GENERATION METHODS, DECOMPOSITIONS AND SOME PROPERTIES OF NONSINGULAR RECURRING SEQUENCES

by

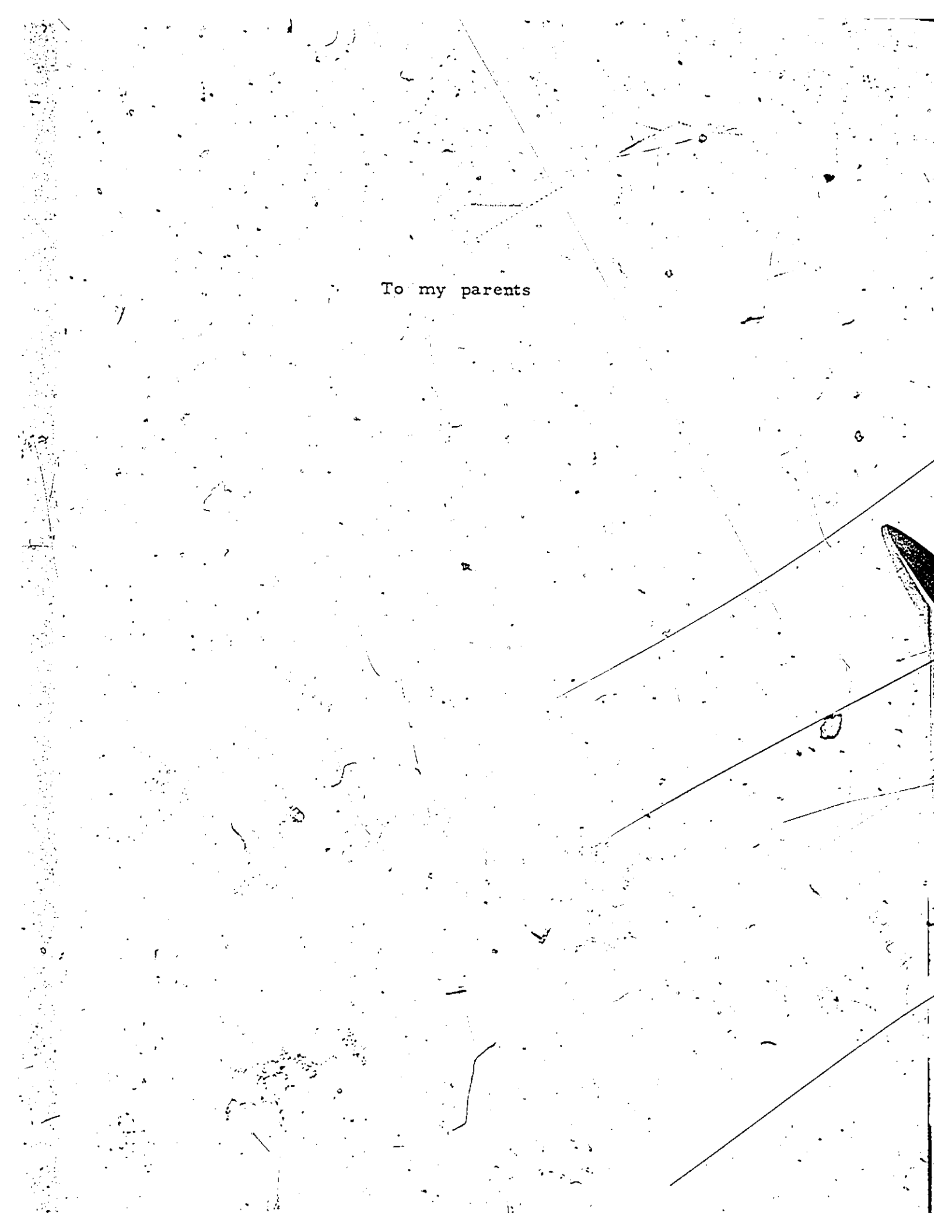
Jang Chang Hwang, M. Sc.

submitted to the School of Graduate Studies, University of Ottawa, in partial fulfilment of the requirements of the degree of Doctor of Philosophy.

Department of Electrical Engineering,
Faculty of Science and Engineering,
The University of Ottawa,
Ottawa, Canada,

June 1974

To my parents



ABSTRACT

In this thesis, several aspects of nonsingular recurring sequences, namely, generation methods, decompositions and some properties of nonsingular recurring sequences, are studied. A synthesis method to find FSR's with input sequences for generating specified sequences of finite periods is obtained. The problem to find a modulo-2-sum decomposition of linear binary sequences of finite periods is examined and a solution is obtained. A more general sequence decomposition problem, decomposing binary sequences of finite periods into logic functions of several binary sequences of finite periods, is also examined and a solution is provided for prime decompositions and conditional non-prime decompositions. A logic-function-sequence-decomposition-machine is introdued for both prime decompositions and conditional non-prime decompositions.

ACKNOWLEDGEMENTS

The author wishes to gratefully acknowledge the assistance provided by his supervisors Prof. C.L. Sheng and Prof. S.G.S. Shiva of the University of Ottawa. Their advice, encouragement and dedication were important factors in making this work both possible and profitable. To his friend, Dr. C.C. Hsieh of National Chiao Tung University, Hsinchu, Taiwan, Rep. of China, a note of thanks for many stimulating discussions and suggestions. Thanks are also due to Miss Lucie Dubeau for typing the manuscript, to Mr. René LeHénaff for his technical assistance.

The author is grateful to the National Research Council of Canada (Grant No. A-1690) and the University of Ottawa for financial assistance.

ACKNOWLEDGEMENTS

The author wishes to gratefully acknowledge the assistance provided by his supervisors Prof. C.L. Sheng and Prof. S.G.S. Shiva of the University of Ottawa. Their advice, encouragement and dedication were important factors in making this work both possible and profitable. To his friend, Dr. C.C. Hsieh of National Chiao Tung University, Hsinchu, Taiwan, Rep. of China, a note of thanks for many stimulating discussions and suggestions. Thanks are also due to Miss Lucie Dubeau for typing the manuscript, to Mr. René LeHénaff for his technical assistance.

The author is grateful to the National Research Council of Canada (Grant No. A-1690) and the University of Ottawa for financial assistance.

NOMENCLATURE

$\hat{a}, \hat{b}, \hat{c}$: sequences of finite periods

$(\hat{a}) = a_0 a_1 \cdots a_{n-1}$: a sequence $\hat{a}$ of period n with initial n-tuple $a_0 a_1 \cdots a_{n-1}$

$\hat{a} = \hat{b} \oplus \hat{c}$: sequence $\hat{a}$ is the modulo-2-sum of sequences $\hat{b}$ and $\hat{c}$, i.e., $a_i = b_i \oplus c_i$ for all $i \geq 0$

$\hat{a} = f(\hat{b}, \hat{c}, \ldots, \hat{d})$: sequence $\hat{a}$ is the logic function of sequences $\hat{b}$, $\hat{c}$, ... and $\hat{d}$, i.e., $a_i = f(b_i, c_i, \ldots, d_i)$ for all $i \geq 0$

per($\hat{a}$): the period of sequence $\hat{a}$

rev($\hat{a}$): reverse sequence of $\hat{a}$

$\bar{\hat{a}}$: complement sequence of $\hat{a}$

LCM: least common multiple

GCD: greatest common divisor

$A \supset B$: A contains B

GF(q): finite field (Galois field) of q elements

f(X): polynomial in X over GF(q)

deg f(X): the degree of f(X)

$f(X) \mid g(X)$: $f(X)$ divides $g(X)$

$f(X) \nmid g(X)$: $f(X)$ does not divide $g(X)$

n is the exponent of $f(X)$: $f(X) \mid (1 - X^n)$ but $f(X) \nmid (1 - X^i)$ for all $i < n$

$(f(X), g(X)) = 1$: $f(X)$ and $g(X)$ are relatively prime

$(f(X), g(X)) \neq 1$: $f(X)$ and $g(X)$ are not relatively prime

$D^i(\hat{a})$: ith cyclic shift of $\hat{a}$

$\lceil x \rceil$: the least integers greater than or equal to x

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Recurring sequences, also called feedback shift register sequences, have been used widely in communication systems, digital systems, electronic instrumentation, etc. Sequences possessing pseudo-noise properties have numerous applications such as unconventional radar [13, 43], radar ranging [4, 12] and Monte Carlo statistical techniques [45], among others.

Linear recurring sequences have been studied extensively by Ward [46], Zierler [49], Peterson [39], Kantz [29], Gill [19] and Golomb [23]. Nonlinear recurring sequences have been studied by Golomb [23], Magelby [33] and Brassch [7]. The contributions of Ward and Zierler established many fundamental properties of linear recurring sequences. Gilbert [17] and Golomb [23] established the pseudo-noise properties of linear shift register generators. A pseudo-noise (PN) sequence satisfies the following properties: in each period of the sequence the number of ONE's differs from the number of ZERO's by at most 1; among the runs of ONE's and of ZERO's in each period, one-half the runs of each kind are of length two, one-eigth are of length three, and so on as long as these fractions give meaningful numbers of runs; if a period of the sequence is compared, term by term, with any cyclic shift of itself,

the number of agreements differs from the number of disagreements by at most 1. All the maximum length linear feedback shift register sequences are PN sequences which can be generated by linear feedback shift registers (LSFR's) with very simple feedback logic. Other length PN sequences can be obtained from nonlinear feedback shift registers. Use a PN sequence and all its cyclic shift together with either ZERO or ONE sequence, we can construct a Hadamard matrix. Zierler [48], Golomb [23] and Elspas [14] discussed determination of cycle lengths of the register in terms of the properties of its recurring polynomial, also called characteristic polynomial. Elspas [14] and Fitzpatrick [15] also developed synthesis procedures in terms of the recurring polynomial. Massey [35] developed a LFSR synthesis method for sequences of finite lengths. The problem of constructing linear shift registers with a minimum number of adders has provoked interesting research on the theory of trinomials over the field GF(2). LFSR's corresponding to trinomials have only one adder. Many authors, among whom are Albert [2] and Golomb [23], have considered the derivation of polynomials that yield highly simplified feedback logic. The recurring polynomial of this kind is a trinomial. Roth [42] has made a study of linear binary shift register circuits using a minimum number of modulo 2 adders. The recurring polynomial is rearranged in a certain form so

that simpler configuration can be obtained. Alltop, Pratt and Burton [3] suggested a class of sequence generators using J-K flip-flops which require no adders of additonal gating.

To facilitate the study of maximal length nonlinear recurring sequences, also called nonlinear m-sequences, Good [24] introduced an important state diagram which has been named Good's diagram, also called de Bruijn graph. Good showed the existence of shift register sequence of length 2^n. de Bruijn [11] who discovered Good's diagram independently, made a significant application of it. de Bruijn proved that the exact number of sequences of length 2^n obtainable from n-stage shift registers is $_2 2^{n-1}{}_n$.

Lempel [30] developed a design method using the homomorphism of the de Bruijn graph. Lempel discussed a homomorphism of the de Bruijn graph that maps a graph of order n onto one of order n-1 and applied it to the design of nonsingular feedback shift registers. Cohn and Even [10] gave a design method to construct the shortest feedback shift register which generates a given finite sequence in the cases of linear and nonlinear feedback logic.

Introduction of nonlinear logic into a linear structure has been noted by Bryant [8], in which he generalized previous work by Heath [27]. Baumert [5] used direct logic and indirect logic FSR to generate specified sequences. Reed and Turn [41] introduced a generalized (m,n)-FSR,

associating m feedback logics with the classical n-stage FSR. The (m,n)-FSR has been shown to be capable of producing sequences of maximum period $m2^n$ for any m and n by cyclic application of properly chosen feedback functions.

Hsieh [28] used combinatorial graph to analyze the sequences and to decompose binary sequences into a modulo 2 sum or logic AND of several component sequences. Mandelbaum [34] made a comparison of linear sequential circuit sequences and arithmetic sequences. The concept of state was defined as equivalent to the remainder or residue in an arithmetic divison process. The properties of infinitely recurring sequences and terminating sequences generated by arithmetic divison and the properties of linear FSR sequences were compared.

Tanaka, Kasahara, Tezuka and Kasahara [44] developed a procedure for easily determining shift registers capable of performing multiplication or taking powers over a finite field. The shift registers are useful in performing computations involved the decoding of cyclic codes. Grodzki [25, 26] made a formal study of the FSR. He defined a k-machine and the computation of the k-machine, and obtained some principal properties of the set of all computations defined in some nonempty set (finite or infinite). The main differences of the k-machine and the conventional FSR are two-fold. First, the sequence

generated by a conventional FSR are over a finite set, while the sequences from a k-machine are over either a finite or an infinite set. Second, the feedback function of a conventional FSR is a total function defined on a finite set, while the feedback function of a k-machine is a partial function defined on either a finite or an infinite set. Hence, the k-machine actually is a generalization of the conventional FSR.

In the context of the preceding discussions, we give in this thesis several aspects of recurring sequences, namely, generation methods, decompositions and some properties of nonsingular recurring sequences. Both linear and nonlinear sequences are considered. There are two major classes of recurring sequences, namely, nonsingular sequences which are pure periodic sequences, and singular sequences which are not pure periodic sequences. In this thesis, only the nonsingular sequences will be studied.

In Chapter 2, some algebraic preliminaries, the general aspects of recurring sequences and the terms required for the development of the following chapters are discussed.

In Chapter 3, the generation methods for both linear and nonlinear recurring sequences are studied. Massey's synthesis method for a linear recurring sequence is discussed. Conventional FSR realizations for the generation of nonlinear recurring sequences using both the direct

logic FSR and indirect logic FSR are discussed. A synthesis method based on a feedback shift register with input sequence is given. Reed and Turn [41] has developed a synthesis method to find a maximum length sequence of period $m2^n$ using the generalized (m,n) - FSR. We give a synthesis method to find an FSR with input sequence for generating a specified sequence of finite period. Instead of using a multiple feedback function, a single feedback function is used. It is also shown that any attempt to shorten the length of the FSR for sequence generation only gives a composite FSR and the total number of memory elements used is not reduced when the appropriate input sequence is not available.

In Chapter 4, a method of decomposing a finite periodic binary linear sequence into a modulo 2 sum of several finite periodic binary linear sequences is presented. A LFSR realization for the generation of a given finite periodic linear sequence will give us the minimum polynomial of the sequence. A fractional polynomial representation of the sequence can be obtained and the decomposition of the sequence can be easily manipulated in the polynomial representation. A necessary and sufficient condition for the modulo-2-sum decomposibility of a linear sequence is obtained. A modulo-2-sum-sequence-decomposition-procedure is given. Some interesting properties of the minimum polynomial of a sequence are obtained.

In Chapter 5, a method of decomposing a finite periodic binary sequence into a logic function of several finite periodic binary sequences is presented. A necessary and sufficient condition for the decomposibility of a binary sequence of finite period is obtained. A partition procedure which is essential for testing the decomposibility of a sequence is given. A logic-function-sequence-decomposition-procedure for both prime decompositions and conditional non-prime decompositions is given. A logic-function-sequence-decomposition-machine is introduced for both prime decompositions and conditional non-prime decompositions.

The thesis is concluded in Chapter 6 with certain remarks.

CHAPTER 2

ALGEBRAIC PRELIMINARIES AND GENERAL ASPECTS OF RECURRING SEQUENCES

2-1. Algebraic Preliminaries

In this section, the definitions of groups, homomorphisms, rings and fields are quoted. These algebraic systems will be defined with increasing complexity from a simple algebraic system called semigroup. A more detailed discussion of these algebraic systems can be obtained from [1, 31].

2-1-1. Semigroups, monoids, groups and homomorphisms

A SEMIGROUP S is a nonempty set of elements with an associative binary operation $*$, i.e., $*: S \times S \rightarrow S$ such that $*(s_1, s_2) = s_1 * s_2$ and $(s_1 * s_2) * s_3 = s_1 * (s_2 * s_3)$ for all $s_1, s_2, s_3 \in S$.

The binary operation $*$ is COMMUTATIVE if $*(s_1, s_2) = *(s_2, s_1)$ or $s_1 * s_2 = s_2 * s_1$ for all $s_1, s_2 \in S$.

A finite semigroup can be conveniently described by a Cayley table.

A MONOID M is a semigroup with an element $e \in M$ which is a unit for $*$, i.e., $e * m = m * e = M$ for all $m \in M$.

A GROUP G is a monid with the following property, for every element $a \in G$, there exists an inverse element $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$.

If the binary operations * of a group is commutative, then the group is called ABELIAN GROUP.

Let S_1 and S_2 be two semigroups with binary operations * and ·, respectively. The mapping $\phi : S_1 \to S_2$ is a HOMOMORPHISM if $\phi(s_1 * s_2) = \phi(s_1) \cdot \phi(s_2)$ for all $s_1, s_2 \in S_1$.

Let $\phi(S_1) = \{ y \mid y \in S_2 \text{ and } y = \phi(s) \text{ for some } s \in S_1 \}$. If $\phi(S_1) = S_2$, then ϕ is an ONTO-HOMOMORPHISM. If $s_1 \neq s_2$ implies $\phi(s_1) \neq \phi(s_2)$ for all $s_1, s_2 \in S_1$, then ϕ is a ONE-TO-ONE HOMOMORPHISM.

A one-to-one and onto-homomorphism is called an ISOMORPHISM.

If the mapping $\phi : S_1 \to S_2$ is an isomorphism, we say that S_1 is isomorphic to S_2.

Group homomorphisms and isomorphisms are defined similarly.

2-1-2. Rings and fields

A RING R is a nonempty set with two associative binary operations + and *, addition and multiplication, respectively, such that

(i) R is an abelian group under addition;

(ii) R is a monoid under multiplication;

(iii) multiplication is distributive (on both sides) over addition.

A FIELD F is a commutative ring in which every non-zero element $a \in F$ has a multiplicative inverse $a^{-1} \in F$. In other words, a field F is an abelian group under addition and

F - {0} is an abelian group under multiplication.

A field of finite order is called a GALOIS FIELD.

A galois field of order q is denoted by GF(q), where q is a power of a prime number. An important characteristic of the Galois field is described in the following well-known

Theorem 2-1 :

Any two Galois fields of the same order are isomorphic to each other.

2-2. Shift Register Feedback Mechanism

A recurring sequence is also called a feedback shift register sequence, it can be obtained from an FSR. An n-stage shift register is a device consisting of n consecutive memory elements. The contents of each memory element shift to the next memory element down the line in time to the regular beat of a clock or to other timing device. A feedback term is computed as a logical function of the contents of n memory elements and fed back into the first memory element of the shift register. A block diagram of the shift register with the feedback mechanism is shown in Fig. 2-1, where M's are memory elements.

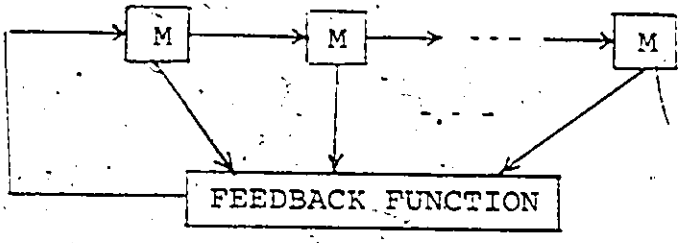


Fig. 2-1. Shift register feedback mechanism.

2-3. Linear Recurring Sequences

Suppose an initial k-tuple $(a_0\ a_1\ \ldots\ a_{k-1})$ is given, a sequence $\hat{a} = \{a_i\}_{i=0}^{\infty}$ over GF(q) satisfying the following recurring relation

$$c_0 a_i + c_1 a_{i-1} + \ldots + c_k a_{i-k} = 0 \quad (i = k, k+1, \ldots) \quad (2\text{-}1)$$

is called a linear recurring sequence.

In the binary case, $q = 2$ and $GF(q) = GF(2) = \{0, 1\}$, the operations $\oplus$ and $\cdot$ of GF(2) have the following truth tables,

$\oplus$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

A transformation T: $V \rightarrow V$ can be defined by Eq. (2-1) as follows,

$$T\,(a_i, a_{i+1}, \ldots, a_{i+k-1})^t = (a_{i+1}, a_{i+2}, \ldots, a_{i+k})^t,$$

where $a_{i+k} = -c_0^{-1}\,(c_1 a_{i+k-1} + c_2 a_{i+k-2} + \ldots + c_k a_i)$, $c_0 \neq 0$, and V is a k-dimensional vector space over GF(q). The transformation T has a matrix representation,

$$T = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 0 & 0 & 0 & \cdots & 1 \\ -c_0^{-1}c_k & -c_0^{-1}c_{k-1} & -c_0^{-1}c_{k-2} & \cdots & -c_0^{-1}c_1 \end{bmatrix}$$

Associated with Eq. (2-1) there is a polynomial $f(X) = c_0 + c_1 X + \ldots + c_k X^k$ called recurring polynomial or characteristic polynomial. In the binary case, corresponding to $f(X)$ there is a linear feedback function $f = c_1 x_1 \oplus c_2 x_2 \oplus \ldots \oplus c_k x_k$ where $c_i \in \{0, 1\}$ for $i = 1, 2, \ldots, k$ and $\oplus$ is a modulo-2-sum or exclusive-OR logic operation. A LFSR with f as a feedback function has a circuit block diagram shown in Fig. 2-2, where $c_i = 0$ and $c_i = 1$ represent respectively disconnected and connected circuit. A linear recurring sequence can be obtained from a LFSR.

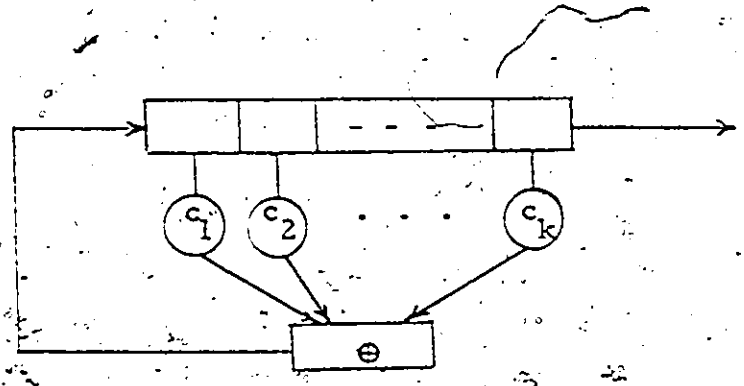


Fig. 2-2. A LFSR circuit block diagram

2-4. Nonlinear Recurring Sequences

The transformation T studied in Section 2-3 can be rewritten as follows,

$$T(a_i, a_{i+1}, \cdots, a_{i+k-1})^t = (a_{i+1}, a_{i+2}, \cdots, f(a_i, a_{i+1}, \cdots, a_{i+k-1}))^t,$$

where $f(a_i, a_{i+1}, \cdots, a_{i+k-1}) = c_0^{-1}(c_1 a_{i+k-1} + c_2 a_{i+k-2} + \cdots + c_k a_i)$

is a linear function of a_i, a_{i+1}, ... and a_{i+k-1}. Suppose $f(a_i, a_{i+1}, \cdots, a_{i+k-1})$ is a non-linear function of a_i, a_{i+1}, ... and a_{i+k-1}, a sequence can be similarly obtained if an initial k-tuple $(a_0\ a_1 \cdots a_{k-1})$ is given.

A nonlinear recurring sequence $\hat{a} = \{a_i\}_{i=0}^{\infty}$ with a given intial k-tuple $(a_0 a_1 \cdots a_{k-1})$ is a sequence satisfying the following nonlinear recurring relation,

$$a_{i+k} = f(a_i, a_{i+1}, \cdots, a_{i+k-1}) \quad i = 0, 1, \cdots$$

where $f(a_i, a_{i+1} \cdots, a_{i+k-1})$ is a nonlinear function of a_i, a_{i+1}, ... and a_{i+k-1}. In the binary case, f is a Boolean function. A nonlinear FSR circuit block diagram with f as a feedback function is shown in Fig. 2-3. A nonlinear recurring sequence can be obtained from a nonlinear FSR.

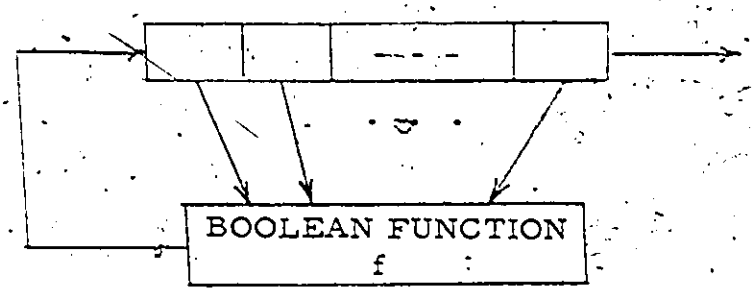


Fig. 2-3. A nonlinear FSR circuit block diagram.

2-5. Direct and Indirect Logic FSR

All the FSR's studied in Sections 2-3 and 2-4 are DIRECT LOGIC FSR's, in a sense that the output sequences are obtained directly from a stage of the FSR's. The output sequences can be obtained from any stage of the FSR's provided the initial states have been properly chosen. Usually the output sequences are obtained from the last stage of the FSR's. If the output sequences are obtained from an output function rather than from a stage of the FSR's, the FSR's are called INDIRECT LOGIC FSR's. A circuit block diagram of an indirect logic FSR is shown in Fig. 2-4, where f and g are respectively feedback and output function.

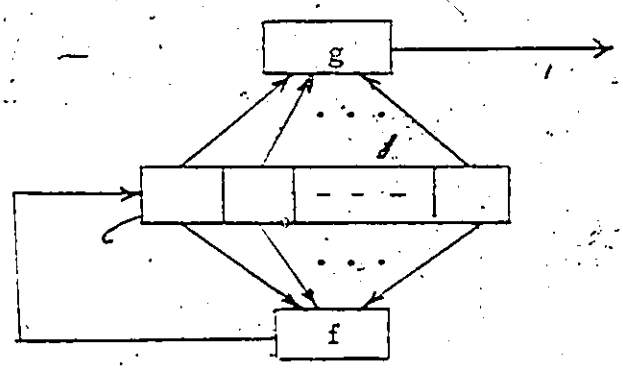
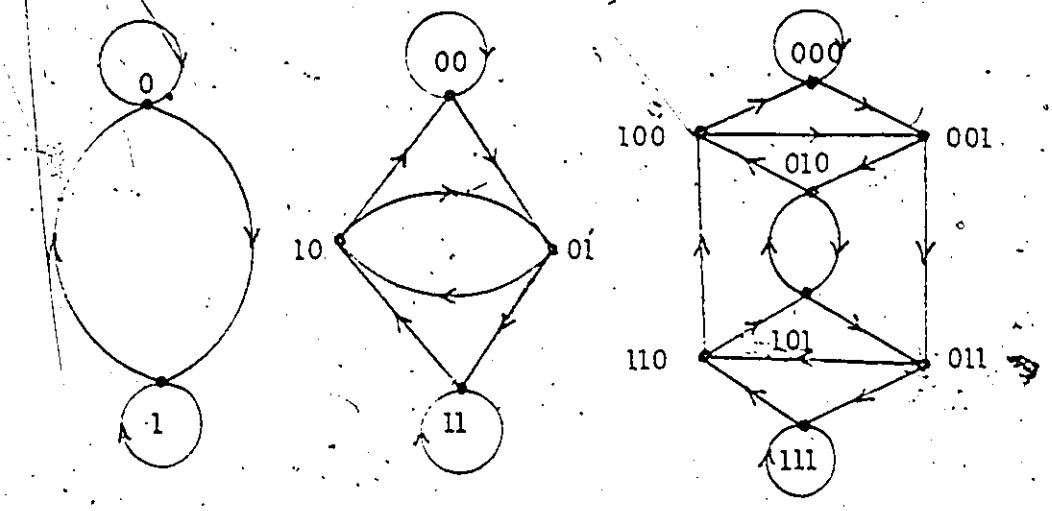


Fig. 2-4. An indirect logic FSR.

2-6. State Transition Diagrams and Good's Diagrams

The transformation $T: V \rightarrow V$ studied in Sections 2-3 and 2-4 defines a state transition diagram. Each vector $v \in V$ defines a state and a state transition from v_1 to v_2 is defined if and only if $T(v_1) = v_2$.

