Novel Test Pattern Generators for Pseudoexhaustive Testing

Rajagopalan Srinivasan, *Member*, *IEEE Computer Society*,
Sandeep K. Gupta, *Member*, *IEEE Computer Society*, and Melvin A. Breuer, *Fellow*, *IEEE*

Abstract—Pseudoexhaustive testing of a combinational circuit involves applying all possible input patterns to all its individual output cones. The testing ensures detection of all detectable multiple stuck-at faults in the circuit and all detectable combinational faults within individual cones. Test pattern generators based on coding theory principles are not tailored to a specific circuit as they do not utilize any structural information. They usually generate test sets that are several orders of magnitude larger than the minimum size pseudoexhaustive test set required for a specific circuit. In this paper, we describe hardware efficient test pattern generators that employ knowledge of the circuit output cone structures for generating minimal test sets. Using our techniques, we have designed generators that generate minimum size test sets for the ISCAS benchmark circuits.

Index Terms—Test pattern generators, linear feedback shift registers, pseudoexhaustive testing.

1 Introduction

Exhaustive testing of a combinational circuit involves exercising the circuit with all possible input patterns. Such testing assumes no fault model and ensures detection of all detectable combinational faults in the circuit. A detectable fault in a combinational circuit is one that can be detected at one or more outputs with a single or multiple input patterns. A combinational fault in a combinational circuit is one that does not cause the circuit to exhibit any sequential behavior and, hence, is detectable with a single input pattern. The test time associated with exhaustive testing increases exponentially with the number of inputs to the circuit. For circuits with a large number of inputs, exhaustive testing is very time consuming and may not be practical.

Pseudoexhaustive testing attempts to reduce the large test time required for exhaustive testing. Pseudoexhaustive testing of a combinational circuit involves exercising each output cone with all possible input patterns. An *output cone* consists of all logic that feed the output. An output is said to *depend* on an input if there exists at least one directed path from that input to the output. The number of inputs on which an output depends is referred to as the *dependency* of that output. The time required for pseudoexhaustive testing is proportional to the dependencies of the outputs. Pseudoexhaustive testing ensures detection of all detectable combinational faults within individual output cones and all detectable multiple stuck-at faults in the circuit.

- *R. Srinivasan is with the Network Communications Group, Intel Corp., 1230 Campus West Dr., Morganville, NJ 07751-1262. E-mail: rajagopalan.srinivasan@intel.com.*
- *S.K. Gupta and M.A. Breuer are with the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089-2562. E-mail: {sandeep, mb}@poisson.usc.edu.*

Manuscript received 4 Mar. 1997; revised 15 Feb. 2000; accepted 10 Apr. 2000.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 104080.

Consider a combinational circuit with n inputs and m outputs. Let k be the maximum value among the dependencies of the m outputs. The circuit can be characterized as an (n, m, k) circuit as only these three parameters—the number of inputs (n), outputs (m), and the maximum dependency (k) among output cones—determine the size of a pseudoexhaustive test set. Exhaustive testing of the circuit requires 2^n patterns which may be prohibitive for large values of n. Pseudoexhaustive testing involves applying exhaustive tests to the m output cones. Generation of optimal pseudoexhaustive test sets for an (n, m, k) circuit is a hard problem. The sizes of the test sets are bounded below and above by 2^k and 2^n, respectively.

Autonomous linear feedback shift registers (LFSRs) are extensively used for generating binary test sequences [1]. LFSRs are characterized by their feedback connections represented as polynomials. For a nonzero initial state, the *period* of an LFSR is the number of states generated prior to repeating the initial state. An n-stage *maximal length* LFSR has a period of $2^n - 1$ states and utilizes a primitive polynomial for its feedback connections. A maximal length LFSR can be enhanced with a nonlinear gate to produce an all-zero state. Pseudoexhaustive test pattern generators (TPGs) are usually based on maximal length LFSRs. TPGs can be classified as either *universal* or *circuit-specific*.

Cellular automata-based [2] TPGs are not considered in this study as their hardware overhead is usually high compared to LFSRs and their feedback connections are usually represented as a characteristic matrix instead of simple polynomials. This study is restricted to external-XOR type LFSR [3] as they naturally form a shift register among successive stages.

Universal TPGs designed for (n, m, k) circuits are applicable for *all circuits with the same values of* n *and* k. They generate test sets containing binary n-tuples that cover all k-dimensional subspaces. In other words, any k columns of the sequence of n-tuples contain all 2^k possible patterns. These test sets can be generated by LFSRs based on constant

0018-9340/00/$10.00 © 2000 IEEE

weight codes [4], linear codes [5], [6], or cyclic codes [7]. Coding theory principles are used for determining the feedback polynomials for the LFSRs. However, for a specific (n, m, k) circuit, a test set of size much smaller than that of the universal test set usually exists.

Circuit-specific TPGs such as LFSR/SRs [8], [9] and LFSR/XORs [10], [11], [12] can be designed for a *given* (n, m, k) circuit by utilizing the knowledge of the circuit output cone structures. An LFSR/SR consists of an LFSR and a shift register (SR) and can be implemented with low hardware overhead. However, the pseudoexhaustive test set generated by LFSR/SR is often significantly greater than the lower bound (2^k). On the other hand, a pseudoexhaustive test set close to the lower bound can be generated by LFSR/XOR. An LFSR/XOR is composed of an LFSR and an XOR network and often incurs high hardware overhead. This paper describes novel circuit-specific TPGs that are efficient both in terms of utilizing hardware and generating minimal test sets. Our TPG designs are based on the principles of LFSR/SRs and LFSR/XORs. The practicality and efficiency of our TPGs are demonstrated on the ISCAS benchmark circuits.

Theoretical upper bounds on pseudoexhaustive test lengths generated by LFSR/SRs and LFSR/XORs are derived in our companion paper [13]. These bounds are sensitive to the ordering of the circuit inputs and we have developed an efficient method to permute the circuit inputs to obtain the best improvement on the output cone-dependent bounds. The quality of these bounds is demonstrated by comparing them to existing bounds [10], [8]. In this paper, we present our TPG designs that generate pseudoexhaustive tests within the tight upper bounds.

The paper is organized as follows: Section 2 deals with the main concepts associated with LFSR/SRs and LFSR/XORs and provides the background and motivation for our work. The characteristics of our TPG designs are described in Sections 3 and 4. Section 5 deals with the TPG design procedure and experimental results are presented in Section 6. The conclusions are presented in the last section.

2 LFSR/SRs and LFSR/XORs

Consider an (n, m, k) circuit with its inputs being driven by an n-stage register. Let the inputs be denoted as $\theta_1, \theta_2, \ldots, \theta_n$ and the register stages be denoted as $s_1, s_2, \ldots, s_n$, respectively. The register is configured as a circuit-specific TPG during the test mode.

The (n, m, k) circuit can be exhaustively tested by configuring the n-stage register as a maximal length LFSR with $P(x)$ as its feedback polynomial. Running the LFSR through its period (and including the all-zero pattern) generates all possible n-bit patterns. The unique sequence of 2^n binary values generated at an individual stage of the LFSR is referred to as a *test signal*. Stage s_i of the register also corresponds to the ith stage ($i = 1, 2, \ldots, n$) of the TPG. The test signal generated by stage s_i is characterized by the *residue* r_i of that stage [8]. In general, the residue r_i is computed as $(r'_i \times x)\ mod\ P(x)$, where r'_i is the residue of the test signal feeding the input of stage s_i. For an external LFSR, since stage s_{i-1} directly feeds stage s_i, the residue r_i is given by $(r_{i-1} \times x)\ mod\ P(x)$. The residue r_1 representing

