1967

The realization and application of parallel linear feedback shift registers

Ross K. Heitzmann

Department:

THE REALIZATION AND APPLICATION OF PARALLEL LINEAR FEEDBACK SHIFT REGISTERS

BY

ROSS K. HEITZMANN - 1936 -

A

THESIS

submitted to the faculty of the

UNIVERSITY OF MISSOURI AT ROLLA

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Rolla, Missouri

1967

Approved by

James H. Tracey (Advisor)

Frank J. Hesse

R E D Jones

B C Gillett

ABSTRACT

Two methods for serial-to-parallel transformation of linear feedback shift registers are briefly discussed. A third method for transformation is rigorously developed using a next-state and output equation representation of the linear feedback shift register. An algorithm is developed for simplifying the parallel machine resulting from serial-to-parallel transformation, where simplification is defined as reduction in the required number of modulo 2 adders. A computer program incorporating serial-to-parallel transformation and the simplification algorithm is provided.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to his major professor, Dr. James H. Tracey, for guidance and for prompt and careful reading of this paper.

The author also appreciates the typing efforts of his wife, Betty.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

The use of cyclic codes for error detection and correction is well known[1,2]. Cyclic codes are very useful because they are readily constructed using algebraic techniques and because coding and decoding are easily accomplished using linear feedback shift registers. In the past, encoding and decoding has been done serially by shifting messages through registers one bit at a time. This paper presents a method for decreasing the necessary message processing time by using parallel encoders and decoders. The remainder of this introductory section presents the basic ideas of cyclic codes and linear feedback shift registers and relates their characteristics to the serial-to-parallel transformation problem.

A general representation for a binary message of length k is

$$M = a_0\ a_1 \cdots a_{k-1} \text{ where } a_i = 0 \text{ or } 1.$$

The binary digits in the message can be thought of as coefficients of a polynomial m(x) as follows:

$$m(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}.$$

The binary operations + and "." are defined as follows:

+	0	1
0	0	1
1	1	0

.	0	1
0	0	0
1	0	1

It should be noted that "+" is the "modulo 2 addition" operation and "." is ordinary multiplication. These

polynomials are distributive, associative, and commutative under the two operations, and also the polynomials factor uniquely into irreducible factors.

A second polynomial, g(x), is the generator polynomial for a cyclic code. The generator is of degree n-k where the information message is of length k and the coded message of length n. The generator polynomial g(x) is used to generate the code polynomial f(x) from the message polynomial m(x) as follows:

1) Multiply m(x) by x^{n-k}

2) Divide $x^{n-k}m(x)$ by g(x) to yield a quotient q(x) and remainder r(x). By the Euclidean Division Algorithm it must be true that

$$x^{n-k}m(x) = g(x)q(x)+r(x).$$

Adding r(x) to both sides (modulo 2) yields

$$x^{n-k}m(x)+r(x) = g(x)q(x)$$

where the code polynomial

$$f(x) = x^{n-k}m(x)+r(x) = g(x)q(x)$$

is of degree n-1.

The code is systematic, that is, the encoding process does not change the information bits. Thus, the information bits are retained as the coefficients of $x^{n-1}, x^{n-2} \ldots x^{n-k}$ and the check digits are the coefficients of $x^{n-k-1} \ldots x^{0}$. Decoding is therefore a relatively simple task. The first k digits are the

information digits, provided that no errors have occurred. Error detecting or correcting is accomplished by dividing the coded polynomial f(x) by the generator polynomial g(x). If

$$f(x) = g(x)q(x)$$

then

$$f(x)/g(x) = q(x)$$

provided that no errors have occurred. If a detectable number of errors has occurred, then division by g(x) will yield a nonzero remainder.

The error detection or correction capability of any cyclic code is determined by the generator polynomial for the code. For example, Peterson and Brown[1] prove that the cyclic code generated by the polynomial

$$g(x) = 1+x$$

is a single-error-detecting code. In addition, of course, the code detects any odd number of errors. Peterson and Brown also develop generator polynomials for double-and triple-error-detecting codes and for burst-error-detecting codes. However, the best known cyclic codes are those for which the generator polynomial is constructed using techniques described by Bose and Chaudhuri. For these codes, given that the desired message length and error-detection capability have been selected, construction amounts to construction of the generator polynomial using the Bose-Chaudhuri algorithm as described by Peterson[2]. Encoding and decoding for the Bose-Chaudhuri codes, as for all cyclic

codes, amounts to division by the generator polynomial. If error correction is to be accomplished the remainder must be evaluated at the decoder.

The capability of a cyclic code to detect errors, is shown by the following example which is an information channel for transmission of four binary digits ($k = 4$). The generator polynomial

$$g(x) = x^3+x+1$$

will provide three check digits for the code. It can be shown that this code has minimum (Hamming) distance three so that it can detect any combination of two errors. If the message 1011 is to be encoded, then

$$m(x) = 1+x^2+x^3$$

$$x^{n-k}m(x) = x^3mx = x^3+x^5+x^6$$

$$x^{n-k}m(x) \div g(x) = (x^3+x^5+x^6)\div(1+x+x^3)$$

$$= q(x)+r(x) \text{ (modulo 2)}$$

$$= (1+x+x^2+x^3)+1.$$

Therefore, the coded message polynomial is

$$g(x)q(x) = x^{n-k}m(x)+r(x) = 1+x^3+x^5+x^6.$$

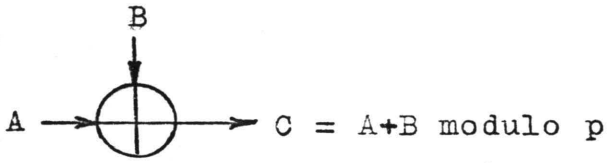
Thus, the binary coded message to be transmitted is 1001011.

From this example it is obvious that encoding essentially consists of division of the message polynomial by the generator polynomial. At the decoder the coded message polynomial is again divided by the generator polynomial. Therefore it is desirable to find a machine which will efficiently accomplish this division. It is

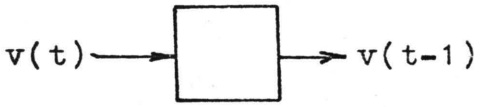
well known that the linear feedback shift register (LFSR) is such a machine.

The LFSR is constructed using the three building blocks described below:

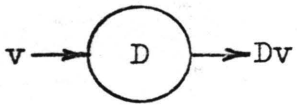
1) The modulo-p adder.



2) The delay element. The output is the input delayed by one clock time. For p = 2 this element might be a flip-flop.



3) The constant multiplier element where D is an element from the residue class ring modulo p. For p = 2 this element is an open circuit for D = 0 and a wire for D = 1.



Given a generator polynomial g(x) with coefficients from the GF [2] (i.e. the prime or Galois field having 2 elements) such that

$$g(x) = a_r x^r + a_{r-1} x^{r-1} + a_{r-2} x^{r-2} + \cdots a_1 x + a_0,$$

the LFSR for dividing by g(x) is as shown in Figure 1. The register is initially set to 0. The (n-1) degree polynomial p(x) which is to be divided by g(x) is shifted, high order first, into the register from the left. After

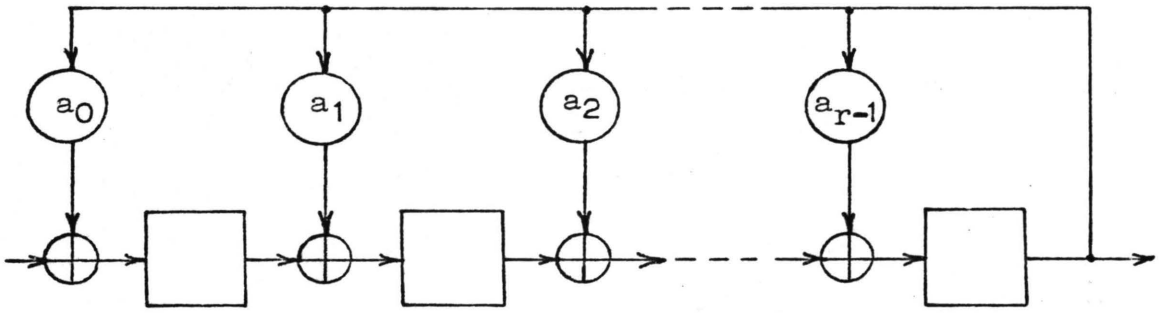


FIGURE 1

LFSR for Division by g(x)

r-1 shifts, the first coefficient of the quotient appears at the output. After n-1 shifts, the entire quotient has appeared at the output and the shift register retains the remainder.

Example:

The (7,4) code, that is; n = 7, and k = 4, might have a generator polynomial

$$g(x) = x^3+x+1.$$

The feedback shift register for accomplishing division by g(x) is as shown in Figure 2. If the 4 bit message to be

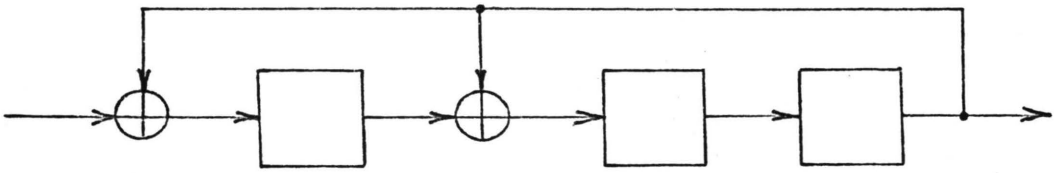


FIGURE 2

LFSR for Division by x^3+x+1

encoded is shifted into this register high order coefficient first, then after three shifts the coefficient will begin

to appear at the output and after seven shifts the remainder will be in the register. Taking a specific example, if m = 1011, then

$$m(x) = 1+x^2+x^3$$

and

$$x^2m(x) = x^3+x^5+x^6.$$

This polynomial, divided by

$$g(x) = 1+x+x^3$$

yields a quotient

$$q(x) = 1+x+x^2+x^3$$

and a remainder r(x) = 1. The flow table below illustrates the operation of the register:

t	I	x_0	x_1	x_2
0	1	0	0	0
1	1	1	0	0
2	0	1	1	0
3	1	0	1	1
4	0	0	1	1
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0

The message to be sent is

$$1+x^3+x^5+x^6 \longrightarrow 1\ 0\ 0\ 1\ 0\ 1\ 1.$$

At the receiver an identical shift register will divide this coded message polynomial by g(x). After 7

shifts the remainder (0 if no errors have occurred) will be in the shift register and an "OR" gate on the outputs of the delay elements must read 0 after the seven shifts if no error has occurred. Figure 3 shows the decoder.

