[9] IEEE Project 802, *Draft IEEE Standard 802.3 CSMA/CD Access Method and Physical layer Specifications*. IEEE CS, Revision D, 1985.
[10] L. Kleinrock and F.A. Tobagi, "Packet Switching in Radio Channel: Part I—Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *IEEE Trans. Comm.*, vol. 23, no. 12, pp. 1,400-1416, Dec. 1975.
[11] S.P. Levitan and C. Foster, "Finding an Extremum in a Network," *Proc. Ninth Int'l Symp. Computer Architecture*, pp. 321-325, Apr. 1982.
[12] C. Martel and W. Moh, "Optimal Prioritized Conflict Resolution on a Multiple Access Channel," *IEEE Trans. Computers*, vol. 40, no. 10, pp. 1,102-1,108, Oct 1991.
[13] C.U. Martel, "Maximum Finding on a Multiple Access Broadcast Network," *Information Processing Letters*, vol. 52, pp. 7-13, 1994.
[14] C. Martel and T. Vayda, "The Complexity of Selection Resolution, Conflict Resolution and Maximum Finding on Multiple Access Channels," *Proc. Third Int'l Workshop Parallel Computation and VLSI Theory*, pp. 401-410, Greece, 1988.
[15] W.M. Moh, Y.J. Chien, T.-S. Moh, and C.U. Martel, "Prioritized Conflict Resolution on a Multiple Access Channel Using Control Mini-Slots," *Proc. Seventh Int'l Conf. Parallel and Distributed Computing Systems*, pp. 214-221, Las Vegas, Oct. 1994.
[16] W.M. Moh, Y.-J. Chien, T.-S. Moh, and C.U. Martel, "Prioritized Conflict Resolution on Multiple Access Broadcast Networks: Algorithms and Performance Evaluations," *Int'l J. Computer Systems Science and Eng.*, vol. 10, no. 4, pp. 234-243, Oct. 1995.
[17] L. Merakos and D. Kazakos, "Multiaccess of a Slotted Channel Using a Control Mini-Slot," *Proc. Int'l Conf. Comm.*, pp. C5.3.1-5.3.6, 1983.
[18] R. Metcalfe and D. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," *Comm. ACM*, vol. 19, no. 7, pp. 395-404, July 1976.
[19] M. Molle, "A Simulation Study of Retransmission Strategies for the Asynchronous Virtual Time CSMA Protocol," *Proc. Performance 83: Int'l Symp. Computer Performance Modeling, Measurement, and Evaluation*, pp. 305-317, May 1983.
[20] B. Mukherjee, A.C. Lantz, N. Matloff, and W. Moh, "Dynamic Control of the p_f-Persistent Protocol Using Channel Feedback," *Proc. IEEE INFOCOM '89*, pp. 858-865, Ottawa, Canada, Apr. 1989.
[21] W.M. Moh, "Distributed Algorithms for Multiple Access Broadcast Networks and Their Applications," PhD dissertation, Division of Computer Science, Univ. of California, Davis, Dec. 1991.
[22] W.M. Moh, C.U. Martel, and T. Moh, "A Dynamic Solution to Prioritized Conflict Resolution on a Multiple Access Channel," *Proc. 1993 Inte'l Conf. Parallel and Distributed Systems*, pp. 414-418, Taipei, Dec. 1993.
[23] B. Mukherjee, and J. Meditch, "The p_f-Persistent Protocol for Unidirectional Broadcast Bus Networks," *IEEE Trans. Comm.*, vol. 36, pp. 1,277-1,286, Dec. 1988.
[24] D. Raychaudhuri, "Announced Retransmission Random Access Protocols," *IEEE Trans. Comm.s*, vol. 33, no. 11, pp. 1,183-1,190, Nov. 1985.
[25] D. Towsley and P.O. Vales, "Announced Arrival Random Access Protocols," *IEEE Trans. Comm.*, vol. 35, no. 5, pp. 513-915, May 1987.
[26] F.A. Tobagi, "Multiaccess Protocols in Packet Communication Systems," *IEEE Trans. Comm.*, vol. 28, no. 4, pp. 468-488, Apr. 1980.
[27] F.A. Tobagi, "Carrier Sense Multiple Access with Message-Based Priority Functions," *IEEE Trans. Comm.*, vol. 30, no. 1, pp. 185-200, Jan. 1982.
[28] T. Vayda, "On the Complexity of Distributed Algorithms for Multiple Access Broadcast Networks," PhD dissertation, Division of Computer Science, Univ. of California, Davis, June 1989.
[29] B. Wah and J. Juang, "Resource Scheduling for Local Computer Systems with a Multiaccess Network," *IEEE Trans. Computers*, vol. 34, no. 12, pp. 1,144-1,157, Dec. 1985.
[30] D.E. Willard, "Log-Logarithmic Selection Resolution Protocols in a Multiple Access Channel," *SIAM J. Computing*, vol. 15, no. 2, pp. 468-477, May 1986.
[31] W. Xu and G. Campbell, "A Distributed Queueing Random Access Protocol for a Broadcast Channel," *Proc. ACM SIGCOMM '93*, San Francisco, Sept 1993, or *ACM Computer Comm. Review*, vol. 23, no. 4, pp. 270-278, Oct. 1993.
[32] M. Molle and G. Watson, "100 base-T/IEEE 802.12/Packet Switching," *IEEE Comm.*, pp. 64-73, Aug. 1996.
[33] F.E. Ross, A. Nexion, and D.R. Vaman, "IsoEthernet: An Integrated Service LAN," *IEEEComm.*, pp. 73-84, Aug. 1996.

