International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024
https://ijgst.com.2024.v13.i2.pp1051-1060

Functional Constraints in the Selection of Gate-Exhaustive Two-Cycle Faults for Test Generation using LFSR Automatic test pattern generator (ATPG)

Mr.D.Satyanarayana [(1)] , Dr.A.Ranganayakulu [(2)] , Itha Manikanta [(3),] Syed Nagoor Basha [(4),] Rudrapati Ashok Kumar [(5),] Pothuluri Kashi Sainath [(6),]

[1,2] *Krishna Chaithanya Institute Of Technology & Sciences, Ece Department, Markapur, Andhra Pradesh.*
[3,4,5,6,] *Krishna Chaithanya Institute Of Technology & Sciences, UG Student-ECE, Markapur, Andhra Pradesh.*

Abstract In this project, there is a complete model for delay defects restricted to gates (cells or subcircuits) in two-cycle gate-exhaustive faults. Nonetheless, there may be an excessive amount of two-cycle gate-exhaustive errors. Cell layout analysis is used to identify flaws that are crucial to find to obtain cell-aware faults. This project offers an alternative perspective on errors that should be found. According to this perspective, it's critical to identify a two-cycle gate-exhaustive problem if it could interfere with the circuit's ability to function. Functional broadside tests can be used to find these kinds of errors. The article describes a test generation procedure that extracts test cubes from functional broadside tests, merges the test cubes into tests, and derives both broadside and skewed-load tests from the resulting test data. This produces unconstrained two-cycle tests, which are more compact and detect more faults than functional broadside tests. Iterative steps enable a progressive rise in both the number of tests and the number of two-cycle gate-exhaustive defects found. The tradeoffs investigated by the approach are illustrated by experimental results for benchmark circuits.

Keywords: LFSR,ATPG,5G Communications, FPGA, Verilog HDL, Xilinx Tool.

I.INTRODUCTION

Two-cycle gate-exhaustive faults model delay defects that are localized to gates (cells or subcircuits) [1]–[6]. A two-cycle gate-exhaustive fault fi for a gate Gi is illustrated by Fig. 1. The fault is associated with two input patterns, pi,0 and pi,1, of Gi . The input patterns are such that a transition from pi,0 to pi,1 creates a transition on the output of the gate. The output transition is denoted by zi,0 → zi,1 in Fig. 1. The fault is denoted by fi = (Gi, pi,0, pi,1). To detect the gate-exhaustive fault fi , a two-cycle test needs to assign pi,0 to the inputs of Gi under the first cycle, pi,1 to the inputs of Gi under the second cycle, and propagate the fault effect zi,1/zi,0 to an observable output during the second cycle. The set of all the two-cycle gate-exhaustive faults provides a comprehensive model for delay defects associated with the gate. However, a gate with n inputs has up to 22n two-cycle gate-exhaustive faults, and the number of two-cycle gate-exhaustive faults for a circuit can be excessive. This is especially true when there are gates with large numbers of inputs.
Defect-aware and cell-aware faults are obtained by performing layout analysis to select faults that are likely to occur and are thus important to detect [7]–[10]. In the case of cell-aware faults, the faults are associated with standard cells. Considering the cells as gates, the set of cell-aware faults is a subset of the set of gate-exhaustive faults. This

addresses the need to select a subset of the two-cycle gate-exhaustive faults as targets for test generation. Other possible selection criteria can be based on the efficient identification of undetectable faults, and the selection of potentially detectable faults as targets for test generation.

Without performing a layout analysis, or an analysis of undetectable faults, this article suggests a different view of two-cycle gate-exhaustive faults that are important to detect. The goal is to select a subset of two-cycle gate-exhaustive faults as targets for test generation and generate an efficient test set for them. Under the view suggested in this article, a two-cycle gate-exhaustive fault is important to detect if it can affect the circuit during functional operation [11]–[13]. Such faults can be identified by generating functional broadside tests [14]–[18]. The functional constraints captured by functional broadside tests limit the fault coverage, in general, and the number of two-cycle gate-exhaustive faults that can be detected in particular. This provides a set of detectable target faults of a more manageable size. The fact that the faults can be detected by functional broadside tests implies that they can affect the circuit during functional operation, making them important to detect.

