DESIGN AND IMPLEMENTATION OF A K=3, RATE ½, $\begin{matrix}110\\111\end{matrix}$ MINIMUM-DISTANCE FEEDBACK DECODER

by

STEPHEN ALEXANDER DYER

B. S., Kansas State University, 1973

A MASTER'S THESIS

submitted in partial fulfillment

of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1974

Approved by:

Donald R. Hummels

Major Professor

THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH THE ORIGINAL
PRINTING BEING
SKEWED
DIFFERENTLY FROM
THE TOP OF THE
PAGE TO THE
BOTTOM.

THIS IS AS RECEIVED
FROM THE
CUSTOMER.

TABLE OF CONTENTS

CHAPTER PAGE

I. INTRODUCTION 1

Convolutional Encoding 1

Decoder Types 5

II. FEEDBACK DECODING 10

Description of Feedback Decoding 10

The Threshold Decoder 12

III. THE MINIMUM-DISTANCE FEEDBACK DECODER 16

IV. HARDWARE IMPLEMENTATION 26

Decoder Implementation 26

Coder Implementation 38

V. EVALUATION AND AN ADDITIONAL CONSIDERATION 40

Performance 40

Decoder Synchronization 43

VI. CONCLUSION 44

BIBLIOGRAPHY 47

LIST OF TABLES

TABLE PAGE

I. Worst-Case Decoder Performance 41

II. Specifications -- Minimum-Distance Feedback Decoder 46

LIST OF FIGURES

FIGURE		PAGE
1.	K=3, n=2, b=1 $\begin{matrix}110\\111\end{matrix}$ Convolutional Coder	2
2.	Trellis Diagram for K=3, Rate ½, $\begin{matrix}110\\111\end{matrix}$ Coder	4
3.	Communication System Using Convolutional Coding and Decoding	6
4.	Binary Symmetric Channel	7
5.	Feedback Decoder Operation	11
6.	K=6, Rate ½, $\begin{matrix}100000\\100111\end{matrix}$ Coder for Threshold Decoding	14
7.	Threshold Decoder for K=6, Rate ½, $\begin{matrix}100000\\100111\end{matrix}$ Code	15
8.	Possible Source Data Sequences	17
9.	Possible Source Data Sequences (Rearranged)	19
10.	Possible Source Data Sequences (Final Arrangement)	20
11.	Development of Scheme for Generating All Possible Trellis Paths Three Branches Long Emanating from Node Corresponding to a Given State	22
12.	Method of Comparing Generated Trellis Paths with Received Sequence	23
13.	Simplified Logic Diagram for Decoder	25
14.	Implementation of Modulo-2 Adders	27
15.	Shift Register Implementation	28
16.	The Arithmetic Unit	30
17.	Logic Diagram for Decoder (Less Decoder Clock and Timing Circuitry)	31

LIST OF FIGURES (CONTINUED)

FIGURE PAGE

18. Decoder Timing Diagram 32,33

19. Timing Circuitry (Counter and Decimal Decoder) . . 35

20. Timing Circuitry (Gating) 36

21. Photograph of Minimum-Distance Feedback Decoder . . 37

22. Implementation of K=3, Rate ½, $\begin{matrix}110\\111\end{matrix}$ Convolutional Coder 39

CHAPTER I

INTRODUCTION

I. CONVOLUTIONAL ENCODING

Convolutional codes are a subclass of parity-check codes. A convolutional encoder consists of a K-stage shift register and n modulo-2 adders which are connected to prescribed stages of the shift register.

A modulo-2 adder is a parity-check device which outputs a logical one if there is an odd number of ones at its inputs, and a logical zero if there is an even number of ones at its inputs. These modulo-2 adders are also called function generators. An encoder with function generators $\begin{matrix}101\\111\end{matrix}$ has two modulo-2 adders, one with taps on the first and third stages of the shift register, and one with taps on all three stages of the shift register. That the coder has a three-stage shift register is implied in the statement of the function generators.

Information bits are shifted through the register b bits at a time, and the resulting coded sequence is taken from the outputs of the modulo-2 adders with the aid of a commutating switch.

A K=3, n=2, b=1 convolutional encoder with function generators $\begin{matrix}110\\111\end{matrix}$ is shown in Figure 1. An example input sequence is shown along with the outputs of each modulo-2

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

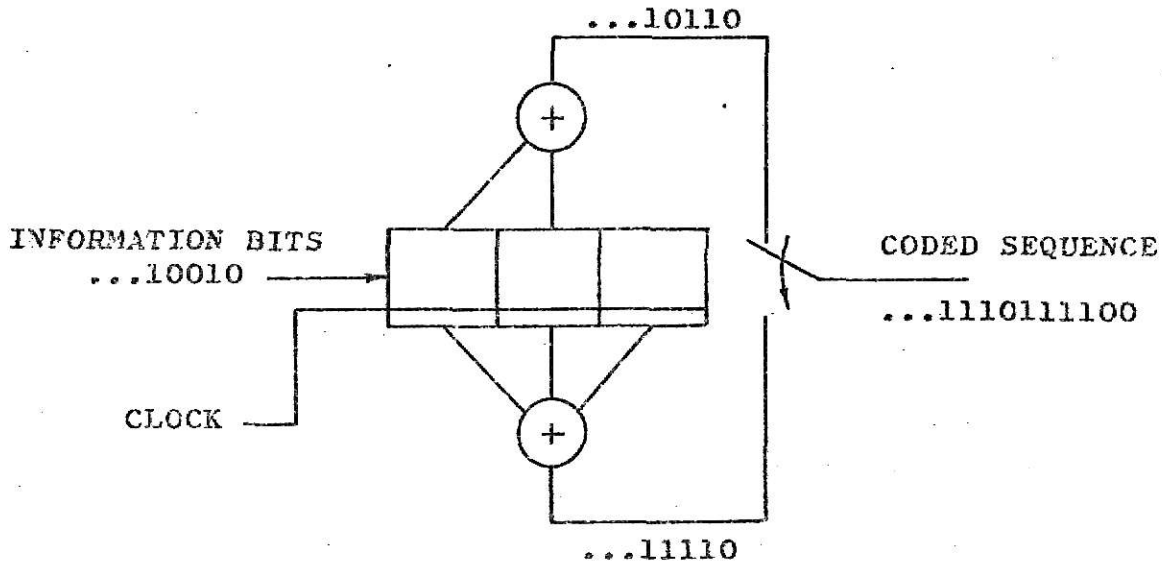


Figure 1. K=3, n=2, b=1 $\frac{110}{111}$ Convolutional Coder.

adder and the resulting coded sequence. The leftmost bit in each sequence is considered the most recent bit, and the register is assumed to have been initially filled with zeros.

The rate R_N of a convolutional code is

$$R_N = \frac{b}{n}$$

where b is a positive integer less than n. In practice, b=1, generally.

The three traditional representations for a convolutional code are the tree diagram, the state diagram, and the trellis diagram, the trellis diagram perhaps being the most easily understood yet terse representation.

Figure 2 shows the trellis representation for the coder of Figure 1. By convention, solid lines designate coder branches produced by logical zero input bits and dashed lines designate branches produced by logical one input bits.

The "state" of a coder consists of all bits in the register except the oldest bit; thus, a coder with constraint length K has states (K-1) bits in length associated with it. As an example, if a K=3 coder is in the 00 state and the forthcoming information bit is a "one", then the next state of the coder will be the 10 state.

The output code symbols associated with each branch are noted beside that branch in the trellis diagram.

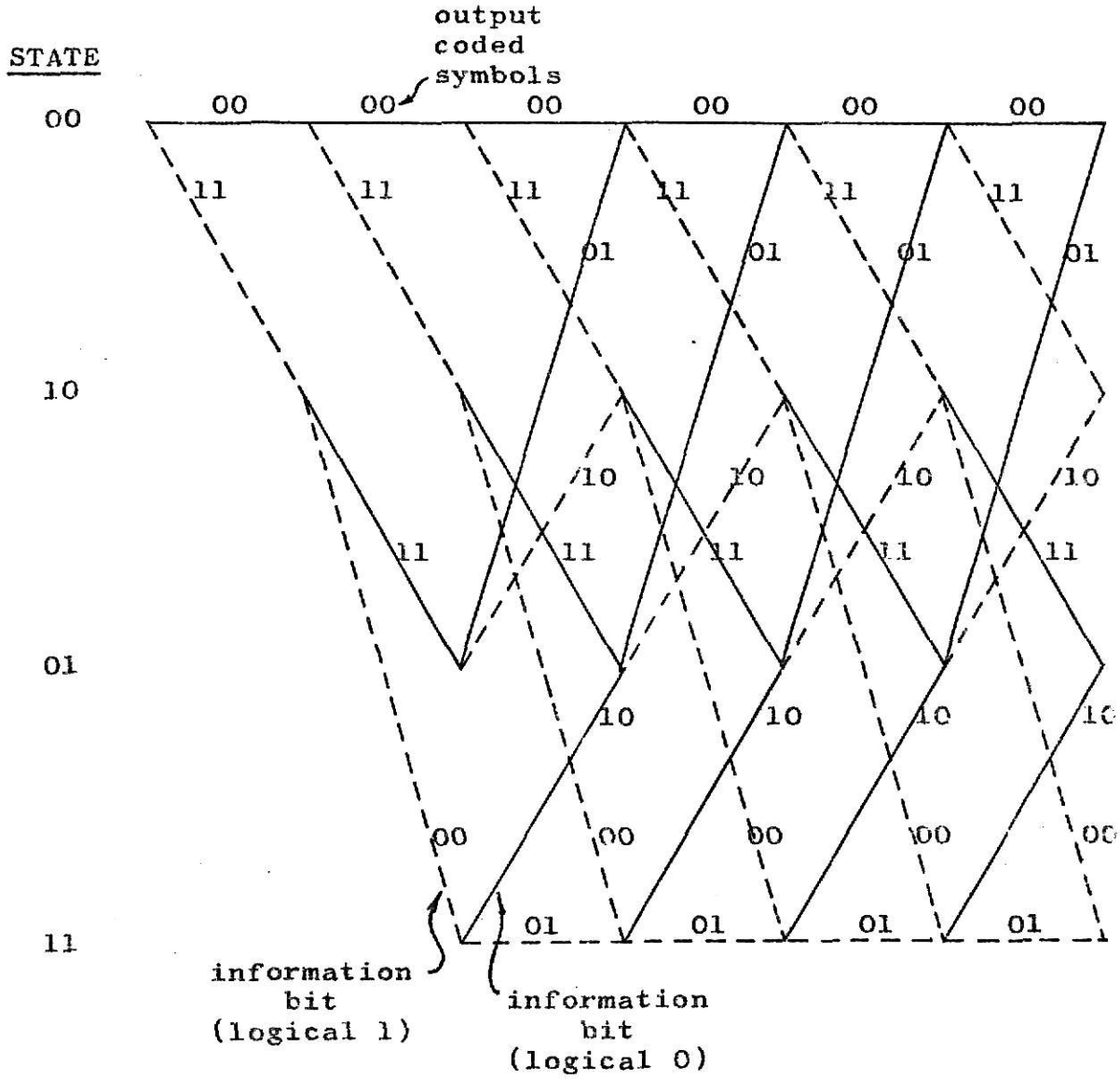


Figure 2. Trellis Diagram for K=3, Rate ½, $\frac{110}{111}$ Coder.

