On Tail Biting Convolutional Codes

HOWARD H. MA, STUDENT MEMBER, IEEE, AND JACK K. WOLF, FELLOW, IEEE

Abstract—**In this paper, we introduce generalized tail biting encoding as a means to ameliorate the rate deficiency caused by zero-tail convolutional encoding. This technique provides an important link between quasi-cyclic block and convolutional codes. Optimum and suboptimum decoding algorithms for these codes are described and their performance determined by analytical and simulation techniques.**

I. Introduction

BOTH block and convolutional codes have been applied successfully to the transmission and storage of digital data over noisy communication channels and storage media [1], [2]. Although a great deal of theory has been developed for block codes, the search for good convolutional codes has largely been the task of the computer. There are obvious advantages to establishing a common framework for these two classes of codes. Unfortunately, there has been little success in developing a general theory that applies to both convolutional and block codes.

Two straightforward methods for converting convolutional codes to block codes are described below. For ease of understanding we describe these methods in terms of a binary $(n, 1, m)$ convolutional code (of rate $1/n$ and constraint length[1] m), although the techniques apply to any convolutional code.

Method 1—Direct Truncation: For L a positive integer, we take as the code words in our block code all sequences of length Ln produced by inputting into the encoder all binary information sequences of length L. The resultant code is an (Ln, L) block code of rate $1/n$. However, the code has the disadvantage that there is little if any error protection afforded to the last information digits inputted into the encoder.

Method 2—Zero Tail: For L a positive integer, we take as the code words in our block code all sequences of length $(L + m)n$ produced by inputting into the encoder a binary sequence of length L *followed by m zeros.* The resultant code is an $((L + m)n, L)$ block code of rate $(1/n)(L/(L + m)) = (1/n)(1 - (m)/(L + m))$. The term $m/(L + m)$ is called the *rate loss* and is due to the zero tail.

Paper approved by the Editor for Coding Theory and Applications of the IEEE Communications Society. Manuscript received February 20, 1985; revised August 1, 1985. This work was supported by the U.S. Air Force Office of Scientific Research under Grant AFOSR-82-0061. The work of H. H. Ma was performed in partial fulfillment of the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst.

H. H. Ma is with the MITRE Corporation, Bedford, MA 01730.

J. K. Wolf is with the Department of Electrical Engineering and Computer Science, University of California at San Diego, La Jolla, CA 92093.

IEEE Log Number 8406878.

[1] We define the constraint length (sometimes termed memory order) m of a rate k/n convolutional code as $m = \max_{1 \le i \le k} K_i$, where K_i is the number of *delay elements* that stores the ith input sequence to the encoder.

A less obvious and third method to convert a convolutional code to a block code has been suggested by Odenwalder [3] and Solomon and van Tilborg [4].

Method 3—Tail Biting: For L a positive integer, we consider all code words produced by $(L + m)$ information digits using an encoder in the following fashion. We first initialize the encoder by inputting the last m information bits into the encoder and ignoring the output. We then input all $(L + m)$ information bits into the encoder and take as our code word the resultant $(L + m)n$ output bits. The resulting block code consisting of all code words formed in this fashion is an $((L + m)n, (L + m))$ code of rate $1/n$. The advantage of this scheme over Method 1 is that all information digits are afforded the same amount of error protection if a maximum likelihood decoder is utilized. The disadvantage is that the maximum likelihood decoder is more complex than that for the convolutional code. This issue will be discussed further later in this paper.

In this paper we give a new method which we call generalized tail biting (or GTB) for converting convolutional codes to block codes with no rate loss. GTB includes as special cases both Methods 1 and 3. The basic idea is very closely related to that of Method 3 but offers a wider selection of codes from which to choose. As we shall see, instead of initializing the encoder entirely with information digits, we use a combination of information digits and zeros to initialize the encoder. The more zeros we use, the simpler will be the maximum likelihood decoder, but the poorer will be the performance of the code. When we use all zeros our method reduces to Method 1, while when we use all information digits our method reduces to Method 3.

In another paper [5] we have explored how GTB can be used to generate a class of unequal error protection codes that can be decoded using majority logic decoding. Here we explore the concept of GTB in general. We show that the resultant codes are quasi-cyclic block codes, explore several decoding strategies including maximum likelihood and suboptimal decoders, and evaluate the performance of these codes by both analysis and simulation.

II. Generalized Tail Biting of Convolutional Codes

We first give a description of generalized tail biting used in conjunction with an (n, k, m) convolutional code over GF(2). (The extension to codes over GF(q) is straightforward.) Consider the encoder shown in Fig. 1. Note that we show k shift registers of equal length m. If some of the registers were to have length less than m, we could still use this model with some of the tap gains equal to zero. Let L be a positive integer and let m' be a nonnegative integer in the range $0 \le m' \le m$. We use the encoder to encode $(L + m')k$ information digits in a manner to be described next. We think of the input as being divided into k data streams each of length $(L + m')$. We then initially load the ith shift register with the *last* m'

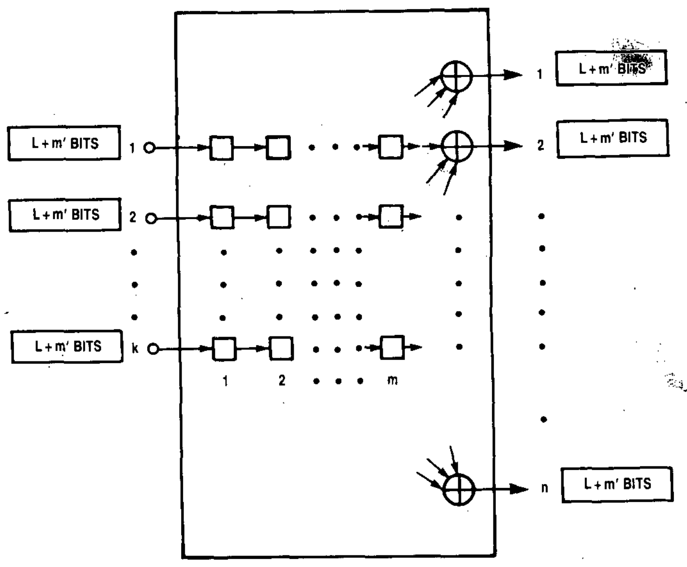


Fig. 1. The rate k/n, constraint length m, binary convolutional encoder with an equivalent block length of $(L + m')$. Note: the input connections to the jth exclusive-OR gate for the jth output are specified by $g_{i,l}^{(j)}$; $i = 1, 2, \cdots, k$; $l = 0, 1, \cdots, m$; $j = 1, 2, \cdots, n$.

