TDA Progress Report 42-123 November 15, 1995

On the Design of Turbo Codes

D. Divsalar and F. Pollara
Communications Systems and Research Section

In this article, we design new turbo codes that can achieve near-Shannon-limit performance. The design criterion for random interleavers is based on maximizing the effective free distance of the turbo code, i.e., the minimum output weight of codewords due to weight-2 input sequences. An upper bound on the effective free distance of a turbo code is derived. This upper bound can be achieved if the feedback connection of convolutional codes uses primitive polynomials. We review multiple turbo codes (parallel concatenation of q convolutional codes), which increase the so-called "interleaving gain" as q and the interleaver size increase, and a suitable decoder structure derived from an approximation to the maximum a posteriori probability decision rule. We develop new rate 1/3, 2/3, 3/4, and 4/5 constituent codes to be used in the turbo encoder structure. These codes, for from 2 to 32 states, are designed by using primitive polynomials. The resulting turbo codes have rates b/n, b=*1, 2, 3, 4, and* n=*2, 3, 4, 5, 6 and include random interleavers for better asymptotic performance. These codes are suitable for deep-space communications with low throughput and for near-Earth communications where high throughput is desirable. The performance of these codes is within 1 dB of the Shannon limit at a bit-error rate of 10^{-6} for throughputs from 1/15 up to 4 bits/s/Hz.*

I. Introduction

Coding theorists have traditionally attacked the problem of designing good codes by developing codes with a lot of structure, which lends itself to feasible decoders, although coding theory suggests that codes chosen "at random" should perform well if their block sizes are large enough. The challenge to find practical decoders for "almost" random, large codes has not been seriously considered until recently. Perhaps the most exciting and potentially important development in coding theory in recent years has been the dramatic announcement of "turbo codes" by Berrou et al. in 1993 [7]. The announced performance of these codes was so good that the initial reaction of the coding establishment was deep skepticism, but recently researchers around the world have been able to reproduce those results [15,19,8]. The introduction of turbo codes has opened a whole new way of looking at the problem of constructing good codes [5] and decoding them with low complexity [7,2].

Turbo codes achieve near-Shannon-limit error correction performance with relatively simple component codes and large interleavers. A required E_b/N_0 of 0.7 dB was reported for a bit-error rate (BER) of 10^{-5} for a rate 1/2 turbo code [7]. Multiple turbo codes (parallel concatenation of $q > 2$ convolutional codes) and a suitable decoder structure derived from an approximation to the maximum a posteriori (MAP) probability decision rule were reported in [9]. In [9], we explained for the first time the turbo decoding

scheme for multiple codes and its relation to the optimum bit decision rule, and we found rate 1/4 turbo codes whose performance is within 0.8 dB of Shannon's limit at BER=10^{-5}.

In this article, we (1) design the best component codes for turbo codes of various rates by maximizing the "effective free distance of the turbo code," i.e., the minimum output weight of codewords due to weight-2 input sequences; (2) describe a suitable trellis termination rule for b/n codes; (3) design low throughput turbo codes for power-limited channels (deep-space communications); and (4) design high-throughput turbo trellis-coded modulation for bandwidth-limited channels (near-Earth communications).

II. Parallel Concatenation of Convolutional Codes

The codes considered in this article consist of the parallel concatenation of multiple ($q \geq 2$) convolutional codes with random interleavers (permutations) at the input of each encoder. This extends the original results on turbo codes reported in [7], which considered turbo codes formed from just two constituent codes and an overall rate of 1/2.

Figure 1 provides an example of parallel concatenation of three convolutional codes. The encoder contains three recursive binary convolutional encoders with m_1, m_2, and m_3 memory cells, respectively. In general, the three component encoders may be different and may even have different rates. The first component encoder operates directly (or through π_1) on the information bit sequence $\mathbf{u} = (u_1, \cdots, u_N)$ of length N, producing the two output sequences $\mathbf{x}_0$ and $\mathbf{x}_1$. The second component encoder operates on a reordered sequence of information bits, $\mathbf{u}_2$, produced by a permuter (interleaver), π_2, of length N, and outputs the sequence $\mathbf{x}_2$. Similarly, subsequent component encoders operate on a reordered sequence of information bits. The interleaver is a pseudorandom block scrambler defined by a permutation of N elements without repetitions: A complete block is read into the the interleaver and read out in a specified (fixed) random order. The same interleaver is used repeatedly for all subsequent blocks.