A block diagram of a communication system using a convolutional code is shown in Figure 3.

When random errors are dealt with, the assumption is almost always made that the received symbol will be the same as the transmitted symbol with probability 1-p, and that the opposite symbol will be received with probability p. The probability p is called the transition probability, and is also the error probability for a symbol. Figure 4 shows the binary symmetric channel (BSC), which satisfies these conditions. The BSC assumption is made in this discussion.

II. DECODER TYPES

There are three well-known types of decoders for use with convolutional codes -- sequential, Viterbi, and feedback. Sequential and Viterbi decoding are considered to be optimal methods, while the feedback method is suboptimal.

Sequential decoding was first considered by Wozencraft and modified later by Fano (4). The Fano algorithm sequentially searches the code tree, attempting to find a path whose metric rises faster than some threshold which is variable. The metric is a measure of the difference between actual received symbols and possible transmitted symbols. The Fano algorithm is attractive for use with codes of large constraint length K since the difference between the

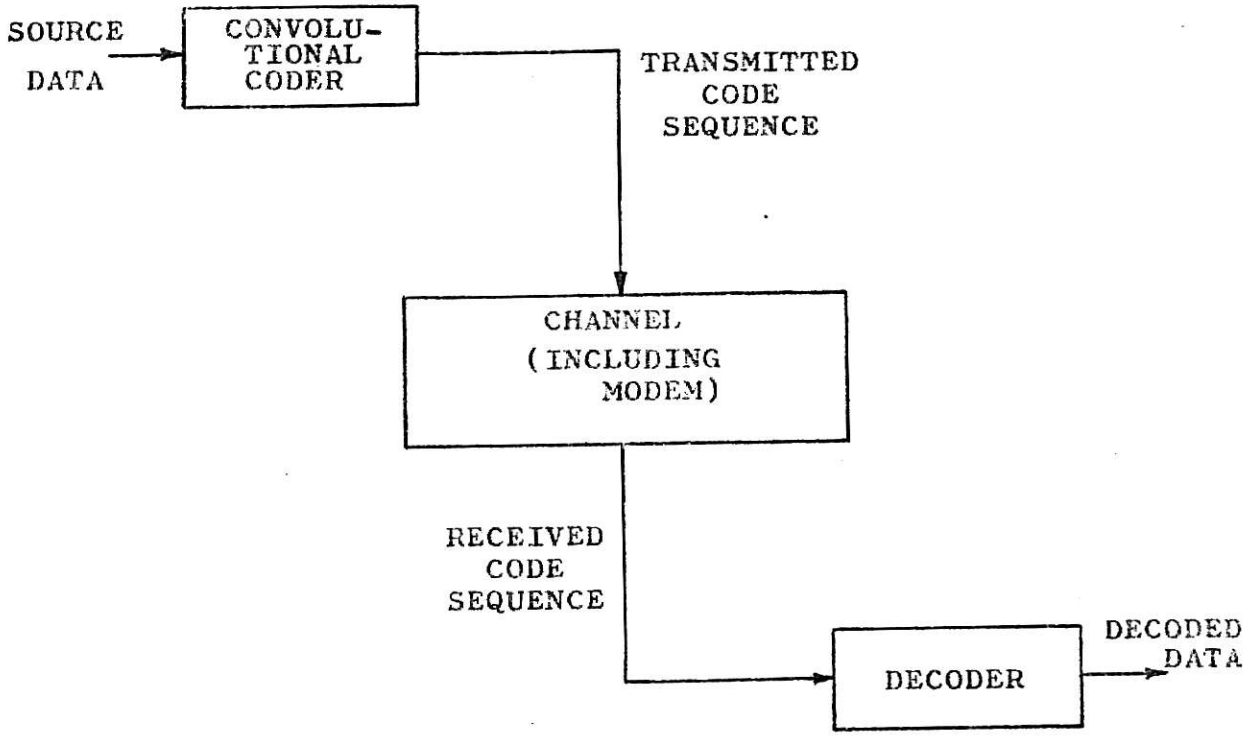


Figure 3. Communications System Using Convolutional Coding and Decoding.

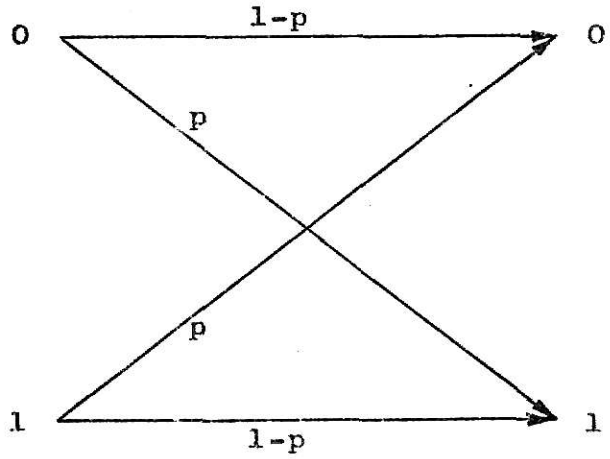


Figure 4. Binary Symmetric Channel.

correct path metric and any incorrect path metric increases with K. A problem arises, though, since the number of incorrect path branches that the decoder searches is a random variable depending on channel noise. Storage overflow and consequential erasure of long sequences are results of large numbers of branches per bit being searched. Another disadvantage of the Fano algorithm lies in the fact that the decoder is not self-synchronizing; data must be blocked (3).

Viterbi, or maximum-likelihood, decoding is merely minimum distance decoding when the channel is considered to be binary symmetric. Distance is defined as the number of symbols in which the received sequence differs from a possible transmitted sequence. Thus, if all message sequences are equally likely, the decoder chooses the data sequence corresponding to the transmitted sequence which differs from the received sequence in the fewest number of symbols.

Feedback decoding offers a somewhat smaller improvement in performance, but is desirable in the respect that implementation for binary input-binary output channels is relatively simple. There is at least one type of feedback decoder -- the threshold decoder -- which utilizes the algebraic properties of specially constructed convolutional codes. Gallager (1) asserts that threshold decoding can be applied to any convolutional code, but Lucky, et al (4), point out that error propagation is a serious problem

if certain restrictions are not imposed upon the code used.

CHAPTER II

FEEDBACK DECODING

I. DESCRIPTION OF FEEDBACK DECODING

Feedback decoding is so-named because decoding decisions are fed back to help determine future decisions. The general procedure used in feedback decoding can be most readily demonstrated by example. Figure 5 shows the trellis diagram for the $\begin{matrix}110\\111\end{matrix}$ convolutional code along with sample transmitted and received sequences; note the single channel error in the received sequence. The leftmost bit in each sequence is the oldest, and previous transmitted bits are assumed to be all zeros with no errors having occurred. Thus, the decoder has been traveling along the all-zeros path up to point a. Below the trellis are the source data producing the transmitted sequence, and the decoded estimate.

The particular decoder in this example looks at three branches of the trellis at a time. The specification will be made that the decoder take the uppermost path when two or more paths are at the same minimum distance from the received sequence. When the decoder is at point a, it looks at the received sequence above the leftmost encirclement while looking at all possible paths three branches in length which emanate from a. It sees that the paths 11 11 10

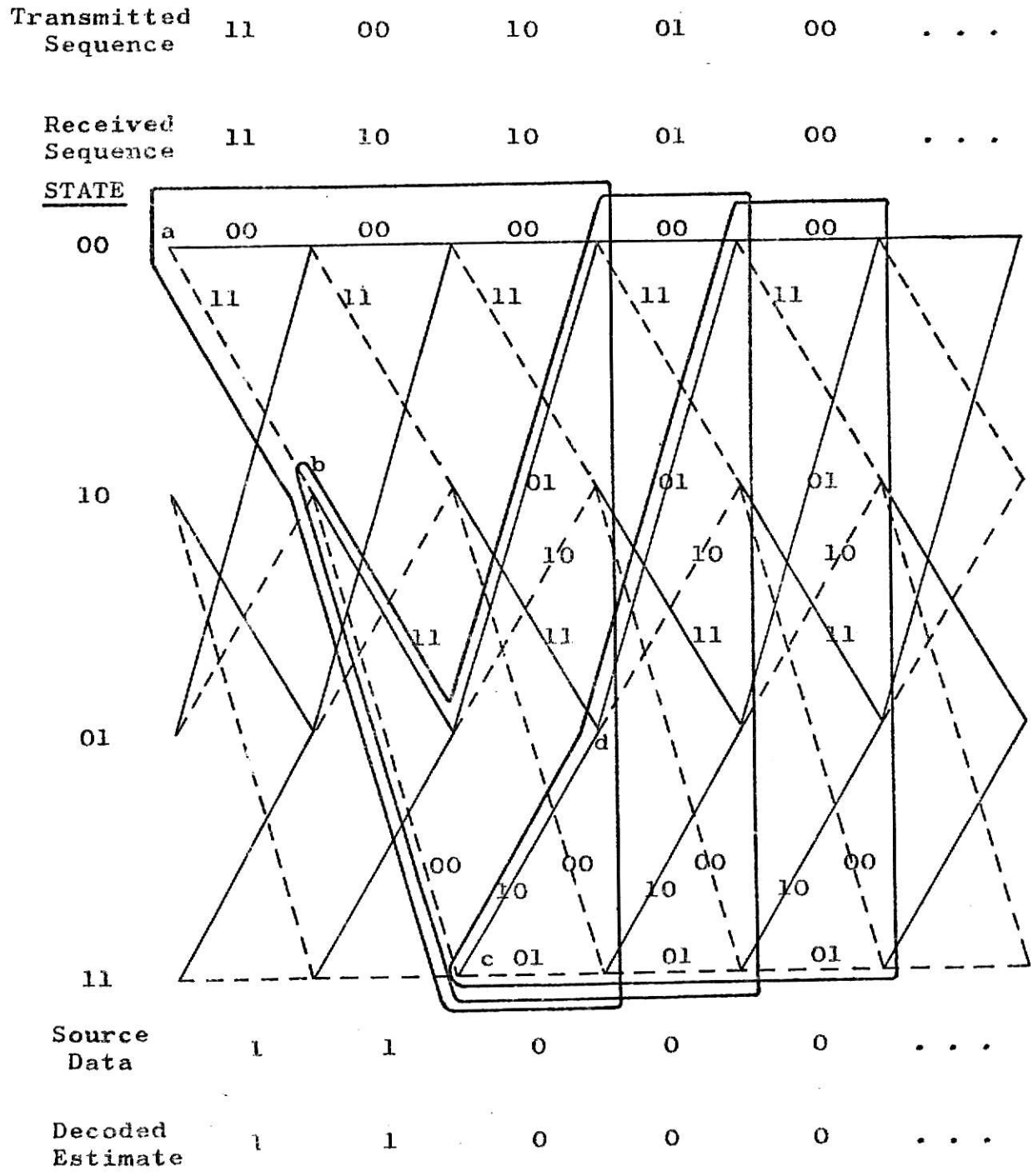


Figure 5. Feedback Decoder Operation.

and 11 00 10 are closest to the received sequence being looked at, both at a distance of one from the received sequence, and consequently decides that the next state is 10. The decoder then moves to point b and, in the process, outputs the "1" which is associated with the branch from a to b. The decoder next looks at the received sequence above the middle encirclement and picks the trellis path within this encirclement which is nearest the received sequence, that path being 00 10 01. The decoder now moves along this chosen path to point c and outputs a "1" since a "1" is associated with the branch from b to c. Similar procedure is used subsequently, the path from point c closest to the observed received sequence being 10 01 00, the corresponding decoding output being "0", and so forth.