Compared with an unconstrained test set, a functional broadside test set is not compact, and it detects fewer faults. To address these limitations of functional broadside tests, it is possible to select two-cycle gate-exhaustive faults by generating functional broadside tests and then apply unconstrained test generation to the selected faults. If the computational effort is manageable, it is also possible to perform test generation for all the two-cycle gate-exhaustive faults, select a subset of the faults as suggested in this article, and then truncate or otherwise adjust the test set to detect the selected faults. However, such a process cannot assess during the selection of two-cycle gate-exhaustive faults how many faults need to be selected for an unconstrained test set of manageable size, and how many two-cycle gate-exhaustive faults will be detected when unconstrained tests are used. To address these issues, the procedure described in this article selects two-cycle gate-exhaustive faults gradually. In addition, instead of repeating unconstrained test generation for increasing subsets of target faults, or repeatedly adjusting a test set that targets all the two-cycle gate exhaustive faults, the test generation procedure suggested in this article replaces functional broadside tests with unconstrained tests gradually to detect the selected faults by extracting and merging test cubes.

The procedure is applied to the set F of all the two-cycle gate-exhaustive faults (excluding only faults for which the single-cycle faults are undetectable, and adding transition faults, as discussed in Section VI). Instead, the procedure can be applied to a set of cell-aware faults. In addition, efficient identification of undetectable two-cycle gate-exhaustive faults can be applied to remove undetectable faults from F. In this case, the procedure from Fig. 2 will magnify the set of faults that can be detected during functional operation with cell-aware faults that are likely to occur. The discussion in this article considers the set of all the two-cycle gate-exhaustive faults to demonstrate the full extent of the procedure.

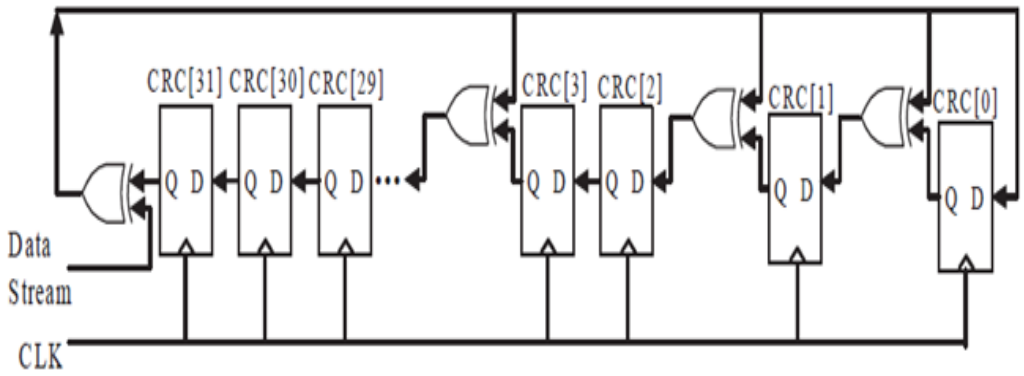
2. FUNCTIONAL BROADSIDE TESTS

Functional broadside tests can be generated by one of several procedures [14]–[18]. For the discussion in this article, functional

https://ijgst.com.2024.v13.i2.pp1051-1060

broadside tests are extracted from functional test sequences [15]. A large pool of functional test sequences is assumed to be available. This is the case when functional test sequences are used for design verification, or the sequences can be generated by a low-complexity sequential test generation procedure [18]. Given a functional test sequence Vj = Vj(0)Vj(1)... Vj(L − 1) of length L, suppose that Vj takes the circuit through the sequence of states S j(0)S j(1)... S j(L), where S j(0) is the initial state of the circuit during functional operation. Two consecutive clock cycles of Vj provide a two-cycle snapshot of functional operation. Considering clock cycles u and u+1, where $0 \leq u < L-1$, a functional broadside test tj,u = S j(u), Vj(u), Vj(u+1)can be obtained, having scan-in state S j(u), and two primary input vectors, Vj(u) and Vj(u+1), that are applied in two consecutive functional capture cycles after S j(u) is scanned in. The final state of the test is scanned out.