Figure 1 shows an example where a rate $r = 1/n = 1/4$ code is generated by three component codes with memory $m_1 = m_2 = m_3 = m = 2$, producing the outputs $\mathbf{x}_0 = \mathbf{u}$, $\mathbf{x}_1 = \mathbf{u} \cdot g_1/g_0$, $\mathbf{x}_2 = \mathbf{u}_2 \cdot g_1/g_0$, and $\mathbf{x}_3 = \mathbf{u}_3 \cdot g_1/g_0$ (here π_1 is assumed to be an identity, i.e., no permutation), where the generator polynomials g_0 and g_1 have octal representation $(7)_{octal}$ and $(5)_{octal}$, respectively. Note that various code rates can be obtained by proper puncturing of $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and even $\mathbf{x}_0$ (for an example, see Section V).

We use the encoder in Fig. 1 to generate an $(n(N+m), N)$ block code, where the m tail bits of code 2 and code 3 are not transmitted. Since the component encoders are recursive, it is not sufficient to set the last m information bits to zero in order to drive the encoder to the all-zero state, i.e., to *terminate* the trellis. The termination (tail) sequence depends on the state of each component encoder after N bits, which makes it impossible to terminate all component encoders with m predetermined tail bits. This issue, which had not been resolved in the original turbo code implementation, can be dealt with by applying a simple method described in [8] that is valid for any number of component codes. A more complicated method is described in [18].

A design for constituent convolutional codes, which are not necessarily optimum convolutional codes, was originally reported in [5] for rate $1/n$ codes. In this article, we extend those results to rate b/n codes. It was suggested (without proof) in [2] that good random codes are obtained if g_a is a primitive polynomial. This suggestion, used in [5] to obtain "good" rate 1/2 constituent codes, will be used in this article to obtain "good" rate 1/3, 2/3, 3/4, and 4/5 constituent codes. By "good" codes we mean codes with a maximum effective free distance d_{ef}, those codes that maximize the minimum output weight for weight-2 input sequences, as discussed in [9], [13], and [5] (because this weight tends to dominate the performance characteristics over the region of interest).

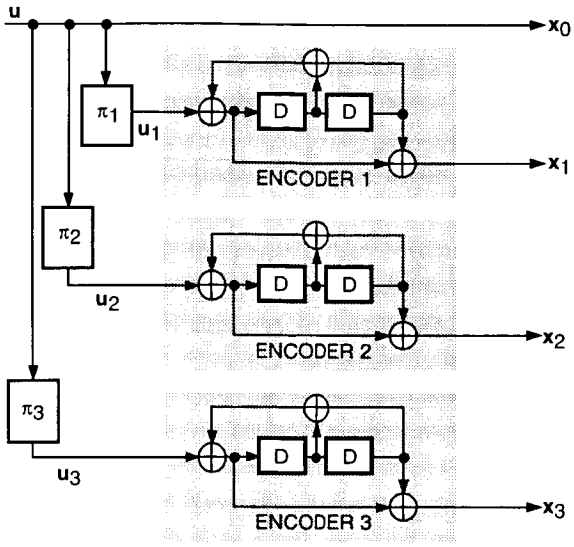

Fig. 1. Example of encoder with three codes.