Note that if a decoding error occurs, the decoder moves to a wrong state and paths from that wrong state are compared with the received sequence. More than likely, another error (or errors) will occur; but, fortunately, this error propagation is limited and within a few constraint lengths the decoder will find its way to the proper path and begin decoding correctly once again.

II. THE THRESHOLD DECODER

The threshold, or majority-logic, decoder incorporates a threshold logic device which outputs a logical one if a

majority of ones is at its inputs. The inputs to the threshold device are a linear combination of syndrome bits.

A popular example of a code useful in threshold decoding is the K=6, rate ½, $\begin{matrix}100000\\100111\end{matrix}$ code whose implementation is shown in Figure 6. The syndrome is

$$s_j = p'_j \oplus i'_j + i'_{j-3} \odot i'_{j-4} \oplus i'_{j-5}$$

where

$$i'_j = i_j \oplus e_j^{(1)}$$

and

$$p'_j = p_j \oplus e_j^{(2)}$$

$e_j^{(1)}$ and $e_j^{(2)}$ being the error digits introduced by the channel on the j^{th} information and parity symbols, respectively.

The syndrome is a function of the noise symbols only and thus can be written as

$$s_j = e_j^{(2)} \oplus e_j^{(1)} \oplus e_{j-3}^{(1)} \oplus e_{j-4}^{(1)} \oplus e_{j-5}^{(1)}$$

The syndromes and combination of syndromes required by the decoder are

$$s_1 = e_1^{(2)} \oplus e_1^{(1)}$$

$$s_2 + s_5 = e_5^{(2)} \oplus e_5^{(1)} \oplus e_2^{(2)} \oplus e_1^{(1)}$$

$$s_4 = e_4^{(2)} \oplus e_4^{(1)} \oplus e_1^{(1)}$$

$$s_6 = e_6^{(2)} \oplus e_6^{(1)} \oplus e_3^{(1)} \oplus e_2^{(1)} \oplus e_1^{(1)}$$

The decoder forms the syndrome from the received data and uses it to determine whether or not a received

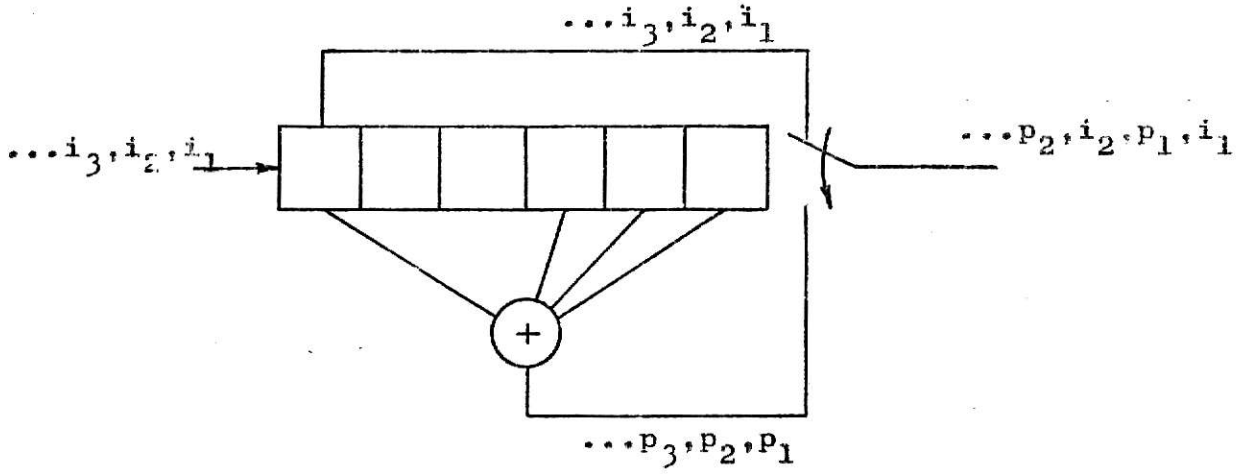


Figure 6. K=6, Rate ½, $\begin{matrix}100000\\100111\end{matrix}$ Coder for Threshold Decoding.

bit is in error. If three or more of the above equations equal one, then the corresponding received data bit is assumed to be in error and is complemented before it leaves the decoder. Figure 7 shows the construction of the decoder.

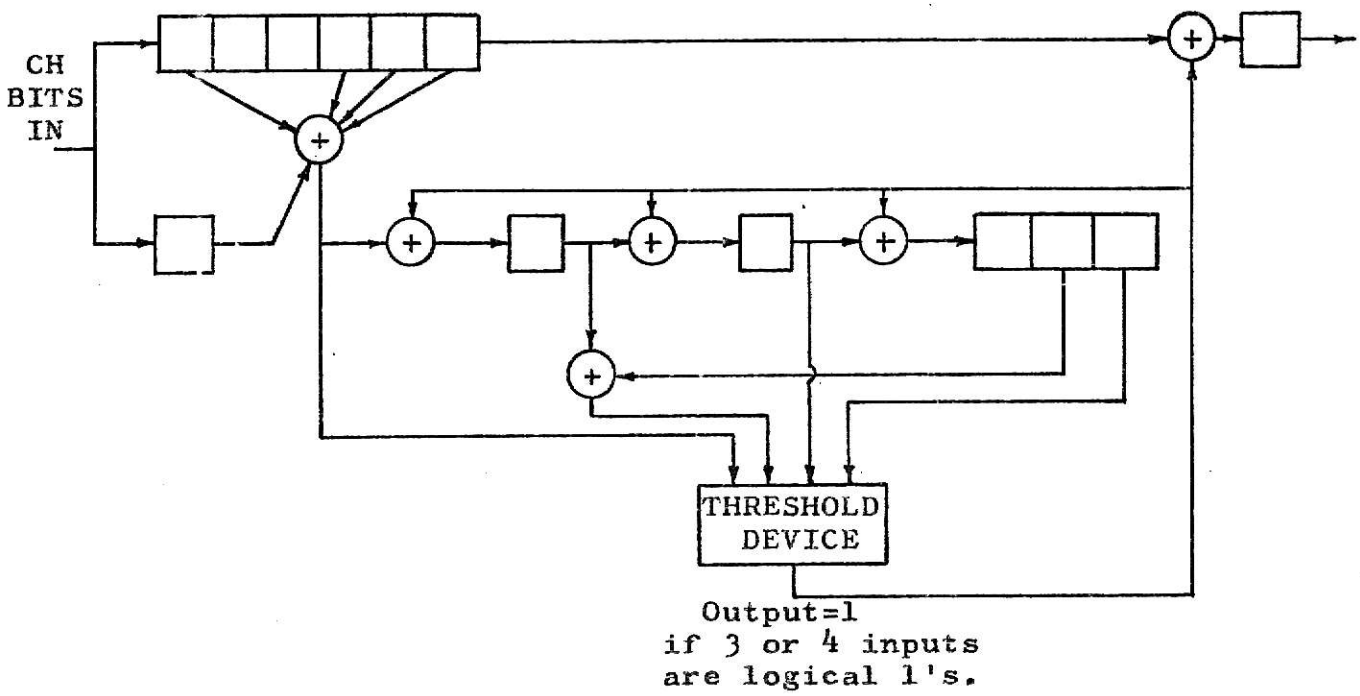


Figure 7. Threshold Decoder for K=3, Rate ½, $\frac{100000}{100111}$ Code.

CHAPTER III

THE MINIMUM-DISTANCE FEEDBACK DECODER

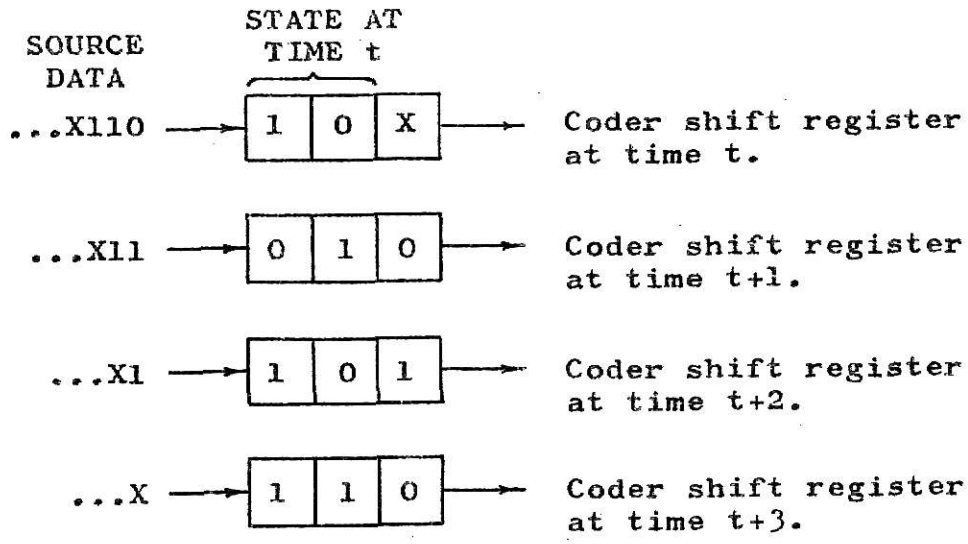
There is extensive literature in the field of information theory that concerns itself with majority-logic decoding of convolutional codes, but no evidence has been found of the design and implementation of a feedback decoder which uses the basic procedure of searching the code trellis. The trellis-searching procedure eliminates the restrictions that are imposed on codes used in conjunction with algebraic feedback decoding techniques.

In this effort, a decoder will be designed which looks ahead three branches into the trellis. This decoder requires relatively little hardware when built as a serial-parallel machine, and the design principle can be extended easily to allow the decoder to look at codes of greater constraint length and consequently correct more complicated error patterns.

It is instructive to look at the possible sequences three bits in length from which a coder in a given state could generate transmitted sequences. Figure 8 shows the shift register contents of a K=3 encoder. For example, if the coder is in the 10 state at time t, and the next bit into the register is a "zero", then the register contents are 010 at time t+1 (the next cycle of the coder). If the next bit is a "one", then the contents are 110. All

STATE AT TIME t	TIME								
00	t+1	000	000	000	000	100	100	100	100
	t+2	000	000	100	100	010	010	110	110
	t+3	000	100	010	110	011	101	011	111
10	t+1	010	010	010	010	110	110	110	110
	t+2	001	001	101	101	011	011	111	111
	t+3	000	100	010	110	001	101	011	111
01	t+1	001	001	001	001	101	101	101	101
	t+2	000	000	100	100	010	010	110	110
	t+3	000	100	010	110	001	101	011	111
11	t+1	011	011	011	011	111	111	111	111
	t+2	001	001	101	101	011	011	111	111
	t+3	000	100	010	110	001	101	011	111

a. Possible source data sequences.



