

Structure Design and Optimization of 2-D LFSR-Based Multisequence Test Generator in Built-In Self-Test

Xinhui Zhang, Chien-In Henry Chen, and Arvindkumar Chakravarthy

Abstract—This paper addresses the optimization of very large scale integration testing systems, specifically the structure design and optimization of a built-in self-test (BIST) design based on two-dimensional (2-D) linear feedback shift registers (LFSRs). The 2-D LFSRs can generate both precomputed test patterns (for detecting random-pattern-resistant faults) and random patterns (for detecting random-pattern-detectable faults) and have the advantages of high fault coverage and at-speed testing. To guarantee solutions, it is necessary and desirable to generate subsequences of the precomputed test patterns through the 2-D LFSRs, where these subsequences retain the order of the test patterns, particularly for testing sequential circuits. For the design and optimization of the 2-D LFSRs, the following two problems need to be solved: 1) the good partitioning of the precomputed test patterns into disjoint subsequences in order to achieve a minimal hardware and 2) the structure design and optimization of the 2-D LFSRs to generate the test patterns in each partitioned subsequence. The optimization of the 2-D LFSRs is modeled as an integer program (a logic optimization model) that determines the coefficients of the recursive Boolean equations that govern the generation of the test patterns. For a sequence of the test patterns, this model finds the minimal-hardware implementation of the 2-D LFSRs. This logic optimization model can be applied to both test-per-scan (serial BIST) and test-per-clock (parallel BIST). This paper presents how this model is embedded in a heuristic framework to partition the test patterns into subsequences from the configurable 2-D LFSRs. The testing hardware is small as the configurable architecture allows the tester to incrementally generate the precomputed test patterns by modification to the feedback of the 2-D LFSRs. Results of benchmark circuits show that significant hardware reduction and higher fault coverage are achieved. The resulting multisequence test generator is a regular structure and is easy to implement. The logic optimization model is applicable to both completely and partially specified test patterns and can be adopted for other LFSR-based structure design and optimization.

Index Terms—Built-in self-test (BIST), deterministic test patterns, linear feedback shift registers (LFSRs), random-pattern-detectable faults, random-pattern-resistant faults, random test patterns, recursive Boolean equations, test-per-clock, test-per-scan.

Manuscript received August 17, 2006; revised September 30, 2007.

X. Zhang is with the Department of Industrial Engineering, Wright State University, Dayton, OH 45435 USA.

C.-I. H. Chen is with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435 USA.

A. Chakravarthy is with the Department of Industrial Engineering, Wright State University, Dayton, OH 45435 USA, and also with the Information Systems Department, Honda of America Manufacturing, Inc., Marysville, OH 43040 USA.

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TIM.2007.911707

I. INTRODUCTION

THE increasing demands for high-density and high-performance integrated circuits dictate the built-in self-test (BIST) schemes to guarantee high fault coverage, which is expected to be produced by a simple test-pattern generator in an acceptable number of vectors. The BIST involves performing the test-vector generation and the output-response analysis on a chip through the built-in hardware. The test-generation techniques in the BIST include pseudorandom testing [1], [2], pseudoexhaustive testing [3]–[5], weighted random testing using linear feedback shift registers (LFSRs) [6]–[8], reseeding of the LFSRs [9]–[11], cellular automata [12]–[15], and counter-based approaches [24]–[28]. The problem with these techniques is that they generally do not provide a high enough fault coverage due to the presence of random-pattern-resistant faults. The existing solution to this problem is either to insert test points [16] or to modify the test-pattern generator to be a scan-based operation in order to generate desirable test patterns to detect the random-pattern-resistant faults [16]–[18]. The test-point insertion requires an extensive modification of the circuit and degrades circuit performance. Modifying the test-pattern generator with scan operation requires large amounts of hardware that cannot perform at-speed testing.

LFSRs are commonly used as test-pattern generators because the generated sequence of test patterns has good randomness properties. However, a conventional LFSR cannot produce a sequence of deterministic ordered vectors that are needed to detect random-pattern-resistant faults [19]. One way to overcome this problem and to improve the fault coverage is to add logic to embed a set of precomputed test patterns to detect the random-pattern-resistant faults within a short time. These patterns are obtained by automatic test-pattern-generation tools and can be stored in a ROM or generated through software. However, neither of these approaches can perform at-speed testing in BIST.

A test-generator scheme based on the configurable 2-D LFSRs was proposed in [20]. This configurable 2-D LFSR test generator generates the following: 1) a sequence of deterministic test patterns to detect random-pattern-resistant faults and 2) a sequence of random test patterns to detect random-pattern-detectable faults. To guarantee solutions, it is desirable to generate subsequences of the deterministic test patterns through the 2-D LFSRs where these subsequences retain the order of the test patterns. For the design and optimization of the 2-D LFSRs, the following two problems need to be solved: 1) the partitioning of the precomputed test patterns into disjoint

TABLE I
SIXTEEN PATTERNS TO BE EMBEDDED

Seq	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	0	1	0	0	0	0	1	1	1	0	0	0	0	1	0
2	0	1	1	0	1	0	0	1	1	0	0	0	0	0	1	0
3	1	1	1	1	0	0	0	1	0	0	1	1	0	0	1	1
4	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	0
5	0	0	1	1	1	0	1	0	0	1	0	1	0	1	1	1
6	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0	1

subsequences for a minimal hardware of the 2-D LFSRs and 2) the structure design and optimization of the 2-D LFSRs to generate the test patterns of each partitioned subsequence.

To illustrate the aforementioned problems in the design and optimization of 2-D LFSRs, consider, for example, the 2-D LFSRs of [20, ex. 1], which are named “example1.” The circuit contains 16 test patterns to be embedded, which are shown in Table I.

The first attempt is to generate all the test patterns at once; however, this requires at least two flip-flop (FF) stages, as shown in Fig. 1, and a hardware area of $871.2 \mu\text{m}^2$ in a 130-nm CMOS process. To further reduce the hardware, it would be necessary to partition the test patterns into subsequences. Different partitions result in different configurations and implementations of 2-D LFSRs.

For example, a particular partitioning of the 16 test patterns into three subsequences will result in one stage of FF array (FFA), as shown in Fig. 2, where CN1 generates the first subsequence of the six patterns (1–6), CN2 generates the next subsequence of the six patterns (7–12), and CN3 generates the last subsequence of the four patterns (13–16). This optimized structure has a hardware area of $617.76 \mu\text{m}^2$, which accounts for almost a 30% reduction in hardware.

It is pointed out that the problem of partitioning the test patterns into subsequences and determining the optimal implementation of 2-D LFSRs is nondeterministic polynomial-time complete. As we will show, even to find a solution to generate a sequence of the test patterns through LFSRs requires the solution of a set of nonlinear Boolean equations. To achieve optimization of the 2-D LFSRs with minimal hardware, both problems have to be solved, and their solutions pose a great challenge to the design of the LFSR-based BIST.

This paper presents a systematic approach to the structure design of 2-D LFSRs through mathematical programming and optimization. The kernel of the system is an integer program model that finds a solution to determine the coefficients in the recursive Boolean equations and by which the generation of test patterns in a 2-D LFSR is governed. For a sequence of the test patterns, this model provides minimal hardware of the 2-D LFSRs. The model is then embedded in a heuristic to partition the test patterns into disjoint subsequences for an optimal configurable 2-D LFSR design. Results of benchmark circuits show that a significant reduction of hardware is achieved. Compared with the previous work, for the test-per-scan BIST, a hardware reduction of as much as 83% can be achieved.

The remainder of the paper is organized as follows. Section II presents LFSRs and 2-D LFSRs; it introduces the optimization

problem of designing these testing structures. A mathematical model to optimize the 2-D LFSRs, as well as its mathematical property, is presented in Section III. A heuristic framework to partition the test patterns into a subsequence is given in Section IV. The experimental results of benchmark circuits are given in Section V. A 2-D LFSR design-automation framework and the synthesis of the benchmark circuits are presented in Section VI. The logic optimization model in Section III can be extended to find the optimal values of don't-care bits in the partially specified test patterns. A logic optimization model for the partially specified test patterns is presented in Section VII. Our conclusion and future research are outlined in Section VIII.

II. STRUCTURE DESIGN AND OPTIMIZATION OF 2-D LFSRS

A. Linear Feedback Shift Registers (LFSRs)

LFSRs are commonly used as test-pattern generators because they can produce a large number of pseudorandom patterns with good randomness properties with a small hardware. Fig. 3 shows the structure of conventional LFSRs, which is comprised of EXCLUSIVE-OR (XOR) gate and M FFs.