III. Design of Constituent Encoders

As discussed in the previous section, maximizing the weight of output codewords corresponding to weight-2 data sequences gives the best BER performance for a moderate bit signal-to-noise ratio (SNR) as the random interleaver size N gets large. In this region, the dominant term in the expression for bit error probability of a turbo code with q constituent encoders is

$$P_b \approx \frac{\beta}{N^{q-1}} Q\left(\sqrt{2r\frac{E_b}{N_0}\left(\sum_{j=1}^{q} d_{j,2}^p + 2\right)}\right)$$

where $d_{j,2}^p$ is the minimum parity-weight (weight due to parity checks only) of the codewords at the output of the jth constituent code due to weight-2 data sequences, and β is a constant independent of N. Define $d_{j,2} = d_{j,2}^p + 2$ as the minimum output weight including parity and information bits, if the jth constituent code transmits the information (systematic) bits. Usually one constituent code transmits the information bits ($j = 1$), and the information bits of others are punctured. Define $d_{ef} = \sum_{j=1}^{q} d_{j,2}^p + 2$ as the effective free distance of the turbo code and $1/N^{q-1}$ as the "interleaver's gain." We have the following bound on d_2^p for any constituent code.

Theorem 1. For any $r = b/(b+1)$ recursive systematic convolutional encoder with generator matrix

$$G = \begin{bmatrix} & \frac{h_1(D)}{h_0(D)} \\ & \frac{h_2(D)}{h_0(D)} \\ \mathbf{I}_{b\times b} & \cdot \\ & \cdot \\ & \frac{h_b(D)}{h_0(D)} \end{bmatrix}$$

where $I_{b\times b}$ is a $b \times b$ identity matrix, $\deg[h_i(D)] \leq m$, $h_i(D) \neq h_0(D)$, $i = 1, 2, \cdots, b$, and $h_0(D)$ is a primitive polynomial of degree m, the following upper bound holds:

$$d_2^p \leq \lfloor \frac{2^{m-1}}{b} \rfloor + 2$$

Proof. In the state diagram of any recursive systematic convolutional encoder with generator matrix G, there exist at least two nonoverlapping loops corresponding to all-zero input sequences. If $h_0(D)$ is a primitive polynomial, there are two loops: one corresponding to zero-input, zero-output sequences with branch length one, and the other corresponding to zero-input but nonzero-output sequences with branch length $2^m - 1$, which is the period of maximal length (ML) linear feedback shift registers (LFSRs) [14] with degree m. The parity codeword weight of this loop is 2^{m-1}, due to the balance property [14] of ML sequences. This weight depends only on the degree of the primitive polynomial and is independent of $h_i(D)$, due to the invariance to initial conditions of ML LFSR sequences. In general, the output of the encoder is a linear function of its input and current state. So, for any output we may consider, provided it depends on at least one component of the state and it is not $h_0(D)$, the weight of a zero-input loop is 2^{m-1}, by the shift-and-add property of ML LFSRs.

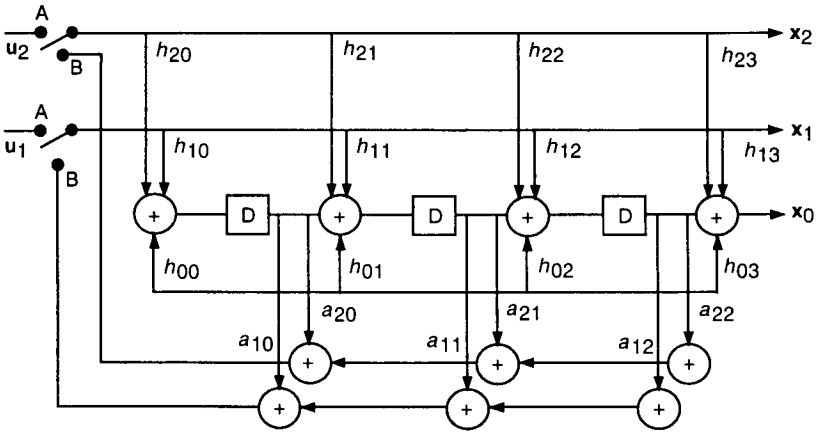

Fig. 2. Canonical representation of a rate $(b + 1)/b$ encoder ($b = 2$, $m = 3$).