The well-known Good's diagram is defined as follows. Consider the set of binary k-tuples B^k, formed by the kth cartesian power of $B = \{0, 1\}$. In B^k a transition $x = (x_1, x_2, \ldots, x_k) \rightarrow x' = (x_1', x_2', \ldots, x_k')$ exists if and only if $(x_1', x_2', \ldots, x_k') = (x_2, x_3, \ldots, x_k, 0)$ or $(x_1', x_2', \ldots, x_k') = (x_2, x_3, \ldots, x_k, 1)$. For a given $x \in B^k$, there are exactly two successors of x, namely, $(x_2, x_3, \ldots, x_k, 0)$ and $(x_2, x_3, \ldots, x_k, 1)$. Similarly, there are exactly two predecessors of x, namely, $(0, x_1, \ldots, x_{k-1})$ and $(1, x_1, \ldots, x_{k-1})$. The kth order Good's diagram is a directed graph with 2^k vertices. The vertices x and x' are joined by a directed arc defined by the transition $x \rightarrow x'$. The Good's diagram of order 1, 2 and 3 is shown in Fig. 2-5.



k = 1 k = 2 k = 3

Fig. 2-5. The Good's diagram of order k = 1, 2 and 3.

2-7. Singular and Nonsingular Sequences

An FSR is called a NONSINGULAR FSR, if the state transition diagram of the FSR has only cycles without branches, i.e., each state of the state transition diagram has only one successor and one predecessor. A sequence generated by a nonsingular FSR is called a NONSINGULAR SEQUENCE. An FSR which is not nonsingular is a SINGULAR FSR. The state transition diagram of a singular FSR has at least one branch. A sequence generated by a singular FSR is called SINGULAR SEQUENCE.

The characteristic of the feedback function of a nonsingular FSR has been obtained by Zierler [49] in the linear case and by Yoeli [47] and Golomb [23] in the nonlinear case as described in the following theorems.

Theorem 2-2 : (Zierler)

A linear recurring sequence is nonsingular, if and only if the coefficient c_0 and c_k of the recurring polynomial $f(X) = c_0 + c_1 X + \ldots + c_k X^k$, which generates the sequence, are not equal to zero.

Theorem 2-3 : (Yoeli, Golomb)

A nonlinear recurring sequence is nonsingular, if and only if the feedback function of the generating FSR is of the form $f(x_1, x_2, \ldots, x_k) = x_1 + f_0(x_2, x_3, \ldots, x_k)$, where $f_0(x_2, x_3, \ldots, x_k) = f(0, x_2, \ldots, x_k)$ is an arbitrary Boolean function in the k-1 variables x_2, x_3, ... and x_k.

In this thesis, only nonsingular sequences will be considered. A sequence $\hat{a} = a_o a_1 \ldots, a_{n-1} \ldots$ is said to be of period n, if and only if $a_i = a_{i+kn}$ for all $i \geq 0$ and $k \geq 0$ and no integer $m < n$ exists such that $a_i = a_{i+km}$ for all $i \geq 0$ and $k \geq 0$.

CHAPTER 3

FEEDBACK SHIFT REGISTER REALIZATION FOR BINARY SEQUENCES OF FINITE PERIODS

In this chapter, the feedback shift register realization for the generation of specified binary sequences of finite periods is studied. A LFSR synthesis method given by Massey [35] will be discussed, which will appear to be useful in the development of the next chapter. A synthesis method based on FSR's with input sequences will be given. While Reed and Turn [41] has given a synthesis method to find maximum length sequences of period $m2^n$ using the generalized (m, n) - FSR's, we give a synthesis method to find FSR's with input sequences to generate some specified sequences of finite periods. Instead of using multiple feedback functions which have been devised by Reed and Turn, a single feedback function will be used. Conventional FSR synthesis methods using both direct logic FSR's and indirect logic FSR's also will be discussed.

3-1. Minimum Length LFSR Realization for Generation of Specified Sequences of Finite Periods

An algorithm to find a minimum length LFSR realization for a sequence of finite length has been achieved by Massey [35]. Massey's method can also be applied to the minimum length LFSR realization for sequences of finite periods by

simply considering two periods of the sequences.

In the following, we will discuss Massey's method. Massey's method starts with finite length sequences. For a given finite length sequence $a_0\ a_1 \cdots a_{N-1}$, it follows from Eq. (2-1) that the L-stage LFSR generates the sequence, if and only if

$$\sum_{i=0}^{L} c_i a_{j-i} = 0, \quad j = L, L+1, \ldots, N-1. \tag{3-1}$$

Let $c_0 = 1$, the recurring polynomial of the LFSR is

$$f(X) = 1 + c_1 X + \cdots + c_L X^L \tag{3-2}$$

By way of convention, we take $f(X) = 1$ for the LFSR of length $L = 0$.

Now, let $\hat{a} = a_0 a_1 \cdots a_{n-1} \cdots$ be an infinite sequence whose first N digits are $a_0 a_1 \cdots a_{N-1}$. Let L_N be the minimum length of the LFSR's that generate $a_0 a_1 \cdots a_{N-1}$. An FSR is said to generate a finite sequence $a_0 a_1 \cdots a_{N-1}$, if this sequence coincides with the first N output digists of the FSR when the memory elements of the FSR are loaded with appropriate initial contents. For $L \geq N$, an FSR can always generate a given sequence by loading the given sequence into the FSR initially. Moreover L_N must be monotonically nondecreasing with increasing N. By way of convention, we shall say that the all-zero sequence is generated by the LFSR with $L = 0$.

For a given finite length sequence $a_0 a_1 \cdots a_{N-1}$ of $\hat{a}$,

let

$$f^{(N)}(X) = 1 + c_1^{(N)}(X) + \ldots + c_{L_N}^{(N)} X^{L_N}$$

be the recurring polynomial of the LFSR with the minimal length L_N which generates the sequence. From Eq. (3-1), we have

$$a_j + \sum_{i=0}^{L_N} c_i^{(N)} a_{j-1} = \begin{cases} 0 & j = L_N, \ldots, N-1 \\ d_N & j = N \end{cases},$$

where d_N, which we call the NEXT DISCREPANCY, is the difference between a_N and the $(N+1)$ th digit generated by the minimal-length LFSR obtained for generating the finite length sequence $a_0 a_1 \cdots a_{N-1}$. If $d_N = 0$, then this LFSR also generates the finite length sequence $a_0 a_1 \cdots a_{N-1} a_N$ of $\hat{a}$ so that $L_{N+1} = L_N$ and we have $f^{(N+1)}(X) = f^{(N)}(X)$. If $d_N \neq 0$, a new LFSR must be found to generate the first $N+1$ digits of $\hat{a}$. In the latter case, let M be the sequence length before the last length change in the minimal-length registers, i.e.,

$$L_M < L_N ,$$

$$L_{M+1} = L_N .$$

It has been proved by Massey that the new recurring polynomial $f^{(N+1)}(X)$ to generate the finite length sequence $a_0 a_1 \cdots a_{N-1} a_N$ and its length L_{N+1} can be found by the following equations:

$$L_{N+1} = \text{Max} \{ L_N, N + 1 - L_N \} \qquad (3\text{-}3)$$

$$f^{(N+1)}(X) = f^{(N)}(X) - d_N d_M^{-1} X^{N-M} f^{(M)}(X) \qquad (3-4)$$

Now, the problem of finding a minimal-length LFSR to generate a finite length sequence $a_0 a_1 \cdots a_{n-1}$ is equivalent to finding the recurring polynomial $f^{(n)}(X)$. Use Eq 's (3-3) and (3-4) iteratively, we can find a minimal-length LFSR to generate a finite length sequence.

For a sequence $a_0 a_1 \cdots a_{n-1} \cdots$ of period n, the same method can be applied by considering two periods of the sequence, the finite length sequence $a_0 a_1 \cdots a_{n-1} a_0 a_1 \cdots a_{n-1}$. However, if a LFSR of length L generates the finite length sequence $a_0 a_1 \cdots a_{n-1} a_0 a_1 \cdots a_{L-1}$, then it automatically generates the whole sequence of period n. Hence, Massey's LFSR synthesis algorithm to find a minimal length LFSR for generating a sequence of finite period can be described as follows.

1) $1 \rightarrow f(X)$, $1 \rightarrow g(X)$, $1 \rightarrow y$,
   $0 \rightarrow L$, $1 \rightarrow b$, $0 \rightarrow N$.

2) If $N = n + L$, stop; otherwise compute
$$d = a_N + \sum_{i=1}^{L} c_i a_{N-i}.$$

3) If $d = 0$, then $y + 1 \rightarrow y$, and go to 6).

4) If $d \neq 0$, and $2L > N$, then
$$f(X) - db^{-1} X^y g(X) \rightarrow f(X),$$
$y + 1 \rightarrow y$ and go to 6).

5) If $d \neq 0$ and $L < N$, then

$f(X) \rightarrow k(X)$ (temporary storage of $f(X)$),

$f(X) - db^{-1}X^{y}g(X) \rightarrow f(X)$,

$N + 1 - L \rightarrow L$,

$h(X) \rightarrow g(X)$,

$d \rightarrow b$,

$1 \rightarrow y$.

6) $N + 1 \rightarrow N$ and return to 2).

In the above algorithm, the operation $\longrightarrow$ replaces the contents of right hand side by the contents of the left hand side, and $f(X)$, $g(X)$, y and b represent $f^{(N)}(X)$, $f^{(M)}(X)$, $N-M$ and d_M of Eq. (3-4), respectively.

Example 3-1 :

Consider the following sequence of period 7,

$\hat{a}$ = 1110100 ...

Apply the algorithm given above, we can obtain a table shown in Table 3-1, from which we can obtain the recurring polynomial $f(X) = 1 + X + X^3$ which generates the sequence. The corresponding minimal-length LFSR which generates the sequence has a circuit block diagram shown in Fig. 3-1.

Table 3-1.

Application of the LFSR synthesis algorithm to the sequence of Example 3-1.

N	L	f (X)	Y	g (X)	b	a_N	d
0	0	1	1	1	1	1	1
1	1	$1 + X$	1	1	1	1	0
2	1	$1 + X$	2	1	1	1	0
3	1	$1 + X$	3	1	1	0	1
4	3	$1 + X + X^3$	1	$1 + X$	1	1	0
5	3	$1 + X + X^3$	1	$1 + X$	1	0	0
6	3	$1 + X + X^3$	1	$1 + X$	1	0	0
7	3	$1 + X + X^3$	1	$1 + X$	1	1	0
8	3	$1 + X + X^3$	1	$1 + X$	1	1	0
9	3	$1 + X + X^3$	1	$1 + X$	1	1	0
10	3	$1 + X + X^3$	1	$1 + X$	1	0	0

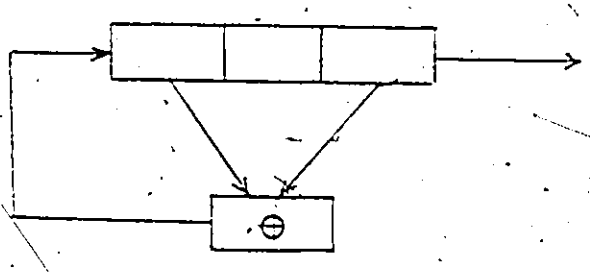


Fig. 3-1. The minimal-length LFSR which generates the sequence of Example 3-1.

3-2. Nonlinear FSR Realization for Generation of Specified Sequences of Finite Periods

The nonlinear FSR synthesis problem has been studied, among others, by Magleby [33], Cohn and Even [10], Baumert [5] and Reed and Turn [41]. Generation of specified sequences by using either direct logic or indirect logic FSR has been studied by Baumert [5]. A multiple feedback function FSR with input sequence for sequence generation has been introduced by Reed and Turn [41].

In this section, the conventional FSR synthesis method will be discussed. A synthesis method based on FSR's with input sequences will be given.

3-2-1. Conventional FSR synthesis methods

In the following, we will discuss the conventional FSR realization for generation of specified sequences of finite periods. Both direct logic FSR and indirect logic FSR realization will be discussed. First, we need the following

Definition 3-1:

The H_k-NUMBER is defined as follows. For a subsequence $a_0a_1 \cdots a_{n+k-2}$ of length $n+k-1$ of a given sequence $a_0a_1 \cdots a_{n-1}$ of period n, a function H_k is defined by a mapping of the set of distinct subsequences of length k of the sequence $a_0a_1 \cdots a_{n+k-2}$ to the positive integers such that $H_k(a_0a_1 \cdots a_{k-1}) = 1$; $H_k(a_1a_2 \cdots a_k) = 2$, if $a_0a_1 \cdots a_{k-1} \neq a_1a_2 \cdots a_k$, otherwise, $H_k(a_1a_2 \cdots a_k) = 1$; $\cdots$; $H_k(a_{p-1}a_p \cdots a_{p+k-2}) = p$, if $a_0a_1 \cdots a_{k-1} \neq a_1a_2 \cdots a_k \neq \cdots \neq a_{p-1}a_p \cdots a_{p+k-2}$;

otherwise, $H_k(a_{p-1}a_p \cdots a_{p+k-2}) = i$ for some i, $1 \le i \le p-1$;
where $p \le n-1$ is a positive integer.

Definition 3-2 :

A sequence $\hat{a}$ of period n is called an INCOMPLETE H_k-SEQUENCE, if in a subsequence $a_0a_1 \cdots a_{n+k-2}$ of length $n+k-1$ of $\hat{a}$, there exist two distinct integers i and j such that

$$H_k(a_ia_{i+1} \cdots a_{i+k-1}) = H_k(a_ja_{j+1} \cdots a_{j+k-1}).$$

Definition 3-3 :

A sequence $\hat{a}$ of period n is called an EXACT H_k-SEQUENCE, if it is not an incomplete H_k-sequence.

From the above definitions, we have the following

Theorem 3-1 :

The following statements hold for a binary sequence of period n.

(i) The largest H_k-number is less than or equal to n.

(ii) If the sequence is an exact H_k-sequence, then any subsequence with the same consecutive symbol has length $\ell \le k$.

(iii) If the sequence is an exact H_k-sequence, then k is greater than or equal to $\lceil \log_2 n \rceil$, where $\lceil x \rceil$ is the least integer greater than or equal to x.

Proof:

(i) and (ii) can be seen easily. We prove (iii) as follows.

Since the sequence is an exact H_k-sequence, there are n distinct k-tuples. But there are at most 2^k binary k-tuples, which implies $2^k \geq n$, i.e., $k \geq \lceil \log_2 n \rceil$.

Q. E. D.

A necessary and sufficient condition for the direct logic FSR realization of a sequence is given by the following.

Theorem 3-2 :

A binary sequence of period n can be generated by a direct logic FSR of length k, if and only if it is an exact H_k-sequence.

Proof:

The sufficiency part will be proved by showing the existence of a direct logic k-FSR which generates an arbitrary exact H_k-sequence.

Let $a_0 a_1 \ldots a_{n-1} \ldots$ be an arbitrary exact H_k-sequence of period n. We can build a state transition diagram from the sequence as shown in Fig. 3-2, where each state is represented by the k consecutive digits of the sequence. The state transition diagram of Fig. 3-2 is sligthly different from the conventional state transition diagram, only those states related to the sequence

are specified.

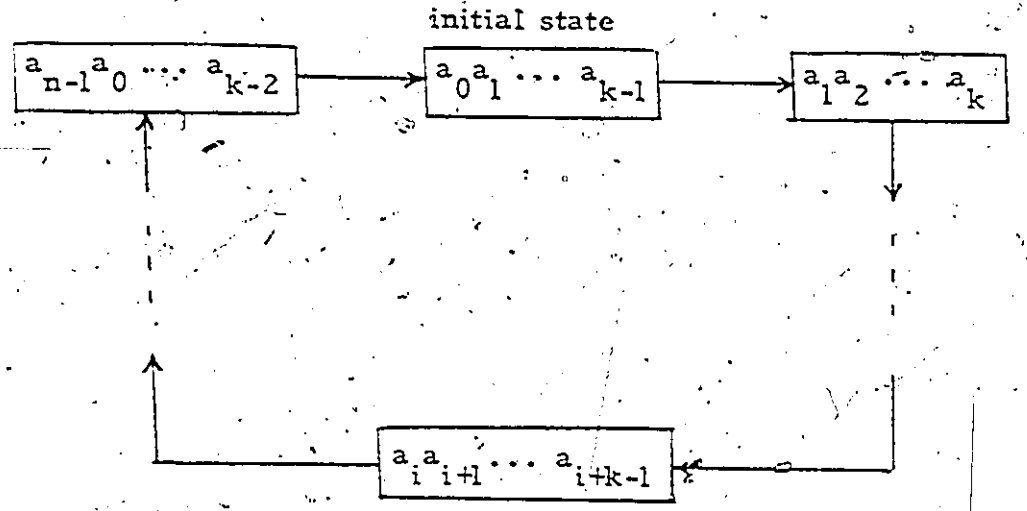


Fig. 3-2. A state transition diagram built from a sequence $a_0a_1 \cdots a_{n-1} \cdots$ of period n.

We can find a feedback function f by setting up a truth table as shown in Table 3-2. Since the sequence is an exact H_k-sequence, all the rows in Table 3-2 are distinct, no contradiction such as $a_ia_{i+1} \cdots a_{i+k-1} = a_ja_{j+1} \cdots a_{j+k-1}$ and $a_{i+k} \neq a_{j+k}$ for distinct integers i and j will occur. Hence the feedback function f exists. A direct logic k-FSR with circuit block diagram shown in Fig. 3-3 will generate the sequence.

Table 3-2.

Truth table of the

feedback function f of Theorem 3-2.

y_0	y_1	. . .	y_{k-1}	f
a_0	a_1	. . .	a_{k-1}	a_k
a_1	a_2	. . .	a_k	a_{k+1}
.	.	. . .	.	.
.	.	. . .	.	.
.	.	. . .	.	.
a_{n-1}	a_0	. . .	a_{k-2}	a_{k-1}

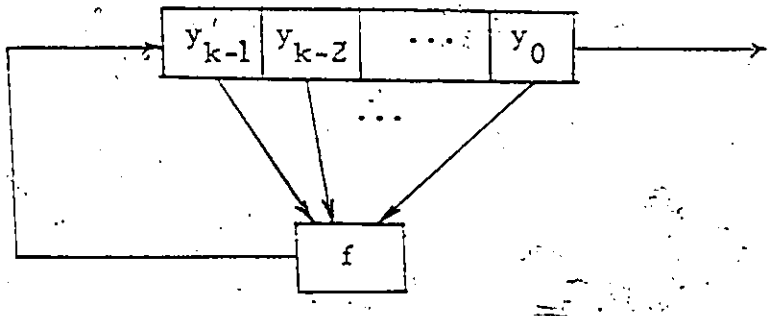


Fig. 3-3. A direct logic k-FSR of Theorem 3-2.

The necessity part is proved as follows. If a sequence $a_0a_1 \ldots a_{n-1} \ldots$ of period n is generated by a direct logic k-FSR, all the n k-tuples $a_0a_1 \ldots a_{k-1}$, $a_1a_2 \ldots a_k$, ... and $a_{n-1}a_0 \ldots a_{k-2}$ which represent the states of the state transition diagram of the FSR are

distinct. If any two k-tuples of the above n k-tuples are equal, then the successors of the two k-tuples must be equal also, and the successors of these two equal successors must be equal again and so on. Then the sequence generated will no longer be of period n and a contradiciton is obtained. Hence the sequence is an exact H_k-sequence.

Q. E. D.

From Theorem 3-2, we have the following

Corollary 3-2-1 :

A binary sequence of finite period which can be generated by a direct logic k-FSR can also be generated by a direct logic k'-FSR for k' > k.

Proof:

We show that if a sequence is an exact H_k-sequence, then it is also an exact $H_{k'}$-sequence for k' > k.

Pick up any two subsequences of length k' from any subsequence of length n+k -1 of the given sequence of period n, say $a_i a_{i+1} \cdots a_{i+k'-1}$ and $a_j a_{j+1} \cdots a_{j+k'-1}$ for $i \neq j$, we have to show that

$$H_{k'}(a_i a_{i+1} \cdots a_{i+k'-1}) \neq H_{k'}(a_j a_{j+1} \cdots a_{j+k'-1}).$$

Assume contrary, i.e.,

$$H_{k'}(a_i a_{i+1} \cdots a_{i+k'-1}) = H_{k'}(a_j a_{j+1} \cdots a_{j+k'-1}),$$

then $H_k(a_i a_{i+1} \cdots a_{i+k-1}) = H_k(a_j a_{j+1} \cdots a_{j+k-1})$,

which contradicts the hypothesis that the sequence is an exact H_k-sequence.

Q. E. D.

Golomb [23] and Baumert [5] have shown that there exists a direct logic k-FSR which can generate at least one sequence of period n, $1 \leq n < 2^k$. In other words, there exists an exact H_k-sequence of period n for every $n \leq 2^k$. We quote the result without proof in the following

Theorem 3-3 : (Golomb, Baumert)

All period of 1 to 2^k can be obtained from a direct logic k-FSR.

From Theorem 3-3, we can prove the following

Theorem 3-4 :

Any binary sequence of period n can be generated by an indirect logic FSR of length $\lceil \log_2 n \rceil$.

Proof:

We prove the theorem by showing the exsitence of an indirect logic FSR of length $\lceil \log_2 n \rceil$ which generates an arbitrary sequence of period n.

From Theorem 3-3, let $k = \lceil \log_2 n \rceil$, there exists an exact H_k-sequence $b_0 b_1 \ldots b_{n-1} \ldots$ of period n. Let $a_0 a_1 \ldots a_{n-1} \ldots$ be an arbitrary sequence of period n. Then the indirect logic k-FSR with circuit block diagram

shown in Fig. 3-4 will generate the sequence $a_0 a_1 \ldots a_{n-1} \ldots$, where f and g are respectively the feedback and output function with truth tables shown in Table 3-3.

Table 3-3.

Truth tables of functions f and g of Theorem 3-4.

y_0	y_1	. . .	y_{k-1}	f	g
b_0	b_1	. . .	b_{k-1}	b_k	a_0
b_1	b_2	. . .	b_k	b_{k+1}	a_1
.	.	. . .	.	.	.
.	.	. . .	.	.	.
.	.	. . .	.	.	.
b_{n-1}	b_0	. . .	b_{k-2}	b_{k-1}	a_{n-1}

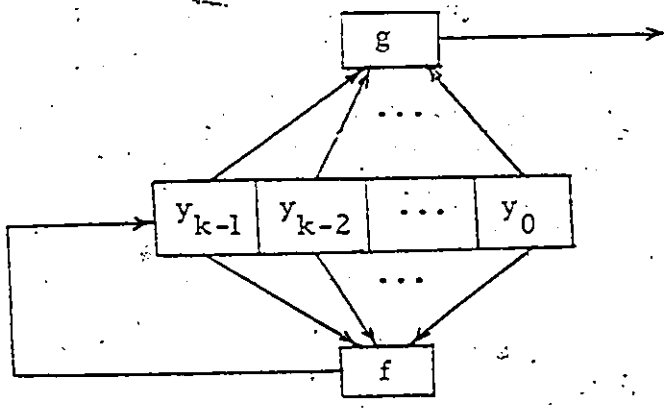


Fig. 3-4. An indirect logic k-FSR of Theorem 3-4.

The proofs of both Theorem 3-2 and Theorem 3-4 are constructive, they also suggest a synthesis method to find an FSR for generating a given specified sequence.

An exact H_k-sequence of period n can also be obtained from the Good's diagram of order k studied in Section 2-6 by tracing the vertices of the diagram to find a cycle with exact n vertices, and picking up the ith digit from each vertix through the cycle found.

Example 3-2 :

Consider a PN sequence 11011100010 ... of period 11.

Since $k = \lceil \log_2 11 \rceil = 4$, we have to find an exact H_4-sequence of period 11. Note that the PN sequence is not an exact H_4-sequence. Now, the sequence 11110011010 ... of period 11 is an exact H_4-sequence. The indirect logic 4-FSR with schematic circuit diagram shown in Fig. 3-5 will generate the PN sequence. The feedback and output function $f = y_0\bar{y}_1 + \bar{y}_0 y_1 + \bar{y}_2\bar{y}_3$ and $g = y_0 y_1 y_2 + y_1 y_2 \bar{y}_3 + \bar{y}_1 y_3$ can be obtained from the truth tables shown in Table 3-4.

Table 3-4.

Truth tables of functions f and g of Example 3-2.

y_0	y_1	y_2	y_3	f	g
1	1	1	1	0	1
1	1	1	0	0	1
1	1	0	0	1	0
1	0	0	1	1	1
0	0	1	1	0	1
0	1	1	0	1	1
1	1	0	1	0	0
1	0	1	0	1	0
0	1	0	1	1	0
1	0	1	1	1	1
0	1	1	1	1	0

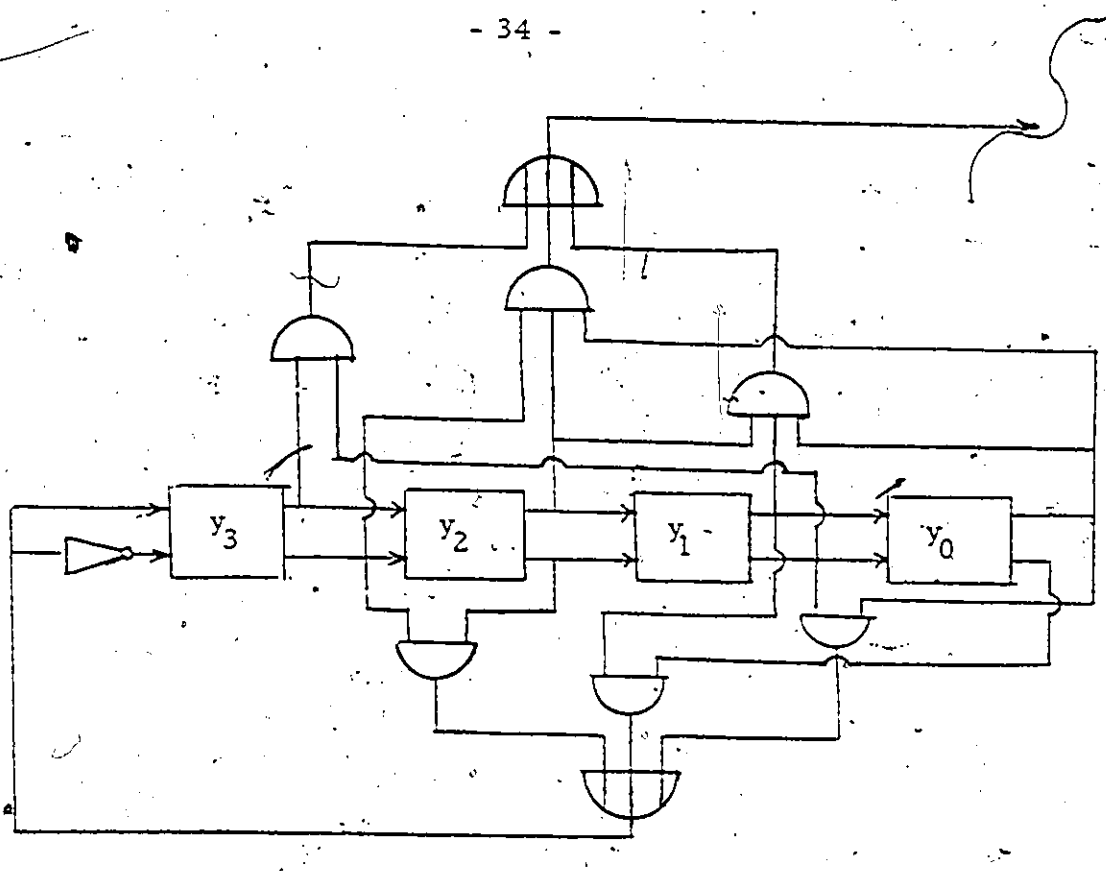


Fig. 3-5. An indirect logic 4-FSR schematic circuit diagram of Example 3-2.

3-2-2. A synthesis method based on the FSR with input sequence.

To generate sequences with a shorter FSR. A multiple feedback function FSR has been devised by Reed and Turn [41] as shown in Fig. 3-6, where f_1, f_2, ... and f_m are multiple feedback functions and an additional input sequence is required to control the operation of the multiple feedback functions. Synthesis method to find a maximum length sequence of period $m2^n$ has been obtained by Reed and Turn, while synthesis method to find an FSR with input sequence to generate a specified sequence of

finite period is still lacking.

