V. Conclusions

We have demonstrated in this paper that for deriving an active g_m-C filter from an LC ladder, both the element replacement method and operational simulation result in identical circuits. Thus, regardless of an application's requirements, either method may be employed in the design. Any discussion on the relative practical merits of the procedures is, therefore, moot and the designer can use the method which appears more convenient for the initial filter descriptions (only equations, or ladder topology with element values), irrespective of any concerns with practical performance issues. Note that throughout the development of an active g_m-C ladder, all transconductors can be assumed to be identical—a constraint that must be relaxed only if gain scaling is used.

References

[1] Y. Tsividis and J. O. Voorman, Eds., *Integrated Continuous-Time Filters: Principles, Design and Implementations*. New York: IEEE Press, 1993.

[2] A. I. Zverev, *Handbook of Filter Synthesis*. New York: Wiley, 1967.

[3] R. Schaumann, M. S. Ghausi, and K. R. Laker, *Design of Analog Filters: Passive, Active RC and Switched Capacitor*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[4] K. Martin and A. S. Sedra, "Design of signal-flow-graph (SFG) active filters," *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 185–195, 1978.

[5] M. A. Tan and R. Schaumann, "Simulating general-parameter *LC*-ladder filters for monolithic realizations with only transconductance elements and grounded capacitors," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 299–307, Feb. 1989.

[6] A. S. Sedra and P. O. Brackett, *Filter Theory and Design: Active and Passive.* Portland, OR: Matrix, 1978.

Test Pattern Generation and Signature Analysis for Burst Errors

Rajendra S. Katti

***Abstract*— In testing certain systems, checking for burst errors is important. This is due to the fact that errors are confined to a certain number of bits. If signature analysis is used to test a circuit then the testing capabilities depend on the polynomial that defines the linear feedback shift register (LFSR) used in the test. In this paper we show that the LFSR that is suitable for checking for burst errors is not suitable for test pattern generation. We propose a method to modify the LFSR that performs signature analysis efficiently into a nonlinear feedback shift register (NLFSR) that is suitable for test pattern generation.**

***Index Terms*— Burst errors, error detection, shift registers, signature analysis, test pattern generation.**

I. Introduction

Linear feedback shift registers can be used to compress a stream of test result data [1]. Signatures can be created from test result data streams by feeding the data into an LFSR. After the data have been clocked through the LFSR, a residue of the data is left in the shift register. This residue is unique to the input data stream and represents its "signature." Another data stream that differs by only one bit from a previous data stream may have a signature that is different from the previous one. Fig. 1 shows an example of a signature analyzer circuit. This circuit consists of an n-bit shift register. Feedback from certain points in this shift register are XORed with the test data stream at each clock cycle and fed back into the shift register. Two signals—START and STOP—mark the start and end of the test data stream. The signature is then used to differentiate a faulty data stream from a fault-free one.

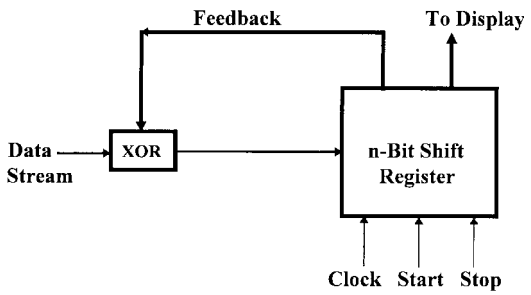

Fig. 1. A signature analyzer circuit.

Other techniques have also been used to compress a stream of test result data [2]. One such technique involves the counting of 0-to-1 and 1-to-0 transitions in the data stream. In this paper we are concerned only with the use of LFSR's for compression of a stream of test result data.

In built-in testing, a set of pseudorandom test patterns are input to the circuit under test and signature analysis is used on the output data stream for detecting errors in the circuit. The test hardware consists of built-in logic-block observers (BILBO). A BILBO is a multipurpose test module which can be reconfigured to function as an input test pattern generator or an output signature analyzer. In this paper, we design a BILBO that can efficiently detect burst errors in the output data stream of the circuit under test.