A functional broadside test considered in this article has two different primary input vectors. Unconstrained tests are also allowed to have different primary input vectors. Instead, it is possible to generate functional broadside tests with equal primary input vectors or impose this condition during the generation of unconstrained tests. A functional test sequence Vj of length L yields L − 1 functional broadside tests tj,u, for $0 \leq u < L - 1$. In iteration $i \geq 1$ of the procedure from Fig. 2, the set of target two-cycle gate-exhaustive faults is denoted by Fi . The procedure extracts functional broadside tests from functional test sequences Vi,0, Vi,1, ... targeting the faults in Fi , until one of three conditions is satisfied.



(1) Let A be the number of transition faults. For a constant $0 < \alpha < 1$, the set Si of functional broadside tests detects α faults from Fi . Stopping the computation of functional broadside tests in iteration i with a limit on the number of detected two-cycle gateexhaustive faults ensures that the number of tests, and the number of detected two-cycle gate-exhaustive faults, increase gradually. It is thus possible to stop the procedure when the number of tests or the number of detected faults is sufficiently large. (2) The total number of detected two-cycle gate-exhaustive faults reaches β, for a constant $\beta \geq 1$. In this case, the procedure terminates at the end of the iteration. (3) The total number of functional test sequences considered in all the iterations reaches a

constant γ. This limit is important for limiting the computational effort. The procedure terminates at the end of the iteration when this limit is reached.

II.EXISTING SYSTEM AND PROPOSED SYSTEM

1.EXISTING SYSTEM

A Linear Feedback Shift Register (LFSR) is a type of shift register where the input bit is a linear function of its previous state. It's commonly used in various digital systems for tasks such as pseudorandom number generation, error detection and correction, and cryptographic algorithms.

Here's how a conventional LFSR works:

1. Structure: An LFSR consists of a shift register, which is a sequence of flip-flops (binary storage elements), and feedback logic that determines the input to the first flip-flop based on the current state of the register.
2. Initialization: At the beginning, the register is loaded with an initial state, often called the seed. This seed can be any non-zero value.
3. Clocking: At each clock cycle, the contents of the register are shifted to the right (or left) by one position. The bit that is shifted out of the register (the least significant bit or the most significant bit, depending on the implementation) becomes the output of the LFSR.
4. Feedback: The output of the LFSR is fed back into the input of the register through a feedback function. This feedback function typically XORs (exclusive OR) certain bits of the register to produce the input for the first flip-flop.
5. Repetition: The process repeats with each clock cycle, generating a sequence of output bits based on the initial state and the feedback function.

The period of an LFSR is the number of clock cycles it takes for the register to return to its initial state. The maximum period of an n-bit LFSR is 2^n-1, and it occurs when the LFSR is in a maximal-length state sequence, meaning it cycles through all possible non-zero states before returning to the initial state.

LFSRs find applications in various areas, including:

1. Pseudorandom Number Generation: LFSRs are often used to generate sequences of pseudorandom bits, which are useful in simulation, testing, and cryptographic algorithms.
2. Error Detection and Correction: In communication systems, LFSRs are used for error detection and correction codes like CRC (Cyclic Redundancy Check).
3. Stream Ciphers: In cryptography, LFSRs are used in stream ciphers to generate a pseudorandom keystream for encrypting plaintext.
4. Built-in Self-Test (BIST): LFSRs are used in built-in self-test circuits to generate test patterns for testing digital circuits.

