NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report No. 32-1019

An Algorithm for the Synthesis of Binary Sequence Detectors

Marvin Perlman

GPO PRICE $ ____________

CFSTI PRICE(S) $ ____________

Hard copy (HC) 3.00

Microfiche (MF) .65

ff 653 July 65

FACILITY FORM 602

N67 14932	
(ACCESSION NUMBER)	(THRU)
18	1
(PAGES)	(CODE)
CR-80990	08
(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

December 15, 1966

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report No. 32-1019

An Algorithm for the Synthesis of Binary Sequence Detectors

Marvin Perlman

Approved by:

Karl W Linnes

K. W. Linnes, *Manager*
Space Instruments Systems Section

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

December 15, 1966

Copyright © 1966
Jet Propulsion Laboratory
California Institute of Technology

Prepared Under Contract No. NAS 7-100
National Aeronautics & Space Administration

CONTENTS

I. Problem Statement 1

II. Application 1

III. Constraint 2

A. Cyclic Permutation of the Original Sequence 2

B. Prefix Coding the Original Sequence 2

IV. Two Methods of Detection 2

A. Shift Register and Decisional Element 2

B. Sequential Network With a Minimum Number of Memory Elements . 3

V. An Algorithm for Determining the Number of Memory Elements and an Optimal State Assignment for Detecting a Given Sequence 3

A. Binary (n, r) Ring Sequence 3

B. State Assignment for a Sequential Network 4

VI. Implementation 6

VII. Further Examples 7

VIII. Selection of the Initial State 11

IX. Selection of a Sequence When Only *n* Is Specified 12

References 13

TABLES

1. **Comparison of $n - 1$ and $1 + [\log_2 n]$** 3
2. **State table** 4
3. **State assignment for detection of 010010111** 5
4. **State assignment for detection of the 21-bit sequence in Example 3** . . 8
5. **State assignment for sequence detector in Example 4** 11

FIGURES

1. A 10-bit sequence which does not satisfy the constraint 2

2. A binary (9, 4) ring sequence 3

3. NAND gate logic network with common collector operation 6

4. Implementation of the detector of sequence 010010111 6

5. Implementation of the detector of the 21-bit sequence in Example 3 . . 9

6. NAND implementation of AND when input variables exceed 12 9

7. Implementation of the detector of sequence 1111100000 9

8. Minimization of next-state functions of the detector in Example 4 . . . 10

9. Generalized feedback shift register for binary (n, r) ring sequence generation 11

10. Sequence generator 12

ABSTRACT

This Report presents an algorithm for synthesizing a sequential network to detect a given n-place binary sequence within a serialized binary data stream. Sequences are characterized as binary (n, r) ring sequences and the state assignment is determined from the n r-bit subsequences. The results are compared with the implementation of a binary sequence detector made up of an $n-1$ stage shift register and an n-input decisional element.

I. PROBLEM STATEMENT

A detector is required to recognize the arrival of a particular n-place binary sequence within a serialized (bit by bit) data stream. Upon the entry of the nth bit of the sequence into the detector and before the entry of the succeeding bit (i.e., during the nth bit interval), the detector is to furnish an output. The detection of a given 31-bit pseudorandom sequence was treated in Ref. 1. This has since been extended to the detection of any given n-place binary sequence within a data stream.

II. APPLICATION

Binary sequences are inserted at the beginning (or end) of a data frame or subframe emanating from a digital data processor of a spacecraft. Sequence detectors are used in the decoding equipment on the ground to provide "flags" which indicate the beginning (or end) of a data block (e.g., a TV frame).

The sequence detector is in essence an electronic combination lock which is opened for one digit period only when the proper sequence of binary digits is entered.

III. CONSTRAINT

The probability of n successive data bits being identical to a given n-place sequence should be reasonably low. *A priori* information about the data is not generally known. However, if each data bit arrives as a 0 or 1 with equal probability, the probability of n successive data bits being identical to a given n-bit sequence is 2^{-n} if the following constraint is imposed: The sequence of bits must be such that after it has been partially completed, the remaining bits or bit cannot be the beginning of the same sequence.

Example 1

Given the sequence in Fig. 1, where a_i precedes a_{i+1} in time. Note that a_5 through a_{10} and a_9 through a_{10} correspond to the first six and the first two digits of the sequence respectively. This is clearly undesirable because there is now a higher probability (i.e., $> 2^{-10}$) that these partially completed sequences together with preceding or succeeding data bits can correspond to the original sequence. The probability that the 10-bit sequence is formed with four data bits followed by a_1 through a_6 is 2^{-4}.

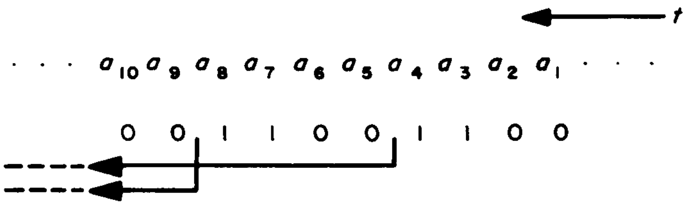
t
$\ldots a_{10}\, a_9\, a_8\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1 \ldots$
0 0 1 1 0 0 1 1 0 0

Fig. 1. A 10-bit sequence which does not satisfy the constraint

The constraint on the sequence to eliminate the preceding possibility may be satisfied in one of two ways.

A. Cyclic Permutation of the Original Sequence

A sequence where the first and last bit differ and the longest run of 0's or 1's appears at the beginning of the sequence satisfies the constraint when the sequence is not a composite of two or more identical sequences. An example is 1101000. Any sequence which is not a composite of two or more identical sequences (such as 1110011100 or 010101), when cyclically permuted such that the longest run of 0's or 1's is at the beginning, will satisfy the constraint. The sequence in Example 1 can be cyclically permuted such that it starts with ...11 or ...0000, i.e.,

1100110000
0000110011

Both satisfy the constraint.

B. Prefix Coding the Original Sequence

Prefix coding is employed in two cases.

1. The sequence is a composite of two or more identical sequences and cannot be made to satisfy the constraint through a cyclic permutation.

2. The sequence contains information which would change under a cyclic permutation. For example, a binary count, say i, may be used to indicate the start of the ith data frame.

The constraint in these two cases can be satisfied by prefixing these sequences with a sufficient number of 0's or 1's. The sequence in Example 1 must be prefixed with 111, resulting in

0011001100111

By prefixing 1110011100 with a single 0, the constraint is satisfied.