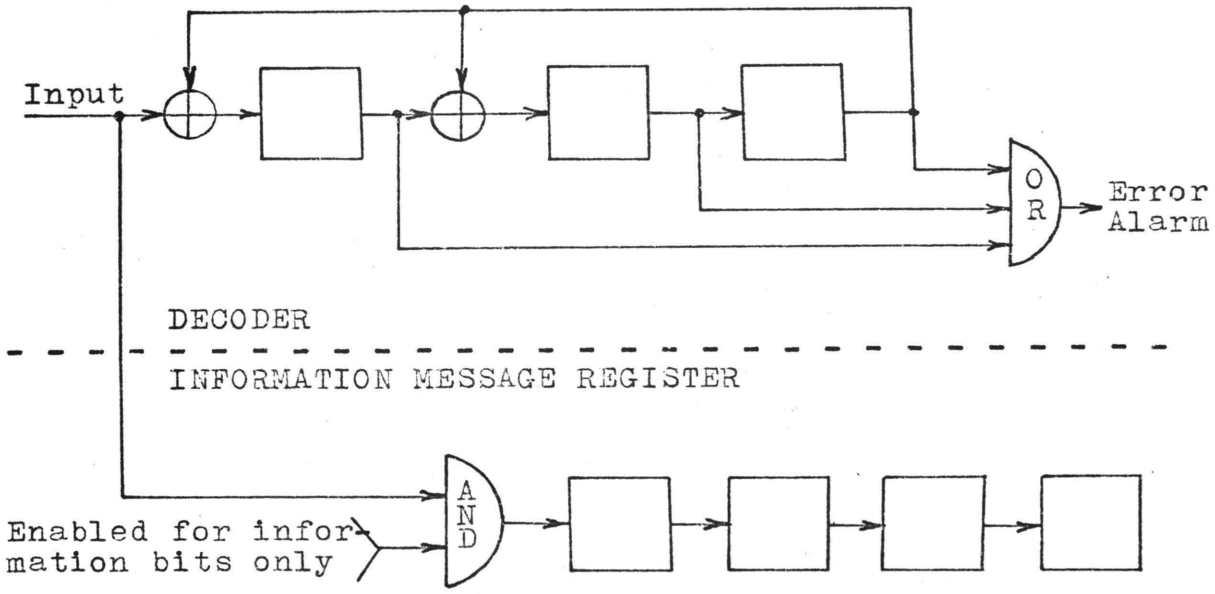


FIGURE 3

Decoder for the (7,4) Code

Cyclic codes, as has already been stated, are readily constructed using well defined algorithms the best known of which is the Bose-Chaudhuri algorithm. Also, the codes can be implemented quite simply using linear feedback shift registers. These shift registers were shown to be serial shift registers which required a total of n clock times to perform error checking (i.e. to calculate the remainder) of an n bit message.

In applications where message processing time is a consideration it is desirable to decrease the encoding and

decoding times. Hence the problem of converting a serial feedback shift register to an equivalent machine which is partially parallel presents itself. The serial machine might be described by the diagram of Figure 4a and the equivalent parallel machine by Figure 4b. Suppose for

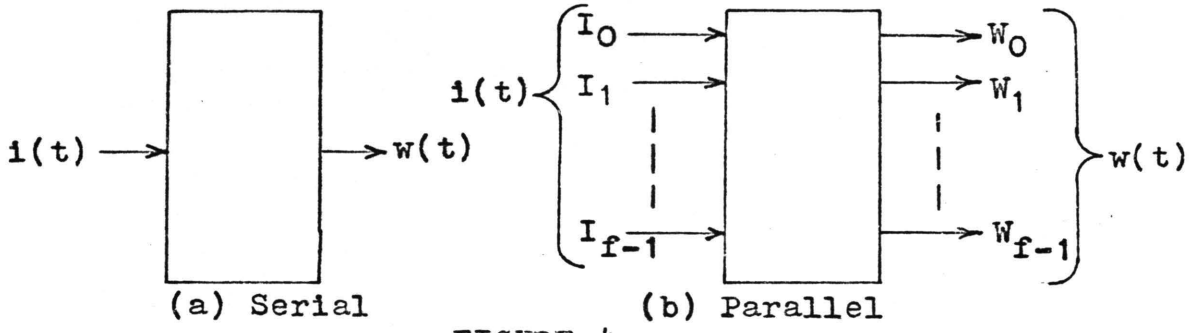


FIGURE 4

Serial and Parallel Equivalent LFSR's

the serial machine that the input sequence is $I_{n-1}\ I_{n-2} \cdots I_2\ I_1\ I_0$ and the output sequence is $W_{n-1}\ W_{n-2} \cdots W_2\ W_1 W_0$. At time $t = 0$ the input is I_0 and the output W_0, at time $t = 1$ the input is I_1 and the output is W_1, and so forth. The input is entered one bit per clock time and the output is provided one bit per clock time. In Figure 4b, the f channel parallel machine enters the input data f bits at a time and provides output data f bits at a time. Thus at time $t = 0$ the input is $I_{f-1}, \ldots I_1, I_0$ and the output is $W_{f-1}, \ldots W_1, W_0$. At time $t = 1$ the input is $I_{2f-1}, \ldots I_{f-1}, I_f$ and the outputs are $W_{2f-1}, \ldots W_f$. The number of clock times required for the parallel machine is q where q is the smallest integer such that

$$q \leq \frac{n}{f}$$

where q is a parameter which can be selected by the designer. The problem to be investigated may now be stated as:

> "Given a serial linear feedback shift register which requires n clock times to accomplish encoding or decoding, find a minimal or near minimal parallel machine which accomplishes the same function in q clock times."

The decrease in message-processing time is, of course, not obtained without a penalty. The penalty is, as is usual for digital machines, an increase in the complexity of the machine. In other words, the decrease in processing time is accompanied by an increase in the amount of hardware necessary to build the encoder and decoder. This will be evident from the example presented in the next section. Thus minimization of the hardware required to realize parallel encoders and decoders is desirable and will be part of the problem treated in this paper.

II. SERIAL-TO-PARALLEL TRANSFORMATION METHODS

This section presents three methods for transformation of serial LFSR's to f-channel parallel analogs. The first two methods are presented essentially as they appear in a paper by Hsiao and Sih [3]. The third method, which seems to be superior to the first two, is based on a method proposed by Hsiao and Sih and extended by Gill [4].

A. Two Elementary Transformation Methods.

Hsiao and Sih call their first method of serial-to-parallel transformation the "Engineering Approach". This method is readily explained by example with the serial LFSR shown in Figure 5. The case in which f = 2 requires

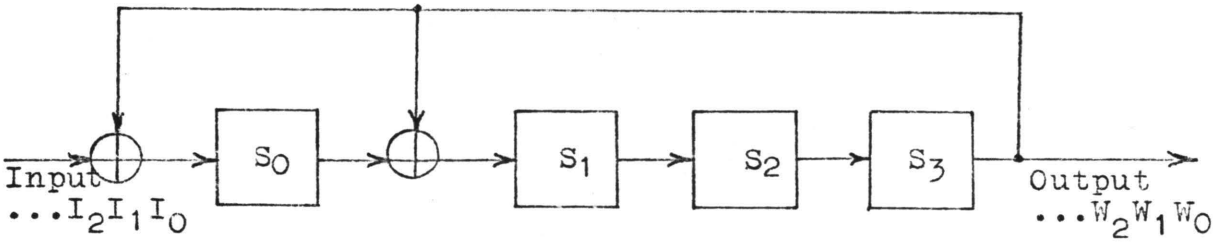


FIGURE 5

LFSR for the Generator $g(x) = 1+x+x^4$

that the input data be entered two bits at a time and the output data be provided two bits at a time. From the serial flow table of Table 1, the initial states of the storage elements are $S_i(0)$.