X = symbol not pertinent to discussion

b. Pictorial explanation of encircled region of (a.) above.

Figure 8. Possible Source Data Sequences.

of this is illustrated in Figure 8 by the contents of the encircled region. Figure 9 rearranges the possible sequences to show that all possible transmitted sequences can be generated by counting from zero to seven in binary.

Figure 10 shows even more clearly how all possible transmitted sequences three bits in length can be generated. All possible trellis paths three branches in length can be generated simply by storing the presumed state of the coder; generating all binary numbers from 000 through 111, inclusive, with a three-bit binary counter; and connecting modulo-2 adders in such a manner that each three-section group of the counter/storage register combination generates the particular code used in the encoder-decoder ensemble. Figure 11 shows schematically what is being done. Storage of the presumed state is accomplished by a two-bit shift register. The $\begin{matrix}110\\111\end{matrix}$ code is used in this design although other codes can be used simply by making different connections to the modulo-2 adders.

The simple scheme outlined in Figure 11 for generating trellis paths now provides the basis for the decoder. It is necessary, basically, to assume a state, generate all possible paths from that state, compare all possible paths with three consecutive branches of received channel bits, pick the generated sequence nearest the received sequence, and output the bit which produced the first branch of the generated sequence.

STATE								
00	000	100	000	100	000	100	000	100
	000	010	100	110	000	010	100	110
	000	001	010	011	100	101	110	111
10	010	110	010	110	010	110	010	110
	001	011	101	111	001	011	101	111
	000	001	010	011	100	101	110	111
01	001	101	001	101	001	101	001	101
	000	010	100	110	000	010	100	110
	000	001	010	011	100	101	110	111
11	011	111	011	111	011	111	011	111
	001	011	101	111	001	011	101	111
	000	001	010	011	100	101	110	111

Figure 9. Possible Source Data Sequences (Rearranged).

			STATE	
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

			STATE	
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

			STATE	
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

			STATE	
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Figure 10. Possible Source Data Sequences (Final Arrangement).

The received sequence of channel bits can be stored in a serial shift register and compared with the sequences generated by the basic device defined in Figure 11e, f, and g by using six modulo-2 adders as shown in Figure 12. The output of each modulo-2 adder is a "zero" if the bits compared are identical and is a "one" if the bits differ from each other.

To find the number of bits in a generated sequence that differ from the corresponding bits of the received sequence, the outputs of these modulo-2 adders are fed into an arithmetic unit which adds the number of logical ones at its inputs and outputs that number in binary. That number is called the Hamming distance when the BSC is considered. The Hamming distance is stored in a three-bit parallel-in/parallel-out shift register which is connected to a three-bit digital comparator. A digital comparator has as its inputs two binary numbers A and B, and generally has as available outputs A<B, A>B, and A=B, each of these outputs being a logical one if the specific condition exists, and logical zero otherwise.

The decoder assumes a state, generates a trellis path, compares it with the received sequence, calculates and stores the Hamming distance, and stores the bit associated with the first trellis branch of the generated sequence. The decoder then generates another trellis path, calculates the Hamming distance and compares it with the

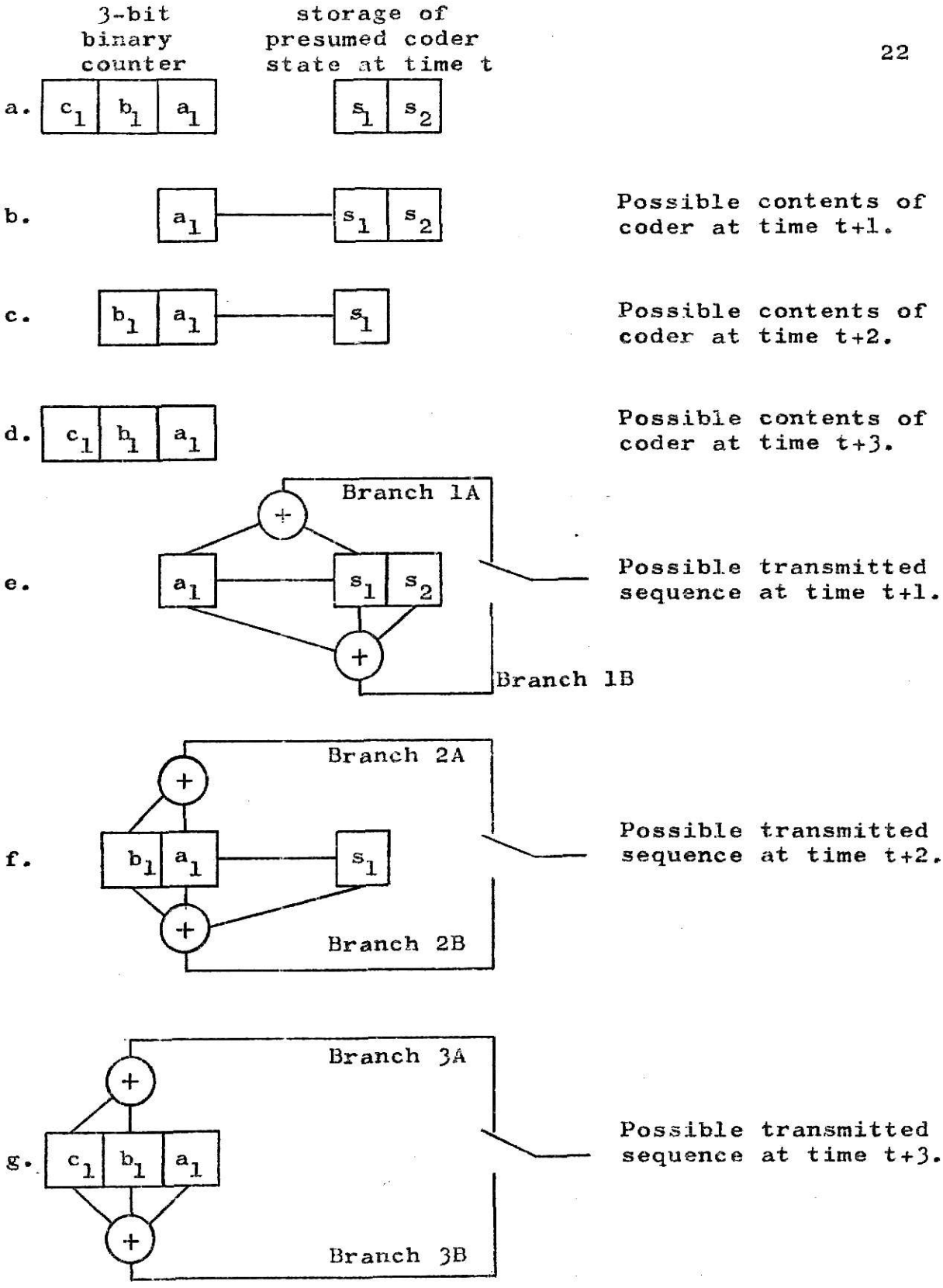
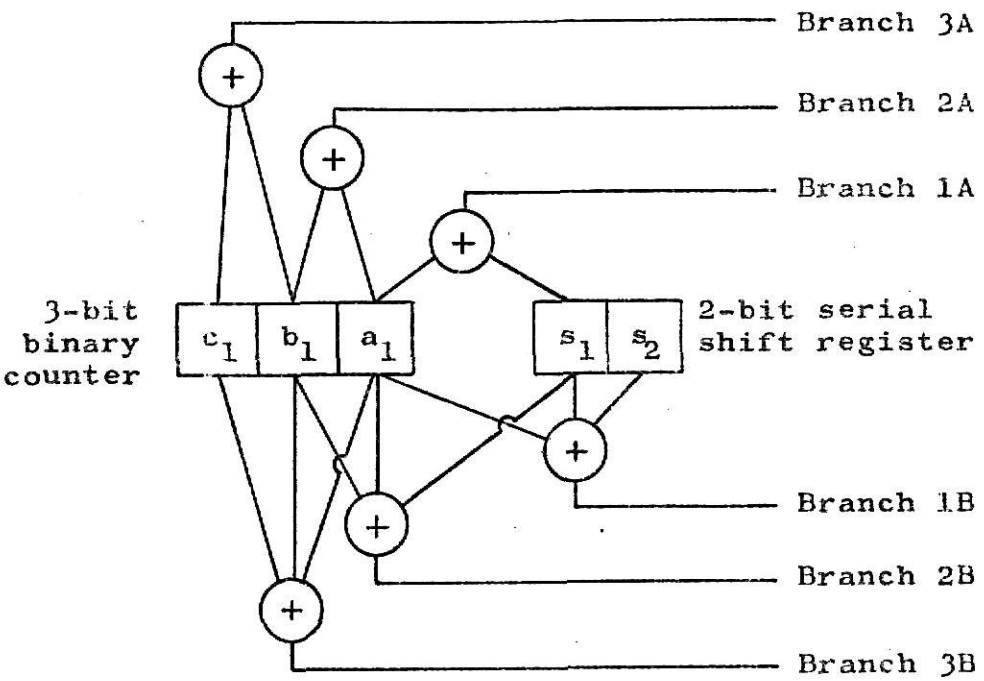


Figure 11. Development of Scheme for Generating All Possible Trellis Paths Three Branches Long Emanating from Node Corresponding to a Given State.



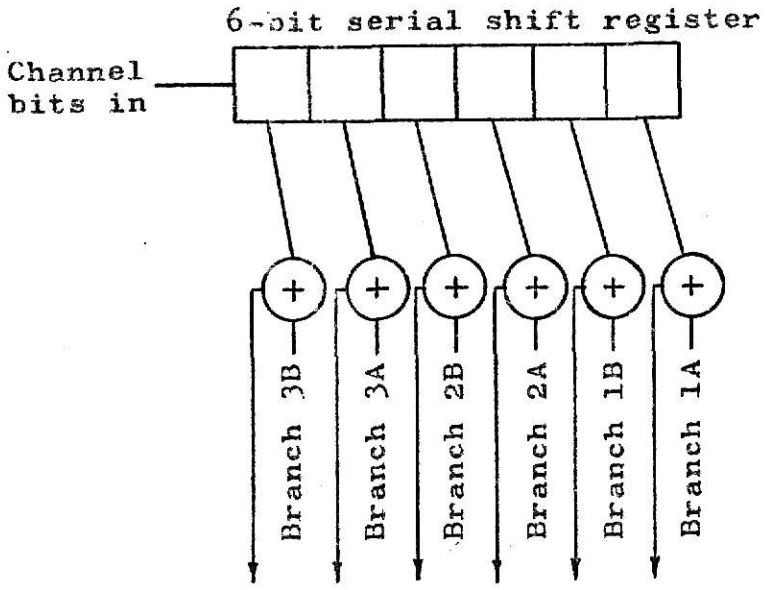


Figure 12. Method of Comparing Generated Trellis Paths with Received Sequence.

stored Hamming distance. If the calculated distance is less than the stored distance, the decoder stores the calculated distance and the bit associated with the first trellis branch of the more recently generated sequence. If the calculated distance is not less than the stored distance, then the stored distance and the stored bit are not bothered. In either case, the process is repeated until all eight possible sequences are generated. At the point where all sequences have been compared with the received sequence, the stored distance is the smallest distance. The stored bit is the bit associated with the first trellis branch of the path having that distance, and is thus outputted as the decoded bit. This bit is also responsible for determining the next state. The decoder goes through the entire process again during the next clock cycle of the transmitter, and continues as long as channel symbols are received.