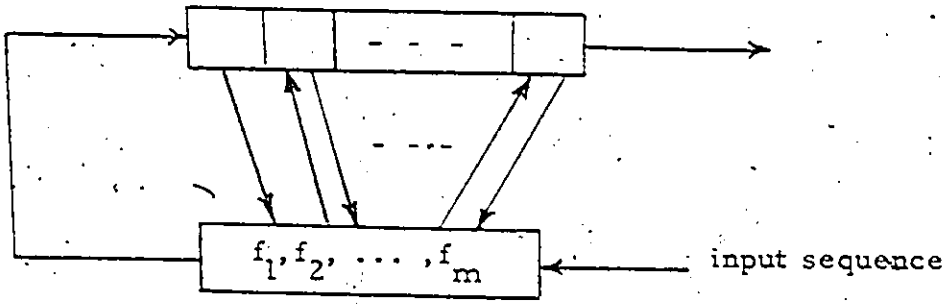


Fig. 3-6. A multiple feedback function FSR with input sequence of Reed and Turn.

Instead of using multiple feedback functions, we modify the model by using a single feedback function. A synthesis method based on the modified model will be given. Specifically, we will prove Theorem 3-5 to be given later. Before we prove the theorem, we need the following preliminary discussion.

Let $a_0 a_1 \ldots a_{n-1} \ldots$ be an arbitrary sequence of period n and let $k' = \lceil \log_2 n \rceil$ and $k < k'$. We can build a state transition diagram from the sequence by picking up k consecutive symbols each time as a state and going through one period of the sequence as shown in Fig. 3-7. Assign $m \geq 2$ input symbols i_0, i_1, ... and i_{m-1} to the state transitions of the states in the state transition diagram as also shown in Fig. 3-7, where $I_i \in \{ i_0, i_1, \ldots, i_{m-1} \}$ for $i = 0, 1, \ldots, n-1$.

The input symbol assignment obeys the following rules. If the states $a_i a_{i+1} \cdots a_{i+k-1} = a_j a_{j+1} \cdots a_{j+k-1}$ and $a_{i+k} = a_{j+k}$, then the same input symbol may be assigned to the state transitions of states $a_i a_{i+1} \cdots a_{i+k-1}$ and $a_j a_{j+1} \cdots a_{j+k-1}$. On the other hand, if $a_i a_{i+1} \cdots a_{i+k-1} = a_j a_{j+1} \cdots a_{j+k-1}$ and $a_{i+k} \neq a_{j+k}$, different input symbols should be assigned to the state transitions of states $a_i a_{i+1} \cdots a_{i+k-1}$ and $a_j a_{j+1} \cdots a_{j+k-1}$.

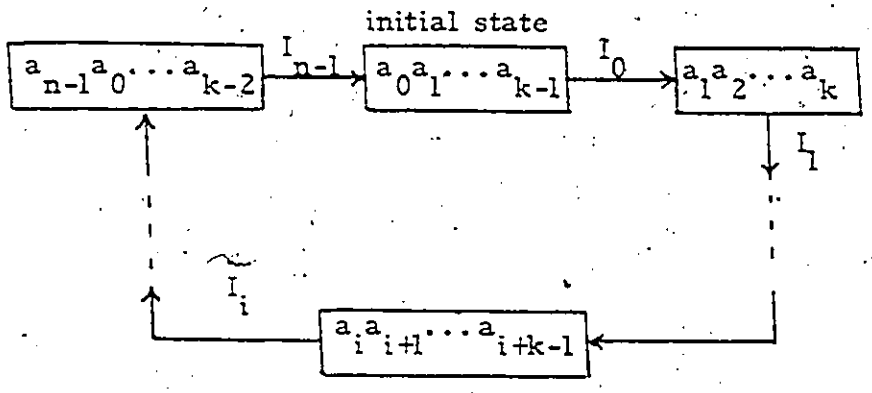


Fig. 3-7. A state transition diagram with assigned input symbols.

The state transition diagram shown in Fig. 3-7 is sligthly different from the conventional one. The same state can appear more than once in the cycle and the state transitions are labelled by the input symbols.

From the state transition diagram, a truth table of the feedback function f can be set up easily as shown in

Table 3-5. Let $r = \lceil \log_2 m \rceil$, and code the m input symbols i_0, i_1, ... and i_{m-1} by m binary r-tuples, then the feedback function f can easily be obtained from Table 3-5. A direct logic k-FSR with input sequence $I_0 I_1 \cdots I_{n-1} \cdots$ of period n can certainly be realized. The realized circuit block diagram is shown in Fig. 3-8.

Table 3-5.

Truth table of the

feedback function f.

I	y_0	y_1	. . .	y_{k-1}	f
I_0	a_0	a_1	. . .	a_{k-1}	a_k
I_1	a_1	a_2	. . .	a_k	a_{k+1}
.	.	.	. . .	.	.
.	.	.	. . .	.	.
.	.	.	. . .	.	.
I_{n-1}	a_{n-1}	a_0	. . .	a_{k-2}	a_{k-1}

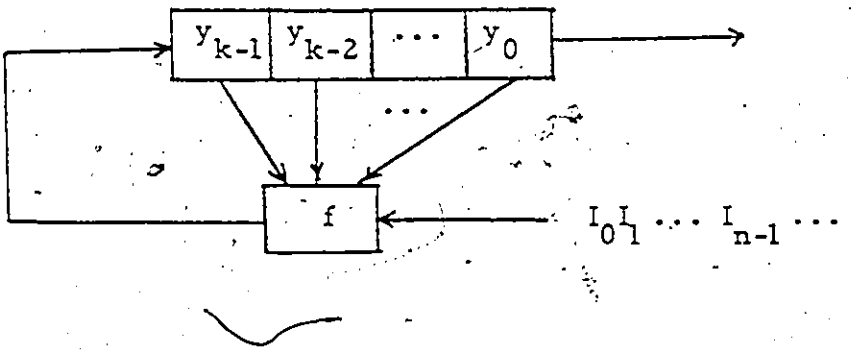


Fig. 3-8. A direct logic k-FSR with input sequence.

However, such a realization is trivial, in a sense that the complexity of the output sequence is passed on to the input sequence without simplification at all. Furthermore, the input sequence may not be a binary sequence, although it can be coded into binary sequences.

If $n = mp$, where m and p are integers, then it may be possible to use an input sequence of period p instead of n. To test whether an input sequence of period p will work or not, we can form an array with $(p + k)$ rows and m columns as follows:

$$
\begin{array}{cc}
 & \begin{array}{ccccc} r_1 & & & & \end{array} \\
\begin{array}{c} r_0 \\ \\ \\ \\ \\ \\ \\ \\ \\ r_{p-1} \\ \\ \\ \end{array} &
\left[\begin{array}{ccccc}
a_0 & a_p & \cdot & \cdot & a_{(m-1)p} \\
a_1 & a_{p+1} & \cdot & \cdot & a_{(m-1)p+1} \\
\cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot \\
a_k & a_{p+k} & \cdot & \cdot & a_{(m-1)p+k} \\
a_{k+1} & a_{p+k+1} & \cdot & \cdot & a_{(m-1)p+k+1} \\
\cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot \\
a_{p-1} & a_{2p-1} & \cdot & \cdot & a_{mp-1} \\
a_p & a_{2p} & \cdot & \cdot & a_0 \\
\cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot \\
a_{p+k-1} & a_{2p+k-1} & \cdot & \cdot & a_{k-1}
\end{array}\right]
\end{array}
$$

Denote the set of consecutive k + 1 rows of the array by row blocks r_0, r_1, ... and r_{p-1} as shown in the array. In each row block r_i, $0 \leq i \leq p-1$, the first k symbols of any column represent a state of the state transition diagram. If an input sequence, which has been assigned to the state transitions of the state transition diagram, has a period p, all the state transitions of the states in a row block r_i will be assigned the same input symbol. Hence to obey the assignment rules, within a row block r_i, any two columns which have the same first k symbols must have the same (k + 1) th symbol. Similarly, in any two row blocks r_i and r_j, $0 \leq i, j \leq p-1$, which have been assigned the same input symbol, any two columns, one in the row block r_i and the other in the row block r_j, which have the same first k symbols must have the same (k + 1) th symbol. This condition can be stated as the following

Theorem 3-5 :

A sequence $a_0 a_1 \ldots a_{n-1}$ of period $n = mp$ can be generated by a direct logic k-FSR with input sequence $I_0 I_1 \ldots I_{p-1} \ldots$ of period p, if and only if in any row block r_i of the above array, $0 \leq i \leq p-1$, any two columns which coincide in the first k symbols also coincide in the last symbol.

From Theorem 3-5, we have the following

Corollary 3-5-1:

In the array of Theorem 3-5, any two distinct row blocks r_i and r_j, $0 \leq i, j \leq p-1$, can be assigned the same input symbol if and only if any two columns, one in the row block r_i and the other in the row block r_j, which coincide in the first k symbols also coincide in the last symbol.

An exhaustive search can be made for all arrays with respect to different combinations of the factorization of n into two positive integers.

Example 3-3 :

Consider the sequence 1001101111 0 0111 0 ... of period 16.

Picking up 3 consecutive symbols at each time and going through one period of the sequence, we can build a state transition diagram as shown in Fig. 3-9.

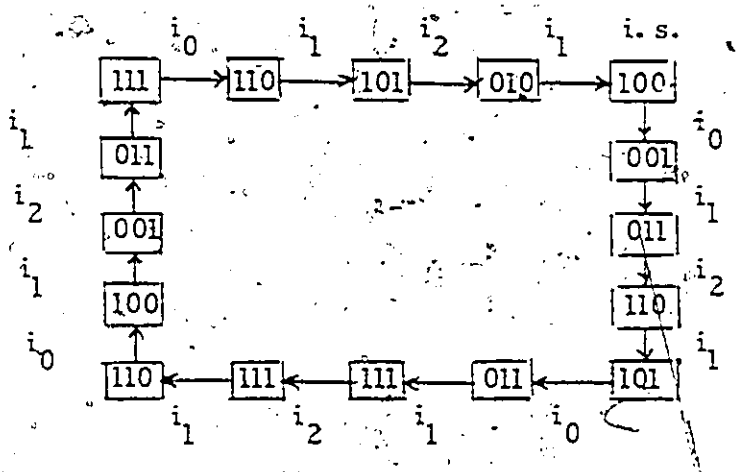


Fig. 3-9. State transition diagram with assigned input symbols of Example 3-3.

From the given sequence, we can form an array as follows:

$$
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
r_0\ \ 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \quad r_1 \\
1 & 1 & 0 & 0 \\
r_2\ \ 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 \quad r_3 \\
1 & 0 & 1 & 0
\end{array}
$$

Since the row blocks r_i, $0 \leq i \leq 3$, of the array satisfy the conditions of Theorem 3-5 and the row blocks r_1 and r_3 satisfy Corollary 3-5-1, we can assign input symbols i_0, i_1 and i_2 to the state transitions of the state transition diagram as shown in Fig. 3-9, and obtain an input sequence $i_0 i_1 i_2 i_1 \ldots$ of period 4.

Assign binary 2-tuples 00, 01 and 10 to the input symbols i_0, i_1 and i_2, respectively. Set up the truth table of the feedback function as shown in Table 3-6. A simplified function

$$f = \bar{x}_0 \bar{y}_1 + \bar{y}_0 \bar{y}_1 + x_0 y_0 y_1 + x_1 y_0 \bar{y}_2 + x_1 \bar{y}_0 y_2$$

can be obtained from Table 3-6. The direct logic 3-FSR with schematic circuit diagram shown in Fig. 3-10 will generate the sequence, where x_0 and x_1 are input terminal variables with input sequences of periods 4 and 2,

Table 3-6.

Truth table of the feedback function of Example 3-6.

x_0	x_1	y_0	y_1	y_2	f
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	1	0
0	1	1	0	0	1
0	1	1	1	0	1
0	1	1	1	1	0
0	1	0	0	1	1
0	1	0	1	1	1
0	1	0	1	0	0
1	0	1	0	1	0
1	0	1	1	1	1
1	0	0	0	1	1
1	0	0	1	1	0

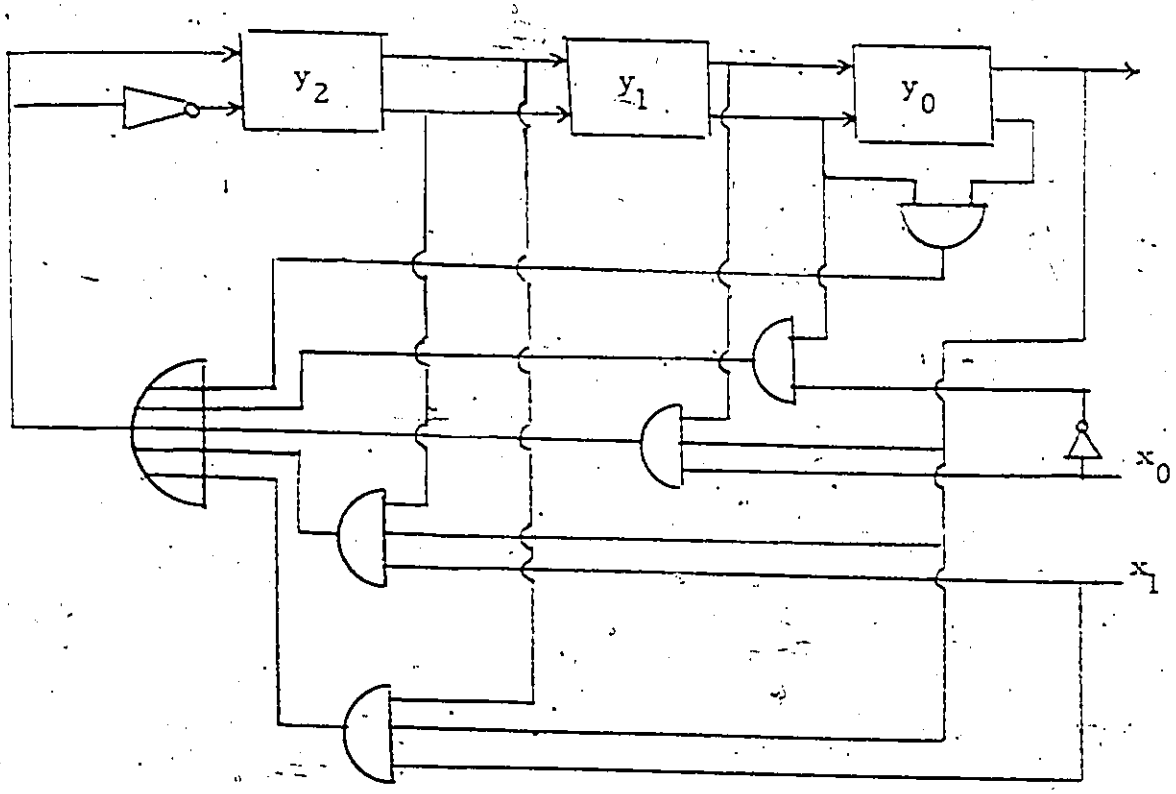
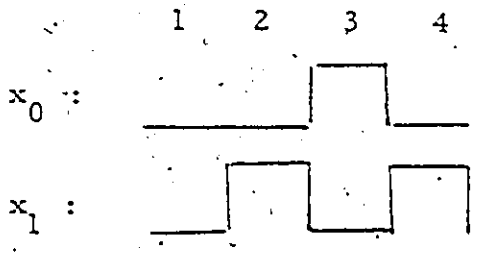


Fig. 3-10. A direct logic 3-FSR with input sequence schematic circuit diagram of Example 3-3.

respectively,



Next, we give a synthesis method based on an indirect logic FSR with input sequence fed into the output function. The method is similar to the previous one in concept. Before the method is discussed, we need the following terms and lemma.

Consider two sequences $S = \{ s_i \}_{i=0}^{\infty}$ and $I = \{ i_j \}_{i=0}^{\infty}$ of period m and p, respectively. Assume that $s_i \neq s_j$ for all $i \neq j$, $0 \leq i, j \leq m-1$, and $i_\ell \neq i_k$, for all $\ell \neq k$; $0 \leq \ell, k \leq p-1$. We can construct a sequence of order pairs as follows,

$$\binom{S}{I} = \left\{ \binom{s_j}{i_j} \right\}_{j=0}^{\infty} .$$

The following lemma which characterized the sequence $\binom{S}{I}$ can be proved easily.

Lemma 3-1:

Let S, I and $\binom{S}{I}$ be the sequences defined above. Then the period of $\binom{S}{I}$ is equal to LCM $\{m, p\}$, and $\binom{s_j}{i_j} \neq \binom{s_k}{i_k}$ for all $j \neq k$, $0 \leq j, k \leq$ LCM $\{m, p\} - 1$.

Now we can prove the following

Theorem 3-6 :

If m and p are relatively prime integers, then any binary sequence of period $n = mp$ can be generated by an indirect logic k-ESR with input sequence of period p fed into the output function, where $k = \lceil \log_2 m \rceil$.

Proof:

Let $a_0a_1 \ldots a_{n-1} \ldots$ be an arbitrary sequence of period $n = mp$, we are going to show that there exists an indirect logic k-FSR with input sequence of period p which generates the sequence. Let $b_0b_1 \ldots b_{m-1} \ldots$ be an exact H_k-sequence of period m, and let $I_0I_1 \ldots I_{p-1} \ldots$ be a sequence of period p such that all the digits within one period of the sequence are distinct, i.e., $I_i \neq I_j$ for all $i \neq j$, $0 \leq i,j \leq p-1$. Then the indirect logic k-FSR with input sequence $I_0I_1 \ldots I_{p-1} \ldots$ of period p fed into the output function shown in Fig. 3-11 will generate the sequence $a_0a_1 \ldots a_{n-1} \ldots$, where f and g are respectively the feedback and output function with truth tables shown in Table 3-7 and Table 3-8, respectively. The existence of the output function g is guaranteed by Lemma 3-1.

Q. E. D.

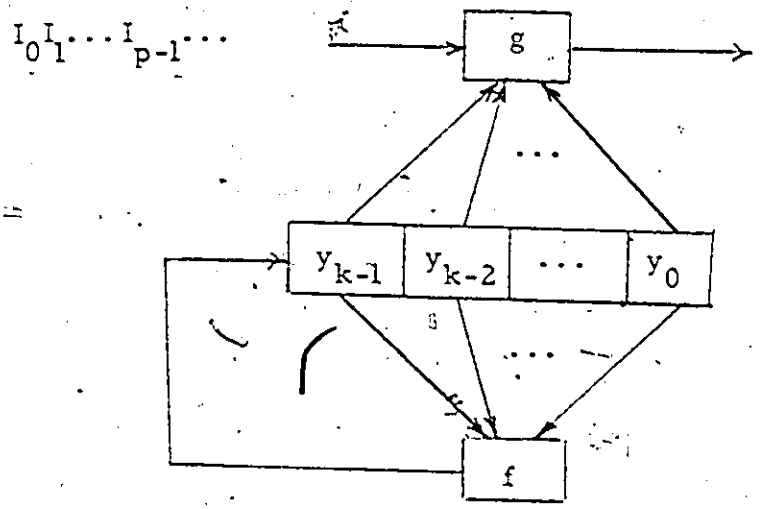


Fig. 3-11. An indirect logic k-FSR with input sequence fed into the output function.

Table 3-7.

Truth table of the feedback function f of Theorem 3-6.

y_0	y_1	. . .	y_{k-1}	f
b_0	b_1	. . .	b_{k-1}	b_k
b_1	b_2	. . .	b_k	b_{k+1}
.	.	. . .	.	.
.	.	. . .	.	.
.	.	. . .	.	.
b_{m-1}	b_0	. . .	b_{k-2}	b_{k-1}

Table 3-8.

Truth table of the output function g of Theorem 3-6.

I	y_0	y_1	. . .	y_{k-1}	g
I_0	b_0	b_1	. . .	b_{k-1}	a_0
I_1	b_1	b_2	. . .	b_k	a_1
.	.	.	. . .	.	.
.	.	.	. . .	.	.
.	.	.	. . .	.	.
I_{p-1}	$b_{(p-1)}$	$b_{(p)}$	. . .	$b_{(p+k-2)}$	a_{p-1}
.	.	.	. . .	.	.
.	.	.	. . .	.	.
.	.	.	. . .	.	.
I_{p-1}	b_{m-1}	b_0	. . .	b_{k-2}	a_{n-1}

(where $x' \equiv x \bmod m$)

If m and p are not relatively prime, it is still possible that we can find an input sequence of period p for the model studied in Theorem 3-6, where p is a factor of the period of the given sequence. To test whether there exists an input sequence of period p for the model of Theorem 3-6 or not, we may use Theorem 3-7 to be given later. The proof of Theorem 3-7 is along the same line as that

of Theorem 3-5.

Theorem 3-7 :

A sequence $a_0a_1 \cdots a_{n-1} \cdots$ of period $n = p - mq$ can be generated by an indirect logic k-FSR with input sequence $I_0I_1 \cdots I_{p-1} \cdots$ of period p fed into the output function, if and only if there exists an exact H_k-sequence $b_0b_1 \cdots b_{m-1} \cdots$ of period m such that in any row block r_i of the array shown in Table 3-9, $0 \leq i \leq p-1$, any two columns which coincide in the first k symbols also coincide in the last symbol.

Corollary 3-7-1:

In the array of Theorem 3-7, any two distinct row blocks r_i and r_j, $0 \leq i, j \leq p - 1$, can be assigned the same input symbol, if and only if any two columns, one in the row block r_i and the other in the row block r_j, which coincide in the first k symbols also coincide in the last symbol.

3-2-3. Minimum number of memory elements used by the FSR without input sequence.

In this section, we will show that if a composite FSR B generates all the sequences generated by an FSR A, then the number of memory elements used by B is greater than or equal to that used by A. Before we proceed the discussion, we need the following terms.

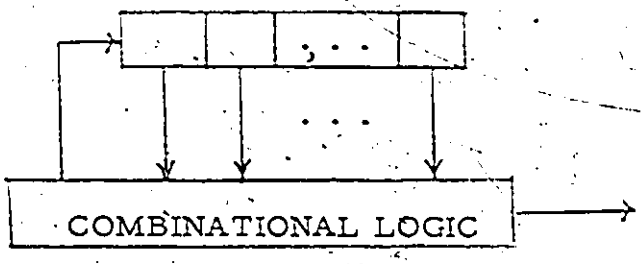
Table 3-9.

An array of Theorem 3-7.

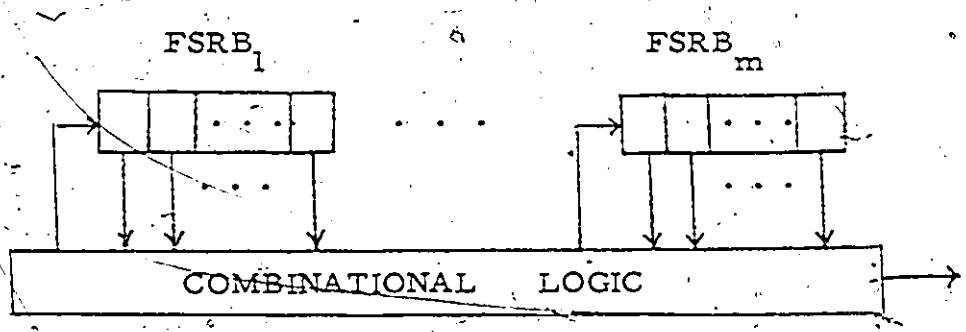
	b_0	$b_{p'}$	. . .	$b_{((\ell-1)p)'}$
	b_1	$b_{(p+1)'}$	. . .	$b_{((\ell-1)p+1)'}$
	.	.	. . .	.
r_0	.	.	. . .	.
	.	.	. . .	.
	b_{k-1}	$b_{(p+k-1)'}$	. . .	$b_{((\ell-1)p+k-1)'}$
	a_0	a_p	. . .	$a_{(\ell-1)p}$
	b_1	$b_{(p+1)'}$	. . .	$b_{((\ell-1)p+1)'}$
	b_2	$b_{(p+2)'}$	. . .	$b_{((\ell-1)p+2)'}$
r_1	.	.	. . .	.
	.	.	. . .	.
	.	.	. . .	.
	b_k	$b_{(p+k)'}$	. . .	$b_{((\ell-1)p+k)'}$
	a_1	a_{p+1}	. . .	$a_{(\ell-1)p+1}$
	.	.	. . .	.
	.	.	. . .	.
	.	.	. . .	.
	$b_{(p-1)'}$	$b_{(2p-1)'}$	. . .	$b_{(\ell p-1)'}$
	$b_{p'}$	$b_{(2p)'}$	. . .	$b_{(\ell p)'}$
r_{p-1}	.	.	. . .	.
	.	.	. . .	.
	.	.	. . .	.
	$b_{(p+k-2)'}$	$b_{(2p+k-2)'}$	. . .	$b_{(\ell p+k-2)'}$
	a_{p-1}	a_{2p-1}	. . .	$a_{\ell p-1}$

($x' \equiv x \bmod m$)

Consider two FSR's A and B as shown in Fig. 3-12 (a) and (b), respectively, where FSRB is a composite FSR which consists of sub-FSR's $FSRB_1$, $FSRB_2$, ... and $FSRB_m$. FSRA is said to be realized by FSRB, if for every state s of FSRA there exists at least one state $(s_1, s_2, \ldots, s_m)$ of FSRB corresponding to s, where s_1, s_2, ... and s_m are states of $FSRB_1$, $FSRB_2$, ... and $FSRB_m$, respectively, and FSRB generates all the sequences generated by FSRA.



(a) FSRA



(b) FSRB

Fig. 3-12. Two FSR's A and B.

Theorem 3-8 :

Let n_A and n_B denote the total number of memory elements used by FSRA and BSRB, respectively, where FSRA is a single FSR and FSRB is a composite FSR discussed above. If FSRB realizes FSRA, then $n_B > n_A$.

Proof:

Let FSRB consist of m sub-FSR's. For every state s of FASA , there exists at least one state $(s_1, s_2, \ldots, s_m)$ of FSRB corresponds to s . Let n be the number of states in FSRA, and n_1, n_2, ... and n_m be the number of states in $FSRB_1$, $FSRB_2$, ... and $FSRB_m$, respectively. Then $n_1 n_2 \ldots n_m > n$, $\lceil \log_2 n_1 n_2 \ldots n_m \rceil > \lceil \log_2 n \rceil$, but $\lceil \log_2 n_1 \rceil + \lceil \log_2 n_2 \rceil + \ldots + \lceil \log_2 n_m \rceil > \lceil \log_2 n_1 n_2 \ldots n_m \rceil$, and $\lceil \log_2 n \rceil$ is the minimum number of memory elements needed to assign the states of FSRA and $\lceil \log_2 n_1 \rceil$, $\lceil \log_2 n_2 \rceil$, ... and $\lceil \log_2 n_m \rceil$ are the minimum number of memory elements needed to assign the states of $FSRB_1$, $FSRB_2$, ... and $FSRB_m$, respectively, hence $n_B > n_A$.

Q. E. D.

3-2-4. Discussion.

Besides the synthesis methods studied in section 3-2-2, the other combination, e.g., both feedback and output function are fed with input sequence, can be similarly studied.

There is a structure similarity between the modified model and the generalized (m,n) - FSR of Reed and Turn. The realization method studied is based on the assumption that the appropriate input sequences are available. If the appropriate input sequence is not available, we need additional FSR's to generate it, then we obtain a composite FSR. Any attempt to shorten the length of the FSR in the FSR realization only gives a composite FSR when the appropriate input sequence is not available, and the total number of memory elements used is not reduced as seen from Theorem 3-8. Therefore, in the FSR realization without available input sequences, first, we find the integer $k = \lceil \log_2 n \rceil$, where n is the period of the given sequence. If the sequence is an exact H_k-sequence, we use a direct logic k-FSR; otherwise, an indirect logic k-FSR, both studied in Section 3-2-1, for realization.

However, in the modern LSI technique, it is useful to have a chip which can be used for different purposes. The previous studied model, FSR with input sequence, can be used for several purposes. Without input sequence, the feedback function is restricted to the state variables of the shift register, it can be used as a conventional FSR sequence generator. A Boolean function is said to be restricted to certain number of variables if

CHAPTER 4

MODULO-TWO-SUM DECOMPOSITION OF BINARY LINEAR SEQUENCES OF FINITE PERIOD

In this chapter, a method of decomposing a finite-periodic-binary-linear-sequence into a modulo-2-sum of several finite-periodic-binary-linear-sequences is presented.

