

Low Power Structure Design of GLFSR for BISR Based Application

¹Kakarla Hari Kishore, ²Dr. Fazal Noor Basha, ³Pavuluri Srinivas, ⁴Atluri Jhansi, ⁵Shaik Moulali

^{1,2,3,4,5}Dept. of ECE, KL University, Guntur, AP, India

Abstract

In this paper, structure design and optimization of a Built-In Self-Repair (BISR) design based on Generalized Linear Feedback Shift Registers (GLFSRs) are described. A new and effective pseudorandom test pattern generator, termed GLFSR, is introduced. These are Linear Feedback Shift Registers (LFSR's) over a Galois field GF. Unlike conventional LFSR's, which are over GF, these generators are not equivalent to cellular arrays and are shown to achieve significantly higher fault coverage. Experimental results are presented in this paper depicting that the proposed GLFSR can attain fault coverage equivalent to the LFSR, but with significantly fewer patterns. The proposed GLFSR structure will be implemented in SPARTAN-3E by using VHDL. The percentage improvement for fault coverage. This approach reduces the number of transitions in the scan chains and thus minimizing power consumption. By using encoding algorithm, the percentage improvement for power consumption.

Keywords

Built-In Self Test (BIST), Built-In Self Repair (BISR), Memory Built-In Self Repair (MBISR), GLFSR'S, Linear Feedback Shift Registers (LFSR's).

I. Introduction

Linear binary Feedback Shift Registers (LFSR's) are extensively used for random pattern generation in the Built-In Self-Test (BIST) environment. This paper introduces a new pseudorandom pattern generator, called a Generalized Linear Feedback Shift Register (GLFSR), which is shown to outperform both LFSR and cellular arrays. These GLFSR's are defined over a higher order Galois field GF. Unlike the LFSR, these GLFSR's are not equivalent to cellular automata, therefore warranting investigation. These GLFSR's provide a uniform framework for study of LFSR, MISR, and multiple MISR. In addition, GLFSR's yield a new structure when the feedback polynomial is primitive and termed as MLFSR. Although this new structure's effectiveness has not yet been fully explored, its effectiveness in the context of test pattern generation is studied here. The results in this paper along with the previously reported work in demonstrate GLFSR as a new and powerful tool for BIST implementation. Importantly, what is shown in this paper is that GLFSR's are significantly more effective than the LFSR, commonly used as a test pattern generator and as an MISR for signature analysis. This is a significant finding when viewed in the context of earlier results which have established that the primitive GLFSR's outperform LFSR's as a signature analyzer. Here, we propose the application of the GLFSR to generate pseudorandom patterns for stuck-at and transition faults. A good pattern generator should generate patterns with high degree of randomness and should have efficient area implementation. A number of LFSR-based pattern generators proposed in the past use combinations of shift registers and XOR gates for particular applications. The GLFSR approach proposed here can be interpreted as a systematic alternative to the popular pattern generators used by IBM which comprises of the standard LFSR and extra XOR gates used at the output of the LFSR as shown in Fig. The outputs of the LFSR are transformed by these extra XOR gates to enhance randomness for

the patterns. These XOR gates are added in a two step process in an adhoc manner. Our GLFSR design provides a one-step technique where the additional XOR gates are integral to the design and are the byproduct of the design methodology. This results in a better random distribution of the patterns and potentially lesser dependencies at the output.

In the paper, we assume a BIST environment where most of the faults are to be detected using the proposed GLFSR's, and then the remaining few very hard-to-detect faults are to be detected by applying predetermined patterns, or by using random weighted patterns. For large circuits, where the testing time is a major concern, using GLFSR's to detect the large fraction of the faults can significantly reduce test time. Specifically, what is shown here is that the patterns generated by the proposed structure

1. can attain the same stuck-at fault coverage in combinational circuits with a small fraction of the number of tests required by LFSR,
 2. can attain the same transition fault coverage as LFSR's with many fewer patterns, and
 3. are more randomly distributed than LFSR's and cellular arrays.
- First, this paper describes the behavior of the GLFSR, suggesting a technique that exhaustively generates all the patterns using the GLFSR.