TABLE 1

Flow Table for a Serial LFSR

t	Input	S_0	S_1	S_2	S_3	Output
0	I_0	$S_0(0)$	$S_1(0)$	$S_2(0)$	$S_3(0)$	W_0
1	I_1	$I_0+S_3(0)$	$S_0(0)+S_3(0)$	$S_1(0)$	$S_2(0)$	W_1
2		$I_1+S_2(0)$	$I_0+S_3(0)+S_2(0)$	$S_0(0)+S_3(0)$	$S_1(0)$	

If the inputs are taken in groups of two, then the parallel machine will make a transition from the state at time t = 0 to the state at time t = 2. Therefore the entries in the table for t = 2 describe the feedback connections for the device and the output column describes the output connections. By inspection of Table 1, the next-state equations for the parallel machine are

$$S_0(t+1) = I_1+S_2(t)$$
$$S_1(t+1) = I_0+S_2(t)+S_3(t)$$
$$S_2(t+1) = S_0(t)+S_3(t)$$
$$S_3(t+1) = S_1(t).$$

The output equations for the parallel machine are

$$W_0(t) = S_3(t)$$
$$W_1(t) = S_2(t).$$

The machine to realize the above sets of equations is shown in Figure 6. Table 2 is a flow table demonstrating the division of $x^8+x^6+x^5$ by $g(x) = 1+x+x^4$ in the parallel machine. By long division, the quotient is x^4+x^2+1 and the remainder is x^3+x^2+x+1 and these results can be verified

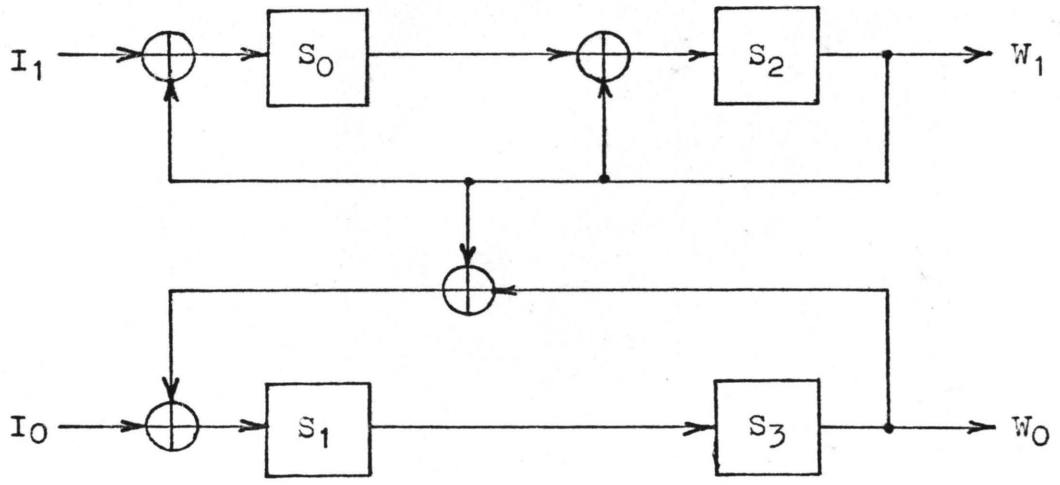


FIGURE 6

Two-Channel Parallel LFSR

in Table 2. The input to the parallel machine is

0 0 0 0 0 1 1 0 1

which is 9 bits. Since 9 is not evenly divisible by two, a zero is affixed in front of the rightmost coefficient to provide

0 0 0 0 0 1 1 0 1 0

where the two rightmost digits are the input at time t = 0, the next two digits are the input at time t = 1, and so forth. This step may be justified by considering the message in polynomial form. Then the rightmost digit is the coefficient of x^{n-1} and affixing the zero to the right of this coefficient amounts to adding $(0)x^n$ to the polynomial which obviously does not change the polynomial and therefore cannot change the remainder obtained after division by g(x).

TABLE 2

Flow Table for a Parallel LFSR

Time	I_1	I_0	S_0	S_1	S_2	S_3	W_0	W_1
0	1	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0
2	0	1	1	0	1	0	0	1
3	0	0	1	0	1	0	0	1
4	0	0	1	1	1	0	0	1
5			1	1	1	1		1

The remainder is in the register after the 5th clock time. The coefficients of x^3 is S_3, x^2 is S_2, x is S_1, and x^0 is S_0. The quotient is provided as follows:

At t = 2 the value of W_1 is the coefficient of x^4.

At t = 3 W_0 and W_1 are the coefficients of x^3 and x^2 respectively.

At t = 4 W_0 and W_1 are the coefficients of x and x^0 respectively.

The method of serial-to-parallel transformation just described is straightforward and intuitively understandable. It does have the major disadvantage of being tedious and unwieldy if g(x) is large or if the desired value of f (the number of parallel channels) is large, because the state of the machine at each clock time must be investigated.

Hsiao and Sih present a second method for serial-to-parallel conversion which is described below. This technique provides the interconnection between memory elements for the parallel machines but provides neither information about the connection of inputs nor information about connections for providing outputs.

The technique utilizes the property of linear feedback shift registers that, if the register is constructed according to a polynomial g(x) and if α is a root of g(x), then the register will give successive powers of α as it is shifted. (This property is described in detail in Chapter 7 of reference 2). The LFSR described in the preceding section will be used here to demonstrate the procedure for obtaining the interconnection information for a two channel parallel machine. The generator polynomial for this machine is

$$g(x) = 1+x+x^4.$$

Therefore, if α is a root of g(x), it must be true that

$$1+\alpha+\alpha^4 = 0$$

or, because the addition is modulo-2

$$\alpha^4 = 1+\alpha.$$

The contents of the four-stage shift register of this example can be described by the polynomial r(x) where

$$r(x) = S_0+S_1x+S_2x^2+S_3x^3.$$

Now if x is replaced by α then it is true that

$$r(\alpha) = S_0+S_1\alpha+S_2\alpha^2+S_3\alpha^3.$$

If a three channel parallel LFSR is to be equivalent to the serial LFSR, then successive shifts of the

parallel register must yield successive powers of α^2. Therefore, the contents of this register must be (as shown by Peterson)

$$r(\alpha^2) = \alpha^2 r(\alpha) = S_0\alpha^2+S_1\alpha^3+S_2\alpha^4+S_3\alpha^5.$$

But, from the preceding paragraph

$$\alpha^4 = 1+\alpha$$

so that

$$\alpha^5 = \alpha(\alpha^4) = \alpha(1+\alpha) = \alpha+\alpha^2.$$

Therefore substitution of the values for α^4 and α^5 into the equation for $r(\alpha^2)$ yields

$$\begin{aligned} r(\alpha^2) &= S_0\alpha^2+S_1\alpha^3+S_2(1+\alpha)+S_3(\alpha+\alpha^2) \\ &= S_0\alpha^2+S_1\alpha^3+S_2+S_2\alpha+S_3\alpha+S_3\alpha^2 \\ &= S_2+(S_2+S_3)\alpha+(S_0+S_3)\alpha^2+S_1\alpha^3. \end{aligned}$$

Now, $r(\alpha^2)$ may be rewritten such that

$$r(\alpha^2) = S_0'+S_1'\alpha+S_2'\alpha^2+S_3'\alpha^3.$$

Then by inspection of coefficients

$$\begin{aligned} S_0' &= S_2 \\ S_1' &= S_2+S_3 \\ S_2' &= S_0+S_3 \\ S_3' &= S_1. \end{aligned}$$

Now the above equations, if properly interpreted and if inputs are ignored, yield the interconnections described by the equations on page 12.

The method described here has two major shortcomings. A very obvious problem is that no information about input or output connections is provided. A second problem is the complexity of solving for coefficients if the generator

polynomial is large or if the desired number of parallel channels is large.

B. A Matrix Method of Transformation

The third technique described by Hsiao and Sih uses companion (connection) matrices for the serial-to-parallel transformation. The presentation in the following section was suggested by the work of Hsiao and Sih, but is more general and ultimately results in a method of transformation which is concise and simple, and lends itself to simplification of the transformed circuit.

1. Transformation of the general linear sequential machine.

The description of the LFSR in terms of generalized linear-sequential-machine theory is desirable because the generalized theory permits a concise mathematical description of such machines which in turn provides a foundation for the theory of serial-to-parallel transformation of LFSR's.

Yau and Wang[5] define a finite state sequential machine M as a system with a finite input space $\bar{I}$, a finite output space $\bar{W}$, a finite internal-state space $\bar{S}$ and two functions f and g. The functions f and g, called the next-state and output functions respectively, are described by

$$\bar{s}(t+1) = f\left[\bar{s}(t),\bar{I}(t)\right]$$
$$\bar{w}(t) = g\left[\bar{s}(t),\bar{I}(t)\right] .$$

where $\bar{s}(t)$, $\bar{I}(t)$, and $\bar{w}(t)$ are, respectively, the internal-state, input and output vectors of M and $\bar{s}(t+1)$ is the next internal-state vector of M. Furthermore, the machine

will be assumed to be deterministic, synchronous, and completely specified. The machine is deterministic if f and g are single valued at every point in their domains, synchronous if every transition is clocked, and completely specified if f and g are defined at every point in their domain. If ℓ, k, and m are the dimensions of the input space, internal-state space, and output space respectively, then $\bar{s}(t)$ is k-dimensional, $\bar{I}(t)$ is ℓ-dimensional, and $\bar{w}(t)$ is m-dimensional.

Let $\bar{S}x\bar{I}$ be the Cartesian product of $\bar{S}$ and $\bar{I}$ so that, for a general sequential machine, the next-state function f and output function g are mappings among the vector spaces $\bar{I}$, $\bar{S}$, $\bar{W}$, and $\bar{S}x\bar{I}$. Then, according to Yau and Wang, a machine M shall be called linear if and only if f is a linear transformation from $\bar{S}x\bar{I}$ into $\bar{S}$, and g is a linear transformation from $\bar{S}x\bar{I}$ onto $\bar{W}$.

Yau and Wang[5] prove that a sequential machine M is linear if and only if there exist transformations

$$f_1: \bar{S} \longrightarrow \bar{S}$$
$$f_2: \bar{I} \longrightarrow \bar{I}$$
$$g_1: \bar{S} \longrightarrow \bar{W}$$
$$g_2: \bar{I} \longrightarrow \bar{W}$$

such that

$$f(\bar{s},\bar{I}) = f_1(\bar{s})+f_2(\bar{I})$$
$$g(\bar{s},\bar{I}) = g_1(\bar{s})+g_2(\bar{I})$$

for every $\bar{s}$ in $\bar{S}$ and every $\bar{I}$ in $\bar{I}$. Further, Birkhoff and MacLane[6] prove that there is a one-to-one correspondence

between a linear transformation of the type

$$f_1 : \bar{S} \longrightarrow \bar{S}$$

and a kxk matrix A with elements from the field F, where $\bar{S}$ is a k-dimensional vector space over a scalar field F.