Consider the canonical representation of a rate $(b + 1)/b$ encoder [20] as shown in Fig. 2 when the switch is in position A. Let $S^k(D)$ be the state of the encoder at time k with coefficients $S_0^k, S_1^k, \cdots, S_{m-1}^k$, where the output of the encoder at time k is

$$X = S_{m-1}^{k-1} + \sum_{i=1}^{b} u_i^k h_{i,m} \tag{1}$$

The state transition for input $u_1^k, \cdots, u_b^k$ at time k is given by

$$S^k(D) = \left[\sum_{i=1}^{b} u_i^k h_i(D) + DS^{k-1}(D) \right] \mod h_0(D) \tag{2}$$

From the all-zero state, we can enter the zero-input loop with nonzero input symbols $u_1, \cdots, u_b$ at state

$$S^1(D) = \sum_{i=1}^{b} u_i h_i(D) \bmod h_0(D) \tag{3}$$

From the same nonzero input symbol, we leave exactly at state $S^{2^m-1}(D)$ back to the all-zero state, where $S^{2^m-1}(D)$ satisfies

$$S^1(D) = DS^{2^m-1}(D) \bmod h_0(D) \tag{4}$$

i.e., $S^{2^m-1}(D)$ is the "predecessor" to state $S^1(D)$ in the zero-input loop. If the most significant bit of the predecessor state is zero, i.e., $S_{m-1}^{2^m-1} = 0$, then the branch output for the transition from $S^{2^m-1}(D)$ to $S^1(D)$ is zero for a zero-input symbol. Now consider any weight-1 input symbol, i.e., $u_j = 1$ for $j = i$ and $u_j = 0$ for $j \neq i$, $j = 1, 2, \cdots, b$. The question is: What are the conditions on the coefficients $h_i(D)$ such that, if we enter with a weight-1 input symbol into the zero-input loop at state $S^1(D)$, the most significant bit of the "predecessor" state $S^{2^m-1}(D)$ is zero. Using Eqs. (3) and (4), we can establish that

$$h_{i0} + h_{i,m} = 0 \tag{5}$$

Obviously, when we enter the zero-input loop from the all-zero state and when we leave this loop to go back to the all-zero state, we would like the parity output to be equal to 1. From Eqs. (1) and (5), we require

$$\left.\begin{aligned} h_{i0} &= 1 \\ h_{i,m} &= 1 \end{aligned}\right\} \tag{6}$$

With this condition, we can enter the zero-input loop with a weight-1 symbol at state $S^1(D)$ and then leave this loop from state $S^{2^m-1}(D)$ back to the all-zero state, for the same weight-1 input. The parity weight of the codeword corresponding to weight-2 data sequences is then $2^{m-1} + 2$, where the first term is the weight of the zero-input loop and the second term is due to the parity bit appearing when entering and leaving the loop. If $b = 1$, the proof is complete, and the condition to achieve the upper bound is given by Eq. (6). For $b = 2$, we may enter the zero-input loop with $\mathbf{u} = 10$ at state $S^1(D)$ and leave the loop to the zero state with $\mathbf{u} = 01$ at some state $S^j(D)$. If we can choose $S^j(D)$ such that the output weight of the zero-input loop from $S^1(D)$ to $S^j(D)$ is exactly $2^{m-1}/2$, then the output weight of the zero-input loop from $S^{j+1}(D)$ to $S^{2^m-1}(D)$ is exactly $2^{m-1}/2$, and the minimum weight of codewords corresponding to some weight-2 data sequences is

$$\frac{2^{m-1}}{2} + 2$$

In general, for any b, if we extend the procedure for $b = 2$, the minimum weight of the codewords corresponding to weight-2 data sequences is

$$\lfloor \frac{2^{m-1}}{b} \rfloor + 2 \tag{7}$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x. Clearly, this is the best achievable weight for the minimum-weight codeword corresponding to weight-2 data sequences. This upper bound can be achieved

if the maximum run length of 1's (m) in the zero-input loop does not exceed $\lfloor 2^{m-1}/b \rfloor$. If $m > \lfloor 2^{m-1}/b \rfloor$, then the minimum weight of the codewords corresponding to weight-2 data sequences will be strictly less than $\lfloor 2^{m-1}/b \rfloor + 2$.

