Padding Of LFSR Seeds with Low Transition Bits for Reduced Input Test Data Volume

Aiswarya Vijayan P., Sajitha A S
Electronics and Communication Engineering Dept.
Nehru College of Engineering and Research Center Thrissur,
India

Abstract:- In testing there are two primary domains one is reducing input test data volume and next is reducing the test power consumption. Commonly used test compression method is on-chip decompression logic based on LFSR and compressed test consist of seeds for LFSR. The padding of LFSR seeds is a method to generate higher bits of LFSR by adding extra bits to an existing seed and random test patterns can be generated. Further using modified dual CLCG more randomization is possible and it increases the fault coverage. By using bit swapping LFSR it is possible to reduce the number of transitions that occurs in a scan chain. Bit swapping lfsr is used to produce seeds for modified dual CLCG. Thus the overall switching activities can be reduced which automatically reduce the power consumption. The primary aim is to reduce the test pattern and to increase the fault coverage with low transition bits.

Keywords:- *bit swapping linear feedback shift register, modified dual CLCG.*

I. INTRODUCTION

A famous person Gorden Moor, the cofounder of Intel made an prediction in 1965 that the number of transistors in an integrated circuit doubles in every two years. This prediction is known as moor's law. Form the close observations on the advancements happening in the industry shows his prediction or observation is true even in 2022. So as per the moor's law the circuit is getting more complex and the number of components in a chip is increasing in a high rate. Therefore, to ensure the proper functioning of the circuit, the test must be performed correctly. The process of designing chips, testing occupies 70 to 80 percent of time. The well-structured testing methods are required to ensure the proper working of circuit and identify the faults. Thus testing plays an important role in the field of VLSI.This paper primary goal is to reduce the test data volume and to determine the seeds that detect or cover more errors with low transitions while also increasing the test data amount. Normally for a complex circuit the test pattern will be more so the volume of test data will be more. It will use a lot of memory. The polynomial linear feedback shift register (LFSR) is employed to generate testp atterns. With LFSR's padding will help to reduce the use of memory space. Along with this another important factor to be considered is power reduction. The seeds with a high rate of transitions cause an increase in switching activity. Thus there is need to find out the seed which have minimum number of transitions and also it must give maximum fault coverage.

On a tester, the input test data is often saved in a compressed format to store in a confined memory of tester. The circuit's scan chains are loaded with the compressed data via a decompression algorithm [1]and[2] on a chip that yields all input test data based on their compressed state. Once the identical test data compressed for input is utilised for making use of numerous test datas, the size of input test data is decreased to a greater extent. For an instance, in [4],[5] and [7]when the test t_i's compressed input test data can be used for making use of t_i in addition to extra testing with chosen bits are complemented. In such a usually employed test data compression technique, the on-chip decompression logic is primarily developed on a LFSR and the LFSR seeds are included in the compressed tests. A couple of techniques were defined[8] where the seeds seem to be changed to offer several tests. The initial technique emphasize on each seed's bits. Whenever a seed s_i is put into the LFSR after certainly considered one among it's the bit is complemented and it generates an alternative test to the one that is exclusively produced via s_i. The 2nd technique makes use of each seed s_i for numerous different LFSRs which might have few bits as compared to s_i. so using numerous LFSRs[9] and[10] is therefore encouraged . With minimal additional hardware, LFSRs run on a single programmable LFSR. Some of the bits of s_i are shortened when s_i is used for an LFSR with a reduced bit range. With each of those techniques, every seed is utilised to create the use of numerous tests. Consequently, the range of seeds stored is decreased. Rather than cutting off a seed s_i, LFSRs seed with less bits are required, therefore this project emphasizes that a seed's padding results in seeds for LFSR with extra bits which is more efficient in identifying target fault. Thus, padding with few numbers of seeds is certainly needed. Additionally, padding is probable to allevate the growth per test range that want to be carried out as every seed is padded in numerous various ways.

Assume that, s_i = 00110011 is an 8-bit LFSR seed to demonstrate the concept of padding. Let P_i = {_, 0, 11} be the set of paddings for s_i, where is the empty padding. For an 8-bit LFSR, the empty padding yields a seed s_{i0} = 00110011; for a 9-bit LFSR, the padding 0 yields a seed s_{i1} = 001100110; and for a 10-bit LFSR, the padding 11 yields a seed s_{i2} = 0011001111. Two extra checks based on si may be used by storing three padding bits. There will be a storage savings if there are any 8-bit seeds may be changed with the three paddings used for s_i. It's possible that the number of tests used will rise. But using LFSRs with additional bits helps to reduce the spike, which permits for the detection of more defects. The process in this work describes adding of additional paddings in succession, given several seeds S. It aims to increase the amount of problems discovered by as many seeds as possible by using each new padding. The idea is to get rid of seeds that are no longer needed. Padding is acceptable if the test set's storage requirements are decreased and the number of tests used is limited. This technique takes a long time to reach its final solution. It has the advantage of avoiding the use of superfluous paddings. By enabling the storage of compressed tests upon the tester and replicating these tests with decompression logic on the chip, test data compression technologies minimise volume of the input test data. If every saved test is utilised to produce numerous other tests, the volume of the input test data could be effectively reduced further.

