. 885-893.

[12] H. Yuan *et al*, "LFSR Reseeding-Oriented Low-Power Test-Compression Architecture for Scan Designs," *J. Electron. Test.,* vol. 34, *(6),* pp. 685-695, 12/01, 2018.

[13] A. A. Al-Yamani and E. J. McCluskey, *"Seed encoding with LFSRs and cellular automata," Proceedings 2003. Design Automation Conference (IEEE Cat. no. 03CH37451),* pp. 560-565, 2003.

[14] B. Koenemann, "LFSR-coded test patterns for scan designs," in *Proceedings of the 2nd European Test Conference - ETC91,* 1991, pp. 237-237.

[15] E. Berlekamp, *Algebraic Coding Theory, Revised Edition.* World Scientific Publishing Co, 2015.

[16] O. Acevedo and D. Kagaris, "On The Computation of LFSR Characteristic Polynomials for Built-In Deterministic Test Pattern Generation," *IEEE Transactions on Computers,* vol. 65, *(2),* pp. 664-669, 2016.

[17] S. Hellebrand *et al*, "Generation of vector pattterns through reseeding of muetiple-polynomial linear feedback shift regist," in 1995, pp. 120.

[18] Yingdi Liu, Nilanjan Mukherjee, Janusz Rajski, Sudhakar M. Reddy, Jerzy Tyszer, "Deterministic stellar BIST for in-system automotive test," in *International Test Conference and International Symposium on Test & Failure Analysis 2018,* USA, 2018, .

[19] S. Jha *et al*, "A cube-aware compaction method for scan ATPG," in 2014, pp. 98-103.

[20] I. Pomeranz, "POSTT: Path-oriented static test compaction for transition faults in scan circuits," in 2017, pp. 1-8.

[21] M. T. He *et al*, "Test-point insertion efficiency analysis for LBIST applications," in *2016 IEEE 34th VLSI Test Symposium (VTS),* 2016, pp. 1-6.

[22] N. A. Touba and E. J. McCluskey, "Test point insertion based on path tracing," in *Proceedings of 14th VLSI Test Symposium,* 1996, pp. 2-8.

[23] O. Acevedo and D. Kagaris, "Using the berlekamp-massey algorithm to obtain LFSR characteristic polynomials for TPG," in *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2012, October 3, 2012 - October 5,* 2012, pp. 233-238.

[24] J. Jeswani, J. Rose and T. Schwarz, "Using algebraic signatures to compress built-in self test on a chip," in 2017, pp. 95-100.

[25] J. H. Anderson, Y. Hara-Azumi and S. Yamashita, "Effect of LFSR seeding, scrambling and feedback polynomial on stochastic computing accuracy," in 2016, pp. 1550-1555.

[26] K. Miyase, S. Kajihara and S. M. Reddy, "A method of static test compaction based on don't care identification," in *Proceedings First IEEE International Workshop on Electronic Design, Test and Applications '2002*, 2002, pp. 392-395.

[27] V. R. Sidorenko and M. Bossert, *"Synthesizing all linearized shift-registers of the minimal or required length," 2010 International ITG Conference on Source and Channel Coding (SCC)*, pp. 1-6, 2010.

8. Appendix 1

Table 1. LFSR optimized size for concatenated technique.

#	Benchmark	Number of patterns (m)	Bit size (n)	m x n	Base line LFSR size	Optimized LFSR size	% LFSR size optimization
1	c432	28	36	1008	505	497	1.6%
2	c499	52	41	2132	1067	1065	0.2%
3	c880	21	60	1260	632	626	0.9%
4	c1355	84	41	3444	1723	1722	0.1%
5	c2670	45	233	10485	5273	5238	0.7%
6	c3540	93	50	4650	2321	2317	0.2%
7	c5315	46	178	8188	4099	4089	0.2%
8	c6288	14	32	448	226	225	0.4%
9	c7552	75	207	15525	7762	7756	0.1%
10	s208	27	19	513	259	253	2.3%
11	s344	14	24	336	169	165	2.4%
12	s349	13	24	312	158	153	3.2%
13	s382	25	24	600	305	297	2.6%
14	s386	63	13	819	409	407	0.5%
15	s400	24	24	576	287	285	0.7%
16	s420	44	35	1540	775	766	1.2%
17	s444	24	24	576	292	283	3.1%
18	s510	56	25	1400	702	696	0.9%
19	s526	49	24	1176	586	584	0.3%
20	s641	22	54	1188	593	591	0.3%
21	s713	21	54	1134	566	564	0.4%
22	s820	94	23	2162	1083	1077	0.6%
23	s832	95	23	2185	1094	1088	0.5%
24	s838	76	67	5092	2548	2540	0.3%
25	s953	76	45	3420	1703	1702	0.1%
26	s1196	113	32	3616	1804	1802	0.1%
27	s1238	125	32	4000	2007	1996	0.5%
28	s1488	101	14	1414	709	701	1.1%
29	s1494	100	14	1400	704	697	1.0%
30	Design A	1200	1437	1.7M	862200	851854	1.2%
31	Design B	1080	1494	1.6M	806760	791432	1.9%
32	Design C	1300	1507	2.0M	979550	969755	1.0%

Table 2. Optimization polynomial set size for non-concatenated technique.