The linear sequential machine M can therefore be described in terms of the equations

$$\bar{s}(t+1) = T\bar{s}(t)+B\bar{I}(t)$$

$$\bar{w}(t) = C\bar{s}(t)+D\bar{I}(t)$$

where T is a kxk matrix, B is a kxℓ matrix, C is an mxk matrix, and D is an mxℓ matrix.

The serial-to-parallel transformation problem can now be considered in terms of general machine theory. Given a linear sequential machine M with ℓ input terminals and m output terminals, find an f channel analog of M, called M', with $(f)\cdot(\ell)$ input terminals and $(f)\cdot(m)$ output terminals. This means that machine M' must accept f of the ℓ-dimensional input vectors simultaneously and must provide f m-dimensional output vectors simultaneously. If the state vectors for machine M are $\bar{s}(t)$, $\bar{s}(t+1)$, . . . , then machine M' must transition from $\bar{s}(t)$ to $\bar{s}(t+f)$ in a single clock time. From this description it follows that M' will operate f times as fast as M.

The serial-to-parallel transformation of a linear sequential machine is described by the following theorem proved by Gill[4]. The proof of the theorem presented in this paper is an alternate to Gill's proof and is included because it is more easily understood in terms of general linear-sequential-machine theory.

Theorem

Given a linear sequential machine M described by the set of equations:

$$\bar{s}(t+1) = T\bar{s}(t)+B\bar{i}(t)$$

$$\bar{w}(t+1) = C\bar{s}(t)+D\bar{i}(t),$$

the f-channel analog of M, M', is described by the set of equations:

$$\bar{s}'(t+1) = T'\bar{s}'(t)+B'\bar{i}'(t)$$

$$\bar{w}'(t) = C'\bar{s}'(t)+D'\bar{i}'(t).$$

Where:

$$\bar{s}'(t) = \bar{s}(t)$$

$$\bar{s}'(t+1) = \bar{s}(t+f)$$

$$\bar{i}'(t) = \begin{bmatrix} \bar{i}(t) \\ \bar{i}(t+1) \\ \vdots \\ \bar{i}(t+f-1) \end{bmatrix}$$

$$\bar{w}'(t+1) = \begin{bmatrix} \bar{w}(t) \\ \bar{w}(t+1) \\ \vdots \\ \bar{w}(t+f) \end{bmatrix}$$

$$T' = T^f$$

$$B' = \begin{bmatrix} T^{f-1}B & T^{f-2}B & \cdots & TB & B \end{bmatrix}$$

$$C' = \begin{bmatrix} C \\ CT \\ CT^2 \\ \vdots \\ CT^{f-1} \end{bmatrix}$$

$$D' = \begin{bmatrix} D & O & O & \cdots & O & O \\ CB & D & O & & \vdots & \vdots \\ CTB & CB & D & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & D & O \\ CT^{f-2}B & CT^{f-3}B & CT^{f-4}B & \cdots & CB & D \end{bmatrix}$$

with each element of D' an $m \times \ell$ matrix.

Proof:

The next state equation for M is

$$\bar{s}(t+1) = T\bar{s}(t)+B\bar{i}(t)$$

By successive substitution, the states at later times in terms of s(t) are

$$\bar{s}(t+2) = T\bar{s}(t+1)+B\bar{i}(t+1)$$

$$\bar{s}(t+2) = T\{T\bar{s}(t)+B\bar{i}(t)\} +B\bar{i}(t+1)$$

$$\bar{s}(t+2) = T^2\bar{s}(t)+TB\bar{i}(t)+B\bar{i}(t+1)$$

$$\bar{s}(t+3) = T\bar{s}(t+2)+B\bar{i}(t+2)$$

$$\bar{s}(t+3) = T\{T^2\bar{s}(t)+TB\bar{i}(t)+B\bar{i}(t+1)\} +B\bar{i}(t+2)$$

$$\bar{s}(t+3) = T^3\bar{s}(t)+T^2B\bar{i}(t)+TB\bar{i}(t+1)+B\bar{i}(t+2)$$

and, proceeding by induction

$$\bar{s}(t+f) = T^f\bar{s}(t)+T^{f-1}B\bar{i}(t)+T^{f-2}B\bar{i}(t+1)+ \cdots +B\bar{i}(t+f-1).$$

Rewriting in matrix form

$$\bar{s}(t+f) = T^f\bar{s}(t)+\begin{bmatrix} T^{f-1}B & T^{f-2}B & \cdots & B \end{bmatrix}\begin{bmatrix} \bar{i}(t) \\ \bar{i}(t+1) \\ \vdots \\ \bar{i}(t+f-1) \end{bmatrix}$$

Therefore, by inspection, the above equation can be rewritten

$$\bar{s}'(t+1) = T'\bar{s}'(t)+B'\bar{i}'(t)$$

and the proof is complete for the next state equation.

The output equation for M is

$$\bar{w}(t) = C\bar{s}(t)+D\bar{i}(t).$$

Again, by successive substitution, the outputs at later times in terms of $\bar{s}(t)$ are

$$\bar{w}(t+1) = C\bar{s}(t+1)+D\bar{i}(t+1)$$

$$\bar{w}(t+1) = C\left\{T\bar{s}(t)+B\bar{i}(t)\right\}+D\bar{i}(t+1)$$

$$\bar{w}(t+1) = CT\bar{s}(t)+CB\bar{i}(t)+D\bar{i}(t+1)$$

$$\bar{w}(t+2) = C\bar{s}(t+2)+D\bar{i}(t+2)$$

$$\bar{w}(t+2) = C\left\{T^2\bar{s}(t)+TB\bar{i}(t)+B\bar{i}(t+1)\right\}+D\bar{i}(t+2)$$

$$\bar{w}(t+2) = CT^2\bar{s}(t)+CTB\bar{i}(t)+CB\bar{i}(t+1)+D\bar{i}(t+2)$$

$$\vdots \qquad\qquad \vdots$$

$$\bar{w}(t+f-1) = C\bar{s}(t+f-1)+D\bar{i}(t+f)$$

$$\bar{w}(t+f-1) = C\left\{T^{f-1}\bar{s}(t)+T^{f-2}B\bar{i}(t)+\ .\ .\ .+B\bar{i}(t+f-2)\right\}+D\bar{i}(t+f-1)$$

$$\bar{w}(t+f-1) = CT^{f-1}\bar{s}(t)+CT^{f-2}B\bar{i}(t)+\ldots+CB\bar{i}(t+f-2)+D\bar{i}(t+f-1).$$

Thus, the output equations that have been developed are

$$\bar{w}(t) = C\bar{s}(t)+D\bar{i}(t)$$

$$\bar{w}(t+1) = CT\bar{s}(t)+CB\bar{i}(t)+D\bar{i}(t+1)$$

$$\bar{w}(t+2) = CT^2\bar{s}(t)+CTB\bar{i}(t)+CB\bar{i}(t+1)+D\bar{i}(t+2)$$

$$\vdots \qquad\qquad \vdots$$

$$\bar{w}(t+f-1) = CT^{f-1}\bar{s}(t)+CT^{f-2}B\bar{i}(t)+\ldots+CB\bar{i}(t+f-2)+D\bar{i}(t+f-1).$$

Rewriting the above set of equations in matrix form yields

$$\begin{bmatrix} \bar{w}(t) \\ \bar{w}(t+1) \\ \bar{w}(t+2) \\ \vdots \\ \bar{w}(t+f-1) \end{bmatrix} = \begin{bmatrix} C \\ CT \\ CT^2 \\ \vdots \\ CT^{f-1} \end{bmatrix} \bar{s}(t) + \begin{bmatrix} D & & \\ CB & D & \\ CTB & CB & \\ \vdots & & \\ CT^{f-2}B & CT^{f-3}B \ldots & D \end{bmatrix} \begin{bmatrix} \bar{i}(t) \\ \bar{i}(t+1) \\ \bar{i}(t+2) \\ \vdots \\ \bar{i}(t+f-1) \end{bmatrix}$$

Thus the above equations may be written

$$\bar{w}'(t) = C'\bar{s}'(t)+D'\bar{i}'(t)$$

and the proof is complete.

This theorem is applicable to any linear sequential machine. For the case where the serial machine is an LFSR with only one input and one output the equations are quite simple.

2. Transformation of linear feedback shift registers.

A binary LFSR with a one-dimensional input space and a one-dimensional output space is described by the equations

$$\bar{s}(t+1) = T\bar{s}(t)+B\bar{i}(t)$$

$$\bar{w}(t) = C\bar{s}(t)+D\bar{i}(t)$$

If the shift register has k memory elements, then the state vectors form a k-dimensional vector space. Thus $\bar{s}(t)$ is written as a column vector

$$\bar{s}(t) = \begin{bmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \\ \vdots \\ s_{k-1}(t) \end{bmatrix}.$$

If the machine has a one dimensional input space the matrix B is a kx1 matrix. A "1" in the i^{th} row of B indicates that the input is connected to the i^{th} memory element. Matrix T is a kxk matrix. A "1" in the i^{th} row and j^{th} column of T indicates a connection from the j^{th} memory element to the i^{th}.

For example, a typical LFSR with three memory elements is described by the next-state equation

$$\begin{bmatrix} s_0(t+1) \\ s_1(t+1) \\ s_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} i(t)$$

or alternately

$$s_0(t+1) = s_0(t)+s_2(t)+i(t)$$
$$s_1(t+1) = s_0(t)+s_1(t)$$
$$s_2(t+2) = s_1(t)+i(t).$$

Again as an example, the output equation for this machine might be

$$w(t) = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} i(t)$$

which yields

$$w(t) = s_0(t)+s_2(t)+i(t).$$

It is easily verified by inspection of Figure 7 that the LFSR shown in the figure corresponds to the next-state and output equations given above. It should be noted that the matrices T, B, C, and D contain all the information necessary to construct the LFSR if the matrices are properly interpreted.