A Multiple-Sequence Generator Based on Inverted Nonlinear Autonomous Machines

Chung-Len Lee, *Member, IEEE*, and Meng-Lieh Sheu

Abstract—A new multiple-sequence generator scheme to generate a set of deterministic *ordered* sequence of patterns followed by random patterns is presented in this paper. This scheme is based on an inverted nonlinear autonomous machine which utilizes a two-dimension-like LFSR with nonlinear inverters. A systematic procedure is also presented to obtain the autonomous machine which is more regular in the structure and utilizes less hardware. The generated deterministic sequence of patterns, which may have *ordered* and *repeated* patterns, and the random patterns are applicable to *sequential circuit testing*.

Index Terms—Multiple-sequence generator, deterministic ordered sequence generation, random pattern generation, autonomous machine, linear feedback shift register, sequential circuit testing.

1 Introduction

A random sequence generator is employed in many applications. The design of a random sequence generator has been intensively studied [1], [2], [3], [4]. Linear feedback shift registers (LFSRs), which can generate a sequence with good randomness properties, are commonly used as the core of the sequence generators.

To produce a set of deterministic patterns(vectors) as soon as possible in the generated random sequence is required in several applications, such as a test pattern generator for efficiently testing circuits. Many approaches have been proposed to embed deterministic patterns into the random sequence [5], [6], [7], [8], [9]. Akers and Jansz [5] proposed a technique, where the pattern generator is constructed by a binary counter and XOR arrays, to embed selected patterns into the random sequence. Hellebrand et al. [6], [9] proposed a scheme based on reseeding of multiple-polynomial LFSR to efficiently generate deterministic test-cubes. These methods only select a pattern (or a test-cube) as a seed to construct a random pattern generator to cover as many unselected deterministic patterns as possible. Besides, the recurrent patterns are not considered in these methods. Dufaza and Cambon [7] proposed a modified LFSR to produce a set of deterministic patterns followed by pseudo-random patterns. Boubezri and Kaminska [8] used a cellular automata to construct a test vector generator to include the precomputed test vectors. Yet, no approach, except the trivial store-pattern method, can generate a set of patterns in a deterministic *ordered* sequence.