Let $\hat{a} = a_0 a_1 \ldots a_{n-1} \ldots$ be a binary sequence of period n. The sequence $\hat{a}$ is said to be a modulo-2-sum of its component sequences $\hat{b}$, $\hat{c}$, ... and $\hat{d}$, i.e., $\hat{a} = \hat{b} \oplus \hat{c} \oplus \ldots \oplus \hat{d}$, if and only if there exist a set of binary sequences $\hat{b} = b_0 b_1 \ldots b_{n_1-1} \ldots$, $\hat{c} = c_0 c_1 \ldots c_{n_2-1} \ldots$, ... and $\hat{d} = d_0 d_1 \ldots d_{n_m-1} \ldots$ of periods n_1, n_2, ... and n_m, respectively, such that $a_i = b_i \oplus c_i \oplus \ldots \oplus d_i$ for all $i \geq 0$, where $\oplus$ is a modulo-2-sum operation.

The compnent sequences will give us a new way of sequence representation. When the sum of periods of the component sequences is less than the period of the original sequence, we need less storage space, although a special procedure is required either to decompose or to recover the sequence.

Sequence decomposition problem has been studied originally by Hsieh [28]. He has used combinatorial

graph to analyze a sequence and to decompose a sequence into a modulo-2-sum or a logic-AND of several component sequences. The main fact for sequence decomposition obtained by him is described as follows. Let $(\hat{a})$ be a given sequence of period n and let $(\hat{b}) = (\hat{a}) \oplus D^r(\hat{a})$ and $(\hat{b}) \neq (\hat{0})$, where $D^r(\hat{a})$ denotes the r th cyclic shift of the sequence $(\hat{a})$. If $per(\hat{a}) < per(\hat{a})/2$ and either $per(\hat{b})$ and r are relatively prime or $pa(\hat{b}) > r$ and r does not divide $per(\hat{b})$, then we can always find two periodic sequences $\hat{c}$ and $\hat{d}$ such that $\hat{a} = \hat{c} \oplus \hat{d}$ and the periods of both $\hat{c}$ and $\hat{d}$ are less than $per(\hat{a})$. In the sequence decomposition procedure, r must be chosen properly. Fortunately, r is related to the factors of the period of the given sequence. The sequence decomposition procedure involves a lot of computations in checking the factors of the period of the given sequence $\hat{a}$ to find a r such that $per((\hat{a}) \oplus D^r(\hat{a})) < per(\hat{a})$ and solving a set of linear simultaneous equations for the component sequences after a factor r which satisfies the previous condition is obtained. Each time only two component sequences can be obtained from the decomposition procedure.

In this chapter, a different approach will be given. The method will involve finding a minimum length LFSR realization of the given sequence. Fortunately, this problem has been solved by Massey [35]. The LFSR realization for the geneartion of a specified sequence $\hat{a}$ of

finite period studied in Section 3-1 will give us the minimum polynomial of the given sequence, say f(X). The sequence â has a fractional polynomial representation I(X)/f(X), where I(X) is a polynomial uniquely determined by the initial k-tuple of the sequence â and the coefficients of f(X), k is the degree of f(X). The sequence decomposition problem can be handled algebraically after a fractional polynomial representation of the sequence is obtained. A necessary and sufficient condition for the nontrivial decomposibility of a sequence will be obtained. A modulo-2-sum-sequence-decomposition-procedure will be given. Hsieh's method will be demonstrated by an example. Some interesting properties of the minimum polynomial of a sequence will be obtained.

In the context of this chapter, the modulo-2-sum operation $\oplus$ and the addition $+$ are considered to be the same.

4-1. Fractional Polynomial Representation of a Sequence

Let $(\hat{a}) = a_0 a_1 \ldots a_{n-1}$ be a sequence of period n. From Section 3-1 of the previous chapter, we can find the minimum polynomial of the sequence â, say $f(X) = 1 + c_1 X + \ldots + c_k X^k$. A polynomial f(X) is called a MINIMUM POLYNOMIAL of a sequence â, if f(X) generates â and no polynomial with degree less than the degree of f(X) can generate â. Now, the sequence â can be

represented by a polynomial $S(X)$ $(1 + X^n + X^{2n} + \ldots)$, where $S(X) = a_0 + a_1 X + \ldots + a_{n-1} X^{n-1}$. Use Eq. (2-1) for simplification, we can compute $f(X)\ S(X)$ as follows

$$f(X)\ S(X) = (1 - X^n)\ I(X), \tag{4-1}$$

where $I(X) = a_0 + (a_1 + c_1 a_0)\ X + \ldots + (a_{k-1} + c_1 a_{k-2} + \ldots + c_{k-1} a_0)\ X^{k-1}$. (4-2)

From Eq. (4-1), we have

$$I(X)/f(X) = S(X)/(1 - X^n) = S(X)\ (1 + X^n + X^{2n} + \ldots),$$

hence, we finally obtain a fractional polynomial representation $I(X)/f(X)$ of the sequence â.

4-2. Modulo-2-Sum Sequence Decomposition

In the following, three theorems will be proved. The first theorem, Theorem 4-1, will provide us a general background for sequence decomposition. The second theorem, Theorem 4-2, will give us a necessary and sufficient condition for the nontrivial decomposibility of a sequence. For a given sequence, if it is decomposable, the solution is in general not unique. The third theorem, Theorem 4-3, will provide us all the solutions of sequence decomposition, once a solution is obtained. Following the theorems, a modulo-2-sum sequence decomposition procedure will be given. Hsieh's method also will be briefly discussed by an example.

Let us denote the set of sequences generated by $f(X)$ by $S(f) = \{ I(X)/f(X) : \deg I(X) < \deg f(X) \}$ and a partial

set of sequences generated by $(f(X))^m$ by

$$S'(f^m) = \{ I(X)/(f(X))^m : \deg I(X) < \deg f(X) \},$$

where $f(X)$ and m must be clearly specified. It is clear that $S'(f^m) \subset S(f^m)$.

Now, let $f(X) = (f_1(X))^{n_1} (f_2(X))^{n_2} \ldots (f_m(X))^{n_m}$, where $f_i(X)$ are irreducible polynomials and n_i are integers for all i, $1 \le i \le m$. By partial fraction expansion theorem, $I(X)/f(X)$ has a partial fraction exapnasion

$$\begin{aligned} I(X)/f(X) = {} & I_{11}(X)/f_1(X) + I_{12}(X)/(f_1(X))^2 + \cdots + I_{1n_1}(X)/(f_1(X))^{n_1} \\ & + I_{21}(X)/f_2(X) + I_{22}(X)/(f_2(X))^2 + \cdots + I_{2n_2}(X)/(f_2(X))^{n_2} \\ & + \cdots \\ & + I_{m1}(X)/f_m(X) + I_{m2}/(f_m(X))^2 + \cdots + I_{mn_m}(X)/(f_m(X))^{n_m}, \end{aligned}$$

where $\deg I_{ij}(X) < \deg f_i(X)$ for all $0 \le i \le m$ and $1 \le j \le n_i$. The right hand side of Eq. (4-3) belong to

$$\begin{aligned} S(F) = {} & S'(f_1) + S'(f_1^2) + \ldots + S'(f_1^{n_1}) + \\ & S'(f_2) + S'(f_2^2) + \ldots + S'(f_2^{n_2}) + \ldots + \\ & S'(f_m) + S'(f_m^2) + \ldots + S'(f_m^{n_m}). \end{aligned}$$

Hence, for every $I(X)/f(X) \in S(f)$, $I(X)/f(X) \in S(F)$ and vice versa. This result can be stated as the following

Theorem 4-1 :

Let $f(X) = (f_1(X))^{n_1} (f_2(X))^{n_2} \ldots (f_m(X))^{n_m}$, where $f_i(X)$ are irreducible polynomials and n_i are integers

for all i, $1 \leq i \leq m$. Then

$$S(f) = S'(f_1) + S'(f_1^2) + \ldots + S'(f_1^{n_1}) + S'(f_2) + S'(f_2^2) + \ldots + S'(f_2^{n_2}) + \ldots + S'(f_m) + S'(f_m^2) + \ldots + S'(f_m^{n_m}).$$

It is clear that $S'(f) = S(f)$. From Theorem 4-1, we have the following

Corollary 4-1-1:

$$S(f^m) = S'(f) + S'(f^2) + \ldots + S'(f^m)$$

Corollary 4-1-2:

Let $f(X) = f_1(X)\ f_2(X) \ldots f_m(X)$, where $f_i(X)$ are irreducible polynomials for all i, $1 \leq i \leq m$. Then $S(f) = S(f_1) + S(f_2) + \ldots + S(f_m)$.

Before we prove Theorem 4-2, we need to define the following terms and lemmas.

Definition 4-1 :

A sequence decomposition is said to be NONTRIVIAL if the sum of the periods of the component sequences is less than the period of the original sequence, and all the periods of the component sequences are not divisible to one another.

The following lemma can be proved directly from the definitions, we state it without proof.

Lemma 4-1 :

If $I_1(X)/f_1(X)$ and $I_2(X)/f_2(X)$ represent the sequences $\hat{a}$ and $\hat{b}$, respectively, then $I_1(X)/f_1(X) + I_2(X)/f_2(X)$ represents the sequence $\hat{a} \oplus \hat{b}$.

Lemma 4-2 :

Let $I(X)/f(X)$ represent a sequence $\hat{a}$, then $f(X)$ is the minimum polynomial of $\hat{a}$, if and only if $(I(X), f(X)) = 1$.

Proof:

We prove first that if $f(X)$ is the minimum polynomial of $\hat{a}$, then $(I(X), f(X)) = 1$. Assume contrary, i.e., $(I(X), f(X)) = h(X)$. Let $f(X) = g(X)\ h(X)$ and $I(X) = I'(X)\ h(X)$, then $I(X)/f(X) = I'(X)/g(X)$, i.e., the polynomial $g(X)$ also generates $\hat{a}$. But, deg $g(X)$ < deg $f(X)$ which contradicts the hypothesis that $f(X)$ is the minimum polynomial of $\hat{a}$.

Next, we shown that if $(I(X), f(X)) = 1$, then $f(X)$ is the minimum polynomial of $\hat{a}$. Let $f'(X)$ be the minimum polynomial of $\hat{a}$ and let $I'(X)/f'(X)$ also represent $\hat{a}$, $(I'(X), f'(X)) = 1$. Since both $I(X)/f(X)$ and $I'(X)/f'(X)$ represent $\hat{a}$, $I(X)/f(X) = I'(X)/f'(X)$ or $I(X)\ f'(X) = I'(X)\ f(X)$. Now, $(I(X), f(X)) = 1$ implies $f(X) \mid f'(X)$, and $(I'(X), f'(X)) = 1$ implies $f'(X) \mid f(X)$, hence $f(X) = f'(X)$.

Q.E.D.

The following lemma has been proved by Zierler [49], we quote it without proof as follows.

Lemma 4-3 : (Zierler)

Let n be the exponent of f(X).

(i) If f(X) is irreducible, then per($\hat{a}$) = n for every nonzero sequence $\hat{a} \in S(f)$.

(ii) If f(X) is the minimum polynomial of a sequense $\hat{a}$, then per($\hat{a}$) = n and per($\hat{b}$) | n for every $\hat{b} \in S(f)$.

Now, from Theorem 4-1, if $f(X) = (f_1(X))^{n_1}(f_2(X))^{n_2} \ldots (f_m(X))^{n_m}$, then any sequence $\hat{a} \in S(f)$ can be represented by

$$\hat{a} = \delta_{11}\hat{a}_{11} \oplus \delta_{12}\overline{\hat{a}_{12}} \oplus \ldots \oplus \delta_{1n_1}\hat{a}_{1n_1}$$
$$\oplus \delta_{21}\hat{a}_{21} \oplus \delta_{22}\hat{a}_{22} \oplus \ldots \oplus \delta_{2n_2}\hat{a}_{2n_2}$$
$$\oplus \ldots$$
$$\oplus \delta_{m1}\hat{a}_{m1} \oplus \delta_{m2}\overline{\hat{a}_{m2}} \oplus \ldots \oplus \delta_{mn_m}\hat{a}_{mn_m},$$

where $\hat{a}_{ij} \in S'(f_i^j)$ and $\delta_{ij} \in \{0, 1\}$ for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n_i$.

Consider the following cases. First, let f(X) be an irreducible polynomial with exponent n, and let $I(X)/f(X)$ represent a sequence $\hat{a}$. Then $I(X)/f(X)$ has a partial sum expansion

$$I(X)/f(X) = I_1(X)/f(X) + I_2(X)/f(X) + \ldots + I_m(X)/f(X),$$

where $I(X) = I_1(X) + I_2(X) + \ldots + I_m(X)$. Let $I_i(X)/f(X)$ represent $\hat{a}_i$ for all i, $1 \leq i \leq m$, then $\hat{a}$ has a trivial decomposition $\hat{a} = \hat{a}_1 \oplus \hat{a}_2 \oplus \ldots \oplus \hat{a}_m$, where $per(\hat{a}_1) = per(\hat{a}_2) = \ldots = per(\hat{a}_m) = per(\hat{a}) = n$. In this trivial decomposition, the total sum of periods of the component sequences is a multiple of the period of the original sequence.

Next, let $f(X) = (g(X))^m$ be the minimum polynomial of a sequence $\hat{a}$ represented by $I(X)/f(X)$, where $g(X)$ is an irreducible polynomial. Then $I(X)/f(X) = I_1(X)/g(X) + I_2(X)/(g(X))^2 + \ldots + I_m(X)/(g(X))^m$, where $\deg I_i(X) < \deg g(X)$ for all i, $1 \leq i \leq m$. Let $I_m(X)/(g(X))^m$ represent $\hat{a}_m$. Since $(I_m(X), (g(X))^m) = 1$, $(g(X))^m$ is the minimum polynomial of $\hat{a}_m$ and so $per(\hat{a}_m) = per(\hat{a})$. In this case, the sum of the periods of the component sequences is greater than the period of the original sequence. Again, we obtain a trivial decomposition.

Finally, in general, if $f(X) = (f_1(X))^{n_1}(f_2(X))^{n_2} \ldots (f_m(X))^{n_m}$ is the minimum polynomial of a sequence $\hat{a}$ represented by $I(X)/f(X)$, to avoid trivial decomposition, we do the partial fraction expansion of $I(X)/f(X)$ as follows,

$$I(X)/f(X) = I_1(X)/(f_1(X))^{n_1} + I_2(X)/(f_2(X))^{n_2} + \ldots + I_m(X)/(f_m(X))^{n_m},$$

where deg $I_i(X)$ < deg $(f_i(X))^{n_i}$ for all i, $1 \leq i \leq m$. If there exists an i such that the exponent of $(f_i(X))^{n_i}$ equals to the exponent of f(X), we still obtain a trivial decomposition. Only when all the exponents of $(f_i(X))^{n_i}$ are less than the exponent of f(X) for all i, there is a nontrivial decomposition. Now, if the exponent of $(f_i(X))^{n_i}$ divides the exponent of $(f_j(X))^{n_j}$ for $i \neq j$, then the period of the sequence represented by $I_j(X)/(f_j(X))^{n_j}$ is equal to the period of the sequence represented by $I_i(X)/(f_i(X))^{n_i} + I_j(X)/(f_j(X))^{n_j}$. Hence, we must merge all the terms $I_i(X)/(f_i(X))^{n_i}$, with exponents of $(f_i(X))^{n_i}$ divide the exponent of $(f_j(X))^{n_j}$, to $I_j(X)/(f_j(X))^{n_j}$. Finally, after merging, we can obtain a partial fraction expansion of $I(X)/f(X)$ as follows,

$$I(X)/f(X) = K_1(X)/g_1(X) + K_2(X)/g_2(X) + \ldots + K_p(X)/g_p(X).$$

Let $K_i(X)/g_i(X)$ represent $\hat{a}_i$ for all i, $1 \leq i \leq p$, then we can obtain a nontrivial decomposition of the sequence $\hat{a}$ as $\hat{a} = \hat{a}_1 \oplus \hat{a}_2 \oplus \ldots \oplus \hat{a}_p$.

From the previous discussions, a necessary and sufficient condition for the nontrivial decomposibility of a sequence can be stated in the following

Theorem 4-2:

A sequence $\hat{a}$ has a nontrivial decomposition, if and only if the minimum polynomial f(X) of $\hat{a}$ is reducible such that all the factors of the power of irreducible factors

of f(X) have exponents less than the exponent of f(X).

The nontrivial solution of the decomposition of binary sequence into modulo-2-sum of several component sequences is not unique in general. The following theorem will show that once a solution is obtained, the other solutions can be easily obtained. First, we need the following

Lemma 4-4 :

For any two binary sequences $\hat{b}$ and $\hat{c}$, the equation $\hat{b} \oplus \hat{c} = \bar{\hat{b}} \oplus \bar{\hat{c}}$ holds, where $\bar{\hat{b}}$ and $\bar{\hat{c}}$ are the complement sequences of $\hat{b}$ and $\hat{c}$, respectively.

Proof:

For every $b_i, c_i \in \{ 0, 1 \}$, it is easy to see that the equation $b_i \oplus c_i = \bar{b}_i \oplus \bar{c}_i$ holds as shown in Table 4-1, therefore $\hat{b} \oplus \hat{c} = \bar{\hat{b}} \oplus \bar{\hat{c}}$.

Q.E.D.

Table 4-1.

Truth tables of $b_i \oplus c_i$ and $\bar{b}_i \oplus \bar{c}_i$.

b_i	c_i	$\bar{b}_i$	$\bar{c}_i$	$b_i \oplus c_i$	$\bar{b}_i \oplus \bar{c}_i$
0	0	1	1	0	0
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Theorem 4-3 :

If a binary sequence $\hat{a}$ has a modulo-2-sum decomposition $\hat{a} = \hat{b} \oplus \hat{c} \oplus \ldots \oplus \hat{d}$, then any even number of the component sequences complemented is also a decomposition of $\hat{a}$.

Proof:

From Lemma 4-4, it follows that for an even number of component sequences, the modulo-2-sum of these sequences is equal to the modulo-2-sum of the complement of these sequences.

Q.E.D.

From the previous discussion, we summarize a modulo-2-sum sequence decomposition procedure as follows.

1. Find the minimum length LFSR realization for the given sequence $\hat{a}$ and get the minimum polynomial $f(X)$ of $\hat{a}$.
2. If $f(X)$ is irreducible, there is no nontrivial solution.
3. If $f(X)$ is reducible, say $f(X) = (f_1(X))^{n_1} (f_2(X))^{n_2} \ldots (f_m(X))^{n_m}$, where $f_i(X)$ are irreducible and n_i are integers for all i, $1 \leq i \leq m$, we test the exponents of $(f_i(X))^{n_i}$ for all i. If there exists an i such that the exponent of $(f_i(X))^{n_i}$ is equal to the exponent of $f(X)$, then there is no nontrivial solution. Merge all the factors $(f_i(X))^{n_i}$, whose exponents divide the

exponent $(f_j(X))^{n_j}$, to $(f_j(X))^{n_j}$, then $f(X)$ can be written as $f(X) = g_1(X)g_2(X) \ldots g_p(X)$, where the exponents of $g_i(X)$ are not divisible to one another.

4. Find $I(X)$ from Eq. (4-2).

5. Find the partial fraction expansion of $I(X)/f(X)$ as follows

$$I(X)/f(X) = I_1(X)/g_1(X) + I_2(X)/g_2(X) + \cdots + I_p(X)/g_p(X),$$

where deg $I_i(X)$ < deg $g_i(X)$ for all i, $1 \leq i \leq p$.

6. Find the sequences represented by $I_i(X)/g_i(X)$ for all i, say $\hat{a}_i$, then $\hat{a} = \hat{a}_1 \oplus \hat{a}_2 \oplus \ldots \oplus \hat{a}_p$ is a solution. Use Theorem 4-3 to obtain all the other solutions.

Example 4-1 :

Consider the following sequence of period 84,

$(\hat{a}) =$ 011010001111110011111011001001100110000111000001010100101000100000010010111101011100.

From the minimum length LFSR realization given in Section 3-1 of the previous chapter, we can find the minimum polynomial $f(X) = 1 + X^4 + X^5 + X^6 + X^{10} + X^{11}$ of $\hat{a}$.

The exponents of the factors $(1 + X)^2$, $(1 + X + X^2)^3$ and $(1 + X + X^3)$ of $f(X)$ are less than the exponent of $f(X)$. The exponent of $(1 + X)^2$ is a factor of the exponent of $(1 + X + X)^3$, thefefore,

$g_1(X) = (1 + X^2)(1 + X + X^2)^3 = 1 + X + X^2 + X^6 + X^7 + X^8$ and $g_2(X) = 1 + X + X^3$. From Eq. (4-2), we can find $I(X) = X + X^2 + X^4 + X^5 + X^8$. Hence $\hat{a}$ can be represented by $I(X)/f(X) = (X + X^2 + X^4 + X^5 + X^8)/(1 + X^4 + X^5 + X^6 + X^{10} + X^{11})$ which has a partial fraction expansion $I(X)/f(X) = (1 + X + X^4 + X^6 + X^7)/(1 + X + X^2 + X^6 + X^7 + X^8) + (1 + X^2)/(1 + X + X^3)$. Now, $(1 + X + X^4 + X^6 + X^7)/(1 + X + X^2 + X^6 + X^7 + X^8)$ and $(1 + X^2)/(1 + X + X^3)$ represent the sequences $(\hat{b}) = 101110110101$ and $(\hat{c}) = 1101001$, respectively. Hence $\hat{a} = \hat{b} \oplus \hat{c}$. The other solution can be obtained from Theorem 4-3 as follows, $(\bar{\hat{b}}) = 010001001010$, $(\hat{c}) = 0010110$ and $\hat{a} = \bar{\hat{b}} \oplus \bar{\hat{c}}$.

We now use Hsieh's method to decompose the same sequence given in Example 4-1. The basic concept of his method is described as follows. Let $D^i(\hat{a})$ denote the ith cyclic shift of sequence $\hat{a}$. Assume the sequence $\hat{a}$ can be decomposed into two sequences $\hat{b}$ and $\hat{c}$, i.e., $\hat{a} = \hat{b} \oplus \hat{c}$, and $per(\hat{b}) = m$, $m \nmid per(\hat{c})$. Then

$$(\hat{a}) \oplus D^m(\hat{a}) = (\hat{b} \oplus \hat{c}) \oplus D^m(\hat{b} \oplus \hat{c})$$
$$= (\hat{b}) \oplus D^m(\hat{b}) \oplus (\hat{c}) \oplus D^m(\hat{c}),$$

but $(\hat{b}) \oplus D^m(\hat{b}) = (\hat{0})$, since $per(\hat{b}) = m$, hence $(\hat{a}) \oplus D^m(\hat{a}) = (\hat{c}) \oplus D^m(\hat{c})$. The operation $(\hat{a}) \oplus D^m(\hat{a})$ eliminates the sequence $\hat{b}$. Let $(\hat{d}) = (\hat{a}) \oplus D^m(\hat{a})$ and $per(\hat{d}) = p$. From $(\hat{d}) = (\hat{c}) \oplus D^m(\hat{c})$ we can obtain a set of simultaneous linear equations.

has no solution. Next, consider $(\hat{a}) \oplus D^{21}(\hat{a}) = 011000$, the following set of simultaneous linear equations

$$c_0 + c_{21 \bmod 6} = c_0 + c_3 = 0$$
$$c_1 + c_4 = 1$$
$$c_2 + c_5 = 1$$
$$c_3 + c_0 = 0$$
$$c_4 + c_1 = 0$$
$$c_5 + c_2 = 0$$

also has no solution. Similarly, the other cases $(\hat{a}) \oplus D^{28}(\hat{a})$ and $(\hat{a}) \oplus D^{14}(\hat{a})$ also will not give us any solution. Consider $(\hat{a}) \oplus D^{12}(\hat{a}) = 1001110$, the following set of simultaneous linear equations

$$d_0 + d_{6 \bmod 3} = d_0 + d_0 = 0$$

$$d_1 + d_1 = 1$$

$$d_2 + d_2 = 1$$

has no solution. Next, try $(\bar{b}) \oplus D^3(\bar{b}) = 000011$, the following set of simultaneous linear equations

$$d_0 + d_3 = 0$$

$$d_1 + d_4 = 0$$

$$d_2 + d_5 = 0$$

$$d_3 + d_0 = 0$$

$$d_4 + d_1 = 1$$

$$d_5 + d_2 = 1$$

has no solution. Next, try $(\bar{b}) \oplus D^4(\bar{b}) = 111000001110$, it also has no solution. Therefore $\bar{b}$ is not decomposable. Hence $\hat{a} = \bar{b} \oplus \hat{c}$, where $(\bar{b}) = 101110110101$ and $(\hat{c}) = 1101001$. Similarly, we can obtain $\bar{a} = \bar{b} \oplus \bar{c}$, where $(\bar{b}) = 010001001010$ and $(\bar{c}) = 0010110$.

From the preceding discussions, it is easy to see that Hsieh's method involves a lot of computations in checking the factors r_i of the period of the given sequence and finding the modulo-2-sum of the sequence and its r_ith cyclic shift, and solving a set of simultaneous linear equations for the component sequences after a factor r_i which satisfies the condition $per((\hat{a}) + D^{r_i}(\hat{a})) < per(\hat{a})$

is obtained. Hsieh's method is a straight forward approach, it has less complexity in compare with our method, but a lot of computations involved in checking the redundancy can not be avoided. Our method involves finding the minimum length LFSR realization of the given sequence. To do this we can use Massey's method. The method also involves finding all the irreducible factors of the minimum polynomial of the given sequence. To do this we can use Berlekamp algorithm [6]. Again, the method requires to find the exponents of the factors of the minimum polynomial. Unfortunately, there is no known method that we can use although only those factors of the period of the given sequence should be tested.

4-3. Some Properties of the Minimum Polynomial of a Sequence

In the following, some interesting properties of the minimum polynomial of a sequence will be obtained.

Theorem 4-4 :

Let $f_1(X)$, $f_2(X)$ and $f(X)$ be the minimum polynomials of the sequences $\hat{a}$, $\hat{b}$ and $\hat{a} \oplus \hat{b}$, respectively, and let $g(X) = LCM\{ f_1(X), f_2(X) \}$, then $f(X) \mid g(X)$.

Proof:

Let $I_1(X)/f_1(X)$, $I_2(X)/f_2(X)$ and $I(X)/f(X)$ represent the sequences $\hat{a}$, $\hat{b}$ and $\hat{a} \oplus \hat{b}$, respectively.

Let $f_1(X) = k_1(X)h(X)$ and $f_2(X) = k_2(X)h(X)$, where $h(X) = GCD\{ f_1(X), f_2(X) \}$, then

$$I(X)/f(X) = I_1(X)/f_1(X) + I_2(X)/f_2(X) =$$

$$(I_1(X)k_2(X) + I_2(X)k_1(X))/g(X)$$

or

$$I(X)g(X) = f(X)(I_1(X)k_2(X) + I_2(X)k_1(X)).$$

But $(I(X), f(X)) = 1$, hence $f(X) \mid g(X)$.

Q.E.D.

From Theorem 4-4, we have the following

Corollary 4-4-1 :

Let $f_1(X), f_2(X), \ldots, f_m(X)$ and $f(X)$ be the minimum polynomials of $\hat{a}, \hat{b}, \ldots, \hat{c}$ and $\hat{a} \oplus \hat{b} \oplus \ldots \oplus \hat{c}$, respectively, and let $g(X) = LCM\{ f_1(X), f_2(X), \ldots, f_m(X) \}$, then $f(X) \mid g(X)$.

From the proof of Lemma 4-2, it is easy to see that a sequence has an unique minimum polynomial, and hence an unique fractional polynomial representation.

Definition 4-2 :

Let $(\hat{a}) = a_0a_1 \ldots a_{n-1}$ be a sequence of period n. The REVERSE SEQUENCE of $(\hat{a})$ is a sequence obtained from $(\hat{a})$ in reverse order. Denote the reverse sequence of $(\hat{a})$ by $rev(\hat{a}) = a_{n-1}a_{n-2} \ldots a_0$.

Let $I(X)/f(X)$ represent a sequence $(\hat{a}) = a_0a_1 \ldots a_{n-1}$, i.e., $I(X)/f(X) = (a_0 + a_1X + \ldots + a_{n-1}X^{n-1})/(1-X^n)$.

Then $I(X^{-1})/f(X^{-1}) = (a_0 + a_1X^{-1} + \ldots + a_{n-1}X^{-(n-1)})/(1-X^{-n})$

$= - X(a_{n-1} + a_{n-2}X \ldots + a_0X^{n-1})/(1-X^n)$.

