A Lanczos Algorithm in a Finite Field and its Application*

M. Serra T. Slater

Dept. of Comp. Science, Univ. of Victoria
Victoria, B.C., Canada

Abstract

Given a matrix in companion form over GF(2), whose characteristic polynomial is irreducible, a tridiagonal matrix, similar to the original one, is found, by constructing the similarity transformation. The theoretical basis is founded on the Lanczos tridiagonalization method, valid in the Complex domain. A variant of the Lanczos method, based on LU decomposition requirements, is modified to apply in the finite field GF(2). The work is derived from an application in VLSI design, where the matrices in companion form and in tridiagonal form represent two similar linear finite state machines, used for pseudo-random pattern generation and digital circuit testing. The construction of the similarity transformation between the matrices makes it possible to obtain directly the separate implementation of the two corresponding machines.

1 Introduction

In applications for digital circuit testing and VLSI design, some simple linear finite state machines, called LFSRs for linear feedback shift registers, are commonly used either as pseudo-random pattern

*This work was supported by research grants from the Natural Sciences and Engineering Research Council of Canada.

generators or signature analyzers [1]. Normally their behaviour is well known and the algebraic theory behind is not considered explicitly. Recently some interest has surfaced towards a different type of linear finite state machine, called Cellular Automata (CA) [4, 10]. The study of these machines proceeded at first through simulation, and only recently some research has shown that their behaviour can be extrapolated directly from the analysis of the corresponding matrix representing their transition functions [11].

The LFSRs can be implemented directly by mapping the nonzero coefficients of the characteristic polynomial of their transition matrix to simple feedback paths of wires. The important detail, usually, is to select an appropriate primitive polynomial, such that the machine, when operating, would cycle through a maximal number of distinct states. The same property was sought for the CA[1], but the maximal length behaviour was analyzed only by simulation. In [11] it was shown that there exists a similarity relation between an LFSR and a linear cellular automata, if they correspond to the same irreducible (primitive) characteristic polynomial.

There still remained the problem of finding the tridiagonal matrix representing a cellular automaton, similar to a specified LFSR, whose behaviour was known. The standard tridiagonalization algorithms, however, depend heavily on properties of the real or complex field, not necessarily shared by finite fields. In this paper, we present a version of the Lanczos tridiagonalization method, with a variant based on LU decomposition requirements, modified to apply to GF(2). This contribution bridges a gap often found between the extensive research available in algebraic methods and the one for finite fields [7], where algorithms described in the former area are useful in the latter, but may not necessarily be transferred directly.

In section 2, some background terminology and useful theorems are stated. In section 3, a very short description of the application is given. Since the original problem is beyond the scope of this paper, a more detailed summary of the application is presented only in the Appendix, in order to place the modified algorithm in more precise context. In section 4, the derivation of the modified

[1]CA is used as an abbreviation for both Cellular Automaton and Cellular Automata, depending on the context.

Lanczos tridiagonalization method is shown, together with the final algorithm and a complete example. In section 5, implementation results are presented.

2 Background

The application for tridiagonal matrices comes from their representation of certain linear finite state machines. The algebraic manipulation of the matrices makes possible a better study of the behaviour of the machines themselves. Some definitions and theorems are necessary for the exposition below.

Definition 1 *[14] A machine M is a* Linear Finite State Machine *if (1) the state space S_M of M, the input space I_M, and the output space Y_M are each vector spaces over the appropriate finite field (here GF(2)); (2) let the vector s_i denote the state of the machine, the vector u_i denote the inputs to the machine, and the vector y_i denote the outputs of the machine. The next state s_i^+ of M is defined by $s_i^+ = Rs_i + Pu_i$ and the output is defined by $y_i = Ts_i + Qu_i$ where R, P, T, and Q are transformation matrices of the appropriate size over the finite field. In the case of an autonomous machine (with no external input u_i), the second term is omitted from each equation.*

Definition 2 *The* characteristic polynomial *of a square matrix A is defined as $\Delta_A(\lambda) = det(\lambda I - A)$.*

Definition 3 *[9, p.148] A polynomial $p(X)$ of degree n which is not divisible by any polynomial of degree k, where $0 < k < n$, is called* irreducible.