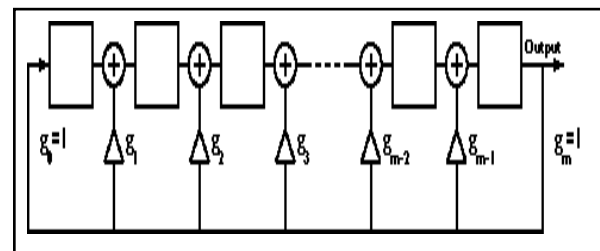


Fig. 1: The generalized LFSR

II. GLFSR—Review

The structure of the GLFSR is illustrated in fig. The Circuit Under Test (CUT) is assumed to have outputs which form the inputs to that GLFSR to be used as the signature analyzer. The inputs and outputs are considered bit binary numbers, interpreted as elements over GF. The GLFSR, designed over GF, has all its elements belonging to GF. Multipliers, adders, and storage elements are designed using conventional binary elements. The feedback polynomial is represented as the i th coefficient multiplies the feedback input over a Galois field which can be realized using XOR gates. As observed in [1] and illustrated in fig. 1, the GLFSR represents a general structure and all known structures like the LFSR, MISR, multiple MISR, etc., being special cases, thus were treated uniformly for the study of aliasing. In addition, the GLFSR with a primitive feedback polynomial $m > 1$, termed MLFSR, represent a new structure that has not been fully explored yet. This new structure MLFSR as a pseudorandom pattern generator is the focus of this paper.

The GLFSR when used to generate patterns for a circuit of n inputs can have m stages, each element belonging to GF where m is at least equal to n . To use the GLFSR as a test pattern generator, a nonzero seed is loaded into the GLFSR and is clocked autonomously to

produce test patterns. Different values of m and n , create different types of GLFSR's, capable of generating different types of patterns for the same n -input circuit. For example, for $n = 36$, we can choose the values of m to be one of the following combinations: (2, 18), (3, 12), (4, 9), (6, 6), (9, 4), etc.

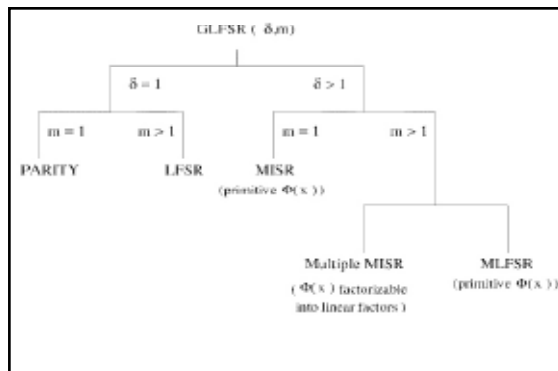


Fig. 2: Special cases of GLFSR

III. Linear Feedback Shift Registers (LFSR)

LFSR is an n -bit shift register which pseudo-randomly scrolls between $2^n - 1$ values, but does it very quickly because there is minimal combinational logic involved. The all zeros case is not possible in this type of LFSR, but the probability of any bit being "1" or "0" is 50% except for that. Therefore, the sequence is pseudorandom in the sense that the probability of a "1" or "0" is approximately 50%, but the sequence is repeatable. Like a binary counter, all $2^n - 1$ states are generated, but in a "random" order that is repeatable. The exclusive-OR gates and shift register act to produce a Pseudo Random Binary Sequence (PRBS) at each of the flip-flop outputs. By correctly choosing the points at which we take the feedback from an n -bit shift register we can produce a PRBS of length $2^n - 1$, a maximal-length sequence that includes all possible patterns (or vectors) of n bits, excluding the all-zeros pattern. In an LFSR, the bits contained in selected positions in the shift register are combined in some sort of function and the result is fed back into the register's input bit. Fig.2 shows a 3bit LFSR. Register bits that do not need an input tap, operate as a standard shift register. It is this feedback that causes the register to loop through repetitive sequences of pseudo-random value. The choice of taps determines how many values there are in a given sequence before the sequence repeats. The implemented LFSR uses a one-to-many structure, rather than a many-to-one structure, since this structure always has the shortest clock-to-clock delay path. The feedback is done so as to make the system more stable and free from errors. Specific taps are taken from the tapping points and then by using the XOR operation on them they are feedback into the registers.

A. Signature Analysis

Signature Analysis is a compression technique based on the concept of cyclic redundancy Checking. The good and faulty circuits produce different signatures. Test Patterns for BIST can be generated at-speed by an LFSR with only a clock input. The outputs of the circuit-under-test must be compared to the known good response. In general, collecting each output response and off-loading it from the circuit under test for comparison is too inefficient to be practical. The General solution is to compress the entire output stream into a single signature value.