A multiple-sequence generator scheme is proposed in this paper to generate a deterministic sequence of patterns, which may have *ordered* and *recurrent* patterns, followed by random patterns. This scheme is applicable to the sequential circuit testing where ordered and recurrent patterns are required. In contrast to the conventional methods, the proposed scheme is a two-dimension-like LFSR and use the inverters to obtain inverted signals, but does not change much the regularity and linearity of the machine structure. Hence, this increases the solution space of constructing

- *C.-L. Lee is with the Department of Electronics Engineering, National Chiao-Tung University, Taiwan, Republic of China..*
- *M.-L. Sheu is with the Chip Implementation Center, National Science Council, Taiwan, Republic of China. E-mail sheu@mbox.cic.edu.tw.*

Manuscript received Apr. 28, 1994.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C95157.

the machine, yet linear methods are still applicable, and minimizes the hardware cost.

In the next section, the two-dimension-like inverted nonlinear autonomous machine is depicted. The generation of multiple sequences by the proposed machine is described in Section 3. To synthesize the machine with less hardware, a procedure based on time frame expansion is developed in Section 4. Experimental results on the test sets of some ISCAS benchmarks [10], [11] are reported in Section 5. Conclusions are made in the last section.

2 Inverted Nonlinear Autonomous Machine

A linear autonomous machine can be modeled as a synchronous sequential circuit without inputs, which is composed of linear logic gates (XORs) and flip-flops(FFs), as shown in Fig. 1. The FFs store the states. The XORs are used to derive the next states and outputs. The states and outputs have a repeated cycle which depends on the machine structure and initial state.

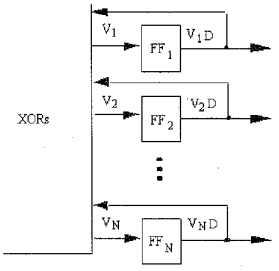

Fig. 1. A linear autonomous machine.

In the autonomous machine, V_i represents the input signal of the ith FF, and V_iD represents a clock delay of V_i (i.e., V_iD is the output of the ith FF). Since the input to each FF is a linear combination (XOR-sum) of outputs of all FFs. Thus, the following N equations are available for representing the linear relations of these signals [2],

$$\mathbf{V}_i = \sum_{j=1}^{N} \mathbf{a}_{ij}\mathbf{V}_j\mathbf{D}, \qquad i = 1 \sim N. \tag{1}$$

The summation and addition are XOR-sum in (1) and the rest part of this paper. And, a_{ij} represents the connection: If $a_{ij} = 0$, no connection exists; and if $a_{ij} = 1$, the output of the jth FF is connected to the input of the ith FF through the XOR gates.

We propose a two-dimension-like autonomous machine, as shown in Fig. 2 and denoted as INLAM(N, M), which is constructed by N number of M-stage feedback shift registers. An M-stage shift register can be built by shifting a signal, V_i, M times and storing each shifted signal into a new FF, respectively, where each FF stores a signal, V_iD^k ($k = 1 \sim M$), representing the kth delay signal of V_i. It can be regarded as an M-time-expansion on each FF in Fig. 1. In this machine, the signals of the first stage of each shift register can be represented by the following equations :

$$\mathbf{V}_i = \sum_{j=1}^{N}\sum_{k=1}^{M} \mathbf{a}_{ijk}\mathbf{V}_j\mathbf{D}^k, \qquad i = 1 \sim N; \tag{2}$$

where $a_{ijk} \in \{0, 1\}$ representing the connection: If $a_{ijk} = 0$, there is no connection; and if $a_{ijk} = 1$, there is an output of the kth FF of signal V_j, i.e., V_jD^k, connected to the input of the first FF of the signal V_i through an XOR tree.

If inverter or XNOR gates are allowed in the inputs of shift registers, the machine becomes an inverted nonlinear autonomous machine. Then, for the inputs with an inverted signal, (2) can be rewritten as

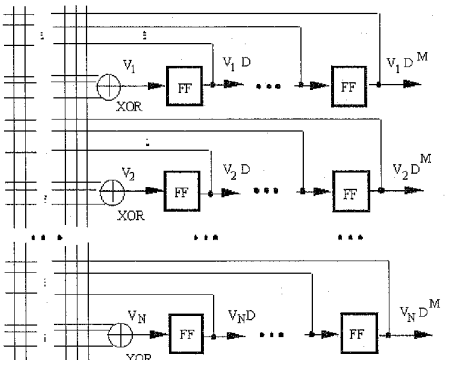

Fig. 2. A two-dimension-like autonomous machine.