#	Benchmark	Number of patterns (m)	Bit size (n)	Baseline polynomial set	Baseline polynomial mean degree	Optimized polynomial set	Optimized polynomial mean degree	% polynomial set reduction
1	c432	28	36	27	28	18	27	33%
2	c499	52	41	50	32	36	32	28%
3	c880	21	60	21	47	18	47	14%
4	c1355	84	41	66	35	49	35	26%
5	c2670	45	233	45	198	38	197	16%
6	c3540	93	50	81	42	66	43	19%
7	c5315	46	178	46	149	39	150	15%
8	c6288	14	32	14	26	11	26	21%
9	c7552	75	207	70	184	60	186	14%
10	s208	27	19	20	15	17	15	15%
11	s344	14	24	11	18	9	17	18%
12	s349	13	24	13	18	12	18	8%
13	s382	25	24	25	16	18	16	28%
14	s386	63	13	34	9	14	10	59%
15	s400	24	24	18	19	11	19	39%
16	s420	44	35	41	26	27	26	34%
17	s444	24	24	23	18	14	18	39%
18	s510	56	25	32	20	7	21	78%
19	s526	49	24	20	22	14	22	30%
20	s641	22	54	22	41	20	41	9%
21	s713	21	54	21	45	20	45	5%
22	s820	94	23	68	17	34	18	50%
23	s832	95	23	59	17	20	18	66%
24	s838	76	67	61	56	46	56	25%
25	s953	76	45	6	22	5	22	17%
26	s1196	113	32	16	24	15	25	6%
27	s1238	125	32	17	23	15	24	12%
28	s1488	101	14	16	11	14	11	13%
29	s1494	100	14	46	9	25	10	46%
30	Design A	1200	1437	876	1164	788	1106	10.0%
31	Design B	1080	1494	788	1315	512	1249	35.0%
32	Design C	1300	1507	910	1191	701	1131	23.0%

Table 3. Trade-off results for total area overhead

#	Benchmark	Number of patterns (m)	Bit size (n)	m x n	Concatenated	Non-concatenated	% Improvement
1	c432	28	36	1008	5950	11516	48%
2	c499	52	41	2132	12681	24438	48%
3	c880	21	60	1260	7520	14927	50%
4	c1355	84	41	3444	20566	37922	46%
5	c2670	45	233	10485	62497	144255	57%
6	c3540	93	50	4650	27688	55428	50%
7	c5315	46	178	8188	49077	112340	56%
8	c6288	14	32	448	2647	5656	53%
9	c7552	75	207	15525	92758	212959	56%
10	s208	27	19	513	3065	5625	46%
11	s344	14	24	336	2017	3536	43%
12	s349	13	24	312	1799	3930	54%
13	s382	25	24	600	3538	6027	41%
14	s386	63	13	819	4824	6542	26%
15	s400	24	24	576	3365	5646	40%
16	s420	44	35	1540	9095	15493	41%
17	s444	24	24	576	3424	6091	44%
18	s510	56	25	1400	8270	10733	23%
19	s526	49	24	1176	6864	10357	34%
20	s641	22	54	1188	7058	14168	50%
21	s713	21	54	1134	6700	15261	56%
22	s820	94	23	2162	12866	18780	31%
23	s832	95	23	2185	12933	17658	27%
24	s838	76	67	5092	30441	56256	46%
25	s953	76	45	3420	20364	12846	-37%
26	s1196	113	32	3616	21514	20959	-3%
27	s1238	125	32	4000	23768	22656	-5%
28	s1488	101	14	1414	8436	10792	22%
29	s1494	100	14	1400	8230	11491	28%
30	Design A	1200	1437	1.7M	10173822	17938843	43.3%
31	Design B	1080	1494	1.6M	9463843	15049653	37.1%
32	Design C	1300	1507	2.0M	11577889	18424655	37.2%

Table 4. Algorithm 2 results for benchmark s349.

#	Binary representation of LFSRs	Seeds stored in memory	Non-concatenated ATPG test patterns
1	10100011110011	01111111110101	0111111111010101111110000
2	1111100101101	1111000000101	1111000000101011111101011
3	10000111010011101010111	10010111100000001000000	1001011110000000100000001
4	11011100101101000011111	10000000010100000000000	1000000001010000000000000
5	10010001110011001011111	00000000001110100000010	0000000000111010000010111
6	11100000100000101111111	0000000001100000000000	0000000001100000000000101
7	10000000110001110010101	00000000001000110000011 00000000001001100000000	0000000000100011000011100 0000000000100110000000000
8	10111001000101001010111	00000000001011000000100	0000000000101100000010111
9	111001110011	000000000111	0000000001111001000111111
10	10101101010111	00001000010011	0000100001001101000001100
11	1110110101101	0110100000001	0110100000001010000011111
12	1100010011101	0000000000010	0000000000010110000011101

Table 5. Algorithm 1 results for benchmark s349.

Binary representation of LFSR	Set-reset value of the LFSRs flip-flops (seed)	Concatenated ATPG test patterns

1111011111001001100111 1
11001111101110001010110
01000101110010101000101
11001001100001110101001
00100010011001010101101
10011101010110010111000
1110011011101000
	0111111111010101111110 00
	01111111110101011111010
	11100101111000011100001
	11101111111110011011110
	01100111111111111001111
	11111011111111111001100
	0011111111111111
		011111111101010111111000011111111101010111110101 1
		10010111100001110000111101111111110011011110011 0
		01111111111110011111111101111111111110011000011 11
		11111111101111110000111101111111101110101110101 1
		01111111111000001110010101111111111101101111000 0
		01111111101111001110101101101000000010100010111 1
		0111111110010110111101101