B. Two types of LFSRs

1. External Feedback
 2. Internal Feedback
- Higher clock frequency

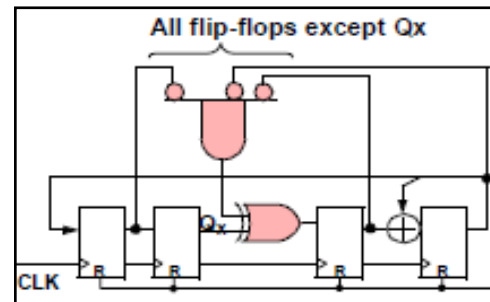


Fig. 3: linear feedback shift register

An LFSR generates periodic sequence

- Must start in a non-zero state,
 - The maximum-length of an LFSR sequence is $2^n - 1$ does not generate all 0s pattern (gets stuck in that state)
- (i). The characteristic polynomial of an LFSR generating a maximum-length sequence is a primitive polynomial
 - (ii). A maximum-length sequence is pseudo-random:
 - number of 1s = number of 0s + 1
 - same number of runs of consecutive 0s and 1s
 - 1/2 of the runs have length 1
 - 1/4 of the runs have length 2

1. Basic 4-bit LFSR XOR-Based

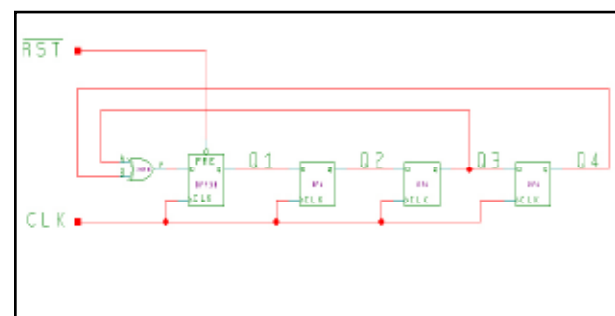


Fig. 4:

2. 4-Bit LFSR [4, 1] States and Output

Register States				Output Stream
Bit 1 (Tap)	Bit 2	Bit 3	Bit 4 (Tap)	
1	1	0	1	1
0	1	1	0	1
0	0	1	1	0
1	0	0	1	1
0	1	0	0	1
0	0	1	0	0
0	0	0	1	0
1	0	0	0	1
1	1	0	0	0
1	1	1	0	0
1	1	1	1	0
0	1	1	1	1
1	0	1	1	1
0	1	0	1	1
1	0	1	0	1
1	1	0	1	0

Fig. 5:

C. Pseudo Random Binary Sequence

PRBS or Pseudo Random Binary Sequence is essentially a random sequence of binary numbers. It is random in a sense that the value of an element of the sequence is independent of the values of any of the other elements. It is 'pseudo' because it is deterministic and after N elements it starts to repeat itself, unlike real random sequences.

Examples of random sequences are radioactive decay and white noise.

A Binary Sequence (BS) is a sequence of N bits, for $j = 0, 1, \dots, N-1$, i.e. m ones and $N-m$ zeros. A binary sequence is pseudo-random (PRBS) if its autocorrelation function,

$$C(v) = \sum_{j=0}^{N-1} (a_j a_{j+v})$$

has only two values:

$$C(v) = m \text{ if } v = 0 \pmod{N}$$

$$C(v) = mc \text{ if } v \neq 0 \pmod{N}$$

where

$$c = (m-1)/(N-1)$$

is called the duty cycle of the PRBS

The implementation of PRBS generator is based on the linear feedback shift register, which consists of 'n' master slave flip-flops. The PRBS generator produces a predefined sequence of 1's and 0's, with 1 and 0 occurring with the same probability PRBS is implemented using LFSR or Linear Feedback Shift Register.

LFSR is an n -bit shift register which pseudo-randomly scrolls between 2^n-1 values, but does it very quickly because there is minimal combinational logic involved. Once it reaches its final state, it will traverse the sequence exactly as before. One of the two main parts of an LFSR is the shift register (the other being the feedback function). A shift register is a device whose identifying function is to shift its contents into adjacent positions within the register or, in the case of the position on the end, out of the register. The position on the other end is left empty unless some new content is shifted into the register. The contents of a shift register are usually thought of as being binary, that is, ones and zeroes. If a shift register contains the bit pattern 1101, a shift (to the right in this case) would result in the contents being 0110; another shift yields 0011. After two more shifts, things tend to get boring since the shift register will never contain anything other than zeroes.