The run property of ML LFSRs [14] can help us in designing codes achieving this upper bound. Consider only runs of 1's with length l for $0 < l < m-1$; then there are 2^{m-2-l} runs of length l, no runs of length $m-1$, and only one run of length m. ❐

Corollary 1. For any $r = b/n$ recursive systematic convolutional code with b inputs, b systematic outputs, and $n-b$ parity output bits using a primitive feedback generator, we have

$$d_2^p \leq \lfloor \frac{(n-b)2^{m-1}}{b} \rfloor + 2(n-b) \tag{8}$$

Proof. The total output weight of a zero-input loop due to parity bits is $(n-b)2^{M-1}$. In this zero-input loop, the largest minimum weight (due to parity bits) for entering and leaving the loop with any weight-1 input symbol is $[(n-b)2^{M-1}]/b$. The output weight due to parity bits for entering and leaving the zero-input loop (both into and from the all-zero state) is $2(n-b)$. ❐

There is an advantage to using $b > 1$, since the bound in Eq. (8) for rate b/bn codes is larger than the bound for rate $1/n$ codes. Examples of codes are found that meet the upper bound for b/bn codes.

A. Best Rate *b/b* + 1 Constituent Codes

We obtained the best rate 2/3 codes as shown in Table 1, where $d_2 = d_2^p + 2$. The minimum-weight codewords corresponding to weight-3 data sequences are denoted by d_3, d_{min} is the minimum distance of the code, and $k = m+1$ in all the tables. By "best" we mean only codes with a large d_2 for a given m that result in a maximum effective free distance. We obtained the best rate 3/4 codes as shown in Table 2 and the best rate 4/5 codes as shown in Table 3.

Table 1. Best rate 2/3 constituent codes.

k	Code generator			d_2	d_3	d_{min}
3	$h_0 = 7$	$h_1 = 3$	$h_2 = 5$	4	3	3
4	$h_0 = 13$	$h_1 = 15$	$h_2 = 17$	5	4	4
5	$h_0 = 23$	$h_1 = 35$	$h_2 = 27$	8	5	5
	$h_0 = 23$	$h_1 = 35$	$h_2 = 33$	8	5	5
6	$h_0 = 45$	$h_1 = 43$	$h_2 = 61$	12	6	6

Table 2. Best rate 3/4 constituent codes.

k	Code generator				d_2	d_3	d_{min}
3	$h_0 = 7$	$h_1 = 5$	$h_2 = 3$	$h_3 = 1$	3	3	3
	$h_0 = 7$	$h_1 = 5$	$h_2 = 3$	$h_3 = 4$	3	3	3
	$h_0 = 7$	$h_1 = 5$	$h_2 = 3$	$h_3 = 2$	3	3	3
4	$h_0 = 13$	$h_1 = 15$	$h_2 = 17$	$h_3 = 11$	4	4	4
5	$h_0 = 23$	$h_1 = 35$	$h_2 = 33$	$h_3 = 25$	5	4	4
	$h_0 = 23$	$h_1 = 35$	$h_2 = 27$	$h_3 = 31$	5	4	4
	$h_0 = 23$	$h_1 = 35$	$h_2 = 37$	$h_3 = 21$	5	4	4
	$h_0 = 23$	$h_1 = 27$	$h_2 = 37$	$h_3 = 21$	5	4	4

Table 3. Best rate 4/5 constituent codes.

k	Code generator					d_2	d_3	d_{min}
4	$h_0 = 13$	$h_1 = 15$	$h_2 = 17$	$h_3 = 11$	$h_4 = 7$	4	3	3
	$h_0 = 13$	$h_1 = 15$	$h_2 = 17$	$h_3 = 11$	$h_4 = 5$	4	3	3
5	$h_0 = 23$	$h_1 = 35$	$h_2 = 33$	$h_3 = 37$	$h_4 = 31$	5	4	4
	$h_0 = 23$	$h_1 = 35$	$h_2 = 27$	$h_3 = 37$	$h_4 = 31$	5	4	4
	$h_0 = 23$	$h_1 = 35$	$h_2 = 21$	$h_3 = 37$	$h_4 = 31$	5	4	4