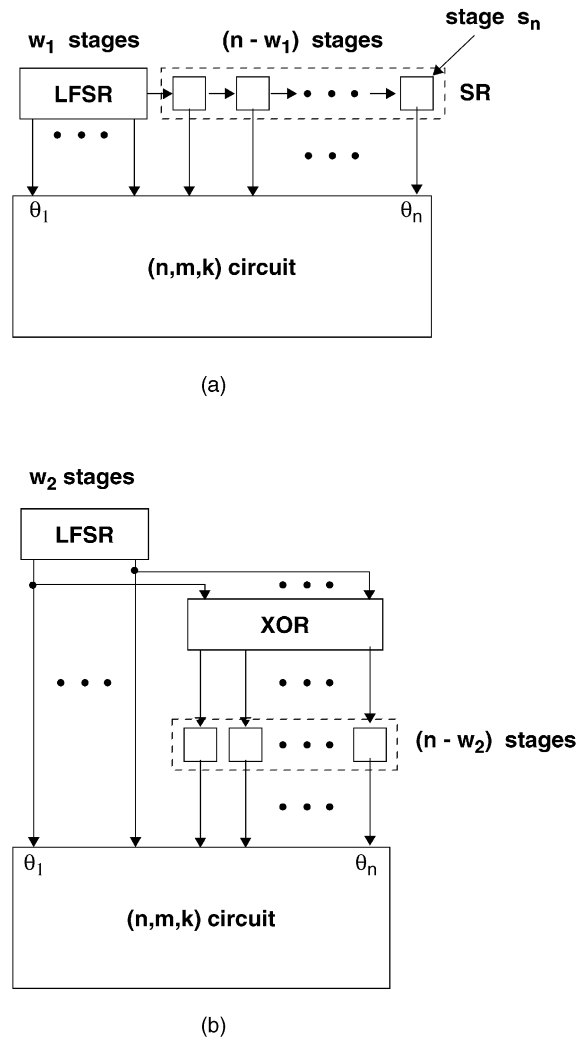

Fig. 1. TPG structures (a) LFSR/SR and (b) LFSR/XOR.

the test signal generated from stage s_1 is considered as the reference with value one (i.e., $r_1 = 1$). The residues $r_1, r_2, \ldots, r_n$, given by $1, x, \ldots, x^{n-1}$, represent the n independent test signals generated by the LFSR stages $s_1, s_2, \ldots, s_n$, respectively.

For an (n, m, k) circuit, an LFSR/SR TPG can be constructed by trying all primitive polynomials of degrees $k, k+1, \ldots, w_1$ (where $w_1 \leq n$) until a TPG of degree w_1 is found that generates an exhaustive set of patterns for each of the m output cones of the circuit. The first w_1 stages of the register are configured as a maximal length LFSR with primitive polynomial $P_1(x)$ of degree w_1. The remaining $(n - w_1)$ consecutive stages are connected as a shift register (SR). The LFSR/SR structure is shown in Fig. 1a. The LFSR stages generate w_1 independent test signals represented by the residues $1, x, \ldots, x^{w_1 - 1}$, respectively. Assuming $n < 2^{w_1}$, the SR stages generate $(n - w_1)$ unique linear combinations of these w_1 independent test signals. For stage s_i, the residue r_i, given by $x^{i-1}\ mod\ P_1(x)$, is a unique linear combination of the residues r_1 through r_{w_1}. For example, if r_i equals $1 + x$, then r_i is a linear combination of the residues r_1 and r_2. Hence, the test signal applied to the input θ_i is a linear combination of the test signals applied to the inputs θ_1 and θ_2. Residues r_{w_1+1} through r_n are fixed by

the polynomial $P_1(x)$. We shall refer to the LFSR/SR structure described above as *single* LFSR/SR. The TPG is represented as a (w_1, n) LFSR/SR composed of n stages and consisting of an LFSR of degree w_1. The following theorem provides the necessary and sufficient condition for exhaustive testing of the output cones of the circuit.

Theorem 1 (Barzilai [8]). *An output cone that depends on the inputs $\theta_{i_1}, \theta_{i_2}, \ldots, \theta_{i_k}$ will be exhaustively tested if and only if the residues $r_{i_1}, r_{i_2}, \ldots, r_{i_k}$ are linearly independent.*

For an (n, m, k) circuit, an LFSR/XOR [10] can be constructed as follows: Let w_2 (where $k \leq w_2 \leq n$) be the required number of independent test signals for the LFSR/XOR structure as per the design procedure in [10]. A predetermined set of w_2 stages of the register (not necessarily the first w_2 stages) is configured as an LFSR with a primitive polynomial $P_2(x)$ of degree w_2. The w_2 stages of the LFSR generate w_2 independent test signals represented by the residues $1, x, \ldots, x^{w_2-1}$, respectively. A predetermined set of $(n - w_2)$ specific linear combinations of these w_2 independent test signals are generated using the remaining $(n - w_2)$ stages of the register and an XOR network. These $(n - w_2)$ test signals are applied to the remaining $(n - w_2)$ inputs. The residues for these $(n - w_2)$ test signals are computed as the linear combinations of the residues $1, x, \ldots, x^{w_2-1}$. The LFSR/XOR structure is shown in Fig. 1b. The TPG is represented as a (w_2, n) LFSR/XOR consisting of an LFSR of degree w_2 and generating n specific test signals for the circuit inputs. LFSR/XORs incur high area overhead due to the XOR network, but generate minimal pseudoexhaustive test sets by utilizing information about cone dependencies.

Both LFSR/SR and LFSR/XOR generate independent test signals from their LFSR stages. For the LFSR/SR, the linear combinations for the remaining stages are fixed by the feedback polynomial. For the LFSR/XOR, any desired linear combination of the test signals can be generated using an XOR network and assigned to the remaining inputs. The flexibility of generating arbitrary linear combinations of test signals in LFSR/XORs does not exist in LFSR/SRs. Hence, for some circuits, LFSR/SRs generate larger test sets than LFSR/XORs. However, LFSR/SRs incur low area overhead due to the avoidance of the XOR network.

Design operations, such as *reconfiguration of feedbacks*, *permutation of stages* and *sharing of test signals*, can be applied to single LFSR/SRs to obtain *reconfigurable* LFSR/SRs [14], *permuted* LFSR/SRs [9], and *sharing* LFSR/SRs [7], respectively. These operations have the potential to avoid linear dependencies in the residue sets of the output cones and thus reduce the test length, but result in increased hardware overhead.

A *reconfigurable LFSR/SR* [14] has the capability to reconfigure its feedback taps to realize different primitive polynomials for different test sessions. For a single LFSR/SR, a *single* feedback polynomial is selected such that the residue sets for all cones are linearly independent. For a reconfigurable LFSR/SR, a *minimal set* of feedback polynomials is selected such that the residue set for each cone is linearly independent for at least one of the polynomials. The feedback taps are reconfigured using multiplexers to realize a different primitive polynomial during each test session. A subset of output cones is exhaustively tested during each session and each cone will be exhaustively tested in at least one of the sessions. Reconfiguration hardware overhead can be minimized by judiciously selecting polynomials that have common feedback taps.

A *permuted LFSR/SR* [9] is essentially a single LFSR/SR whose stages form a permutation of stages of the register. In other words, stage s_i of the TPG need not necessarily drive the input θ_i of the circuit. The inputs are permuted such that the residue sets for all cones are linearly independent. A single LFSR/SR that could not be used because of the fixed assignment of residues may lead to an acceptable solution after reassigning the residues. The permutation of the inputs can result in hardware overhead due to the routing among the TPG stages.

A *sharing LFSR/SR* [7] allows sharing of test signals among different inputs, i.e., a residue can be assigned to more than one input. Unrelated inputs are assigned the same residues and identical test signals are applied to them.

These design operations enhance the capabilities of single LFSR/SRs by only a limited amount. In the next section, we will describe a new TPG design, called *convolved LFSR/SR*, which is an important extension to single LFSR/SR. For simplicity of presentation, we shall discuss convolved LFSR/SRs without reconfiguration of feedbacks, permutation of stages, and sharing of test signals. Nevertheless, convolved LFSR/SRs can also be constructed to take advantage of these design operations.

3 Convolved LFSR/SRs

A convolved LFSR/SR is derived from a single LFSR/SR design. Circuit inputs are sequentially assigned residues generated by the successive stages of a single LFSR/SR. During the assignment process, it may not be possible to assign a residue to an input due to linear dependencies for some output. Stages whose residues cause linear dependencies are skipped, as shown in Fig. 2a. This single LFSR/SR ensures linear independence for all outputs, but has more stages than the input register. The extra stages required by the single LFSR/SR can be avoided by using XOR gates, as shown in Fig. 2b. The resulting structure is referred to as *convolved LFSR/SR*.