Overall, conventional LFSRs are versatile and widely used in digital systems due to their simplicity, efficiency, and effectiveness in various applications.

2.PROPOSED SYSTEM: ATPG LFSR

International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024
https://ijgst.com.2024.v13.i2.pp1051-1060

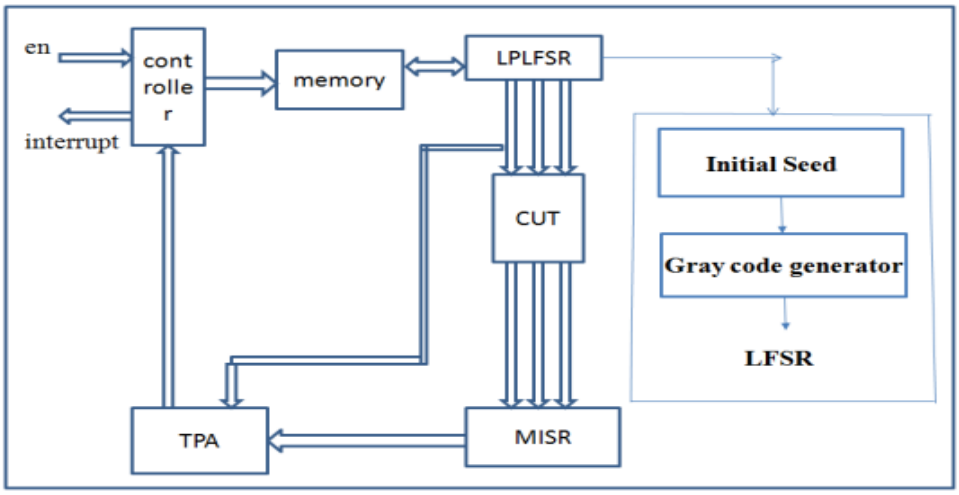


Fig2. Low power ATPG system

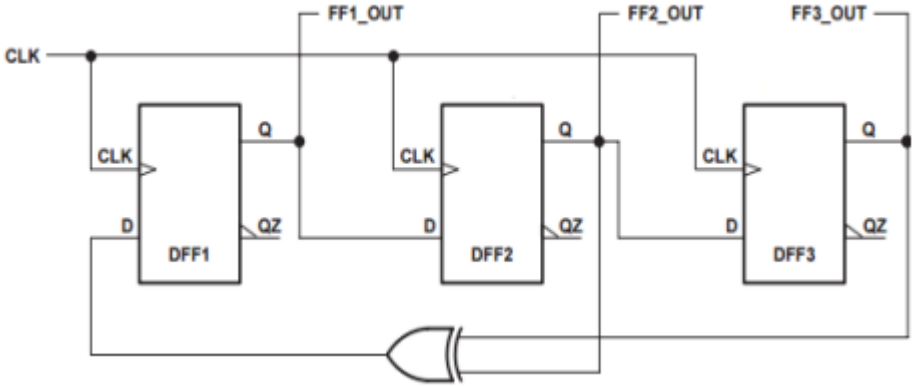


Fig3. LFSR Circuit

Creating a detailed block diagram for LFSR-based ATPG involves illustrating the key components and their interconnections. Below is a simplified block diagram to help you understand the major elements involved in LFSR-based ATPG:

Key Components:

Test Pattern Generator:

Responsible for generating test vectors that will be applied to the circuit under test (CUT).

Linear Feedback Shift Register (LFSR):

The core component that generates pseudo-random bit sequences. The contents of the shift register are shifted left at each clock cycle.

Feedback Logic:

Comprises XOR gates that calculate the feedback bit based on selected bits from the LFSR. This feedback influences the randomness and periodicity of the generated sequences.

Clock Generator:

Generates clock signals to control the shifting of bits in the LFSR.

Test Vector Generation:

Uses the output of the LFSR to generate test vectors. These vectors are applied to the CUT to detect faults.