The basic design of the decoder is now complete. Timing needs to be accomplished; and, although the decoder clock and the timing circuitry constitute a significant portion of the hardware, the design is straightforward once the sequence of events taking place within the decoder is defined. Figure 13 gives a simplified logic diagram of the decoder.

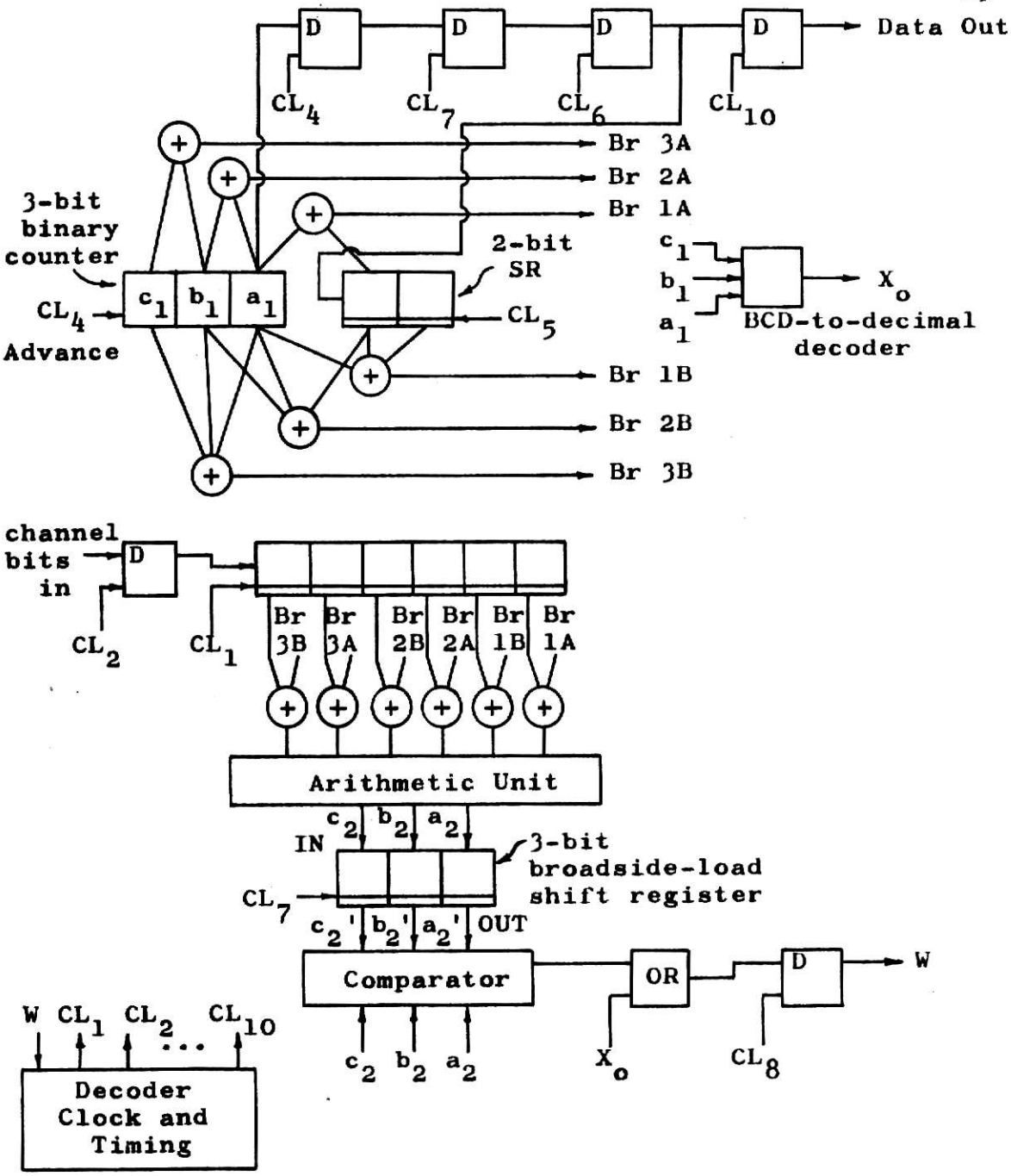


Figure 13. Simplified Logic Diagram for Decoder.

CHAPTER IV

HARDWARE IMPLEMENTATION

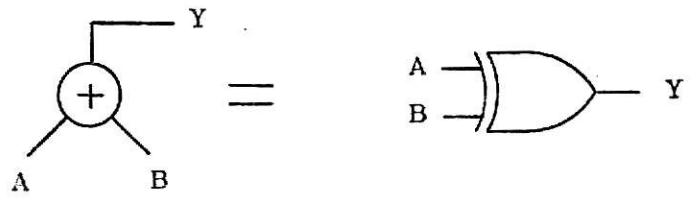
I. DECODER IMPLEMENTATION

Although only a basic logic block diagram for the decoder has been presented, the task of implementation requires little more effort with the availability of medium-scale integration (MSI) in all of the popular logic families. The 54/74 series TTL family is chosen for this decoder, but similar MSI logic blocks exist in complementary symmetry metal-oxide-semiconductor (CMOS) and emitter-coupled logic (ECL).

A two-input modulo-2 adder is nothing more than an exclusive-OR gate, which is shown with its truth table in Figure 14a. Modulo-2 adders of three or more inputs can be constructed by connecting exclusive-OR gates in series. Three-input and four-input modulo-2 adders are shown in Figure 14b and c, respectively.

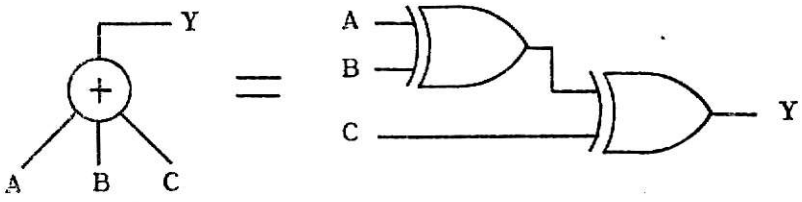
Shift registers, both serial and parallel-in/parallel-out, are available as MSI packages; however, it is generally less expensive to build a two- or three-stage register from flip-flops. Figure 15 shows the implementation of the two-stage serial shift register and the three-stage parallel-in/parallel-out shift register used in the decoder.

The binary counter, the BCD-to-decimal decoder, and the

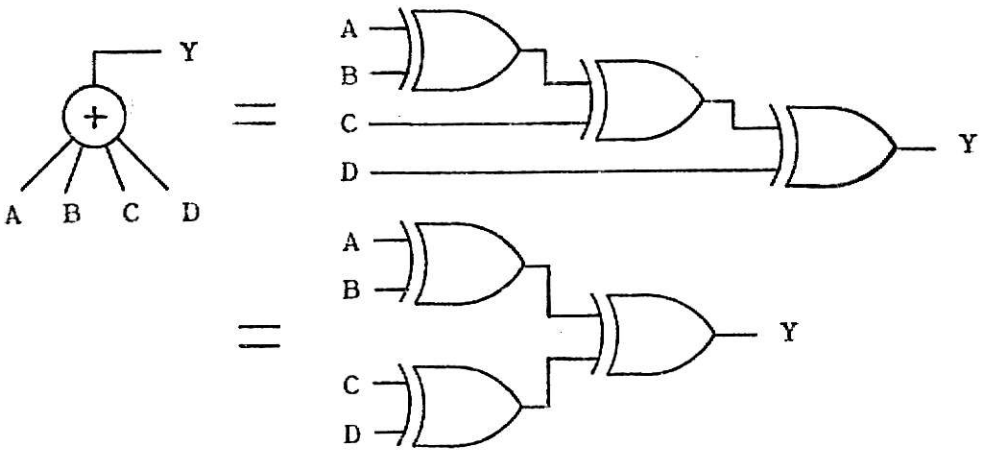


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

a. Two-input modulo-2 adder, exclusive-OR gate and its truth table.

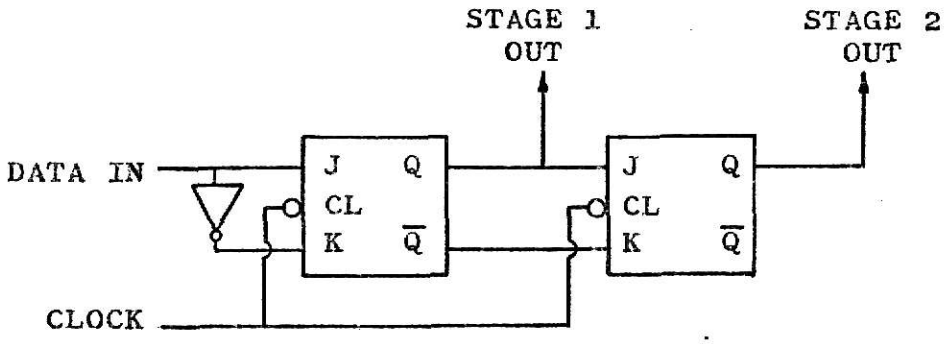


b. Three-input modulo-2 adder.

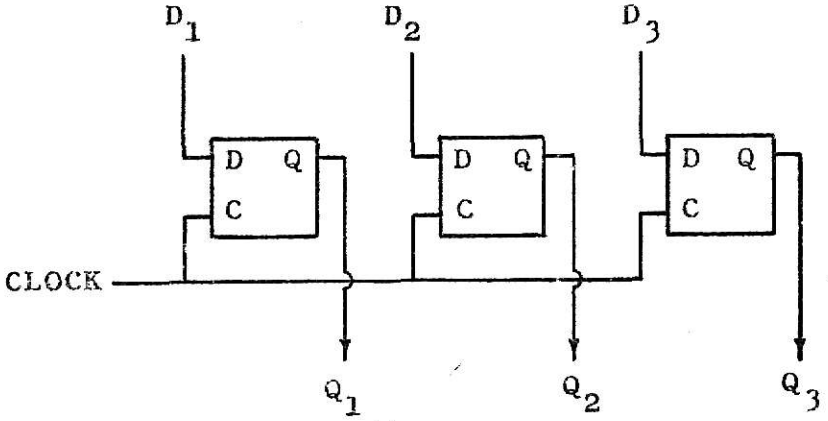


c. Four-input modulo-2 adder.

Figure 14. Implementation of Modulo-2 Adders.



a. Two-stage serial shift register using hex-inverter and and two J-K flip-flops.



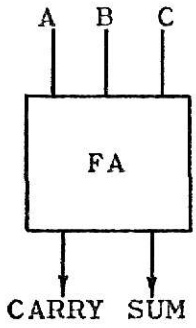
b. Three-stage parallel-in/parallel-out shift register using three D flip-flops.