A (w, n) convolved LFSR/SR for an (n, m, k) circuit can be designed as follows: Residues generated by a single LFSR/SR of degree w are considered for assignment to circuit inputs. The inputs are assigned residues one at a time, avoiding linear dependencies with already assigned residues. Let the inputs θ_1 through θ_i be assigned the residues r_1 through r_i, respectively. Stages s_1 through s_i of the convolved LFSR/SR are identical to the single LFSR/SR. Assume that input θ_{i+1} cannot be assigned any of the residues $r_{i+1}, r_{i+2}, \ldots, r_{i+j-1}$ because all of them are linearly dependent on already assigned residues for some output. Residue r_{i+j} is then selected for assignment to the input θ_{i+1}, as shown in Fig. 2a. The single LFSR/SR requires $(j - 1)$ extra stages (shown as shaded stages in the figure) whose residues are not assigned to inputs. These extra stages are avoided in the convolved LFSR/SR design. Stage s_{i+1} of the convolved LFSR/SR is made to generate

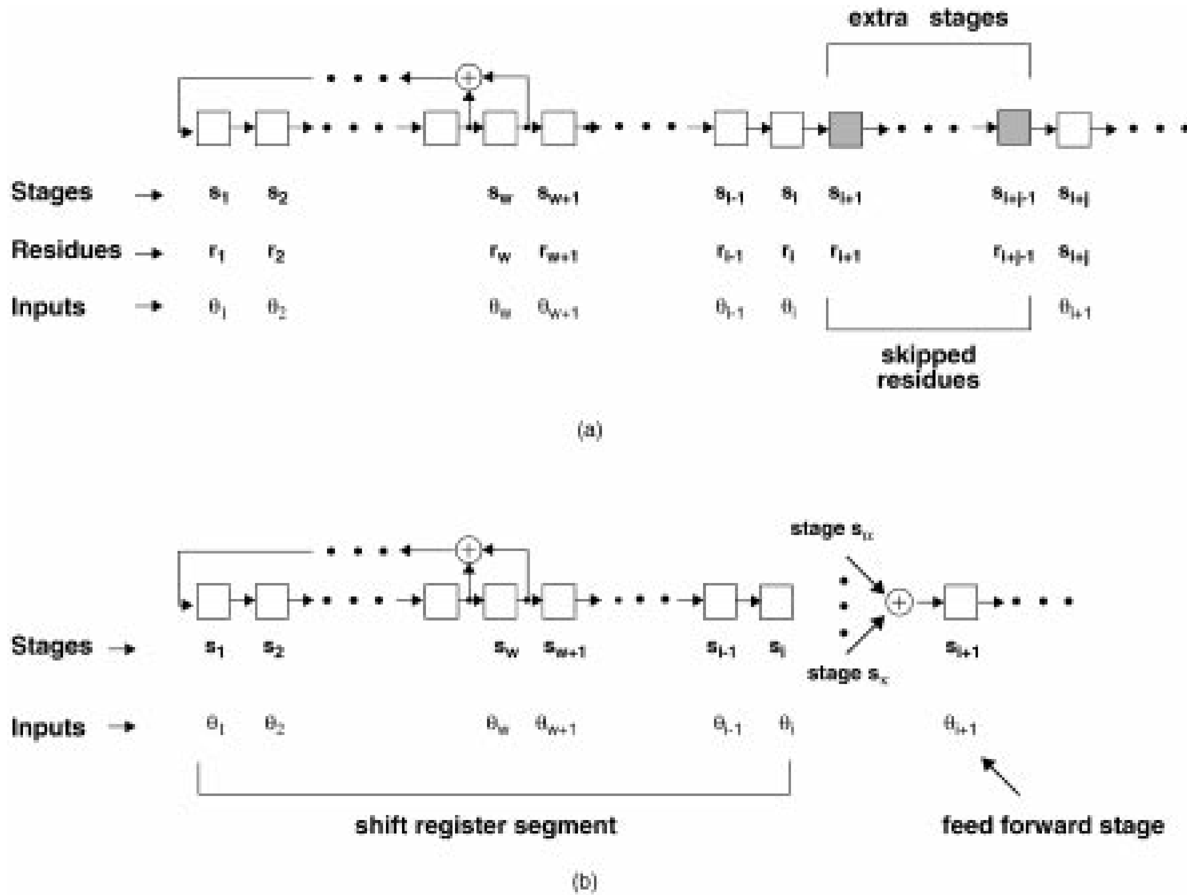

Fig. 2. Convolved LFSR/SR: (a) Residue assignment for inputs; (b) TPG stages.

the residue r_{i+j} by feeding the residue r_{i+j-1} at its input. Let the residue r_{i+j-1} be a linear combination of the residues $r_\alpha, r_\beta, \ldots, r_\kappa$, generated by the stages $s_\alpha, s_\beta, \ldots, s_\kappa$, respectively. These residues are combined using XOR gates and drive the input of stage s_{i+1}, as shown in Fig. 2b. Stage s_{i+1} is referred to as the *feedforward stage* and a maximal set of contiguous stages, e.g., stages s_1 through s_i, is referred to as a *shift register segment*. The assignment process is continued until all the inputs are assigned residues such that the residue sets for all output cones are linearly independent. The stages between the feedforward stages form shift register segments. The area overhead for the convolved LFSR/SR design is given by the number and size of XOR gates used to realize the individual feedforward stages.

Example 1. Consider the $(6, 5, 3)$ circuit shown in Fig. 3 with inputs θ_1 through θ_6 and outputs O_1 through O_5. The input dependencies for the five outputs are given by $\{\theta_1, \theta_2, \theta_3\}$, $\{\theta_1, \theta_3, \theta_4\}$, $\{\theta_2, \theta_3, \theta_5\}$, $\{\theta_2, \theta_4, \theta_6\}$, and $\{\theta_3, \theta_5, \theta_6\}$, respectively. Let the inputs be driven by an input register whose stages are denoted by s_1 through s_6.

Let us design a $(3, 6)$ convolved LFSR/SR for this example circuit. Consider the residues from a single LFSR/SR based on $P_1(x): x^3 + x + 1$. Residues r_1 through r_4 are assigned to inputs θ_1 through θ_4 without any linear dependence problem. Residue r_5 cannot be assigned to input θ_5 since the residue set $\{r_2, r_3, r_5\} = \{x, x^2, x^2 + x\}$ for output O_3 is linearly dependent. Skipping residue r_5, inputs θ_5 and θ_6 are assigned residues r_6 and r_7, respectively, as shown in Fig. 4a. This assignment ensures linear independence of the residue sets for all five outputs. The extra stage required by the single LFSR/SR can be avoided using a two-input XOR gate for the convolved LFSR/SR. Stage s_5 of the convolved LFSR/SR can generate the residue r_6 by feeding r_5 at its input. Residue r_5 is obtained by combining the residues r_2 and r_3. Hence, the linear combination of the outputs of stages s_2 and s_3 is fed as input to stage s_5, as shown in Fig. 4b. The convolved LFSR/SR can exhaustively test all outputs with $2^3 = 8$ patterns.

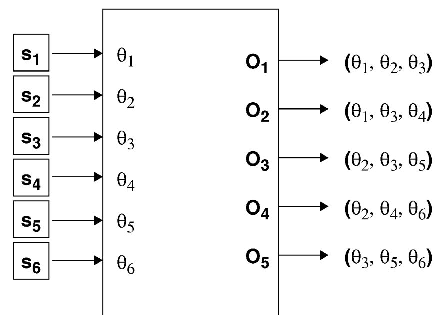

Fig. 3. A (6, 5, 3) circuit.