The output V_1 of LFSRs is governed by the following Boolean equation:

$$V_1 = C_1 V_1 D^1 \oplus C_2 V_1 D^2 \oplus \dots \oplus C_M V_1 D^M \quad (1)$$

where $V_1 D^i$, $i = 1, \dots, M$ is the i th delay of FF, and C_i , $i = 1, \dots, M$ represents whether the $V_1 D^i$ is connected to the XOR gate or not. Coefficient C_i 's take the values of either one (connected) or zero (not connected) and determine the structure of the LFSR.

When triggered by clock signals, the output V_1 is shifted to the right, and the LFSR outputs periodic patterns with a maximal length of $2^M - 1$, where M is the number of FFs. This repeatability suggests that the patterns generated by the LFSRs are pseudorandom in nature and that the random properties depend on its initial state and the previous generating equation. Note that the mathematical symbol \oplus , instead of $+$, is used in the Boolean equations that will be used in our optimization model. Conventional LFSRs cannot produce a sequence of deterministic ordered patterns that are needed to detect random-pattern-resistant faults. One way to overcome this problem is to embed a set of precomputed test patterns in 2-D LFSRs.

B. Two-Dimensional Linear Feedback Shift Registers (2-D LFSRs)

Fig. 4 shows the structure of 2-D LFSRs with N LFSRs. Each LFSR has M stages of FFs, inverters, and XOR gates. Let V_i , $i = 1, \dots, N$ be the row vector $\langle 01 \dots \rangle$ that needs to be generated from the i th LFSR, and let $V_i D^k$, $k = 1, \dots, M$ represent the k th delay of vector V_i . Then, for a 2-D LFSR, the generation of V_i through the feedback network is governed by the following Boolean equation:

$$V_i = \left(\bigoplus_{j=1}^N \bigoplus_{k=1}^M C_{ijk} V_j D^k \right) \oplus B_i \quad \forall i = 1 \dots N. \quad (2)$$

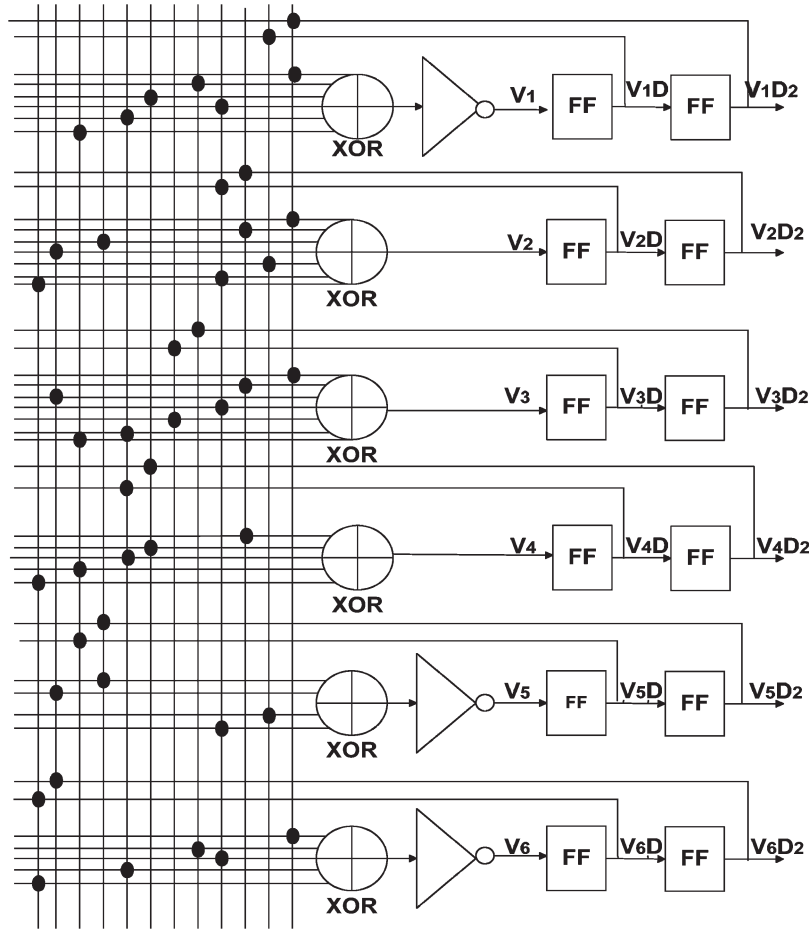


Fig. 1. Two-dimensional LFSRs of [20, ex. 1].

If $C_{ijk} = 1$, the signal $V_j D^k$ is connected to the XOR gate to generate V_i . If $B_i = 1$, an inverter is added to the input of the shift register.

The aforementioned equation can also be written in a different form. To do so, let $P = \{P_1, P_2, \dots, P_S\}$ be the set of deterministic ordered test patterns to be generated by the 2-D LFSR, each with N elements. Furthermore, let p_{si} be the i th element in pattern s , as in

$$\begin{aligned} & P_1, P_2, P_3, \dots, P_S \\ V_1 & : p_{11}, p_{21}, p_{31}, \dots, p_{S1} \\ V_2 & : p_{12}, p_{22}, p_{32}, \dots, p_{S2} \\ & \vdots \\ V_N & : p_{1N}, p_{2N}, p_{3N}, \dots, p_{SN}. \end{aligned}$$

Then, the Boolean (2) can be rewritten in terms of the elements in the test patterns as follows:

$$p_{si} = \left(\bigoplus_{J=1}^N \bigoplus_{K=1}^M C_{ijk} p_{s-k,j} \right) \oplus B_i \quad \forall i = 1, \dots, N, s = 1, \dots, S, \text{ and } k < s. \quad (3)$$

Equations (2) and (3) are recursive in nature and are therefore called the recursive Boolean equations. These equations govern the generation of the test patterns. The solution of these

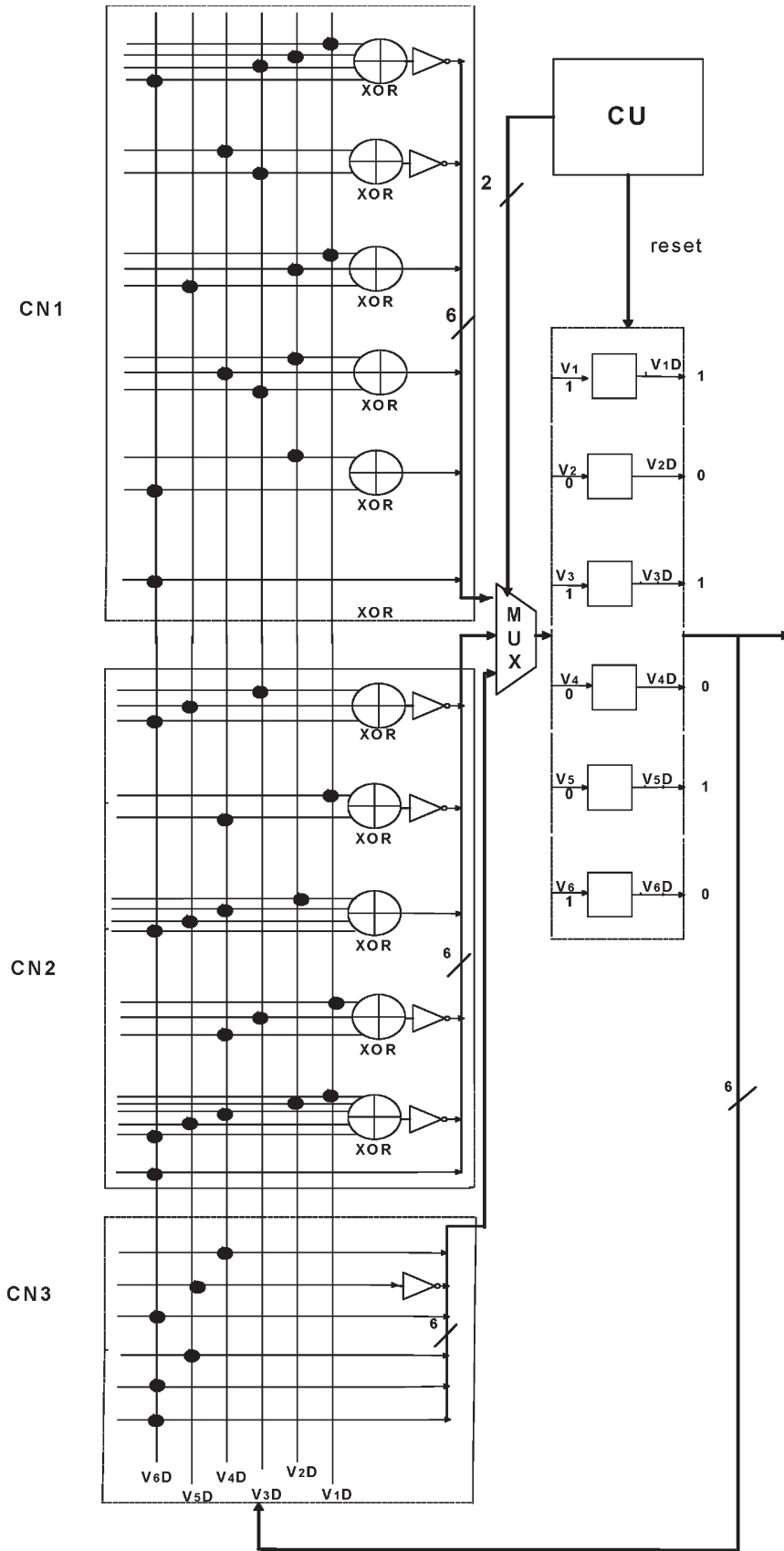
equations (determining the coefficients of these equations) determines the structure design of 2-D LFSRs.