By enabling the storage of compressed tests upon the tester and replicating these tests with decompression logic on the chip, test data compression technologies minimise volume of the input test data. If every saved test is utilised to produce numerous other tests, the volume of the input test data could be effectively reduced further. There are methods for creating tests that are built-in. Three tests are generated from each saved test: a broadside test and two skewed load tests. This enables the number of saved tests to be decreased without sacrificing fault coverage. By shifting its scan-in state for a further one or many clock ycles, skewed load provides a special opportunity to increase a stored test into numerous unique skewed load tests. Neither the prior writings specifically in this field discussed skewed load tests or highlighted this possibility for reducing the volume of test data beyond compression. This is significant because skewed load tests are routinely utilised in typical scan circuits to discover delay issues. The volume reducing approach for the new input test data’s core concept is as follows.

To take benefit from a multi cycle test's capacity in order to find more delay faults than a two cycle test, the test's functional clock cycles must be run with a fast clock, enabling the activation and detection of delay faults. This makes test generation and fault simulation more difficult. Sequential fault modelling of delay faults over numerous clock cycles is required by the processes. In addition, they necessitate more complex delay fault models in which the increased delay caused by a fault is explicitly considered [11]-[17]. This study tackles the problem by proposing a novel method for doing multi cycle tests. This method saves multi cycle tests but only uses two cycle broadside tests. This provides for a similar reduction in the number of saved tests. This makes it possible to decrease the amount of tests that are kept, similar to when multi cycle tests are utilised, while yet letting makes test generation and fault simulation techniques for broadside tests to be employed. Alternatively, storing multi cycle tests as proposed in this study enables the implementation of additional two cycle tests for the same numberiof saved tests. These tests can be used as diagnostic tests or to find additional target defects. Compared to the situation, where broadside tests are stored, the number of tests applied is constantly increased. As a result, the approach presented in this study, when utilised to minimise the amount of stored tests, it decreases the input test data volume but not the test application time. It is anticipated that output responses will employ output compaction.

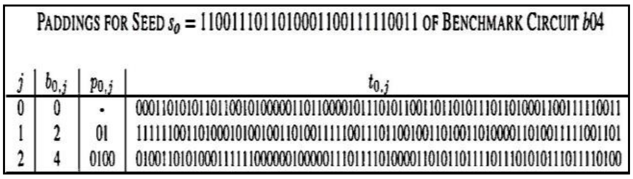
By saving compressed output and tests, and combining output compaction and on-chip decompression logic for test applications, test data compression technique lower the need of storage the test set [11]-[13]. The methodologies mentioned in [14]-[21] employ the similar input test data for many test applications, which improves the capacity to decrease the volume of test data input. The input test data volume can be lowered without compromising fault coverage since the same input test data is used in different ways for applying multiple tests. The input test data is divided in [14] and [18], and an alternative arrangement of the data stored is used to apply a greater number of tests. A stored test s_i is generally applied to apply a single test t_i, is utilised in [15], [16], and [21] pertaining to multiple test with complemented bits according to t_i. The identical input test data are utilized in both the skewed load and broadside tests in [17]. To acquire more test from the same input test data, the circuits are clocked many times in functional mode or scan shift in [19]-[20].

A circuit's number of routes can be enormous. As a result, the path delay faults numbers that a circuit might define is enormous. These are frequently the issues with the longest path delays. Many path delay defects, particularly those connected with the longest paths, are, however, often unnoticed. When test data compression is utilised, a distinct form of limitations on the capacity to detect path delay issues takes place. The compressed tests are saved on the tester in this situation. On-chip decompression logic expands a compressed test into a test that is subsequently applied to the circuit. The number of bits in a compressed test is limited. As a result, not all tests can actually be compressed and used with decompression logic. The path delay faults are chosen within the framework of test data compression must take into account two criteria. (1) A path delay problem on a specific path may be unnoticed. (2) Even if it is detectable, one of its checks may not be able to be applied through the decompression logic. To substitute a test for a seed which can’t be compressed, a straightforward technique is to generate a fresh test. Faults are discovered through tests with limited numbers of the given values (test cubes), and the majority of these test cubes are LFSR seeds. As a result, only few test cubes will need change. Path delay fault tests require assigning more different values than other fault modelling tests. As a result Path delay tests are probably going to have more specific values, and seeds for additional test cubes may be unavailable.

II. PADDING OF LFSR

Make S as a collection of seeds for a collection of target faults F. Every seed in S is calculated using the identical B0-bit LFSR for the sake of clarity. When the B_0-bit LFSR uncompresses a seed s_i,S, it produces a test t_i. T = {t_i: s_i , S} is the test set obtained from S, and it discovers all of the flaws in F. A seed s_i,S is connected with a collection of paddings P_i=$p_{i,0}$, $p_{i,1}$,..., $p_{i,mi}$-1 in the approach proposed in this article. The padding $p_{i,j}$ has $b_{i,j}$ bits for 0 ≤j≤ mi. $P_{i,j}$ = -, with $b_{i,j}$ = 0, indicating that that no padding is used as a specific instance.