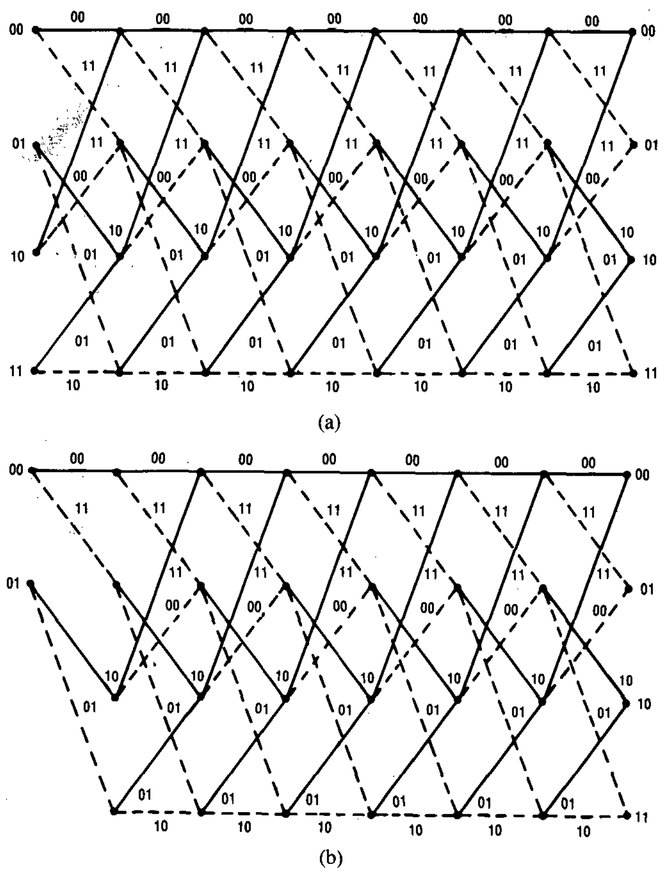


Fig. 2. (a) A rate 1/2 FTB trellis with $L = 5$ and $m = 2$. (b) A rate 1/2 PTB trellis with $L = 6$, $m' = 1$, and $m = 2$.

data bits in the ith input sequence and with $(m - m')$ zeros, $i = 1, 2, \cdots, k$. The scheme would work equally well whether we put the zeros in the left or right of the register, but for definiteness we assume the zeros are in the *rightmost* stages. We then input the entire $(L + m')k$ information digits in the usual manner and output $(L + m')n$ digits. This encoding procedure results in an $((L + m')n, (L + m')k)$ block code.

It should be noted that this scheme results in no fractional rate loss over the original convolutional code. Furthermore, it should be clear that if $m' = 0$ we have direct truncation (Method 1), while if $m' = m$ we have tail biting (Method 3). For future reference we will refer to the situation where $0 < m' < m$ as partial tail biting (or PTB) and the situation when $m' = m$ as full tail biting (or FTB).

III. Maximum Likelihood Decoding

In 1967, Viterbi [6] introduced an algorithm for decoding convolutional codes which initially only was thought to be asymptotically optimum but was later shown to be a maximum likelihood algorithm. Although there are a number of important aspects to this algorithm (which has become known as the Viterbi algorithm), the most important for our purposes is that the code words are represented by paths in a "trellis" and the algorithm finds the path in the trellis that corresponds to the maximum likelihood code word.

When this algorithm is applied to block codes formed from convolutional codes by the "zero tail" method (Method 2), one looks for the best path in the trellis that starts in the zero state and ends in that state at a depth of $(L + m)$ in the trellis. When this algorithm is applied to block codes formed from convolutional codes by the direct truncation method (Method 1), one looks for the best path that starts in the zero state and ends in any state at a depth of L in the trellis. The complexity of the algorithm in these two cases is the same, since the Viterbi algorithm works in a manner such that there is no more difficulty in finding the best path to all states at a given depth in the trellis as in finding the best path to one state at that depth.

The situation is different, however, when we apply the algorithm to block codes formed from convolutional codes by full tail biting (Method 3) or by generalized tail biting. We first consider the case of full tail biting. The algorithm must find the best path in the trellis subject to the constraint that the path starts and ends in the same state (the ending state being at a depth of $(L + m)$ in the trellis). However, this common starting and ending state can be any one of the 2^{mk} states of the trellis. It appears that the most straightforward method of accomplishing this decoding is to run the Viterbi algorithm a total of 2^{mk} times, once for each possible starting state. This may be an intolerable increase in complexity, so that we may wish to explore simple suboptimum decoding algorithms. Before doing this, however, we consider how to decode block codes formed from convolutional codes by partial tail biting.

When PTB is utilized, the starting state of the encoder can only be in one of $2^{m'k}$ states. For each starting state we know the encoder ends in one of $2^{((m-m')k)}$ states. Thus, we now need to run the Viterbi algorithm a total of $2^{m'k}$ times, once for each possible starting state. If $m' = 0$ we have direct truncation and we need to run the Viterbi algorithm only once, while if $m' = m$ we have full tail biting and we need to run the algorithm a full 2^{mk} times.

When compared to FTB codes of the same constraint lengths, PTB codes provide less error protection for certain information digits, but with a simpler decoder. On the other hand, PTB codes provide better error protection than that for direct truncation codes of the same constraint lengths, but with a more complex decoder. Therefore, the PTB scheme affords an added code design dimension between FTB and direct truncation schemes to trade error protection performance against complexity.

Fig. 2 illustrates the code trellises for full tail biting codes of $m = 2$ and partial tail biting codes of $m = 2$ and $m' = 1$.

Four Viterbi trials are needed in the first case, while two Viterbi trials are needed in the second case.