$$\overline{\mathbf{V}}_i = \sum_{j=1}^{N}\sum_{k=1}^{M} \mathbf{a}_{ijk}\mathbf{V}_j\mathbf{D}^k, \tag{3-a}$$

or

$$\mathbf{V}_i = \sum_{j=1}^{N}\sum_{k=1}^{M} \mathbf{a}_{ijk}\mathbf{V}_j\mathbf{D}^k + 1, \tag{3-b}$$

since both (2) and (3-a) are linear equations. The possibility of finding the relation of a signal V_i with other signals is doubled with the aid of the inverter. Hence, finding the relation of a signal V_i with other signals becomes easier. Then we can combine (2) and (3-b) into :

$$\mathbf{V}_i = \sum_{j=1}^{N}\sum_{k=1}^{M} \mathbf{a}_{ijk}\mathbf{V}_j\mathbf{D}^k + \mathbf{C}_i \quad , i = 1 \sim N; \tag{4}$$

where, if $C_i = 0$, the number of inverted signals contributing to V_i is even, XOR gate is used in this case; and if $C_i = 1$, the number of inverted signals contributing to V_i is odd, XNOR gate is used.

3 Generation of Deterministic Ordered Multiple Sequences by Using Inverted Nonlinear Autonomous Machines

An inverted nonlinear autonomous machine is capable of generating a set of deterministic patterns which allow ordered sequence and recurrent patterns. The problem of generating such patterns by using an inverted nonlinear autonomous machine can be stated as follows :

> "Build a multiple-sequence generator by using an inverted nonlinear autonomous machine to generate a sequence of N-bit-wide patterns with their first L patterns matching a deterministic ordered sequence, followed by random patterns. The size of each shift register or/and XOR gates used are kept minimal."

Each L-length bit-sequence in the above problem can be regarded as a signal function. N signal functions are to be generated from the constructed autonomous machine. Finding the dependency of these signal functions can significantly reduce the complexity of the constructed autonomous machine. It is generally hard to directly find the dependency of these functions simultaneously. However, if the two-dimension-like inverted nonlinear autonomous machine scheme is to be used, the problem can be apparently solved by using N number of L-bit cyclic shift registers to store all the N sequences. For these N shift registers, the dependency of each signal with other signals and their delay signals can be found by a systematic method which is discussed in the next section. In general, a large amount of signal dependency can be found and the N number of L-bit cyclic shift registers can be reduced to a closely-interconnected shift register cluster which is the multiple sequences generator.

An example is used to demonstrate the above approach. In Fig. 3a, six 4-bit patterns are needed to be generated in the deterministic order. Four 6-bit shift registers can be used to store all the patterns. However, four 3-bit inverted LFSRs can be used to generate these four sequences respectively as shown in Fig. 3b. A closely-interconnected shift register cluster, which employs the two-dimension-like inverted nonlinear autonomous machine scheme by using less flip-flops, can be derived as shown in Fig. 3c. The signals V_1, V_2, V_3, and V_4 can be expressed to be

$$V_1 = V_2 + V_1D \tag{5-a}$$

$$V_2 = V_3 + V_2D \tag{5-b}$$

$$V_3 = V_1D + V_3D + V_4D \tag{5-c}$$

$$V_4 = \overline{V_1 + V_1D} \tag{5-d}$$

respectively. The derivation of these equations are further demonstrated in the next section.

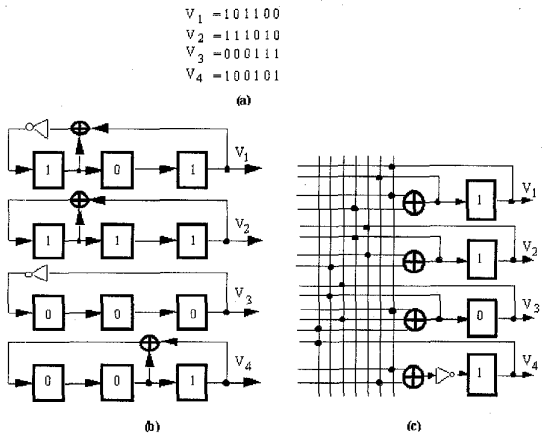

Fig. 3. (a) A set of 4-bit wide 6-length sequences, (b) generated by four independent inverted LFSRs, (c) and by an inverted nonlinear autonomous machine.