Let the degree of $f(X)$ be k, then

$-X^{k-1}I(X^{-1})/X^k f(X^{-1}) = (a_{n-1} + a_{n-2}X + \ldots + a_0X^{n-1})/(1-X^n)$.

Hence we have the following

Theorem 4-5 :

Let $f(X)$ be the minimum polynomial of a sequence $\hat{a}$, then $X^k f(X^{-1})$ is the minimum polynomial of $rev(\hat{a})$, where $k = \deg f(X)$.

Definition 4-3 :

The COMPLEMENT SEQUENCE of a sequence $(\hat{a}) = a_0a_1 \ldots a_{n-1}$ is a sequence obtained from $(\hat{a})$ by complementing all the digits of $\hat{a}$. Denote the complement sequence of $(\hat{a})$ by $(\bar{\hat{a}}) = \bar{a}_0\bar{a}_1 \ldots \bar{a}_{n-1}$.

Note that $(\hat{a}) \oplus (\bar{\hat{a}}) = (\hat{1})$, where $(\hat{1}) = 1$ is a ONE sequence. The one sequence $(\hat{1})$ can be represented by $1/(1 - X)$.

Let $I(X)/f(X)$ and $I'(X)/f'(X)$ represent the sequences $(\hat{a})$ and $(\bar{\hat{a}})$, respectively. Then $I(X)/f(X) + I'(X)/f'(X) = 1/(1-X) = 1/(1 + X)$, where $-1 = 1$ in GF(2). Two cases arise:

Case 1:- $(1 + X) \mid f(X)$

Let $f(X) = (1 + X)g(X)$ and $I(X)/f(X) = 1/(1 + X)$

$h(X)/g(X)$, then $I'(X)/f'(X) = h(X)/g(X)$.

Case 2: $(1 + X) \nmid f(X)$

$$I'(X)/f(X) = 1/(1 + X) + I(X)/f(X)$$

$$= (f(X) + I(X)(1 + X))/(1 + X)f(X).$$

We summarize the above result in the following

Theorem 4-6 :

Let $f(X)$ be the minimum polynomial of a sequence $\hat{a}$,

(i) if $(1 + X) | f(X)$, then the minimum polynomial of $(\bar{\hat{a}})$ is $f(X)/(1 + X)$;

(ii) if $(1 + X) \nmid f(X)$, then the minimum polynomial of $(\bar{\hat{a}})$ is $(1 + X)f(X)$.

In the following, we will prove some useful lemmas for the development of the theorems followed.

Lemma 4-5 :

The sequences of $S(f)$ form a ring.

Proof:

Let $\hat{a}$ and $\hat{b}$ be two arbitrary sequences of $S(f)$ then $\hat{c} = \hat{a} \oplus \hat{b}$ is also a sequence of $S(f)$. Let $I_1(X)/f(X)$ and $I_2(X)/f(X)$ represent $\hat{a}$ and $\hat{b}$, respectively, then $(I_1(X) + I_2(X))/f(X)$ represents $\hat{c}$. Define multiplication as follows. Let $I(X) = I_1(X)I_2(X) \bmod f(X)$ and let $I(X)/f(X)$ represent $\hat{d}$, then $\hat{d} = \hat{a} \cdot \hat{b}$. Note that $\hat{a} \oplus \hat{0} = \hat{0} \oplus \hat{a} = \hat{a}$,

$I(X) \cdot 1 = 1 \cdot I(X) = I(X)$ and the multiplication $\cdot$ is distributive over the addition $\oplus$. Therefore, the sequences of S(f) form a ring.

Q.E.D.

Let α be a root of $f(X) = 1 + c_1X + \ldots + c_kX^k$, $c_k \neq 0$. Let R be the set of all residue polynomials in α modulo $f(\alpha)$ with coefficients over GF(2), then $R = \{ r_0 + r_1\alpha + \ldots + r_{k-1}\alpha^{k-1} : r_i \in GF(2), 0 \leq i \leq k-1 \}$ is a ring.

We identify S(f) and R as follows.

Lemma 4-6 :

S(f) is isomorphic to R.

Proof:

The mapping ϕ: $S(f) \longrightarrow R$ defined by $\phi(I(X)/f(X)) = I(\alpha)$ is an isomorphism.

Q.E.D.

Definition 4-4 :

A CYCLIC SHIFT of a sequence $(\hat{a}) = a_0a_1 \ldots a_{n-1}$ is defined by $D(\hat{a}) = a_{n-1}a_0 \ldots a_{n-2}$. An ith cyclic shift of $(\hat{a})$ is $D^i(\hat{a}) = a_{n-i}a_{n-i+1} \ldots a_{n-i-1}$.

Let f(X) be the minimum polynomial of a sequence $(\hat{a}) = a_0a_1 \ldots a_{n-1}$. Let the degree of f(X) be k, a state transition diagram is uniquely determined by f(X) as shown in Fig. 4-1, where only those states related to

the sequence $(\hat{a})$ are specified. From the state transition diagram, start from the initial state $(a_0 a_1 \cdots a_{k-1})$, pick up the left most digit of each state and go through the cycle, we can obtain the sequence $(\hat{a})$. Similarly, start from the initial state $(a_{n-1} a_0 \cdots a_{k-2})$, we can obtain the sequence $D(\hat{a})$.

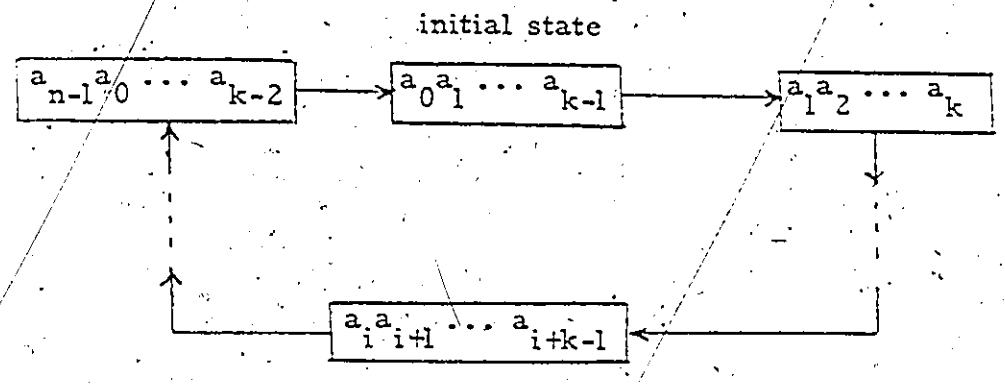


Fig. 4-1. A state transition diagram defined by $f(X)$.

Hence, $f(X)$ also generates $D(\hat{a})$. Furthermore, no polynomial $g(X)$ of degree less than the degree of $f(X)$ can generate $D(\hat{a})$. Since the polynomial $g(X)$ which generates $D(\hat{a})$ will generate $(\hat{a})$ also, which will contradict the hypothesis that $f(X)$ is the minimum polynomial of $(\hat{a})$.

This result can be stated as the following

Lemma 4-7 :

If $f(X)$ is the minimum polynomial of a sequence $(\hat{a})$, then $f(X)$ is also the minimum polynomial of $D^i(\hat{a})$ for all i.

Lemma 4-8 :

If $I(\alpha)$ corresponds to a sequence $(\hat{a})$ then $\alpha I(\alpha)$ corresponds to $D(\hat{a})$.

Proof:

Let $I(X)/f(X)$ and $I'(X)/f(X)$ represent the sequence $(\hat{a}) = a_0 a_1 \ldots a_{n-1}$ and $D(\hat{a})$, respectively. Then

$$I(X)/f(X) = (a_0 + a_1 X + \ldots + a_{n-1}X^{n-1})/(1-X^n),$$

$$I'(X)/f(X) = (a_{n-1} + a_0 X + \ldots + a_{n-2}X^{n-1})/(1-X^n).$$

Hence, $I'(X)(1 - X^n) - X\, I(X)(1 - X^n) = a_{n-1}(1 - X^n)\, f(X)$, or $I'(X) - X\, I(X) = a_{n-1} f(X)$. But $f(\alpha) = 0$, hence $I'(\alpha) = \alpha I(\alpha)$.

Q.E.D.

Theorem 4-7 :

Let $I(X)/f(X)$ represent a sequence $(\hat{a})$ with minimum polynomial $f(X)$. Let $(\hat{b}) = (\hat{a}) \oplus D^r(\hat{a})$ and $(\hat{b}) \neq (\hat{0})$, where $r < n$ and n is the exponent of $f(X)$. If $per(\hat{b}) < per(\hat{a})$, then

(i) $((1 + X^r), f(X)) \neq 1$;

(ii) $f(X)$ is reducible.

Proof:

(i) The sequence $(\hat{b})$ can be represented by $I(X)/f(X) + X^r I(X)/f(X) = I(X)(1 + X^r)/f(X)$. If $(I(X)(1 + X^r), f(X)) = 1$, then $f(X)$ is the minimum poly-

nomial of $(\hat{b})$ and per$(\hat{b}) = n$, which contradicts the hypothesis that per$(\hat{b})$ < per $(\hat{a})$. Therefore, $(I(X)(1 + X^r), f(X)) \neq 1$, but $(I(X), f(X)) = 1$, so $((1 + X^r), f(X)) \neq 1$.

(ii) Assume contrary, i.e., $f(X)$ is irreducible, then per$(\hat{b}) = n$. Again, we obtain a contradiciton. Hence, $f(X)$ is reducible.

Q.E.D.

Theorem 4-8 :

Let $f(X)$ be the minimum polynomial of a sequence $(\hat{a})$. If $f(X)$ is reducible such that all the factors of $f(X)$ have exponents less than the exponent of $f(X)$, then there exists an integer r such that $(\hat{b}) = (\hat{a}) \oplus D^r(\hat{a})$, $(\hat{b}) \neq (\hat{0})$ and per$(\hat{b})$ < per $(\hat{a})$.

Proof:

Let $I(X)/_{f(X)}$ represent $(\hat{a})$. Let $h(X)$ be a factor of $f(X)$, say $f(X) = h(X)g(X)$, with the exponent of $g(X)$ less than the exponent of $f(X)$. There exists an integer r such that $h(X) \mid (1 + X^r)$, say $(1 + X^r) = h(X)k(X)$, then $I(X)/_{f(X)} + X^r I(X)/_{f(X)} = I(X)(1 + X^r)/_{f(X)} = I(X)k(X)/_{g(X)}$. Now let $(\hat{b}) = (\hat{a}) \oplus D^r(\hat{a})$, then per $(\hat{b})$ < per$(\hat{a})$.

Q.E.D.

It is interesting to compare Theorem 4-7 and Theorem 4-8 with the following result obtained by Hsieh [28].

Theorem 4-9 : (Hsieh)

Let $(\underline{b}) = (\hat{a}) \in D^r(\hat{a})$ and $(\underline{b}) \neq (\hat{0})$. If $per(\underline{b}) < per(\hat{a})/2$ and either $(per(\underline{b}), r) = 1$ or $per(\underline{b}) > r$ and $r \nmid per(\underline{b})$ ‡, then we can always find two periodic sequences $(\hat{c})$ and $(\hat{d})$ such that $(\hat{a}) = (\hat{c}) \oplus (\hat{d})$, $per(\hat{c}) < per(\hat{a})$ and $per(\hat{d}) < per(\hat{a})$.

4-4. Discussion

In this chapter, a different approach to find a modulo-2-sum decomposition of binary sequences of finite periods has been obtained. The generalization over GF(q) follows parallel lines.

The method studied in this chapter is passed on to the LFSR realization of the given sequence which gives fractional polynomial representation of the sequence. The partial fraction expansion of the fractional polynomial is equivalent to the decomposition of the LFSR which generates the sequence. Hence, actually we have transformed the sequence decomposition problem into machine decomposition problem.

A more general sequence decomposition problem, decomposing a binary sequence of finite period into a number of component sequences of finite periods such that the original sequence is a logic function of its component

---

‡ The condition that either $(per(\underline{b}), r) = 1$ or $per(\underline{b}) > r$ and $r \nmid per(\underline{b})$ is added by the author. The necessity of this additional condition is clearly indicated by the previous example.

component sequences, will be studied in the next chapter.

CHAPTER 5

LOGIC FUNCTION DECOMPOSITION OF BINARY SEQUENCES OF FINITE PERIODS

In this chapter, a method of decomposing a finite periodic binary sequence into a logic function of several finite periodic binary sequences is presented.

Let $\hat{a} = a_0 a_1 \cdots a_{n-1} \cdots$ be a binary sequence of period n. The sequence $\hat{a}$ is said to be a logic function of its component sequences $\hat{b}$, $\hat{c}$, ... and $\hat{d}$, i.e., $\hat{a} = f(\hat{b}, \hat{c}, \ldots, \hat{d})$, if and only if there exists a set of binary sequences $\hat{b} = b_0 b_1 \cdots b_{n_1-1} \cdots$, $\hat{c} = c_0 c_1 \cdots c_{n_2-1} \cdots$, ... and $\hat{d} = d_0 d_1 \cdots d_{n_m-1}$ of periods n_1, n_2, ... and n_m, respectively, and a Boolean function $f(x_1, x_2, \ldots, x_m)$ such that $a_i = f(b_i, c_i, \ldots, d_i)$ for all $i \geq 0$.

The sequence decomposition problem has been studied originally by Hsieh [28]. A solution to decompose a binary sequence of finite period into a logic-AND of several binary sequences of finite periods has been obtained by Hsieh.

In this chapter, a more general sequence decomposition problem will be studied, namely, to decompose a binary sequence of finite period into a logic function of several binary sequences of finite periods. A solution

will be provided for both prime decompositions and conditional non-prime decompositions. In a sequence decomposition, if the periods of all the component sequences are either prime or a power of prime numbers, then it is called a prime decomposition. A necessary and sufficient condition for the decomposibility of a sequence will be obtained for the general case. A necessary and sufficient condition for the decomposibility of a sequence for both prime decompositions and conditional non-prime decompositions also will be obtained. A logic-function-sequence-decomposition-procedure for both prime decompositions and conditional non-prime decompositions will be given. A logic-function-sequence-decomposition-machine will be introduced for both the prime decompositions and the conditional non-prime decompositions.

5-1. Special Logic Function Sequence Decompositions

In the following, we will consider the decomposition of a sequence into a special logic function of several component sequences.

A binary sequence $(\bar{a}) = a_0 a_1 \ldots a_{n-1}$ of period $n = m \cdot p$ can be decomposed into an exclusive-OR, logic-AND or logic-OR function of two binary sequences $(\bar{b}) = b_0 b_1 \ldots b_{m-1}$ and $(\bar{c}) = c_0 c_1 \ldots c_{p-1}$ of periods m and p, respectively, if and only if the following corresponding set of simultaneous equations has a solution

$$b_i \bmod m \ \ominus \ c_i \bmod p = a_i, \qquad (5\text{-}1)$$

$$b_i \bmod m \cdot c_i \bmod p = a_i, \qquad (5\text{-}2)$$

$$b_i \bmod m + c_i \bmod p = a_i, \qquad (5\text{-}3)$$

for $i = 0, 1, \ldots, n-1$, where $\ominus$, $\cdot$ and $+$ are exclusive-OR, logic-AND and logic-OR operations, respectively. The truth of this assertion is obvious, since if the set of equations in Eq's (5-1), (5-2) and (5-3) has a solution then $(\hat{a}) = (\hat{b}) \ominus (\hat{c})$, $(\hat{a}) = (\hat{b}) \cdot (\hat{c})$ and $(\hat{a}) = (\hat{b}) + (\hat{c})$, respectively, and vice versa. For example, let $(\hat{a}) = 100101101001000$ be a sequence of period 15. The following set of simultaneous equations,

$$b_0 \cdot c_0 = 1, \quad b_2 \cdot c_0 = 1, \quad b_1 \cdot c_0 = 0,$$
$$b_0 \cdot c_1 = 0, \quad b_0 \cdot c_1 = 1, \quad b_2 \cdot c_1 = 1,$$
$$b_2 \cdot c_2 = 0, \quad b_1 \cdot c_2 = 0, \quad b_0 \cdot c_2 = 0,$$
$$b_0 \cdot c_3 = 1, \quad b_2 \cdot c_3 = 1, \quad b_1 \cdot c_3 = 0,$$
$$b_1 \cdot c_4 = 0, \quad b_0 \cdot c_4 = 0, \quad b_2 \cdot c_4 = 0,$$

has a solution

$$\begin{cases} b_0 = b_2 = 1 \\ b_1 = 0 \end{cases}, \qquad \begin{cases} c_0 = c_1 = c_3 = 1 \\ c_2 = c_4 = 0 \end{cases}$$

Therefore, $(\hat{a}) = (\hat{b}) \cdot (\hat{c})$, where $(\hat{b}) = 101$ and $(\hat{c}) = 11010$.

Repeat the preceding decomposition, we can find a special function sequence decomposition of a given

sequence into several component sequences. Each time we factorize the period of the sequence considered into two factors and try to solve a set of simultaneous euqations for the component sequences. We can also obtain a logic function sequence decomposition by applying some combinations of the previous special function sequence decompositions. For example, suppose we first apply the binary operation + to a sequence $\hat{a}$ and obtain a logic-OR decomposition of $\hat{a}$ as $\hat{a} = \hat{b} + \hat{c}$, and we next apply the binary operation $\cdot$ to the sequence $\hat{c}$ and obtain a logic-AND decomposition of $\hat{c}$ as $\hat{c} = \hat{e} \cdot \hat{f}$, then we finally obtain a logic function decomposition of $\hat{a}$ as $\hat{a} = \hat{b} + \hat{e} \cdot \hat{f}$.

In the previous decomposition, we have factorized the period of the sequence considered into two factors. In general the assertion is also truth for any two integers whose LCM equals to the period of the sequence considered.

5-2. A Necessary and Sufficient Condition for the Decomposibility of a Sequence

From a given sequence $(\hat{a}) = a_0 a_1 \ldots a_{n-1}$ of period n, we can easily construct an antonomous sequential machine (ASM) M with state transition and output table shown in Table 5-1.

In some cases, the ASM can be decomposed into a number of sub-ASM's. A set of sequences generated by these sub-ASM's

can be considered as a result of decomposition of the given sequence. A necessary and sufficient condition for the decomposibility of a sequence is given in the following Theorem.

Table 5-1.

State transition and output table of an ASM M.

⇨	state transition	output
s_0	s_1	a_0
s_1	s_2	a_1
.	.	.
.	.	.
.	.	.
s_{n-2}	s_{n-1}	a_{n-2}
s_{n-1}	s_0	a_{n-1}

First we need the following

Definition 5-1 :

A Boolean function f is said to be K-DIVISIBLE, if and only if there exist k functions f_i for $i = 1, 2, \ldots, k$ such that

$f(x_{11}, x_{12}, \ldots, x_{1n_1}, x_{21}, x_{22}, \ldots, x_{2n_2}, \ldots, x_{k1}, x_{k2}, \ldots, x_{kn_k})$

$= f(f_1(x_{11}, x_{12}, \ldots, x_{1n_1}), f_2(x_{21}, x_{22}, \ldots, x_{2n_2}),$

$f_k(x_{k1}, x_{k2}, \ldots, x_{kn_k}))$.

Theorem 5-1 :

A binary sequence of finite period can be decomposed into a logic function of k component sequences of finite periods, if and only if there exists an ASM decomposition of the corresponding ASM , which generates the sequence, into k sub-ASM's with a state assignment such that the output function is k-divisible.

Proof:

The sufficiency part of the theorem is proved as follows. Let M be an ASM which generates a given arbitrary sequence â. Let M be decomposed into k sub-ASM's M_1, M_2, ... and M_k. Let $\{x_{i1}, x_{i2}, \ldots, x_{in_i}\}$ be sets of binary variables assigned to the states of sub-ASM's M_i for $i = 1, 2, \ldots$ and k such that the output function is k-divisible, where $n_i \geq 1$ are integers, then

$f(x_{11}, x_{12}, \ldots, x_{1n_1}, x_{21}, x_{22}, \ldots, x_{2n_2}, \ldots, x_{k1}, x_{k2}, \ldots, x_{kn_k}) = f(f_1(x_{11}, x_{12}, \ldots, x_{1n_1}), f_2(x_{21}, x_{22}, \ldots, x_{2n_2}), \ldots, f_k(x_{k1}, x_{k2}, \ldots, x_{kn_k}))$.

For every output

$a_i = f(a_{11}, a_{12}, \ldots, a_{1n_1}, a_{21}, a_{22}, \ldots, a_{2n_2}, \ldots, a_{k1}, a_{k2}, \ldots, a_{kn_k})$

of ASM M, there are k outputs $b_i = f_1(a_{11}, a_{12}, \ldots, a_{1n_1})$, $c_i = f_2(a_{21}, a_{22}, \ldots, a_{2n_2})$, ... and $d_i = f_k(a_{k1}, a_{k2}, \ldots, a_{kn_k})$ of sub-ASM's M_1, M_2, ... and M_k, respectively, such that $a_i = f(b_i, c_i, \ldots, d_i)$. Hence $\hat{a} = f(\hat{b}, \hat{c}, \ldots, \hat{d})$.

The necessity part of the theorem is clear, since the k component sequences can be generated by k sub-ASM's, the output function of the composite ASM is certainly k-divisible.

Q.E.D.

Theorem 5-1 is an existence theorem. A logic-function-sequence-decomposition-procedure for both prime decompositions and conditional non-prime decompositions will be given in the following sections.

5-3. Prime Decompositions

Before we discuss the prime decomposition of a sequence, let us first examine the periodicity between a sequence and its component sequences.

Theorem 5-2

Let n be the period of a sequence $\hat{a}$ and n_1, n_2, ... and n_m be the periods of its component sequences $\hat{b}$, $\hat{c}$, ... and $\hat{d}$, respectively, such that $\hat{a} = f(\hat{b}, \hat{c}, \ldots, \hat{d})$, then $n = \text{LCM}\{n_1, n_2, \ldots, n_m\}$.

Proof:

It is clear that $per(\bar{b}) = per(\bar{b})$, $per(\bar{b} \cdot \bar{c}) = LCM\{ per(\bar{b}), per(\bar{c}) \}$ and $per(\bar{b} + \bar{c}) = LCM\{ per(\bar{b}), per(\bar{c}) \}$ for any sequences $\bar{b}$ and $\bar{c}$. The theorem follows by applying repeatedly the above facts to the terms of $f(\bar{b}, \bar{c}, \ldots, \bar{d})$.

Q.E.D.

From theorem 5-2, it is clear that if the period of a sequence is either a prime or a power of a prime number, then there is no nontrivial decomposition. Hence we have the following

Definition 5-2 :

A sequence decomposition is called PRIME DECOMPOSITION, if and only if all the periods of the component sequences are either prime numbers or powers of prime numbers.

In the following, a partition procedure which is essential for sequence decompositions will be developed. The partition procedure will partition the digits of an unknown sequence within a period into two groups, called partition pair, so that assignment can be made on the partition pair to obtain the component sequence.

Let $(\bar{a}) = \overline{a_0 a_1 \ldots a_{n-1}}$ be a sequence of period $n = p_1^{k_1} p_2^{k_2} \ldots p_m^{k_m}$, where p_i are prime numbers and $k_i \geq 1$ are integers for all i, $1 \leq i \leq m$. Let $n_i = p_i^{k_i}$

for all i, $1 \leq i \leq m$, and $(\hat{b}) = b_0 b_1 \cdots b_{n_1-1}$, $(\hat{c}) = c_0 c_1 \cdots c_{n_2-1}$, ... and $(\hat{d}) = d_0 d_1 \cdots d_{n_m-1}$ be a set of unknown sequences of periods n_1, n_2, ... and n_m, respectively. If the function $f(x_1, x_2, \ldots, x_m)$ of Table 5-2 exists, then the sequence $\hat{a}$ has a prime decomposition $\hat{a} = f(\hat{b}, \hat{c}, \ldots, \hat{d})$.

In Table 5-2, we have assumed $n_1 < n_2 < \ldots < n_m$ for presentation convenienece. If the function $f(x_1, x_2, \ldots, x_m) = x_1 \oplus x_2 \oplus \ldots \oplus x_m$, we obtain a modulo-2-sum decomposition $\hat{a} = \hat{b} \oplus \hat{c} \oplus \ldots \oplus \hat{d}$. Similarly, if $f(x_1, x_2, \ldots, x_m) = x_1 x_2 \ldots x_m$ or $f(x_1, x_2, \ldots, x_m) = x_1 + x_2 + \ldots + x_m$, then we obtain a logic-AND decomposition $\hat{a} = \hat{b} \cdot \hat{c} \ldots \hat{d}$ or a logic-OR decomposition $\hat{a} = \hat{b} + \hat{c} + \ldots + \hat{d}$.

Now, the problem is how to find an assignment of the unknown sequences $(\hat{b}) = b_0 b_1 \cdots b_{n_1-1}$, $(\hat{c}) = c_0 c_1 \cdots c_{n_2-1}$, ... and $(\hat{d}) = d_0 d_1 \cdots d_{n_m-1}$ such that the function $f(x_1, x_2, \ldots, x_m)$ exists, when the only information given is the values in the range of $f(x_1, x_2, \ldots, x_m)$ which is given by the original sequence $(\hat{a}) = a_0 a_1 \cdots a_{n-1}$.

From Table 5-2, let us first fix the values of x_1, x_2, ... and x_{m-1}, say b_i, c_j, ... and e_k, respectively.

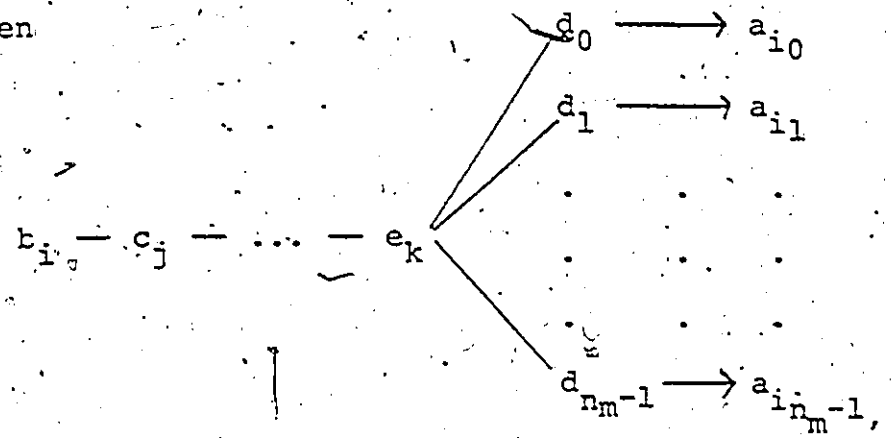
Table 5-2.