C. Synthesis and Optimization of 2-D LFSRs

The synthesis procedure of 2-D LFSRs is comprised of two steps: 1) generating a set of deterministic ordered test patterns through the 2-D LFSRs to detect the random-pattern-resistant faults and 2) optimizing the test structure of the 2-D LFSRs with a minimal hardware. Given a set of precomputed test patterns, the design of the 2-D LFSRs to generate these patterns is determined by the coefficients in the recursive Boolean equations.

An ideal case is that a set of values for the coefficients C_{ijk} and B_i in (3) exists, and all the test patterns can be generated once without being partitioned into subsequences. If such a solution exists, the test patterns are directly applied to the circuit under test (CUT). However, such solutions are not always possible if the set of patterns S or the vector size N is large. In this situation, the test patterns are partitioned into subsequences of patterns and generated by the 2-D LFSRs using a reconfigurable architecture.

A configurable test generator based on the 2-D LFSRs was proposed in [20]. This configurable test generator generates the following: 1) a deterministic sequence of test patterns for random-pattern-resistant faults and 2) random test patterns



Note: CN denotes configuration network.

Fig. 2. New configurable 2-D LFSRs of example 1 using optimization.

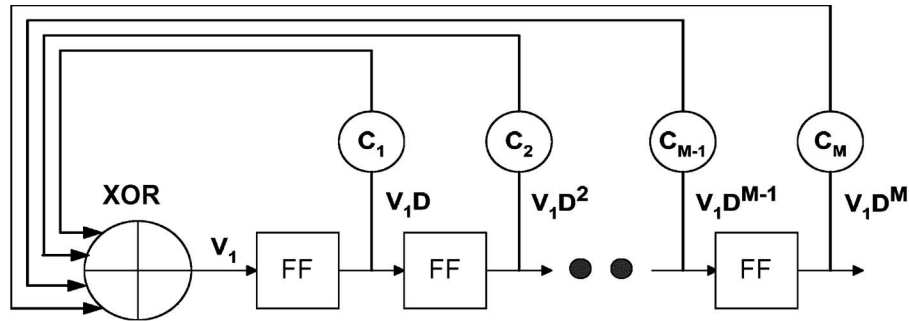


Fig. 3. Conventional LFSRs.

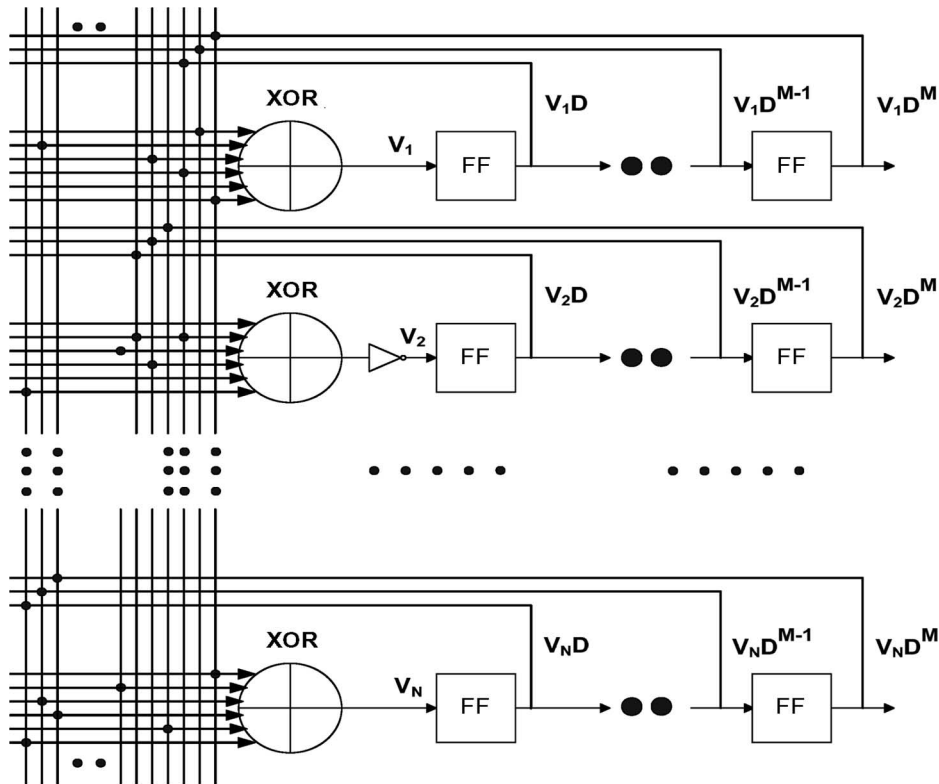


Fig. 4. Structure of 2-D LFSRs.

for random-pattern-detectable faults. The structure of the 2-D LFSRs mainly consists of four function blocks—the FFA, configuration network (CN), multiplexers (MUX), and control unit (CU). They are shown in Fig. 5.

In Fig. 5, the FFA is an $N \times M$ FFA, where N is the number of inputs of a CUT, and M is the number of stages of the 2-D LFSR. The number of stages (M) in the 2-D LFSRs should be kept as small as possible. If the coefficients in (3) cannot be solved with a small value M , then the original set of test patterns is partitioned into subsequences. Each sequence of the test patterns is sequentially generated by the 2-D LFSR. To further reduce the hardware, each of these 2-D LFSRs has its own CN but shares the M -stage FFA. A CN consists of XOR gates and an inverter if necessary and accepts the feedback connections from the FFA. The MUX, which is controlled by CU, selects one of the CNs to feed the feedback signals to the FFA. The CU also controls the setting of the initial states of the FFA.

The optimal design of 2-D LFSR-based testing structure requires solutions to the following two problems: 1) the partitioning of the precomputed test patterns into subsequences and 2) the structure design and optimization of the 2-D LFSRs to generate the test patterns of each subsequence. The problem can be stated as follows.

Given a set of precomputed test patterns $P = \{P_1, P_2, P_3, \dots, P_L\}$, each of which being N bit wide to be embedded, the optimal design of 2-D LFSR is to find a minimum hardware implementation to generate these patterns through each of the subsequences S_1, S_2, \dots, S_i , and $P = \sum S_i$. The configurable 2-D LFSR optimization problem is formulated to the following three objectives.

- 1) The number of FF stage of 2-D LFSR ($M = \max\{M_i, \forall i = 1, \dots, s\}$) is minimized.
- 2) For each subsequence S_i with M stages, the additional hardware (XOR and inverter gates) is minimized.
- 3) The total hardware of 1) and 2) is minimized.

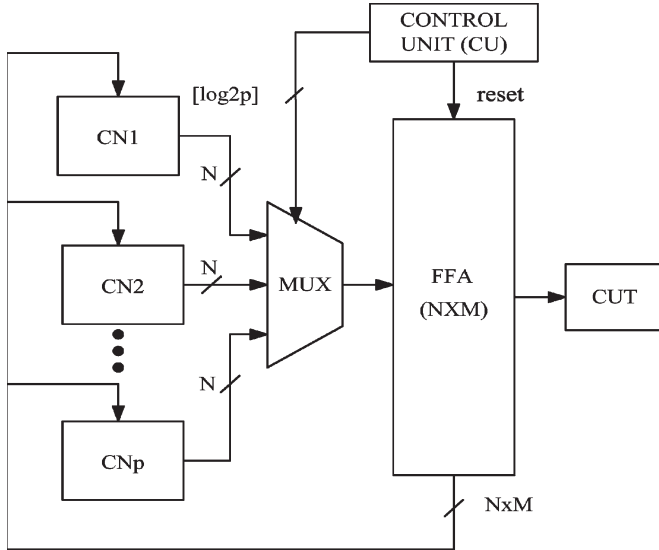


Fig. 5. Configurable 2-D LFSRs.