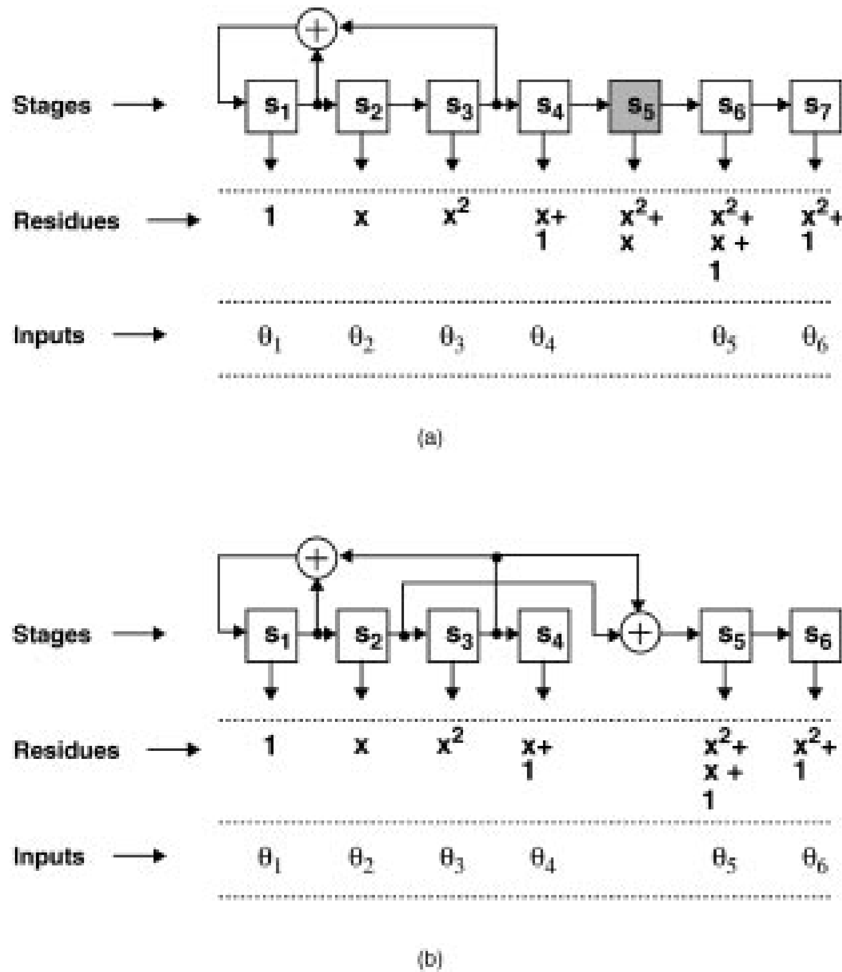

Fig. 4. Convolved LFSR/SR: (a) Residue assignment for inputs; (b) TPG stages.

Theorem 2. *A (w, n) convolved LFSR/SR exists for generating pseudoexhaustive tests for an (n, m, k) circuit if and only if there exists a (w, n) LFSR/XOR for the circuit.*

Proof. *(If)* Consider any (w, n) LFSR/XOR designed for the (n, m, k) circuit. The LFSR/XOR can be transformed to an equivalent (w, n) convolved LFSR/SR as follows: The residues for the test signals generated by the LFSR/XOR stages are determined from the XOR network. These residues can be generated from the corresponding stages of the convolved LFSR/SR by making some stages feedforward stages if necessary. In constructing the convolved LFSR/SR, design operations such as sharing of residues and permutation of inputs may be required.

(Only if) It is sufficient to show that any convolved LFSR/SR can be transformed to an equivalent LFSR/XOR. The residues of the convolved LFSR/SR stages are determined from the TPG structure. The XOR network for the LFSR/XOR can be designed such that the corresponding stages of the LFSR/XOR generate the assigned residues. □

Convolved LFSR/SRs have great potential to generate minimal test sets. They *bridge the gap* between LFSR/SRs and LFSR/XORs. A trivial convolved LFSR/SR is one without any feedforward XOR gates and is simply an LFSR/SR. On the other extreme, any stage of a convolved LFSR/SR can be made a feedforward stage to generate any desired residue using XOR gates similar to LFSR/XOR. Typically, convolved LFSR/SRs achieve low test lengths, like LFSR/XORs, and require low area overhead, like LFSR/SRs. The linear independence for the outputs is assured by adding XOR gates to stages whenever necessary. However, most of the stages form shift register segments, thereby avoiding high area overhead.

Though the functionality provided by a convolved LFSR/SR can also be realized using a simple LFSR/SR whose stages that give rise to linear dependencies are skipped, the number of stages skipped can be prohibitively large. The hardware overhead due to the use of feedforward stages in a convolved LFSR/SR is typically much lower than that due to skipped stages in the corresponding LFSR/SR realization.

4 Multiple LFSR/SRs

A multiple LFSR/SR forms a special case of a convolved LFSR/SR. It is composed of two or more independent single LFSR/SRs that are run in parallel. The single LFSR/SRs have identical feedback polynomials, but may have different shift register lengths and initial seeds.

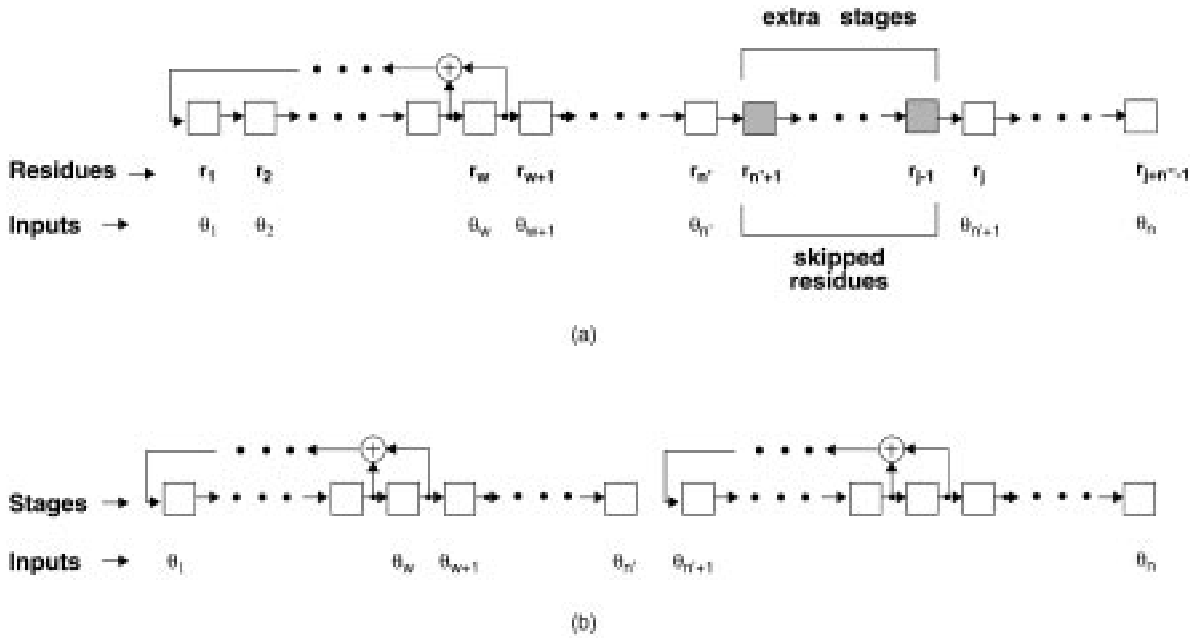

Fig. 5. Multiple LFSR/SR: (a) Residue assignment for inputs; (b) TPG stages.

A multiple LFSR/SR is also derived from a single LFSR/SR. A multiple LFSR/SR for an (n, m, k) circuit can be constructed as follows: Assume the residue assignment for the inputs shown in Fig. 5a ensures linear independence for all output cones. Let both n' and n'' be greater than or equal to w and the sum of n' and n'' be n. Residues r_1 through $r_{n'}$ are assigned to inputs θ_1 through $\theta_{n'}$, respectively. Residues $r_{n'+1}$ through r_{j-1} are skipped and not assigned to any input. Residues r_j through $r_{j+n''-1}$ are assigned to inputs $\theta_{n'+1}$ through θ_n, respectively. The single LFSR/SR requires $(j - n' - 1)$ extra stages (shown as shaded stages in the figure) whose residues are not assigned to inputs. These extra stages are avoided in the multiple LFSR/SR design. Stages s_1 through $s_{n'}$ of the input register are modified to a (w, n') single LFSR/SR. Stages $s_{n'+1}$ through s_n of the input register are modified to another (w, n'') single LFSR/SR. A (w, n) multiple LFSR/SR composed of a (w, n') and a (w, n'') single LFSR/SRs is shown in Fig. 5b. Both single LFSR/SRs have identical LFSRs of degree w.

A residue assigned to an input can be expressed as a linear combination of the residues $r_1, r_2, \ldots, r_w$. The residues $r_1, r_2, \ldots, r_w$ are generated by the LFSR stages of the first single LFSR/SR. Let an initial seed S be applied to the LFSR portion of the first single LFSR/SR. From this seed, the initial contents for the rest of the stages of the multiple LFSR/SR can be determined. For example, if an input θ_i is assigned a residue which is a linear combination of the residues $r_{i_1}, r_{i_2}, \ldots, r_{i_k}$ (where $i_1, i_2, \ldots, i_k < w$), then the initial content of stage s_i is a linear combination of the initial contents of the stages $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$. This initialization ensures that stage s_i of the multiple LFSR/SR generates the residue assigned to input θ_i.