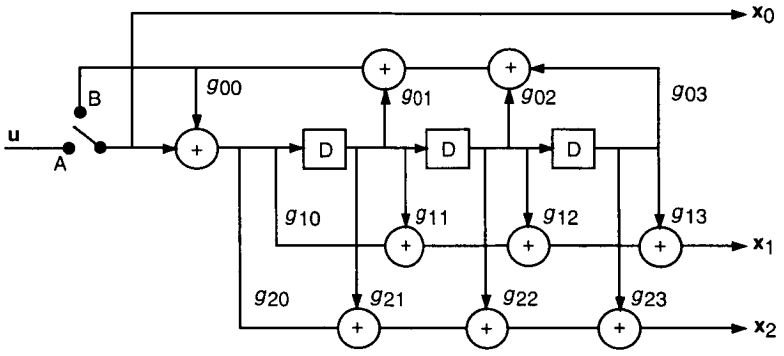

Fig. 3. Rate 1/*n* code.

B. Trellis Termination for *b/n* Codes

Trellis termination is performed (for $b = 2$, as an example) by setting the switch shown in Fig. 2 in position B. The tap coefficients $a_{i0}, \cdots, a_{i,m-1}$ for $i = 1, 2, \cdots, b$ can be obtained by repeated use of Eq. (2) and by solving the resulting equations. The trellis can be terminated in state zero with at least m/b and at most m clock cycles. When Fig. 3 is extended to multiple input bits (b parallel feedback shift registers), a switch should be used for each input bit.

C. Best Punctured Rate 1/2 Constituent Codes

A rate 2/3 constituent code can be derived by puncturing the parity bit of a rate 1/2 recursive systematic convolutional code using, for example, a pattern $P = [10]$. A puncturing pattern P has zeros where parity bits are removed.

Consider a rate 1/2 recursive systematic convolutional code $(1, g_1(D)/(g_0(D))$. For an input $u(D)$, the parity output can be obtained as

$$x(D) = \frac{u(D)g_1(D)}{g_0(D)} \tag{9}$$

We would like to puncture the output $x(D)$ using, for example, the puncturing pattern $P[10]$ (decimation by 2) and obtain the generator polynomials $h_0(D)$, $h_1(D)$, and $h_2(D)$ for the equivalent rate 2/3 code:

$$G = \begin{bmatrix} 1 & 0 & \dfrac{h_1(D)}{h_0(D)} \\ 0 & 1 & \dfrac{h_2(D)}{h_0(D)} \end{bmatrix}$$

We note that any polynomial $f(D) = \sum a_i D^i$, $a_i \in GF(2)$, can be written as

$$f(D) = f_1(D^2) + Df_2(D^2) \tag{10}$$

where $f_1(D^2)$ corresponds to the even power terms of $f(D)$, and $Df_2(D^2)$ corresponds to the odd power terms of $f(D)$. Now, if we use this approach and apply it to the $u(D)$, $g_1(D)$, and $g_0(D)$, then we can rewrite Eq. (9) as

$$x_1\left(D^2\right) + Dx_2\left(D^2\right) = \frac{\left(u_1\left(D^2\right) + Du_2\left(D^2\right)\right)\left(g_{11}\left(D^2\right) + Dg_{12}\left(D^2\right)\right)}{g_{01}\left(D^2\right) + Dg_{02}\left(D^2\right)} \tag{11}$$

where $x_1(D)$ and $x_2(D)$ correspond to the punctured output $x(D)$ using puncturing patterns $P[10]$ and $P[01]$, respectively. If we multiply both sides of Eq. (11) by $(g_{01}(D^2) + Dg_{02}(D^2))$ and equate the even and the odd power terms, we obtain two equations in two unknowns, namely $x_1(D)$ and $x_2(D)$. For example, solving for $x_1(D)$, we obtain

$$x_1(D) = u_1(D)\frac{h_1(D)}{h_0(D)} + u_2(D)\frac{h_2(D)}{h_0(D)} \tag{12}$$

where $h_0(D) = g_0(D)$ and

$$\left.\begin{aligned} h_1(D) &= g_{11}(D)g_{01}(D) + Dg_{12}(D)g_{02}(D) \\ h_2(D) &= Dg_{12}(D)g_{01}(D) + Dg_{11}(D)g_{02}(D) \end{aligned}\right\} \tag{13}$$

From the second equation in Eq. (13), it is clear that $h_{2,0} = 0$. A similar method can be used to show that for P[01] we get $h_{1,m} = 0$. These imply that the condition of Eq. (6) will be violated. Thus, we have the following theorem.