Truth table of function

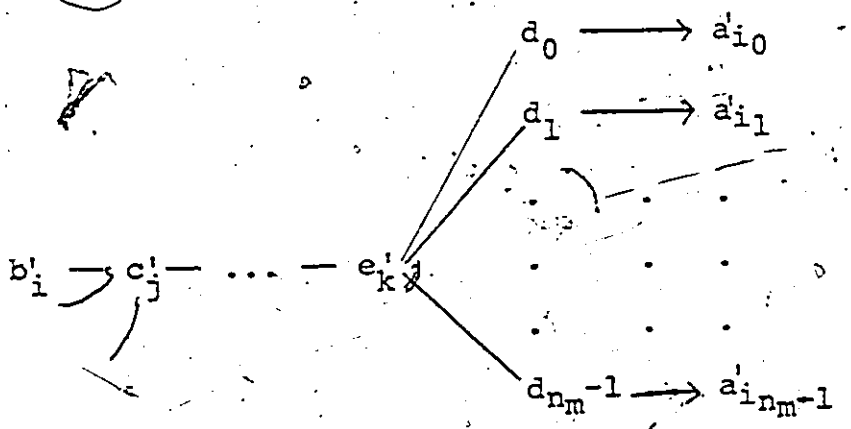
$f(x_1, x_2, \ldots, x_m)$

x_1	x_2	. . .	x_m	f
b_0	c_0	. . .	d_0	a_0
b_1	c_1	. . .	d_1	a_1
.	.	. . .	.	.
.	.	. . .	.	.
.	.	. . .	.	.
b_{n_1-1}	c_{n_1-1}	. . .	d_{n_1-1}	a_{n_1-1}
.	.	. . .	.	.
.	c_{n_2-1}	. . .	d_{n_2-1}	a_{n_2-1}
.	.	. . .	.	.
.	.	. . .	d_{n_m-1}	a_{n_m-1}
.	.	. . .	.	.
.	.	. . .	.	.
.	.	. . .	.	.
b_{n_1-1}	c_{n_2-1}	. . .	d_{n_m-1}	a_{n-1}

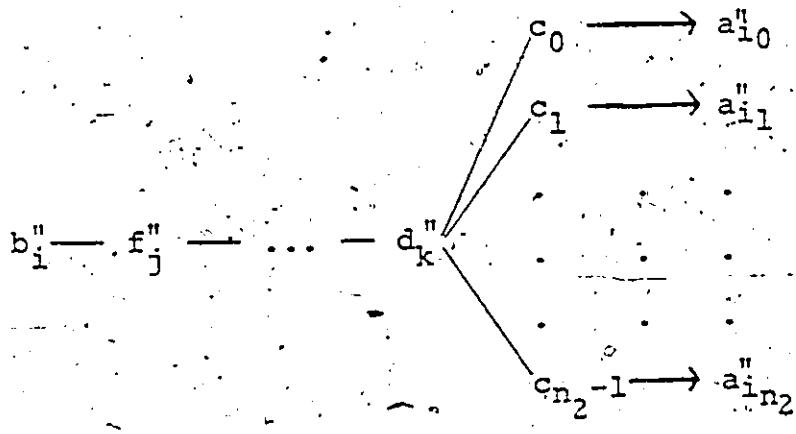
Then



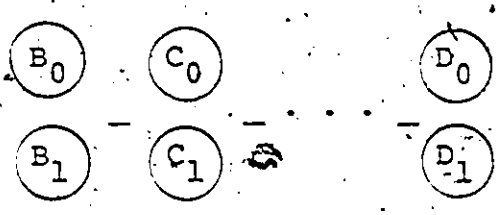
where $\longrightarrow$ means that when the function f takes a value from its domain at the left hand side, it gets a value from its range at the right hand side. For a consistent assignment, the values of d_0, d_1, ... and d_{n_m-1} must be consistent with that of a_{i_0}, a_{i_1}, ... and $a_{i_{nm-1}}$. Hence the set $\{d_0, d_1, \ldots, d_{n_m-1}\}$ either remains unchanged or is partitioned into two sets. The former corresponds to the case when all the values of a_{i_0}, a_{i_1}, ... and $a_{i_{n_m-1}}$ are the same; the latter corresponds to the case when there are two different values among a_{i_0}, a_{i_1}, ... and $a_{i_{n_m-1}}$. Next, we change the values of x_1, x_2, ... and x_{m-1}, say b'_i, c'_j ... and e'_k, respectively. Then



Assume that both $\{ a_{i_0}, a_{i_1}, \ldots, a_{i_{n_m-1}} \}$ and $\{ a'_{i_0}, a'_{i_1}, \ldots, a'_{i_{n_m-1}} \}$ have two different values among themselves, then we can obtain two sets of partition of $\{ d_0, d_1, \ldots, d_{n_m-1} \}$. If these two sets of partition do not coincide with each other, then the function $f(x_1, x_2, \ldots, x_m)$ fails to exist. In this case, the sequence has no prime decompositon. Check the partitions for all possible values of x_1, x_2, .. and x_{m-1}. If all the partitions coincide, we obtain a partition of $\{ d_0, d_1, \ldots, d_{n_m-1} \}$ into a partition pair, say D_0 and D_1. Next, we can permute the variables x_1, x_2, ... and x_m. For example, we can fix the values of x_1, x_3, ... and x_m, say b''_i, f''_j, ... and d''_k, respectively. Then



By the same procedure, we can partition $\{ c_0, c_1 \ldots c_{n_2-1} \}$, say C_0 and C_1. Similarly, we can obtain a partition pair of $\{ b_0, b_1, \ldots, b_{n_1-1} \}$, say B_0 and B_1 and so on. Finally, we can get a set of partition pairs



Now, 0 and 1 can be assigned to each partition pair. Since there are totally m partition pairs, there are totally 2^m possible assignements. Each assignement will give us a solution. The function $f(x_1, x_2, \ldots, x_n)$ can be obtained from the assignement given to the set of partition pairs.

The partition procedure discussed above can be described in a more systematic way. To obtain a partition pair we have to check only the values in the right hand side of $\longrightarrow$ of the previous procedure. We can construct an array from the given sequence by counting down the sequence and picking up the digits from the sequence systematically in certain order. Before we proceed the discussion further, we need the following

Lemma 5-1 :

Let $b_0b_1 \ldots b_{n_1-1} \ldots$, $c_0c_1 \ldots c_{n_2-1} \ldots$, ... and $d_0d_1 \ldots d_{n_m-1} \ldots$ be a set of sequences of periods n_1, n_2, ... and n_m, respectively, where n_1, n_2, ... and n_m are relatively prime to one another. Let $k = \text{LCM}\{n_2, n_3, \ldots, n_m\} = n_2n_3 \ldots n_m$ and let the sequences be arranged in the following array

b_0 c_0 . . . d_0

b_1 c_1 . . . d_1

. . . . . .

. . . . . .

. . . . . .

b_{n_1-1} . . . . . .

. . . . . .

c_{n_2-1} . . . . .

. . . . . .

. . . . . d_{n_m-1}

. . . . . .

. . . . . .

. . . . . .

Then the first row of the m-tuple $(b_1\ c_0, \ldots, d_0)$ will appear in the position of the ℓkth row, where ℓ is the least integer such that $\ell k - jn_1 = 1$.

Proof:

Since the $(m-1)$-tuple $(c_0, \ldots, d_0)$ will appear in a period of k and b_1 is in the second position of the sequence $b_0 b_1 \ldots b_{n_1-1} \ldots$, to match b_1 to $(c_0, \ldots, d_0)$, we must have $\ell k = jn_1 + 1$ for some integers ℓ and j. Since k and n_1 are relatively prime to each other, there always exist two integers ℓ and j such that $\ell k - jn_1 = 1$. Therefore, the first row of the m-tuple

$(b_1, c_0, \ldots, d_0)$ will appear in the position of the ℓkth row, where ℓ is the least integer such that $\ell k - jn_1 = 1$.

Q.E.D.

Now, let $\hat{a} = a_0 a_1 \ldots a_{n-1} \ldots$ be a sequence of period $n = p_1^{k_1} p_2^{k_2} \ldots p_m^{k_m} = n_1 n_2 \ldots n_m$ and let $\ell_1 = \ell(n/n_1)$, where ℓ is the least integer such that $\ell(n/n_1) - jn_1 = 1$. We can build an array of n rows and n_1 columns from the given sequence as follows,

a_0	a_{ℓ_1}	$a_{2\ell_1}$	. . .	$a_{(n_1-1)\ell_1}$
$a_{(n_1-1)\ell_1+1}$	a_1	a_{ℓ_1+1}	. . .	$a_{(n_1-2)\ell_1+1}$
$a_{(n_1-2)\ell_1+2}$	$a_{(n_2-1)\ell_1+2}$	a_2	. . .	$a_{(n_1-3)\ell_1+2}$
.	.	.	. . .	.
.	.	.	. . .	.
$a_{\ell_1+n_1-1}$	$a_{2\ell_1+n_1-1}$	$a_{3\ell_1+n_1-1}$	. . .	a_{n_1-1}
a_{n_1}	$a_{\ell_1+n_1}$	$a_{2\ell_1+n_1}$	. . .	$a_{(n_1-1)\ell_1+n_1}$
.	.	.	. . .	.
.	.	.	. . .	.
a_{ℓ_1+n-1}	$a_{2\ell_1+n-1}$	$a_{3\ell_1+n-1}$	. . .	a_{n-1}

where the first digit of the first row is obtained from the first digit of the given sequence, the other digits of the first row are obtained from the $k\ell_1$ th digits of the given sequence for $k = 1, 2, \ldots, (n_1-1)$; the second digit of the

second row is obtained from the second digit of the given sequence, the otehr digits of the second row are obtained from the $(k\ell_1 + 1)$th digits of the given sequence for $k = 1, 2, \ldots, (n_1 - 1)$; and so on. From Lemma 5-1, it can be seen easily that each row in the above array corresponds to a set of values in the right hand side of $\longrightarrow$ of the previous partition procedure with respect to partitionning $\{ b_0, b_1, \ldots, b_{n_1-1} \}$. Hence the above array can be used for checking whether a partition pair of $\{ b_0, b_1, \ldots, b_{n_1-1} \}$ exists or not. If in the above array, all the rows, except those rows of the same symbol, are either the same or complementary to one another, then the partition pair exists and can be obtained easily by grouping those b_i corresponding to 0 in one group and those b_i corresponding to 1 in the other group. Furthermore, due to the periodicity of the sequence, the first n/n_1 rows of the array form a subarray, the array consists of n_1 subarrays with the subarray of the first n/n_1 rows repeated n_1 times. Hence, we have to consider only the subarray of the first n/n_1 rows of the array as follows

$$
\begin{array}{cccc}
a_0 & a_{\ell_1} & \cdots & a_{(n_1-1)\ell_1} \\
a_{(n_1-1)\ell_1+1} & a_1 & \cdots & a_{(n_1-2)\ell_1+1} \\
\cdots & \cdots & & \\
\cdots & a_{n/n_1-1} & \cdots &
\end{array}
$$

where a_{n/n_1-1} appears in the position of the (n/n_1)th row and the $(n/n_1 \bmod n_1)$ th column. The subarray shown above has totally n elements, it contains all the digits in one period of the sequence.

Similarly, let $\ell_2 = \ell(n/n_2)$, where ℓ is the least integer such that $\ell(n/n_2) - jn_2 = 1$, we can build an array of n/n_2 rows and n_2 columns as follows

$$\begin{array}{ccccc} a_0 & & a_{\ell_2} & & a_{(n_2-1)\ell_2} \\ a_{(n_2-1)\ell_2+1} & & a_1 & \cdots & a_{(n_2-2)\ell_2+1} \\ & & \vdots & & \\ \cdots & & a_{n/n_2-1} & & \cdots \end{array}$$

which will check the existence of the partition pair of $\{ c_0, c_1, \ldots, c_{n_2-1} \}$, and so on. Finally, let $\ell_m = \ell(n/n_m)$, where ℓ is the least integer such that $\ell(n/n_m) - jn_m = 1$, we can build an array of n/n_m rows and n_m columns as follows

$$\begin{array}{ccccc} a_0 & & a_{\ell_m} & & a_{(n_m-1)\ell_m} \\ a_{(n_m-1)\ell_m+1} & & a_1 & \cdots & a_{(n_m-2)\ell_m+1} \\ & & \vdots & & \\ \cdots & & a_{n/n_m-1} & & \cdots \end{array}$$

which will cehck the existence of the partition pair of $\{ d_0, d_1, \ldots, d_{n_m-1} \}$. If all the partition pairs

exist, we can assign 0 and 1 to each partition pair and obtain a prime decomposition. We summarize the preceding discussions in the following

Theorem 5-3 :

A sequence $\hat{a} = a_0 a_1 \ldots a_{n-1} \ldots$ of period $n = p_1^{k_1} p_2^{k_2} \ldots p_m^{k_m} = n_1 n_2 \ldots n_m$ has a prime decomposition, if and only if the m arrays of n/n_1, n/n_2, ... and n/n_m rows and n_1, n_2, ... and n_m columns, respectively, built above satisfy the condition that for every array all the rows, except those rows of the same symbol, are either the same or complementary to one another.

From the previous discussion, a logic-function-sequence-decomposition-procedure for prime decompositions can be outlined as follows..

1. Find all the powers of prime factors of n, say $n = p_1^{k_1} p_2^{k_2} \ldots p_m^{k_m} = n_1 n_2 \ldots n_m$, where $n_i = p_i^{k_i}$ for all i, $1 \le i \le m$, and n is the period of the given sequence. Let i = 1.
2. Find $\ell_i = \ell(n/n_i)$, where ℓ is the least integer such that $\ell(n/n_i) - jn_i = 1$, and build the array of n/n_i rows and n_i columns as shown in Theorem 5-3. If the array doesn't satisfy the condition of Theorem 5-3, there is no solution for prime decomposition; otherwise, get a partition pair..
3. Increase i by 1. If $i \le m$, go to Step 2.

4. Assign 0 and 1 to all the partition pairs, find the component sequences and the function, and get a solution. Other solutions can be obtained from different assignments assigned to the partition pairs.

Example 5-1 :

Consider the following sequence $\hat{a}$ of period 180

($\hat{a}$) 10111111110111101110, 00110111100110111010,
11111111111111101011, 01110011101110001111,
11110111111110111111, 00111011100111011110,
10111011110111001110, 11110011111110011011.

Here, we have $n = 180 = 2^2 \cdot 3^2 \cdot 5 = 4 \cdot 9 \cdot 5$, $n_1 = 4$, $n_2 = 9$ and $n_3 = 5$.

For $i = 1$, to find ℓ such that $45\ell - 4j = 1$, we have $\ell = 1$ and $j = 11$, hence $\ell_1 = 45\ell = 45$ and we can build an array of 45 rows and 4 columns as follows

1	1	1	1
1	0	1	1
1	0	1	1
1	0	1	1
1	1	1	1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	1	1.

Since the array satisfies the condition of Theorem 5-3, we obtain a partition pair $\{b_0, b_2, b_3\}$ and $\{b_1\}$.

For $i = 2$, to find ℓ such that $20\ell - 9j = 1$, we have $\ell = 5$ and $j = 11$, hence $\ell_2 = 20\ell = 100$ and we can build an array of 20 rows and 9 columns as follows

1	0	0	0	1	1	0	1	1
1	0	0	0	1	1	0	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	0	0	0	1	1	0	1	1

Since the array satisfies the condition of Theorem 5-3, we obtain a partition pair $\{c_0, c_4, c_5, c_7, c_8\}$ and $\{c_1, c_2, c_3, c_6\}$.

For $i = 3$, to find ℓ sucht that $36\ell - 5j = 1$, we have $\ell = 1$ and $j = 7$, hence $\ell_3 = 36\ell = 36$ and we can build an array of 36 rows and 5 columns as follows

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

.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
1	1	1	1	1

Since the array satisfies the condition of Theorem 5-3, we obtain a partition pair $\{d_0, d_4\}$ and $\{d_1, d_2, d_3\}$.

Assign 0 to $\{b_1\}$, $\{c_1, c_2, c_3, c_6\}$ and $\{d_1, d_2, d_3\}$ and 1 to $\{b_0, b_2, b_3\}$, $\{c_0, c_4, c_5, c_7, c_8\}$ and $\{d_0, d_4\}$, three component sequences $(\hat{b}) = 1011$, $(\hat{c}) = 100011011$ and $(\hat{d}) = 10001$ can be obtained. From the assignment, we can set up a truth table of the function f as shown in Table 5-3. From Table 5-3, we can obtain the function $f = x_2 + x_1\bar{x}_3$. Therefore, the sequence $\hat{a}$ has a prime decomposition $\hat{a} = \hat{c} + \hat{b} \cdot \bar{\hat{d}}$.

Table 5-3.

Truth table of function f of Example 5-1.

x_2	x_3	x_3	f
1	1	1	1
0	0	0	0
1	0	0	1
0	1	1	1
1	1	0	1
0	1	1	1
1	0	1	0
0	0	1	1

The other solutions can be obtained by assigning different assignments to the partition pairs obtained.

5-4. Conditional Non-Prime Decompositions

A little modification of the previous prime decomposition procedure will enable us to find a conditional non-prime decomposition of a sequence. Let n be the period of a given sequence, first we have to find a set of integers n_1, n_2, ... and n_m such that $n = \text{LCM}\{n_1, n_2, \ldots, n_m\}$ and any $m - 1$ integers of the m integers n_1, n_2, ... and n_m have a least common multiple (LCM) less than n. Furthermore, in order to find a non-trivial decomposition, the integers n_1, n_2, ... and n_m should be chosen such that $n_1 + n_2 + \ldots + n_m < n$. One way of choosing those n_i is to factorize n into a product of factors n_i, each factor doesn't have to be a prime or power of a prime number. In this section, we are going to solve the following problem. Assume that we have obtained a combination satisfying the above-mentioned condition, we want to know whether there exists a decomposition with respect to the combination obtained or not, and if it exists, find the solutions.

In the prime decomposition of Section 5-3, n_1, n_2, ... and n_m are relatively prime to one another. Hence, in Lemma 5-1, n_1 and $k = \text{LCM}\{n_2, \ldots, n_m\}$ are relatively prime, and so there always exist some integers i and j

such that $ik - jn_1 = 1$. This condition when applied to Table 5-2 means that for every m-tuple $(b_i, c_j, \ldots, d_k)$, we can always find the next m-tuple $(b_{i+1}, c_j, \ldots, d_k)$. Hence, when n_1, n_2, ... and n_m are relatively prime to one another, all the m-tuples $(b_i, c_j, \ldots, d_k)$ for $0 \leq i \leq n_1 - 1$, $0 \leq j \leq n_2 - 1$, ... and $0 \leq k \leq n_m - 1$ will appear in Table 5-2. The situation is not so in the conditional non-prime case. First, we need the following

Lemma 5-2 :

Let $b_0 b_1 \ldots b_{n_1-1} \ldots$, $c_0 c_1 \ldots c_{n_2-1} \ldots$, ... and $d_0 d_1 \ldots d_{n_m-1} \ldots$ be a set of sequences of periods n_1, n_2, ... and n_m, respectively. Let $k = \text{LCM}\{n_2, \ldots, n_m\}$ and $k < \text{LCM}\{n_1, n_2, \ldots, n_m\}$. Let the sequences be arranged in the following array.

b_0	c_0	. . .	d_0
b_1	c_1	. . .	d_1
.	.	. . .	.
.	.	. . .	.
.	.	. . .	.
b_{n_1-1}	.	. . .	.
.	.	. . .	.
.	c_{n_2-1}	. . .	.
.	.	. . .	.
.	.	. . .	d_{n_m-1}
.	.	. . .	.

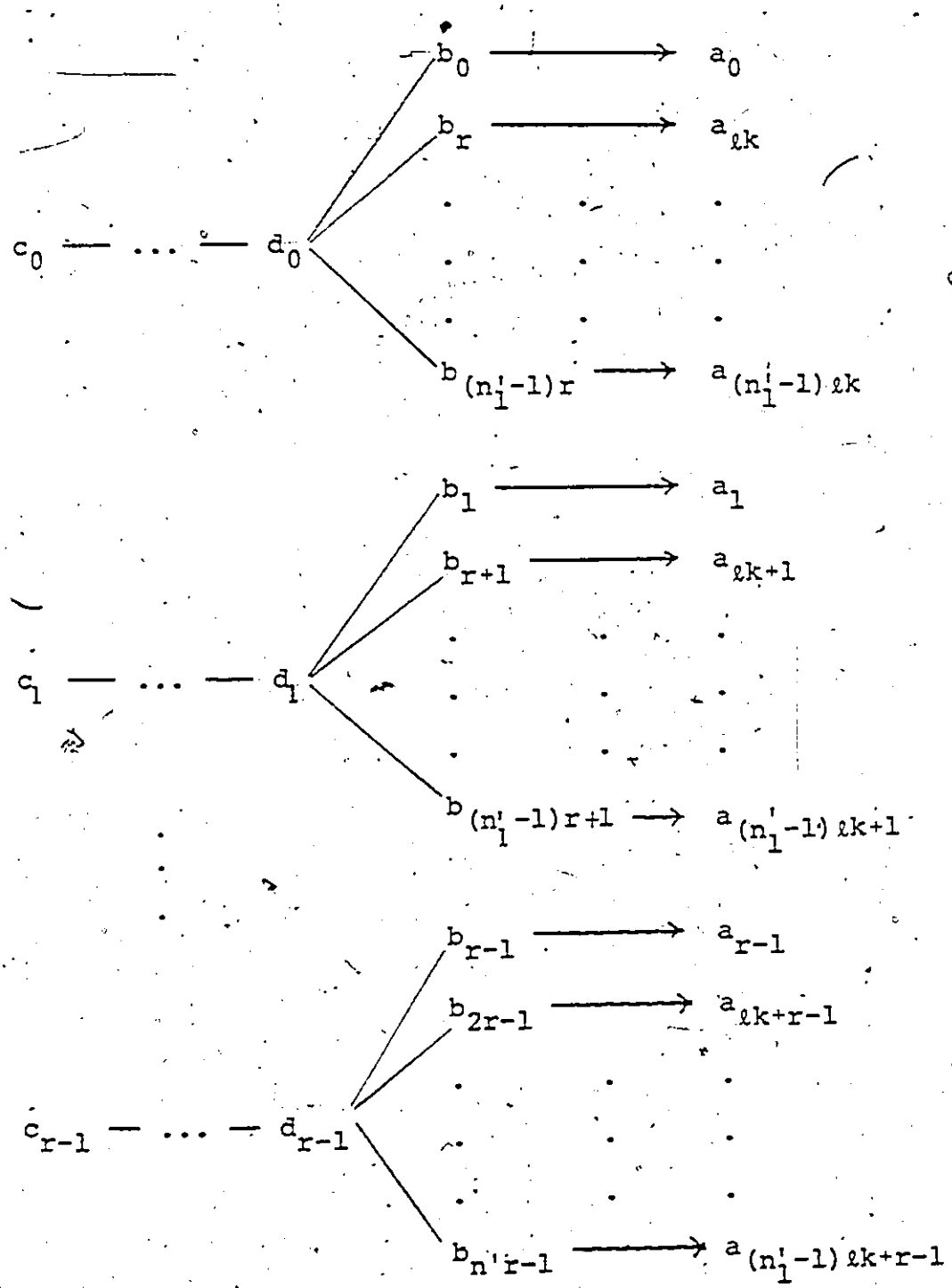
If GCD $\{ k, n_1 \} = r > 1$, then the row of the m-tuple $(b_1, c_0, \ldots, d_0)$ will not appear in the array, and the least i such that the row of the m-tuple $(b_i, c_0, \ldots, d_0)$ will appear in the array is r, and the first row of the m-tuple $(b_r, c_0, \ldots, d_0)$ will appear in the position of the ℓk th row, where ℓ is the least integer such that $\ell k - j n_1 = r$.

Proof:

As shown in Lemma 5-1, the first row of the m-tuple $(b_1, c_0, \ldots, d_0)$ if it exists will appear in the position of the ℓk th row, where ℓ is the least integer such that $\ell k - j n_1 = 1$. Now, since GCD $\{ k, n_1 \} = r$, let $k = rk'$ and $n_1 = r n_1'$, then $\ell k - j n_1 = r (\ell k' - j n_1') = 1$ has no solution for any integers ℓ and j. Hence, the row of the m-tuple $(b_1, c_0, \ldots, d_0)$ will not appear in the array. Similarly, $\ell k - j n_1 = v$ has no solution for any integers ℓ and j, if $1 < v < r$. But $\ell k - j n_1 = r (\ell k' - j n_1') = r$ implies $\ell k' - j n_1' = 1$, where k' and n_1' are relatively prime, and there always exist some integers ℓ and j such that $\ell k' - j n_1' = 1$ or $\ell k - j n_1 = r$. Therefore, the least i such that the row of the m-tuple $(b_i, c_0, \ldots, d_0)$ will appear in the array is r, and the first row of the m-tuple $(b_r, c_0, \ldots, d_0)$ will appear in the position of the ℓk th row of the array, where ℓ is the least integer such that $\ell k - j n_1 = r$.

Q.E.D.

Let us consider the partition procedure studied in Section 5-3 again. If $n = \text{LCM}\{n_1, n_2, \ldots, n_m\}$, $k = \text{LCM}\{n_2, \ldots, n_m\} < n$ and $n_1 = n_1' r$, where $r = \text{GCD}\{k, n_1\}$, from Lemma 5-2, we have



By the same reasoning of the previous partition procedure, we can partition the set $\{ b_0, b_1, \ldots, b_{n_1-1} \}$. But, this time, instead of partitioning the set $\{ b_0, b_1, \ldots, b_{n_1-1} \}$, we partition the set of subsets $\{ b_0, b_r, \ldots, b_{(n_1'-1)r} \}$, $\{ b_1, b_{r+1}, \ldots, b_{(n_1'-1)r+1} \}$, ... and $\{ b_{r-1}, b_{2r-1}, \ldots, b_{n_1'r-1} \}$. A partition pair of the set $\{ b_0, b_1, \ldots, b_{n_1-1} \}$ can be obtained from the set of partition pairs of the subsets by choosing one group from each partition pair of the subsets and grouping these groups together to form a group of the partition pair of the set $\{ b_0, b_1, \ldots, b_{n_1-1} \}$. There are totally 2^r possible assignments to obtain a partition pair of the set $\{ b_0, b_1, \ldots, b_{n_1-1} \}$. Among the 2^r possible assignments, an assignment should be chosen such that the resulting sequence has a period n_1.

The partition procedure can also be described by an array analysis as we did in the previous section. With the condition stated above, an array obtained will be split into r sub-arrays. Corresponding to the subset $\{ b_0, b_r, \ldots, b_{(n_1'-1)r} \}$ is the following sub-array,

$$
\begin{array}{cccc}
a_0 & a_{\ell k} & \cdots & a_{(n_1'-1)\ell k} \\
a_{(n_1'-1)\ell k+r} & a_r & \cdots & a_{(n_1'-2)\ell k+r} \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & a_{(n/n_1-1)r} & \cdots & \cdots
\end{array}
;
$$

corresponding to the subset $\{ b_1, b_{r+1}, \cdots, b_{(n_1'-1)r+1} \}$ is the following sub-array

$$\begin{array}{cccc} a_1 & a_{\ell k+1} & \cdots & a_{(n_1'-1)\ell k+1} \\ a_{(n_1'-1)\ell k+r+1} & a_{r+1} & \cdots & a_{(n_1'-2)\ell k+r+1} \\ \cdots & a_{(n/n_1-1)r+1} & \cdots & \end{array} ;$$

corresponding to the subset $b_{r-1}, b_{2r-1}, \cdots, b_{n_1'r-1}$ is the following sub-array,

$$\begin{array}{cccc} a_{r-1} & a_{\ell k+r-1} & \cdots & a_{(n_1'-1)\ell k+r-1} \\ a_{(n_1'-1)\ell k+2r-1} & a_{2r-1} & \cdots & a_{(n_1'-2)\ell k+2r-1} \\ \cdots & a_{(n/n_1-1)r+r-1} & \cdots & \end{array}$$

Each subarray has n/n_1 rows and n_1/r columns, and hence has an area of n/r. The total elements of the r subarrays is n again, and the r subarrays contain all the digits in one period of the given sequence.

Similarly, we can build other arrays, each of the arrays built will be split into several subarrays if the number n_i and $k = \text{LCM } \{n_j\}$, $1 \le j \le m$ and $j \ne i$, are not relatively prime.

From Theorem 5-3, we have the following

Corollary 5-3-1 :

A sequence $(a) = a_0 a_1 \ldots a_{n-1}$ of period n has a non-trivial non-prime decomposition with component sequences of periods $n_1, n_2, \ldots$ and n_m such that $n = \text{LCM}\{n_1, n_2, \ldots, n_m\}$ and any m - 1 integers of the m integers $n_1, n_2, \ldots$ and n_m have a LCM less than n, if and only if the m arrays of $n/n_1, n/n_2, \ldots$ and n/n_m rows and $n_1, n_2, \ldots$ and n_m columns, repsectively, built above satisfy the condition that for every array or for every subarray of an array if it splits, all the rows, except those rows of the same symbol, are either the same or complementary to one another.

From the previous discussions, a conditional nontrivial non-prime sequence decomposition procedure can be outlined as follows.

1. Find a set of m integers $n_1, n_2, \ldots$ and n_m such that $n = \text{LCM}\{n_1, n_2, \ldots, n_m\}$, $n_1 + n_2 + \ldots + n_m < n$ and any m - 1 integers of the m integers have a LCM less than n, where n is the period of the given sequence. Let i = 1.
2. Find $k_i = \text{LCM}\{n_j\}$, $1 \leq j \leq m$ and $j \neq i$. Find $r_i = \text{GCD}\{k_i, n_i\}$ and $\ell_i = \ell k_i$, where ℓ is the least integer such that $\ell k_i - j n_i = r_i$. Build an array of n/n_i rows and n_i columns as shown in Corollary 5-3-1. If $r_i > 1$ the array built will be split into r_i sub-

arrays. If the array doesn't satisfy the condition of Corollary 5-3-1, then there is no solution with respect to the combination $(n_1, n_2, \cdots, n_m)$ of Step 1; otherwise, get a partition pair. In case $r_i > 1$, find the partition pairs corresponding to all the subarrays, then get a partition pair by choosing a suitable assignment such that the resulting sequence will have a period n_i.