International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024
https://ijgst.com.2024.v13.i2.pp1051-1060

Circuit Under Test (CUT):
The digital circuit that is being tested for faults. Test vectors generated by the ATPG are applied to the CUT, and the responses are analyzed to detect faults.

Theoretical calculations of CRC using 8-bit generator polynomial for the given 8-bit data “1011 1011” is shown in bellow table

Data	Present state	Next state
1	xxxx xxxx	1111 1111
0	1111 1111	0111 1111
1	0111 1111	1011 0011
1	1011 0011	0101 1001
1	0101 1001	0010 1100
0	0010 1100	1001 1010
1	1001 1010	0100 1101
1	0100 1101	0010 0110 (CRC)

Table 2.1

In the above table first column represents the serial data inputs, second column represents the Present state for every single bit data and the last column is to represent the next state values for the same data. In the first row and first column for the “1” data present state value is “1111 1111” and next state value is “0111 1111” for the given polynomial x8 + x5 + x4 + 1 Using above LFSR circuit. Similarly we can also find the present state, next state values for the data”011 1011” and the next state value for the last bit is theoretical CRC for the given bit sequence. In the above case the CRC is “0010 0110” or “2C”16.

III. RESULTS AND ANALYSIS DISCUSSION

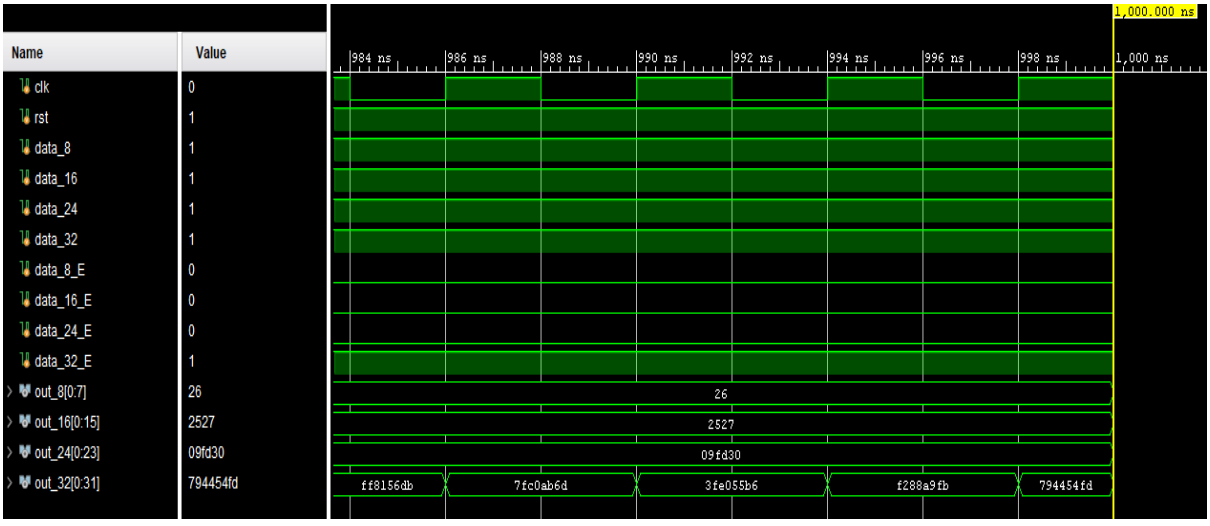


FIG1.SIMULATION OUTPUT OF ATPG LFSR

International Journal of Gender, Science and Technology
ISSN: 2040-0748

UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024

https://ijgst.com.2024.v13.i2.pp1051-1060

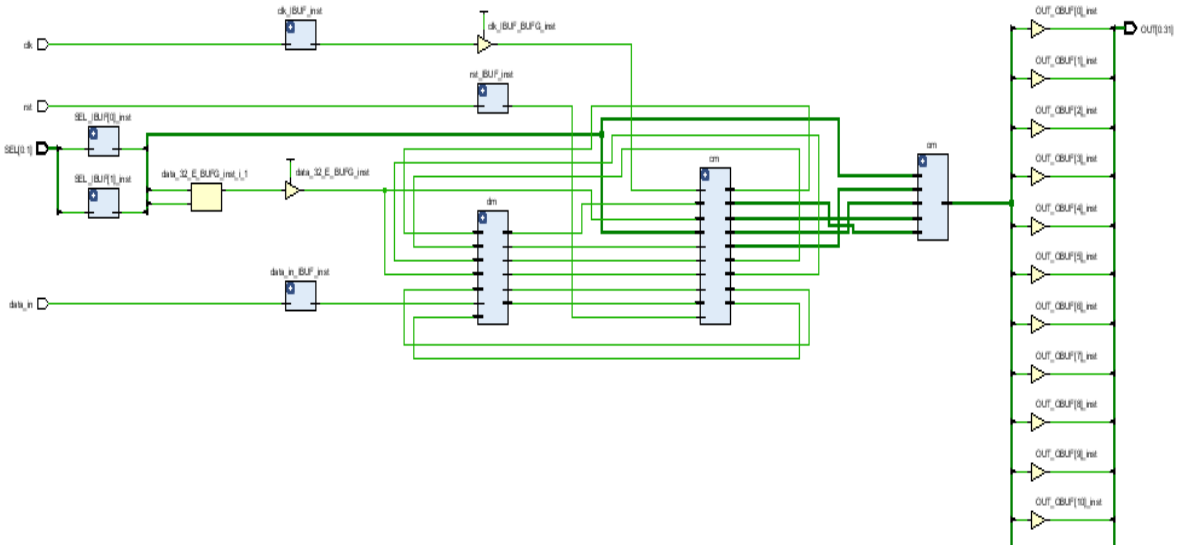


FIG2.SYNTHESIZED DESIGN OF ATPG LFSR

Resource	Utilization	Available	Utilization %
LUT	50	117120	0.04
FF	164	234240	0.07
IO	37	204	18.14
BUFG	2	352	0.57

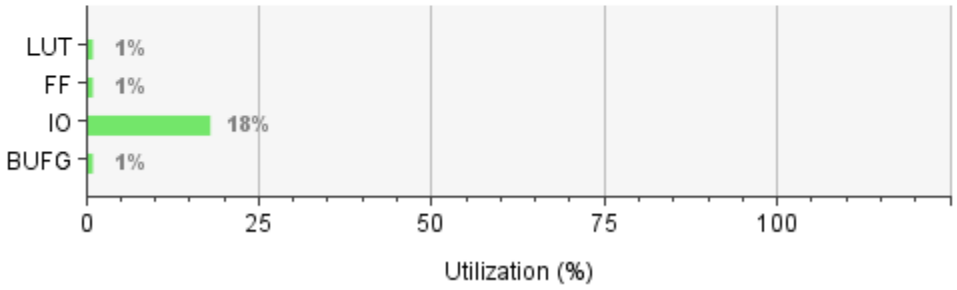


FIG3.UTILIZATION OF LUTS FFS AND IO BUFFERS DESIGN OF ATPG LFSR

International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024
https://ijgst.com.2024.v13.i2.pp1051-1060