Example 2. A multiple LFSR/SR for the $(6, 5, 3)$ circuit shown in Fig. 3 can be designed as follows: The residue assignment for the inputs shown in Fig. 6a satisfies the linear independence of residue sets for all outputs. We can construct a multiple LFSR/SR composed of two $(3, 3)$ single LFSR/SRs, as shown in Fig. 6b. Both single LFSR/SRs are based on the same primitive polynomial $x^3 + x + 1$. The stages of the first LFSR/SR generate residues r_1, r_2, and r_3. Let an initial seed $S_1 = 100$ be applied to the first LFSR/SR. The initial seed S_2 for the second LFSR/SR is determined such that its stages generate the residues r_5, r_6, and r_7. Input θ_4 is assigned the residue r_5, which is the linear combination of the residues r_2 and r_3. Hence, the initial content of stage s_4 is zero, which is a linear combination of the initial contents of the stages s_2 and s_3. Thus, the initial seed of the second LFSR/SR is computed as $S_2 = 011$. The stages of the multiple LFSR/SR generate the assigned residues shown in Fig. 6a. This multiple LFSR/SR generates $2^3 = 8$ patterns to exhaustively test all the five outputs.

The following theorem characterizes the relation between multiple LFSR/SRs and convolved LFSR/SRs.

Theorem 3. *A (w, n) multiple LFSR/SR exists for generating pseudoexhaustive tests for an (n, m, k) circuit if and only if there exists a (w, n) convolved LFSR/SR for the circuit where the length of each shift register segment is at least w.*

Proof. *(If)* For the (w, n) convolved LFSR/SR, since the shift register segments are of length at least w, each one of them can be modified as an independent single LFSR/SR with the same feedback polynomial as that of the convolved LFSR/SR. The initial seeds for the single LFSR/SRs are the same as the initial seeds of the shift register segments of the convolved LFSR/SR.

(Only if) It is sufficient to show that any multiple LFSR/SR can be transformed to an equivalent convolved LFSR/SR. The independent single LFSR/SRs of the (w, n) multiple LFSR/SR can be modified to form shift register segments of a (w, n) convolved LFSR/SR. The initial seed of the multiple LFSR/SR determines the

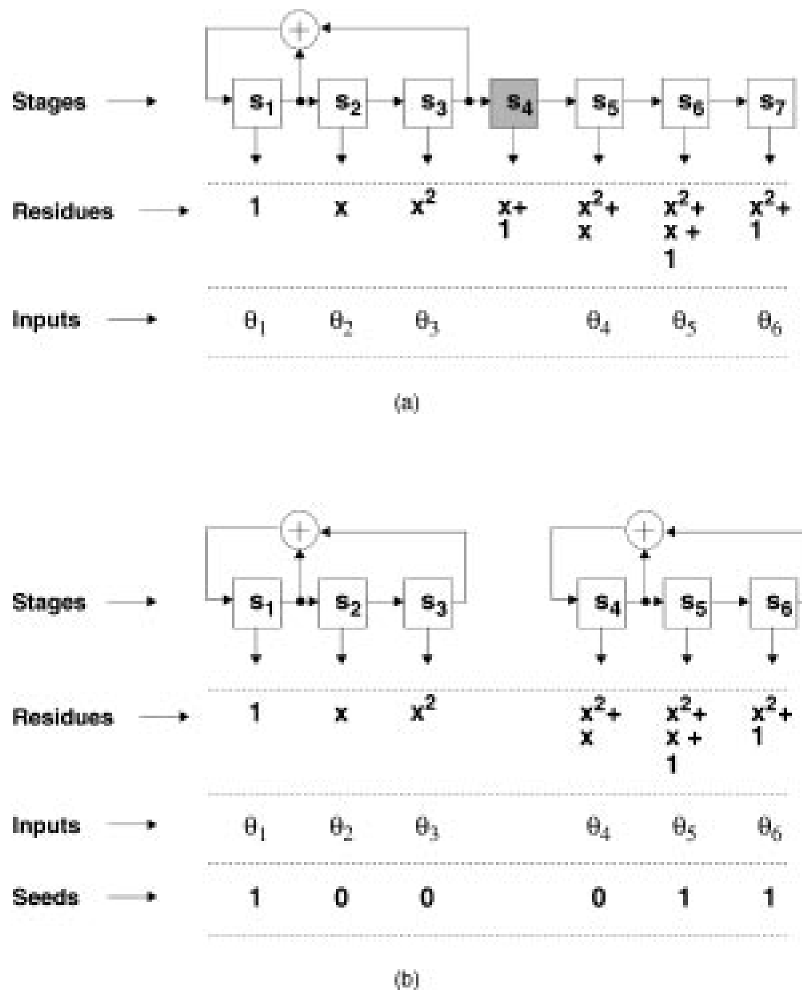

Fig. 6. A multiple LFSR/SR design: (a) Residue assignment for inputs; (b) TPG stages.

residues generated by the individual stages. For the convolved LFSR/SR, the feedforward stages are made to generate their assigned residues by using XOR networks. The remaining stages of the convolved LFSR/SR automatically generate their respective assigned residues. The resulting convolved LFSR/SR has each shift register segment having at least w stages. □

Multiple LFSR/SRs utilize an XOR network for realizing the feedback polynomial of the individual single LFSR/SRs. On the contrary, convolved LFSR/SRs utilize XOR networks for realizing the feedforward stages. The hardware overhead due to the XOR networks is the deciding factor for selecting either a multiple LFSR/SR or a convolved LFSR/SR. It is usually hard to construct a multiple LFSR/SR that generates as small a test set as that of an "*optimal*" convolved LFSR/SR due to the restriction on the length of the shift register segments.

Fig. 7 highlights the characteristics of various TPG designs based on empirical observations. Among the four TPG designs, single LFSR/SRs generate the largest test sets, but often incur the least hardware overhead. At the other extreme, LFSR/XORs generate the smallest test sets, but usually require the most hardware. Convolved LFSR/SRs generate small test sets like LFSR/XORs, but require less hardware. The arrows indicate the possible transformations from one TPG design to another. Any convolved LFSR/design can be transformed to an LFSR/XOR design and vice versa. Similarly, any multiple LFSR/SR design can be transformed to a convolved LFSR/SR design. However, a convolved LFSR/SR design can be transformed to a multiple LFSR/SR design provided the shift register segments are of lengths greater than or equal to the degree of the LFSR (refer to Theorem 3). Similarly, a multiple LFSR/SR design can be transformed to a single LFSR/SR design provided the length of the shift register segment equals the number of inputs to the circuit.

5 Design Procedure

For an (n, m, k) circuit, designing a convolved LFSR/SR that is optimal in terms of both pseudoexhaustive test length and hardware overhead involves:

1. considering each primitive polynomial of degree k and if necessary of higher degree,
2. assigning all possible combinations of residues that can be generated by the polynomial to circuit inputs,
3. ensuring linear independence of all residues assigned to circuit inputs for each output cone,
4. minimizing the number of feedforward stages, and

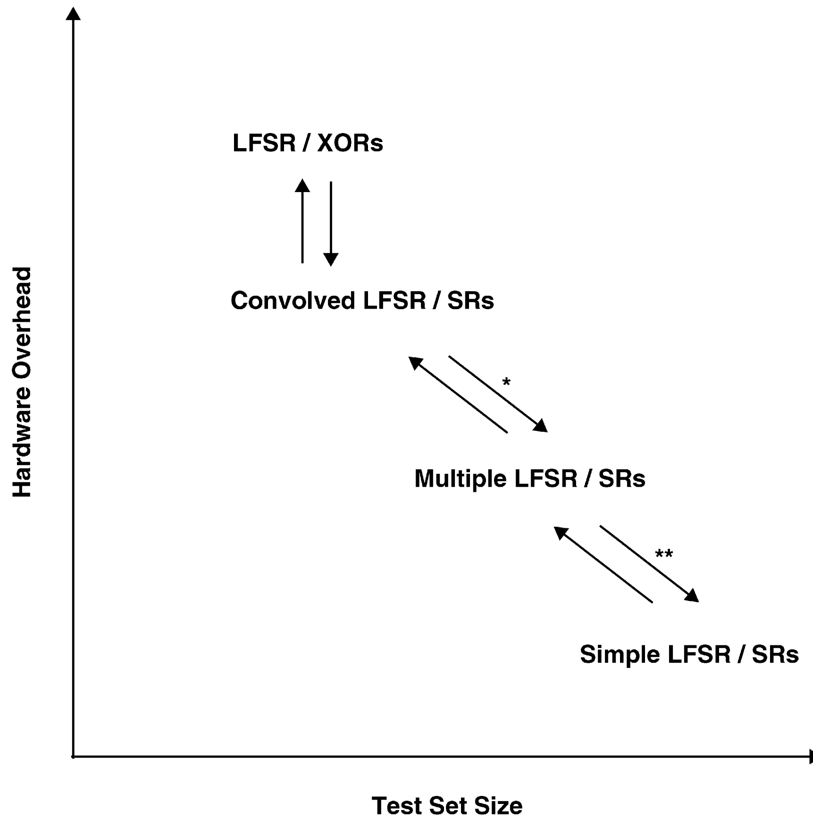

Fig. 7. Characteristics of various TPGs.