3. Increase i by 1. If $i \leq m$, go to Step 2.

4. Assign 0 and 1 to all the partition pairs, find the component sequences and the function, and get a solution. Other solutions can be obtained by assigning different assignments to the partition pairs.

Example 5-2 :

Consider the following sequence $\hat{a}$ of period 60,

$(\hat{a})$ = 101011100011001111111010101001

1100111011111010100011011111011.

Find a combination $(n_1, n_2, n_3) = (4, 5, 6)$. Since $LCM\{4, 5, 6\} = 60$, $4 + 5 + 6 = 15 < 60$, $LCM\{4, 5\} = 20 < 60$, $LCM\{5, 6\} = 30 < 60$ and $LCM\{4, 6\} = 12 < 60$, the combination (4, 5, 6) satisfies all the conditions we want.

For i = 1, find $k_1 = LCM\{5, 6\} = 30$ and $r_1 = GCD\{30, 4\} = 2$. To find ℓ such that $30\ell - 4j = 2$,

we have $\ell = 1$ and $j = 7$, hence $\ell_1 = 30\ell = 30$ and we can build an array of 15 rows and 4 columns as follows. Since $r_1 = 2$, the array splits into 2 subarrays.

b_0	b_2	b_1	b_3
1	1	0	1
0	1	0	0
1	1	1	1
.	.	.	.
.	.	.	.
.	.	.	.
0	1	1	1

Since the array satisfies the condition of Corollary 5-3-1, we obtain 2 partition pairs corresponding to the 2 subarrays as follows, $\{ b_0 \}$ and $\{ b_2 \}$, $\{ b_1 \}$ and $\{ b_3 \}$. Assign $\{ b_0 \}$ and $\{ b_3 \}$ to the same group, finally we obtain a partition pair, $\{ b_0, b_3 \}$ and $\{ b_1, b_2 \}$.

For $i = 2$, find $k_2 = \text{LCM } \{ 4, 6 \} = 12$ and $r_2 = \text{GCD } \{12, 5 \} = 1$. To find ℓ sucht that $12\ell - 5j = 1$, we have $\ell = 3$ and $j = 7$, hence $\ell_2 = 12\ell = 36$ and we can build an array of 12 rows and 5 columns as follows

c_0	c_1	c_2	c_3	c_4
1	1	0	0	1
0	0	0	0	0
1	1	1	1	1

.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
1	1	0	0	1

Since the array satisfies the condition of Corollary 5-3-1, we obtain a partition pair, $\{ c_0, c_1, c_4 \}$ and $\{ c_2, c_3 \}$.

For $i = 3$, find $k_3 = \text{LCM} \{ 4, 5 \} = 20$ and $r_3 = \text{GCD} \{ 20, 6 \} = 2$. To find ℓ such that $20\ell - 6j = 2$, we have $\ell = 1$ and $j = 3$, hence $\ell_3 = 20\ell = 20$ and we can build an array of 10 rows and 6 columns as follows. Since $r_3 = 2$, the array splits into 2 subarrays.

d_0	d_2	d_4	d_1	d_3	d_5
1	1	1	0	0	1
1	1	1	0	0	0
1	1	1	0	0	1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	1	1	1

Since the array satisfies the condition of Corollary 5-3-1, we obtain 2 partition pairs corresponding to the 2 subarrays as follows, $\{ d_0, d_2, d_4 \}$ and $\emptyset$, $\{ d_1, d_3 \}$ and $\{ d_5 \}$. Assign $\{ d_0, d_2, d_4 \}$ and $\{ d_5 \}$ to the same

group, finally, we obtain a partition pair, $\{ d_0, d_2, d_4, d_5 \}$ and $\{ d_1, d_3 \}$.

Assign 1 to $\{ b_0, b_3 \}$, $\{ c_0, c_1, c_4 \}$ and $\{ d_0, d_2, d_4, d_5 \}$; and 0 to $\{ b_1, b_2 \}$, $\{ c_2, c_3 \}$ and $\{ d_1, d_3 \}$, then three component sequences $(\hat{b}) = 1001$, $(\hat{c}) = 11001$ and $(\hat{d}) = 101011$ can be obtained. From the assignment, we can set up a truth table of the function f as shown in Table 5-4. From Table 5-4, we can easily obtain the function $f = x_1x_2 + \bar{x}_1x_3$. Therefore, the sequence $\hat{a}$ has a non-prime decomposition $\hat{a} = \hat{b} \cdot \hat{c} + \bar{\hat{b}} \cdot \hat{d}$.

Table 5-4.

Truth table of function f of Example 5-2.

x_1	x_2	x_3	f
1	1	1	1
0	1	0	0
0	0	1	1
1	0	0	0
0	1	1	1
1	0	1	0
0	0	0	0
1	1	0	1

Find all the combinations $(n_1, n_2, \ldots, n_m)$ of Step 1 of the previous procedure, we can find all the possible conditional non-prime decompositions of a sequence. However, the following combinational problems remain to be answered. Given an integer n, how many combinations $(n_1, n_2, \ldots, n_m)$ which satisfy the conditions of Step 1 of the previous procedure are there? Is there any algorithm which can be used for finding all the combinations mentioned above? In the Step 1 of the previous procedure, when the integer n is not very large, we can easily find a combination $(n_1, n_2, \ldots, n_m)$ by cut and try. Once a combination is obtained, the procedure will test whether a decomposition with respect to the combination obtained exists or not, and if it exists, find the solutions.

5-5. Logic Function Sequence Decomposition Machines

In this section, we would like to introduce a machine for both prime sequence decompositions and conditional non-prime sequence decompositions. The machine will consist of a number of shift registers and some control blocks. Actually the machine built is an implementation of the previous decomposition procedures.

5-5-1. Prime decomposition machines

A logic-function-sequence-decomposition-machine for prime decompositions has a structure block diagram shown

in Fig. 5-I. The operation mechanisms of this machine are described as follows.

1. Find the prime factors of n,

$$n = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m} = n_1 n_2 \cdots n_m$$

Set I = 1.

2. If $I \leq m$, do the following; otherwise, go to Step 11.

Load shift register SRA with a period of the given sequence.

Transfer the contents of shift register SRA to shift register SRB, i.e., SRA(i) $\longrightarrow$ SRB(i) for all i, $0 \leq i \leq n - 1$.

Calculate $k_I = n/n_I$ and find ℓ_I such that $\ell_I k_I - j_I n_I = 1$.

3. Transfer the contents of the first memory element of shift register SRB to the first memory element of the shift register SRC of length n_I, i.e., SRB(0) $\longrightarrow$ SRC(0).

Set J = 1, M = 0, N = 1 and switch A OFF.

4. If $N \leq n/n_I$, do the following; otherwise, go to Step 10.

Cyclic shift left the contents of SRB $\ell_I k_I$ times, i.e.,

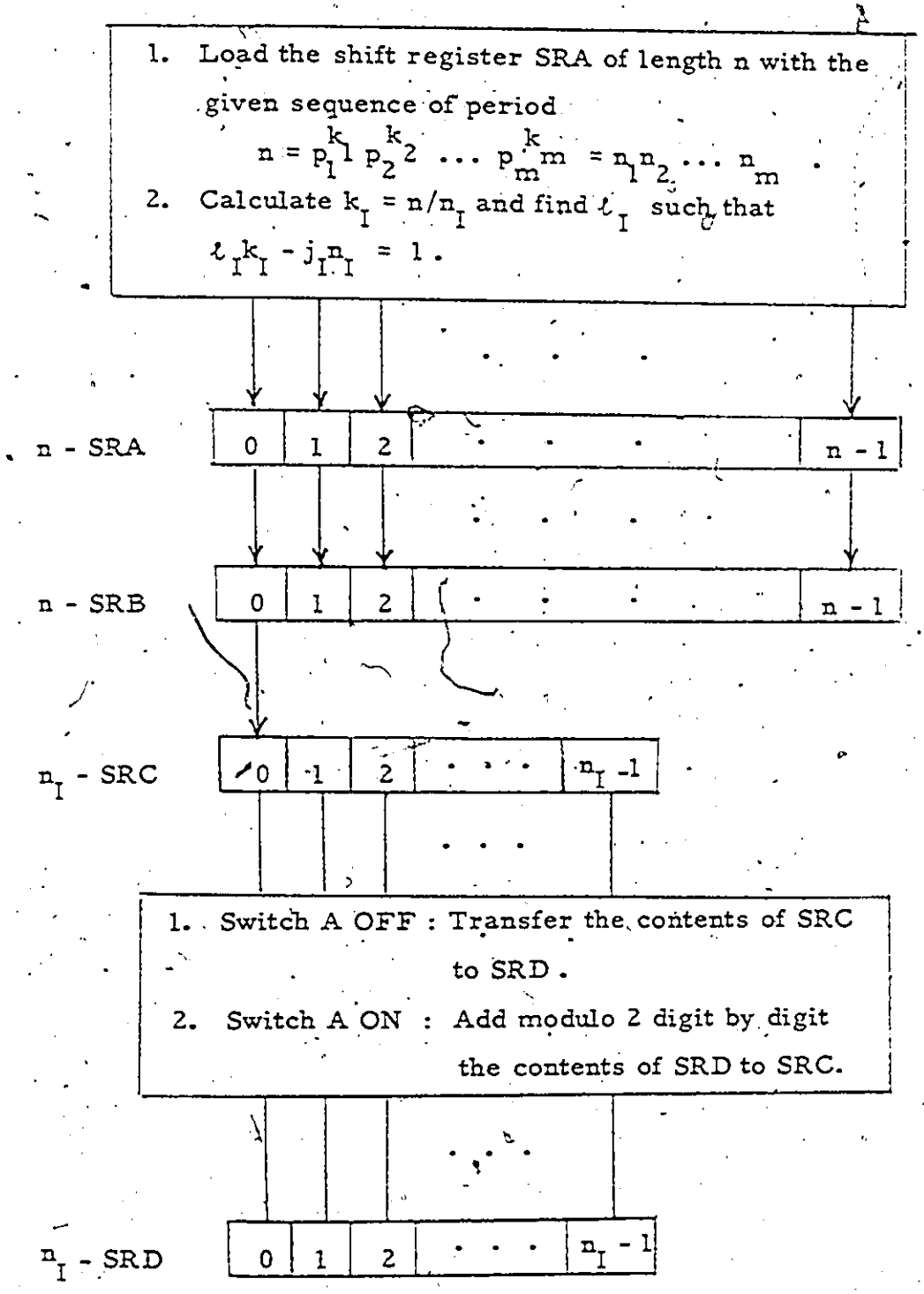


Fig. 5-1. A logic function sequence decomposition machine for prime decomposition.

$SRB(\ell_I k_I) \longrightarrow SRB(0)$

$SRB(\ell_I k_I + 1) \longrightarrow SRB(1)$

.

.

.

$SRB(\ell_I k_I - 1) \longrightarrow SRB(n - 1)$

Cyclic shift left the contents of SRC once, i.e.,

$SRC(1) \longrightarrow SRC(0)$

$SRC(2) \longrightarrow SRC(1)$

.

.

.

$SRC(0) \longrightarrow SRC(n_I - 1)$

Transfer the contents of the first memory element of SRB to the first memory element of SRC, i.e., $SRB(0) \longrightarrow SRC(0)$.

5. Increase J by 1, i.e., $J + 1 \longrightarrow J$.
   If $J \leq n_I$, go to Step 4.
6. Test the contents of SRC.
   If the contents of SRC are either 000 ... 0 or 111 ... 1, go to Step 8.
7. Calculate $P \equiv M \bmod n_I$ and cyclic shift right the contents of SRC P times. Test the condition of switch A. If

switch A is OFF, transfer the contents of SRC to shift register SRD of length n_I, i.e., SRC(i) $\longrightarrow$ SRD(i) for all i, $0 \leq i \leq n_I - 1$, set switch A ON, and go to next step; otherwise, go to Step 9.

8. Increase N by 1, i.e., $N + 1 \longrightarrow N$.

Set J = 1.

Cyclic shift left the contents of SRA once, i.e.,

SRA(1) $\longrightarrow$ SRA(0)

SRA(2) $\longrightarrow$ SRA(1)

SRA(0) $\longrightarrow$ SRA(n - 1).

Transfer the contents of SRA to SRB, i.e., SRA(i) $\longrightarrow$ SRB(i) for all i, $0 \leq i \leq n - 1$. Transfer the contents of the first memory element of SRB to the first memory element of SRC, i.e., SRB(0) $\longrightarrow$ SRC(0).

Increase M by 1, i.e., $M + 1 \longrightarrow M$, and go to Step 4.

9. Add modulo 2 digit by digit the contents of SRD to SRC, i.e.,

SRD(i) $\oplus$ SRC(i) $\longrightarrow$ SRC(i)

for all i, $0 \leq i \leq n_I - 1$.

If the results in SRC are either 000 ... 0 or 111 ... 1, increase both M and N by 1, i.e., $N + 1 \longrightarrow N$ and $M + 1 \longrightarrow M$, and go to Step 4; otherwise, no solution.

10. Get a component sequence from SRD. Increase I by 1, i.e., $I + 1 \longrightarrow I$, and go to Step 2.

11. Find the function. From the component sequences obtained, a truth table of the function can easily be set up and the function can be obtained from the truth table.

Get a solution.

In the preceding operation mechanisms, J takes care of the number of digits examined in a row of the previous array, N keeps tracking the number of rows examined, I takes care of the number of arrays examined, and M keeps tracking the number of times a row should be shift cyclically such that every row will be matched with the row of the previous array. From the sequences obtained in SRD, we can easily obtain the partition pairs. But, we can also keep the assignement matched with the sequence obtained, hence the component sequences are obtained directly from SRD without any further assignment. A flow chart of the operation mechanisms is given in Fig. 5-2.

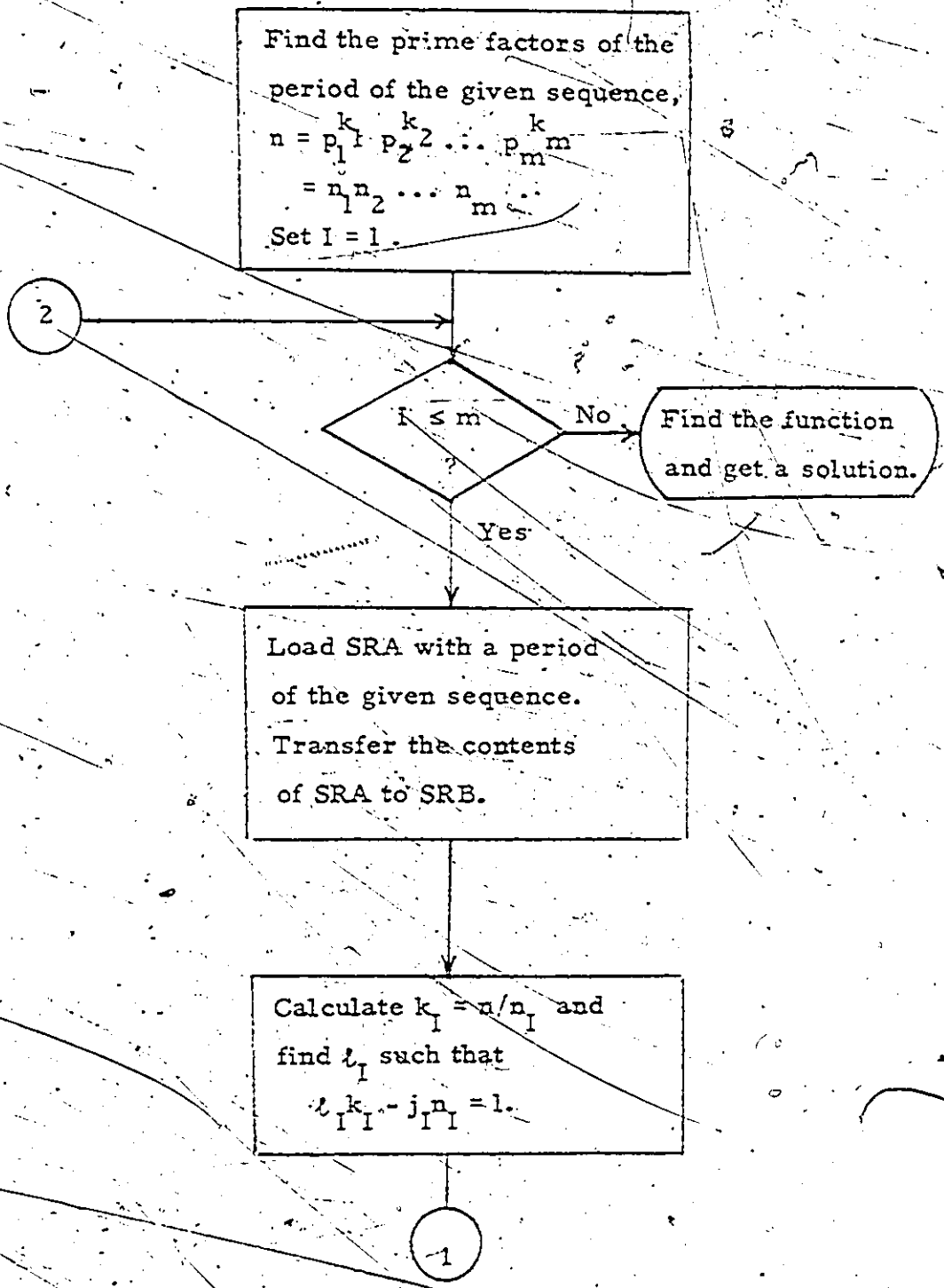


Fig. 5-2. A flow chart of the operation mechanisms of a prime decomposition machine.

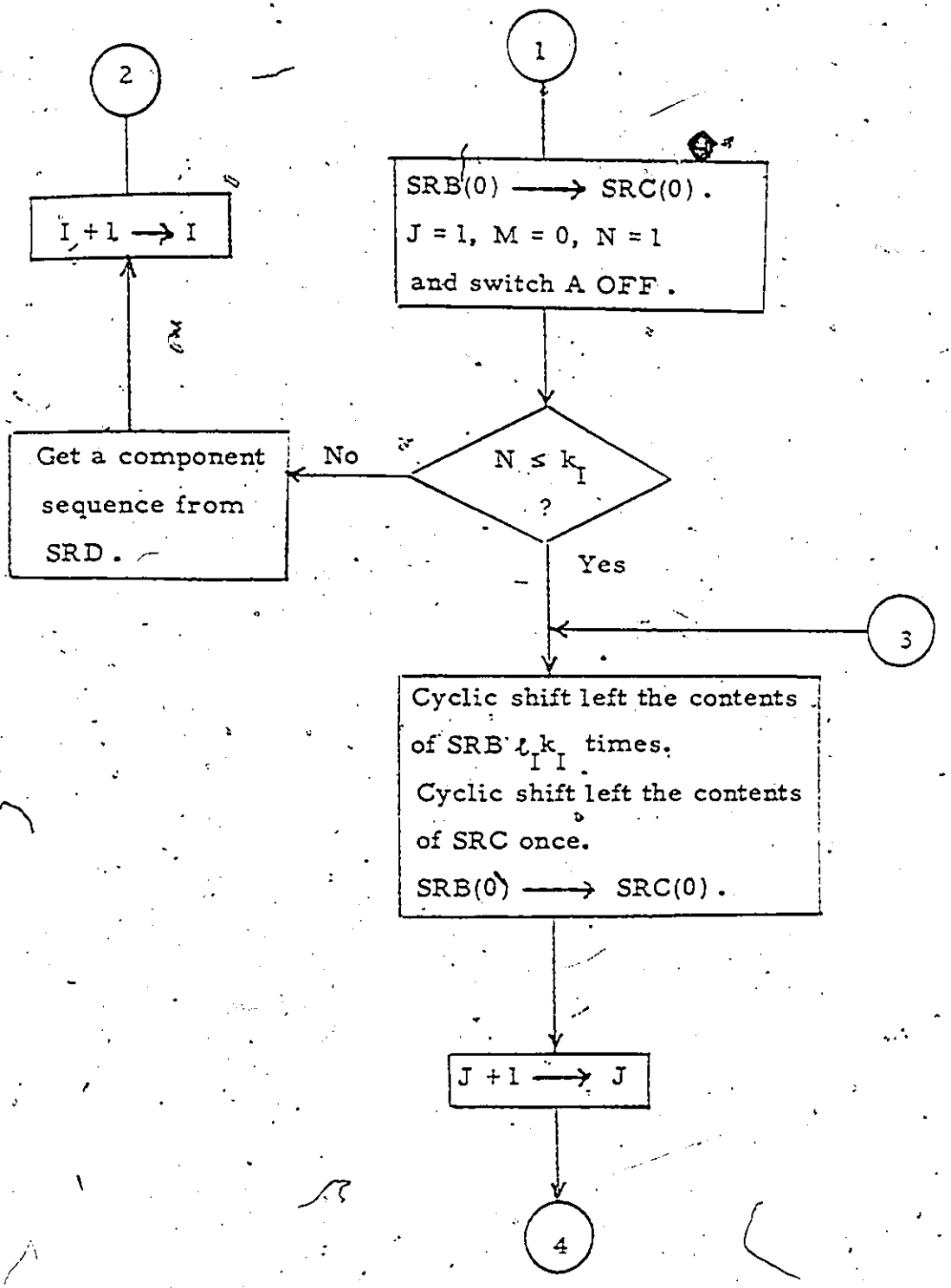


Fig. 5-2. (continued).

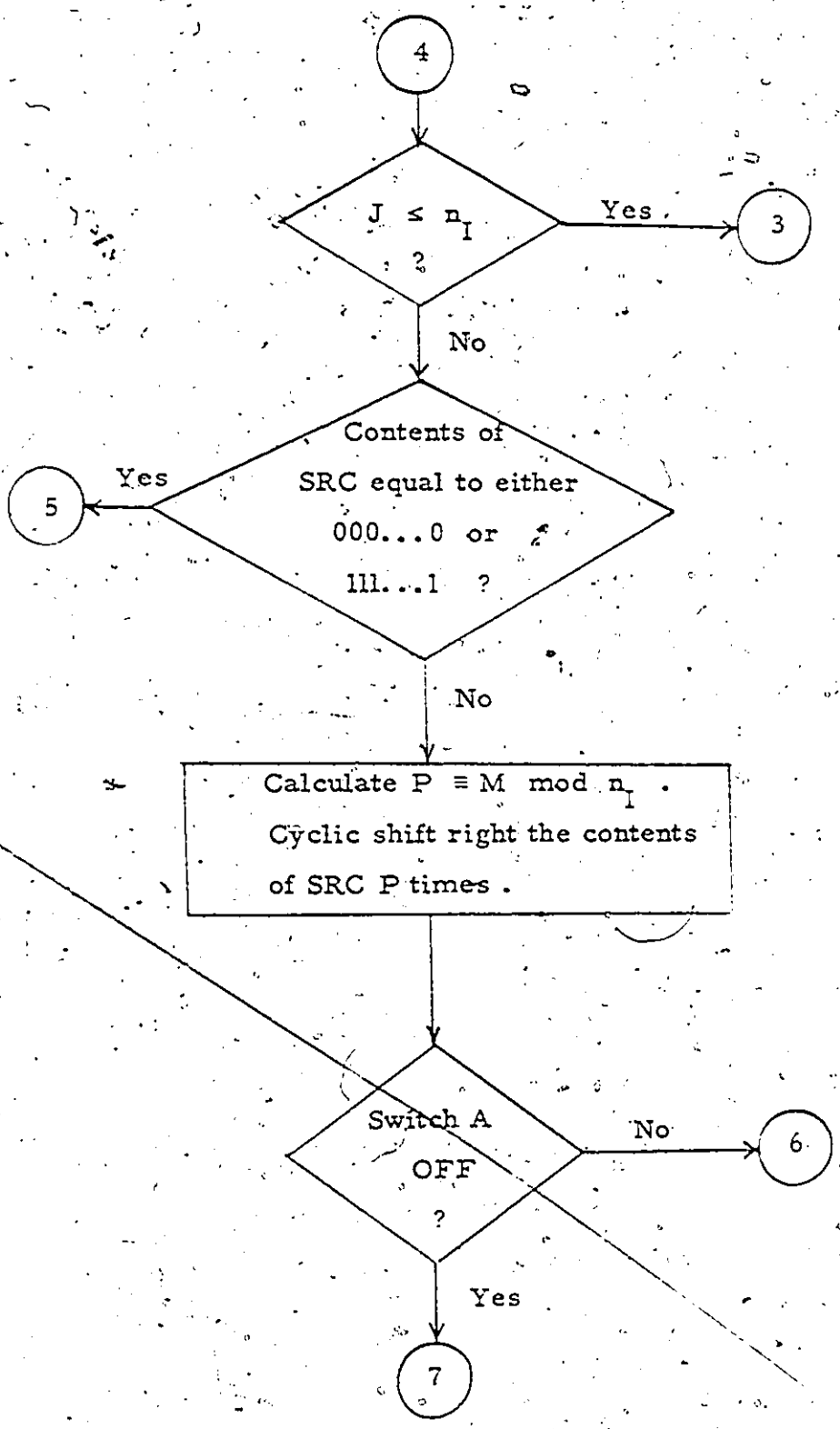


Fig. 5-2. (continued).

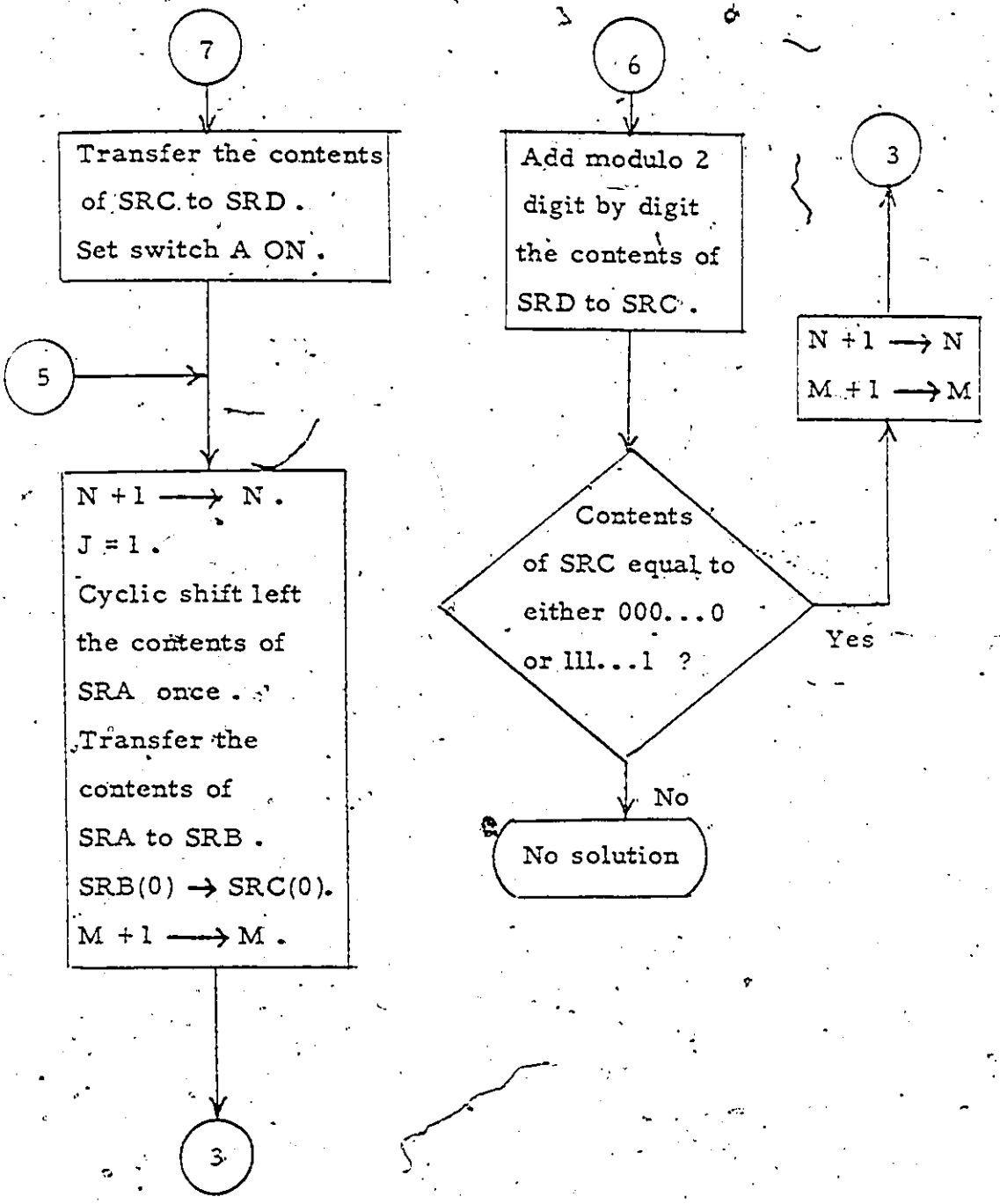


Fig. 5-2. (continued).