Theorem 1 *[13, p.269] Similar matrices have the same characteristic polynomial.*

Theorem 2 *If two matrices A and B have the same characteristic polynomial $\Delta(\lambda)$ which is irreducible (or primitive), then A is similar to B.*

Definition 4 *[9, p.158,161] An element α of a finite field is called a* primitive element *if it generates the multiplicative group of the field. The minimum polynomial of a primitive element is called a* primitive polynomial, *and all its roots are primitive elements, where minimum refers to the monic polynomial of smallest degree such that $m(\alpha) = 0$.*

An irreducible polynomial of degree n can be tested to be primitive if and only if it divides $X^m - 1$ for no m less than $q^n - 1$, where q is the characteristic of the field [9, p.161].

Definition 5 *A* companion *matrix is defined to be of the form:*

$$\begin{pmatrix} 0 & 1 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & & 0 & 1 & 0 \\ 0 & 0 & 0 & & 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{k-3} & -a_{k-2} & -a_{k-1} \end{pmatrix}$$

where the a_i are the coefficients of the monic characteristic polynomial. Since the underlying field is GF(2), where addition and subtraction are equivalent, we can omit the negative signs. Some authors also may define a companion matrix to be the transpose of the one shown above.

Definition 6 *A tridiagonal* 90/150 matrix *has the following structure:*

$$\begin{pmatrix} a_{11} & 1 & 0 & 0 & & 0 & 0 & 0 \\ 1 & a_{22} & 1 & 0 & & 0 & 0 & 0 \\ 0 & 1 & a_{33} & 1 & & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & & a_{n-2,n-2} & 1 & 0 \\ 0 & 0 & 0 & 0 & & 1 & a_{n-1,n-1} & 1 \\ 0 & 0 & 0 & 0 & & 0 & 1 & a_{n,n} \end{pmatrix}.$$

All elements of the subdiagonal and the super diagonal are 1, while the elements of the diagonal can be 1 or 0, depending on the rule of the linear finite state machine that they are representing [11]. It can be shown that, in fact, the only acceptable structure for the required CA is such a tridiagonal matrix. The name comes from the names, *90* and *150*, of the computation rules of the corresponding machine [15]. Moreover, such a matrix is always non-derogatory [2] and the structure is necessary (but not sufficient) for the characteristic polynomial to be irreducible [11].

3 The Application and the Problem

This research started from a direct application of tridiagonal matrices over GF(2). The complete presentation can be found in [11], but a summary is included in the Appendix. In the application, a tridiagonal matrix represents the linear transition function from current state to next state of a particular finite state machine called a cellular automata. However, it is desirable for the tridiagonal matrix to have a primitive characteristic polynomial, as in that case the corresponding CA has a maximal length cycle structure and thus produces a maximal length sequence of patterns [11]. Primitive polynomials have been tabulated and they can be mapped directly into the implementation of another linear finite state machine, the LFSR.

In fact the transition matrix for an LFSR is a matrix in companion form, where the coefficients of the characteristic polynomial occupy the last row. The requirement from the previous research was to have a choice of implementations between a CA and an LFSR, since the CA presents better randomness in the sequence of patterns generated. While the LFSR matrix was known immediately, the corresponding CA tridiagonal matrix had to be found algorithmically.

The general underlying algebraic problem is to compute a tridiagonal matrix similar to a companion matrix, whose characteristic

[2] A matrix is non-derogatory if its minimum polynomial is equal to its characteristic polynomial, where the minimum polynomial is the monic polynomial of lowest degree which annihilates the matrix.

polynomial is irreducible or primitive. In this paper a constructive approach is taken, involving a pruned search. The basic Lanczos algorithm has been changed and adapted to work in the finite field GF(2).

4 The Revised Lanczos Methods

There exists an algorithm used for tridiagonalizing complex matrices known as the *Lanczos tridiagonalization algorithm* [13]. Although this algorithm, as it is usually stated, cannot be used in finite fields, the theory behind it is useful. From it we can aim to build a constructive algorithm for GF(2). The following discussion is based on [13, p.338–339].