III. LOGIC OPTIMIZATION MODEL

The main objective of the structure design of 2-D LFSRs is to find the optimal 2-D LFSRs to generate a precalculated subsequence of test patterns S with the least number of M stages of FFA. This is equivalent to determining the coefficients C_{ijk} and B_i in the recursive Boolean (3), with an objective of minimizing the CN which is comprised of inverters and XOR gates.

A. Determination of Coefficients in Recursive Boolean Equations

Assume that a set of S test patterns is to be generated by 2-D LFSRs with N bits and M FF stages (see Fig. 4). Let p_{sl} be the s th element in pattern l , and let w_1 and w_2 be the hardware areas of a two-input XOR gate and an inverter, respectively. The problem of determining the coefficients in the recursive Boolean equations is restated as follows:

$$\text{Minimize } \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^M w_1 x_{ijk} + \sum_{i=1}^N w_2 y_i \quad (4)$$

$$\text{subject to } p_{si} = \left(\bigoplus_{j=1}^N \bigoplus_{k=1}^M x_{ijk} p_{s-k,j} \right) \oplus y_i$$

$$\forall i = 1, \dots, N, s = 1, \dots, S, \text{ and } k < s. \quad (5)$$

The objective is to minimize the respective hardware areas of inverters and XOR gates. The variable x_{ijk} corresponds to C_{ijk} in (3), which represents if a feedback from bit j of FF stage k is connected to the input of the XOR gate at bit i . The variable y_i corresponds to B_i in (3), which represents if an inverter is present at the output of the XOR gate at bit i . These coefficients specify the Boolean equations and also determine the configuration of the 2-D LFSR. Constraint (5) states that the recursive Boolean equations governing the generation of the S test vectors must be satisfied.

B. Linear Integer Program Model

The model [in (4) and (5)] is not a mathematical model due to the existence of the Boolean operation XOR (\oplus). To transform this model into a mathematical program, which is solvable, such as an integer program, the following notations are used.

Indices

i, j Indices of bits $1, \dots, N$.

k Index of FF stages $1, \dots, M$.

s Index of vectors in a subsequence ($s = 1, \dots, S$).

Parameters

p_{si} Value (zero or one) of bit i in vector sequence s ($i = 1, \dots, N$, and $s = 1, \dots, S$).

w_1, w_2 Hardware areas of a two-input XOR gate and an inverter, respectively.

Decision variables

x_{ijk} Binary variable that represents if a feedback from bit j of FF stage k is connected to the input of the XOR gate at bit i ($i = 1, \dots, N$, $j = 1, \dots, N$, and $k = 1, \dots, M$).

y_i Binary variable that represents if an inverter is present at the output of the XOR gate at bit i ($i = 1, \dots, N$).

ν_{si} Number of nonzero inputs to the XOR gates to generate bit i of vector s ($i = 1, \dots, N$, and $s = 1, \dots, S$).

h_{si} Smallest integer that is greater than or equal to $(1/2)\nu_{si}$; equivalently, $h_{si} = \lceil (1/2)\nu_{si} \rceil$ ($i = 1, \dots, N$, and $s = 1, \dots, S$).

The mathematical model is then formulated as follows. The objective [see (6)] is to minimize the hardware areas of inverters and XOR gates.

$$\text{Minimize } w_1 \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^M x_{ijk} + w_2 \sum_{i=1}^N y_i \quad (6)$$

subject to

$$\sum_{j=1}^N \sum_{k=1}^M x_{ijk} p_{s-k,j} + y_i = \nu_{si}$$

$$\forall i = 1, \dots, N, s = 1, \dots, S, \text{ and } k < s \quad (7)$$

$$h_{si} \geq 0.5\nu_{si}$$

$$\forall i = 1, \dots, N, s = 1, \dots, S \quad (8)$$

$$h_{si} \leq 0.5\nu_{si} + 0.5$$

$$\forall i = 1, \dots, N, s = 1, \dots, S \quad (9)$$

$$2h_{si} - \nu_{si} = p_{si}$$

$$\forall i = 1, \dots, N, s = 1, \dots, S \quad (10)$$

$$h_{si} \geq 0 \text{ and integer}$$

$$\forall i = 1, \dots, N, s = 1, \dots, S \quad (11)$$

$$x_{ijk}, y_i \geq 0 \text{ and binary}$$

$$\forall i = 1, \dots, N, s = 1, \dots, S. \quad (12)$$

Representation of XOR Operations: Constraint (7) records the number of inputs to the XOR gate from all feedback plus the presence of an inverter to form bit i of vector s . This summation value is denoted as ν_{si} . The presence of an inverter is equivalent to adding an input of one to the XOR gate. The output of XOR of these inputs can be summarized as follows.

- 1) If ν_{si} is an odd number, then the output of the XOR is one.
- 2) If ν_{si} is an even number, then the output of the XOR is zero.

To represent if ν_{si} is an even number, a new variable h_{si} is introduced and defined as $h_{si} = \lceil 1/2\nu_{si} \rceil$, which is the smallest integer that is greater than or equal to $(1/2)\nu_{si}$. This is reinforced in constraints (8) and (9). For example, if $\nu_{si} = 3$, then $h_{si} \geq 0.5\nu_{si} = 1.5$, and $h_{si} \leq 0.5\nu_{si} + 0.5 = 2$. Because h_{si} has to be an integer, then $h_{si} = 2$. If $\nu_{si} = 4$, then $h_{si} \geq 0.5\nu_{si} = 2$, and $h_{si} \leq 0.5\nu_{si} + 0.5 = 2.5$. Following the same integrity requirement, $h_{si} = 2$.

The outcome of these XOR operations can thus be defined as $p_{si} = 2h_{si}\nu_{si}$. To illustrate, using the same examples, if $\nu_{si} = 3$ (an odd number), then $p_{si} = 2h_{si} - \nu_{si} = 2 \times 2 - 3 = 1$. If $\nu_{si} = 4$ (an even number), then $p_{si} = 2h_{si} - \nu_{si} = 2 \times 2 - 4 = 0$. This is exactly the outcome of the XOR operations of the ν_{si} nonzero inputs previously defined in 1) and 2).

Furthermore, p_{si} is the value at bit i in the next vector s , and this is described in (10). Finally, (11) and (12) specify that all variables must be integers. These constraints dictate that the aforementioned model is an integer program. This integer program model [see (6)–(12)] is the kernel of the 2-D LFSR optimization and is referred as a logic optimization model in this paper.

C. Model Analysis and Properties

1) *Model Size and Complexity*: The aforesaid logic optimization model can be decomposed by bit operations; thus, the model can be solved 1 b at a time. For a specific bit i , the model contains $4S$ constraints and $1(y_i) + 2S(\nu_{si} \text{ and } h_{si}) + NM(x_{ijk})$ variables. Of these variables, y_i and x_{ijk} are variables of either zero or one; ν_{si} and h_{si} are variables of integer. To comprehend the computational size of the problem, consider the AMD microcontroller (am2910) as an example where a precomputed sequence of 30 test patterns is to be embedded in a 2-D LFSR, and each pattern has 20 b. Using the logic optimization model for $M = 2$, each bit has 120 constraints and 101 variables. This model has to be solved 20 times: one for each bit.

The model, although seemingly small, can extremely be difficult to solve. In fact, it will be shown in Section V that even finding the feasible solution may require long computational time when a branch and bound algorithm and the default setting to solve the integer program [21] are used.

2) *Selection of Initial Vectors*: In the solution of this model, M initial vectors are set as the initial states of the FFA. The selection of seed vectors will further reduce the hardware overhead. Specifically, if the first M test vectors to be generated are selected as initial states, the hardware overhead is minimal due to the following proposition.

Proposition 1: Using the first M deterministic sequence of test patterns as the initial states of the FFA gives the minimum number of feedback for any M stages of the FFA.

The total number of test patterns will be reduced from S to $S - M$ if the first M patterns are selected as the initial states of the FFA. Mathematically, this is equivalent to changing the in-

dex from $s = 1, \dots, S$ to $s = M + 1, \dots, S$ in all constraints. The reduced model provides a relaxation of the original model, and its solution is the lower bound of the original model; thus, it is optimal. However, it needs to be mentioned that setting the initial states of the FFA to the first M patterns is likely to require an additional control logic, incurring more cost, and is therefore not always preferred.