When the padding p_{ij} and the seed s_i are combined, you get a seed $s_{i,j}$ = s_i θ pi,j with $B_{i,j}$ = B_0 + $b_{i,j}$ bits. The resulting test is $t_{i,j}$ when the $B_{i,j}$ bit LFSR is used to decompress $s_{i,j}$. The results of the tests performed with various paddings varies greatly. Table I displays the results of tests performed on benchmark circuit b04 with a 28-bit seed s0 and three paddings. The paddings are 0 bytes, 2 bytes, and 4 bytes. The decompression LFSRs are rudimentary LFSRs with 28,30, and 32 bits from [23].The compressed test set for b04 contains 22 seeds and paddings for the 28-bit LFSR. Table 3.3 shows the first seeds and paddings (including s_0 from Table I).



PADDINGS FOR SEED s_0 = 1100111011010001100111110011 OF BENCHMARK CIRCUIT b04

j	$b_{0,j}$	$p_{0,j}$	$t_{0,j}$
0	0	-	0001101010110110010100000110110000101110101100110110101110110100011001111110011
1	2	01	111111001101000101001001101001111100111011001001101001101000011010011111001101
2	4	0100	010011010100011111100000010000011101111010000110101101111011101010111011110100

Table 1: Paddings for seed s_0

Let S contain n seeds and associated sets of paddings in general. By employing paddings for some seeds and eliminating others, the approach outlined decreases S's storage requirements. The total number of paddings is equal to applying a number of tests based on S. The approach outlined allows for more tests runs being conducted in order to reduce storage requirements. A reverse-order fault simulation approach that removes extraneous paddings moderates the rise. It is additionally mitigated by the fact that tests produced by LFSRs with more bits discover more defects, adding to test compaction. The process also includes a limit on the number of tests that can be used.

The tester is supposed to employ the paddings to generate seeds of the type s_i θ $p_{i,j}$. A computer called the site controller loads (compressed) tests into the tester memory in the test configuration described in [24], which the tester subsequently applies to the circuit-under-test. The site controller is anticipated to create padded seeds for the tester after receiving set S. in this test environment. The test application procedure is identical to that for unpadded seeds, and the only requirement for on-chip decompression logic is a programmed LFSR with minimal hardware overhead [7].

A. *Test Pattern Generator*

The LFSR is based on test cubes required to detect target defects. As more test cubes and defect models are compacted, this gets increasingly difficult. The XOR network covers the limits of the test data decompression logic, allowing for test generation for an extended circuit. This enhanced circuit discovers LFSR seeds without having to compute test cubes first. Seeds that provide test cubes for diagnosis by changing seed types that generate fault detection tests. The technique involves two phases: one without diagnostic test cubes and the other with diagnostic test cubes but without aiming to replicate them completely. For defect detection, a similar approach is used to compute a compact set of seeds. Rather of compacting test cubes, calculating seeds, and testing cubes, this method modifies initially random seeds to yield tests that are similar to those in a small, precisely defined test set. These techniques assume that a test that differentiates or identifies target defects is not unique, and that a meaningful test does not have to fully comply with a certain test or test cubes. Eventually, the algorithms tweak until the test it generates, an initial seed meets the goal of discovering or differentiating target flaws. The modification of an initial seed for target fault detection has extra benefits on path delay faults, according to this method. As stated below, This benefit is connected to choosing path delay faults when there are a lot of undetectable path delay faults. Modifies initially random seeds so that the tests they develop are comparable to those found in a small, fully described test set by computing seeds and compacting test cubes.

B. *Procedure for Padding*

This section explains how to choose paddings to reduce the amount of space needed to store a set of seeds S for a set of faults F. S_{init} computes the initial set of seeds for a B_0-bit LFSR. A seed si , S_{init} is linked to a collection of paddings P_i = {-} that solely contains the empty padding. S_{init} requires ST(S_{init}) = nB_0 bits for calculating its storage requirements. AP(S_{init}) = |S_{init}| is the number of applied tests.

The process takes into account a predetermined set of padding options. There are 2^l potential paddings for a length of l. As a result, as long as l is small enough, all paddings of length l = 0, 1,... can be considered. Even for tiny values of l, experimental data for benchmark circuits show that not all paddings are helpful. Furthermore, the effect of varied paddings on storage requirements is the same. As a result of these findings, the following set was chosen. Paddings with lengths of l = 0, 1, and 2 are all included in ∏.