LFSR's can function as test pattern generators and signature analyzers and hence form a major part of a BILBO. LFSR's are specified by polynomials. Such a description of LFSR's is further explained in the next section. LFSR's used as signature analyzers for the detection of burst errors can be specified by polynomials with certain properties. Such polynomials however result in LFSR's that are not suitable for test pattern generation. In this paper we propose the use of a nonlinear feedback shift register (NLFSR) for test pattern generation. This NLFSR can be easily modified for use as a signature analyzer for the detection of burst errors. This property of the NLFSR is used to design a BILBO for burst error detection. The importance of burst error detection in microprocessor testing has been described in [1].

In Section II we describe LFSR's for the detection of burst errors. In Section III we describe the proposed NLFSR's for test pattern generation. Section IV describes a BILBO for burst error detection, and Section V gives some examples. Section VI describes a technique to modify the NLFSR's for use in exhaustive testing.

II. LFSR's for Burst Errors

LFSR's are attractive for generating a sequence of binary words. Applications of LFSR's include error-correcting codes [3], pseudorandom sequence generation [4], test-pattern generation in VLSI circuits [5], [8], and program counters [6]. The characteristic poly-

Manuscript received January 5, 1996; revised July 31, 1996. This paper was recommended by Associate Editor B. W. Lee.

The author is with the Department of Electrical Engineering, North Dakota State University, Fargo, ND 58105 USA (e-mail: katti@plains.nodak.edu).

Publisher Item Identifier S 1057-7130(98)00779-4.

nomial of an n-bit LFSR has the form $p(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1x^1 + 1$. The c_i are either 1 or 0. The period of an LFSR is the smallest m such that

$$x^m \equiv 1[\bmod p(x)].$$

The maximum period of an n-bit LFSR is $2^n - 1$. A characteristic polynomial that results in the maximum period is called a primitive polynomial. All primitive polynomials are irreducible. The converse of the above statement, however, is not always true.

Cyclic codes are derived from generator polynomials. Encoding is performed by multiplying the message polynomial by the generator polynomial. Decoding is performed by dividing the received polynomial by the generator polynomial. Both encoding and decoding can be performed by LFSR's that are specified by the generator polynomial of the code. We state the following theorems from [7] about generator polynomials. Note that the theorems are true if $p_1(x)$ is primitive (defined in [3]).

Theorem 1: A cyclic code generated by $p(x) = (x + 1)p_1(x)$ detects all single, double, and triple errors if the length of the code is no greater than the period of $p_1(x)$.

A cyclic code generated by a polynomial that has $(x + 1)$ as one of its factors can also detect all odd number of errors.

Theorem 2: A cyclic code generated by $p(x) = (x^n + 1)p_1(x)$ detects any combination of two bursts, provided $(n+1)$ is equal to or greater than the sum of the lengths of the bursts, $p_1(x)$ is irreducible and of degree at least as great as the length of the shorter burst, and provided the length of the code is no greater than the least common multiple of n and the period of $p_1(x)$.

Theorem 3: Any cyclic code generated by a polynomial of degree $(n-k)$ detects any burst error of length $(n-k)$ or less. The fraction of bursts of length $b > (n - k)$ that are undetected is $2^{-(n-k)}$ if $b > (n - k + 1), 2^{-(n-k-1)}$ if $b = (n - k + 1)$.