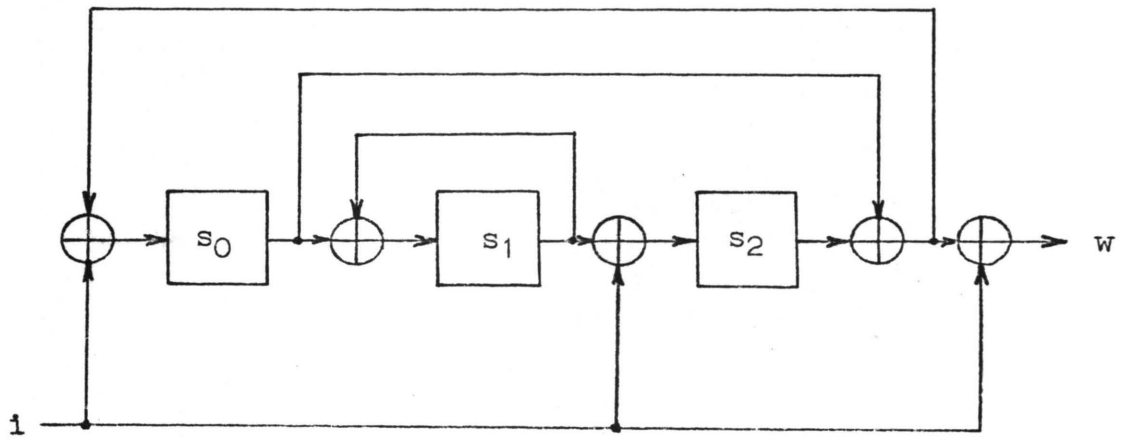


FIGURE 7

A Three-Stage LFSR

The serial LFSR for encoding and decoding is described by a generator polynomial such that each stage of the register except the last one feeds only the succeeding stage. The last stage is fed back to preceding stages according to coefficients of the generator polynomial. A comparison of the serial LFSR of Figure 7 with the serial LFSR shown in Figure 2 on page 6 clearly shows their differences. The connection matrix, T, for the serial LFSR which accomplishes division by a polynomial is formed by constructing the companion matrix of the generator polynomial. According to Birkhoff and MacLane[6] the companion matrix for a polynomial

$$g(x) = a_0 + a_1 x + \ldots + a_k x^k$$

is

$$T_c = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & -a_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -a_{k-1} \end{bmatrix}$$

For a binary machine the coefficients a_i are elements from GF [2] so that

$$-a_i \equiv a_i \text{ modulo } 2$$

and therefore each element in the last column of T_c is either one or zero.

The input for the machine is connected only to the first memory element and the output is the output of the last memory element. Thus the equations describing a serial linear-feedback-shift-register which accomplishes division by the generator polynomial are

$$\begin{bmatrix} s_0(t+1) \\ s_1(t+1) \\ \vdots \\ s_{k-1}(t+1) \end{bmatrix} = T_c \begin{bmatrix} s_0(t) \\ s_1(t) \\ \vdots \\ s_{k-1}(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} i(t)$$

$$w(t) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0(t) \\ s_1(t) \\ \vdots \\ s_{k-1}(t) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} i(t).$$

Example:

The companion matrix to the generator polynomial

$$g(x) = 1+x+x^4$$

is

$$T_c = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} .$$

Therefore the next-state equation is

$$\begin{bmatrix} s_0(t+1) \\ s_1(t+1) \\ s_2(t+1) \\ s_3(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \\ s_3(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} i(t)$$

and the output equation is

$$\begin{bmatrix} w(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \\ s_3(t) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} i(t) .$$

From these equations

$$s_0(t+1) = s_3(t)+i(t)$$
$$s_1(t+1) = s_0(t)+s_3(t)$$
$$s_2(t+1) = s_1(t)$$
$$s_3(t+1) = s_2(t)$$

and

$$w(t) = s_3(t).$$

These equations completely describe the LFSR shown in Figure 2 on page 6.

The serial-to-parallel transformation for the general linear sequential machine applies of course to the LFSR. Thus, if the serial machine is to be transformed to an f-channel analog, the describing equations become

$$\bar{s}'(t+1) = T_c'\bar{s}'(t)+B'\bar{i}'(t)$$

$$\bar{w}'(t+1) = C'\bar{s}'(t)+D'\bar{i}'(t)$$

where T_c', B', C', and D' are as described previously for linear sequential machines. Forming these matrices is quite simple as shown in the following example.

Example:

Given that a serial LFSR has a generator polynomial

$$g(x) = 1+x+x^4$$

and that a two-channel analog of this machine is desired (i.e. f = 2), then

$$T_c' = T_c^2 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$B' = \begin{bmatrix} TB & B \end{bmatrix}$$

$$T_cB = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

therefore

$$B' = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C' = \begin{bmatrix} C \\ CT \end{bmatrix}$$

$$CT_c = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

therefore

$$C' = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D' = \begin{bmatrix} D & \\ CB & D \end{bmatrix}$$

$$CB = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}$$

therefore

$$D' = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} .$$

Thus, the equations for the parallel analog are

$$\begin{bmatrix} s_0'(t+1) \\ s_1'(t+1) \\ s_2'(t+1) \\ s_3'(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_0'(t) \\ s_1'(t) \\ s_2'(t) \\ s_3'(t) \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} i_0(t) \\ i_1(t) \end{bmatrix}$$

where the input $i_1(t)$ is the input which was described as $i(t+1)$ in the serial machine, and

$$\begin{bmatrix} w_0(t) \\ w_1(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0'(t) \\ s_1'(t) \\ s_2'(t) \\ s_3'(t) \end{bmatrix}$$

where the output $w_1(t)$ is the output which was described as $w(t+1)$ in the serial machine. It should be noted that the machine described by the above equations is identical to that shown in Figure 6 on page 13.

The serial-to-parallel transformation of an LFSR, as shown by the previous example, consists of manipulation of four matrices. The techniques for obtaining the matrices for the parallel machine are straightforward but can be tedious for large matrices. Because the transformations are expressed mathematically they may be readily programmed on a digital computer, as indicated by the simplicity of the program in Appendix A. In addition, inspection of the T_c and B matrices reveals that the matrices T_c' and B' can be formed very easily.

The f-channel analog of an LFSR having companion matrix T_c will have an interconnection matrix $T_c' = T_c^f$. Obviously T_c^f can be found by repeatedly multiplying T_c by itself, but a simpler way of finding T_c^f will now be developed. If T_c is a kxk companion matrix with elements t_{ij} and Q is any kxk matrix with elements q_{ij}, then the product matrix QT_c with elements $(qt)_{ij}$ has the following properties:

1. $(qt)_{ij} = q_{i(j+1)}$ for $j = 1, 2, \ldots k-1$
2. $(qt)_{ik} = q_{i1}t_{1k}+q_{i2}t_{2k}+ \ldots +q_{ik}t_{kh}$

This can be demonstrated by multiplying the matrices:

$$QT_c = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1k} \\ q_{21} & & & \\ \vdots & & & \\ q_{k1} & \cdot & \cdot & q_{kk} \end{bmatrix} \begin{bmatrix} 0 & 0 & \cdots & 0 & t_{1k} \\ 1 & 0 & \cdots & 0 & t_{2k} \\ 0 & 1 & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & 0 & \\ 0 & 0 & & 1 & t_{kk} \end{bmatrix} .$$

It is important to note that

$$t_{ij} = 1 \text{ for } j = i-1 \text{ and } j \neq k$$
$$= 0 \text{ for } j \neq i-1 \text{ and } j \neq k$$

Thus property 1 becomes obvious upon multiplication of the two matrices. Elements of the k^{th} column of T_c are coefficients of the characteristic polynomial associated with the companion matrix. Therefore elements of the k^{th} column of the product matrix are calculated using conventional matrix multiplication.

Matrix T_c^2 can be calculated by application of the two properties as follows:

$$T_c^2 = T_c \cdot T_c = Q \cdot T_c .$$

Then the first k-1 columns of T_c^2 are formed by shifting columns 2, 3, . . . k of T_c to the left one place according to property 1. The k^{th} column of T_c^2 is formed by observing that t_{1k}, t_{2k}, . . . t_{kk} are coefficients of the generator polynomial

$$g(x) = t_1 + t_2 x + \ldots + t_k x^k .$$

The t_i's are either 0 or 1 depending upon the particular generator polynomial. Thus any element $(qt)_{ik}$ in the last column of T_c^2 is simply the sum of the elements in row 1

of T_c which are in a column corresponding to a nonzero t_i. To find T_c^3 the process is repeated so that

$$T_c^3 = T_c^2 T_c$$

where now $T_c^2 = Q$. To obtain T^f, this process must be repeated for T_c^4, T_c^5 and so on for a total of f-1 iterations.

Example:

If $g(x) = 1+x+x^3+x^4$ and $f = 3$, then

$$T_c = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Then column 1 of T_c^2 is column 2 of T_c, column 2 of T_c^2 is column 3 of T_c, and column 3 of T_c^2 is column 4 of T_c. The 4th column of T_c indicates that the element in row i of the 4th column of T_c^2 is to be a linear sum of the elements in row i and columns 1, 2, and 4 of T_c. Column 3 is ignored because element 3 in column 4 of T_c is 0. Thus, T_c^2 is formed by shifting columns to the left, then adding the elements in each row modulo 2 to form each element in the 4th column to obtain

$$T_c^2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

Then T_c^3 is formed by iterating the process so that

$$T_c^3 = T_c^2 T_c = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} .$$

Investigation of properties of the B and B' matrices also reveals a simple method for forming B'. For serial LFSR's where the input is connected only to the first state the B matrix is

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 0 \end{bmatrix} .$$

Therefore the product of a square matrix Q premultiplying B yields a column matrix which is the first column of Q. This statement can be verified by matrix multiplication. It has been shown that

$$B' = \begin{bmatrix} T^{f-1}B & T^{f-2}B & \cdots & TB & B \end{bmatrix}$$

so that the f^{th} column of B' is B, the $(f-1)^{th}$ column is the 1^{st} column of T, the $(f-2)^{nd}$ is the 1^{st} column of T^2, and so fort h. Thus if the example used for the T_c matrix, with

$$g(x) = 1+x+x^3+x^4$$

and f = 3, is used, the B' matrix can be written by inspe ction of the matrices T_c and T_c^2 as

$$B' = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} .$$

The matrix method of transforming serial LFSR's to equivalent parallel machines requires only a knowledge of the generator polynomial for the serial machine and the number of parallel channels desired in the parallel machine. A computer program, described in Appendix A, has been written such that, if g(x) and f are specified, the resulting T' and B' matrices are supplied by the program. Thus transformation becomes completely automatic to the program user.