Include all 0 and 1 paddings, as well as paddings with a single 0 or a single 1. For longer durations, include all 0 and 1 paddings, as well as paddings with a single 0 or a single 1. Paddings for l = 4 are, for example, 0000, 1000, 0100, 0010, 0001, 1111, 0111, 1011, 1101, and 1110. 7 + (L-2)(L+5) or O(L^2) is the amount of paddings in ∏. This means that the procedure's runtime grows quadratically with L. Paddings with l > 16 bits are rarely used, according to the test findings with L = 20. Furthermore,if the greatest number of padding bits used in iteraton I ≥ *1* is $l_{\max,i}$, iterations i +1, i + 2,... rarely use paddings with more than $l_{\max,i}$ bits in an iterative application of the technique. Because the runtime is proportional to L2, a cheoice of L that is too big increases the runtime unnecessarily. As a result, for iteration 1, L = min{16, B_0=/2} is employed. L = $l_{\max,i-1}$ for iteration I > *1*. It's worth noting that using a different set of paddings could result in lower storage requirements. To take advantage of this observation, the method can be resumed with a different set of paddings after it has finished with one set.

The techniqueis restarted with the following setsof paddings for the experiments in this article. All paddings of length $0 \leq l \leq 16$ with no 0s, no 1s, a single 0 or a single 1 are included in this set ∏=∏'. The set $\prod^m$ contains all the paddings of length $0\leq l \leq 16$ that have m 0s or m 1s for m ≥2. All the potential paddings of length $0 \leq l \leq 16$ are included in the union of all the sets $\prod^m$ with $l \leq m \leq 8$. Allowing the padding to come before or after the seed is another option.

The process for lowering S's storage requirements is shown in Fig.1 S = S_{init} at first. The technique iteratively considers the paddings one by one. After an iteration in which S's storage requirements do not decrease, the procedure ends. During an iteration, the method performs the steps below for each padding B. It constructs a fresh set of seeds S_{new} using B. B is utilised as a padding for each seed in S_{new}, which increases fault coverage. If a seed s_i with its current set of paddings P_i does not increase the fault coverage after is added to the subset of paddings for numerous seeds, s_i (along with P_i) is excluded from S_{new}.

The technique uses forward-looking reverse-order fault simulation to eliminate unneeded paddings from S_{new} once it is built. This procedure may also reveal that S_{new} has more seeds that can be eliminated. The approach uses two conditions to assess whether S_{new} is better than S when considering the final set of seeds, S_{new}. The first criterion stipulates that S_{new}'s storage requirements be smaller than S's. The second requirement is that S_{new}'s number of applied tests does not surpass S_{init}'s by more than a constant percentage. As a result, if $ST(S_{new}) < ST(S)$ and $AP(S_{new}) \leq |S_{init}|(1 + \forall /100)$, the procedure accepts S_{new}. It assigns S = S_{new} in this scenario and considers additional paddings in relation to the new set S. S_{new} is discarded if this is not the case.

The experiments demonstrate that even when $\forall=\infty$ is used, the approach keeps $AP(S_{new})$ within 10% of $AP(S_{init}) = |S_{init}|$. A bigger number of applicable tests is only attained in a small number of circumstances. The process is run with ∀ = 20% to address these scenarios.

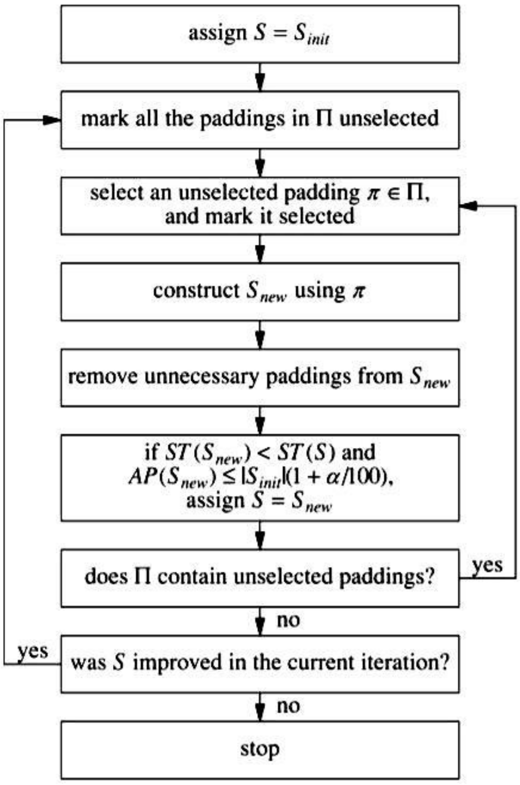

assign $S = S_{init}$
mark all the paddings in Π unselected
select an unselected padding $\pi \in \Pi$, and mark it selected
construct S_{new} using π
remove unnecessary paddings from S_{new}
if $ST(S_{new}) < ST(S)$ and $AP(S_{new}) \leq |S_{init}|(1 + \alpha/100)$, assign $S = S_{new}$
does Π contain unselected paddings?
yes
no
was S improved in the current iteration?
yes
no
stop

Fig. 1: Procedure for selecting paddings

S_{new} = F is the starting point for the building of S_{new}, and F contains all the target defects. The process takes each seed from S into account individually. Let P_i ={ $p_{i,0}$, $p_{i,1}$,.., $p_{i,mi}$ -1 } when si , S is taken into account. The process computes $s_{i,j} = s_i \theta p_{i,j}$ for $0 \leq j <$ mi and the related test $t_{i,j}$, after which fault simulation with fault dumping of F is carried out under $t_{i,j}$. The technique tosses out s_i and P_i if no defect from F is found. As a result, S_{new} contains fewer seeds than S. In any other case, the process goes as follows.