Definition 7 *[3] For a given* $(2n-1)$*-vector* $[y_1, y_2, \cdots, y_{2n-2}, y_{2n-1}]$*, a* Hankel *matrix has the form*

$$\begin{pmatrix} y_1 & y_2 & y_3 & \cdots & y_n \\ y_2 & y_3 & y_4 & \cdots & y_{n+1} \\ y_3 & y_4 & y_5 & \cdots & y_{n+2} \\ \vdots & \vdots & \vdots & & \vdots \\ y_n & y_{n+1} & y_{n+2} & \cdots & y_{2n-1} \end{pmatrix}$$

In particular, the matrix is symmetric and is uniquely specified by y*, which we call the* defining vector.

Definition 8 *For a given* n*-vector* x *and* $n \times n$ *matrix* M*, define the* Krylov *matrix* $K(M,x)$ *with respect to* M *and* x *to be* $K(M,x) = [x; Mx; M^2x; \ldots; M^{n-1}x]$ *in column vector notation.*

If $X = K(M,x)$ is nonsingular, then $X^{-1}MX$ is the transpose of the companion matrix for the characteristic polynomial of M.

Definition 9 *A factorization of a square matrix A into the product LU of a lower and and upper triangular matrix is called an* LU decomposition *of the matrix A. [13, p.131]*

Definition 10 *[13, p.23] A square matrix A is* Upper Hessenberg *if for all $i > j+1$, the elements $\alpha_{ij} = 0$; it is* Lower Hessenberg *if for all $i < j+1$, the elements $\alpha_{ij} = 0$. A is said to be* tridiagonal *if it is both upper and lower Hessenberg.*

More directly, an Upper Hessenberg matrix has all entries below the subdiagonal equal to zero, and conversely for a Lower Hessenberg matrix. Furthermore, if R is any nonsingular upper triangular matrix and $S = XR$, where X is a Krylov matrix as above, then $S^{-1}MS$ is *Upper Hessenberg.*

Assume that for some n–vector y, the matrix $Y = K(M^t, y) = [y; M^t y; (M^t)^2 y; \ldots; (M^t)^{n-1} y]$ is nonsingular and $Y^t X = LU$ where L is lower triangular and U is upper triangular. Denote $R = U^{-1}$ and $T = (L^t)^{-1}$. Then R and T are nonsingular upper triangular matrices and $T^t Y^t = [(L^t)^{-1}]^t LUX^{-1} = UX^{-1} = R^{-1}X^{-1} = (XR)^{-1}$. As noted above, $(XR)^{-1}M(XR) = T^t Y^t MXR$ is upper Hessenberg and its transpose $R^t X^t M^t Y T$ is therefore lower Hessenberg.

If the above computations are repeated with M replaced by M^t, then Y and X switch roles, as do R and T, with the result that $R^t X^t M^t Y T$ is upper Hessenberg. Since a matrix which is simultaneously lower and upper Hessenberg is tridiagonal, we have found a similarity transformation to tridiagonal form.

Suppose the initial matrix M, from above, is already the transpose of a companion matrix, called C^t, with an irreducible characteristic polynomial. Then, if we let $x = [1, 0, \ldots, 0]^t$, it is easy to show that $X = K(C^t, x)$ is the identity matrix, and that $Y = K(C, y)$ is a *Hankel* matrix for any nonzero vector y, as defined above. Then, with $R^t X^t C Y T$ being tridiagonal, substituting appropriately for $X = I$, $Y^t = LU$, $T = (L^t)^{-1}$ and $R = U^{-1}$, we have that $UC^t U^{-1}$ is the required tridiagonal matrix.

The discussion above outlines the theory for a tridiagonalization algorithm. The existence problem resides in the fact that if $\Delta(\lambda)$ is irreducible with companion matrix M, then there exists a vector y such that $Y = K(M, y)$ is nonsingular and has an LU factorization. In the complex field such vectors always can be found, according to [5, p.20-21]. In GF(2), a finite set of candidate vectors must be

constructed according to the LU decomposition requirements, as shown below.

The nonsingularity of Y is guaranteed by the following Theorem.

Theorem 3 *[7, Theorem 6.51, p.214] If M is the companion matrix of an irreducible polynomial of degree n over a finite field, then $K(M^t, y)$ is nonsingular for any nonzero n-vector y.*

The next Theorem is valid over any field and specifies necessary and sufficient conditions for the existence of an LU factorization.

Theorem 4 *[6] An $n \times n$ matrix M has an LU factorization if and only if the leading principal minors of size $\leq n-1$ are nonzero. M is nonsingular and has an LU factorization if and only if all leading principal minors are nonzero.*

The next Theorem concerns the periods of the state cycles for a transition matrix with irreducible characteristic polynomial.