Two uses for a shift register are:

1. Convert between parallel and serial data
2. Delay a serial bit stream.

The conversion function can go either way fill the shift register positions all at once (parallel) and then shift them out (serial) or shift the contents into the register bit by bit (serial) and then read the contents after the register is full (parallel). The delay function simply shifts the bits from one end of the shift register to the other, providing a delay equal to the length of the shift register.

IV. Proposed BISR Scheme

A. Redundancy Organization

The redundancy organization of a RAM affects not only the repair rate but also the area cost of the BIRA circuitry. Figure shows a RAM cell array with redundancy rows and columns. In the figure, a 512-bit RAM has two spare rows at the bottom and four spare columns on the right. If there is a faulty row, any of the Spare Rows (SR0 and SR1) can be used to replace it. However, the spare columns are used differently: we partition them into several Spare Column Groups (SCGs).

In the fig., two spare columns are grouped into an SCG, i.e., the group size is 2 groups SCG0 (resp. SCG1) contains columns SC0

and SC1 (resp. SC2 and SC3). Moreover, each SCG is logically divided into segments for better utilization, i.e., the segments of a spare column are not physically divided by local sense amplifiers. The switching from the main memory to the spare column groups is controlled by the BISR circuit, so only the multiplexers induce additional access time and area cost. In fig, one SCG has four segments (SEG0, SEG1, SEG2, and SEG3).

The segments are identified by the first two most significant bits (MSBs) of the row address. Let the row addresses be $a_3a_2a_1a_0$, then a_3a_2 specifies the segment where the addressed row sits. Different segments of an SCG can be used to repair defective cells in different columns of the main memory. For example, if the cells C0, C1, C2, and C3 (see fig. 1) are faulty, then SEG0 and SEG1 of SCG0 can be used to replace them. Note that the spare rows also can be logically divided into segments, though this is not shown in the example. We have developed a simulator for evaluating the RA algorithms of redundancy repairable memories

B. BISR Architecture and Procedure

Fig. depicts the block diagram of the proposed BISR scheme, including the BIST module, BIRA module, and test wrapper for the memory. The BIST circuit detects the faults in the main memory and spare memory. It is programmable at the March element level. The BIRA circuit performs redundancy allocation using the proposed RA algorithm. The test wrapper switches the memory between the test/repair mode and normal mode. In the test/repair mode the memory is accessed by the BIST module, while in the normal mode the wrapper selects the data outputs either from the main memory or the spare memory (replacing the faulty memory cells) depending on the control signals from the BIRA module. If a fault is detected (ERR outputs a pulse), the test process pauses and the BIST module exports the FS to the BIRA module, which then performs the RA procedure. When the procedure is completed and the memory testing is not finished yet, the BIRA module issues a continue signal (CNT) to resume the test process.

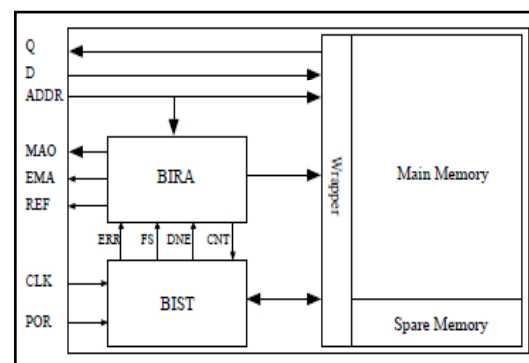


Fig. 6: Block diagram of the proposed BISR scheme

The BISR procedure is shown in Fig. Upon turning on the power, the BIST module starts to test the spare memory. Once a fault is detected, it informs the BIRA module to mark the defective spare row or column as faulty through the Error (ERR) and Fault Syndrome (FS) signals. After finishing the spare memory test, it tests the main memory. During the RA procedure, if a spare row is requested but there is no more spare row, the BIRA module exports the faulty row address through the EMA (Export Mask Address) and MAO (Mask Address Output) signals. The memory will then be operated at a down-graded mode by software-based address remapping. For example, assume that a memory with multiple blocks is used for buffering, and the blocks are chained

by pointers. If some block is faulty and should be masked, then the pointers are updated to invalidate the block. The size of the memory is reduced, as one block is removed. The system still works if a smaller buffer is allowed, though the performance may be affected. This approach effectively increases the yield of the products. The number of blocks that can be invalidated normally depends on the performance penalty that can be tolerated. If the down-grade mode is not allowed, the MAO is removed and the EMA indicates whether the memory is repairable