We know that a signature analyzer based on a polynomial that generates a cyclic code with certain error detection capabilities also has the same error detection capabilities [1]. We can therefore conclude from the above theorems that a signature analyzer based on a polynomial with $(x+1)$ as one of its factors has good error detection capabilities. This is especially true for burst errors. For example, a signature analyzer that is an LFSR based on the polynomial $p(x) = (x+1)(x^4 + x+1)$ can detect two bursts of length 2 or less, any odd number of errors, a burst of 5 or less, 93.8% of the bursts of length 6, 96.9% of longer bursts [7]. However, if this same LFSR were used as a test pattern generator, then it would generate only 15 test patterns as $p(x) = (x + 1)(x^4 + x + 1)$ has a period of 15. We will show next that an NLFSR based on the same polynomial will result in a period of 30. Therefore, the NLFSR can generate more test vectors than the LFSR.

Examples of any kind can be constructed by using Theorem 3 above. Let us consider a generator polynomial of degree $1000 = n - k$. Then any burst of length 1000 or less can be detected. The fraction of bursts of length 1001 that cannot be detected is 2^{-999} and the fraction of bursts of length greater than 1001 that cannot be detected is 2^{-1000}.

We will show that the period of the proposed NLFSR based on the polynomial $(x + 1)p(x)$ is twice that of the period of the LFSR with characteristic polynomial $p(x)$. This will result in a BILBO that uses the NLFSR for test pattern generation and the LFSR for signature analysis.

III. NLFSR'S BASED ON POLYNOMIALS WITH $(x+1)$ AS ONE FACTOR

Let us assume that the characteristic polynomial of an LFSR has one factor equal to $(x + 1)$ and all other factors of higher degree.

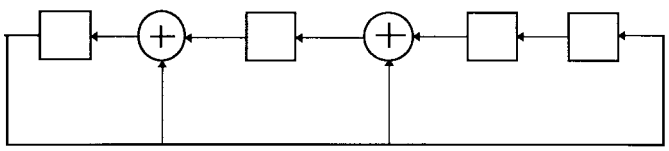

Fig. 2. An LFSR with characteristic polynomial $(x + 1)(x^3 + x + 1)$.

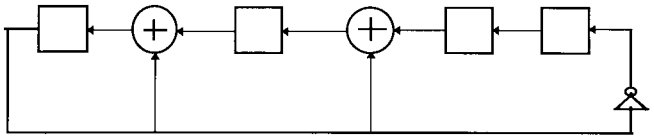

Fig. 3. The LFSR of Fig. 2 with an inverter in its feedback path.

The characteristic polynomial $p(x)$ of such an LFSR is given by the following equation:

$$p(x) = (x + 1)p_1(x)p_2(x)\cdots p_n(x). \tag{1}$$

In the above equation $p(x)$ has $(n + 1)$ distinct factors and all the $p_i(x)$ have degree greater than 1. The period of the above polynomial can be calculated as follows:

$$\begin{aligned}\mathrm{Period}[p(x)] = \mathrm{LCM}(&\mathrm{Period}[(x + 1)], \mathrm{Period}[p_1(x)],\\ &\cdots, \mathrm{Period}[p_n(x)]).\end{aligned} \tag{2}$$

Since $\mathrm{Period}[(x + 1)] = 1$, the above equation can be rewritten as,

$$\mathrm{Period}[p(x)] = \mathrm{LCM}(\mathrm{Period}[p_1(x)], \cdots, \mathrm{Period}[p_n(x)]). \tag{3}$$

We will show next that by adding an inverter in the feedback path of the LFSR, the period of the LFSR can be doubled. Let us demonstrate this first with an example. Consider the LFSR shown in Fig. 2 that has a characteristic polynomial given by $p(x) = (x+1)(x^3+x+1)$. Therefore, $\mathrm{Period}[p(x)] = \mathrm{LCM}(1, 7) = 7$. Fig. 3 shows the same LFSR with an inverter in the feedback path. Notice the inverter changes the input to the right most flip-flop of the LFSR. Unless otherwise stated, we assume that this is the position of the inverter in the feedback path for the rest of the paper. The period of this FSR (feedback shift register) is 14. The new FSR of Fig. 3 is nonlinear. We will refer to the polynomial $p(x)$ on which this nonlinear FSR is based as the constructor polynomial (for a polynomial used to construct a nonlinear FSR). Note that LFSR's are of two types: those that use external XOR's in the feedback path (Fig. 5) and those that use internal XOR's in between flip-flops (Fig. 2). The contents of the FSR of Fig. 3 after each shift are shown below. The first column shows the contents of the LFSR of Fig. 2, and the second and third columns show the contents of the FSR of Fig. 3. The third column is a continuation of the second column:

0001	0001	1010
0010	0011	1000
0100	0111	1100
1000	1111	0100
1101	0010	1001
0111	0101	1110
1110	1011	0000

We now state the following lemmas before stating the main result of this section in Theorem 4. All the lemmas assume that the period of $p(x)$ is P.

Lemma 1: $(x+1)$ does not divide $p(x)$ if and only if the following equation holds:

$$(x^P + x^{P-1} + \cdots + 1) \bmod p(x) = 1 \tag{4}$$

Proof: Let

$$(x^P + x^{P-1} + \cdots + 1) \bmod p(x) = a(x)$$

where $a(x)$ is some polynomial in x. Multiplying both sides of the above equation by $(x+1)$, we get

$$(x+1)(x^P + x^{P-1} + \cdots + 1) \bmod p(x) = (x+1)a(x) \bmod p(x).$$

The above equation can be rewritten as

$$(x^{P+1} + 1) \bmod p(x) = (x+1)a(x) \bmod p(x).$$

Since $p(x)$ has a period of P, $x^P \bmod p(x) = 1$. Therefore, the left-hand side of the above equation is simply $(x+1)$. This implies that

$$(x+1) = (x+1)a(x) \bmod p(x)$$

or

$$(x+1) \equiv (x+1)a(x) (\mathrm{mod}\, p(x)).$$

Since $a(x) = 1$ if and only if $gcd[(x+1), p(x)] = 1$, both the "if" part and the "only if" parts of the lemma are proved. □

Lemma 2: If $(x+1)$ does not divide $p(x)$ then the following equation holds:

$$(x^{P-1} + x^{P-2} + \cdots + 1) \bmod p(x) = 0. \tag{5}$$

Proof: From Lemma 1 we know that

$$(x^P + x^{P-1} + \cdots + 1) \bmod p(x) = 1.$$

Adding x^P on both sides, we get

$$(x^{P-1} + \cdots + 1) \bmod p(x) = 1 + x^P \bmod p(x).$$

This completes the proof as $x^P \bmod p(x) = 1$. □

Lemma 3: If $(x+1)$ divides $p(x)$ then the following equations hold if $p(x)$ is given by (1).

$$(x^P + x^{P-1} + \cdots + 1) \bmod p(x) \neq 1 \tag{6}$$

$$(x^{2P} + x^{2P-1} + \cdots + 1) \bmod p(x) = 1 \tag{7}$$

$$(x^{2P-1} + x^{2P-2} + \cdots + 1) \bmod p(x) = 0. \tag{8}$$

Proof: From Lemma 1 we know that for the following equation to be true:

$$(x^P + x^{P-1} + \cdots + 1) \bmod p(x) = 1$$

$gcd[(x+1), p(x)]$ must equal 1. This implies that the above equation cannot be true if $(x+1)$ is a factor of $p(x)$. This completes the proof of (6). We shall now derive (7). The left-hand side of (7) can be rewritten as

$$x^P(x^P + x^{P-1} + \cdots + 1) + (x^{P-1} + \cdots + 1) \bmod p(x).$$

Since $x^P \equiv 1 \bmod p(x)$ and addition is modulo-2 [in GF(2)], the above polynomial becomes equal to the following:

$$x^P \bmod p(x) = 1.$$

Equation (8) follows from (7) by adding x^{2P} to both sides of (7). □

Theorem 4: Let the characteristic polynomial of an LFSR be given by (1). Let the period of this LFSR be P. The period of the nonlinear FSR with a constructor polynomial given by (1) is $2P$.