Theorem 2. If the parity puncturing pattern is $P = [10]$ or $P = [01]$, then it is impossible to achieve the upper bound on $d_2 = d_2^p + 2$ for rate 2/3 codes derived by puncturing rate 1/2 codes.

The best rate 1/2 constituent codes with puncturing pattern $P = [10]$ that achieve the largest d_2 are given in Table 4.

Table 4. Best rate 1/2 punctured constituent codes.

k	Code generator		d_2	d_3	d_{min}
3	$g_0 = 7$	$g_1 = 5$	4	3	3
4	$g_0 = 13$	$g_1 = 15$	5	4	4
5	$g_0 = 23$	$g_1 = 37$	7	4	4
	$g_0 = 23$	$g_1 = 31$	7	4	4
	$g_0 = 23$	$g_1 = 33$	6	5	5
	$g_0 = 23$	$g_1 = 35$	6	4	4
	$g_0 = 23$	$g_1 = 27$	6	4	4

D. Best Rate 1/*n* Constituent Codes

For rate $1/n$ codes, the upper bound in Eq. (7) for $b = 1$ reduces to

$$d_2^p \leq (n-1)(2^{m-1}+2)$$

This upper bound was originally derived in [5], where the best rate 1/2 constituent codes meeting the bound were obtained. Here we present a simple proof based on our previous general result on rate b/n codes. Then we obtain the best rate 1/3 and 1/4 codes.

Theorem 3. For rate $1/n$ recursive systematic convolutional codes with primitive feedback, we have

$$d_2^p \leq (n-1)(2^{m-1}+2)$$

Proof. Consider a rate $1/n$ code, shown in Fig. 3. In this figure, $g_0(D)$ is assumed to be a primitive polynomial. As discussed above, the output weight of the zero-input loop for parity bits is 2^{m-1} independent of the choice of $g_i(D)$, $i = 1, 2, \cdots, n-1$, provided that $g_i(D) \neq 0$ and that $g_i(D) \neq g_0(D)$, by the shift-and-add and balance properties of ML LFSRs. If $S(D)$ represents the state polynomial, then we can enter the zero-input loop only at state $S^1(D) = 1$ and leave the loop to the all-zero state at state $S^{2^m-1}(D) = D^{m-1}$. The ith parity output on the transition $S^{2^m-1}(D) \rightarrow S^1(D)$ with a zero input bit is

$$x_i = g_{i0} + g_{i,m}$$

If $g_{i0} = 1$ and $g_{i,m} = 1$ for $i = 1, \cdots, n-1$, the output weight of the encoder for that transition is zero. The output weight due to the parity bits when entering and leaving the zero-input loop is $(n-1)$ for each case. In addition, the output weight of the zero-input loop will be $(n-1)2^{m-1}$ for $(n-1)$ parity bits. Thus, we established the upper bound on d_2^p for rate 1/n codes. ❐

We obtained the best rate 1/3 and 1/4 codes without parity repetition, as shown in Tables 5 and 6, where $d_2 = d_2^p + 2$ represents the minimum output weight given by weight-2 data sequences. The best rate 1/2 constituent codes are given by g_0 and g_1 in Table 5, as was also reported in [5].