The procedure assigns $p_{i,mi}$ = B, computes $s_{i,mi}$ and $t_{i,mi}$, then simulates faults with F fault dropping under $t_{i,mi}$. The technique discards $p_{i,mi}$ if none of the errors from F are found. Otherwise, $p_{i,mi}$ in P_i is included. S_i with P_i is also included to S_{new}. Only fault simulation with F fault dropping is necessary for S_{new} building. The maximum number of imulated tests is constrained by AP(S) + |S|, where |S| denotes the contribution of B and AP(S) is the number of tests applied based on S. As shown, the quantity of applied tests based on S is frequently very similar to the quantity of seeds in S_{init}.

When S_{new} is taken into account, paddings could no longer be required. The approach employs forward-looking reverse-order fault simulation as follows to remove extra paddings. The algorithm that builds S_{new} saves the index of the first seed i and the first padding j such that the test $t_{i,\ j}$ detects f for every fault f , F. The procedure executes fault simulation with fault dropping of F while taking into account the seeds and paddings in the reverse order. The following factors are taken into account for each seed i and padding j during this operation. The problem won't be discovered later in the reverse-order fault simulation procedure if $t_{i,j}$ is the first test to find any faults when f , F. As a result, $t_{i,j}$ needs to be simulated. The process simulates faults by dropping F beneath $t_{i,j}$ and indicates this instance, the process signals that $p_{i,j}$ can be eliminated and skips the simulation of $t_{i,j}$. After the reverse-order fault simulation pass is finished, all marked paddings are eliminated. If a seed s_i stays with P_i, it is also eliminated from S_{new}.

C. Bit Swapping LFSR

Bit swapping LFSR[25] is a simple test pattern generator using a standard LFSR and a 2:1 multiplexer, the bit-swapping linear feedback shift register (BS-LFSR) was created shown in fig.2. It is based on a simple bit swapping technique applied to the output sequence of a conventional LFSR. By reducing the total of transitions in the scan input of the CUT, BS-LFSR lowers the average and instantaneous weighted switching activity (WSA) during test operation.

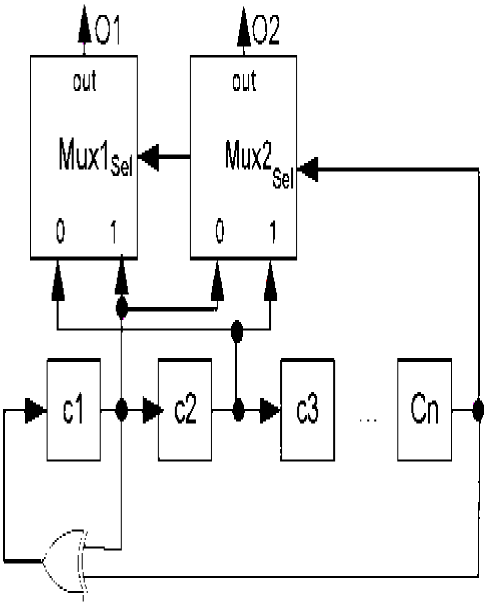


Fig. 2: Bit swapping LFSR

If the first two cells (c1 and c2) have been selected for swapping and cell has a selection line, then O2 (the output of MUX2) will produce a total transition savings of 2^{n-2} compared to the number of transitions produced by each LFSR cell, while o1 has no savings. This is true for an external n-bit maximal-length LFSR that implements the prime polynomial $x^n + x + 1$. (i.e., the savings in transitions is concentrated in one multiplexer output, which means that O2 will save 50 percent of the original transitions produced by each LFSR cell).

D. Modified Dual CLCG

LFSR and linear congruential generator (LCG) are the most frequent and low complexity PRBGs. However, these PRBGs significantly fail random tests and are insecure due to its linearity structure [12], [13]. In the literature, there are many reports of investigations on PRBG based on LFSR [14], chaotic map and congruent modulo. Due to its significant prime factorization challenge, the Blum-Blum-Shub generator (BBS) is among the tested polynomial time unpredictable and cryptographically safe key generators [15][16]. The hardware implementation for executing the large prime integer modulus and computing the huge specia prime integer is rather difficult, despite being secure. There are several BBS PRBG architectures, which are covered in [17] and [18]. A low hardware complexity coupled LCG (CLCG) has been suggested in [17] and [18] to reduce it because the majority of them either use a lot of hardware space or have excessive clock latency. The CLCG approach, which couples two LCGs to create the pseudorandom bit at each clock cycle, is more secure than chaotic-based PRBGs and single LCGs [19]. The discrete fourier transform (DFT) test and five other important NIST statistical tests show that the CLCG approach fails, despite an increase in security [20]. DFT analysis uncovers periodic patterns in CLCG, revealing it to be a weak generator. To fix this,[20] presented another PRBG approach, i.e. dual-CLCG that involves two inequality comparisons and four LCGs to generate pseudorandom bit sequence. Only when inequality equations are held does the dual-CLCG approach produce one-bit random output. As a result, it is impossible to produce pseudorandom bits throughout each iteration. To produce random bits in constant clock time, it is therefore extremely difficult to create an efficient architecture.