5-5-2. Number of operations involved in decomposing a sequence using a prime decomposition machine.

In the following, we would like to count the number of operations involved in decomposing a sequence using the prime decomposition machine of Section 5-5-1. From which we can estimate the times involved in decomposing a sequence using that machine.

Suppose we have a sequence of period $n = p_1^{k_1} p_2^{k_2} \ldots p_m^{k_m} = n_1 n_2 \ldots n_m$. Since each time we obtain a component sequence we must reload the SRA again, the total number of times loading the SRA with a period of the given sequence will be m. The number of times to transfer the contents of SRA to SRB is n for getting a component sequence, hence the total number of times to transfer the contents of SRA to SRB in decomposing a sequence is mn. The number of times to transfer the contents of the first memory element of SRB to the first memory element of SRC is n for getting a component sequence, hence the total number of times to transfer SRB(0) to SRC(0) is mn. The number of times to transfer the contents of SRC to SRD is one for getting a component sequence, hence the total number of times to transfer the contents of SRC to SRD is m. The number of times of cyclic shift of SRA for getting a component sequence is n, hence the total number of times of cyclic shift

of SRA is mn. Similarly, the total number of times of cyclic shift of SRB is mn. The number of times of cyclic shift of SRC for getting a component sequence is $^{n}/_{n_i}$, hence the total number of times of cyclic shift of SRC is $^{n}/_{n_1} + {}^{n}/_{n_2} + \ldots + {}^{n}/_{n_m}$. The total number of times testing the condition of switch A is $^{n}/_{n_1} + {}^{n}/_{n_2} + \ldots + {}^{n}/_{n_m}$. The total number of times testing the contents of SRC to see whther they are equal to 000 ... 0 or 111 ... 1 or not is $2(^{n}/_{n_1} + {}^{n}/_{n_2} + \ldots + {}^{n}/_{n_m})$. The total number of times adding modulo 2 the contents of SRD to SRC depends on the sequence given. Taking average, it will be equal to $\frac{1}{2}(^{n}/_{n_1} + {}^{n}/_{n_2} + \ldots + {}^{n}/_{n_m})$. The total number of times testing whether the contents of SRC are equal to or complementary to the contents of SRD is $2 \cdot 1/2(n/n_1 + n/n_2 + \ldots + n/n_m)$ in average.

Now, let T_ℓ denote the unit loading time, T_t denote the unit time to transfer the contents of one shift register to the other shift register, T_c denote the unit time shifting cyclically the contents of a shift register, T_a denote the unit time adding modulo 2 the contents of one shift register to the contents of the other shift register, T_s denote the unit time testing the condition of switch A, T_k denote the unit time testing the contents of a shift register, let $N = {}^{n}/_{n_1} + {}^{n}/_{n_2} + \ldots + {}^{n}/_{n_m}$, then the average time needed to decompose a sequence of period $n = n_1 n_2 \ldots n_m$ is $t_\ell(m) + T_t\,(mn + mn + m) + T_c\,(mn + mn + N) + T_s\,(N) + T\,(\frac{1}{2}N) + T_k\,(2N + N) = mT_\ell + (2n + 1)mT_t + (2mn + N)\,T_c + NT_s + \frac{1}{2}NT_a + 3NT_k$.

5-5-3. Conditional non-prime decomposition machines

A little modification of the previous prime decomposition machine will give us a conditional non-prime decomposition machine. The operations of the conditional non-prime decomposition machine start with a combination $(n_1, n_2, \ldots, n_m)$ which satisfies the following conditions, $n = \text{LCM}\{ n_1, n_2, \ldots, n_m \}$ where n is the period of a given sequence, $n_1 + n_2 + \ldots + n_m < n$ and any m - 1 integers of the m integers n_1, n_2, ... and n_m have a LCM less than n. In the following discuss, we will assume that we can find such a combination. A structure block diagram of the modified conditional non-prime decomposition machine is shown in Fig. 5-3. The operation mechanisms of this machine are described as follows.

1. Find a combination $(n_1, n_2, \ldots, n_m)$ such that $n = \text{LCM}\{ n_1, n_2, \ldots, n_m \}$, $n_1 + n_2 + \ldots + n_m < n$ and any m - 1 integers of the m integers n_1, n_2, ... and n_m have a LCM less than n.
   Set I = 1.
2. If $I \leq m$, do the following; otherwise, go to Step 13.
   Load the shift register SRA with a period of the given sequence. Transfer the contents of SRA to SRB.
   Calculate $k_I = \text{LCM}\{ n_i \}$ for all i, $1 \leq i \leq m$ and $i \neq I$.
   Find $\text{GCD}\{ k_I, n_I \} = r_I$, $k_I = k'_I r_I$ and $n_I = n'_I r_I$.

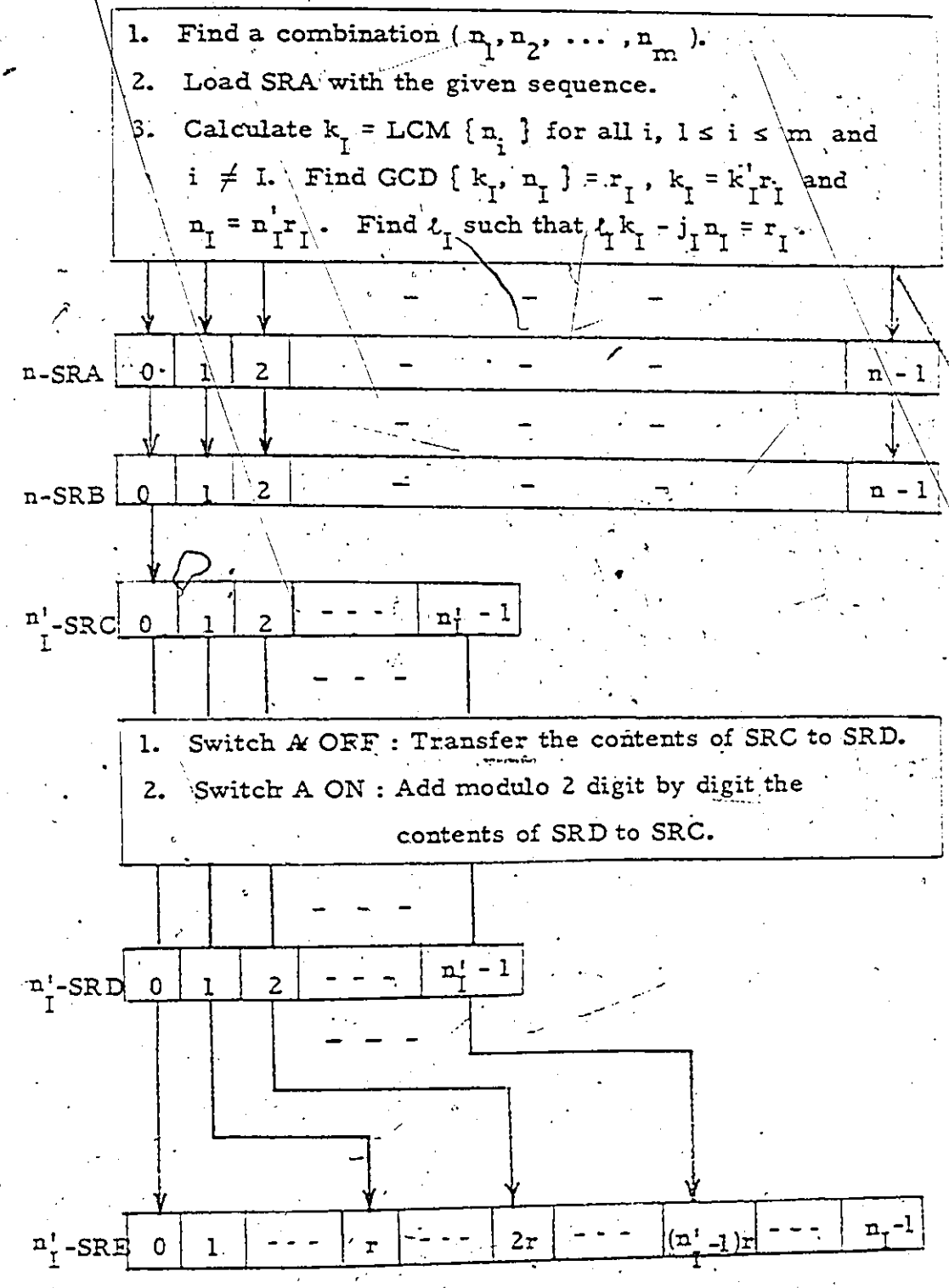


Fig. 5-3. A conditional non-prime logic function sequence decomposition machine.

Find ℓ_I such that $\ell_I k_I - j_I k_I = r_I$.

Set Q = 1.

3. SRB(0) $\longrightarrow$ SRC(0).

Set J = 1, M = 0, N = 1 and switch A OFF.

4. If N $\leq n/n_I$, do the following; otherwise go to Step 10.

Cyclic shift left the contents of SRB $\ell_I k_I$ times.

Cyclic shift left the contents of SRC once.

SRB(0) $\longrightarrow$ SRC(0).

5. J + 1 $\longrightarrow$ J.

If J $\leq n'_I$, go to Step 4.

6. Test the contents of SRC.

If the contents of SRC are either 000 ... 0 or 111 ... 1, go to Step 8.

7. Calculate P $\equiv$ M mod n'_I and cyclic shift right the contents of SRC P times.

Test the condition of switch A.

If switch A is OFF, transfer the contents of SRC to SRD, set the switch A ON and go to next step; otherwise, go to Step 9.

8. N + 1 $\longrightarrow$ N, set J = 1.

Cyclic shift left the contents of SRA ℓ_I times.

Transfer the contents of SRA to SRB.

SRB(0) $\longrightarrow$ SRC(0).

M + 1 $\longrightarrow$ M and go to Step 4.

9. Add modulo 2 digit by digit the contents of SRD to SRC.

   If the results in SRC are either 000 ... 0 or 111 ... 1, N + 1 $\longrightarrow$ N and M + 1 $\longrightarrow$ M, and go to Step 4; otherwise, no solution.

10. Transfer the contents of SRD to SRE by interleaving r memory element space apart, i.e.,

    SRD(0) $\longrightarrow$ SRE(0)

    SRD(1) $\longrightarrow$ SRE(r)

    . . .

    . . .

    . . .

    SRD(n'_I - 1) $\longrightarrow$ SRE((n'_I - 1)r)

    Q + 1 $\longrightarrow$ Q.

11. If Q $\leq r_I$, do the following; otherwise, go to next step.

    Cyclic shift left the contents of SRE once.

    Load SRA with the given sequence.

    Cyclic shift left the contents of SRA Q times.

    Transfer the contents of SRA to SRB.

    Go to Step 3.

12. If the contents of SRE have a period less than n_I, complement the contents of

SRE(0), SRE(r), ... and SRE($(n'_I - 1)r$);

or/and SRE(1), SRE(r + 1), ... and SRE($(n'_I - 1)r + 1$);

. . .

or/and SRE(r - 1), SRE(2r - 1), ... and SRE($n'_I r - 1$),

until the contents of SRE have no period less than n_I is found.

Get a component sequence from SRE.

I + 1 $\longrightarrow$ I and go to Step 2.

13. Find the function from the component sequences obtained and get a solution.

In the preceding operation mechanisms, J, N, M and I do the same functions as they did in the previous operation mechanisms. Q keeps tracking the number of subarrays examined. In Step 12, complement the contents of certain sets of memory elements of SRE is equivalent to reassignment the partition pairs obtained from the subarrays of the previous procedure. With respect to a combination $(n_1, n_2, \ldots, n_m)$, the number of operations and the computation time involved in decomposing a sequence using a conditional non-prime decomposition machine can be similarly obtained by using the same analysis of Section 5-5-2.

5-6. Discussions

5-6-1. Logic function sequence decomposition parallel machines

The previously studied decomposition procedures mainly check a set of arrays of the same area which have been constructed by a special algorithm. The procedures check the arrays one by one. From a statistical point of view, there is no difference which array should be checked first, because all the arrays have the same area. The arrays can also be checked simultaneously so that decomposition time can be saved. In the decomposition using the logic-function-sequence-decomposition-machine, this means we need a prallel machine. A parallel machine can be built by combining several previously studied machines together, which we call the component machines. In the decomposition procedure, all the component machines will operate simultaneously so that the component sequences can be obtained simultaneously.

5-6-2. The ratio of prime decomposable sequences

In the following, we try to find the ratio of the number of binary sequences of finite periods which are prime decomposable for the general interesting.

The total number of possible binary sequences of period $n = p^k$, where p is a prime number and k is an integer, is $2^n - 2^{p^{k-1}} = 2^{p^{k-1}} (2^{(p-1)p^{k-1}} - 1)$. The total

number of binary sequences of period $n = p_1^{k_1}p_2^{k_2} \ldots p_m^{k_m} = n_1 n_2 \ldots n_m$ which are prime decomposable is

$$\prod_{i=1}^{m} 2^{p_i^{k_i-1}(2^{(p_i-1)}p_i^{k_i-1}-1)} = \prod_{i=1}^{m} 2^{n_i} = 2^{\sum_{i=1}^{m} n_i}$$

The total number of possible binary sequences of period $n = p_1^{k_1}p_2^{k_2} \ldots p_m^{k_m} = n_1 n_2 \ldots n_m$ is

$$2^n - \sum_{i=1}^{m} 2^{p_i^{k_i-1}} 2^{n/n_i} \approx 2^n.$$

Hence, the ratio of the number of binary sequences of period $n = n_1 n_2 \ldots n_m$ which are prime decomposable is approximately

$$2^{(\sum_{i=1}^{m} n_i - \prod_{i=1}^{m} n_i)}$$

5-6-3. A way of sequence storing

Let A be the set of sequences which are not prime decomposable and B be the set of sequences which are prime decomposable. Then the sequences can be stored in the following way. If a sequence belongs to A, store the whole sequence of length n in storage SA; on the other hand, if a sequence belongs to B, store the component sequences in storage SB. To obtain a sequence, if it is in storage SA, we can obtain it directly from the storage; if it is in storage SB, we should pass the component sequences through a recovering procedure to recover the sequence. A recovering machine shown in Fig. 5-4 can be used for sequence recovering purpose.

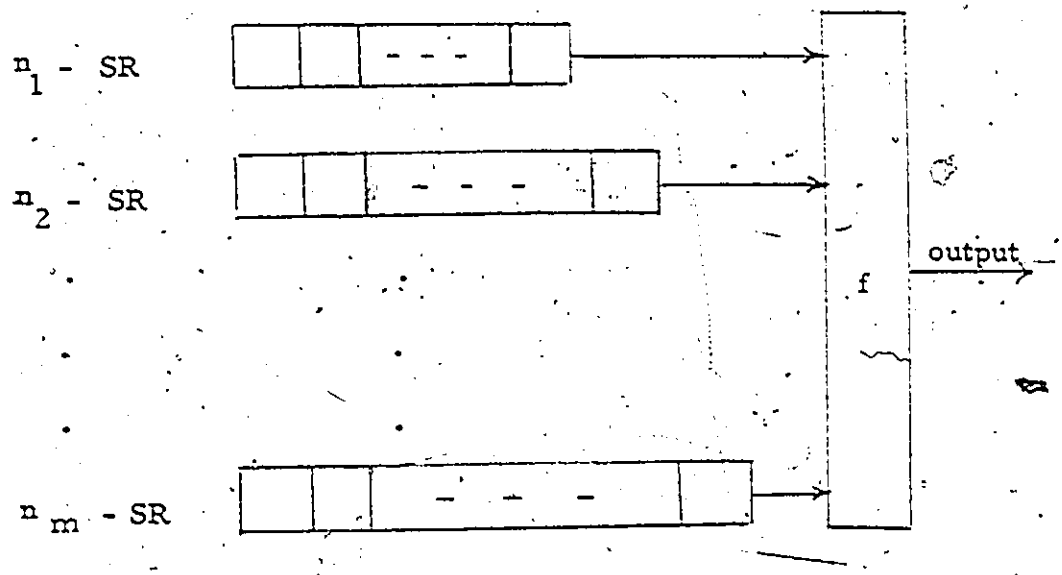


Fig. 5-4. A recovering machine.

The operations of the recovering machine of Fig. 5-4 can be described as follows. The component sequences of length n_1, n_2, ... and n_m are loaded in the shift registers n_1-SR, n_2-SR, and n_m-SR, respectively. The original sequence can be recovered from the output of the logic function, which is the logic function obtained from the decomposition procedure, by shifting all the SR's cyclically.

5-6-4. General non-prime decomposition

The conditional non-prime decomposition procedure studied in Section 5-4 is constrained by the following condition. For a combination $(n_1, n_2, \ldots, n_m)$, we must have $n_1 + n_2 + \ldots + n_m < n$ and $n = \text{LCM}\{n_1, n_2, \ldots, n_m\}$, and any $m-1$ integers of the m integers n_1, n_2, ... and n_m must have LCM less than n. If the LCM of some

m - 1 integers of the m integers equals to n, it is still possible to have a non-prime sequence decomposition existed. But the previous decomposition procedure will fail to find such a decomposition. In this case, the corresponding array constructed will have n_i one column subarrays and the partition procedure will not be able to find a partition pair of the corresponding component sequence. For example, the following sequence of period 60,

$(\hat{a})$ = 100110011111100100011001100000
01111111001100111011000000011001,

have a non-prime decomposition $\hat{a} = \hat{b}\hat{c} + \bar{\hat{c}}\hat{d}$, where $(\hat{b})$ = 1001, $(\hat{c})$ = 111001 and $(\hat{a})$ = 100110001111000, but LCM { 4, 15 } = 60, the previous decomposition procedure will not be able to find a partition pair corresponding to sequence $\hat{c}$. A general sequence decomposition procedure to find a decomposition of this kind remains to be answered.

CHAPTER 6

CONCLUDING REMARKS

This thesis has mainly dealt with the structure theory of nonsingular recurring sequences. Several aspects of nonsingular recurring sequences have been studied, namely, generation methods, decompositions and some properties of nonsingular recurring sequences. In Chapter 3, the realization problem of finding FSR's with input sequences for generation of specified sequences of finite periods has been examined and a solution has been obtained. In Chapter 4, the problem to find a modulo-2-sum decomposition of linear binary sequences of finite periods has been examined and a solution has been obtained. The generalization over GF(q) follows the parallel lines. In Chapter 5, a more general sequence decomposition problem, decomposing binary sequences of finite periods into logic functions of several binary sequences of finite periods, has been examined and a solution has been provided for prime decompositions and conditional non-prime decompositions. A logic-function-sequence-decomposition-machine has also been introduced for both prime decompositions and conditional non-prime decompositions.

The previously studied conditional non-prime decomposition has to satisfy the following condition. For a combination $(n_1, n_2, \ldots, n_m)$, any $m - 1$ integers of the

m integers n_1, n_2, ... and n_m must have a LCM less than n. Without this condition, a nontrivial decomposition is still possible. The general non-prime sequence decomposition problem remains to be answered. In Chapter 4, the method to find a modulo-2-sum decomposition of binary sequences of finite periods is passed on to the LFSR realization of the given sequences which gives fractional polynomial representation of the sequences. The partial fraction expansion of the fractional polynomials is equivalent to the decomposition of the LFSR's which generate the sequences. Similarly, the logic function sequence decomposition problem can be passed on to the ASM realization of the given sequences. By Theorem 5-1, we have to find an ASM decomposition of the corresponding ASM, which generates the sequence, into k sub-ASM's with a state assignment such that the output function is k-divisible. This may solve the general non-prime sequence decomposition problem. However, how to find a state assignment such that the output function is k-divisible is a new created problem which still remains to be answered.

REFERENCES

1. A.A. Albert, Fundamental Concepts of Higher Algebra, University of Chicago Press, Chicago, Illinois, 1956.

2. A.A. Albert, "On certain trinomial equations in finite fields," Annals of Mathematics, Vol. 66, No. 1, pp. 170-178, July 1957.

3. W.O. Alltop, A.V. Pratt and R.C. Burton, "Algebraic theory of flip-flop sequence generators", Info. and Control, vol. 12, pp. 193-205, 1968.

4. T.C. Bartee and P.E. Wood, "Coding for tracking radar ranging", MIT Lincoln Laboratory, Technical Report No. 318, June 1963.

5. L.D. Baumert, "Generation of specified sequences", in "Digital communications with space application" edited by S.W. Golomb, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1964.

6. E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.

7. R.H. Braasch, Sequences from Nonlinear Feedback Shift REgisters, Ph.D. dissertation, University of New Mexico, Albuquerque, 1966.

8. P.R. Bryant, F.G. Heath and R.D. Killich, "Counting with feedback shift registers by means of a jump technique", IRE Trans. on Elec. Comp., Vol. EC-11,

pp. 285-286, April 1962.

9. L.L. Campbell, "Two properties of pseudo-random sequences", IRE Trans. on Info. Theory, p. 32, March 1959.

10. M. Cohn and S. Even, "The design of shift register generators for finite sequences", IEEE Trans. on Comp., Vol. C-18, No. 9, pp. 660-662, July 1969.

11. N.G. de Bruijn, "A combinational problem", Proceeding Koninklijke Nederlandse Akademic van Wetenschappen, Vol. 49 (part 2), pp. 758-764, Sept. 1946.

12. M. Easterling, "Methods for obtaining velocity and range information for CW radars", Jel Propulsion Laboratory, Pasadena, Calif., Technical Report No. 32-657, Sept. 1964.

13. B. Elspas, "A radar system based on statistical estimation and resolution consideration", Applied Electronics Laboratory, Standford University, S STanford, Calif., Technical Report No. 361-1, Aug. 1955.

14. B. Elspas, "Theory of autonomous linear sequential networks", IRE Trans. on Circuit Theory, Vol. CT-6, pp. 45-60, March 1959.

15. G.B. Fitzpatrick, "Synthesis of binary ring counters of given periods", Association for computing Machines Journal, Vol. 7, No. 3, pp. 287-297, July 1960.

16. H. Gallarre and M.A. Harrison, "Decomposition of linear sequential machines", Math. Systems Theory, Vol. 3, pp. 246-287, 1969.

17. E.N. Gilbert, "Quasi random binary sequences", Bell Telephone Laboratories, Memorandum MM-53-1400-42, Nov. 1953.

18. A. Gill, "A theorem concerning compact and cyclic sequences", IRE Trans. on Info. Theory, Vol. IT-8, p. 255, April 1962.

19. A. Gill, Linear Sequential Circuits, McGraw-Hill, New York, 1966.

20. A. Ginzburg, Algebraic Theory of Automata, Academic Press, New York, 1968.

21. R. Gold, "Characteristic linear sequences and their coset functions", J. SIAM Appli. Math., Vol. 14, No. 5, pp. 980-985, Sept. 1966.

22. S.W. Golomb, (Editor), Digital Communications with Space Applications, Prentice-Hall, Inc., Eanglewood Cliffs, New Jersey 1964.

23. S.W. Golomb, Shift Register Sequences, Holden-Day, Inc., San Francisco, Calif., 1967.

24. I.J. Good, "Normal recurring decimals", Journal of the London Mathematical Society, Vol. 21 (part 3), pp. 167-169, 1946.

25. Z. Grodzki, "The theory of shift-registers", Info. and Control, Vol. 21, pp. 196-205, 1972.

26. Z. Grodzki, "The complexity of shift-registers", Info. and Control,, Vol. 21, pp. 206-210, 1972.

27. F.G. Heath and M.W. Gribble, "Chain codes and their electronic applications", The Institute of Electrical Engineers, Monograph No. 392M, July 1960.

28. C.C. Hsieh, Decomposition and Generation of Finite Sequences, Ph.D. dissertation, The Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, Rep. of China, 1972.

29. W.H. Kantz (Editor), Linear Sequential Swtiching Circuits -- Selected Technical Papers, Holden-Day, Inc., San Francisco, Calif., 1965.

30. A. Lempel, "On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers", IEEE Trans. on Comp., Vol. C-19, No. 12, pp. 1204-1209, Dec. 1970.

31. S. MacLane and G. Birkhoff, Algebra, The MacMillan Company, New York, 1967

32. F.J. MacWilliams, "The structure and properties of binary cyclic alphabets", Bell System Tech. J., Vol. 44, pp. 303-332, 1965.

33. K.B Magleby, "The synthesis of nonlinear feedback shift registers", Standford Electronics Laboratory, Standford, Calif., Tech. Report 6207-1, 1973.

34. D. Mandelbaum, "A comparison of linear sequentail circuits and arithmetic sequences", IEEE Trans. on Elec. Comp., Vol. EC-16, No. 2, pp. 151-157, April 1967.

35. J.C. Massey, "Shift-registers synthesis and BCH decoding", IEEE Trans. on Info. Theory, Vol. IT-15, No. 1, pp. 122-127, Jan. 1969.

36. F.J. Mowle, "An algorithm for generating stable feedback shift register of order n", J. ACM, Vol. 14, pp. 529-542, 1967.

37. F.J. Mowle, "Readily programmable procedures for the analysis of nonlinear feedback shift registers", IEEE Trans. on Comp., Vol. C-18, No. 9, pp. 824-829, Sept. 1969.

38. M. Perlman, "The decomposition of the states of a linear feedback shift register into cycles of euqal length", IEEE Trans. on Comp., Vol. C-19, No. 11, pp. 1029-1035, Nov. 1970.

39. W.W. Peterson, Error-Correcting Codes, MIT Press, Cambridge, Mass., 1961.

40. W.W. Peterson and E.J. Weldon, Jr., Error-Correcting Codes, second edition, MIT Press, Cambrdige, Mass., 1972.

41. I.S. Reed and R. Turn, "A generalization of shift register sequence generators", JACM, Vol. 16, pp. 461-473, 1969.

42. H.H. Roth, "Linear binary shift register circuits utilizing a minimum number of mod-2 adders", IEEE Trans. on Info. Theory, Vol. IT-11, No. 2, pp.215-220, April 1965.

43. W.M. Siebert, "A radar detection philosophy", IRE Trans. on Ifo. Theory, Vol. IT-2, p. 219, Sept. 1956.

44. H. Tanaka, M. Kasahara, Y. Tezuka and Y. Kasahara, "Computation over Golois field using shift registers", Info. and Control. Vol. 13, pp. 75-84, 1968.

45. R.C. Tansworth, "Random number generated by linear reucrrence modulo-two", Mathematics of Computation, Vol. 19, No. 90, pp. 201-208, April 1965.

46. M. Ward, "The arithmetical theory of linear recurring series". Trans. Amer. Math. Soc., Vol. 35, pp. 600-628, 1933.

47. M. Yoeli, "Nonlinear feedback shift registers", IBM Development Lab., Poughkeepsie, New York, Tech. Rept. TR 00.809, Sept. 1961.

48. N. Zierler, "Several binary-sequence generators", MIT Lincoln Laboratory, Cambridge, Mass., Tech. Rept. No. 95, Sept. 1955.

49. N. Zierler, "Linear recurring sequences", J. SIAM, Vol. 7, No. 1, pp. 31-48, March 1959.

VITA

Date of Birth	28 Sept. 1939
Place of Birth	Hsinchu, Taiwan, Rep. of China
Education	
i) High School	Hsinchu High School, Hsinchu, Taiwan, Rep. of China
ii) Universities	Cheng Kung University, Tainan, Taiwan, Rep. of China
	Chiao Tung University, Hsinchu, Taiwan, Rep. of China
	University of Western Ontario, London, Ontario, Canada
iii) Degrees	B. Sc. (Electrical Engineering) 1963
	M. Sc. (Electronic Engineering) 1966
	M. Sc. (Computer Science) 1970