IV. Suboptimum Decoding for Full Tail Biting Codes

The major disadvantage of the maximum likelihood decoding of full tail biting codes is that we need to perform 2^{mk} "Viterbi trials" for an (n, k, m) code. However, if one can predict the starting state (for example, with the aid of a genie), only one Viterbi trial would suffice. A suboptimum decoding strategy would be to first estimate the starting (and ending) state and then to decode using a single Viterbi trial. The following two algorithms represent two different suboptimum decoding strategies. The first is a probabilistic approach proposed by Bar-David [7] and the second is an algebraic approach proposed here. Both algorithms estimate the common starting-ending state by making use of the constraint that the starting state is the same as the ending state. We discuss first the Bar-David approach. The basic steps are

0) Choose an arbitrary starting state.

1) Decode using a maximum likelihood decoder which finds the best path from that starting state to all possible ending states.

2) Check the decoded code word to see if the starting state is the same as the ending state. If yes, stop, otherwise go to step 3.

3) Use the previous ending state as the new starting state. Check if this starting state has been tried before. If yes, go to step 0, otherwise return to step 1.

If the number of trials exceeds 2^{mk}, the decoder simply outputs the best winning path at that time.

An alternate suboptimum decoding algorithm that utilizes the algebraic nature of the FTB codes is called the two-step algorithm. The first step in this algorithm is to obtain an ordered list of the 2^{mk} starting states using an algebraic method called "continued fractions." The second step in this algorithm is to perform Viterbi runs using each entry on the list as their starting state. Details of this two-step algorithm are omitted here and the interested reader is referred to [8] for a complete description.

Simulations were performed to evaluate the code performance of the two suboptimum decoding schemes—the Bar-David algorithm and the two-step algorithm. Fig. 3 and Table I give some of our simulation results for a rate 1/2 code of constraint length 7 used in conjunction with a binary symmetric channel. In Fig. 3, four performance curves are shown. The top two curves display the performance of the two suboptimum schemes discussed here, while the third one shows the performance of the maximum likelihood (ML) scheme for FTB codes. The bottom curve illustrates the error performance of a standard Viterbi decoder for zero-tail convolutional codes. The rate loss factor $m/(L + m)$ results in an additional penalty of about 1 dB (for $L = 27$), which is not accounted for on this curve. The following observations can be made.

1) The error performance degradation of using maximum likelihood decoding of FTB codes (without any rate loss) is about 0.75 dB at a 10^{-4} BER, as compared to the error performance of a standard zero-tail (with rate loss) Viterbi decoder with the same constraint length ($m = 2, \cdots, 8$). The decoding complexity is also increased by a factor of 2^m.

2) The two suboptimum decoding schemes for FTB codes provide tradeoffs between error performance and decoding complexity. For both of these schemes, the bit error probability increases only algebraically in comparison with that of Viterbi FTB decoding. (This follows from the fact that the curves are parallel.) Roughly speaking, it requires half of the Viterbi trials needed for the ML decoding of FTB codes, with a further performance degradation of 0.5–0.65 dB. The savings in complexity become larger when the SNR is higher.

3) The two-step scheme performs better than the Bar-David scheme by about 0.15 dB at a BER of 10^{-4}. This result is to be expected, since the two-step scheme utilizes the algebraic structure of the code to obtain information about the starting state.

4) The savings in decoding complexity of both suboptimum schemes increase when the noise is less severe. For all signal-to-noise ratios of interest, the two-step scheme requires comparable, but fewer, Viterbi trials than those of the Bar-David scheme (see Table I). Furthermore, the decoding complexity of the Bar-David scheme fluctuates more (with the noise conditions) than that of the two-step scheme. For example, only half of the Viterbi trials in the Bar-David scheme are needed for the two-step scheme when $p = 0.045$, but almost all (eight fewer) of the Viterbi trials in the Bar-David scheme are needed for the two-step scheme when $p = 0.029$.

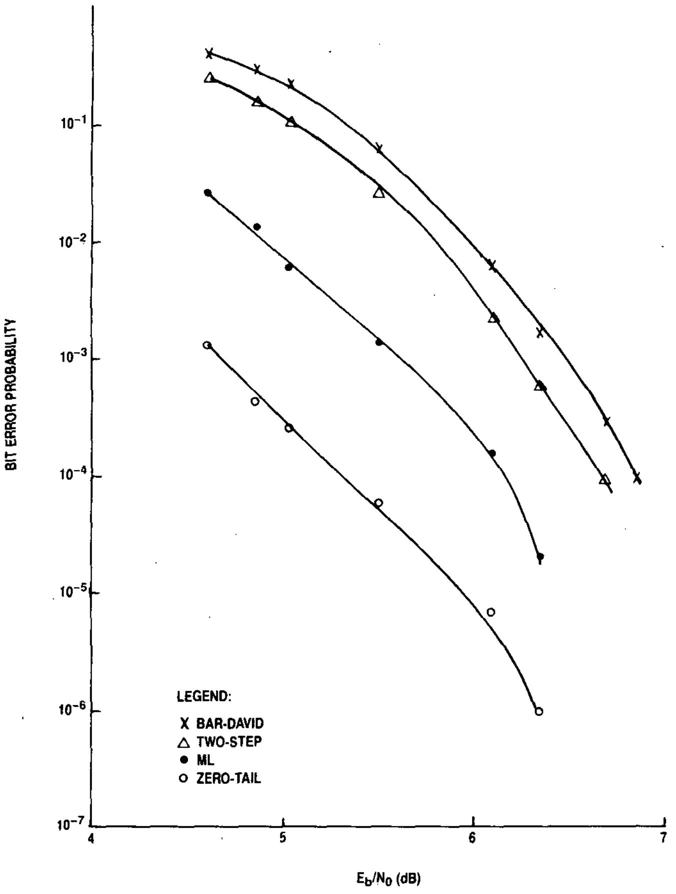


Fig. 3. Error performance of optimum and suboptimum FTB decoding algorithms on a binary symmetric channel ($R_c = 1/2$, $m = 7$, $L = 27$).