IV. TWO METHODS OF DETECTION

A. Shift Register and Decisional Element

A shift register made up of $n - 1$ two-state memory elements together with an n-input decisional element can serve as a detector. The register serially stores $n - 1$ bits. These and the nth bit (just prior to entering the register) are sensed by an n-input decisional element. Thus by "funneling" a stream of binary digits through the register, one of 2^n possible n-bit sequences can be located wherever it occurs. This method, though straightforward, is uneconomical in terms of the number of memory elements required.

B. Sequential Network With a Minimum Number of Memory Elements

The minimum number of states the detector must assume is equal to n, the number of bits in the sequence. Thus, the minimum of memory elements required is $1 + [\log_2 n]$. The bracketed term denotes the nearest integer which is less than $\log_2 n$. A comparison of $n - 1$ and $1 + [\log_2 n]$ for various values of $n \geqq 3$ appears in Table 1.

Table 1. Comparison of $n - 1$ and $1 + [\log_2 n]$

n	$n - 1$	$1 + [\log_2 n]$
3	2	2
4	3	2
5	4	3
6	5	3
7	6	3
8	7	3
9	8	4
·	·	·
·	·	·
·	·	·
16	15	4
17	16	5
·	·	·
·	·	·
·	·	·
32	31	5
33	32	6
·	·	·
·	·	·
·	·	·
64	63	6
65	64	7

The minimum-state sequential network requires a fraction of the number of memory elements (for $n > 3$) needed in the shift register approach. Unfortunately, there is no known algorithm for assigning combinations of state-values to each state such that the combinational logic (implemented with decisional elements) is of minimal complexity. In the case of the shift register, the combinational logic will in effect be a single n-input decisional element.

The algorithm to be presented in this Report enables a logic designer to:

1. Synthesize a detector for an n-bit sequence with r memory elements where $1 + [\log_2 n] \leqq r \leqq n - 1$. Sequences for which $r > n - 1$ are not considered. Make a state assignment which results in combinational logic of reasonable complexity. When compared to the shift-register detector, the overall complexity is significantly reduced. The reduction is most pronounced when n approaches 2^r.

2. Select an n-bit sequence for any given value of n such that its detector can be synthesized with $1 + [\log_2 n]$ memory elements. Make a state assignment such that the complexity of the combinational logic approaches that of the shift-register detector when implemented with diode gates.

Note that the constraint discussed in Sec. III is implied when referring to an n-bit sequence.

V. AN ALGORITHM FOR DETERMINING THE NUMBER OF MEMORY ELEMENTS AND AN OPTIMAL STATE ASSIGNMENT FOR DETECTING A GIVEN SEQUENCE

A. Binary (n, r) Ring Sequence

A given n-bit sequence (to be detected) may be characterized as a binary (n, r) ring sequence. The number of bits in the sequence is n, and r is a (fixed) number of consecutive bits in each of n distinct subsequences.

Example 2

The sequence 010010111 is a binary (9, 4) ring sequence. This is illustrated in Fig. 2.

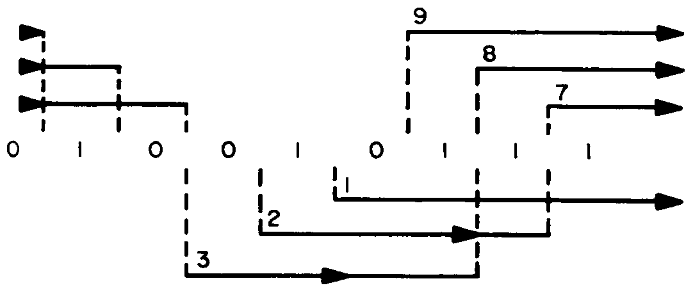

Fig. 2. A binary (9, 4) ring sequence

Six of the nine 4-bit subsequences are bracketed. The sequence is an ordered cycle such that the last bit is followed by the first as if the sequence were repeated or formed a ring. The nine 4-bit subsequences are:

0111
1011
0101
0010
1001
0100
1010
1101
1110

Note that $2^r \geqq n$. The minimum r that yields n distinct r-bit subsequences is of interest. Other values of r, where $r_{\min} \leqq r \leqq n - 1$, will also form n distinct subsequences. The subsequences represent the n states the detector must assume in the detection of the sequence. The length of the subsequences, r, represents the number of memory elements required.

The synthesis of the detector is illustrated by an example. A state table for the sequential network which is to serve as a detector is first constructed. This is done as follows for the sequence in Example 2.

```
0 1 0 0 1 0 1 1 1
9 8 7 6 5 4 3 2 1 bit number
```

B. State Assignment for a Sequential Network

In Table 2 the states of the sequential network are labeled numerically with an initial state designation of 1. The number of states, 9, corresponds to the number of bits in the sequence. The input to the detector is represented by the Boolean variable x.

Table 2. State table

Present state	Next state		Present output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
1	1	②	0	0
2	1	③	0	0
3	1	④	0	0
4	⑤	4	0	0
5	1	⑥	0	0
6	⑦	3	0	0
7	⑧	2	0	0
8	1	⑨	0	0
9	①	3	1	0

The arrival of the first 1 on the x input line (i.e., possible start of the sequence) causes the state transition from 1 (initial present state) to 2 (next state). Should each succeeding bit be part of the sequence to be detected, the sequential network progresses through each state in numerical order. This is indicated by the encircled next states in the state table. During the time the network is in present state 9 and a 0 is on the input line, the detector's (present) output is 1.

If at any time a bit is received which is not in the sequence, though previous bits were identical to the start of the sequence, the network must return to the initial state 1 or one of the states 2 through 4. Since the sequence begins with a 1, whenever a 0 arrives improperly located in the sequence, the network must return to state 1. However, whenever a 1 arrives improperly located in the sequence, the network must return to either state 2, 3, or 4. For example, assuming the network is in present state 6 (meaning the 5 previous bits corresponded to the first 5 bits in the sequence) and the 6th bit is 1 instead of 0, the network should not progress to state 7. It should instead return to state 3 since bit 5 and bit 6 (now entering) correspond to the first two bits in the sequence. Thus, the 5th bit of the 6-bit block could possibly be the start of the sequence to be detected.