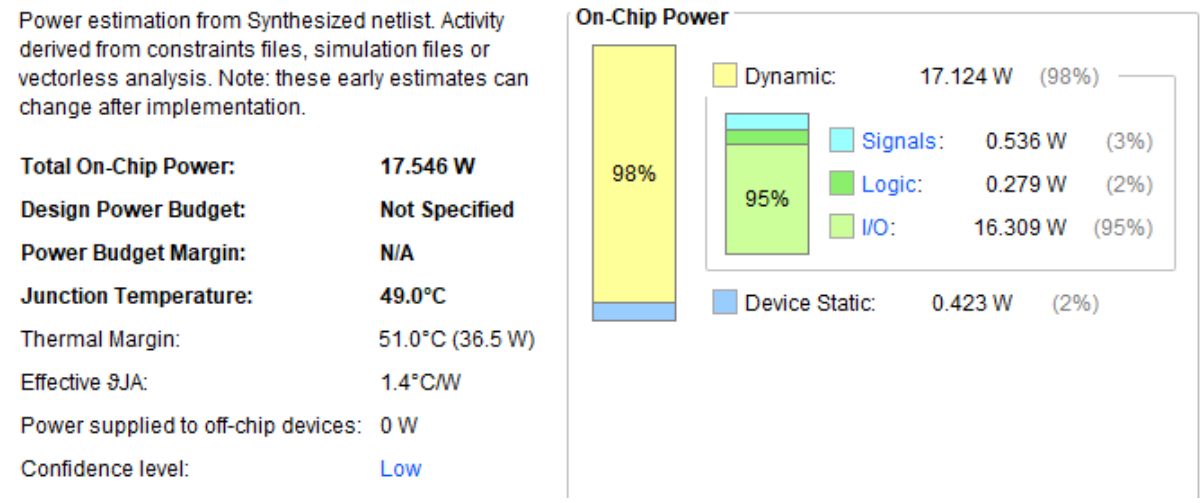


FIG4. DESIGN OF POWER REPORT ATPG LFSR

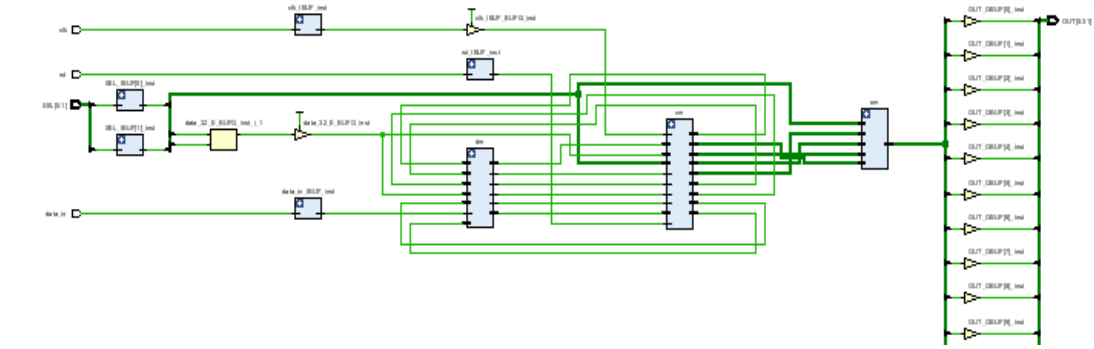


FIG4. OPTIMIZED DESIGN OF ATPG LFSR

CONCLUSION

In conclusion, Automatic Test Pattern Generation (ATPG) utilizing Linear Feedback Shift Registers (LFSRs) represents a vital component in the design, testing, and fault diagnosis of digital circuits. The utilization of LFSRs in ATPG offers several advantages, including simplicity, efficiency, and effectiveness in generating test patterns for detecting faults.

LFSR-based ATPG techniques have proven to be instrumental in various applications, ranging from built-in self-test (BIST) for manufacturing testing to functional testing and fault diagnosis in digital designs. Their versatility allows them to adapt to different

fault models, testing requirements, and emerging technologies.

Looking towards the future, the scope of ATPG using LFSRs remains promising. There are several avenues for further advancement, including the development of advanced fault models, optimization of efficiency, adaptation to emerging technologies, integration with machine learning, addressing security testing requirements, and catering to the unique challenges of IoT and cyber-physical systems.