4 Synthesis Process

The problem of finding an inverted nonlinear autonomous machine, which is capable of generating N-bit wide random patterns with first L patterns matching a given N-bit wide L-length ordered sequence, can be formulated as follows :

Given a set of N-bit wide L-length multiple bit-sequences,

$$V_1: b_{11}, b_{12}, b_{13}, \ldots, b_{1L}$$
$$V_2: b_{21}, b_{22}, b_{23}, \ldots, b_{2L}$$
$$\ldots \qquad \ldots$$
$$V_N: b_{N1}, b_{N2}, b_{N3}, \ldots, b_{NL}$$

It is to find a construction of an INLAM(N, m), where V_1, V_2, ..., V_N are its outputs to generate its first L patterns exactly matching the order of the given sequences, and m is the size of each shift register. In the INLAM(N, m), the N number of output sequences can be expressed in the form of (4), respectively, i.e., the N signals, $V_i(i = 1 \sim N)$, can be represented in the following equations :

$$\mathbf{V}_i(\mathbf{n}) = \sum_{j=1}^{N} \sum_{k=1}^{m} \mathbf{a}_{ijk} \mathbf{V}_j(\mathbf{n} - k) + \mathbf{C}_i \quad , i = 1 \sim N,\ n = (m+1) \sim L \tag{6}$$

where $V_i(n) = b_{in}$, $v_j(n-k) = v_j(n)D^k = b_{j(n-k)}$. The larger the number m is, the easier to solve the (6), yet the higher the hardware overhead.

It is to find a smaller m such that the (6) has solutions. Every coefficient a_{ijk} and C_i ,which satisfy (6) so that each V_1 can be represented by a combination of other signals and their delay signals, can be used to construct an inverted nonlinear autonomous machine.

A procedure listed in the following is used to solve (6). The procedure starts by first trying to find the linear dependency of each signal (or its inverted signal), one by one with other signals without delays. If a signal (or its inverted signal) is found to be linear dependent on other signals, this signal can be directly constructed from other dependent signals. The Solve_Equations procedure is used to solve the linear dependency of these equations where only binary variables and XOR sum operations are allowed. If all signals are examined and there remain unsolved signals, one more time frame is expanded, i.e., each signal is given one additional delay to be additional signal sources for unsolved signals. Repeat this time frame expansion on each unsolved signal. This procedure stops in the worst case when at most (L – 1) time frames are expanded. For this case, the unsolved signal is constructed by an inverted (L – 1)-bit cyclic shift register. The procedure is as follows:

```
/* procedure to find the dependency among a set of bit sequences where */
/* L is the length of each sequence, N is the number of bit sequences */
Find_Dependency;
{
for ( m = 0 to (L-1) ){              /* m is the index count of time frame expansions */
    if ( all signals are solved ) return FIND;
    for ( each unsolved signal Vi ){
        Build_Equations;
        if ( Solve_Equations(L-m) = SOLVED){
            mark signal Vi SOLVED;
            }
         else {
            Build_Inverted_Equations;      /* Inverting the equation's value */
            if ( Solve_Equations(L-m) = SOLVED ){
                mark signal Vi SOLVED;
                };
            };
        };      /* next unsolved signal */
    };      /* next time frame */
}      /* end */
```

The procedure builds (L – m) linear equations for an unsolved signal V_i, after m time frame expansions. In these equations, there are (mN + N – 1) number of a_{ijk}s. So if m increases by one, the number of equations will decrease by one and the number of a_{ijk}s will increase by N. The linear dependency becomes easier to be found with more a_{ijk}s on less number of equations. The procedure does not guarantee to always achieve the optimal solution, but the solution with a minimal time expansions can be very often found.