The state assignment for the sequential network is taken from the ordered subsequences in the binary (9, 4) ring sequence. The successive states through which the detector progresses when each bit of the sequence appears in order is made to correspond to successive subsequences. Thus, once an assignment is made for initial state 1, the state assignment is complete. With r memory elements, there are $r - 1$ initial states such that one of the memory elements will track the incoming sequence. In the example there are three of the nine states when chosen as an initial state that will result in one memory element tracking the incoming sequence. Minimization of the combinational logic by means of a computer program (Ref. 2) shows that these initial states result in optimal minimization when state assignments correspond to the n subsequences in the binary (n, r) subsequence. The logic designer, therefore, may reduce the possible number of assignments from $(2^r - 1)!/[r!\,(2^r - n)!]$ (see Ref. 3) to $r - 1$. Any of the $r - 1$ assignments yields combinational logic of reasonable complexity. Usually, one or more will be superior to others. In many cases, the feedback shift register (FSR) which is capable of generating the (n, r) sequence provides a clue as to which of the $r - 1$ initial states, hence state assignments, is the most economical. This will be discussed in a later section.

Four memory elements are required in the example. Let $d_1d_2d_3d_4$ and $D_1D_2D_3D_4$ represent the present and next (combination of) states, respectively. By selecting 1010 as an initial state, $D_2 = x$. Table 3 illustrates this where all the states of Table 2 have been assigned.

Table 3 is divided into three parts for explanatory purposes. The top nine entries show how the detector behaves when the sequence is entered. The next nine entries show its behavior when a bit is on the x input line, which is not properly part of the sequence. For example, a total state, $xd_1d_2d_3d_4$ of 10101 indicates that the 5 bits previously entered correspond to the first 5 of the sequence (hence the network progressed to state 6). However, the present input $x = 1$ instead of $x = 0$ as in the sequence. The network's next state, $D_1D_2D_3D_4$, is 1110 or state 3 since the previous and present input could be the first two bits in the sequence. The lower portion of Table 3 represents unused total states (i.e., unused internal states combined with the input state). The next states for these entries are treated optionally.

Table 3. State assignment for detection of 010010111

x	d_1	d_2	d_3	d_4	Present state	D_1	D_2	D_3	D_4	Next state
1	1	0	1	0	1	1	1	0	1	2
1	1	1	0	1	2	1	1	1	0	3
1	1	1	1	0	3	0	1	1	1	4
0	0	1	1	1	4	1	0	1	1	5
1	1	0	1	1	5	0	1	0	1	6
0	0	1	0	1	6	0	0	1	0	7
0	0	0	1	0	7	1	0	0	1	8
1	1	0	0	1	0	0	1	0	0	9
0	0	1	0	0	9	1	0	1	0	1
0	1	0	1	0	1	1	0	1	0	1
0	1	1	0	1	2	1	0	1	0	1
0	1	1	1	0	3	1	0	1	0	1
1	0	1	1	1	4	0	1	1	1	4
0	1	0	1	1	5	1	0	1	0	1
1	0	1	0	1	6	1	1	1	0	3
1	0	0	1	0	7	1	1	0	1	2
0	1	0	0	1	8	1	0	1	0	1
1	0	1	0	0	9	1	1	1	0	3
0	0	0	0	0		ϕ	ϕ	ϕ	ϕ	
0	0	0	0	1		ϕ	ϕ	ϕ	ϕ	
0	0	0	1	1		ϕ	ϕ	ϕ	ϕ	
0	0	1	1	0		ϕ	ϕ	ϕ	ϕ	
0	1	0	0	0		ϕ	ϕ	ϕ	ϕ	
0	1	1	0	0		ϕ	ϕ	ϕ	ϕ	
0	1	1	1	1		ϕ	ϕ	ϕ	ϕ	
1	0	0	0	0		ϕ	ϕ	ϕ	ϕ	
1	0	0	0	1		ϕ	ϕ	ϕ	ϕ	
1	0	0	1	1		ϕ	ϕ	ϕ	ϕ	
1	0	1	1	0		ϕ	ϕ	ϕ	ϕ	
1	1	0	0	0		ϕ	ϕ	ϕ	ϕ	
1	1	1	0	0		ϕ	ϕ	ϕ	ϕ	
1	1	1	1	1		ϕ	ϕ	ϕ	ϕ	

Minimizing $D_1D_2D_3D_4$ and Z, the output, as functions of x, d_1, d_2, d_3, and d_4 results in the following:

$$D'_1 = xd'_2d_4 + xd_2d_3 + x'd'_1d'_3d_4 \tag{1}$$

$$D_1 = (x' + d_2 + d'_4)(x' + d'_2 + d'_3)(x + d_1 + d_3 + d'_4) \tag{1a}$$

$$D_2 = x \tag{2}$$

$$D_3 = x'd_1 + d_2 \tag{3}$$

$$D'_4 = x'd_1 + d'_3 \tag{4}$$

$$Z = x'd_2d'_3d'_4 \tag{5}$$

where the symbols $'$ and $+$ denote complementation and logical addition (i.e., OR), respectively. Adjacent Boolean variables, such as d_1d_2, denote logical multiplication (i.e., AND).

The expression (1a) indicates that D_1 is simpler in the conjunctive form. The complement of the function D'_1 was minimized in disjunctive form as shown in (1).

By making each entry under x identical to the corresponding entry under d_1 in the first nine entries, the memory element represented by d_2 tracks the input (i.e., $D_2 = x$ as previously asserted). In the next nine entries $x = d'_1$. These entries account for all the used total states.

VI. IMPLEMENTATION

All functions have been minimized under the criterion of minimum two-level AND-OR gating. Implementation, however, is done with NAND gates acting as decisional elements since the cost ratio of the *RS* flip-flop to the NAND gate is apparent. The *RS* flip-flop serves as a memory element. The cost ratio of an *RS* flip-flop to a diode gate (AND-OR having the same number of inputs as the NAND gate) would be even higher. Thus the economy realized by reducing the number of memory elements at the expense of adding NAND gates (i.e., fewer equivalent *RS* flip-flops) is less than that attainable with diode AND-OR gates.

The *RS* flip-flop has the following characteristic equation.

$$D = S' + Rd \qquad (6)$$

$$R'S' = 0 \qquad (6a)$$

S and *R* represent the state-values of the set and reset inputs which are voltage levels. The present output, *d*, is called the assertion output. Also available is *d'*, the negation output. The present outputs *d* and *d'* are also voltage levels and are represented by state-values. The voltage level and state value correspondences are:

$$0V \longleftrightarrow 0$$

$$+V \longleftrightarrow 1$$

D represents the next state of the flip-flop and is defined as the assertion output after a negative-going clock-input transition has been applied. Equation (6) is a Boolean difference equation which expresses the next state, *D*, as a function of *R*, *S*, and *d*. *R* and *S* are enabling or preconditioning signals. The effect of *RS* occurs after a negative-going clock transition. Thus, time (the occurrence of a clock transition) is implied in (6). The *RS* input combination of 00 is forbidden. This is expressed in (6a) algebraically as a constraint.