Proof: We will refer to the polynomial $p(x)$, on which this nonlinear FSR is based, as the constructor polynomial (for a polynomial used to construct a nonlinear FSR). Let P be the period of a polynomial $p(x)$ with $(x+1)$ as one of its factors. The ith count of the LFSR with characteristic polynomial $p(x)$ is $x^{i-1} \bmod p(x), 1 \leq i \leq P$. The ith count of the FSR with constructor polynomial $p(x)$ with an inverter in its feedback path is $(x^{i-1} + x^{i-2} + \cdots + 1) \bmod p(x), 1 \leq i \leq 2P$. Let the initial contents of the FSR be 1. The contents of the FSR after the ith shift would be

$$(x^i + x^{i-1} + \cdots + x + 1) \bmod p(x).$$

Let M be the period of the nonlinear FSR. Therefore,

$$(x^M + x^{M-1} + \cdots + x + 1) \bmod p(x) = 1.$$

Adding 1 to both sides of the above equation, we get

$$x(x^{M-1} + \cdots + x + 1) \bmod p(x) = 0.$$

This implies that

$$(x^{M-1} + \cdots + x + 1) \bmod p(x) = 0.$$

From the previous three equations, we get

$$x^M \bmod p(x) = 1.$$

Since the LFSR based on the characteristic polynomial $p(x)$ has a period P,

$$x^P \bmod p(x) = 1.$$

Therefore, P must divide M. From Lemma 3, we know that M is not equal to P as

$$(x^P + x^{P-1} + \cdots + x + 1) \bmod p(x) \neq 1.$$

Therefore, the smallest value of M must be $2P$. Therefore,

$$(x^{2P} + x^{2P-1} + \cdots + x + 1) \bmod p(x) = 1.$$

Therefore, the period of the nonlinear FSR is $2P$. □

From the above discussion, we can conclude that an LFSR based on a polynomial with $(x+1)$ as one of its factors is good for signature analysis. The same LFSR can be modified with an inverter in its feedback path to obtain an NLFSR. This NLFSR can then be used for test pattern generation. The design of a BILBO based on the above facts is described in the next section.

IV. Modified BILBO

In this section we will demonstrate the design of a BILBO that is based on the polynomial $p(x) = (x+1)(x^3 + x + 1)$. Fig. 4 shows a BILBO based on the polynomial $p(x)$. Two control inputs B_1 and B_2 determine one of the four modes in which the BILBO operates. These four modes are described below.

Mode 1: $B_1 = 0, B_2 = 1$. All the flip-flops are reset.

Mode 2: $B_1 = 1, B_2 = 1$. The BILBO behaves as a latch. The input data x_1 through x_4 can be simultaneously clocked into the flip-flops and can be read from the Q and $\overline{Q}$ outputs.

Mode 3: $B_1 = 0, B_2 = 0$. The BILBO works as an NLFSR with an inverter in its feedback path. In this mode, the BILBO is a test pattern generator.