The example in Fig. 3a is used for demonstrating this procedure: Each signal is first checked to see whether it is a linear combination of other signals for $m = 0$. For sequence V_1, the following equations are built. Namely,

$$V_1(1) = a_{120}V_2(1) + a_{130}V_3(1) + a_{140}V_4(1) \tag{7-a}$$

$$V_1(2) = a_{120}V_2(2) + a_{130}V_3(2) + a_{140}V_4(2) \tag{7-b}$$

$$\ldots \qquad \ldots \quad \ldots$$

$$V_1(6) = a_{120}V_2(6) + a_{130}V_3(6) + a_{140}V_4(6) \tag{7-f}$$

After substituting the values of $V_i(n)$s into the above equations, the above equations become

$$1 = a_{120} + a_{140} \tag{8-a}$$

$$0 = a_{120} \tag{8-b}$$

$$1 = a_{120} \tag{8-c}$$

$$1 = a_{130} + a_{140} \tag{8-d}$$

TABLE 1
COMPARISON OF OUR SYNTHESIZED MULTIPLE TEST SEQUENCES GENERATOR WITH THE DESIGN OF PVG IN [7]

	No. of FFs	No. of Gates	Ordered Sequence	Additional Patterns	Design Regularity
This work	12	23	Yes	0	Yes
PVG[7]	18	40	No	2	No

TABLE 2
SYNTHESIS RESULTS ON SOME ISCAS BENCHMARKS

	#inputs	#patterns	#expansions	#FFs	#XORs	time(sec)**
74181	14	16	2	24	49	0.2
c432	32	44	3	81	282	482
c499	32	60	8	228	247	564
c880	48	34	4	172	204	316
c1355*	32	64	10	311	195	507
c1908*	32	64	7	209	278	414
c2670	64	63	8	496	584	598
c3540*	48	64	6	273	338	379
c5315*	128	64	5	628	884	744
c6288	32	30	4	120	103	86
c7552*	128	64	4	517	786	655
s27	4	15	3	12	18	0.1
s298*	3	64	24	70	28	168
s344*	9	64	6	52	84	338
s713*	32	64	6	192	186	447
s953	16	16	1	16	82	1.4
s1423	16	36	3	46	86	216

** partial test set*
*** SUN Sparc station 2*

$$0 = a_{120} + a_{130} \quad (8\text{-e})$$

$$0 = a_{130} + a_{140} \quad (8\text{-f})$$

A conflict exists in (8-b) and (8-c). Therefore, there is no solution for these equations. Since the derived equations for sequences V_2, V_3, and V_4 also have the conflicting condition, no sequence can be directly derived from the combination of other sequences without delay. Then, one more time frame expands, i.e., $m = 1$, to get the V_iD signals. To solve for V_1, the following equations are built :

$$V_1(2) = a_{111}V_1(1) + a_{121}V_2(1) + a_{131}V_3(1) + a_{141}V_4(1) + a_{120}V_2(2) + a_{130}V_3(2) + a_{140}V_4(2) \quad (9\text{-a})$$

$$V_1(3) = a_{111}V_1(2) + a_{121}V_2(2) + a_{131}V_3(2) + a_{141}V_4(2) + a_{120}V_2(3) + a_{130}V_3(3) + a_{140}V_4(3) \quad (9\text{-b})$$

. . . . . .

$$V_1(6) = a_{111}V_1(5) + a_{121}V_2(5) + a_{131}V_3(5) + a_{141}V_4(5) + a_{120}V_2(6) + a_{130}V_3(6) + a_{140}V_4(6) \quad (9\text{-e})$$

Substituting the values of $V_i(n)$s into the above equations, the following equations are obtained:

$$0 = a_{111} + a_{121} + a_{141} + a_{120} \quad (10\text{-a})$$

$$1 = a_{121} + a_{120} \quad (10\text{-b})$$

$$1 = a_{111} + a_{121} + a_{130} + a_{140} \quad (10\text{-c})$$

$$0 = a_{111} + a_{131} + a_{141} + a_{120} + a_{130} \quad (10\text{-d})$$

$$0 = a_{121} + a_{131} + a_{130} + a_{140} \quad (10\text{-e})$$

There are many solutions for these equations. A solution is $a_{120} = a_{111} = 1$ and other a_{ijk}s are equal to zero. For the solution, V_1 can be represented by XOR sum of V_2 and V_1D, i.e., $V_1 = V_2 + V_1D$. The solutions for other sequences are obtained in a similar way and listed in (5). The inverted nonlinear autonomous machine to generate these sequences has been shown in Fig. 3c.