Figure 15. Shift Register Implementation.

digital comparator are available in MSI. The arithmetic unit, although not available in its required form as a single package, can be constructed using two MSI full adders and one MSI binary adder. A full adder has three inputs and an output consisting of a SUM and a CARRY. Figure 16a gives the truth table for the full adder, from which it can be seen that the output is the two-bit binary sum of the number of logical ones at the inputs of the adder. Since in the decoder there are six inputs to the arithmetic unit, two full adders are utilized and their outputs used as inputs to a binary adder. A binary adder has as its output the sum of two binary numbers which are its inputs. Figure 16b shows the implementation of the arithmetic unit using the full adders and the binary adder. The 54/74 series full adder has an inverted CARRY output which must be followed by an inverter to provide the actual full adder function as defined in Figure 16a.

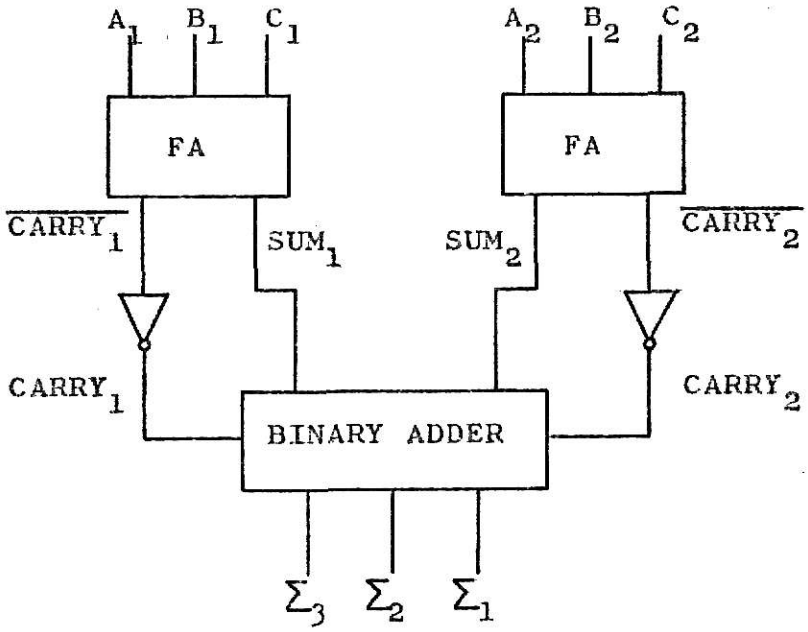
Figure 17 presents a more detailed logic diagram of the decoder than does Figure 13: enough information is given that the decoder, less the timing circuitry, can be constructed with the only additional reference being a 54/74 series TTL product specifications manual.

The required timing circuitry is defined most easily and tersely by use of the timing diagram in Figure 18. Note is made beside each clock pulse as to whether it is free-running or gated. Of the many methods of deriving



A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

a. The full adder and its truth table.



b. Implementation of the arithmetic unit.

Figure 16. The Arithmetic Unit.

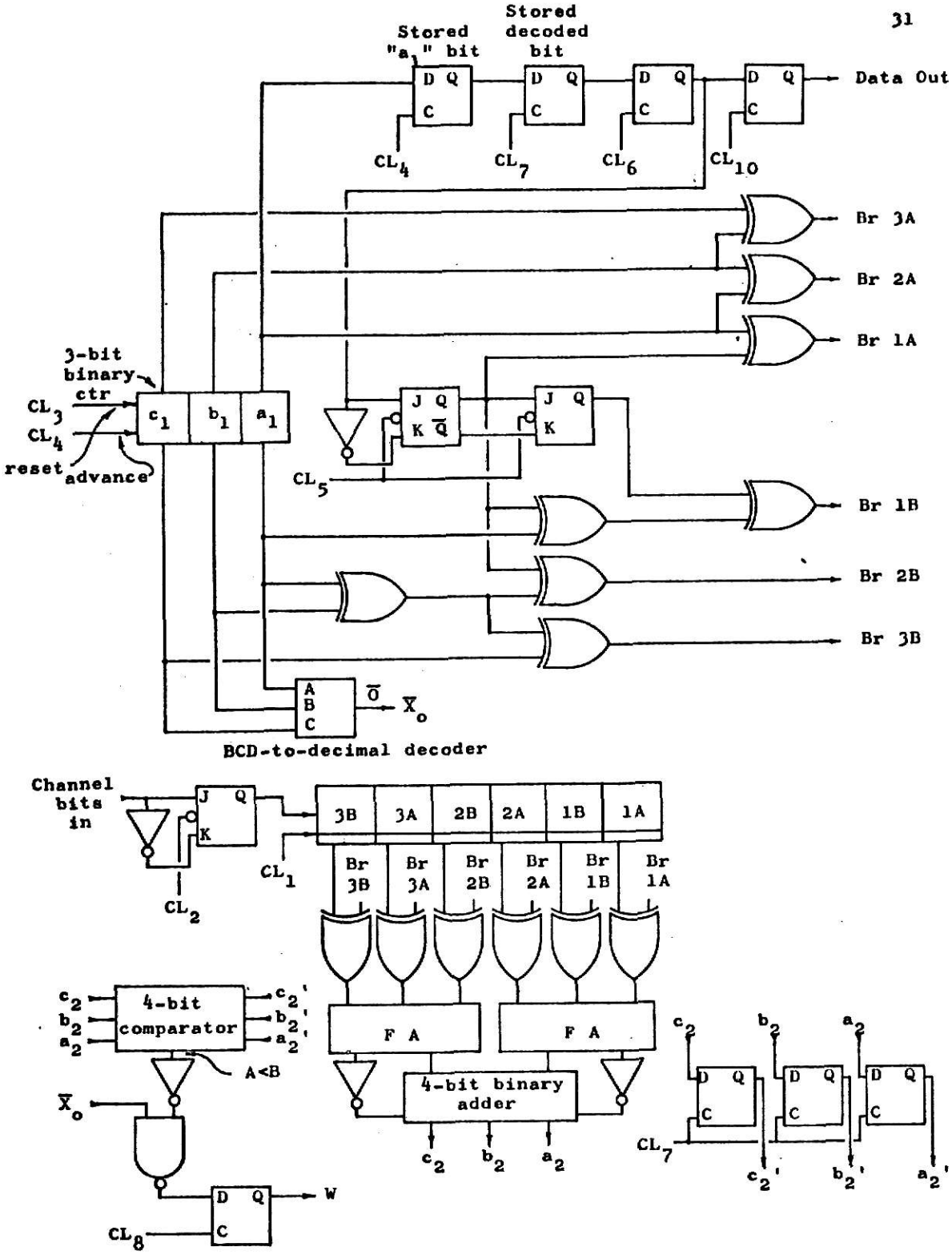


Figure 17. Logic Diagram for Decoder (Less Decoder Clock and Timing Circuitry).

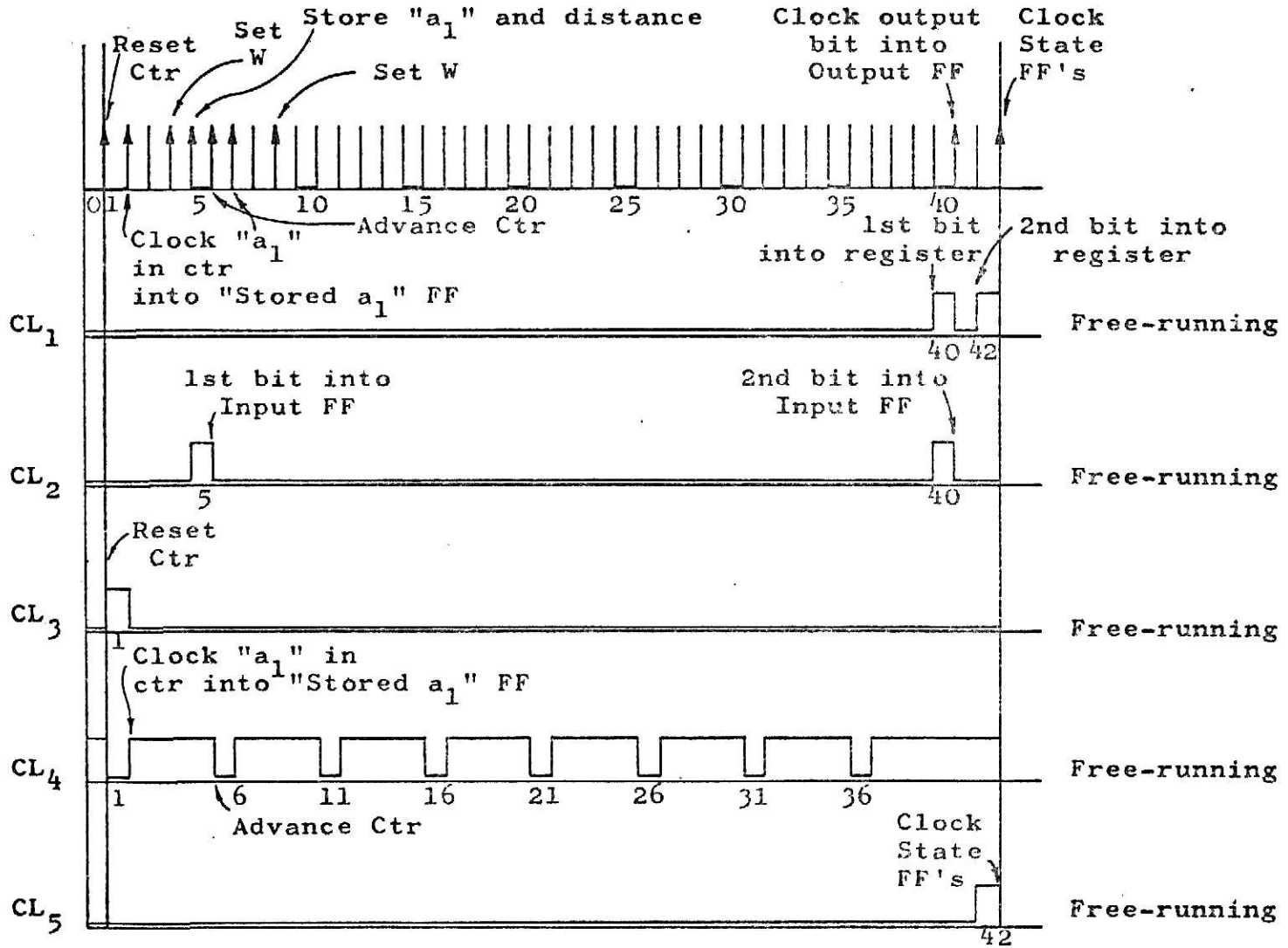


Figure 18. Decoder Timing Diagram (Partial).