5. minimizing the hardware in terms of the number of XOR gates required to realize each feedforward stage.

The exponential computational complexity involved in determining an optimal convolved LFSR/SR can be avoided by undertaking a pragmatic approach to design a suboptimal TPG. Practically, a (w, n) convolved LFSR/SR can be designed for an (n, m, k) circuit as follows: A primitive polynomial $P(x)$ of degree w is selected as the feedback polynomial. The polynomial can generate $(2^w - 1)$ unique residues from r_1 through r_{2^w-1} and all of them can be considered for assignment to the inputs. The number of possible residues is exponential in the degree of the polynomial. Only a few residues, say r_1 through r_N, are considered for input assignment. This limits the number of convolved LFSR/SR designs to be considered. The shift register segments are constrained to have a desired minimum length, say l. This constraint attempts to reduce the number of feedforward stages and, hence, the area overhead due to the XOR networks. A (w, n) multiple LFSR/SR can be obtained by restricting $l \geq w$ and a (w, n) single LFSR/SR can be obtained by restricting $l = n$. It may be necessary to repeat the design procedure for different primitive polynomials to obtain an efficient TPG.

Procedure TPG

Input: Input dependencies for all the outputs of (n, m, k) circuit;
A primitive polynomial $P(x)$ of degree $w \geq k$;
(l) :: minimum length requirement for the noninitial shift register segments;
(N) :: maximum number of residues considered for assignment.

Output: Residue assignment for the inputs;
Initial seeds for the shift register segments;
XOR network for the feedforward stages.

1. **residues ()**;
/* determines residues for the inputs such that the sets of residues for all outputs are linearly independent. */
2. **seeds ()**;
/* determines the seeds for the TPG stages such that the inputs are driven by the test signals that represent the corresponding assigned residues. */
3. **network ()**;
/* determines the XOR network to implement the linear combinations of the test signals for the feedforward stages. */

Procedure residues ()

1. Assign residues r_1 through r_w to inputs θ_1 through θ_w.
2. $i \leftarrow (w+1)$; $j \leftarrow (w+1)$.
   /* residue r_j is assigned to input θ_i */
3. While not all inputs are assigned residues do
   a. If $(i = w)$ and $(j = w+1)$ exit with failure.
   b. If $(N-j) < (n-i)$ /* not enough residues left for remaining inputs */
      then $\{i \leftarrow i'; j \leftarrow (j'+1)\}$
      (where $i' = i-1$ and $r_{j'}$ is the residue assigned to input $\theta_{i'}$).
   c. Assign residue r_j to input θ_i.
   d. If the assigned residues are linearly independent for all outputs,
      then $i \leftarrow (i+1)$.
   e. $j \leftarrow (j+1)$.
   f. If the last shift register segment length $< l$
      /* requirements not met */
      then $\{i \leftarrow i''; j \leftarrow (j''+1)\}$
      (where i'' is the first input in the last shift register segment and $r_{j''}$ is the residue assigned to input $\theta_{i''}$).
4. Print the residue assignment for the inputs.

Procedure seeds ()

1. For every input θ_i do
   a. Determine the residue $r_{i'}$ assigned to input θ_i.
   b. If $r_{i'}$ contains the term r_1 $(=1)$ initialize stage s_i to 1,
      else initialize stage s_i to 0.
2. Print the initial values of all stages.

Procedure network ()

1. For every feedforward stage s_i do
   a. Determine the residue $r_{i'}$ that should appear at the input of stage s_i.
   b. $R \leftarrow \emptyset$. /* R contains the collection of residues to realize $r_{i'}$ */
   c. While $(r_{i'} \neq 0)$ do
      i. Determine the residue $r_{j'}$ from a stage $j(< i)$ which differs from the residue $r_{i'}$ in minimum number of terms.
      ii. $R \leftarrow R \cup r_{j'}$: $r_{i'} \leftarrow r_{i'} \oplus r_{j'}$.
   d. Combine the residues in R using two-input XOR gates.

5.1 Determination of Residues

The iterative search procedure progressively assigns residues to the inputs. The linear independence among the residues for all outputs is checked after every assignment. The linear independence is determined by the matrix rank determination procedure outlined in [8]. Backtracking occurs whenever there are not enough residues left for the unassigned inputs or a shift register segment length becomes less than l. The procedure reports after all inputs are assigned residues. Alternate primitive polynomials are considered if a convolved LFSR/SR cannot be determined with $P(x)$. The convolved LFSR/SR will generate a minimum test set only if the degree of the selected polynomial equals k. Note that if the TPG designed by this procedure requires a test length greater than 2^k, then operations such as reconfiguration, permutation, and sharing can be used to attempt to obtain a TPG with lower test length.

5.2 Determination of Seeds

The first w stages generate independent signals and these stages are initialized with the seed $100\ldots0$. All other stages generate linear combinations of these signals as given by their assigned residues. For stage s_i, if the assigned residue is a linear combination of the residues $r_{i_1}, r_{i_2}, \ldots, r_{i_k}$ (where $i_1, i_2, \ldots, i_k < w$), then the initial content of stage s_i is a linear combination of the initial contents of the stages $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$. Among the stages $s_1, s_2, \ldots, s_w$, only stage s_1 has nonzero initial content. Hence, stage s_i is initialized to nonzero value only if its assigned residue contains the term $r_1 = 1$. The initial seed determined by this method ensures that the TPG stages generate their respective assigned residues.

5.3 Determination of XOR Network

The feedback and the feedforward stages need XOR gates to realize their assigned residues. The XOR network required by the feedback stage is specified by its feedback taps. The XOR network for a feedforward stage s_i is determined as follows: From the residue assigned to the input θ_i, the residue (say $r_{i'}$) of the test signal that needs to be fed at the input of stage s_i is determined. This residue $r_{i'}$ is obtained as the linear sum of a minimal number of residues from the earlier stages. Since it employs a greedy heuristic, the procedure *network* gives an upper bound on the number of two-input XOR gates to realize the feedforward stages.

Multiple LFSR/SRs can be realized if the shift register segments are of length at least equal to the degree of the feedback polynomial $P(x)$. In this case, each shift register segment is transformed into an independent single LFSR/SR with $P(x)$ as the feedback polynomial.

6 Experimental Results

We have designed various pseudoexhaustive TPGs for the ISCAS85 combinational benchmark circuits [15]. The circuits were partitioned using our partitioning procedure [16] so that all output cones have 20 or fewer inputs. The test length for the TPGs is tabulated in Table 1. The first three columns in the table describe the characteristics of the benchmark circuits before and after the application of the partitioning procedure. Three circuit specific TPGs—convolved LFSR/SRs, multiple LFSR/SRs, and single LFSR/SRs—were designed for the partitioned circuits. The last three columns in the table denote the pseudoexhaustive test length for these TPG designs.