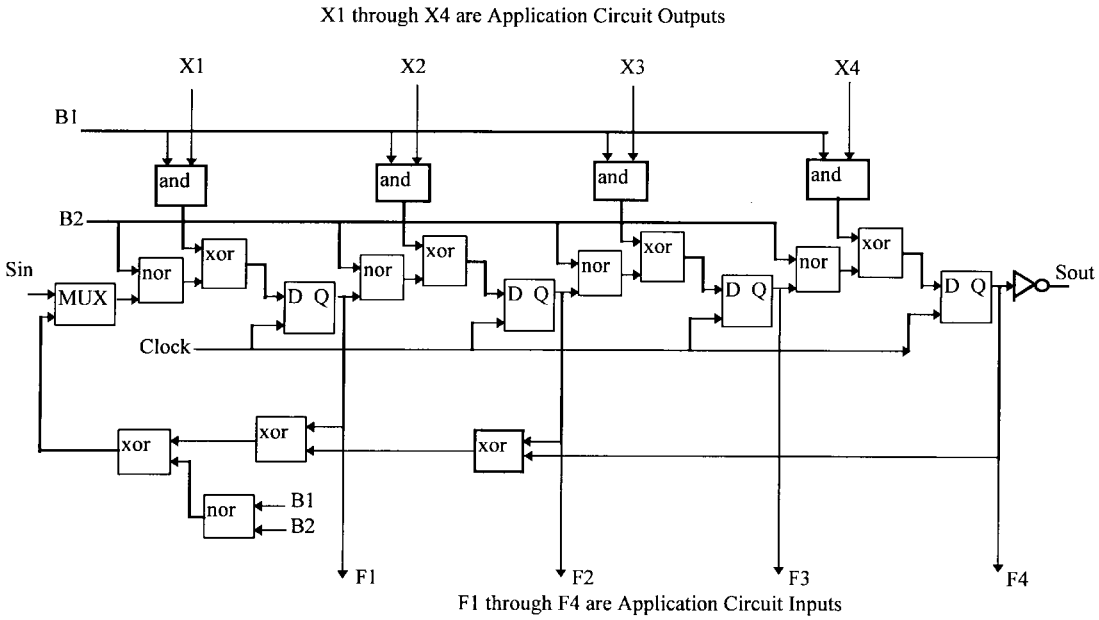

Fig. 4. A BILBO based on the polynomial $(x + 1)(x^3 + x + 1)$.

Mode 4: $B_1 = 1, B_2 = 0$. The BILBO is converted into a multiple input signature register. There is no inverter in its feedback path. The BILBO therefore acts as a regular LFSR based on the polynomial $p(x)$.

The only difference between the above-described BILBO and a conventional BILBO is in Mode 3. The BILBO described above behaves as a nonlinear feedback shift register in this mode. This allows the BILBO to generate test patterns more effectively. When the BILBO acts as a test pattern generator in Mode 3, it generates $2^n - 2$ test patterns, where n is the number of flip-flops in the NLFSR. The BILBO can be easily modified using techniques described in Section VI, to generate all 2^n test patterns for exhaustive testing. The above statement is true only if $p(x) = (x + 1)p_1(x)$ and $p_1(x)$ is primitive. For the BILBO of Fig. 4, $n = 4$, hence 14 test patterns are generated. These test patterns are shown below. The first column shows the counts of the LFSR without an inverter in its feedback path and the second and third columns show the 14 counts of the NLFSR:

0001	0001	1110
1000	0000	1111
1100	1000	0111
0110	0100	1011
1011	0010	1101
0101	1001	0110
0010	1100	0011

It is therefore better to use the NLFSR for test pattern generation. If exhaustive testing is required, then the BILBO can be modified using the technique described in Section VI, to generate all the 16 test patterns. One should note that the modulo-2 sum of the every pair of consecutive counts of the NLFSR results in a count in the LFSR.

V. Examples

Let us consider polynomials $p(x) = (x + 1)p_1(x)$, where $p_1(x)$ is primitive. Let a BILBO be based on such a polynomial $p(x)$. When the BILBO functions as a signature analyzer, it is good at detecting burst errors. When the BILBO functions as a test pattern generator, it can be used for exhaustive testing as it behaves as an NLFSR. One can choose $p_1(x)$ such that $p(x)$ has only four terms. This reduces the number of XOR gates in the feedback path of the BILBO. As an example consider the following 4-term polynomial, $p(x) = (x + 1)(x^{15} + x + 1) = x^{16} + x^{15} + x^2 + 1$. Since the polynomial $x^{15} + x + 1$ is primitive, the BILBO can generate 65 534 test patterns in Mode 3. In Mode 4 the BILBO can be used to detect two bursts of length 2 or less, any number of odd errors, a burst of 16 or less, 99.997% of the bursts of length 17, and 99.998% of longer bursts. The above statement follows from Theorems 1, 2, and 3. Note that since $p(x)$ has only four terms, only two XOR gates are required to implement an LFSR based on $p(x)$.