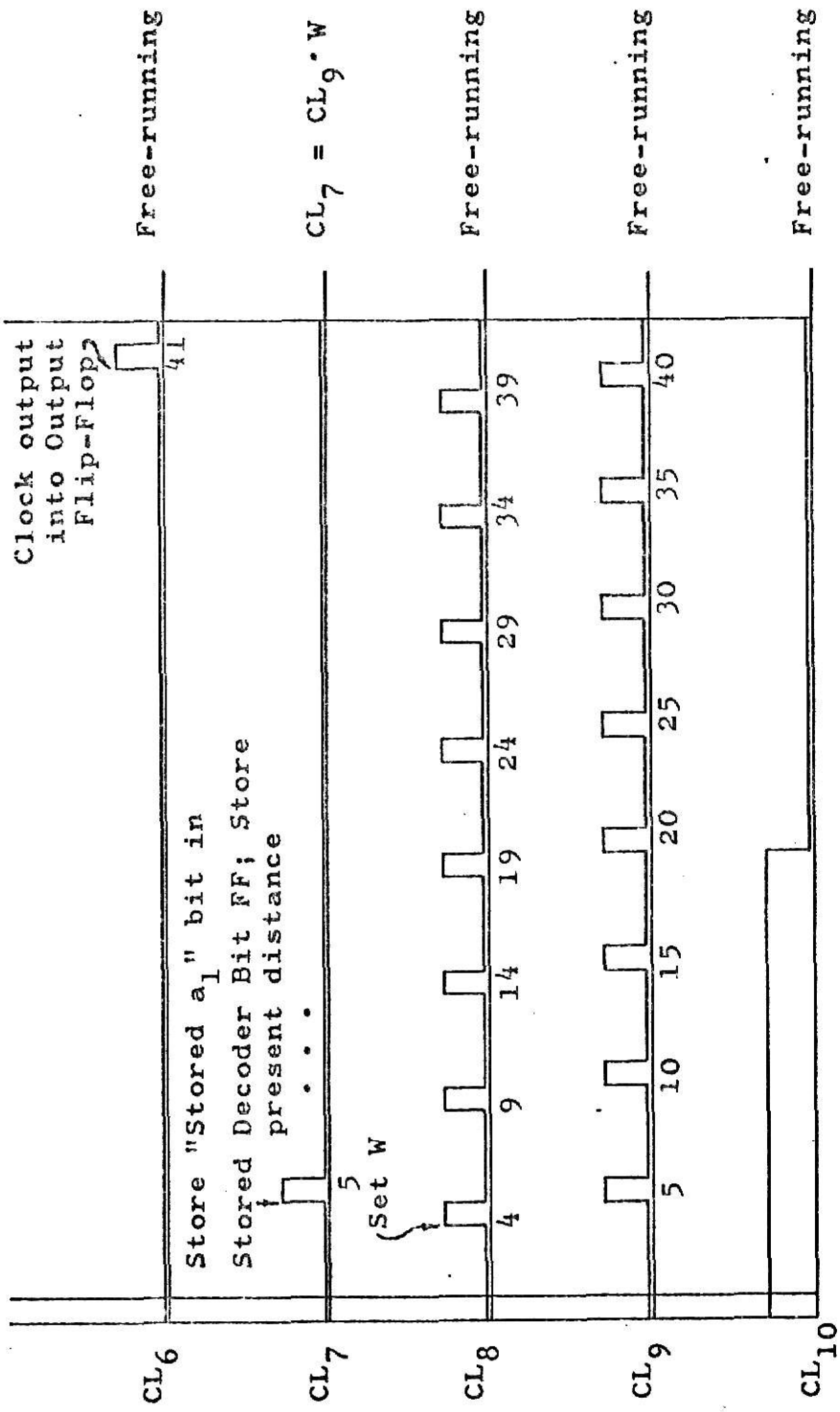


Figure 18 (continued). Decoder Timing Diagram (Partial).

the required timing pulses, the one shown in Figures 19 and 20 is perhaps the best for this situation since the timing is rather complicated. Changes in timing circuitry are easily effected when the counter/decimal decoder combination is used.

An important consideration is that of the maximum operating speed of the decoder. The largest propagation delay occurs between the CL_4 and the CL_8 pulses, and is given by the following:

$$\begin{aligned}\text{Propagation delay} = {} & \text{Counter set-up time} \\ & + 3*t_{pd}(\text{exclusive-OR gates}) \\ & + t_{pd}(\text{full adder}) \\ & + t_{pd}(\text{binary adder}) + t_{pd}(\text{comparator}) \\ & + t_{pd}(\text{inverter}) + t_{pd}(\text{NAND gate})\end{aligned}$$

The worst-case propagation delay is approximately 450 nsec. with 54/74 TTL and there are three decoder clock cycles between the ADVANCE COUNTER and the SET W pulses. Therefore, the maximum decoder clock frequency is 6.7 MHz. The decoder clock goes through forty-three cycles for each decoded bit, thus restricting the information bit rate to 156 kbits/sec.

Figure 21 shows the decoder, which is assembled on a 4" X 6" card with standard 22-terminal edge connector.

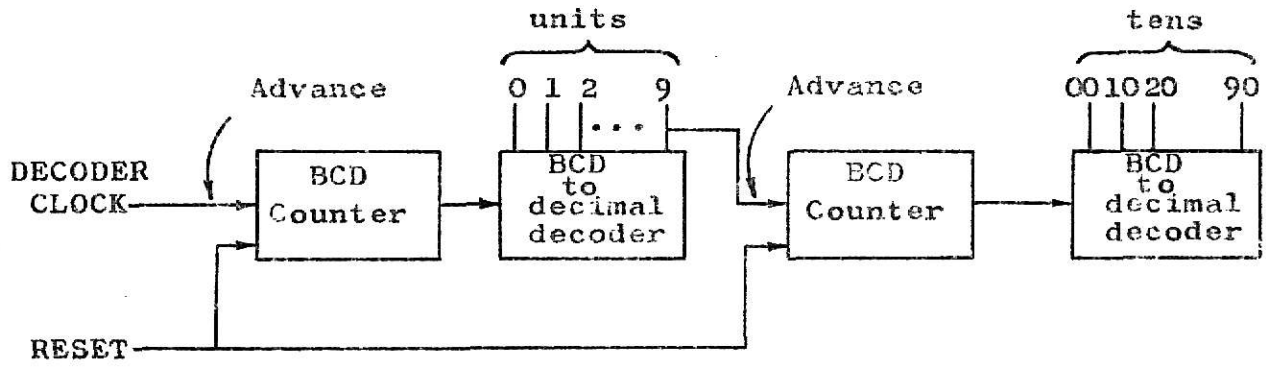


Figure 19. Timing Circuitry (Counter and Decimal Decoder)

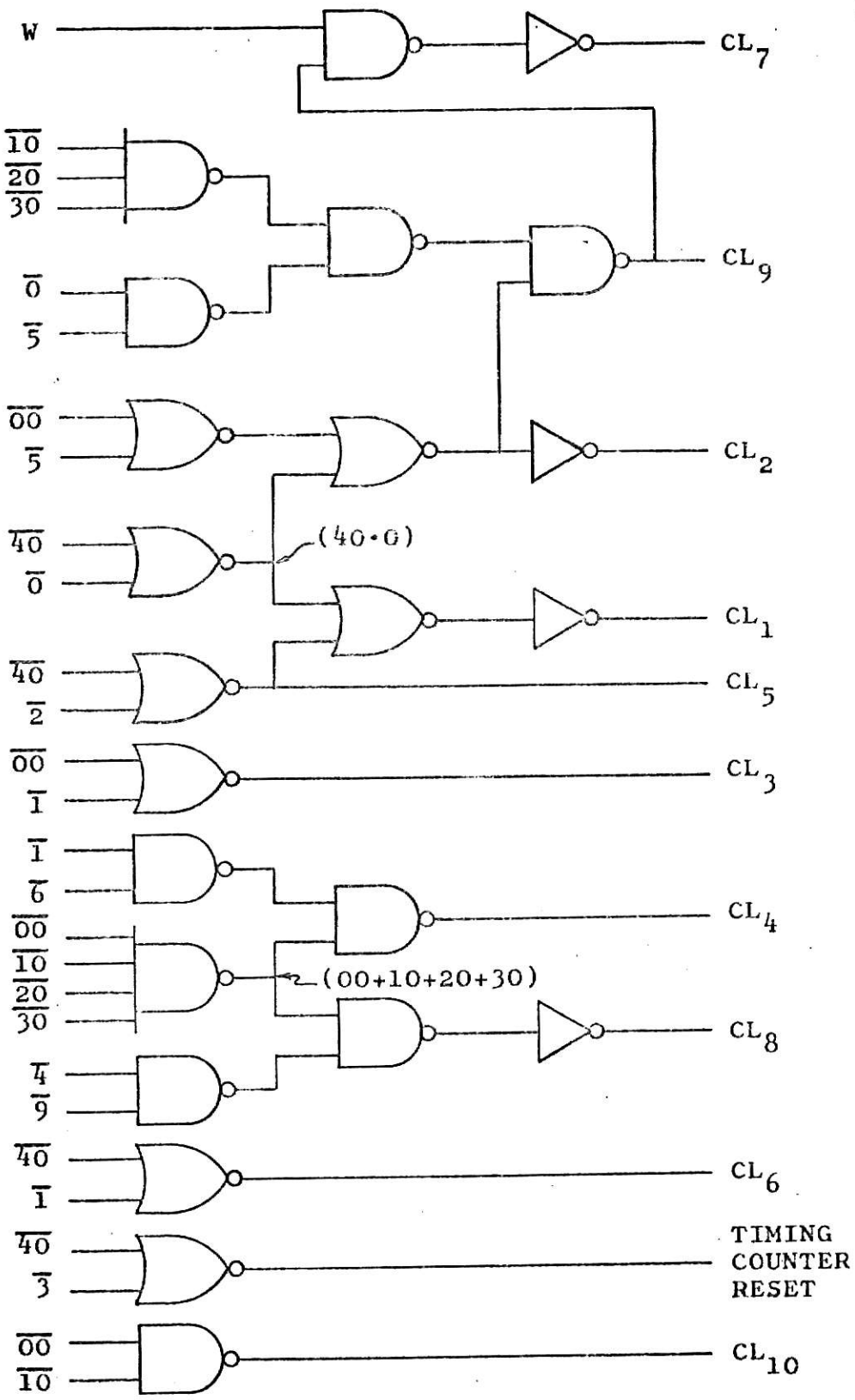


Figure 20. Timing Circuitry (Gating).

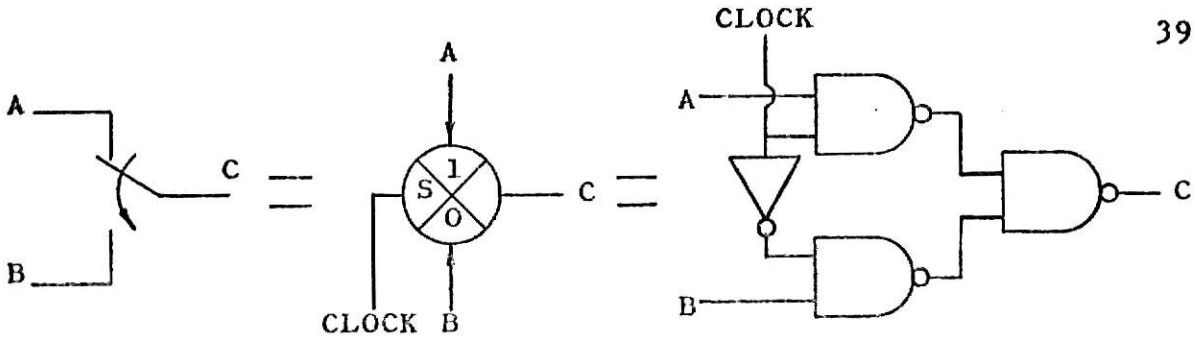
THIS BOOK CONTAINS SEVERAL DOCUMENTS THAT ARE OF POOR QUALITY DUE TO BEING A PHOTOCOPY OF A PHOTO.