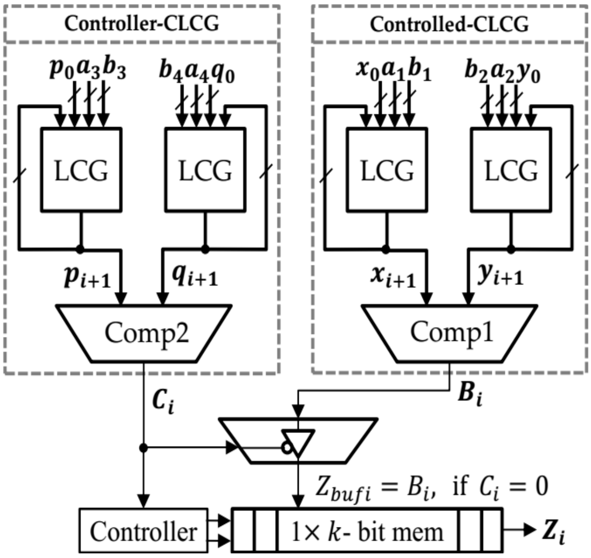


Fig. 3: Architecture of dual clcg

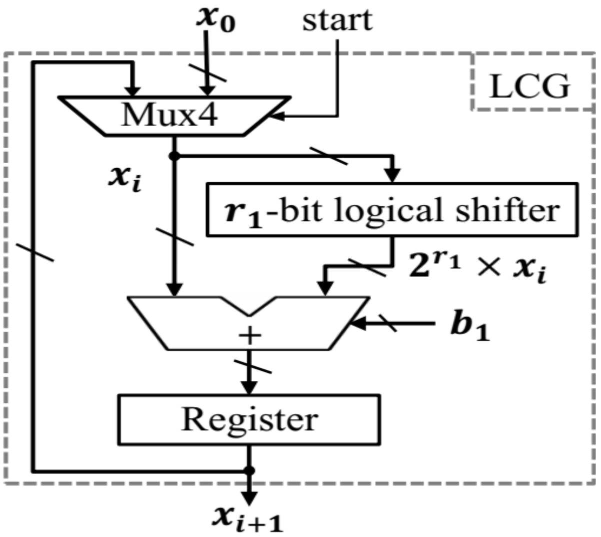


Fig. 4: Architecture of the linear congruential generator

A novel PRBGomethod[26] and its architecture are offered in this study to address the aforementioned issues with the current dual-CLCG method and its design shown in fig.3 and fig.4. The "Modified dual-CLCG" approach is the modified version of the PRBG method that has been proposed. The suggested improved dual-CLCG approach, which is defined mathematically as follows, produces pseudorandom bits by congruentially adding the outputs of two coupled linear congruential generators (CLCGs) by modulo 2.

$$x_{i+1}= a_1 \times x_i + b_1 \bmod 2n \qquad (1)$$
$$y_{i+1} = a_2 \times y_i + b_2 \bmod 2n \qquad (2)$$
$$p_{i+1} = a_3 \times p_i + b_3 \bmod 2n \qquad (3)$$
$$q_{i+1} = a_4 \times q_i + b4 \bmod 2n \qquad (4)$$

The congruential modulo-2equation[25] is used to generate the pseudorandom bit sequence Zi (5)

$$Z_i =(B_i + C_i)\mathrm{mod}2 =B_i \text{ xor } C_i \qquad (5)$$

$$B_i = \begin{cases} 1, & if\ xi+1 > yi+1 \\ 0, & else \end{cases}$$

And

$$C_i=\begin{cases} 1, & if\ pi+1 > qi+1 \\ 0, & else \end{cases}$$

Here, the initial seeds are x_0, y_0, p_0, and q_0 generated using bit swapping LFSR while the constant parameters are a_1, b_1, a_2, b_2, a_3, b_3, a_4 and b_4. The prerequisites for obtaining the maximum length period are the same as those for the dual-CLCG approach currently in use. Equation is used to specify the congruential modulo-2 addition of two separate connected LCG outputs in the proposed modified dual-CLCG approach (5). As a result, the congruential modulo-2 addition generates one-bit random output after each iteration without skipping any random bits. Since the connected LCG outputs has a maximum length period of 2n for an n-bit modulus operand. To perform the modulo-2 addition operation, it takes only single XOR logic. Therefore, with equation (5), the proposed PRBG method can reduce the large memory area used in the existing dual-CLCG method and also can achieve the full-length period of 2n.