5 EXPERIMENTAL RESULTS

For comparison, the example sequences in [7] is synthesized by using our proposed scheme to form a multiple sequences generator. The sequence generator is required to first generates the following 6-bit 16-length ordered sequences followed by pseudo-random patterns:

N = 6, L = 16

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	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

note: pattern 12 is a recurrent pattern of pattern 4.

The synthesized results are

$$V_1 = (V_2 + V_4 + V_5) + V_2D + V_3D^2 \quad (11\text{-a})$$

$$\overline{V}_2 = V_5 + V_6D + (V_1 + V_2)D^2 \quad (11\text{-b})$$

$$V_3 = (V_2 + V_3 + V_4 + V_5)D + (V_1 + V_2 + V_6)D^2 \quad (11\text{-c})$$

$$V_4 = (V_4 + V_5 + V_6)D + (V_2 + V_4)D^2 \quad (11\text{-d})$$

$$\overline{V}_5 = (V_1 + V_2)D + (V_5 + V_6)D^2 \quad (11\text{-e})$$

$$V_6 = (V_1 + V_4) + V_4D + V_2D^2 \quad (11\text{-f})$$

A comparison of the synthesized sequence generator to the design of PVG method in [7] is listed in Table 1. For our sequence generator, as shown in Fig. 4, two time frame expansions are necessary for solving the dependency among these six signals, so 12 flip-flops are needed to store the delay signals. Six multiple-input

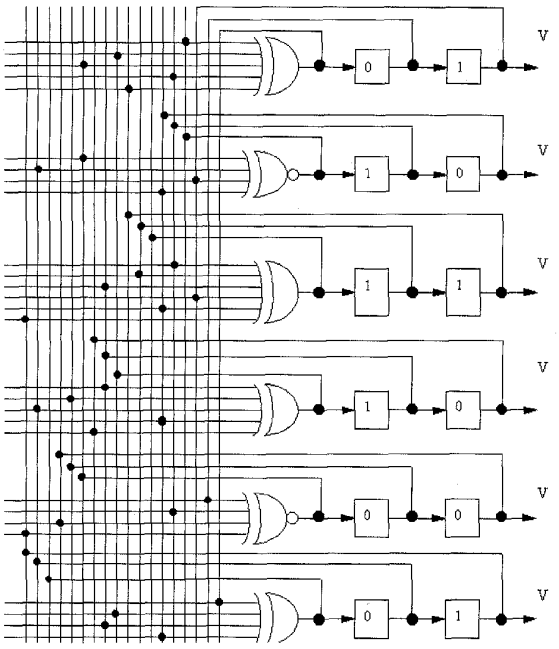

Fig. 4. The synthesized multiple-sequence generator to generate the example test sequences in [7].

XOR gates (or 23 two-input XOR gates) are used to construct this multiple sequences generator to generate a set of sequences where their first 16 patterns match *exactly* the deterministic sequences followed by pseudo-random patterns. And, the period of the generated pseudorandom sequences is 1,260. For the design of PVG method, the generated patterns were not in the deterministic order, which had to be rearranged and two additional patterns needed in order to get an optimal solution. Besides, the PVG design used a counter to control LFSRs, and the connections between LFSRs are more complicated and irregular than our scheme.