V. BIST Module

The BIST module block diagram is shown in fig., which consists of a Controller (CTR) and a Test Pattern Generator (TPG) for handling test operations and generating test stimulus, respectively. In addition to the Clock (CLK), the BIST only needs the Power-On Reset (POR) signal to initiate the test procedure. The BDN (BIST done), ERR (Error Indicator), FS (Fault Syndrome), and CNT (continue) are signals between the BIST and BIRA modules. The TPG output signals are connected to the memory under test. The BNS (BIST Normal Selection) signal is used to switch the memory between test/repair mode and normal mode.

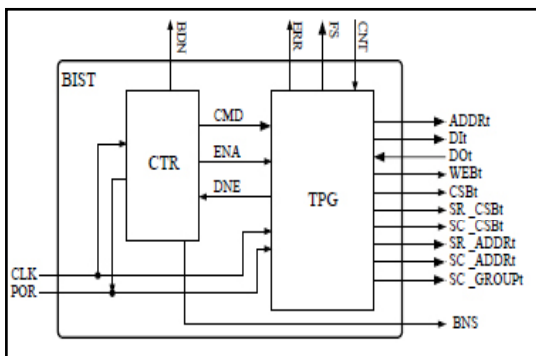


Fig. 7: BIST Module Block Diagram

The CTR is a typical finite state machine. The TPG executes the test Command (CMD) provided by the CTR. When a fault is detected, it pauses and sends the ERR and FS signals to inform the BIRA module to perform RA. When the RA finishes, the BIRA module sends the CNT signal to resume the TPG process. The BIST implementation is typical, similar to our previous design .

A. BIST – An Overview

A digital system is tested and diagnosed during its lifetime on numerous occasions. It is very critical to have quick and very high fault coverage testing. One common and widely used in semiconductor industry for IC chip testing is to ensure this is to specify test as one of the system functions and thus becomes self-test. A system designed without an integrated test strategy which covering all levels from the entire system to components is being described as chip-wise and system-foolish. A properly designed Built-In-Self-Test (BIST) is able

B. BIST Implementation

Figure shows the BIST hardware architecture in more detail. In this project, the BIST module in the IOP is developed based on the architecture in fig. Basically, a design with embedded BIST architecture consists of a test controller, hardware pattern generator, input multiplexer, Circuit Under Test (CUT) which in this project is the IOP and output response compactor. Optionally, a design with BIST capability may include also the comparator and Read-Only-Memory (ROM). As shown in fig., the test controller

is used to control the test pattern and test generation during BIST mode. Hardware pattern generator functions to generate the input pattern to the CUT. Normally, the pattern generator generates exhaustive input test patterns to the CUT to ensure the high fault coverage. For example, a CUT with 10 inputs will require 1024 test patterns. Primary Inputs are the input for CUT during the non BIST mode or in other words, functional mode. Input multiplexer is used to select correct inputs for the CUT for different mode. During BIST mode, it selects input from the hardware pattern generator while during functional mode, selects primary inputs. Output response compactor acts as compactor to reduce the number of circuit responses to manageable size that can be used as the signature and stored on the ROM. Implementation of the pattern generation as well as the response compactor will be discussed in more details in section below. Fig., to offset the cost of added test hardware while at the same time ensuring the reliability, testability and reduce maintenance cost.

C. BIST Process

Fig. shows, the BIST system hierarchy for the 3 level of packaging which is the system level, board level and chip level. The system consists of several PCBs (or boards). Each of the PCB has multiple chips. The system Test Controller can activate self-test simultaneously on all PCBs. Each Test Controller on each PCB board can activate self-test on all the chips on the board. The chip Test Controller runs the self-test on the chip and transmits the result out to the board Test Controller. The board Test Controller accumulates test results from all chips on the PCB and sends the results to the system Test Controller. The system Test Controller uses all of these results to determine if the chips and board are faulty. As mentioned earlier, a BIST block can optionally consist of a ROM and a comparator. ROM is used to store the golden signature obtained from simulation at the pre-silicon phase. A comparator is used to compare the signature obtained during BIST mode with the golden signature. If the signature matched with the golden signature, then the chip is considered as fault free. On the other hand, if the signature is not matching with the golden signature, then the chip is considered as faulty. From fig., the wires from primary inputs to the input multiplexer and the wires from circuit output P to primary outputs is not able to be tested by BIST. These wires require another testing method such as an external ATE or JTAG Boundary Scan hardware.