The TPGs effectively utilize the information about the circuit output cone dependencies. For each circuit, only two primitive polynomials were considered while designing convolved LFSR/SRs and multiple LFSR/SRs and up to 100 primitive polynomials were considered while designing single LFSR/SRs. Both convolved LFSR/SRs and multiple

TABLE 1
Test Length for Partitioned Benchmark Circuits

Ckt	(n,m,k)		Test Length		
	Before Partitioning	After Partitioning	Convolved LFSR/SR	Multiple LFSR/SR	Single LFSR/SR
c432	(36,7,36)	(56,27,20)	2^{20}	2^{20}	2^{20}
c499	(41,32,41)	(49,40,14)	2^{14}	2^{14}	2^{15}
c880	(60,26,45)	(70,36,17)	2^{17}	2^{17}	2^{17}
c1355	(41,32,41)	(49,40,14)	2^{14}	2^{14}	2^{15}
c1908	(33,25,33)	(47,39,20)	2^{20}	2^{20}	2^{21}
c2670	(233,140,120)	(262,169,20)	2^{20}	2^{20}	2^{20}
c3540	(50,22,50)	(108,80,20)	2^{20}	2^{20}	2^{22}
c5315	(178,123,67)	(215,160,20)	2^{20}	2^{20}	2^{22}
c6288	(32,31,32)	(99,98,20)	2^{20}	2^{20}	—
c7552	(207,108,194)	(286,187,20)	2^{20}	2^{20}	2^{22}

TABLE 2
Convolved LFSR/SR Designs for Partitioned Benchmark Circuits

Ckt	(n,m,k)	Polynomial	Residue Assignment	SR Segments
c432	(56,27,20)	20 6 4 1 0	1-36 49-68	36 20
c499	(49,40,14)	14 5 3 1 0	1-16 53-68 151-167	16 16 17
c880	(70,36,17)	17 3 0	1-24 118-146 194-210	24 29 17
c1355	(49,40,14)	14 5 3 1 0	1-16 53-68 151-167	16 16 17
c1908	(47,39,20)	20 6 4 1 0	1-20 56-82	20 27
c2670	(262,169,20)	20 3 0	1-100 103-244 260-279	100 142 20
c3540	(108,80,20)	20 3 0	1-20 129-152 157-176	20 24 20
			210-232 379-399	23 21
c5315	(215,160,20)	20 3 0	1-110 121-142 156-177	110 22 22
			208-227 577-597 683-702	20 21 20
c6288	(99,98,20)	20 6 4 1 0	1-20 209-229 272-292	20 21 21
			350-386	37
c7552	(286,187,20)	20 3 0	1-49 51-114 133-190 196-229	49 64 58 34
			246-267 290-310 456-493	22 21 38

LFSR/SRs generate minimum test sets for all the partitioned circuits. For example, the partitioned $c3540$ circuit has a maximum dependency of 20 inputs and, hence, any pseudoexhaustive test set must contain at least 2^{20} patterns. The best single LFSR/SR found generated a test set with 2^{22} patterns.

LFSR/XORs can be designed by transforming the convolved LFSR/SR designs (refer to Theorem 2). LFSR/XORs will also generate minimum test sets for all the partitioned circuits. However, many XOR gates may be needed for realizing the LFSR/XOR structures.

Because of practical limitations, single LFSR/SRs were determined by considering only 100 primitive polynomials of each degree. Due to this restriction, there may exist single LFSR/SRs for the partitioned circuits with smaller degrees than those shown. For the partitioned $c6288$ circuit, no single LFSR/SR design was found, in spite of trying 100 primitive polynomials of degrees from 20 to 40. The circuit contains a few outputs driven by 19 consecutive inputs and one nonconsecutive input. Single LFSR/SRs failed to generate exhaustive test sets for these cones. On the other hand, both convolved LFSR/SRs and multiple LFSR/SRs generated test sets due to their flexibility in assigning residues. This flexibility is evident by another set of convolved LFSR/SR designs for the partitioned benchmark circuits that can be found in [17].

Table 2 presents the details about the residue assignment for the convolved LFSR/SR designs for the partitioned benchmark circuits. The first two columns provide the characteristics of the partitioned circuits. The exponent terms for the feedback polynomial, the residue assignment for the stages, and the length of shift register segments are given by the third, fourth, and fifth columns, respectively. For example, for the partitioned $c432$ circuit, the convolved

TABLE 3
Single LFSR/SR Designs for Partitioned Benchmark Circuits

Ckt	(n,m,k)	Polynomial	Residue Assignment
c432	(56,27,20)	20 11 7 5 4 3 2 1 0	1-56
c499	(49,40,14)	15 9 8 6 5 3 0	1-49
c880	(70,36,17)	17 10 8 6 5 3 2 1 0	1-70
c1355	(49,40,14)	15 9 8 6 5 3 0	1-49
c1908	(47,39,20)	21 8 6 3 2 1 0	1-47
c2670	(262,169,20)	20 9 5 4 0	1-262
c3540	(108,80,20)	22 11 10 7 4 2 0	1-108
c5315	(215,160,20)	22 9 7 5 3 2 0	1-215
c6288	(99,98,20)	—–	—-
c7552	(286,187,20)	22 9 8 6 5 3 0	1-286

TABLE 4
Comparison among Circuit Specific TPG Designs

		Convolved		Multiple		Simple	
Ckt	(n,m,k)	TL	XOR	TL	XOR	TL	XOR
c432	(56,27,20)	2^{20}	6	2^{20}	6	2^{20}	7
c499	(49,40,14)	2^{14}	10	2^{14}	9	2^{15}	5
c880	(70,36,17)	2^{17}	7	2^{17}	3	2^{17}	7
c1355	(49,40,14)	2^{14}	10	2^{14}	9	2^{15}	5
c1908	(47, 39,20)	2^{20}	10	2^{20}	6	2^{21}	5
c2670	(262,169,20)	2^{20}	3	2^{20}	3	2^{20}	3
c3540	(108,80,20)	2^{20}	11	2^{20}	5	2^{22}	5
c5315	(215,160,20)	2^{20}	7	2^{20}	6	2^{22}	5
c6288	(99,98,20)	2^{20}	18	2^{20}	12	–	–
c7552	(286,187,20)	2^{20}	11	2^{20}	7	2^{22}	5

LFSR/SR design is based on the feedback polynomial $P(x) : x^{20} + x^6 + x^4 + x + 1$. The residue assignment for all the 56 input register stages is given by the fourth column. Stages s_1 through s_{36} are assigned residues r_1 through r_{36}, respectively. Stage s_{37} forms a feedforward stage and stages s_{37} through s_{56} are assigned residues r_{49} through r_{68}. The lengths of the two shift register segments are 36 and 20, respectively. All convolved LFSR/SR designs have very few shift register segments and each segment is of length at least equal to the degree of the LFSR. The residue assignments can be easily mapped into multiple LFSR/SRs for the circuits.

Table 3 presents the details about the single LFSR/SR designs for the partitioned benchmark circuits. The feedback polynomials and the residue assignments are given by the third and fourth columns, respectively. The hardware overhead involved in these designs is given by the number of 2-input XOR gates required to realize the LFSR structure. For example, for the partitioned $c432$ circuit, seven XOR gates are needed to realize the LFSR of degree 20.

A comparison of test lengths and hardware overhead among the TPG designs is given in Table 4. Columns 3, 5, and 7 present the pseudoexhaustive test lengths for the TPG designs. The hardware overhead in terms of 2-input XOR gates required for realizing the TPG designs is given in columns 4, 6, and 8, respectively. For example, the number of XOR gates utilized in TPG designs for the partitioned $c432$ circuit is computed as follows: The convolved LFSR/SR residue assignment for $c432$ is comprised of two shift register segments with stage 36 forming a feedforward stage. This stage can be realized as the linear sum of the test signals from the output of stages 7, 9, 15, and 19 using three XOR gates. The LFSR requires three XOR gates and, hence, the convolved LFSR/SR requires six XOR gates. For this circuit, a multiple LFSR/SR can be realized by transforming the two shift register segments into two independent single LFSR/SRs. Since each LFSR requires three XOR gates, the multiple LFSR/SR design also requires six XOR gates. On the other hand, a single LFSR/SR design requires seven XOR gates for this circuit. It should be noted that the TPG hardware overhead reported here are in addition to the hardware needed to partition the original benchmark circuits. Overheads due to routing of signals and feedback connections are ignored. The hardware overhead for partitioning the benchmark circuits is reported in [16].

Both convolved LFSR/SRs and multiple LFSR/SRs generate minimum test sets for all the partitioned benchmark circuits. However, multiple LFSR/SR designs usually incur less hardware overhead than convolved LFSR/SRs. It should be noted that only an upper bound on the number of XOR gates required for convolved LFSR/SRs is given in the table. This is due to the fact that the procedure *network* uses a suboptimal heuristic to determine the XOR network for the feedforward stages. Single LFSR/SRs generate minimum test sets only for three circuits $c432$, $c880$, and $c2670$. For these circuits, the multiple LFSR/SRs utilize either the same number or fewer XOR gates than the single LFSR/SRs. For the remaining circuits, single LFSR/SRs generate up to four times the size of the minimum test sets. For these circuits, multiple LFSR/SRs require less than twice the number of XOR gates utilized by single LFSR/SRs.