Theorem 5 *[7, Theorem 6.28, p.199] Let M be an $n \times n$ matrix with irreducible characteristic polynomial $\Delta(\lambda)$ such that $\Delta(0) \neq 0$. Then for any nonzero n-vector x, the sequence of vectors $\{x, Mx, M^2x, \ldots\}$ is periodic with least period ord $(\Delta(\lambda))$, where ord $p(x)$ is defined to be the least positive integer e such that $p(x)|(x^e - 1)$.*

Note that for $e = 2^n - 1$, this also defines a primitive polynomial in GF(2), whose only cycle has order $2^n - 1$. Consider the problem of actually finding a vector y such that $Y = K(M, y)$ (for some companion matrix M) is nonsingular and has an LU factorization. For small values of n it is simple to compute an exhaustive list of all possible nonsingular $n \times n$ Hankel matrices over GF(2) that have LU factorizations.

As an example, for $n = 4$ there are precisely 8 such matrices, listed in Table 1, representing each matrix by the defining vector of length 7.

These vectors are obtained in general through a binary search. The idea is to construct a Hankel matrix whose leading principal

Table 1: LU sequences of order 4

1	0	1	0	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	1
1	0	1	1	1	1	0
1	1	0	0	1	0	0
1	1	0	0	1	1	1
1	1	0	1	0	0	0
1	1	0	1	0	1	1

minors are nonzero, in order to fulfill the LU decomposition requirements. For $n = 4$, the matrix has the form:

$$\begin{pmatrix} y_1 & y_2 & y_3 & y_4 \\ y_2 & y_3 & y_4 & y_5 \\ y_3 & y_4 & y_5 & y_6 \\ y_4 & y_5 & y_6 & y_7 \end{pmatrix}$$

For Table 1, an example of the search follows. Note that y_1 is always 1. Then for y_2 we have the constraint that $y_3 + y_2^2 = y_3 + y_2 = 1$, that is $y_3 \neq y_2$, for the leading minor of order 2. There are thus 2 possibilities, to satisfy the LU decomposition:

$$(a) \quad \begin{matrix} 1 & 0 & 1 & y_4 \\ 0 & 1 & y_4 & y_5 \\ 1 & y_4 & y_5 & y_6 \\ y_4 & y_5 & y_6 & y_7 \end{matrix} \qquad or \qquad (b) \quad \begin{matrix} 1 & 1 & 0 & y_4 \\ 1 & 0 & y_4 & y_5 \\ 0 & y_4 & y_5 & y_6 \\ y_4 & y_5 & y_6 & y_7 \end{matrix}$$

For case (a), the condition for the leading minor of size 3 reduces to $y_5 = y_4$, leading to 2 more possibilities:

$$(aa) \quad \begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & y_6 \\ 0 & 0 & y_6 & y_7 \end{matrix} \qquad or \qquad (ab) \quad \begin{matrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & y_6 \\ 1 & 1 & y_6 & y_7 \end{matrix}$$

Similarly, for case (b), the condition is $y_4 \neq y_5$, giving:

$$(ba)\quad \begin{matrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & y_6 \\ 0 & 1 & y_6 & y_7 \end{matrix} \qquad or \qquad (bb)\quad \begin{matrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & y_6 \\ 1 & 0 & y_6 & y_7 \end{matrix}$$

So far the 4 partial vectors constructed are:

$$\begin{matrix} 1 & 0 & 1 & 0 & 0 & y_6 & y_7 \\ 1 & 0 & 1 & 1 & 1 & y_6 & y_7 \\ 1 & 1 & 0 & 0 & 1 & y_6 & y_7 \\ 1 & 1 & 0 & 1 & 0 & y_6 & y_7 \end{matrix}$$

The process continues in a similar manner to fix values for y_6 and y_7.

We refer to a vector of length $2n-1$ as an *LU sequence* of order n if the Hankel matrix it represents is nonsingular and has an LU factorization. Also we refer to the vector consisting of the first n components of an LU sequence as an *LU state vector.* We see from Table 1 that for $n = 4$, only 4 of the 15 possible nonzero state vectors are LU state vectors. In this case they are (1010),(1011),(1100),(1101). The following Theorems give precise values for the number of LU sequences and LU state vectors of a given order.