If the output equation were also desired, the program could be readily expanded to perform calculation of the C' and D' matrices. It should also be noted here that, for a general linear sequential machine, the transformation program would not be very difficult. However, more computer time would be required because the simplification described earlier applies only to the very special class of LFSR's used for encoding and decoding polynomial codes.

III. REDUCTION OF MACHINE COMPLEXITY

The next-state and output equations for the parallel LFSR, as derived using the foregoing algorithm, do not necessarily specify the simplest machine. The simplest machine shall be defined as the machine which requires the least number of modulo 2 adders under the following constraints:

1) Only two-input adders are to be used
2) No adders will be shared.

The first constraint was chosen because most manufacturers of integrated circuits list two-input exclusive-or gates (which are equivalent to modulo 2 adders) as one of their "off-the-shelf" items. There is no apparent pattern to the way in which the adders are shared so that the second constraint seems necessary if some criterion of machine complexity is to be established. It should be noted that the second constraint is meant to imply that sharing of adders is permitted only after the simplest machine, in terms of the above constraints, is obtained.

Under these constraints the number of adders required for realization of a machine can be determined by investigation of the T', B', C', and D' matrices. For example, in the next-state equation the first row of the T' and B' matrices indicate inputs to the first stage (s_0). More specifically, the input to the first stage is the modulo 2 sum of all feedback connections indicated by "ones"

entered in the T' and B' matrices. Clearly, then the number of two-input adders needed for the first stage is one less than the total number of ones in the first rows of matrices T' and B'. The number of adders required for succeeding stages is determined in a similar fashion. It follows that the total number of two-input adders required for realizing the state equation is the total number of ones in both matrices minus the number of rows (k) in the matrices.

From the above it is apparent that some method for reducing the number of ones in the T', B', C', and D' matrices is desirable. This section of the paper presents an algorithm which tends to reduce the number of ones in the T' and B' matrices. However, before a formal algorithm is developed it is necessary to:

1) prove that premultiplication of the next-state equation of a linear sequential machine M, by any nonsingular matrix Q, provides a new machine M' isomorphic to machine M,
2) demonstrate the usefulness of machine M' as an encoder or decoder, or both,
3) demonstrate that premultiplication of the next-state equation of M by matrix Q can result in circuit simplification.

After the three problems listed above have been solved the algorithm will be developed.

A. Isomorphism in Decoders

It has been shown earlier in this report that a linear sequential machine M is described by the vector spaces $\bar{S}$, $\bar{I}$, and $\bar{W}$ and two linear functions f and g. The vector spaces $\bar{S}$, $\bar{I}$, and $\bar{W}$ are, respectively the next-state, input and output spaces and the functions f and g are, respectively, the input and output functions. Thus, for conciseness it can be said that

$$M = (\bar{S}, \bar{I}, \bar{W}, f, g)$$

where the symbols are as described above.

Given some linear sequential machine M, the next-state and output equations are

$$f(\bar{s},\bar{I}) = \bar{s}(t+1) = T\bar{s}(t)+B\bar{I}(t)$$

$$g(\bar{s},\bar{I}) = \bar{w}(t) = C\bar{s}(t)+D\bar{I}(t).$$

It will be assumed that the state space is k dimensional. Then if the next-state equation is premultiplied by some nonsingular kxk matrix Q the result is

$$Q\bar{s}(t+1) = QT\bar{s}(t)+QB\bar{I}(t)$$

which can also be written

$$Q\bar{s}(t+1) = QTQ^{-1}Q\bar{s}(t)+QB\bar{I}(t).$$

Then a new state vector $\bar{\sigma}(t)$ can be defined such that

$$Q\bar{s}(t) = \bar{\sigma}(t)$$

so that

$$Q\bar{s}(t+1) = \bar{\sigma}(t+1).$$

Then the next-state equation can be rewritten

$$\bar{\sigma}(t+1) = QTQ^{-1}\,\bar{\sigma}(t)+QB\bar{I}(t).$$

The output equation

$$\bar{w}(t) = C\bar{s}(t)+D\bar{I}(t)$$

can be written

$$\bar{w}(t) = CQ^{-1}Q\bar{s}(t)+D\bar{i}(t).$$

But

$$Q\bar{s}(t) = \bar{\sigma}(t)$$

so that the output equation becomes

$$\bar{w}(t) = CQ^{-1}\bar{\sigma}(t)+D\bar{i}(t).$$

Thus, as indicated by the next-state and output equations, a new machine M' has been described such that

$$M' = (\bar{S}', \bar{I}', \bar{W}', f', g')$$

where

$$f'(\bar{\sigma},\bar{i}) = \bar{\sigma}(t+1) = QTQ^{-1}\bar{\sigma}(t)+QB\bar{i}(t)$$

$$g'(\bar{\sigma},\bar{i}) = \bar{w}(t) = CQ^{-1}\bar{\sigma}(t)+D\bar{i}(t).$$

It is helpful at this point to consider the physical implications of the process described above. A comparison of the next-state equation for machines M and M' shows that the interconnections between memory elements are described by matrix T for machine M and by matrix QTQ^{-1} for machine M'. Similarly, the connection of inputs to memory elements is described by matrix B for machine M and by matrix QB for machine M'. A comparison of the output equations reveals different interconnection matrices, C for machine M and CQ^{-1} for machine M', but identical input-connection (D) matrices. Obviously then the two machines M and M' "look different", but it will be shown that the machines are isomorphic.

Two linear sequential machines are isomorphic if the two machines are identical except for a relabelling of

the inputs, internal states and outputs. That is to say, two linear sequential machines $M = (\overline{S}, \overline{I}, \overline{W}, f, g)$ and $M' = (\overline{S}', \overline{I}', \overline{W}', f', g')$ are isomorphic if and only if there exist three one-to-one mappings

$$h_1: \overline{S} \longrightarrow \overline{S}'$$
$$h_2: \overline{I} \longrightarrow \overline{I}'$$
$$h_3: \overline{W} \longrightarrow \overline{W}'$$

such that

$$h_1\left[f(\overline{s},\overline{i})\right] = f'\left[h_1(\overline{s}),\ h_2(\overline{i})\right]$$
$$h_3\left[g(\overline{s},\overline{i})\right] = g'\left[h_1(\overline{s}),\ h_2(\overline{i})\right] .$$

Thus to show that machine M', obtained by multiplying the next-state equation of M by Q, is isomorphic to M, the mappings h_1, h_2, and h_3 must first be identified. It will be assumed that the correspondences

$$h_1 \longleftrightarrow Q$$
$$h_2 \longleftrightarrow I_k$$
$$h_3 \longleftrightarrow I_k$$

exist where Q is the nonsingular kxk matrix described earlier and I_k is the kxk identity matrix. Then the mappings must be tested to determine whether they satisfy the relationships

$$h_1\left[f(\overline{s},\overline{i})\right] = f'\left[h_1(\overline{s}),\ h_2(\overline{i})\right]$$
$$h_3\left[g(\overline{s},\overline{i})\right] = g'\left[h_1(\overline{s}),\ h_2(\overline{i})\right].$$

Investigation of the next state equation reveals that

$$h_1\left[f(\overline{s},\overline{i})\right] = Q\left[f(\overline{s},\overline{i})\right] = QT\overline{s}(t)+QB\overline{i}(t)$$

and that

$$f'\left[h_1(\overline{s}),h_2(\overline{i})\right] = f'\left[Q\overline{s},\overline{i}\right] = QTQ^{-1}\cdot Q\overline{s}(t)+QB\overline{i}(t)$$
$$= QT\overline{s}(t)+QB\overline{i}(t).$$

Inspection of the two equations above verifies that

$$h_1\left[f(\bar{s},\bar{i})\right] = f'\left[h_1(\bar{s}),h_2(\bar{i})\right].$$

Now, investigation of the output equation reveals that

$$h_3\left[g(\bar{s},\bar{i})\right] = g(\bar{s},\bar{i}) = C\bar{s}(t)+D\bar{i}(t)$$

and that

$$g'\left[h_1(\bar{s}),h_2(\bar{i})\right] = g'\left[Q\bar{s},\bar{i}\right] = CQ^{-1}Q\bar{s}(t)+D\bar{i}(t)$$
$$= C\bar{s}(t)+D\bar{i}(t).$$

Inspection of the above equations verifies that

$$h_3\left[g(\bar{s},\bar{i})\right] = g'\left[h_1(\bar{s}),h_2(\bar{i})\right].$$

Thus, it follows that machine M' is isomorphic to machine M and that the only difference between M and M' is a relabelling of the internal states.

At this point it has been shown that a linear sequential machine M' can be constructed which differs from a linear sequential machine M only in the labelling of its internal states. It remains to be shown that construction of such a machine can lead to realizations which are simpler than the original machine and that a communication system which uses cyclic error-detecting codes can be constructed using the simpler machines.

A communication system in which a cyclic error-detecting code is used will now be defined. The encoder shall be an LFSR which provides a remainder in q clock-times, that is, a parallel LFSR. If the encoder is called machine M, then the decoder shall be denoted M', where M' is the machine obtained by multiplication of the next-state equation of M by a nonsingular matrix Q. Then for the

system thus defined, attempts at simplifying the decoder are permitted; but the encoder must not be changed. Justification for defining the communication system in this way requires consideration of the functions of the decoder and encoder.