III. RESULT AND DISCUSSION

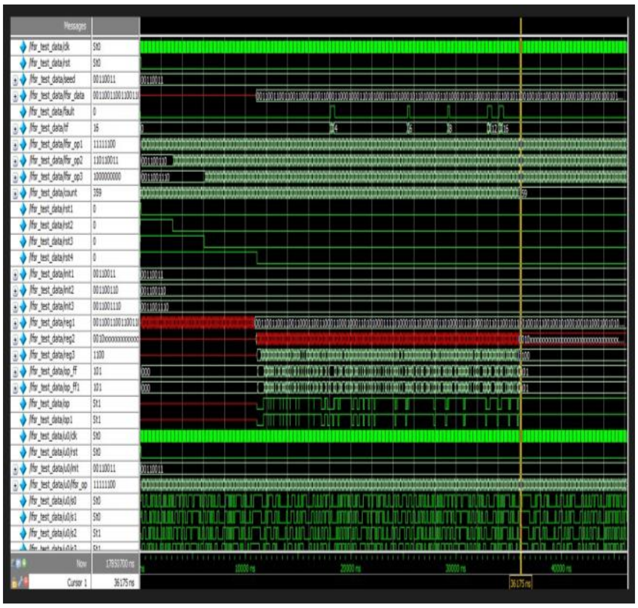


Fig. 5: Existing method

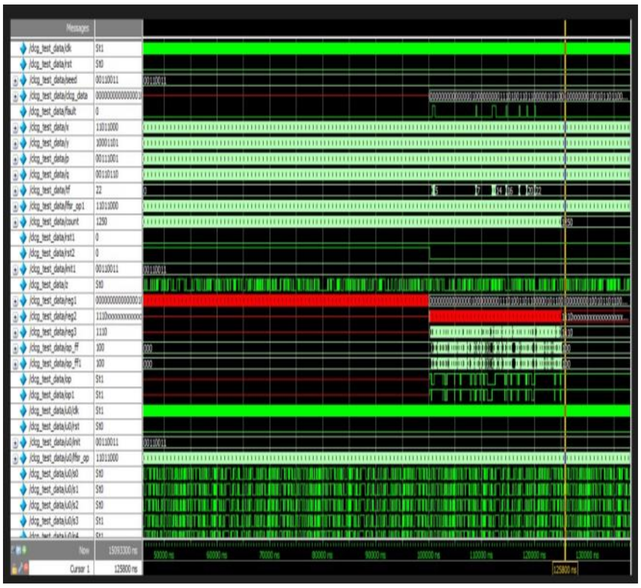


Fig. 6: Proposed method

For padding of LFSR, an eight bit LFSR is taken. The seed for 8 bit LFSR is 00110011. For making 9 bit LFSR one bit is padded to existing seed. Here 0 is padded with 8 bit seed and 9 bit seed is generated as 001100110. Similarly two bits 10 is added to generate 10 bit LFSR. S27 bench mark circuit is used testing and obtain fault coverage. For getting fault coverage stuck at 0 fault is inserted in one of the gate in S27 benchmark circuit.

Fig.5 shows the existing method and fig.6 is the proposed method in which lfsr_test_data/clk indicates the clock for the circuit and lfsr_test_data/rst is the common reset. lfsr_test_data/seed is the input 8 bit seed value. lfsr_test_data/fault is 0 indicate no fault and 1 indicated presence of fault. lfsr_test_data/tf shows the total number of faults. lfsr_test_data/op_ff is the output of S27 benchmark circuit and lfsr_test_data/op_ff1 is S27 benckmark circuit with a stuck at 1 fault. When op_ff is not equal to op_ff1 a fault is detected thus lfsr_test_data/fault value changes to 1 and a number is added to lfsr_test_data/tf. By comparing fig 5.1 and 5.2 it is visible that fault coverage in existing method is 16 and fault coverage in proposed method is 22. The use of bit swapping LFSR in modified dual CLCG reduces the number of transitions which leads to reduced switching activities.

IV. CONCLUSION

The padding of LFSR seeds helps to generate the higher bit lfsr by padding extra bits to existing seeds this method also helps to reduce the test data volume. In VLSI testing most algorithms focuses on either reduction of test data volume or reduction in average power consumption. Hence by using bit swapping LFSR with padding technique reduces the number of transitions thus the switching activities are reduced gradually results in low power consumption with reduced test data volume. Also using modified dual CLCG method randomization is possible and it will improve the fault coverage as compared to existing padding method. From the analysis of experiments fault coverage obtained from the existing method is 16 and the fault coverage of proposed method 22. Here the main focus is given to reduce the test data volume along with reduced number of transitions.

REFERENCES

[1.] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller, and B. Koenemann, "OPMISR: The foundation for compressed ATPG vectors," in Proc. Int. Test Conf., 2001, pp. 748–757.

[2.] N. A. Touba, "Survey of test vector compression techniques," IEEE Des. Test. Comput., vol. 23, no. 4, pp. 294–303, Apr. 2006.