When *R* and *S* are complementary, the *RS* flip-flop acts as a delay element. The assertion output assumes the state value of *R* after a negative-going clock input. This can be deduced from (6).

For $S = R'$

$$D = R + Rd = R$$

When implementing sequence detectors, the *RS* flip-flop will be used as a delay element.

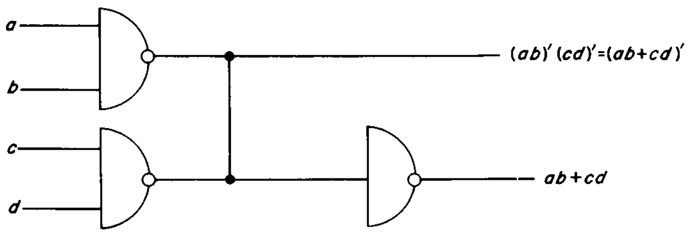

Fig. 3. NAND gate logic network with common collector operation

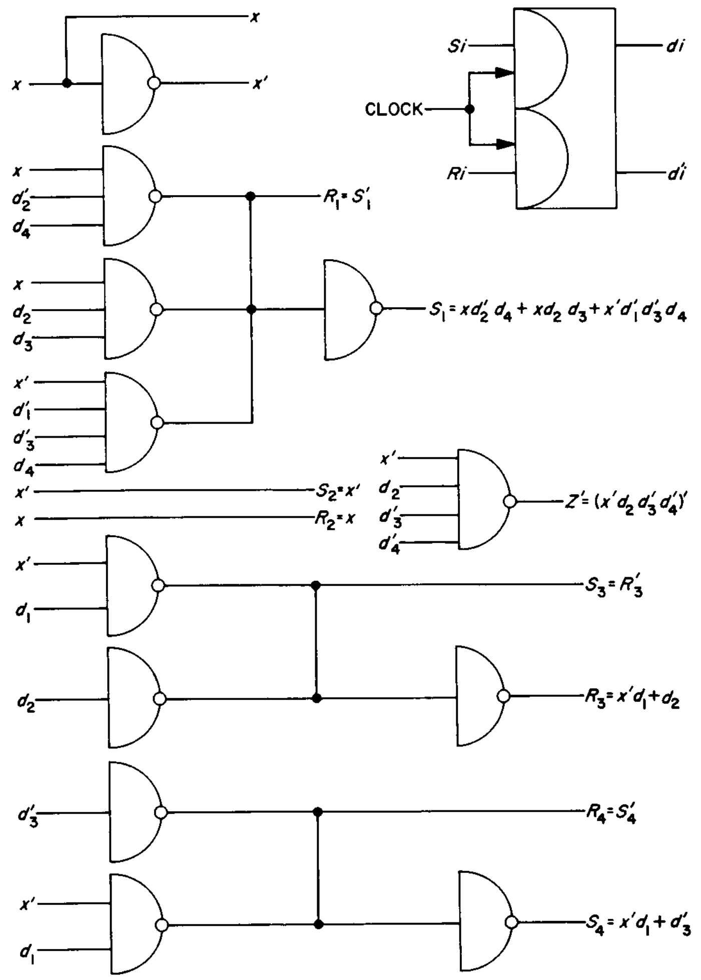

Fig. 4. Implementation of the detector of sequence 010010111

The output of a NAND gate is a function of its inputs only and is expressed as:

$$Z = (d_1 d_2 \cdots d_n)'$$

Z has a state-value of 0 when all inputs have a state-value of 1 and 0 otherwise. Two or more NAND gates may be operated with a common collector resistor yielding a NAND-AND operation for the previously defined voltage level and state-value correspondence. This is illustrated in Fig. 3. The NAND gate is symbolized as an AND-NOT. Note that unused inputs assume a $+V$ voltage level. Thus, a NAND gate with a single input acts as an inverter.

The implementation of the detector in Example 2 is shown in Fig. 4. The cost is 4 *RS* flip-flops and 12 NAND gates (with 4 or fewer inputs). The *RS* flip-flop utilizes a minimum of two transistors whereas the NAND gate has only one. The *RS* flip-flop may be considered as two cross-coupled NANDs with additional circuitry for the clock input. Thus a conservative cost ratio of the *RS* flip-flop to the NAND gate is 3 to 1. Therefore, the cost of the detector in Fig. 1 is equivalent to 24 NAND gates. The shift-register approach would require 8 *RS* flip-flops and 2 NAND gates. One is used to invert x and the other (with 9 inputs) is used to sense the input and the contents of the register. This is equivalent to 26 NAND gates. The 2 NAND gate reduction represents a modest improvement. However, this example represents the case where n is small (see Table 1) and $n/2^r = 9/16$. As n approaches 2^r for a given r, the improvement increases.

VII. FURTHER EXAMPLES

Example 3–Given the sequence

```
100011001010111110000
|    |     |    |   |
21   16    10   5   1
```

which can be characterized as a binary (21, 5) ring sequence. Thus, $n/2^r = 21/32$. The state assignment is shown in Table 4. Memory element d_2 tracks the input x as in Example 2. Note that, in general, there is no assignment such that d_1 tracks x (i.e., D_1 cannot equal x). Thus the $r - 1$ assignments mentioned previously are those for which either $D_2, D_3, \cdots$, or D_r equals x.

The minimized expressions for the next state of each memory element are:

$$D_1 = xd_1d_5' + x'd_1'd_5 + x'd_2'd_3'd_4' \tag{7}$$

$$D_2 = x \tag{8}$$

$$D_3' = d_2' + xd_1' \tag{9}$$

$$D_4 = xd_1d_3 + x'd_1'd_3 + d_1d_2'd_3 \tag{10}$$

$$D_5 = xd_1d_4 + x'd_1'd_4 + d_1d_3'd_4 \tag{11}$$

$$Z = xd_1d_2'd_3'd_5 \tag{12}$$

The implementation appears in Fig. 5. The cost is 5 *RS* flip-flops and 17 NAND gates (with 5 or fewer inputs). This is equivalent to 32 NAND gates. In the shift-register approach 20 *RS* flip-flops and 4 NAND gates are required. The 4 NAND gates are needed to sense the contents of the register and the input. In practice, the input expansion to a NAND gate is limited to 12. Therefore, two levels of NAND gates are required as shown in Fig. 6.