V. Quasi-Cyclic Block Codes and Convolutional Codes with Full Tail Biting

In 1979, Solomon and van Tilborg [4] established a connection between quasi-cyclic block codes and convolutional codes. In their paper they demonstrated that the block code that resulted from full tail biting of a rate 1/2 convolutional code was a quasi-cyclic code. Here we provide a proof of this result for rate $1/n$ and $(n - 1)/n$ convolutional codes in a manner somewhat different from the discussion of Solomon and van Tilborg.

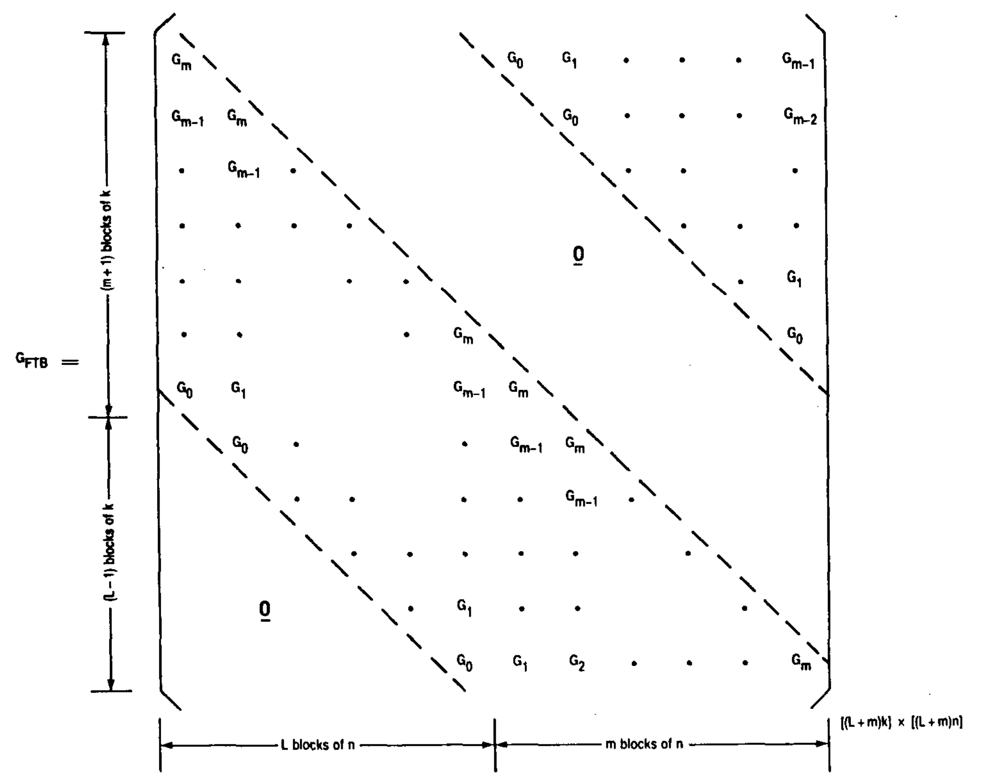
Theorem 1: A rate k/n ($k = 1$ or $n - 1$), constraint length m, FTB convolutional code with information sequence length

TABLE I
SIMULATION RESULTS ($R_c = 1/2$, $m = 7$, $L = 27$)

		ZERO-TAIL VITERBI		VITERBI FTB		BAR-DAVID		TWO-STEP STATE ESTIMATION	
BSC p	(E_b/N_0), dB	BER	VITERBI TRIALS	BER	VITERBI TRIALS	BER	VITERBI TRIALS	BER	VITERBI TRIALS
0.0449	4.609	1.40×10^{-3}	1	2.93×10^{-2}	128	1.52×10^{-1}	116.21	4.17×10^{-2}	67.42
0.041	4.812	4.17×10^{-4}	1	1.63×10^{-2}	128	5.65×10^{-2}	72.10	2.31×10^{-2}	60.17
0.0371	5.032	1.87×10^{-4}	1	6.02×10^{-3}	128	4.21×10^{-2}	67.65	2.01×10^{-2}	58.24
0.0293	5.501	6.01×10^{-5}	1	1.62×10^{-3}	128	5.96×10^{-3}	62.25	3.15×10^{-3}	54.27
0.0215	6.129	7.21×10^{-6}	1	1.71×10^{-4}	128	5.01×10^{-4}	56.79	3.12×10^{-4}	47.11
0.0195	6.298	1.01×10^{-6}	1	2.11×10^{-5}	128	1.04×10^{-4}	47.32	5.21×10^{-5}	41.32

(without tail) L is equivalent to an $((L + m)n, (L + m)k)$ quasi-cyclic block code with period n.

Proof: By taking the scalar matrix approach [9], we obtain the FTB generator matrix from the observation that the m data tail blocks cause the first nL columns of this matrix to have the full $k(m + 1)$ rows characterized by $G_0, G_1, \cdots, G_m$ and the subsequent nm columns to wrap around and create nm end-around cyclic shifts of the sequence $G_0, G_1, \cdots, G_m$ followed by $(L - 1)$ zero blocks of dimension $k \times n$. The resulting FTB generator matrix for a rate k/n, constraint length m, and information sequence length L code is



where

$$G_l = \begin{bmatrix} g_{1,l}^{(1)} & g_{1,l}^{(2)} & \cdots & g_{1,l}^{(n)} \\ g_{2,l}^{(1)} & g_{2,l}^{(2)} & \cdots & g_{2,l}^{(n)} \\ \vdots & \vdots & & \vdots \\ g_{k,l}^{(1)} & g_{k,l}^{(2)} & \cdots & g_{k,l}^{(n)} \end{bmatrix}_{k\times n}$$

Here $g_{i,l}^{(j)}$ is equal to 1 or 0, corresponding to whether the lth stage of the shift register for the ith input contributes to the jth output ($i = 1, 2, \cdots, k; j = 1, 2, \cdots, n; l = 0, 1, \cdots, m$). Since each k-row of the FTB generator matrix is an n-end-around cyclic shift of the other, the code thus generated is a quasi-cyclic code with period n. Q.E.D.