VI. Results

A. 8 bit Lfsr schematic diagram

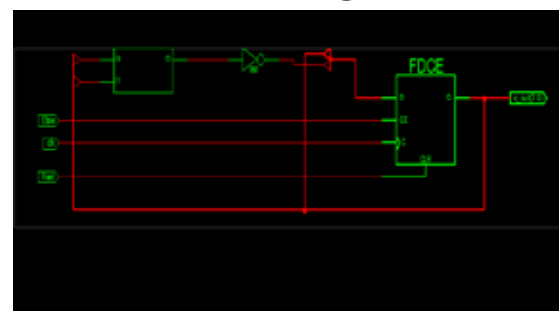


Fig. 8:

B. Patter generator using xor waveform

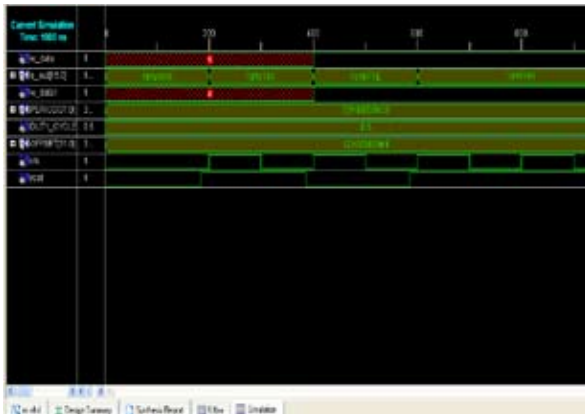


Fig. 9:

C. PRBS waveforms

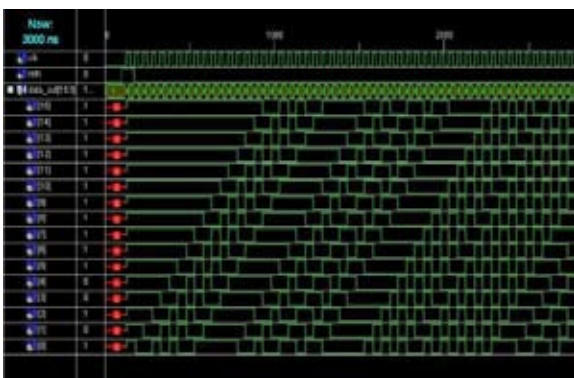


Fig. 10:

D. BIST waveforms

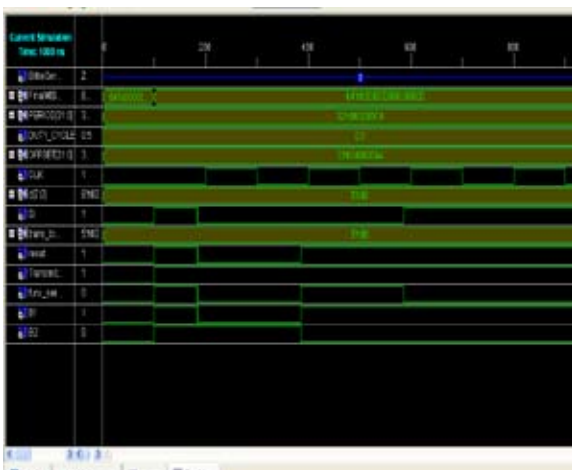


Fig. 11:

VII. Conclusion

A new and effective pseudorandom pattern generator, GLFSR, is proposed here. The implementation techniques are discussed first. Patterns generated by the GLFSR's are demonstrated to be evenly distributed and, hence, have the potential to detect most faults rapidly. Experimental results on combinational circuits demonstrate that GLFSR's can detect 90 to 95% of the stuck-at and transition faults requiring much fewer patterns than LFSR. These experimental results also show that the "randomness" of the patterns generated by the GLFSR's is much higher than the LFSR's and the cellular arrays. This can be of significance when detecting faults in very large circuits with a large number of

inputs. Further work is underway in studying the effectiveness of GLFSR's for sequential circuits and also with respect to other nontraditional faults.