7 Conclusions

In this paper, we have presented new hardware efficient TPG designs to generate minimal pseudoexhaustive test sets for combinational circuits. These TPGs utilize the information about the circuit output cone dependencies. Convolved LFSR/SRs have great potential to generate minimum test sets as demonstrated by the experiments on the combinational benchmark circuits. These structures can be used to derive other test pattern generators such as LFSR/XORs and multiple LFSR/SRs. The flexibility of manipulating residues with a XOR network that is absent in single LFSR/SRs but present in LFSR/XORs makes convolved LFSR/SRs powerful test pattern generators.

We have also studied theoretical upper bounds on pseudoexhaustive test lengths [13] employing the knowledge of the circuit output cone structures. Our theoretical work justifies the experimental observation (refer to Table 1) that, in general, the upper bound on the degree of the LFSR is very close to the maximum cone size for the partitioned benchmark circuits.

Acknowledgments

This work was supported by the Advanced Research Projects Agency and monitored by the Federal Bureau of Investigation under Contract No. JFBI90092. A preliminary version of this work was presented at the 1993 International Test Conference.

References

[1] P.H. Bardell, W.H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques.* New York: John Wiley & Sons, 1987.

[2] *Additive Cellular Automata: Theory and Applications,* P.P.Chaudhuri, ed. Los Alamitos, Calif.: IEEE CS Press, 1997.

[3] M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design.* IEEE Press, 1994.

[4] D.T. Tang and L.S. Woo, "Exhaustive Test Pattern Generation with Constant Weight Vectors," *IEEE Trans. Computers,* vol. 32, no. 12, pp. 1,145-1,150, Dec. 1983.

[5] D.T. Tang and C.L. Chen, "Logic Test Pattern Generation Using Linear Codes," *IEEE Trans. Computers,* vol. 33, no. 9, pp. 845-850, Sept. 1984.

[6] L.T. Wang and E.J. McCluskey, "Condensed Linear Feedback Shift Register (LFSR) Testing—A Pseudoexhaustive Test Technique," *IEEE Trans. Computers,* vol. 35, no. 4, pp. 367-370, Apr. 1986.

[7] L.T. Wang and E.J. McCluskey, "Circuits for Pseudoexhaustive Test Pattern Generation," *IEEE Trans. Computer-Aided Design,* vol. 7, no. 10, pp. 1,068-1,080, Oct. 1988.

[8] Z. Barzilai, D. Coppersmith, and A. Rosenberg, "Exhaustive Bit Pattern Generation in Discontiguous Positions with Applications to VLSI Testing," *IEEE Trans. Computers,* vol. 32, no. 2, pp. 190-194, Feb. 1983.

[9] D. Kagaris and S. Tragoudas, "Cost-Effective LFSR Synthesis for Optimal Pseudo-Exhaustive BIST Test Sets," *IEEE Trans. VLSI Systems,* vol. 1, no. 4, pp. 526-536, Dec. 1993.

[10] S.B Akers, "On the Use of Linear Sums in Exhaustive Testing," *Proc. 15th Int'l. Symp. Fault-Tolerant Computing,* pp. 148-153, June 1985.

[11] E. Wu and P.W. Rutkowski, "PEST—A Tool for Implementing Pseudo-Exhaustive Self Test," *Proc. First European Design Automation Conf.,* pp. 639-643, Mar. 1990.

[12] C.H. Chen, "BISTSYN—A Built-In Self Test Synthesizer," *Proc. Int'l Conf. Computer-Aided Design,* pp. 240-243, Nov. 1991.

[13] R. Srinivasan, S.K. Gupta, and M.A. Breuer, "Bounds on Pseudo-Exhaustive Test Lengths," *IEEE Trans. VLSI Systems,* vol. 6, no. 3, pp. 420-431, Sept. 1998.

[14] J.G. Udell, "Reconfigurable Hardware for Pseudo-Exhaustive Test," *Proc. Int'l Test Conf.,* pp. 522-530, Sept. 1988.

[15] F. Brglez and H. Fujiwara, "A Neutral Netlist of Ten Combinational Benchmark Circuits and a Target Translator in FORTRAN," *Proc. Int'l. Symp. Circuits and Systems,* pp. 663-698, June 1985.

[16] R. Srinivasan, S.K. Gupta, and M.A. Breuer, "An Efficient Partitioning Strategy for Pseudo-Exhaustive Testing," *Proc. Design Automation Conf.,* pp. 242-248, June 1993.

[17] R. Srinivasan, S.K. Gupta, and M.A. Breuer, "Novel Test Patttern Generators for Pseudo-Exhaustive Testing," *Proc. Int'l Test Conf.,* pp. 1,041-1,050, Oct. 1993.

Rajagopalan Srinivasan received the BTech degree in electrical engineering (electronics) from the Indian Institute of Technology, Madras, the ME degree in electrical communication engineering from the Indian Institute of Science, Bangalore, and the MS and PhD degrees in computer engineering from the University of Southern California, Los Angeles. He is a senior staff design engineer in the Network Components Division of Intel Corporation based in Morganville, New Jersey. Previously, he was a member of the technical staff at Lucent Bell Laboratories and has worked on various data and optical ASIC design, verification, and testability projects. Prior to joining Lucent, he worked as lead CAD engineer for the Corporate CAD Department of Sun Microsystems in Menlo Park, California. He is a member of the IEEE Computer Society.

Sandeep Gupta received his Bachelors degree in electrical engineering from the Indian Institute of Technology, Kharagpur, India, in 1985 and obtained MS and PhD degrees in electrical and computer engineering from the University of Massachusetts at Amherst in 1989 and 1991. He is an associate professor in the Department of Electrical Engineering-Systems at the University of Southern California, Los Angeles. He is also an associate editor of the *IEEE Transactions on Computers*. His research interests are in the area of VLSI testing and design. He is currently involved in projects on test and validation of deep submicron circuits, testing multicore systems-on-silicon, and delay testing and diagnosis of digital circuits. He is also involved in a project on testing and verification of network protocols.

He is a recipient of the US National Science Foundation's Research Initiation Award (1992) and CAREER Award (1995). He is also a recipient of the Northrop Grumman Assistant Professorship (1995) and Zumberge Fellowship (1996) at the University of Southern California. He also received the Honorable Mention Award from the International Test Conference in 1997. He is a member of the IEEE Computer Society.

Melvin A. Breuer received his PhD degree in electrical engineering from the University of California, Berkeley, and is currently a professor of both electrical engineering and computer science at the University of Southern California, Los Angeles, and is the Charles Lee Powell Professor of Computer Engineering. He was chairman of the Department of Electrical Engineering-Systems from 1991-1994. His main interests are in the area of computer-aided design of digital systems, design-for-test and built-in self-test, and VLSI circuits.

Dr. Breuer is the editor and coauthor of *Design Automation of Digital Systems: Theory and Techniques* (Prentice Hall); editor of *Digital Systems Design Automation: Languages, Simulation and Data Base* (Computer Science Press); coauthor of *Diagnosis and Reliable Design of Digital Systems* (Computer Science Press); coeditor of *Computer Hardware Description Languages and their Applications* (North-Holland); coeditor and contributor to *Knowledge Based Systems for Test and Diagnosis* (North-Holland); and coauthor of *Digital System Testing and Testable Design* (Computer Science Press, 1990 and reprinted in 1995 by the IEEE Press). He has published more than 200 technical papers and was formerly the editor-in-chief of the *Journal of Design Automation and Fault Tolerant Computing*, on the editorial board of the *Journal of Electronic Testing*, the coeditor of the *Journal of Digital Systems*, and the program chairman of the Fifth International IFIP Conference on Computer Hardware Description Languages and Their Applications. He was coauthor of a paper that received an honorable mention award at the 1997 International Test Conference. He is a fellow of the IEEE; was a Fullbright-Hays scholar (1972); received the 1991 Associates Award for Creativity in Research and Scholarship from the University of Southern California, the 1991 USC School of Engineering Award for Exceptional Service, and the IEEE Computer Society's 1993 Taylor L. Booth Education Award. He was chair of the Faculty of the School of Engineering, USC, for the 1997-1998 academic year. He is currently chairman of the EE-Systems Department.