Based on this equivalence for rate 1/2 codes, Solomon and van Tilborg then proceeded to find a wide class of block codes (such as quadratic-residue codes and simple Reed–Solomon codes) that are quasi-cyclic and, thus, can be "FTB" convolutionally encoded and decoded. They also attempted to minimize the required constraint lengths to implement quasi-cyclic codes. We now exploit this equivalence in a different manner to establish a general code design philosophy that applies to both convolutional and block codes. First, we truncate optimum convolutional codes using FTB encoding to produce interesting quasi-cyclic block codes. Second, we take good quasi-cyclic block codes (e.g., the extended Golay (24, 12) code) and see what convolutional codes result.

Table II lists the rate 1/2, FTB-encoded, Odenwalder optimum nonsystematic codes [10] for various L's and m's. Also included in this table are the minimum distances of the best (N, K) block codes with the same parameter. We observe that the best minimum distance for a given N and a given K is usually achieved by an FTB convolutional code with reasonably short constraint length. The minimum distance of the FTB-

TABLE II
SOME $(n(L + m), k(L + m))$ $d_{\min}$ TAIL BITING ODENWALDER CODES

GOOD BLOCK CODES $(N,K)d_{\min}$	$m=2$ $(d_{free}=5)$	$m=3$ $(d_{free}=6)$	$m=4$ $(d_{free}=7)$	$m=5$ $(d_{free}=8)$	$m=6$ $(d_{free}=10)$	$m=7$ $(d_{free}=10)$	$m=8$ $(d_{free}=12)$
	(4,2)2						
(6,3)3	$\underline{\underline{(6,3)2}}$	$\underline{\underline{(6,3)2}}$					
(8,4)4	(8,4)2	(8,4)2	(8,4)2				
(10,5)4	$\underline{\underline{(10,5)3}}$	$\underline{\underline{(10,5)3}}$	$\underline{\underline{(10,5)3}}$	(10,5)2			
(12,6)4*	(12,6)3	(12,6)3	(12,6)3	$\underline{(12,6)4}$	(12,6)2		
(14,7)4*	$\underline{(14,7)4}$	$\underline{(14,7)4}$	(14,7)3	(14,7)3	$\underline{(14,7)4}$	(14,7)2	
(16,8)5*	(16,8)4	(16,8)4	(16,8)4	(16,8)4	(16,8)4	$\underline{(16,8)5}$	(16,8)2
(18,9)6*	(18,9)5	(18,9)5	(18,9)5	(18,9)5	$\underline{(18,9)6}$	(18,9)5	(18,9)4
(20,10)6*	(20,10)5	(20,10)5	(20,10)4	(20,10)5	$\underline{(20,10)6}$	(20,10)4	$\underline{(20,10)6}$
(22,11)7	(22,11)5	$\underline{\underline{(22,11)6}}$	(22,11)4	(22,11)5	(22,11)4	(22,11)4	(22,11)4
(24,12)8*	(24,12)5	(24,12)6	(24,12)5	(24,12)5	(24,12)6	(24,12)5	$\underline{(24,12)8}$
(26,13)7		$\underline{\underline{(26,13)6}}$	$\underline{\underline{(26,13)6}}$	(26,13)5	(26,13)4	(26,13)5	$\underline{\underline{(26,13)6}}$
(28,14)8		(28,14)6	(28,14)6	(28,14)5	(28,14)6	(28,14)6	(28,14)6
(30,15)8			$\underline{\underline{(30,15)7}}$	$\underline{\underline{(30,15)7}}$	(30,15)6	$\underline{\underline{(30,15)7}}$	(30,15)6
(32,16)8*			(32,16)7	(32,16)7	$\underline{(32,16)8}$	(32,16)7	$\underline{(32,16)8}$
(34,17)8			$\underline{\underline{(34,17)7}}$	$\underline{\underline{(34,17)7}}$	(34,17)6	$\underline{\underline{(34,17)7}}$	(34,17)6
(36,18)8*			(36,18)7	$\underline{(36,18)8}$	$\underline{(36,18)8}$	(36,18)5	$\underline{(36,18)8}$
(38,19)8*			(38,19)7	$\underline{(38,19)8}$	(38,19)6	(38,19)7	$\underline{(38,19)8}$
(40,20)9				$\underline{\underline{(40,20)8}}$	$\underline{\underline{(40,20)8}}$	(40,20)7	$\underline{\underline{(40,20)8}}$
(42,21)10				(42,21)8	(42,21)8	(42,21)8	(42,21)8
(44,22)10				(44,22)8	(44,22)8	(44,22)8	(44,22)8
(46,23)10					(46,23)8	$\underline{\underline{(46,23)9}}$	(46,23)8
(48,24)12					(48,24)10	(48,24)9	(48,24)8
.						.	.
.						.	.
(56,28)12						(56,28)10	(56,28)10
.							.
.							.
(68,34)13							$\underline{\underline{(68,34)12}}$

CAPTION:
*-$d_{\min}$ achieved
$\underline{\qquad\qquad}$ achieves $d_{\min}$
$\underline{\underline{\qquad\qquad}}$ achieves $d_{\min}$-1

encoded code is upper bounded by the free distance of the corresponding convolutional code. That is, the minimum distance will not increase after it reaches d_{free} no matter how large L is made. It takes approximately four constraint lengths for that to happen (see Table II). Thus, the code with the largest minimum distance for a given m is about an $(8m, 4m)$ code with $d_{\min} = d_{\text{free}}$. It is to be noted that this class of quasi-cyclic block codes can be soft-decision decoded using the Viterbi algorithm for the corresponding FTB convolutional code.

Quasi-cyclic block codes are usually in systematic form. Thus, we study only the relationship between these codes and good systematic convolutional codes. If we take the original columns of the quasi-cyclic (or FTB) generator matrix in a different order, an echelon canonical generator matrix $[I|A]$ results. For rate 1/2 quasi-cyclic codes with period 2, this equivalent class of $(2K, K)$ block codes with generator matrix of the form $[I|A]$ are called double circulant (DC), since both the identity matrix I and A are $K \times K$ circulant matrices. DC codes have very good minimum distance properties, but are difficult to decode using algebraic techniques. Table III lists some good DC codes with their achieved minimum distances where G is the octal representation of the top row of the matrix A.