Theorem 6 *The number of LU sequences of order n is 2^{n-1}.*

Proof: The proof is by induction on n. For $n = 1$, $2^{1-1} = 1$ and there is only one LU sequence of order 1, namely (1). Assume that for some $n > 1$ there are 2^{n-2} LU sequences of order $n-1$. Let $x = (x_1, x_2, \ldots, x_{2n-3})$ be one of these sequences and let

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_{n-1} \\ x_2 & x_3 & \cdots & x_n \\ \vdots & \vdots & & \vdots \\ x_{n-1} & x_n & \cdots & x_{2n-3} \end{pmatrix}$$

be the Hankel matrix generated by x. By the definition of an LU sequence, the row vectors of X are linearly independent. Select an

element x_{2n-2} arbitrarily from GF(2), let $x' = (x_1, x_2, \ldots, x_{2n-3}, x_{2n-2}, \square)$ and let

$$X' = \begin{pmatrix} x_1 & x_2 & \cdots & x_{n-1} & x_n \\ x_2 & x_3 & \cdots & x_n & x_{n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n-1} & x_n & \cdots & x_{2n-3} & x_{2n-2} \\ x_n & x_{n+1} & \cdots & x_{2n-2} & \square \end{pmatrix}$$

be the Hankel matrix generated by x', where $\square$ indicates a missing component to be filled in when x_{2n-1} is specified.

From the independence of X, there is a unique row vector $a = [\alpha_1, \alpha_2, \ldots, \alpha_{n-1}]$ such that $aX = [x_n, x_{n+1}, \ldots, x_{2n-2}]$, the first $n-1$ components in the last row of X'. Let $x_{2n-1} = 1 + \alpha_1 x_n + \alpha_2 x_{n+1} + \cdots + \alpha_{n-1} x_{2n-2}$, then x_{2n-1} is the *unique* element of GF(2) which makes X' linearly independent when substituted for $\square$. Consequently, each LU sequence x of order $n-1$ is the parent of two LU sequences of order n, which implies that there are 2^{n-1} LU sequences of order n. ∎

Corollary 6.1 *The number of LU state vectors of length n is* $2^{\lfloor \frac{n}{2} \rfloor}$.

Proof: The statement concerning the number of LU state vectors follows immediately from the Theorem above, since the construction of an LU sequence involves two alternating phases. In the first phase, an arbitrary element of GF(2) is added to a previously constructed sequence, and in the second phase, a uniquely determined element is added. ∎

4.1 The Constructive Algorithm

The revised Lanczos method constructs a similarity transformation from the companion matrix C of a polynomial $p(x)$ to a tridiagonal matrix provided there is some vector y such that $K(C, y)$ is nonsingular and has an LU factorization. By Theorem 6 there are precisely $2^{\lfloor \frac{n}{2} \rfloor}$ LU state vectors and these are the only possible choices for y. The new algorithm operates by executing a loop in which an LU state vector y is generated, the matrix $Y = K(C, y)$ is

computed and it is factorized into a $Y = LU$. If successful, UC^tU^{-1} is a tridiagonal matrix similar to C (and also to C^t).

The following notation is used in the statement of the algorithm.

(i) The set of LU state vectors of order n is denoted by $LUS(n)$.

(ii) The Krylov matrix with respect to a matrix M and vector y is denoted by $K(M, y)$.

Algorithm :

Purpose: To compute a tridiagonal matrix over GF(2) which is similar to the companion matrix of a given polynomial.

Input *n:* the degree of the polynomial.
C: the companion matrix of the polynomial.

Output A tridiagonal matrix over GF(2) which is similar to C.

```
(1)     Compute the set LUS(n);
(2)     found := FALSE; empty := FALSE;
(3)     while ( found = FALSE and empty = FALSE ) do
          begin
(4)         if ( LUS(n) = ∅ ) then
(5)           empty := TRUE;
            else
              begin
(6)             select y ∈ LUS(n); LUS(n) = LUS(n) − {y};
(7)             Y := K(C, y);
(8)             if ( Y is nonsingular and has an LU
                    factorization) then
(9)               found := TRUE endif
              end
            endif
          end while
(10)    if ( found = TRUE ) then
          begin
```