A program has been written to synthesize a multiple sequences generator circuit from a set of given deterministic ordered sequences according to the procedure mentioned in Section 4. Table 2 lists the results on the synthesized generator circuits for the test sequence for some ISCAS benchmark circuits. The test sequences were generated for 100% fault efficiency by SLOPE [12] for combinational circuits and high fault coverage by STG3 [13] for sequential circuits, respectively. For a reasonable run-time, we only selected the foremost part of the test sequences with an adequate length. The table lists the number of patterns (i.e., the length of the selected deterministic test sequences), the maximal number of time frame expansions, the number of flip-flops and the number of 2-input XOR gates needed to construct the generators for each circuit. Also, the times spent for synthesizing each generator are listed. In general, the number of FFs is equal to the number of inputs multiplied by the number of time frame expansions, but some FFs whose outputs are not dependent signals of other signals can be removed. Also, the number of 2-input XOR gates may be further reduced by sharing the same XORs with the same input signals. The synthesizing time is much more dependent on the patterns of a deterministic sequence. In general, it is roughly proportional to the number of inputs and the number of deterministic patterns.

6 Conclusions

An inverted nonlinear autonomous machine, which comprises flip-flops, XOR gates and nonlinear inverters, to generate a set of patterns which may have *ordered* sequence and *recurrent* patterns, has been presented in this paper. The machine is a two-dimension-like feedback shift register configuration and can generate specified multiple deterministic *ordered* sequences, followed by random patterns. An iterative time frame expansion method has been presented to find the dependency of each sequence with other sequences and their delay sequences. The problem of finding the construction of the proposed machine is formulated as to solve a set of $(L - m)$ linear equations with $(mN + N - 1)$ variables, where N is the number of the sequences, L is the length of a sequence and m is the size of each shift register. From the experimental results, the obtained autonomous machine, which is more regular and uses less hardware, is more applicable to non-scan *sequential circuit testing* where the *ordered* sequence and *recurrent* patterns are necessary to test the circuits.

Acknowledgments

This work was supported in part by the National Science Council, Taiwan, Republic of China, under contracts NSC-82-0404-E009-183 and NSC-83-E009-028.

References

[1] S.W. Golumb, *Shift Register Sequences*. Laguna Hills, Calif.: Aegean Park Press, 1982.
[2] W.W. Peterson and E.J. Weldon, *Error-Correcting Codes*, 2nd ed. The Colonial Press, 1972.
[3] E.J. McCluskey, *Logic Design Principles: with Emphasis on Testable Semicustom Circuits*. Englewood Cliffs, N.J.: Prentice Hall, 1986.
[4] M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, chapter 11, pp. 457-477. Computer Science Press, 1990.
[5] S.B. Akers and W. Jansz, "Test Set Embedding in a Built-In Self-Test Environment," *Proc. Int'l Test Conf.*, pp. 257-263, 1989.
[6] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns through Reseeding of Multiple-Polynomial LFSRs," *Proc. Int'l Test Conf.*, pp. 120-129, 1992.
[7] C. Dufaza and G. Cambon, "LFSR-based Deterministic and Pseudo-Random Test Pattern Generator Structures," *Proc. European Test Conf.*, pp. 27-34, 1991.
[8] S. Boubezari and B. Kaminska, "Cellular Automata Synthesis Based on Pre-Computed Test Vectors for Built-In Self-Test," *Proc. Int'l Conf. Computer-Aided Design*, pp. 578-583, 1993.
[9] S. Venkataraman, J. Rajski, S. Hellebrand, and S. Tarnick, "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers," *Proc. Int'l Conf. Computer-Aided Design*, pp. 572-577, 1993.
[10] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Designs and a Special Translator in Fortran," *Proc. Int'l Symp. Circuits and Systems*, 1985.
[11] F. Brglez et al., "Combinational Profiles of Sequential Benchmark Circuits," *Proc. Int'l Symp. Circuits and Systems*, pp. 1,929-1,934, 1989.
[12] S.J. Chuang, C.L. Lee , W.Z. Shen, C.W. Jen, and J.E. Chen, "SLOPE: A Test Pattern Generator Based on Stop Line Oriented Path End Algorithm," *Proc. Int'l Symp. Circuits and Systems*, Helsinki, 1988.
[13] W. T. Cheng, "The BACK Algorithm for Sequential Test Generation," *Proc. Int'l Conf. Computer Design*, pp. 66-69, 1988.