Theorem 1 states that the entire class of DC codes are equivalent to rate 1/2 FTB convolutional codes with parity tap connections specified by G and, therefore, can be soft-decision Viterbi decoded. For example, the extended Golay (24, 12) code with $d_{\min} = 8$ can be FTB encoded with a nine-stage ($m = 8$) systematic convolutional encoder with $g^{(2)} =$ 1 0 1 1 1 1 0 1 1.

TABLE III
A LIST OF RATE 1/2 SYSTEMATIC CONVOLUTIONAL CODES AND $(2K, K)$ QUASI-CYCLIC BLOCK CODES

		JOHANNESSON		COSTELLO		QUASI-CYCLIC (DOUBLE CIRCULANT)				
K(= 2m)	m	$g^{(2)}$	d_{free}	$g^{(2)}$	d_{free}	K(= L + m)	G	d_{min}	EQUIVALENT m	L
2	1	6*	3	6*	3	3	3	3	1	2
4	2	7*	4	7*	4	4	7	4	2	2
6	3	64	4	7	4	5	7	4	2	3
8	4	72*	5	72*	5	6	7	4	2	4
10	5	73	6	73	6	7	7	4	2	5
12	6	734	6	73	6	8	27	5	4	4
14	7	715	6	732*	7	9	117	6	6	3
16	8	715	7	7324	7	10	57	6	5	5
18	8	7154	8	7324	8	11	257	7	7	4
20	10	7152	8	7324*	8	12	573	8	8	4
22	11	7153	9	7324	8	13	653	7	8	5
24	12	67114	9	73244	9	14	727	8	8	6
26	13	67114	9	73246	10	15	2167	8	10	5
28	14	67115	10	73246	10	16	557	8	8	8
						17	557	8	8	9
						18	573	8	8	10
						19	553	8	8	11
						20	5723	9	11	9
						21	14573	10	12	9
						44	-	16	41	3
						54	-	20	-	-

* Same as quasi-cyclic code of the same m.

Also included in Table III are Johannesson's [11] and Costello's [12] best systematic convolutional codes and their achieved free distance. Notice that the DC generator matrices for K up to 8 are exactly the same as those of Johannesson's and Costello's convolutional codes with the same equivalent constraint lengths.[2] This means that good systematic convolutional codes are also good FTB-generated codes or, equivalently, good DC codes for $K \leq 8$. However, this is not the case for $K > 8$. The difference can be explained by the different performance criteria for designing block and convolutional codes. For block code design, the information block length K (or $(L + m)k$ in convolutional code terminology) is minimized for a given $d_{\min}$. For convolutional code design, the constraint length m is minimized for a given d_{free}. Depending on the specific performance criterion, good codes can be taken from either the quasi-cyclic or the convolutional code family. Interestingly, most FTB-encoded DC codes achieve even better minimum distances than the free distances of Johannesson's or Costello's codes. For example, the DC (22, 11), (24, 12), (28, 14), (32, 16), (34, 17), (36, 18), (38, 19), and (42, 21) codes achieve minimum distances one higher than the free distances of Johannesson's systematic codes of the same equivalent constraint lengths. These codes form a new class of systematic convolutional codes that perform better (in terms of achievable free distance) than the best known systematic convolutional codes. Similar results for rate 1/3 and rate 2/3 FTB codes were reported in [8].

VI. Generalized Transfer Function Technique For Full Tail Biting Codes

Before we discuss the performance evaluation of FTB codes, we define some distance measures useful for studying these codes. For the (n, k, m) FTB codes, there are 2^{km} as many codewords as those for (zero-tail) conventional encoding. From the linearity of FTB codes, we can assume, without loss of generality, that the transmitted symbols are all zeros. For convenience of analysis, we thus separate the additional code paths into those that are merged and those that are unmerged with the all-zero path. By merged paths (with the all-zero path) we mean those code paths that intersect with the all-zero path at a certain node in the trellis. For the merged code paths, we define the column distance function and the free or minimum distance. For the unmerged code paths, we define the truncation distance and the critical and sufficient truncation lengths. For the combined merged and unmerged code paths, we define the minimum distance function.

Definition 1: The column distance function of order i, d_i, for an (n, k, m) FTB code is the minimum Hamming weight of all merged codewords of length i that start and end in S_j ($j = 0, 1, \cdots, 2^{km} - 1$). For $i = \infty$ and noncatastrophic codes, $\lim_{i\to\infty} d_i = d_{\text{free}}$ is called the free distance of the code.

Definition 2: The truncation distance function of order τ, d_τ, for an (n, k, m) FTB code is the minimum Hamming weight of all unmerged codewords of length τ that start and end in S_j ($j = 1, 2, \cdots, 2^{km} - 1$). The sufficient truncation length τ' is the smallest τ for which $d_\tau = d_{\text{free}} + 1$. The critical truncation length τ^* is the smallest τ for which $d_\tau = d_{\text{free}}$.

Definition 3: The minimum distance function of order l, $d_{\min}(l)$, for an (n, k, m) FTB code is the minimum Hamming weight of all codewords (merged and unmerged) of length l that start and end in S_j ($j = 0, 1, \cdots, 2^{km} - 1$); i.e., $d_{\min}(l) = \min\{d_{i=l}, d_{\tau=l}\}$.