THIS IS AS RECEIVED FROM CUSTOMER.

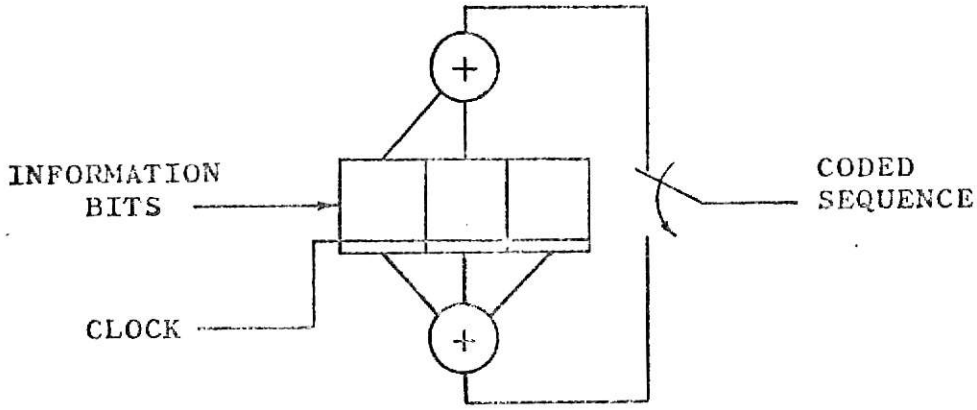
Figure 21. Photograph of Minimum-Distance Feedback Decoder.

II. CODER IMPLEMENTATION

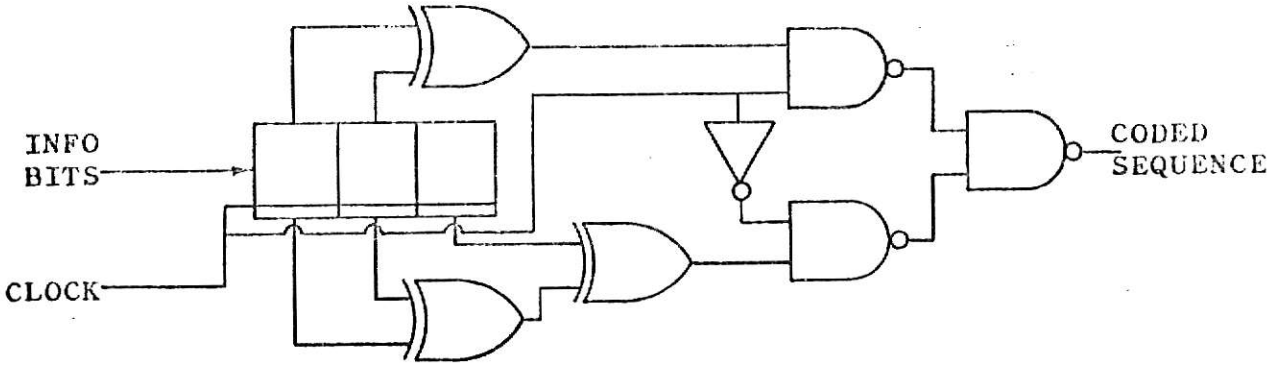
Coder implementation is straightforward, with only a three-stage shift register, two modulo-2 adders, and a commutating switch being needed. The commutating switch for a rate ½ code is simply a data switch, which is shown in Figure 22a. Only three TTL packages are required for the coder.



a. Data switch.



b. Coder.



c. Implementation.

Figure 22. Implementation of K=3, Rate ½, $\begin{matrix}110\\111\end{matrix}$ Convolutional Coder.

CHAPTER V

EVALUATION AND AN ADDITIONAL CONSIDERATION

I. PERFORMANCE

Table I describes some possible error sequences and states the results of the decoding action. The results are worst-case, which occur when the information sequence is the all-ones sequence. The algorithm implemented chooses the trellis path of minimum distance from the received sequence, and, if more than one path has the smallest distance associated with it, the decoder picks the uppermost path. One-error and two-error sequences are listed; two channel errors spaced more than six bits apart are decoded properly and therefore are not listed. Reliable decoding cannot be expected with three or more closely-spaced channel errors. As can be noted from the chart, channel error position has an effect on the decoding action. For example, two channel errors occurring in the same branch produce two decoding errors, while proper decoding is obtained if each of the two adjacent channel errors occurs in a different branch. As can be seen, error-free decoding can be expected if channel errors occur singly and at intervals of not less than six channel bits. Examination of the trellis reveals that, at most, five branches of correct channel symbols will bring the

TABLE I

WORST-CASE DECODER PERFORMANCE ($\begin{smallmatrix}110\\111\end{smallmatrix}$ CODE)

	ERROR SEQUENCES										DECODE PROPERLY ?	NUMBER OF DECODE ERRORS
	X	X	X	X	X	X	X	X	X	X		
One error	1	0									YES	
	0	1									YES	
Two errors —												
Space 0	1	1									NO	2
	0	1	1	0							YES	
Space 1	1	0	1	0							YES	
	0	1	0	1							YES	
Space 2	1	0	0	1							YES	
	0	1	0	0	1	0					NO	2
Space 3	1	0	0	0	1	0					NO	2
	0	1	0	0	0	1					NO	3
Space 4	1	0	0	0	0	1					NO	3
	0	1	0	0	0	0	1	0			YES	
Space 5	1	0	0	0	0	0	1	0			YES	
	0	1	0	0	0	0	0	1			YES	

"1" denotes channel error on specified bit.

"0" denotes occurence of no channel error on specified bit.

decoder to the correct trellis path.

Different decoding action results when different codes are used. There are quite a few non-trivial K=3 codes, and the decoder can be modified to accept different codes of the same constraint length simply by changing the modulo-2 adder connections to the counter and the register storing the decoder state. There are some codes which are catastrophic when used with this decoder, a particular one being the $\frac{101}{111}$ code. It is possible for the decoder, when using this code, to follow an incorrect path and decode incorrectly indefinitely, even when no channel errors occur. The problem can be remedied by modifying the decoder to look ahead seven branches at a time instead of only three, but justification of the additional expense and effort is questionable. Simulation is the simplest method of determining the decoding action on different codes.

Decoder performance with the $\frac{110}{111}$ code is not enhanced noticably by increasing the number of branches looked at by the decoder during each cycle. Rather, code rate must be decreased or constraint length increased. Increasing the constraint length of the coder is perhaps the more desirable since code efficiency is not affected; however, the decoder must go through more steps during each decoding cycle since the trellis would contain more branches.

II. DECODER SYNCHRONIZATION

An important consideration that has not yet been addressed is that of decoder synchronization. The only level of synchronization required is that of node synchronization, i. e., the first and second parity streams must be identified but the data need not be blocked. One method of assuring proper decoding is to send each parity stream over a separate subchannel. If both streams are sent over the same channel, proper decommutation must occur at the receiver. One way to accomplish this is to employ a device which monitors the shift register that stores the Hamming distance. If the distance differed from zero an excessive number of times in a given interval, the decoder would switch synchronization positions. The problem of synchronization would be non-existant also if the receiver could reconstruct the clock in such a manner that the clock retained its proper phase.

CHAPTER VI

CONCLUSION

A convolutional feedback decoder has been designed and implemented, utilizing an algorithm based on the fundamental description of feedback decoding. While threshold feedback decoding is practical only when codes with certain restrictions are used, the minimum-distance feedback decoder described herein can be used with almost any convolutional code of a given constraint length. However, a threshold decoder having error-correcting ability similar to that of the minimum-distance feedback decoder is more economical to construct. The threshold decoder exemplified requires only twelve TTL packages while the minimum-distance decoder requires 29 TTL packages, many of which are MSI.

With the basic decoder defined, modifications can be made with relatively little difficulty. Three such modifications are mentioned.

1. The input flip-flop design and timing can be altered in such a manner that the decoder clock can be built into the decoder and set at a fixed frequency which is independent of the decoding frequency.

2. Maximum decoder speed can be increased significantly in a straightforward manner. An approximately eightfold increase in maximum decoding rate can be effected by

replacing the three-bit binary counter by eight fixed three-bit numbers for 000 to 111, inclusive. Eight sets of modulo-2 adders, eight arithmetic units, seven comparators and additional gating would also be necessary. The decoder timing circuitry would, on the other hand, become simpler since more operations would be accomplished in parallel.

3. While the decoder described is capable of correcting single errors spaced at least five channel bits apart, interleaving and deinterleaving of data can be accomplished to make this basic decoder effective on burst error channels.

Table II lists the specifications of the constructed minimum-distance feedback decoder.

TABLE II

S P E C I F I C A T I O N S

MINIMUM-DISTANCE FEEDBACK DECODER

Constraint Length (in information bits)	Three bits.
Rate	One-half (100% bandwidth expansion)
Encoder-Decoder Delay	Four bit times.
Error-Correcting Capability	Reliable decoding occurs on single channel errors spaced at least five channel bits apart.
Type of Data Accepted	Hard quantized.
Synchronization	External. Provided by channel, receiver, or additional circuitry used in conjunction with decoder.
Decoder Clock	External.
Maximum Frequency of Decoder Clock	6.7 MHz.
Maximum Decoder Speed (information bit decoding rate)	150 kHz. (Decoder clock frequency must be larger than 43 times the decoding frequency but smaller than 82 times the decoding frequency.)
Interface	All clock and I/O signals are at standard TTL levels. Each decoder input is equivalent to one standard 54/74 series TTL gate load. All connections are made by means of 22-terminal edge connector on decoder printed circuit board.
Mechanical	Decoder is constructed on 4" X 6" printed circuit board.
Power Required	5.0 VDC @ 2 amps.
Limits on Operating Temperature	0°C to 70°C.

BIBLIOGRAPHY

1. Gallager, Robert G., Information Theory and Reliable Communication (John Wiley and Sons, Inc., New York, 1968).

2. Lin, Shu, An Introduction to Error-Correcting Codes (Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1970).

3. Linkabit Corp., Convolutional Codes: The Key to Effective Error Control and Efficient Communication (1970).

4. Lucky, R. W., J. Salz, and E. J. Weldon, Jr., Principles of Data Communication (McGraw-Hill, Inc., New York, 1965).

5. Wozencraft, John M., and Irwin Mark Jacobs, Principles of Communication Engineering (John Wiley and Sons, Inc., New York, 1965).

DESIGN AND IMPLEMENTATION OF A K=3, RATE ½, $\begin{matrix}110\\111\end{matrix}$ MINIMUM-DISTANCE FEEDBACK DECODER

by

STEPHEN ALEXANDER DYER

B. S., Kansas State University, 1973

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment

of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1974

Convolutional encoding and feedback decoding of convolutional codes are discussed. A feedback decoder which utilizes the basic procedure of searching the code trellis is described and a method of mechanizing this procedure is given. Details of a complete design and hardware implementation are presented for a constraint length three decoder which has the ability to correct single-bit channel errors. Suggestions are made for increasing decoder operating speed and error-correcting capability.