IV. PARTITIONING TEST PATTERNS FOR CONFIGURABLE 2-D LFSRS

For a configurable 2-D LFSR-based test scheme, an additional yet important problem that needs to be addressed is the partitioning of the test patterns. To understand the complexity of this problem, first, recall that the hardware cost is composed of three parts: FFA shared by all CNs, XOR gates, and inverters of each CN. Let w_3 be the hardware area of an FF; hence, the total area W of the configurable 2-D LFSRs can be represented as

$$W = \sum_{k=1}^K \left(w_1 \sum_{i=1}^N \sum_{j=1}^N \sum_{m=1}^{M_k} x_{ijm} + w_2 \sum_{i=1}^N y_i \right) + M \times N \times w_3 \quad (13)$$

where K is the number of subsequences, and M_k is the minimal number of FF stages required to generate a subsequence k and an $M = \max(M_k, \text{where } k = 1, \dots, K)$ —the largest of the M_k that will be shared by all subsequences. Equation (13) is a nonlinear equation.

It is pointed out that the number of FF stages (M) determines the complexity of the hardware and should be kept as small as possible. However, if M is set too small, it will result in many partitioned subsequences (i.e., increase CNs, XOR gates, and inverters). As a result, the overall hardware increases. In view of this, the following heuristic is developed.

The algorithm starts with one FF stage ($M = 1$) and iteratively find a partition of L vectors into K subsequences. The partition starts with the test vector s (initially, $s = 1$), adds the test vector one at a time to the subsequence, solves the logic optimization model for the added test vector, and repeats this process. The process stops until a feasible solution to the logic optimization model does not exist, for example, the addition of a new vector e to the subsequence fails. At this time, the test patterns from the vector s to the vector $e - 1$ comprise a subsequence. The algorithm then starts with the test vector $s = e$ for another partition of the subsequence and proceeds until $e = L$ and all the test vectors are partitioned into different subsequences. Once the algorithm completes the partition for $M = 1$, it then continues with $M = 2$ and $s = 1$ until the increment of M will not lead to a solution with minimal hardware.

Step 1) Set $\bar{M} = L, \bar{W} = +\infty$; /* L : the number of output patterns, thus, the maximum number of stages*/

Step 2) For $M = 1, \dots, \bar{M}$ /*Partition with M FF Stages*/

Step 2.a) Set $s = l, e = 2$ and $K = 1$ /* s : start pattern, e : end pattern, K : number of subsequences*/

Step 2.b) Do the following until $e = L$ /*every vector is included*/

Step 2.c) For every bit $n = 1$ to N /*every bit is being optimized*/

Step 2.c.1) Solve model (6)–(12) with subsequence from s to e for bit n

Step 2.c.2) If the solution exists, set $flag = true$

Step 2.c.3) Otherwise, set $flag = false$; break to 2.e);

Step 2.d) End For

Step 2.e) If $flag = true$, the logic optimization model is feasible to all bits, update $e = e + 1$; /*Add one more vector in the subsequence*/

Step 2.f) Otherwise, set $K = K + 1$, $s = e - 1$, $e = e/K$: number of subsequences; s : start pattern, e : end pattern*/ Restart a new subsequence */

Step 2.g) End Do

Step 2.h) Calculate W of (13), Set $\overline{W} = \min\{\overline{W}, W\}$ /* the minimum hardware so far*/

Step 2.i) Set $\overline{M} = \lfloor \overline{W}/N \times w_3 \rfloor$ /* update \overline{M} , the maximum number of subsequences*/

Step 3) End For

Notice that, for a given configuration of a hardware (\overline{W}), as shown in (13), the optimal configuration with the hardware cost is $W = \sum_{k=1}^K (w_1 \sum_{i=1}^N \sum_{j=1}^N \sum_{m=1}^{M_k} x_{ijm} + w_2 \sum_{i=1}^N y_i) + M \times N \times w_3 \leq \overline{W}$. In other words, $M \times N \times w_3 \leq \overline{W}$. Thus, $\overline{M} \leq \lfloor \overline{W}/N \times w_3 \rfloor$, which is an upper bound of the number of FF stages (M) in Step 2.h).

Consider the benchmark circuit (am2910), which has 30 test patterns to be embedded, with each pattern being 20 b wide. Here, the hardware areas of the two-input XOR, inverter, and 1-b FF in a 130-nm CMOS process are used in calculating the hardware cost, where $w_1 = 11.52 \mu\text{m}^2$, $w_2 = 4.32 \mu\text{m}^2$, and $w_3 = 38.88 \mu\text{m}^2$. The algorithm starts with one FF stage ($M = 1$). In order to generate the minimum number of subsequences, the algorithm starts from the first vector ($s = 1$) and attempts to add one at a time ($e = e + 1$) as many test vectors into the subsequence as possible. It stops until the solution of logic optimization model becomes infeasible with the addition of a new vector. As for the case of the am2910, the addition of the 21st test vector ($e = 21$) to the subsequence makes the solution of the logic optimization model infeasible, which results in maximal 20 test patterns (1–20) for the first subsequence. The algorithm then uses the vector 21 as the seed and proceeds in a similar way to partition as many following vectors to the second subsequence as possible. This results in the second subsequence of ten test patterns (21–30). The 2-D LFSRs generating these two subsequences (1–21 and 21–30) of the test patterns contain 224 XOR feedbacks, 19 inverters, and 20 FFs. It consumes an overall hardware (hardware area) of $3428.64 \mu\text{m}^2$. Now, since $\overline{W} = 3428.64$ and, in step 2.h), $\overline{M} \leq 3$ (an upper bound of the number of FF stages in 2-D LFSR), the algorithm precedes the partitioning of the test patterns for the 2-D LFSRs with two stages of FFA. For $M = 2$, all the test patterns can be partitioned into one subsequence. The 2-D LFSRs to generate one subsequence (1–30) of the test patterns contains 158 XOR feedbacks, nine inverters, and 40 FFs and consumes an overall hardware area of $3402.02 \mu\text{m}^2$. The solution with $M = 3$ did

TABLE II
DIFFERENT PARTITIONS USING ONE STAGE AND TWO SUBSEQUENCES FOR AM2910

Partitions	XOR feedback	Inverters	Hardware area (μm^2)
1-22, 22-30	--	--	--
1-21, 21-30	224	19	3428.64
1-20, 20-30	202	20	3179.52
1-19, 19-30	184	18	2963.52
1-18, 18-30	183	17	2947.68
1-17, 17-30	174	19	2852.64
1-16, 16-30	176	19	2875.68
1-15, 15-30	185	17	2970.72
1-14, 14-30	182	18	2940.48
1-13, 13-30	194	23	3100.32
1-12, 12-30	--	--	--

not provide any better results and is, hence, not reported. The solution with $M = 2$ provides the best result and is reported as the final solution.

It has to be mentioned that the aforesaid procedure is based on a greedy algorithm, and there is no guarantee that the obtained solution is optimal. Consider the am2910 as an example. Let us examine all partitions with one FF stage and two subsequences (1– i and i –30) by varying i . From the heuristic procedure, the first subsequence is 1–21; apparently, for any $i < 21$, the subsequence vectors 1– i remain feasible. The smallest value of i can be determined by the second sequence (i –30) because when i gets smaller, the second sequence gets larger. In view of this, the value of i is set to the smallest value so that the second subsequence i –30 is feasible. For this example, the smallest value of i turns out to be 13, and it results in nine feasible partitions (1–21, 21–30), ..., (1–13, 13–30). These partitions and their optimized hardware are shown in the Table II.

The best partition (1–17, 17–30) contains 174 XOR feedbacks and 19 inverters with an area of $2852.64 \mu\text{m}^2$, whereas the worst one (1–21, 21–30) contains 224 XOR feedbacks and 19 inverters with an area of $3428.64 \mu\text{m}^2$. The latter is almost 20% larger in hardware area than the obtained best solution.

V. EXPERIMENTAL RESULTS ON BENCHMARK CIRCUITS

A comprehensive set of experiments was conducted using the benchmark circuits to evaluate the performance of the logic optimization model and the configuration structure of the 2-D LFSR scheme. The logic optimization model was implemented using Mosel and solved with Xpress provided by the DASH Optimization [22]. Xpress is a state-of-the-art linear and integer programming software that uses a branch-and-bound algorithm

TABLE III
BENCHMARK CIRCUITS

Circuit	Bits(N)	Patt. (S)	DFF	Gates
am2910	20	30	87	937
div16	33	21	50	856
mul16	18	34	55	648
pcont2	9	7	24	4168
piir8	9	19	56	11364
c6288	32	8	0	2416
c7552	207	28	0	3512
s5378	35	31	179	2779
s9234	35	22	211	5597
s15850	14	41	534	9772
s13207	62	74	638	7853
s35932	35	19	1728	16065
s38417	12	27	1638	22179
s38484	12	30	1454	19253

to solve the integer program. The default settings were chosen.

A. Benchmark Circuits and Precomputed Test Patterns

Table III shows the benchmark circuits and the characteristics of the precomputed test patterns, including the number of bits (N) and the number of patterns (S). The test cases are classified into three groups. The first group consists of five synthesized benchmark circuits and was used in [20]. The second group consists of two combinational benchmark circuits, and the third group consists of seven sequential benchmark circuits. The second and third groups were selected from [23] of a core-based system-on-a-chip CUT.