Using the linearity of FTB codes as a basis, we now generalize the transfer function approach by Viterbi [13] to evaluate the error performance of FTB codes with maximum likelihood decoding. For conventional (zero-tail) convolutional encoding with Viterbi decoding, we are only interested in the transfer function that includes codewords which begin and end in the all-zero state. For the truncated Viterbi decoding (in which each bit decision made at time t corresponds to the survivor with the best metric value at time $t + T$), we are also interested in codewords that begin in the all-zero state S_0, but are unmerged with that state in the middle of the trellis, and finally end in a nonzero state [14]. For FTB encoding and Viterbi FTB decoding, we generalize the transfer function to include, in addition to the zero-tail codewords, the merged and unmerged (with the all-zero state) codewords that start and end in the same nonzero state. Those codewords that start and end in different states should not be considered as codewords for the FTB codes, due to the imposed FTB encoding constraint. Thus, for an (n, k, m) convolutional code with a zero-tail transfer function $T(X, Y, Z)$, the bit error probability on a binary symmetric channel (with transition probability p) of a Viterbi decoder at the $(L + m)$th branch is upper bounded by

$$P_b(E) < \frac{1}{k}\left[\frac{\partial T(X, Y, Z)}{\partial Y} + \sum_{i=1}^{2^{mk}-1}\frac{\partial T_i(X, Y, Z)}{\partial Y} + \sum_{i=1}^{2^{mk}-1}\frac{\partial T_{i,i}^{L+m}(X, Y, Z)}{\partial Y}\right], \quad (1)$$

$$(X = \sqrt{4p(1-p)}\,,\ Y = Z = 1)$$

where T_i is the composite transfer function of $T_{0,i}$ and $T_{i,0}$ (i.e., $T_i = T_{0,i} \times T_{i,0}$), $T_{i,j}$ is the transfer function that starts in S_i and ends in S_j, and $T_{i,i}^{\tau}$ is the transfer function that includes unmerged codewords of length τ that start and end in the same state S_i. The power of X corresponds to the Hamming weight of the encoder output, the power of Y corresponds to the Hamming weight of the encoder input, and the power of Z corresponds to the number of branches. The first term in (1) represents the decoding errors made by a standard zero-tail decoder, whereas the second term represents the decoding errors due to the merged codewords of the FTB encoding, and the third term represents the decoding errors due to the unmerged codewords of the FTB encoding.

Example 1: Consider the rate 1/2, $m = 2$ code with $g^{(1)}(D) = 1 + D^2$ and $g^{(2)}(D) = 1 + D + D^2$. The modified transfer function for the zero-tail trellis is

$$T(X, Y, Z) = \frac{X^5YZ^3}{1 - XYZ(1+Z)}. \quad (2)$$

First we consider the additional code paths that are merged with the all-zero state for the FTB codes. The composite transfer functions can be written as

$$T_1 = T_{0,1} \times T_{1,0} = \frac{X^2YZ(1-XYZ)}{1-XYZ(1+Z)} \times \frac{X^3Y^2}{1-XYZ(1+Z)} = \frac{X^5Y^3Z(1-XYZ)}{[1-XYZ(1+Z)]^2} \to d_{\min} = 5$$

$$T_2 = T_{0,2} \times T_{2,0} = \frac{X^3YZ^2}{1-XYZ(1+Z)} \times \frac{X^2Z(1-XYZ)}{1-XYZ(1+Z)} = \frac{X^5YZ^3(1-XYZ)}{[1-XYZ(1+Z)]^2} \to d_{\min} = 5$$

$$T_3 = T_{0,3} \times T_{3,0} = \frac{X^3Y^2Z^2}{1-XYZ(1+Z)} \times \frac{X^3Z^2(1-XYZ^2)}{1-XYZ(1+Z)} = \frac{X^6Y^2Z^4(1-XYZ^2)}{[1-XYZ(1+Z)]^2} \to d_{\min} = 6.$$

[2] We define the equivalent constraint length of a quasi-cyclic code as the number of digits represented by G, but excluding the starting and ending zeros.

TABLE IV
BINARY FTB CONVOLUTIONAL CODE PERFORMANCE SUMMARY (ODENWALDER CODES)

m	d_{free}	$C'_{d_{free}}$	$A'_{d_{free}}$	$A'_{d_{free}} \leq 2^{m-1}+1$ (WORST-CASE BOUND)	$\tau'^{\dagger}$	$\tau^{*\dagger}$	EQUIVALENT BLOCK CODE AT τ'	BEST BLOCK CODE
2	5	1	3	3	$11^{(6)}$	$9^{(5)}$	(22,11)5	(22,11)7
3	6	1	5	5	$14^{(7)}$	$11^{(6)}$	(28,14)6	(28,14)8
4	7	2	9	9	$22^{(8)}$	$15^{(7)}$	(44,22)7	(44,22)10
5	8	1	8	17	$24^{(10)}$	$18^{(8)}$	(48,24)8	(48,24)12
6	10	11	33	33	$28^{(12)}$	$24^{(10)}$	(56,28)10	(56,28)12
7	10	1	9	65	$34^{(12)}$	$27^{(10)}$	(68,34)10	(68,34)13
8	12	11	129	129	$41^{(15)}$	$34^{(12)}$	(82,41)12	(82,41)16

$^{\dagger}d_{\tau'}$ AND d_{τ^*} ARE DENOTED IN THE PARENTHESES ABOVE τ' AND τ^* RESPECTIVELY.

For the additional unmerged code path, the transfer functions are

$$T_{1,1}^{\tau}(X, Y, Z) = \frac{1-XYZ}{1-XYZ(1+Z)}$$

$$T_{2,2}^{\tau}(X, Y, Z) = T_{1,1}^{\tau}(X, Y, Z)$$

$$T_{3,3}^{\tau}(X, Y, Z) = \frac{1-XYZ^2}{1-XYZ(1+Z)}.$$

The original zero-tail code has $d_{\min} = 5$ from (2). For the FTB merged portion, T_1 contains a term X^5Y^3Z and T_2 contains a term X^5YZ^3, while T_3 contains no terms of weight 5. Thus, there are two additional codewords with $d_{\text{free}} = 5$, one with one nonzero information bit and the other with three nonzero information bits. From the transfer functions of the unmerged code paths, it is easily verified that $\tau^* = 9$ and $\tau' = 11$. Hence, a path memory of 11 should be sufficient to ensure a negligible increase in bit error probability due to FTB encoding over a standard Viterbi decoder for this code. A (22, 11) $d_{\min} = 5$ block code results if we truncate this convolutional code at $\tau' = 11$ by FTB (since $d_{\min}(11) = \min\{d_{\text{free}}, d_{\tau'}\} = \min\{5, 6\} = 5$).