(11) compute $Y = LU$;
(12) **return** UC^tU^{-1};
end
else
(13) **output** *"no LU sequence exists"* **endif**

4.1.1 A Complete Example

Example 1 *This is an example of the application of the algorithm to the polynomial* $p(x) = x^6 + x^4 + x^3 + x + 1$ *which is primitive over GF(2).*

We wish to find a 90/150 matrix which has $p(x)$ as its characteristic polynomial, given a companion matrix, C, for $p(x)$. By Theorem 6 there are precisely $2^{\lfloor \frac{6}{2} \rfloor}$ LU state vectors which we list below

y_1	1	0	1	0	0	0
y_2	1	0	1	0	0	1
y_3	1	0	1	1	1	0
y_4	1	0	1	1	1	1
y_5	1	1	0	0	1	0
y_6	1	1	0	0	1	1
y_7	1	1	0	1	0	0
y_8	1	1	0	1	0	1

For LU state vector $y_1 = [101000]$, we have

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \mu_7 \\ 1 & 0 & 0 & 0 & \mu_7 & \mu_8 \\ 0 & 0 & 0 & \mu_7 & \mu_8 & \mu_9 \\ 0 & 0 & \mu_7 & \mu_8 & \mu_9 & \mu_{10} \\ 0 & \mu_7 & \mu_8 & \mu_9 & \mu_{10} & \mu_{11} \end{pmatrix}$$

and we compute

$$Y = K(C, y_1) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

which does not have an LU factorization. Similarly, LU state vector $y_2 = [101001]$ does not yield an LU factorization. However for $y_3 = [101110]$ we have

$$\begin{aligned} K(C, y_3) &= \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\ &= LU \end{aligned}$$

We then compute

$$UC^tU^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

which is a 90/150 tridiagonal matrix as required. ∎

5 Implementation and Results.

The constructive Lanczos-based algorithm was first implemented in C using Sun workstations and used to construct tables. All irreducible and primitive polynomials up to degree 32 from the tables in [9] have been processed and for each of them the diagonal vector for the state transition matrix of the CA has been found. Recently, the algorithm has been rewritten for an IBM 3090 with a vectorizing compiler, and work is currently under way to produce the CA corresponding to some of the primitive polynomials up to degree 300 listed in [1].

For all above polynomials, we have a unique tridiagonal matrix (up to reversal of the diagonal), similar to the companion matrix. The complete set of tables is available in [12].

References

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

[2] B. Elspas. The theory of autonomous linear sequential networks. *IRE Trans. on Circuit Theory*, CT-6(1):45–60, 1959.

[3] R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[4] P.D. Hortensius, R.D. McLeod, W. Pries, D.M. Miller, and H.C. Card. Cellular automata-based pseudorandom number generators for built-in self-test. *IEEE Trans. on Computer Aided Design of Circuits and Systems*, 8(8):842–859, August 1989.

[5] A.S. Householder. *The Theory of Matrices in Numerical Analysis.* Blaisdell Publishing, 1964.

[6] C.R. Johnson, D.D. Olesky, and P. van den Driessche. Inherited matrix entries: LU factorizations. *SIAM J. on Matrix Analysis and Applications*, 10(1):94–104, 1989.

[7] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and their Applications.* Cambridge University Press, 1986.

[8] E.J. McCluskey. *Logic Design Principles with emphasis on Testable Semicustom Circuits.* Prentice-Hall, 1986.

[9] W. Wesley Peterson and E.J. Weldon. *Error-Correcting Codes.* MIT Press, Cambridge, Ma, 1972.

[10] W. Pries, A. Thanailakis, and H.C. Card. Group properties of cellular automata and VLSI applications. *IEEE Trans. Computers*, C-35(12):1013–1024, Dec. 1986.

[11] M. Serra, T. Slater, J.C. Muzio, and D.M. Miller. The analysis of one dimensional linear cellular automata and their aliasing properties. *IEEE Trans. on Computer Aided Design of Circuits and Systems*, to appear, 1990.

[12] T. Slater and M. Serra. Tables of linear hybrid 90/150 cellular automata. Technical Report DCS-105-IR, Univ. of Victoria, Dept. of Comp. Science, Univ. of Victoria, Victoria, B.C., Jan 1989.

[13] G.W. Stewart. *Introduction to Matrix Computations.* Academic Press, 1973.