First, the decoder receives a coded message which shall be called $m^*(x)$. The decoder must divide $m^*(x)$ by $g(x)$ and form a remainder $r^*(x)$. If $r^*(x) = 0$, then no detectable errors have occurred during transmission; but any nonzero remainder indicates error. It has previously been shown that the coefficients of the remainder are the elements of the state vector of the machine after q clock times. The states of machine M and M' are, respectively, $\overline{s}(q)$ and $Q\overline{s}(q)$ after q clock times (assuming the machines are initially in the zero state). Thus if state $\overline{s}(q)$ is the zero vector, then it follows that state $Q\overline{s}(q)$ is also the zero vector. This ensures that, if no errors have occurred, either machine will provide a zero remainder. On the other hand, if a detectable number of errors has occurred machine M would provide a nonzero remainder, but machine M' would also have a nonzero remainder so that it is clear that either machine can be used as a decoder.

The function of the encoder is to divide the message polynomial by the generator polynomial and provide a remainder. The encoded message is of the form

$$x^{n-k}m(x)+r(x) = g(x)q(x).$$

Suppose that machine M is in state $\overline{s}(q)$ after q clock

times and that the elements of $\bar{s}(q)$ are correct coefficients of r(x). Then a machine M' with states $Q\bar{s}(t)$ would be in state $Q\bar{s}(q)$ after q clock times. In general the elements of $Q\bar{s}(q)$ are not the correct coefficients of r(x) and therefore machine M' does not provide the desired encoding functions. Therefore, attempts at simplification are restricted to the decoder in the following development.

B. Development of an Algorithm for Machine Simplification.

It will now be shown by an example that multiplication of the next-state equation of a machine M by a nonsingular matrix Q can produce a simpler isomorphic machine M'. If the generator polynomial for a decoder is $x^5+x^4+x^2+1$, then the companion matrix is

$$T_c = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} .$$

Given that the machine is to have 6 parallel channels, a computer program written for the serial-to-parallel transformation provides the following:

$$T_c' = T_c^6 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} ; \quad B' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} .$$

In this example the first state requires five adders, the second stage four adders, and so forth, for a total of 20 adders.

Now a machine M' isomorphic to the above machine is to be specified. The nonsingular 5x5 matrix to be used for this example is

$$Q = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} .$$

In this particular case it happens that

$$Q = Q^{-1} .$$

Now the matrices for the next-state equation f' are

$$QT_c'Q^{-1} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} ; \quad QB' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} .$$

Inspection of the above matrices indicates that 16 two-input modulo 2 adders are required. Therefore a reduction of four adders is realized.

The above example has demonstrated that the Q matrix selected for the example did indeed reduce the number of adders necessary to implement the decoder. However the example does not show how to select the Q matrix which results in the simplest machine. For, although some

nonsingular matrices lead to simpler machine realizations, there are others which lead to more complex realizations. One method for finding the matrix or matrices which lead to the simplest realization of machine M' is simply to try every possible nonsingular matrix.

Suppose every possible nonsingular kxk matrix is formed, the inverse calculated, and matrix multiplication performed as described in the example. Then inspection of each set of matrices $QT_c'Q^{-1}$ and QB' will yield a simplest realization. But the number of possible nonsingular kxk matrices with elements from the field of integers modulo 2 is very large. Birkhoff and MacLane[6] prove that a kxk matrix over some field is nonsingular if and only if its rows are linearly independent. Thus, if the rows are considered vectors in a k-dimensional space, then a nonsingular matrix must have no two rows alike, and no row which is a vector sum of two or more other rows of the matrix.

These row properties can be used to construct all possible nonsingular matrices. Further, the requirement that the rows be linearly independent leads to a simple expression for the total possible number of different nonsingular kxk matrices, N_k, with elements from GF [2]. Thus it can be shown that

$$N_k = (2^k-1)(2^k-2)(2^k-4) \ldots (2^k-2^{k-1}).$$

or

$$N_k = \prod_{i=0}^{k-1} (2^k-2^i).$$

From the above expression it is found that there exist 168 different nonsingular 3x3 matrices, about 20,000 different nonsingular 4x4 matrices, and about 10 million different nonsingular 5x5 matrices. Obviously the total number of nonsingular matrices rises very rapidly with k. The process of finding all nonsingular 5x5 matrices with corresponding inverses would be a formidable task even with a computer. Clearly some method of selecting only a small number of matrices from the millions of possibilities is necessary.

Because nonsingular matrices must be formed and because the inverse of the matrix must be found in each case, two meaningful criteria for selection of the Q matrices are; 1) ease in which the matrix is formed and 2) the ease with which the matrix inverse is formed. Investigation of the properties of square matrices and identity matrices indicates some simple methods for forming nonsingular matrices and inverses. These methods will be developed in the following paragraphs.

Birkhoff and MacLane[6] prove that a kxk matrix has rank k if and only if it is row equivalent to the identity matrix I_k. By definition a matrix Q is row equivalent to matrix I_k if Q can be obtained from I_k by a finite succession of the following elementary row operations:

1) The interchange of any two rows

2) The addition of any row to any other row.

Notice that a third operation, multiplication of any row by a nonzero scalar, is omitted because the operation

is trivial in GF [2] . Birkhoff and MacLane also prove that a kxk matrix has rank k and is nonsingular if and only if its rows are linearly independent.

From the above properties it follows that any kxk matrix formed by a finite succession of elementary row operations is a nonsingular matrix. Further, the k linearly independent rows of a nonsingular matrix form a basis for the vector space $V_k(F)$ (where F is GF [2]). Thus it follows that the set of kxk matrices generated by elementary row operations on I_k is the set of all possible nonsingular kxk matrices.

To specify the set of nonsingular matrices to be tried for network simplification, some properties of the matrix inverse may be noted. Birkhoff and MacLane prove that if a kxk matrix is reduced to the identity by a sequence of row operations, the same sequence of operations applied to the identity will give the inverse of the kxk matrix. Thus if a nonsingular matrix Q is formed by applying a sequence of elementary row operations on the identity matrix, the easiest method of finding its inverse is to apply the same sequence of row operations to the matrix Q. Now the properties of vectors (or rows) with elements from GF [2] become important. If the identity matrix I_k is described in terms of the identity vectors $\bar{e}_i$ as

$$I_k = \begin{bmatrix} \bar{e}_1 \\ \bar{e}_2 \\ \vdots \\ \bar{e}_k \end{bmatrix}$$

where

$$\bar{e}_1 = [1\ 0\ 0\ \ldots\ 0]$$
$$\bar{e}_2 = [0\ 1\ 0\ \ldots\ 0]$$
$$\bar{e}_3 = [0\ 0\ 1\ \ldots\ 0]$$
$$\vdots$$
$$\bar{e}_k = [0\ 0\ 0\ \ldots\ 0\ 1]$$

then the result of adding row 1 to row 2 is

$$\begin{bmatrix} \bar{e}_1 \\ \bar{e}_1+\bar{e}_2 \\ \bar{e}_3 \\ \vdots \\ \bar{e}_k \end{bmatrix}.$$

It should be noted that to obtain the identity matrix from the above matrix the necessary row operation is addition of row 1 to row 2 because $\bar{e}_1+\bar{e}_1 \equiv 0 \bmod 2$. Thus in this case the matrix is equal to its inverse. From this example it is clear that any matrix formed by adding row 1 of the identity matrix to any other row will be equal to its inverse. Further, if any row of the identity matrix is added to any other row, the resulting matrix will be equal to its inverse. From the above an algorithm can

be developed for forming nonsingular matrices having the property that any matrix thus formed is equal to its inverse. The algorithm consists of the following steps:

1) Add row 1 to row 2 to obtain matrix Q_1 such that $Q_1 = Q_1^{-1}$.

2) Add row 1 to row 3 to obtain matrix Q_2, where

$$Q_2 = \begin{bmatrix} \bar{e}_1 \\ \bar{e}_1+\bar{e}_2 \\ \bar{e}_1+\bar{e}_3 \\ \bar{e}_4 \\ \vdots \\ \bar{e}_k \end{bmatrix}.$$

It is apparent that $Q_2 = Q_2^{-1}$ because the identity is obtained by adding row 1 to rows 2 and 3 of Q_2.

3) Repeat the process by adding row 1 to rows 4, 5, . . . n. Clearly each of the k-1 matrices formed will be of the form $Q_j = Q_j^{-1}$. It is apparent that

$$Q_{k-1} = \begin{bmatrix} \bar{e}_1 \\ \bar{e}_1+\bar{e}_2 \\ \bar{e}_1+\bar{e}_3 \\ \vdots \\ \bar{e}_1+\bar{e}_k \end{bmatrix}.$$

4) Reiterate the above process, again successively adding row 1 to each of the other rows. The first matrix formed by step 4 is Q_k where

$$Q_k = \begin{bmatrix} \bar{e}_1 \\ \bar{e}_2 \\ \bar{e}_1+\bar{e}_3 \\ \bar{e}_1+\bar{e}_4 \\ \vdots \\ \bar{e}_1+\bar{e}_k \end{bmatrix}.$$

It is easily verified that the sequence of row operations applied to I_k to form Q_k, when applied to Q_k, reduces to the identity I_k. Thus $Q_k = Q_k^{-1}$. Further $Q_{k+1} = Q_{k+1}^{-1}$ and so forth.

5) At the end of step 4, when $\bar{e}_1$ is added to $\bar{e}_k$, the resulting matrix is again the identity. Thus steps 1 through 4 can be reiterated for rows two through k to form a set of nonsingular matrices having the property that each matrix is its own inverse.