The shift-register detector is equivalent to 64 NAND gates. Therefore, a reduction of 2 to 1 is realized with the sequential network in Fig. 2.

There are sequences where commonality of terms in the next-state functions of the sequential network gives rise to gate sharing. In this case, the overall reduction over that of a shift-register detector is significant even though n is a fraction of 2^r. Sequences which have adjacent 1's and adjacent 0's in equal numbers are examples of this.

Example 4

Given the sequence

1111100000

which may be characterized as a binary (10, 5) sequence. In this case, $n/2^r = 10/32 < 1/3$. For the initial state

Table 4. State assignment for detection of the 21-bit sequence in Example 3

x	d_1	d_2	d_3	d_4	d_5	Present state	D_1	D_2	D_3	D_4	D_5	x	d_1	d_2	d_3	d_4	d_5	D_1	D_2	D_3	D_4	D_5
0	0	1	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0
0	0	0	1	0	0	2	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0
0	0	0	0	1	0	3	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0
0	0	0	0	0	1	4	1	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0
1	1	0	0	0	0	5	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	6	1	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0
1	1	1	1	0	0	7	1	1	1	1	0	0	1	1	1	0	0	0	0	1	0	0
1	1	1	1	1	0	8	1	1	1	1	1	0	1	1	1	1	0	0	0	1	0	0
1	1	1	1	1	1	9	0	1	1	1	1	0	1	1	1	1	1	0	0	1	0	0
0	0	1	1	1	1	10	1	0	1	1	1	1	0	1	1	1	1	0	1	0	0	0
1	1	0	1	1	1	11	0	1	0	1	1	0	1	0	1	1	1	0	0	0	1	0
0	0	1	0	1	1	12	1	0	1	0	1	1	0	1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	13	0	1	0	1	0	0	1	0	1	0	1	0	0	0	1	0
0	0	1	0	1	0	14	0	0	1	0	1	1	0	1	0	1	0	0	1	0	0	0
0	0	0	1	0	1	15	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	0
1	1	0	0	1	0	16	1	1	0	0	1	0	1	0	0	1	0	0	0	0	0	1
1	1	1	0	0	1	17	0	1	1	0	0	0	1	1	0	0	1	0	0	1	0	0
0	0	1	1	0	0	18	0	1	1	1	0	1	0	1	1	0	0	0	1	0	0	0
0	0	0	1	1	0	19	0	0	0	1	1	1	0	0	1	1	0	0	1	0	0	0
0	0	0	0	1	1	20	1	0	0	0	1	1	0	0	0	1	1	0	1	0	0	0
1	1	0	0	0	1	21	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0
0	0	0	1	1	1		ϕ	ϕ	ϕ	ϕ	ϕ	1	0	0	1	1	1	ϕ	ϕ	ϕ	ϕ	ϕ
0	0	1	0	0	1		ϕ	ϕ	ϕ	ϕ	ϕ	1	0	1	0	0	1	ϕ	ϕ	ϕ	ϕ	ϕ
0	0	1	1	1	0		ϕ	ϕ	ϕ	ϕ	ϕ	1	0	1	1	1	0	ϕ	ϕ	ϕ	ϕ	ϕ
0	1	0	0	1	1		ϕ	ϕ	ϕ	ϕ	ϕ	1	1	0	0	1	1	ϕ	ϕ	ϕ	ϕ	ϕ
0	1	0	1	0	0		ϕ	ϕ	ϕ	ϕ	ϕ	1	1	0	1	0	0	ϕ	ϕ	ϕ	ϕ	ϕ
0	1	1	0	1	0		ϕ	ϕ	ϕ	ϕ	ϕ	1	1	1	0	1	0	ϕ	ϕ	ϕ	ϕ	ϕ
0	1	1	1	0	1		ϕ	ϕ	ϕ	ϕ	ϕ	1	1	1	1	0	1	ϕ	ϕ	ϕ	ϕ	ϕ

assignment of 01111, $D_2 = x$ and the following optimal next-state and output functions are the result:

$$D'_1 = xd'_1 + x'd_2 + d_5 \tag{13}$$

$$D_2 = x \tag{14}$$

$$D_3 = xd'_1 + d_2 \tag{15}$$

$$D_4 = xd'_1 + x'd_2 + d_3 \tag{16}$$

$$D_5 = xd'_1 + x'd_2 + d_4 \tag{17}$$

$$Z = xd_1d_5 \tag{18}$$

Note that the terms $xd'_1 + x'd_2$ appear in three of the next-state functions and are utilized in deriving R_3 and S_3. (See Fig. 7 for the implementation.) The cost is 5 *RS* flip-flops and 10 NAND gates or equivalently 25 NAND gates. The inputs to each NAND gate are 3 or fewer. In the shift-register approach, 9 *RS* flip-flops and 1 NAND gate with 9 inputs are required. This is equivalent to 28 NAND gates.

The detector for sequences of the form in Example 4 has next-state and output functions which are identical to (13) through (18) where subscript 5 is replaced by s, the number of stages. A sequence of s 0's followed by s 1's would have a detector characterized by the following equations:

$$D'_1 = xd'_1 + x'd_2 + d_s$$

$$D_2 = x$$

$$D_3 = xd'_1 + d_2$$

$$\cdot$$

$$\cdot$$

$$\cdot$$

$$D_i = xd'_1 + x'd_2 + xd_{i-1}$$

where $3 < i \leq s$.

Example 5

The following 30-bit sequence may be characterized as a binary (30, 5) ring sequence.