[14] H.S. Stone. *Discrete Mathematical Structures and Their Applications.* Science Research Associates Inc., 1973.

[15] S. Wolfram. Statistical mechanics of cellular automata. *Rev. of Modern Physics*, 55:601–644, 1983.

A Appendix: The Finite State Machines Application

This appendix contains a summary of the application of tridiagonal 90/150 matrices and of the Lanczos algorithm for GF(2). It is not necessarily intended for publication here, but it is provided for information and clarification.

A.1 Linear Finite State Machines: CA and LFSR

A basic result in Linear Algebra is that a linear transformation in a vector space is represented by a matrix and that, in turn, such a transition matrix represents a linear transformation. A function $f : V \rightarrow V'$ is a *linear function* from a vector space V into a vector space V' over the same scalar field K as V if, for all c_1 and c_2 in K and all v_1 and v_2 in V,

$$f(c_1 v_1 + c_2 v_2) = c_1 f(v_1) + c_2 f(v_2).$$

The linear transformations describe the behaviour of a corresponding linear finite state machine, whose definition follows.

Definition 11 *[14] A machine M is a* Linear Finite State Machine *if*

1. *the state space S_M of M, the input space I_M, and the output space Y_M are each vector spaces over the appropriate finite field (here GF(2));*

2. *let the vector s_i denote the state of the machine, the vector u_i denote the inputs to the machine, and the vector y_i denote the outputs of the machine. The next state s_i^+ of M is defined by*

$$s_i^+ = Rs_i + Pu_i$$

and the output is defined by

$$y_i = Ts_i + Qu_i$$

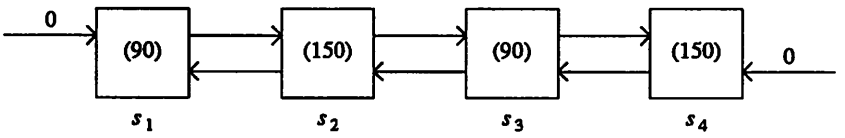

Figure 1: Example of Cellular Automaton

where R, P, T, and Q are transformation matrices of the appropriate size over the finite field. In the case of an autonomous machine (with no external input u_i), the second term is omitted from each equation.

The particular instance of machines under investigation is called a One Dimensional Linear Cellular Automata, defined as a uniform array of identical cells in an n-dimensional space. They are characterized by the cellular geometry and the rule to compute the successor state. Cells are restricted to local neighbourhood interaction and have no global communication. Cells use an algorithm to compute their successor state, called its *computation rule* [15], based on the information received from its nearest neighbors.

Figure 1 is an example of the type of machine being described. Each site, labelled s_i, $1 \leq i \leq k$, can contain the values 0 or 1, and receives an input from each neighbour, s_{i-1} and s_{i+1}, at every clock cycle. The state of the CA is given by the bit pattern $s_1, \ldots, s_k$, where k is the length of the CA. It is said to have *null boundary* conditions, as shown, when the first and last site always receive a 0.

The next state of the CA is determined by the current state and the rules which govern its behaviour. In Figure 1, no external input is entered in the CA, except for the boundaries where the input is constant. Only linear one-dimensional CA are of interest to us. The linear rules used in the example are Rules 90 and 150, as classified in [15]. They are defined as follows:

$$
\begin{aligned}
Rule90 : s_i^+ &= s_{i-1} \oplus s_{i+1} \\
Rule150 : s_i^+ &= s_{i-1} \oplus s_i \oplus s_{i+1}
\end{aligned}
$$

where s_i^+ denotes the next state for site s_i and $\oplus$ denotes addition Mod 2 over GF(2). Hence Rule 90 updates the machine only with inputs from the two neighbours, while Rule 150 also includes the present state of the site. In Figure 1, an *hybrid* CA is shown, as sites do not all use the same rule.

Since the rules of operation are linear, a state transition matrix can be written for a given such CA, derived from the next state equations defined by the rules used. For the CA shown in Figure 1 and the linear rules described above, the state transition matrix is given by

$$
\mathrm{B} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}
$$

Notice that for any one dimensional linear CA using nearest neighbour rules, this matrix is *tridiagonal.* It can be shown that, in fact, the only acceptable structure is such a tridiagonal matrix with unit super diagonal and sub diagonal, and we denote it as a *90/150 matrix.* Moreover, such a matrix is always non-derogatory and the structure is necessary for the characteristic polynomial to be irreducible (but not sufficient) [11]. The matrix B has a characteristic polynomial, and, if it is non-derogatory, similarity transformations can produce its companion matrix C which has the same characteristic polynomial.