It is apparent that steps 1 through 4 of the algorithm provide 2(k-1) matrices. Thus, because the matrix has k rows, the algorithm provides $k \cdot 2(k-1)$ different nonsingular matrices for which the inverse is identical to the matrix. Then for the case where k = 5 the

algorithm provides 40 different nonsingular matrices out of a possible 10 million different nonsingular matrices.

The algorithm for simplification can be summarized as a technique for selecting a small sample of nonsingular matrices from a large population of such matrices followed by substitution of each matrix in the sample into the relationship

$$T_c^* = QT_c'Q^{-1}$$
$$B^* = QB'$$

where T_c^* and B^* are the interconnection matrices for a machine isomorphic to the original machine. The simplest machine from this sample is determined by counting the number of ones in the T_c^* and B^* matrices.

A computer program, given in Appendix A, has been written incorporating the serial-parallel transformation algorithm described earlier and the simplification algorithm described in this section. Required inputs to the program are:

1) The coefficients of the generator polynomial
2) The number of channels (f) desired in the parallel analog.

The program provides the following information to the user:

1) $T_c' = T_c^f$ and B' matrices
2) A listing of all Q matrices generated by the simplification algorithm along with the matrices $QT_c'Q^{-1}$ and QB' corresponding to each Q.

Table 3 summarizes the outcomes of several computer runs using the program discussed above. Selection of the generator polynomials for these trials was arbitrary. The number of gates listed under column A is the number required by the T_c' and B' matrices as obtained by the serial-parallel transformation algorithm. Columns B and C indicate the best and worst cases respectively obtained in the simplification attempts with Q.

TABLE 3

Summary of Results of Computer Runs

Generator Polynomial	Number of Parallel Channels (f)	Number of Two Input Adders Required		
		A	B	C
		No Simplification	Best Case	Worst Case
$1+x^2+x^4+x^5$	6	20	16	27
	8	26	22	31
	12	34	32	39
$1+x+x^2+x^4+x^5$	6	18	15	22
	8	22	20	26
	12	34	31	39
$1+x+x^2+x^4+x^5+x^7+x^9$	6	34	29	42

The table indicates that some reduction in the required number of adders was achieved in every case. The reduction ranged from 6 percent to 20 percent with an average for the seven trials of 13 percent. A comparison of the best and

worst cases indicates that, depending upon the original T' and B' matrices, reductions of up to 40 percent of the adders might be realized.

One likely reason for the apparent success of this algorithm is that the matrices constructed all have a low density of ones as elements. The desirability of this property can be demonstrated by looking at the B' matrix for a parallel LFSR. From the properties of the B' matrix described on page 33 it should be evident that the last column of B' is always the identity column $[1\ 0\ 0\ .\ .\ .\ 0]^T$. Similarly, the second last column is $[0\ 1\ 0\ 0\ .\ .\ .\ 0]^T$ and so forth so that if the B' matrix has k or less columns (k is the size of matrix T_c), then all of its columns are identity vectors. Also, if the B' matrix has more than k columns, then its last k columns are identity vectors. Investigation of a particular matrix

$$B' = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

then shows the effect of the Q matrix on the B matrix. The transformation of the B' matrix by the algorithm amounts to calculation of the product

$$QB' = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

with the result

$$QB' = \begin{bmatrix} q_{13} & q_{12} & q_{11} \\ q_{23} & q_{22} & q_{21} \\ q_{33} & q_{32} & q_{31} \\ q_{43} & q_{42} & q_{41} \end{bmatrix} .$$

Now if the Q matrix has a high density of ones this means that many of the q_{ij}'s are ones. Obviously the product matrix QB' will then have a large number of ones. But the goal of this simplification procedure is to decrease the number of ones in the B and T matrices. Therefore the above result tends to conflict with the desired result.

On the basis of the result of the computer runs and with the support of the argument just given it seems safe to conclude that the algorithm will provide simplification for a significant number of cases. Further, the computer program of Appendix A provides a useful and easy to use tool for the process of simplification.

IV. SUMMARY

This paper presents three methods for serial-to-parallel transformation of linear feedback shift registers. The best method appears to be a matrix method using the next-state and output equations of the machine based on a theorem presented by Gill. This matrix method is carefully developed, simplifications are indicated and examples for its use are provided. Next an algorithm is developed for simplifying the machines obtained by the above transformation. However it was shown that this algorithm may be used only at the decoder. The algorithm for the serial-to-parallel transformation and for circuit simplification were programmed in Fortran IV. The results of a number of sample problems demonstrates the usefulness of the simplification algorithm.

BIBLIOGRAPHY

1. PETERSON, W.W. and BROWN, D. T., Cyclic codes for error detection. Proceedings of the IRE, Vol. 49 No. 1, January, 1961. pp. 228-235.

2. PETERSON, W. W. (1961) Error correcting codes. Massachusetts, The MIT Press. 285 p.

3. HSIAO, M. Y. and SIH, K. Y., Serial-to-parallel transformation of linear-feedback shift-register circuits. IEEE Transactions on Electronic Computers Vol. EC-13, No. 6, December, 1964, pp. 738-740.

4. GILL, Arthur, On the series-to-parallel transformation of linear sequential circuits. IEEE Transactions on Electronic Computers Vol. EC-15, No. 1, February, 1966. pp. 107-108.

5. YAU, S. S. and WANG, K. C., Linearity of sequential machines. IEEE transactions on Electronic Computers. Vol. EC-15, No. 3, June, 1966, pp. 337-354.

6. BIRKHOFF, Garrett and MAC LANE, Saunders (1965) A brief survey of modern algebra (second edition). New York, The Macmillan Company. 279 p.

APPENDIX A

This appendix presents the computer program written in Fortran IV for the IBM 360 computer at the University of Missouri at Rolla.

Data supplied to the program shall include:

1. N which is the number of parallel channels desired (called f in the body of the report)
2. L which is the size of the T_c matrix (called k in the body of the report)
3. MATRIX, which is the T_c matrix.

The T_c matrix data shall be entered one row per program data card.

```
      DIMENSION MATRIX(20,20),MIN(20),LAST(20),MATB(20,40)
      DIMENSION MATQ(20,20),MATT(20,20),MATU(20,20),MATBB(20,40)
      READ(1,100) N,1
  100 FORMAT(212)
      DO 10 I=1,L
   10 READ(1,101)(MATRIX(I,J),J=1,L)
  101 FORMAT(25I1)
      DO 2 I=1,L
    2 MIN(I)=MATRIX(I,L)
      N1=N-1
      DO 40 K=1,N1
      DO 20 I=1,L
      MATB(I,N)=0
      LAST(I)=0
      K1=N-K
      MATB(I,K1)=MATRIX(I,1)
      DO 20 J=1,L
   20 LAST(I)=LAST(I)+MATRIX(I,J)*MIN(J)
      DO 30 I=1,L
      L1=L-1
      DO 30 J=1,L1
      J1=J+1
   30 MATRIX(I,J)=MATRIX(I,J1)
      DO 40 I=1,L
   40 MATRIX(I,L)=LAST(I)
      MATB(1,N)=1
      WRITE(3,102)
      DO 60 I=1,L
      DO 50 J=1,L
   50 MATRIX(I,J)=MOD(MATRIX(I,J),2)
```

```
      DO 70 JJ=1,N
   70 MATB(I,JJ)=MOD(MATB(I,JJ),2)
   60 WRITE(3,103)(MATRIX(I,J),J=1,L),(MATB(I,JJ),JJ=1,N)
  103 FORMAT (4X,60I1)
  102 FORMAT (4X,'TMATRIX BMATRIX')
      DO 105 I=1,L
  105 MATQ(I,I)=1
      WRITE(3,153)
      L1=L-1
      DO 110 KK=1,L
      DO 110 M=1,2
      DO 110 K=1,L1
      K1=KK+K
      IF(L-K1)116,117,117
  116 K1=MOD(K1,L)
  117 DO 115 J=1,L
      MATQ(K1,J)=MATQ(KK,J)+MATQ(K1,J)
  115 MATQ(K1,J)=MOD(MATQ(K1,J),2)
      DO 120 I=1,L
      DO 120 J=1,L
      MATT(I,J)=0
      DO 120 M1=1,L
  120 MATT(I,J)=MATT(I,J)+MATQ(I,M1)*MATRIX(M1,J)
      DO 130 I=1,L
      DO 130 J=1,L
      MATU(I,J)=0
      DO 135 M1=1,L
  135 MATU(I,J)=MATU(I,J)+MATT(I,M1)*MATQ(M1,J)
  130 MATU(I,J)=MOD(MATU(I,J),2)
      DO 140 I=1,L
      DO 140 J=1,N
      MATBB(I,J)=0
      DO 145 M1=1,L
  145 MATBB(I,J)=MATBB(I,J)+MATQ(I,M1)*MATB(M1,J)
  140 MATBB(I,J)=MOD(MATBB(I,J,2)
      WRITE(3,152)
      DO 110 I=1,L
  110 WRITE(3,151)(MATQ(I,J),J=1,L),(MATU(I,J),J=1,L),
                                     (MATBB(I,J),J=1,N)
  151 FORMAT(4X,80I1)
  152 FORMAT(//)
  153 FORMAT(//4X,'Q T B MATRICES')
      CALL EXIT
      END
```

VITA

The author was born on August 3, 1936, in Sheboygan, Wisconsin. He received his primary and secondary education in Sheboygan. He served four years in the U.S. Air Force. He has received his college education at Kansas City Junior College, in Kansas City, Missouri; Kansas State University, in Manhattan, Kansas; the University of Michigan, Ann Arbor, Michigan; the University of Minnesota, in Minneapolis, Minnesota; and the University of Missouri at Rolla. He received the Bachelor of Science Degree in Electrical Engineering from Kansas State University in January, 1963. He has been enrolled in the Graduate School of the University of Missouri at Rolla since September, 1965.