B. Nonconfigurable Test-Per-Scan 2-D LFSR

The logic optimization model can also be used to solve a test-per-scan 2-D LFSR design without any modification. The partition algorithm discussed in Section IV will start with $M = 1$ and solve the optimization model [see (6)–(12)] with all the included patterns. If the model is feasible, then a feasible serial 2-D LFSR is obtained. Otherwise, the procedure continues with $M = M + 1$ until a feasible solution is obtained. This scheme is called a nonconfigurable 2-D LFSR.

Table IV reports the experimental results of the nonconfigurable 2-D LFSR design for each benchmark circuit. The second and third columns in the table list the number of FF stages and FFs in the FFA. The fourth and fifth columns list the numbers of XOR feedbacks and inverters. The hardware area, which is computed according to (13), is reported in the sixth column. The seventh column reports the total computation time. Finally, for comparison, the eighth column lists the number of FF stages in [20].

As these results show, a significant reduction of hardware is achieved when compared with the results reported in [20]. As for the sample am2910, previously, results require a 13-stage FFA; however, using the proposed optimization method, a two-

TABLE IV
COMPUTATION RESULTS FOR NONCONFIGURABLE 2-D LFSR

Benchmark Circuit	FF stages	Flip-flops	XOR feedback	Inverters	Hardware area (μm^2)	Comp. time (sec)	FF stages [20]
Example1 [20]	2	12	34	3	871.20	1.3	2
am2910	2	40	158	9	3,402.72	2,744.5	13
div16	1	33	188	17	3,510.72	32.3	14
mul16	--	--	--	--	--	--	10
pcont2	1	9	18	1	550.08	0.4	2
piir8	3	27	44	1	1,549.44	2.8	3
c6288	1	32	54	6	1,880.64	2.4	N/A
c7552	1	207	892	52	18,537.12	272,521.8	N/A
s5378	--	--	--	--	--	--	N/A
s9234	2	70	149	8	4,461.12	846.7	N/A
s13207	--	--	--	--	--	--	N/A
s15850	--	--	--	--	--	--	N/A
s35932	1	35	172	12	3,382.56	199.8	N/A
s38417	3	36	88	2	2,410.56	1711.1	N/A
s38484	3	36	104	4	2,603.52	31,053.0	N/A

Note: "--" denotes no feasible solution is obtained in a reasonable amount of computation time.

stage FFA is sufficient. This result is constantly across most of the test cases—"am2910," "div16," and "pcont2"—in the first group. For the benchmark circuits in the second and third groups, the optimal structure design required a small number of FF stages, suggesting that small hardware is needed.

Although these results are promising, there exist several benchmark circuits where no feasible solutions are obtained in a reasonable amount of computation time. While part of this problem is due to the lack of advanced algorithms to solve these optimization problems faster, it is also possible that no feasible design exists for these circuits using a small number of FF stages.

C. Configurable Test-Per-Clock 2-D LFSRs

Table V shows the experimental results of the configurable test-per-clock 2-D LFSRs for benchmark circuits. In a configurable 2-D LFSR, the test patterns are divided into a number (the second column) of subsequences (the exact partition is shown in the last column). Each subsequence corresponds to a 2-D LFSR structure and shares the same FFA. For all configurable 2-D LFSRs, only one FF stage is needed and is thus not reported in Table V.

From Table V, it is seen that, in all cases, setting $M = 1$, one FF stage is adequate for generating a test structure. For many cases (shown in bold), the hardware cost of the configurable 2-D LFSRs is much less than that of the nonconfigurable 2-D LFSRs; the hardware has been reduced by 31.82% for s9234, 29.66% for s38417, 26.17% for s38484, 61.56% for piir8, and 49.17% for example 1. For other circuits, both the configurable and nonconfigurable 2-D LFSRs achieve about same results with one sequence and one FF stage. It should be mentioned that the resulting 2-D LFSR-based test generator were able to achieve high fault coverage. These results were similar to that reported in [20] and are thus not included in Tables IV and V.

TABLE V
COMPUTATION RESULTS FOR CONFIGURABLE 2-D LFSR

Benchmark Circuit					Hardware	Comp.	Sub-
	Flip- Sequences	XOR flops	Inver- feedback	ters	area (μm^2)	Time (sec)	sequences (S)
example1 [20]	3	6	32	8	617.76	9.0	1-6,7-12,13-17
am2910	2	20	224	19	3428.64	75.5	1-21,21-30
div16	1	33	188	17	3510.72	32.3	1-21
mul16	3	18	217	21	3278.88	54.91	1-17,17-32,32-34
Pcont2	1	9	18	1	550.08	0.4	1-7
Piir8	3	9	52	5	959.04	14.8	1-9,9-15,15-19
c6288	1	32	54	6	1880.64	2.4	1-8
c7552	1	207	892	52	18537.12	272521.8	1-28
s5378	2	35	325	26	5205.60	1202.64	1-29,29-31
s9234	2	35	171	15	3384	77.3	1-15,15-22
s13207	4	62	998	63	14168.16	86726.33	1-28,28-53,53-58,58-74
s15850	3	14	226	24	3240	55.7	1-16,16-31,31-41
s35932	1	35	172	12	3382.56	199.8	1-19
s38417	3	12	117	13	1859.04	11.8	1-14,14-25,25-27
s38484	3	12	134	15	2063.52	13.8	1-13,13-21,21-30

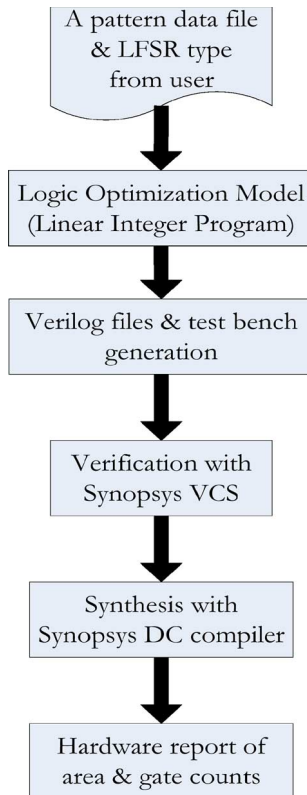


Fig. 6. Two-dimensional LFSR design-automation framework.

VI. TWO-DIMENSIONAL LFSR DESIGN AUTOMATION

A 2-D LFSR design-automation framework is shown in Fig. 6. Given a sequence of predetermined test patterns, users have the choice of selecting either a configurable or a non-configurable 2-D LFSR as the test generator. The test patterns and the selected LFSR type are then analyzed by the linear

integer program of the logic optimization model for hardware minimization. An FFA module in Verilog is generated based on the optimization result, and the number of stages of FFs, say k stages, is also determined. These k stages of FFs are initialized with zeros and ones according to the first k test patterns. For the configurable 2-D LFSRs, the CNs, MUX, and CU, which are in Verilog and needed for the generation of the required test patterns, are generated. Along with these modules in Verilog, a Verilog test bench is also generated for 2-D LFSR design verification. Following it, logic synthesis is performed, and a hardware report of gate counts and area is then presented. Table VI presents the synthesis of 2-D LFSRs for 13 benchmark circuits. Among the 13 benchmark circuits, div16 and mul16 have the least number of gate counts and have a higher BIST hardware. Six benchmark circuits (c6288, pcont2, piir8, s35932, s38417, and s38484) have a BIST hardware less than 3%. It is shown that, overall, the average BIST hardware is about 7.54%, which is considered low compared with most practical BIST designs.