[3.] Pomeranz and S. M. Reddy, "A storage based built-in test pattern generation method for scan circuits based on partitioning and reduc tion of a precomputed test set," IEEE Trans Comput., vol. 51, no. 11, pp. 1282–1993, Nov. 2002.

[4.] Pomeranz and S. M. Reddy, "Static test data volume reduction using complementation or modulo-M addition," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 6, pp. 1108–1112, Jun. 2011.

[5.] D. Czysz, G. Mrugalski, N. Mukherjee, J. Rajski, P. Szczerbicki, and J. Tyszer, "Deterministic clustering of incompatible test cubes for higher power-aware EDT compression," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 30, no. 8, pp. 1225–1238, Aug. 2011.

[6.] Pomeranz, "On the computation of common test data for broadside and skewed-load tests," IEEE Trans. Comput., vol. 61, no. 4, pp. 578–583, Apr. 2012.

[7.] Y. Liu, N. Mukherjee, J. Rajski, S. M. Reddy, and J. Tyszer, "Deterministic stellar BIST for in-system automotive test," in Proc. Int. Test Conf., pp. 1–9, 2018.

[8.] Pomeranz, "Input test data volume reduction using seed comple mentation and multiple LFSRs," in Proc. VLSI Test Symp., 2020, pp. 1–6.

[9.] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift register," in Proc. Int. Test Conf., 1992, pp. 120–129.

[10.] O. Acevedo and D. Kagaris, "Using the Berlekamp–Massey algorithm to obtain LFSR characteristic polynomials for TPG," in Proc. Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst., 2012, pp. 233–238

[11.] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," IEEE Internet Things J., vol. 5, no. 4, pp. 2483–2495, Aug. 2018.

[12.] E. Zenner, "Cryptanalysis of LFSR-based pseudorandom generators— A survey," Univ. Mannheim, Mannheim, Germany, 2004.

[13.] J. Stern, "Secret linear congruential generators are not cryptographically secure," in Proc. 28th Annu. Symp. Found. Comput. Sci., Oct. 1987, pp. 421–426.

[14.] D. Xiang, M. Chen, and H. Fujiwara, "Using weighted scan enable signals to improve test effectiveness of scan-based BIST," IEEE Trans. Comput., vol. 56, no. 12, pp. 1619–1628, Dec. 2007.

[15.] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo□random number generator," SIAM J. Comput., vol. 15, no. 2, pp. 364–383, 1986.

[16.] Sidorenko and B. Schoenmakers, "Concrete security of the Blum Blum-Shub pseudorandom generator," in Cryptography and Coding (Lecture Notes in Computer Science), vol. 3796. Berlin, Germany: Springer, Nov. 2005, pp. 355–375.

[17.] K. Panda and C. K. Ray, "FPGA prototype of low latency BBS PRNG," In Proc. IEEE Int. Symp. Nanoelectron. Inf. Syst. (INIS), Indore, India, Dec. 2015, pp. 118–123.

[18.] P. P. Lopez and E. S. Millan, "Cryptographically secure pseudorandom bit generator for RFID tags," in Proc. Int. Conf. Internet Technol. Secured Trans., London, U.K., vol. 11, Nov. 2010, pp. 1–6.

[19.] R. S. Katti and R. G. Kavasseri, "Secure pseudo-random bit sequence generation using coupled linear congruential generators," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Seattle, WA, USA, May 2008, pp. 2929–2932.

[20.] R. S. Katti, R. G. Kavasseri, and V. Sai, "Pseudorandom bit generation using coupled congruential generators," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 3, pp. 203–207, Mar. 2010.

[21.] Sidorenko and B. Schoenmakers, "Concrete security of the BlumBlum-Shub pseudorandom generator," in Cryptography and Coding (Lecture Notes in Computer Science), vol. 3796. Berlin, Germany: Springer, Nov. 2005, pp. 355–375.

[22.] K. Panda and C. K. Ray, "FPGA prototype of low latency BBS PRNG," In Proc. IEEE Int. Symp.

Nanoelectron. Inf. Syst. (INIS), Indore, India, Dec. 2015, pp. 118–123.

[23.] P. H. Bardell, W. H. McAnney, and J. Savir, Built-In Test for VLSI Pseudorandom Techniques. New York, NY, USA: Wiley Intersci., 1987.

[24.] S. Bodhe, M. E. Amyeen, C. Galendez, H. Mooers, I. Pomeranz, and S. Venkataraman, “Reduction of diagnostic fail data volume and tester time using a dynamic N-cover algorithm,” in Proc. VLSI Test Symp., 2016, pp. 1–6.

[25.] S. Abu-Issa and S. F. Quigley, "Bit-Swapping LFSR and Scan-Chain Ordering: A Novel Technique for Peak- and Average-Power Reduction in Scan-Based BIST," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 5, pp. 755-759, May 2009, doi: 10.1109/TCAD.2009.2015736.

[26.] K. Panda and K. C. Ray, "Modified Dual-CLCG Method and its VLSI Architecture for Pseudorandom Bit Generation," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 3, pp. 989-1002, March 2019, doi: 10.1109/TCSI.2018.2876787.