101101110011111010010001100000

30 25 20 15 10 5 1

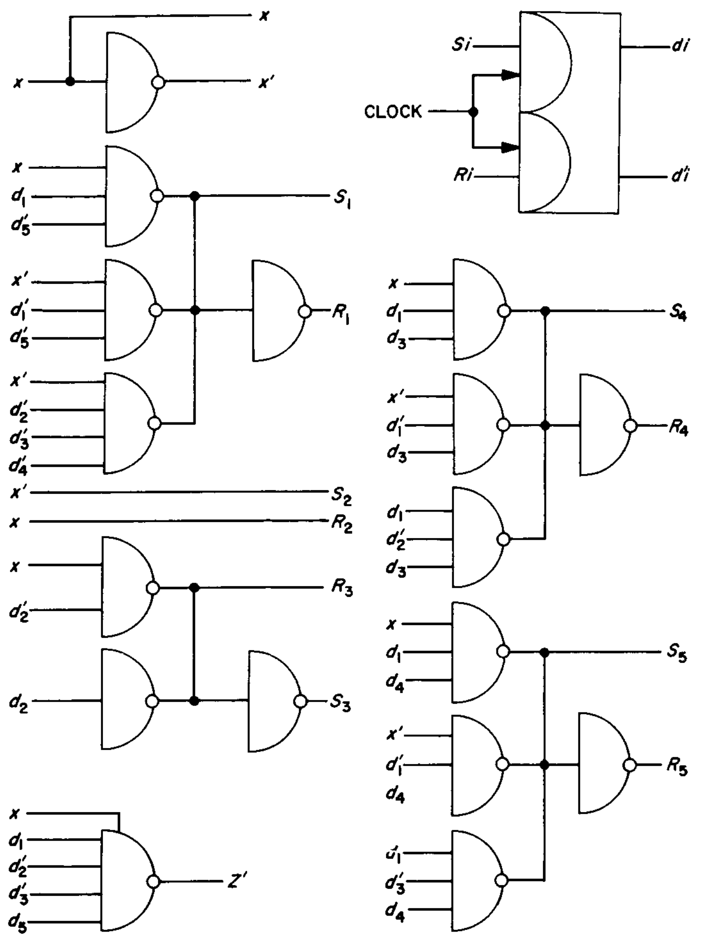

Fig. 5. Implementation of the detector of the 21-bit sequence in Example 3

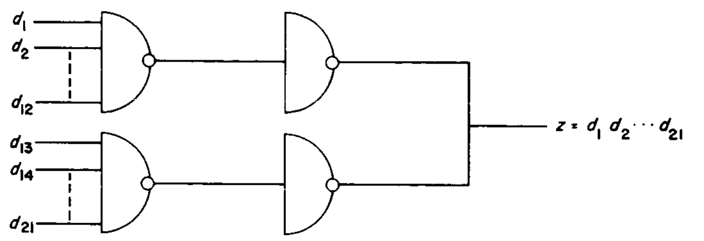

Fig. 6. NAND implementation of AND when input variables exceed 12

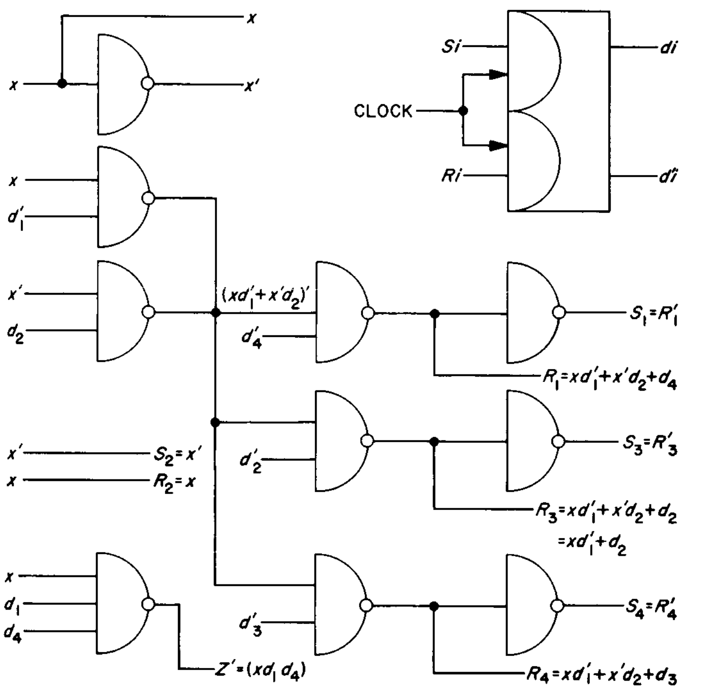

Fig. 7. Implementation of the detector of sequence 1111100000

The next-state functions, excluding D_3, are mapped on a Karnaugh Chart in Fig. 8 for minimization purposes. The minimized next-state and output functions are

$$D_1 = x'd_2'd_4'd_5' + x'd_2'd_4d_5 + xd_2d_4'd_5 + xd_2d_4d_5' + x'd_3'd_4'd_5' \tag{19}$$

$$D_2 = x'd_1d_2' + xd_1d_2 + x'd_1d_3'd_4'd_5' \tag{20}$$

$$D_3 = x \tag{21}$$

$$D_4' = xd_2' + d_3' \tag{22}$$

$$D_5' = d_2d_4' + x'd_4' + x'd_2d_3 \tag{23}$$

$$Z = xd_1'd_2d_3'd_4 \tag{24}$$

The cost is 5 *RS* flip-flops and 19 NAND gates with 5 or fewer inputs. A shift-register detector requires 29 *RS* flip-flops and 6 NAND gates (6 are needed to derive the output because of the limitation of 12 inputs per gate). The former is equivalent to 34 NAND gates as opposed to 93 NAND gates for the latter. This is almost a 3:1 reduction. In this case, the sequential network approach is more economical than the corresponding shift-register detector for a given n and $n/2^r$ ratio. If the $n/2^r$ ratio is maintained while n is increased, improvement over the shift register becomes more pronounced.

xd_1d_2 \ $d_3d_4d_5$	000	001	011	010	110	111	101	100
000	1		1			1		1
001	1			ϕ				
011	1							
010	1		1			1	ϕ	1
110							ϕ	
111		1		1	1		1	
101		1		ϕ	1		1	
100		$D_1 = x'd_2'd_4'd_5' + x'd_2'd_4d_5 + xd_2d_4'd_5 + xd_2d_4d_5' + x'd_3'd_4'd_5'$						

000	001	011	010	110	111	101	100
				1	1	1	1
			ϕ	1	1	1	1
				1	1	1	1
				1	1	ϕ	1
						ϕ	
				1	1	1	1
			ϕ	1	1	1	1
$D_4' = xd_2' + d_3'$							

000								
001				ϕ				
011	1							
010	1	1	1	1	1	1	ϕ	1
110							ϕ	
111	1	1	1	1	1	1	1	1
101				ϕ				
100				$D_2 = x'd_1d_2' + xd_1d_2 + x'd_1d_3'd_4'd_5'$				

		1	1	1	1		
		1	ϕ				
		1	1	$D_5' = d_2d_4' + x'd_4' + x'd_2d_3$			
		1	1	1	1	ϕ	
1	1	1	1	1	1	ϕ	1
		1	1	1	1		
		1	ϕ	1	1		
1	1	1	1	1	1	1	1

Fig. 8. Minimization of next-state functions of the detector in Example 4

VIII. SELECTION OF THE INITIAL STATE

The configuration of the simplest feedback shift register which is capable of generating the sequence often provides information as to which of the $r - 1$ possible initial states yields the best state assignment (i.e., which of the memory elements $d_2, d_3, \cdots, \text{or } d_r$ should track x). Every binary (n, r) ring sequence can be generated by an r-stage shift register where a Boolean function of its contents is fed back. This is termed a feedback shift register or FSR. See Fig. 9.