In general, the transfer function of short constraint length ($m \leq 5$) FTB convolutional codes can be obtained using Mason's gain formula. This function provides a complete spectrum of the weight distribution of the code. For longer constraint-length FTB codes, FTB Viterbi decoding assuming all-zero received symbols can be executed to obtain the important performance parameters such as the free distances and critical truncation lengths of the codes. Table IV lists some of these results for rate 1/2 optimum free-distance codes, where $A'_{d_{\text{free}}}$ and $C'_{d_{\text{free}}}$ are the total numbers of merged codewords with weight equal to the free distance for the FTB and zero-tail encoding, respectively. Also listed are the equivalent block codes obtained by truncating convolutional codes at τ' using FTB and the best known block codes of the same block length.

VII. SUMMARY AND CONCLUSIONS

Here we have treated a new method called generalized tail biting for converting convolutional codes into block codes. We explored both optimum and suboptimum decoding algorithms for these codes. Then we considered how good block codes can be obtained from good convolutional codes and vice versa. Finally, we described an analytical technique for upper bounding the performance of these codes. Although many of our decoding and performance analysis techniques were described for full tail biting, most of these generalize to the case of partial tail biting.

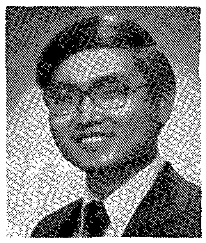
ACKNOWLEDGMENT

The first author thanks the general support of the MITRE Corporation. In particular, he wishes to thank R. D. Haggarty and E. A. Palo for their encouragement and stimulation during the course of this research.

REFERENCES

[1] E. R. Berlekamp, "The technology of error-correcting codes," *Proc. IEEE,* vol. 68, pp. 564-593, May 1980.
[2] Special Issue on Error Correcting Codes, *IEEE Trans. Commun. Technol.,* vol. COM-19, pp. 741-868, Oct. 1971.
[3] J. P. Odenwalder, private communications, Sept. 1981.
[4] G. Solomon and H. C. A. van Tilborg, "A connection between block and convolutional codes," *SIAM J. Appl. Math.,* vol. 37, pp. 358-369, Oct. 1979.
[5] H. H. Ma and J. K. Wolf, "Binary unequal error-protection block codes formed from convolutional codes by generalized tail-biting," *IEEE Trans. Inform. Theory,* to be published.
[6] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory,* vol. IT-13, pp. 260-269, Apr. 1967.
[7] I. Bar-David, private communications, Nov. 1982.
[8] H. H. Ma, "Generalized tail biting convolutional codes," Ph.D. dissertation, Univ. Massachusetts, Amherst, Feb. 1985.
[9] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1983, ch. 10.
[10] J. P. Odenwalder, "Optimal decoding of convolutional codes," Ph.D. dissertation, Dep. Syst. Sci., Univ. California, Los Angeles, 1970, pp. 62-68.
[11] R. Johannesson, "Robustly-optimum rate one-half binary convolutional codes," *IEEE Trans. Inform. Theory,* vol. IT-21, pp. 464-468, July 1975.
[12] D. J. Costello, Jr., "A construction technique for random-error-correcting convolutional codes," *IEEE Trans. Inform. Theory,* vol. IT-15, pp. 631-636, Sept. 1969.
[13] A. J. Viterbi, "Convolutional codes and their performance in communication system," *IEEE Trans. Commun. Technol.,* vol. COM-19, pp. 751-772, Oct. 1971.
[14] F. Hemmati and D. J. Costello, Jr., "Truncation error probability in Viterbi decoding," *IEEE Trans. Commun.,* vol. COM-25, pp. 530-532, May 1977.

★



Howard H. Ma (S'78-M'78-S'81) was born in Taiwan on December 28, 1951. He received the B.S. degree in communication engineering from the National Chiao-Tung University, Taiwan, in 1974, the M.S. degree in electrical engineering from the University of Houston, Houston, TX, in 1978, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst, in 1985.

From 1974 to 1976 he served as an Administrative Officer in the Army of the Republic of China. From 1976 to 1978 he worked as a Teaching Assistant at the University of Houston. Since August 1978, he has been a member of the Technical Staff at the MITRE Corporation, Bedford, MA, where he has been engaged in research on various aspects of signal processing and digital communications. His research interests include coding theory, detection and estimation theory, and spread-spectrum communications. He has published papers on the performance of the Rayleigh fast-fading channel and on sequential decoding algorithms.

Dr. Ma is a member of the Communications Software Committee of the IEEE Communications Society. He was corecipient of the 1984 MITRE's Best Papers Award for the paper "Error-correcting codes against the worst-case partial-band jammer," published in the IEEE TRANSACTIONS ON COMMUNICATIONS.

★

Jack K. Wolf (S'54-M'60-F'73) received the B.S.E.E. degree from the University of Pennsylvania, Philadelphia, in 1956, and the M.S.E., M.A., and Ph.D. degrees from Princeton University, Princeton, NJ, in 1957, 1958, and 1960, respectively.

He was a member of the Department of Electrical Engineering at New York University, New York, NY, from 1963 to 1965, and the Polytechnic Institute of Brooklyn, Brooklyn, NY, from 1965 to 1973. He was Chairman of the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst, from 1973 to 1975. During 1968-1969, he was a member of the Mathematics Research Center at Bell Laboratories. From 1971 to 1972 he was an NSF Senior Postdoctoral Fellow at the University of Hawaii, Honolulu. From 1979 to 1980 he was a Guggenheim Fellow at the University of California at San Diego and the Linkabit Corporation. From 1973 to 1984 he was a Professor of Electrical Engineering at the University of Massachusetts, Amherst. He is currently a Professor of Electrical Engineering and Computer Science at the University of California, San Diego, where he is also a member of the Center for Magnetic Recording Research. His research interests are in information theory, algebraic coding theory, communications systems, computer networks, and magnetic recording.

Dr. Wolf was corecipient of the 1975 Information Theory Group Paper Award for the paper "Noiseless coding of correlated information sources" (coauthored with D. Slepian). He was Cochairman of the 1969 International Symposium on Information Theory. He served on the Board of Governors of the Information Theory Group from 1970 to 1976, and since 1980. He was President of the Information Theory Group in 1974. From 1981 to 1984 he was International Chairman of Commission C of URSI. His IEEE Fellow Award reads, "For contributions to coding theory."