References

- [1] D. K. Pradhan, S. K. Gupta, "A new framework for designing & analyzing BIST techniques and zero aliasing compression", in IEEE Trans. Comput., Vol. 40, 1991.
- [2] A. M. Patel, "Two-level coding for error control in magnetic disk storage products", IBM J. Res. Develop., Vol. 33, No. 4, 1989.
- [3] D. K. Pradhan, "A theory of Galois switching functions", IEEE Trans. Comput., Vol. C-27, pp. 239-249, 1978.
- [4] M. Chatterjee, "An integrated framework for synthesis for testability", Ph.D. dissertation, Dept. Comput. Sci., Texas A&M University, 1998.
- [5] K. S. Trivedi, "Probability and Statistics with Reliability, Queueing, and Computer Science Applications", Englewood Cliffs, NJ: Prentice Hall, 1982.
- [6] P. H. Bardell, W. H. McAnney, J. Savir, "Built-in Test for VLSI: Pseudo-Random Techniques," New York: Wiley, 1987.
- [7] P. H. Bardell, "Design considerations for parallel pseudorandom pattern generators", J. Electron. Testing: Theory and Applications, vol. 1, pp. 73-87, 1990.
- [8] "Analysis of cellular automata used as a pseudorandom pattern generators", in Int. Test Conf., 1990, pp. 762-768.
- [9] D. K. Pradhan, M. Y. Hsiao, A. M. Patel, S. Y. Su, "Shift registers designed for on-line fault detection", presented at Int. Conf. Fault-Tolerant Computing, 1978.
- [10] K. Iwasaki, F. Arakawa, "An analysis of the aliasing probability of multiple input signature registers in the case of 2m-ary symmetric channel", IEEE Trans. Computer-Aided Design, Vol. 9, 1990.
- [11] D. Knuth, "Semi-Numerical Algorithms. Reading", MA: Addison Wesley, 1981.
- [12] P. D. Hortensius, R. D. McLeod, H. C. Card, "Parallel random number generation for VLSI systems using cellular automata", IEEE Trans. Comput., Vol. 38, pp. 1466-1473, 1989.



Pavuluri Srinivas was born in ongole, prakasam(dt), a.p ,india. He received the B.Tech. degree in Electronics & Communications Engineering from VRS & YRN college of engineering & technology, chirala in 2010 and pursuing M.Tech (VLSI) in K L University. . His research interests include Digital VLSI Design and Fault Diagnosis and testing.



Harikishore Kakarla was born in Vijayawada, Krishna (Dist.), AP, India. He received B.Tech. in ECE, from St. Johns College of Engg. & Tech, Kurnool (Dist.), AP, India, M.Tech from G. Pulla Reddy Engineering College, Kurnool, AP, India. He is pursuing Ph.D in the area of VLSI in KL University, Vijayawada, AP, India. He is working as Assistant Professor for Department of ECE, KL University, Vijayawada, AP, India. He has published one National Conference.



Dr. Fazal Noorbasha was born on 29th April 1982. He received the B.Sc. degree in Electronics Science from BCAS College, Bapatla, Guntur, A.P., Affiliated to the Acharya Nagarjuna University, Guntur, A.P., India, in 2003, M.Sc. degree in Electronics from the Dr. HariSingh Gour Central University, Sagar, M.P., India, in 2006, M.Tech. Degree in VLSI technology, with specialization in Embedded systems from the North Maharashtra University, Jalgaon, M.S., INDIA in 2008 And Ph.D. from Department Of Physics and Electronics, Dr. HariSingh Gour Central University, Sagar, M.P., India, in 2011. Presently he is Assistant Professor, Department of ECE, KL University, where he has been engaged in the research and development of low-power, high-speed CMOS VLSI technology, Memory Processors LSI's, Digital Image Processing, Embedded Systems and Nanotechnology. He has published over 16 technical papers in international and National reputed journals and conferences.



Shaik Moulali was born in guntur, guntur(dt), a.p ,india. He received the B.Tech. degree in Electronics & Communications Engineering from JNTU, Hyderabad in 2010 and pursuing M.Tech (VLSI) in K L University. His research interests include Analog VLSI Design, Digital VLSI Design and Low Power Memory Design and Fault Diagnosis.