3.REFERENCES

[1] E. J. McCluskey, "Quality and single-stuck faults," in Proc. IEEE Int. Test Conf. (ITC), Oct. 1993, p. 597.
[2] R. D. Blanton and J. P. Hayes, "Properties of the input pattern fault model," in Proc. Int. Conf. Comput. Design VLSI Comput. Processors, Oct. 1997, pp. 372–380.
[3] K. Y. Cho, S. Mitra, and E. J. McCluskey, "Gate exhaustive testing," in Proc. IEEE Int. Conf. Test, Nov. 2005, pp. 1–7.
[4] R. Guo, S. Mitra, E. Amyeen, J. Lee, S. Sivaraj, and S. Venkataraman, "Evaluation of test metrics: Stuck-at, bridge coverage estimate and gate exhaustive," in Proc. 24th IEEE VLSI Test Symp., Apr. 2006, pp. 66–71.
[5] A. Jas, S. Natarajan, and S. Patil, "The region-exhaustive fault model," in Proc. 16th Asian Test Symp. (ATS), Oct. 2007, pp. 13–18.
[6] I. Pomeranz, "Maximal independent fault set for gate-exhaustive faults," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 40, no. 3, pp. 598–602, Mar. 2021.
[7] D. Kim, M. E. Amyeen, S. Venkataraman, I. Pomeranz, S. Basumallick, and B. Landau, "Testing for systematic defects based on DFM guidelines," in Proc. IEEE Int. Test Conf., Oct. 2007, pp. 1–10.
[8] F. Hapke et al., "Cell-aware test," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 33, no. 9, pp. 1396–1409, Sep. 2014.
[9] A. Sinha, S. Pandey, A. Singhal, A. Sanyal, and A. Schmaltz, "DFMaware fault model and ATPG for intra-cell and inter-cell defects," in Proc. IEEE Int. Test Conf. (ITC), Oct. 2017, pp. 1–10.
[10] Y.-H. Huang et al., "Methodology of generating dual-cell-aware tests," in Proc. IEEE 35th VLSI Test Symp. (VTS), Apr. 2017, pp. 1–6.
[11] J. Rearick, "Too much delay fault coverage is a bad thing," in Proc. Int. Test Conf., Nov. 2001, pp. 624–633.
[12] J. Saxena et al., "A case study of ir-drop in structured at-speed testing," in Proc. Int. Test Conf. (ITC), Sep. 2003, pp. 1098–1104.
[13] S. Sde-Paz and E. Salomon, "Frequency and power correlation between at-speed scan and functional tests," in Proc. IEEE Int. Test Conf., Oct. 2008, pp. 1–9.
[14] I. Pomeranz, "On the generation of scan-based test sets with reachable states for testing under functional operation conditions," in Proc. 41st Annu. Conf. Design Automat. (DAC), Jun. 2004, pp. 928–933.
[15] I. Pomeranz and S. M. Reddy, "Generation of functional broadside tests for transition faults," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 25, no. 10, pp. 2207–2218, Oct. 2006.
[16] M. Valka et al., "A functional power evaluation flow for defining test power limits during at-speed delay testing," in

Proc. 16th IEEE Eur. Test Symp., May 2011, pp. 153–158.
[17] A. Touati et al., “Exploring the impact of functional test programs reused for power-aware testing,” in Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE), Mar. 2015, pp. 1277–1280.
[18] I. Pomeranz, “Static test compaction procedure for large pools of multicycle functional broadside tests,” IET Comput. Digit. Techn., vol. 12, no. 5, pp. 233–240, Sep. 2018.
[19] A. H. El-Maleh and Y. E. Osais, “Test vector decomposition-based static compaction algorithms for combinational circuits,” ACM Trans. Design Autom. Electron. Syst., vol. 8, no. 4, pp. 430–459, Oct. 2003.
[20] I. Pomeranz, “Low-power test generation by merging of functional broadside test cubes,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 7, pp. 1570–1582, Jul. 2014.
[21] I. Pomeranz, “Skewed-load test cubes based on functional broadside tests for a low-power test set,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 3, pp. 593–597, Mar. 2015.
[22] I. 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.
[23] J. Savir and S. Patil, “Scan-based transition test,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 12, no. 8, pp. 1232–1241, Aug. 1993.