VI. Exhaustive Testing

In this section we consider modifying an n-stage NLFSR so that it has a period of 2^n. The n-stage NLFSR generates $2^n - 2$ distinct states. For exhaustive testing, it is also necessary to generate the remaining two states. It is possible to reconfigure the NLFSR with some extra logic so that it generates all 2^n states. This NLFSR is referred to as a modified NLFSR. We now describe an approach to design the modified NLFSR with an example. Consider the LFSR that is based on the polynomial $p(x) = (x + 1)(x^3 + x + 1) = x^4 + x^3 + x^2 + 1$. This is shown in Fig. 5. The 7 states or counts that the LFSR generates were described in Section IV. An NLFSR based on the same polynomial with an inverter in its feedback is shown in Fig. 6. The 14 counts that the NLFSR generates were also given in Section IV. The two remaining counts that need to be generated are 0101 and 1010. Notice that these two counts by themselves form a cycle. This means that if the NLFSR is initialized with any one of these two counts, then a single shift in the NLFSR would result in the other count. Therefore, if the NLFSR is initialized with 0101 then a single shift would result in the state of the NLFSR being 1010. Another shift would take the NLFSR back to the state 0101.

This gives us a technique to design the modified NLFSR as follows. Initialize the NLFSR to 0010. Let the NLFSR go through its 14 states. The NLFSR is now in the state 0100. Then remove the inverter from the feedback path such that a shift would take the NLFSR to the state 1010. Insert the inverter back into the feedback path. A shift would now take the NLFSR into the state 0101. Remove the inverter from the feedback path again. A shift in the NLFSR would now take it back to the initial state 0010.

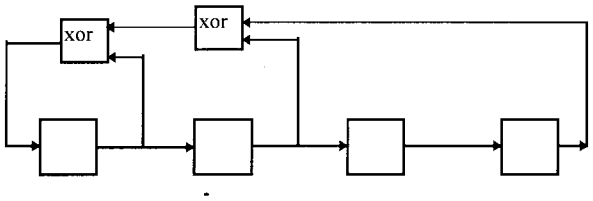

Fig. 5. An LFSR based on the polynomial $(x+1)(x^3+x+1)$.

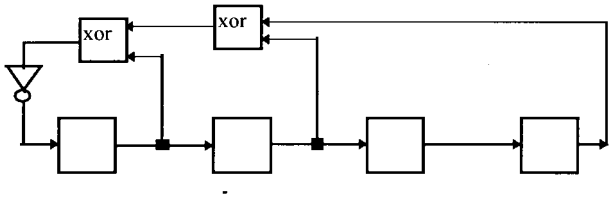

Fig. 6. An NLFSR based on the polynomial $(x+1)(x^3+x+1)$.

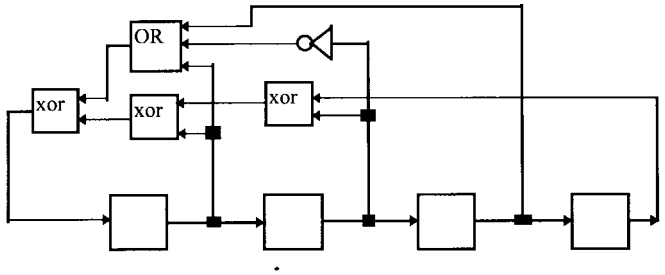

Fig. 7. A modified NLFSR based on the polynomial $(x+1)(x^3+x+1)$.

The removal and insertion of the inverter can be obtained by replacing the inverter by a two input XOR gate. One of the inputs to the gate obtains its inputs from three stages of the NLFSR. This control input converts the XOR gate into an inverter at the appropriate times. This results in the modified NLFSR generating all the 16 test patterns. The modified NLFSR is shown in Fig. 7.