VII. LOGIC OPTIMIZATION MODEL FOR PARTIALLY SPECIFIED TEST PATTERNS

In a sequence of test patterns, there are certain bits that may have no direct impact on the fault detection and can be set to either zero or one. These bits are hence called the don't-care bits. While the don't-care bits would not affect the fault detection, the value assignment to the don't-care bits will have a direct impact on the final form of the LFSR structure design. The traditional approach of arbitrarily specifying the values to the don't-care bits will overconstrain the synthesis of the LFSR structure design and produce a design of more hardware. For example, in Table VII, the bit 5 of the vector 15 is a don't-care bit, which is noted as X. If the bit is set to zero, the hardware of

TABLE VI
 SYNTHESIS OF 2-D LFSRS

Benchmark Circuit	FF		2D LFSRs		CU		2D LFSRs	BIST
	Conf.	Stages	FF	Gates (Mux and CN)	FF	Gates	+ CU	Hardware
							Area (μm^2)	
am2910	2	1	20	172	6	23	2601	1.65%
c6288	1	1	32	58	-	-	1449	0.47%
div16	1	1	33	151	-	-	3081	22.10%
mult16	3	1	18	161	8	42	2811	29.83%
pcont2	1	1	9	18	-	-	585	0.78%
piir8	3	1	9	47	7	43	1059	0.94%
s13207	4	1	62	799	9	54	11010	13.98%
s15850	3	1	14	186	8	53	2706	3.51%
s35932	1	1	35	158	-	-	2928	1.35%
s38417	3	1	12	102	7	41	1758	0.93%
s38484	3	1	12	130	7	37	1914	0.95%
s5378	2	1	35	261	6	31	4065	14.86%
s9234	2	1	35	180	6	34	2841	6.72%
Ave.	-	-	-	-	-	-	-	7.54%

 TABLE VII
 SIXTEEN PATTERNS TO BE EMBEDDED

Seq	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	0	1	0	0	0	0	1	1	1	0	0	0	0	1	0
2	0	1	1	0	1	0	0	1	1	0	0	0	0	0	1	0
3	1	1	1	1	0	0	0	1	0	0	1	1	0	0	1	1
4	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	0
5	0	0	1	1	1	0	1	0	0	1	0	1	0	1	X	1
6	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0	1

2-D LFSRs contains two FF stages. However, if the bit value is set to one, then at least three FF stages are needed to generate the 16 test patterns. The two synthesized 2-D LFSRs are shown in Fig. 7.

The logic optimization model in Section III can be extended to find the optimal values of the don't-care bits in the partially specified test patterns. To do so, let z_{si} be a binary variable representing the value (zero or one) of an unspecified bit i in the partially specified test sequence s ($i = 1, \dots, N$, and $s = 1, \dots, S$). The logic optimization model for the partially specified test patterns is stated as follows:

$$\text{Minimize } w_1 \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^M x_{ijk} + w_2 \sum_{i=1}^N y_i$$

subject to (8)–(12), and

$$\sum_{j=1}^N \sum_{k=1}^M x_{ijk} z_{s-k,j} + y_i = v_{si} \quad \forall i = 1, \dots, N, s = 1, \dots, S, \text{ and } k < s. \quad (14)$$

Here, the constraint (14), replacing the constraint (7) in the previous model, takes the partially specified test patterns as inputs, and calculates the total number of inputs to the XOR gate, from all feedback to form bit i of vector s . Con-

straints (8)–(12) of the previous logic optimization model are same used for this model.

The aforementioned model is a quadratic integer program. To translate it into a linear integer program, let us define $q_{ijks} = x_{ijk} z_{s-k,j}$, and the value is either zero or one. The constraint (14) can be rewritten as

$$\sum_{j=1}^N \sum_{k=1}^M q_{ijks} + y_i = v_{si} \quad \forall i = 1, \dots, N, s = 1, \dots, S. \quad (15)$$

The fact that q_{ijks} equals $x_{ijk} z_{s-k,j}$ can be modeled through the following constraints:

$$q_{ijks} \leq x_{ijk} \quad \forall i = 1, \dots, N, j = 1, \dots, N, k = 1, \dots, M, s = 1, \dots, S \quad (16)$$

$$q_{ijks} \leq z_{s-k,j} \quad \forall i = 1, \dots, N, j = 1, \dots, N, k = 1, \dots, M, s = 1, \dots, S \quad (17)$$

$$q_{ijks} \geq z_{s-k,j} + x_{ijk} - 1 \quad \forall i = 1, \dots, N, j = 1, \dots, N, k = 1, \dots, M, s = 1, \dots, S \quad (18)$$

$$q_{ijks} \geq 0 \text{ and binary} \quad \forall i = 1, \dots, N, j = 1, \dots, N, k = 1, \dots, M, s = 1, \dots, S. \quad (19)$$

Herein, constraints (16) and (17) state that q_{ijks} equals zero if either x_{ijk} or $z_{s-k,j}$ equals zero. Constraint (18) states that q_{ijks} equals one if and only if both x_{ijk} and $z_{s-k,j}$ equal one. These constraints exactly define the product of x_{ijk} and $z_{s-k,j}$

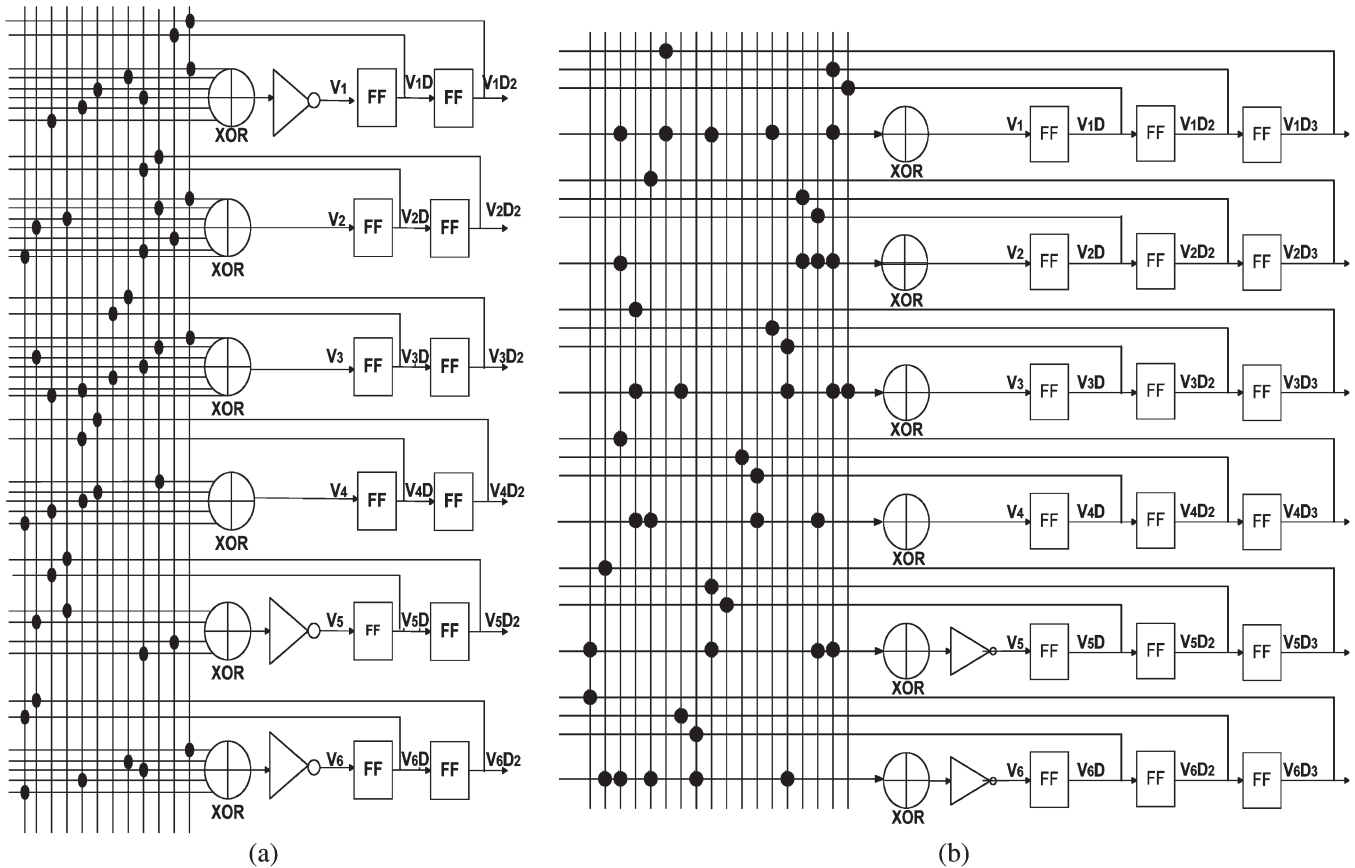


Fig. 7. Two-dimensional LFSRs by setting the don't-care bit to (a) zero and (b) one.

or $q_{ijk s} = x_{ijk} z_{s-k,j}$. As such, the linear logic optimization for partially specified test patterns can be written as

$$\text{Minimize } w_1 \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^M x_{ijk} + w_2 \sum_{i=1}^N y_i$$

subject to (8)–(12), (15), and (16)–(19).

VIII. CONCLUSION

BIST is a collection of possibilities, the choice of which depends on the particular application. Factors to consider include fault coverage and system performance. However, for any BIST structure, the optimal structure design to achieve minimal hardware is desirable. For the LFSR-based test structure, the solution of the test structure depends on the determination of the coefficients in the recursive Boolean equations that govern the generation of test patterns. In this paper, a logic optimization model is proposed to optimize 2-D LFSRs in order to generate a set of precomputed test patterns. A mathematical model was developed to determine the coefficients of a 2-D LFSR. Together with a heuristic algorithm to partition the precomputed test patterns into subsequences, a viable approach to the optimal configurable 2-D LFSR design is presented. The resulting 2-D LFSR-based multisequence test generator is able to generate a given test-vector set at significantly lower hardware compared with a previous work while retaining high fault coverage. It has been shown that the average BIST hardware of

13 benchmark circuits is about 7.54%, which is considered low compared with other BIST designs.