The properties of Cellular Automata (CA) have been studied by Wolfram [15] and later by Pries et al [10]. Some particular classes of linear CA have been found to have very good randomness properties, using standard randomness tests. In fact, they are superior to the more common linear feedback shift registers (LFSRs) which have been used for pseudo-random pattern generation. LFSRs have also been used as random pattern generators for input stimuli to a

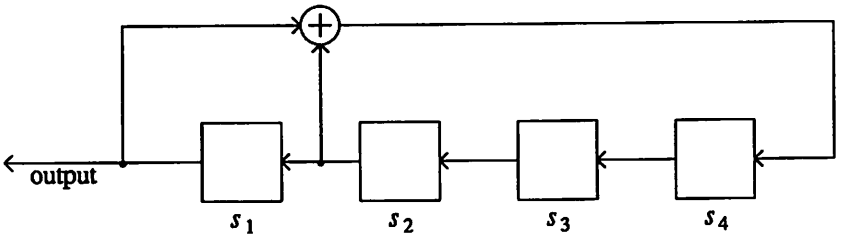

Figure 2: Type 2 LFSR

binary circuit under test and as testers for signature analysis [8], as their random sequencing and inexpensive implementation makes them suitable for hardware testing.

LFSRs are the original linear finite state machines used in almost all testing applications and pseudo-random pattern generation over GF(2). It is worth giving a brief overview. Figure 2 shows the structure of an LFSR, given in Type 2 form. The LFSR is made up of simple memory elements, and Exclusive-Or gates which perform addition Mod 2, chained together and controlled by a synchronous clock. The linear finite-state machine, which defines the LFSR, performs polynomial division over GF(2), where the LFSR implements the divisor, the serial stream of inputs represents the dividend, the serial output stream gives the quotient, while the last state of the LFSR describes the remainder polynomial (see [9]).

A set of linear equations can be used to describe the transition to the next state (see below). For example, the equations for the LFSR of Figure 2 are:

$$
\begin{aligned}
s_1^+ &= s_2 \\
s_2^+ &= s_3 \\
s_3^+ &= s_4 \\
s_4^+ &= s_1 \oplus s_2
\end{aligned}
$$

The corresponding matrix for the transformation is:

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

whose characteristic polynomial is $\lambda^4 + \lambda + 1$, which is also represented in the LFSR construction.

If there is no input stream, then the LFSR behaves as an autonomous linear finite state machine, and the sequences of states produced can be used as a pseudo-random pattern generator, while the output sequence can be used to produce group codes. The general theory of LFSRs dates back to coding theory and can be found in [2, 9], or in more general books on discrete mathematics [14].

A.2 The Similarity

The research in [11] was aimed at proving the isomorphism in the behaviour of a CA and an LFSR, if the two machines are based on the same irreducible(or primitive) polynomial as characteristic polynomial of their transition matrix. The isomorphism in behaviour implies that the machines have the same cycle structure, which is directly determined by the minimum polynomial of the transition matrix [14]. If the characteristic polynomial is irreducible, then it is equal to the minimum polynomial.

In fact, the machines shown in the examples above have the same behaviour in their cycle structure, as the two matrices B and C have the same irreducible characteristic polynomial and are similar. Since a lot was known about LFSRs the interest turned to finding a linear CA isomorphic to a given LFSR. That is, the problem was of finding a tridiagonal matrix similar to a given companion matrix.

The first prerequisite, as shown in [11] is that the matrix has an irreducible characteristic polynomial. If the characteristic polynomial is also primitive, then the cycle structure of the machine has only two cycles, one with the all zero state and one which chains together the remaining $2^n - 1$ states, where n is the number of cells

in the machine. A list of primitive polynomials over GF(2), from which one can easily construct a maximal length LFSR, is available, and the algorithm presented constructs the corresponding one dimensional linear CA. While the CA has the same cycle structure as the LFSR, the sequencing of states shows better randomness when used as a pseudo-random pattern generator.