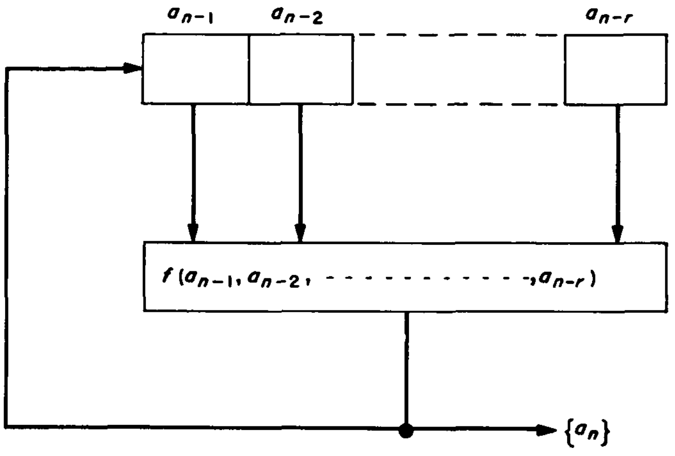

Fig. 9. Generalized feedback shift register for binary (n, r) ring sequence generation

The bit being fed back during the nth clock time interval is

$$a_n = f(a_{n-1}, a_{n-2}, \cdots, a_n) \qquad (25)$$

Successive subsequences provide a state table for determining $f(a_{n-1}, a_{n-2}, \cdots, a_n)$ in minimized form. The columns under $d_1d_2d_3d_4d_5$ and D_1 in the left half of Table 5

Table 5. State assignment for sequence detector in Example 4

x	d_1	d_2	d_3	d_4	d_5	Present state	D_1	D_2	D_3	D_4	D_5	x	d_1	d_2	d_3	d_4	d_5	D_1	D_2	D_3	D_4	D_5
0	0	0	0	0	0	4	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1
0	1	0	0	0	0	5	1	1	0	0	0	1	1	0	0	0	0	0	0	1	0	1
1	1	1	0	0	0	6	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0
1	0	1	1	0	0	7	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0
0	0	0	1	1	0	8	0	0	0	1	1	1	0	0	1	1	0	0	0	1	0	1
0	0	0	0	1	1	9	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
0	1	0	0	0	1	10	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	1
1	0	1	0	0	0	11	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	1	0	0	12	1	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
0	1	0	0	1	0	13	0	1	0	0	1	1	1	0	0	1	0	0	0	1	0	1
1	0	1	0	0	1	14	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0
0	1	0	1	0	0	15	1	1	0	1	0	1	1	0	1	0	0	0	0	1	0	1
1	1	1	0	1	0	16	1	1	1	0	1	0	1	1	0	1	0	0	0	0	0	1
1	1	1	1	0	1	17	1	1	1	1	0	0	1	1	1	0	1	0	0	0	1	0
1	1	1	1	1	0	18	1	1	1	1	1	0	1	1	1	1	0	0	0	0	1	0
1	1	1	1	1	1	19	0	1	1	1	1	0	1	1	1	1	1	0	0	0	1	0
1	0	1	1	1	1	20	0	0	1	1	1	0	0	1	1	1	1	0	0	0	1	0
0	0	0	1	1	1	21	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0	1
0	1	0	0	1	1	22	1	1	0	0	1	1	1	0	0	1	1	0	0	1	0	1
1	1	1	0	0	1	23	1	1	1	0	0	0	1	1	0	0	1	0	0	0	0	0
1	1	1	1	0	0	24	0	1	1	1	0	0	1	1	1	0	0	0	0	0	1	0
1	0	1	1	1	0	25	1	0	1	1	1	0	0	1	1	1	0	0	0	0	1	0
0	1	0	1	1	1	26	1	1	0	1	1	1	1	0	1	1	1	0	0	1	0	1
1	1	1	0	1	1	27	0	1	1	0	1	0	1	1	0	1	1	0	0	0	0	1
1	0	1	1	0	1	28	1	0	1	1	0	0	0	1	1	0	1	0	0	0	1	0
0	1	0	1	1	0	29	0	1	0	1	1	1	1	0	1	1	0	0	0	1	0	1
1	0	1	0	1	1	30	0	0	1	0	1	0	0	1	0	1	1	0	0	0	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1
0	0	0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0	1
0	0	0	0	0	1	3	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1
0	0	1	0	1	0		ϕ	ϕ	ϕ	ϕ	ϕ	1	0	1	0	1	0	ϕ	ϕ	ϕ	ϕ	ϕ
0	1	0	1	0	1		ϕ	ϕ	ϕ	ϕ	ϕ	1	1	0	1	0	1	ϕ	ϕ	ϕ	ϕ	ϕ

provide the information necessary to determine (25). D_1 corresponds to a_n whereas $d_1 d_2, \cdots, d_5$ correspond to $a_{n-1}, a_{n-2}, \cdots, a_{n-5}$, respectively. Thus

$$a_n = a'_{n-2} \oplus a_{n-4} \oplus a_{n-5}$$

where $\oplus$ denotes sum modulo 2 or the EXCLUSIVE-OR operation. The FSR for the sequence $\{a_n\}$ is shown in Fig. 10. The contents of the ith stage at clock time interval n(i.e., a_{n-i}) becomes the contents of the $(i+1)$th at clock time interval $n+1$. That is

$$a_{(n+1)-(i+1)} = a_{n-i}$$

It is assumed that the initial loading $a_{-1}a_{-2} \cdots a_{-5}$, where $n = 0$, is any one of the subsequences. The state assignment is such that an $(i+1)$th stage of the detector tracks x when the ith stage of the generator is in the feedback path. In the previous example, an initial state was chosen such that $D_3 = X$ since a_{n-2} is in the feedback. The case where