As there has been no systematic approach to solving these Boolean equations, there are thus no existing algorithms to solve the LFSR-based design problem. The nonlinear Boolean equations (2) and (3), whose solution through linear integer program and logic optimization models [(6)–(12)] determines the structure design of 2-D LFSRs, have already demonstrated promising results and were able to handle both completely and partially specified test patterns. Similar methodology and models can be developed to other LFSR-based structure designs. Efficient yet general mathematical programming algorithms through decomposition techniques, bounding procedures, and advanced heuristics to further improve the 2-D LFSR-based design are currently under investigation.

REFERENCES

- [1] P. Bardell, W. H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*. New York: Wiley, 1987.
- [2] K. D. Wagner, C. K. Chin, and E. J. McCluskey, "Pseudorandom testing," *IEEE Trans. Comput.*, vol. C-36, no. 3, pp. 332–343, Mar. 1987.
- [3] C.-I. H. Chen and J. Yuen, "Automated synthesis of pseudo-exhaustive test generator in VLSI BIST design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 3, pp. 273–291, Sep. 1994.
- [4] D. Kagaris, F. Makedon, and S. Tragoudas, "A method for pseudo-exhaustive test pattern generation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 9, pp. 1170–1178, Sep. 1994.
- [5] H.-J. Wunderlich and S. Hellebrand, "The pseudoexhaustive test of sequential circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 1, pp. 26–33, Jan. 1992.

- [6] F. Brglez *et al.*, "Hardware-based weighted random pattern generation for boundary scan," in *Proc. IEEE Int. Test Conf.*, 1989, pp. 264–274.
- [7] R. Kapur, S. Patil, T. J. Sneathen, and T. W. Williams, "Design of an efficient weighted random pattern generation system," in *Proc. Int. Test Conf.*, 1994, pp. 491–500.
- [8] F. Muradali, V. K. Agarwal, and B. Nadeau-Dostie, "A new procedure for weighted random built-in self-test," in *Proc. Int. Test Conf.*, 1990, pp. 660–669.
- [9] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift registers," in *Proc. IEEE Int. Test Conf.*, 1992, pp. 120–129.
- [10] S. Venkataraman, J. Rajski, S. Hellebrand, and S. Tarnick, "An efficient BIST scheme based on reseeding of multiple polynomial linear feedback shift registers," in *Proc. Int. Conf. Comput.-Aided Des.*, 1993, pp. 572–577.
- [11] S. Hellebrand *et al.*, "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 223–233, Feb. 1995.
- [12] P. D. Hortensius, R. D. McLeod, W. Pries, D. Miller, and H. C. Card, "Cellular automata-based pseudorandom number generators for built-in self-test," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 8, no. 8, pp. 842–859, Aug. 1989.
- [13] S. Boubezari and B. Kaminska, "A deterministic built-in self-test generator based on cellular automata structures," *IEEE Trans. Comput.*, vol. 44, no. 6, pp. 805–816, Jun. 1995.
- [14] S. Chiusano, F. Como, P. Prinetto, and M. Sonza Reorda, "Cellular automata for sequential test pattern generation," in *Proc. IEEE VLSI Test Symp.*, Apr. 1997, pp. 60–65.
- [15] F. Fummi and D. Sciuto, "Implicit test pattern generation constrained to cellular automata embedding," in *Proc. IEEE VLSI Test Symp.*, 1997, pp. 54–59.
- [16] G. Kiefer and H.-J. Wunderlich, "Deterministic BIST with multiple scan chains," in *Proc. Int. Test Conf.*, 1998, pp. 1057–1064.
- [17] G. Kiefer, H. Vranken, E. J. Marinissen, and H.-J. Wunderlich, "Application of deterministic logic BIST on industrial circuits," in *Proc. Int. Test Conf.*, 2000, pp. 105–114.
- [18] N. A. Toubia and E. J. McCluskey, "Altering a pseudo-random bit sequence for scan-based BIST," in *Proc. Int. Test Conf.*, 1996, pp. 167–175.
- [19] E. B. Eichelberger and E. Lindbloom, "Random-pattern coverage enhancement and diagnosis for LSSD logic self-test," *IBM J. Res. Develop.*, vol. 27, no. 3, pp. 262–272, May 1983.
- [20] C.-I. H. Chen and K. George, "Configurable two-dimensional linear feedback shifter registers for parallel and serial built-in self-test," *IEEE Trans. Instrum. Meas.*, vol. 53, no. 4, pp. 1005–1014, Aug. 2004.
- [21] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York: Wiley, 1988.
- [22] *Dash Optimization, Xpress-Mosel: User Guide*, Prentice-Hall, Englewood Cliffs, NJ, 2002.
- [23] V. Iyengar and K. Chakrabarty, "Test bus sizing for system-on-a-chip," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 449–459, May 2002.
- [24] I. Hamzaoglu and J. H. Patel, "Reducing test application time for built-in-self-test test pattern generators," in *Proc. VLSI Test Symp.*, Apr. 2000, pp. 369–375.
- [25] N. Kranitis, M. Psarakis, D. Gizopoulos, A. Paschalis, and Y. Zorian, "An effective deterministic BIST scheme for shifter/accumulator pairs in data paths," in *Proc. Int. Symp. Quality Electron. Des.*, Mar. 2001, pp. 343–349.
- [26] H. Liang, M. Yi, X. Fang, and C. Jiang, "A BIST scheme based on selecting state generation of folding counters," in *Proc. Int. Test Symp.*, Dec. 2005, pp. 144–149.
- [27] G. Kiefer and H.-J. Wunderlich, "Using BIST control for pattern generation," in *Proc. Int. Test Symp.*, Nov. 1997, pp. 347–355.
- [28] K. Chakrabarty, B. T. Murray, and V. Iyengar, "Deterministic built-in test pattern generation for high-performance circuits using twisted-ring counters," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 5, pp. 633–636, Oct. 2000.



Xinhui Zhang received the Ph.D. degree in operations research and industrial engineering from the University of Texas at Austin.

He is an Assistant Professor with the Department of Industrial Engineering, Wright State University, Dayton, OH. His research interests are in mathematical programming and optimization, particularly in solving large problems in the field of manufacturing, logistics and transportation, media management, service operations, and engineering design optimization. He has participated in and led several research projects, such as airline crew recovery, advertising allocation for television networks, and equipment scheduling for mail processing facilities. He has published in the *IIE Transactions*, *Computer and Operations Research*, the *European Journal of Operations Research*, and the *Journal of the Operational Research Society*.

Dr. Zhang is an active member of the Institute for Operations Research and Management Sciences. Prior to his career in operations research, he was a Mechanical Engineer and received the outstanding award (No. 2) at the first national students' software competition in China.



Chien-In Henry Chen received the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 1989.

He has been with the Department of Electrical Engineering, Wright State University, Dayton, OH, since 1989. He has also been a Consultant for a number of U.S. semiconductor companies. He has primarily been working on the design and test of digital, mixed-signal ICs and system-on-a-chip, low-power very large scale integration (VLSI) and field-programmable gate arrays for signal processing, communications, global positioning systems, and digital wideband microwave receivers. His research was supported by the Department of Defense, federal agencies, industrial companies, and the state of Ohio. He has written about 100 publications in IEEE journals, international journals, and IEEE conference proceedings.

Dr. Chen has been a Technical Committee Member of the IEEE International ASIC/SOC Conference, the IEEE Instrumentation and Measurement Technology Conference, the Annual Conference of the IEEE Industrial Electronics Society, and the IEEE International Symposium on Circuits and Systems. He was a plenary speaker at the 6th VLSI Design/CAD Symposium and served as a Guest Editor of the *VLSI Design Journal* in 2002.



Arvindkumar Chakravarthy received the B.E. degree in electronics and instrumentation from the University of Madras, Chennai, India, in 1999 and the M.S. degree in electrical engineering from Wright State University, Dayton, OH, in 2002, where he is currently working toward the Ph.D. degree in mathematical modeling and optimization with the Department of Industrial Engineering.

He has worked on optimization problems in very large scale integration testing and optimization, crew scheduling in airlines, and equipment and workforce scheduling in U.S. Postal Service mail-processing facilities. He is also currently with the Information Systems Department, Honda of America Manufacturing, Inc., Marysville, OH, as a Lead Software Engineer.