VII. Discussion and Conclusions

We first comment on the quality of the nonlinear FSR (feedback shift register) presented with respect to test pattern generation. This part of the discussion is based on [9]. We will compute the probability of linear dependence according to the procedure suggested in [9]. The results in [9] are valid for any feedback shift register. Consider an m-stage FSR. Let us assume that a circuit under test has k inputs, where $k \leq m$. If we choose k outputs of the FSR as test inputs for the circuit under test, then the probability of linear dependence gives us a measure of how close this test set is to being an exhaustive test set. Let n be the period of the FSR. The probability of linear dependence is given as [9]

$$P(k) = \frac{n(n-1)(n-3)(n-7)\cdots(n+1-2^{k-1})}{n(n-1)(n-2)\cdots(n-k+1)}.$$

Higher values of $P(k)$ imply that the probability of the test set being exhaustive is more. For example, for the FSR of Fig. 5 we obtain $P(3) = 0.8$, and for the FSR of Fig. 6 (the same FSR with an inverter in its feedback), we obtain $P(3) = 0.917$. This implies that the proposed nonlinear FSR has better testing capabilities than the corresponding FSR without an inverter in its feedback. For a 4-stage LFSR with period 15, we obtain $P(3) = 0.923$. This LFSR is slightly better than the proposed FSR. However, this LFSR is not good at detecting burst errors. Thus the proposed FSR's testing capabilities are slightly degraded while having good burst error detection capabilities.

We will now comment on the hardware overhead for the proposed BILBO and compare this to the hardware overhead of a conventional BILBO. Consider an m-stage BILBO. The hardware overhead for a conventional BILBO is a 2-input AND gate, a 2-input NOR gate, and a 2-input XOR gate per stage of the BILBO. A 2-to-1 multiplexer is also required. The hardware overhead for the proposed BILBO is the hardware overhead for a conventional BILBO and a 2-input NOR gate and a 2-input XOR gate. These two gates are added in the feedback path of the proposed BILBO. Thus the only extra hardware needed in the proposed BILBO is these two gates.

We now describe the main contribution of this brief. The brief describes a new nonlinear feedback shift register that can be used for the generation of test patterns or for signature analysis in digital systems. The proposed register can also be used as a counter and efficient count recovery algorithms can be derived for it. In this brief we have shown that the proposed register is good at exhaustive testing of a digital system. It is also good at detecting burst errors in digital systems.

We have described a BILBO for efficient detection of burst errors. Burst errors are important in testing microprocessors [1]. The BILBO described can also be used for exhaustive testing. This is accomplished by using a nonlinear feedback shift register for test pattern generation.

References

[1] J. E. Smith, "Measures of the effectiveness of fault signature analysis," *IEEE Trans. Comput.*, vol. C-29, pp. 510–514, June 1980.
[2] J. P. Hayes, "Transition count testing of combinational logic circuits," *IEEE Trans. Comput.*, vol. C-25, pp. 613–620, June 1976.
[3] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed. Cambridge, MA: MIT Press, 1972.
[4] H. C. A. van Tilborg, *An Introduction to Cryptology*, Norwell, MA: Kluwer Acadamic, 1988.
[5] L. Wang and E. J. McCluskey, "Circuits for pseudoexhaustive test pattern generation," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 1068–1080, Oct. 1988.
[6] D. W. Clark and L. Weng, "Maximal and near-maximal shift register sequences: Efficient event counters and easy discrete logarithms," *IEEE Trans. Comput.*, vol. 43, pp. 560–567, May 1994.
[7] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proc. IRE*, pp. 228–235, Jan. 1961.
[8] L. Wang and E. J. McCluskey, "Hybrid designs generating maximum-length sequences," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 91–99, Jan. 1988.
[9] C. L. Chen,"Linear dependencies in linear feedback shift registers," *IEEE Trans. Comput.*, vol. C-35, Dec. 1986.