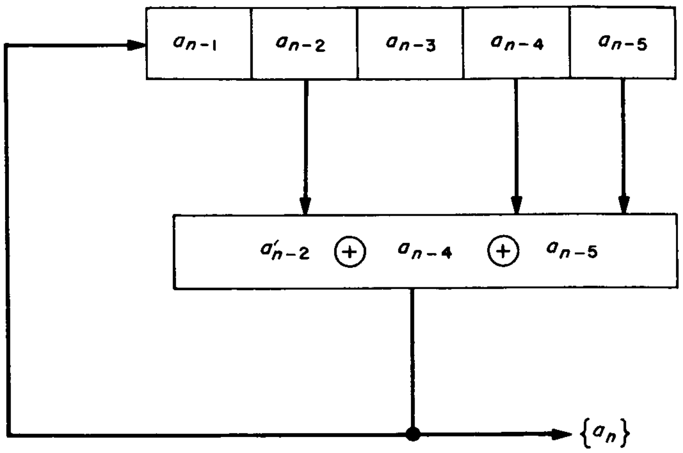

Fig. 10. Sequence generator

$D_5 = X$ gives results which are equivalent in terms of complexity (i.e., minimum AND-OR diode count). As indicated previously, there is no assignment which results in $D_1 = X$.

IX. SELECTION OF A SEQUENCE WHEN ONLY n IS SPECIFIED

In Ref. 4, it is proved that a binary (n, r) ring sequence exists for every n and r where $2^r \geqq n$. Therefore, a sequence with a specified n can be found such that r is minimum. This means a detector can be synthesized with a minimum number of memory elements.

For every $r > 1$, there are $\varphi(2^r - 1)/r$ FSR's capable of generating a maximal-length sequence whose length n equals $2^r - 1$. [$\varphi(n)$ is the Euler phi-function which denotes the number of positive integers equal to or less than n]. The detector for a maximal-length sequence is one of the most efficient in terms of hardware. In Ref. 1, the detector for a 31-bit maximal-length sequence was synthesized. The next-state functions were minimized simultaneously for each possible initial state by means of a general-purpose computer program (see Ref. 2). The results were optimum for the case where a memory element (in accordance with Sec. VII) tracked the input.

The feedback function for a maximal-length sequence generator can be determined from tables of irreducible polynomials (see Ref. 5). In turn, the sequence $\{a_n\}$ and all the subsequences $a_{n-1}a_{n-2} \cdots a_{n-r}$ can be found. (The subscript n denotes clock-time interval and is not to be confused with sequence length n). Furthermore, there is an algorithm for modifying the feedback function such that any sequence length from 1 to $2^r - 1$ can be obtained (Ref. 6). Of interest are sequence lengths from 2^{r-1} to $2^r - 1$ though a length of 2^{r-1} could be obtained with $r - 1$ memory elements.

The 31-bit sequence previously mentioned is:

```
1001011001111100011011101010000
|     |    |    |    |    |   |
31    25   20   15   10   5   1
```

The feedback function for the generator is

$$a_n = a_{n-2} \oplus a_{n-5}$$

The next-state functions for the detector are

$$D_1 = x'd'_2 d_5 + x d_1 d'_5 + x'd'_3 d'_4 d_5 \tag{26}$$

$$D_2 = (x'd'_2 + x d_2) d_1 + x'd'_3 d'_4 d'_5 \tag{27}$$

$$D_3 = x \tag{28}$$

$$D_4 = (x' + d_2) d_3 \tag{29}$$

$$D_5 = (x'd'_2 + x d_2) d_4 + x'd'_3 d_4 \tag{30}$$

and

$$Z = xd'_1d_2d'_3d'_5d_6 \tag{31}$$

By modifying the feedback function where

$$a_n = a_{n-2} \oplus a_{n-5} \oplus W$$

a specified number of bits within the 31-bit sequence is skipped. W is 1 for a particular nonzero combination of states in the FSR. Two examples follow:

For $n = 18$

$$W = a_{n-1}a'_{n-2}a'_{n-3}a'_{n-4}a'_{n-5}$$

and the generated sequence is

```
100101100111110000
'  '    '    '   '
18 15   10   5   1
```

Note that bits 2 through 14 in the original 31-bit sequence have been skipped.

The next-state functions for the detector are

$$D_1 = x'd'_2d_5 + xd_2d'_5 + x'd'_3d'_4 \tag{32}$$

D_2 through D_5 are the same as (27) through (30). Also, Z can be the same as (31).

For $n = 27$

$$W = a_{n-1}a_{n-2}a'_{n-3}a_{n-4}a'_{n-5}$$

The sequence is the same as the 31-bit sequence with bits 7 through 10 deleted.

As in the case for $n = 27$, only D_1 differs.

$$\begin{aligned} D_1 = {} & xd_2d'_3(d_1d'_3d_4d'_5)' \\ & + x'd'_2d_3 + xd'_3d'_4d'_5 \end{aligned} \tag{33}$$

Thus, the 18-bit and 27-bit sequence detector require one more NAND gate than the parent 31-bit sequence (whose cost is 5 *RS* flip-flops and 17 NAND gates). Detectors for sequences of length $n = 16$ through $n = 30$ are also of the same complexity as that for the 31-bit sequences. These, of course, are derived from the same 31-bit sequence.

REFERENCES

1. Perlman, M., "Binary Sequence Detectors," Jet Propulsion Laboratory, Pasadena, Calif., *Space Programs Summary, No. 37-31*, Vol. IV, pp. 211–214.

2. Burgess, C. R., "Boolean Algebra Minimizer," SHARE Program LLBAM 1197, Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Massachusetts, September 1961.

3. McCluskey, E. J., Jr., and Unger, S. H., "A Note on the Number of Internal Variable Assignments for Sequential Switching Circuits," *IRE Transactions on Electronic Computers*, Vol. EC-8, No. 4, December 1959, pp. 439–440.

4. Yoeli, M., "Binary Ring Sequences," *American Mathematical Monthly*, Vol. 69, November 1962, pp. 852–855.

5. Peterson, W. W., *Error-Correcting Codes*, John Wiley and Sons, Inc., New York, 1961.

6. Baumert, L. D., *Table of Period Generators*, Technical Report No. 32-564, Jet Propulsion Laboratory, Pasadena, Calif., November 1, 1962.