Table 5. Best rate 1/3 constituent codes.

k	Code generator			d_2	d_3	d_{min}
2	$g_0 = 3$	$g_1 = 2$	$g_2 = 1$	4	∞	4
3	$g_0 = 7$	$g_1 = 5$	$g_2 = 3$	8	7	7
4	$g_0 = 13$	$g_1 = 17$	$g_2 = 15$	14	10	10
5	$g_0 = 23$	$g_1 = 33$	$g_2 = 37$	22	12	10
	$g_0 = 23$	$g_1 = 25$	$g_2 = 37$	22	11	11

Table 6. Best rate 1/4 constituent codes.

k	Code generator				d_2	d_3	d_{min}
4	$g_0 = 13$	$g_1 = 17$	$g_2 = 15$	$g_3 = 11$	20	12	12
5	$g_0 = 23$	$g_1 = 35$	$g_2 = 27$	$g_3 = 37$	32	16	14
	$g_0 = 23$	$g_1 = 33$	$g_2 = 27$	$g_3 = 37$	32	16	14
	$g_0 = 23$	$g_1 = 35$	$g_2 = 33$	$g_3 = 37$	32	16	14
	$g_0 = 23$	$g_1 = 33$	$g_2 = 37$	$g_3 = 25$	32	15	15

E. Recursive Systematic Convolutional Codes With a Nonprimitive Feedback Polynomial

So far, we assumed that the feedback polynomial for recursive systematic convolutional code is a primitive polynomial. We could ask whether it is possible to exceed the upper bound given in Theorem 1 and Corollary 1 by using a nonprimitive polynomial. The answer is negative, thanks to a new theorem by Solomon W. Golomb (Appendix).

Theorem 4.[1] For any rate $1/n$ linear recursive systematic convolutional code generated by a nonprimitive feedback polynomial, the upper bound in Theorem 3 cannot be achieved, i.e.,

$$d_2^p < (n-1)(2^{m-1}+2)$$

Proof. Using the results of Golomb (see the Appendix) for a nonprimitive feedback polynomial, there are more than two cycles (zero-input loops) in LFSR. The "zero cycle" has weight zero, and the weights of other cycles are nonzero. Thus, the weight of each cycle due to the results of the Appendix is strictly less than $(n-1)2^{m-1}$. If we enter from the all-zero state with input weight-1 to one of the cycles of the shift register, then we have to leave the same cycle to the all-zero state with input weight-1, as discussed in Theorem 1. Thus, $d_2^p < (n-1)(2^{m-1}+2)$. ❐

Theorem 5. For any rate $b/b+1$ linear recursive systematic convolutional code generated by a nonprimitive feedback polynomial, the upper bound in Theorem 1 cannot be excceded, i.e.,

$$d_2^p \leq \lfloor \frac{2^{m-1}}{b} \rfloor + 2$$

Proof. Again using the results of the Appendix, there is a "zero cycle" with weight zero and at least two cycles with nonzero weights, say q cycles with weights $w_1, w_2, \cdots, w_q$. The sum of the weights of all cycles is exactly 2^{m-1}, i.e., $\sum w_i = 2^{m-1}$. For a $b/b+1$ code, we have b weight-1 symbols. Suppose that with b_i of these weight-1 symbols we enter from the all-zero state to the ith cycle with weight w_i; then we have to leave the same cycle to the all-zero state with the same b_i symbols for $i = 1, 2, \cdots, q$, such that $\sum b_i = b$. Based on the discussion in the proof of Theorem 1, the largest achievable minimum output weight of codewords corresponding to weight-2 sequences is $\min(w_1/b_1, w_2/b_2, \cdots, w_q/b_q) + 2$. But it is easy to show that $\min(w_1/b_1, w_2/b_2, \cdots, w_q/b_q) \leq (\sum w_i / \sum b_i) = 2^{m-1}/b$. ❐

[1] The proofs of Theorems 4 and 5 are based on a result by S. W. Golomb (see the Appendix), University of Southern California, Los Angeles, California, 1995. Theorem 4 and Corollary 2 were proved for more general cases when the code is generated by multiple LFSRs by R. J. McEliece, Communications Systems and Research Section, Jet Propulsion Laboratory, Pasadena, California, and California Institute of Technology, Pasadena, California, 1995, using a state-space approach.

Corollary 2. For any rate b/n linear recursive systematic convolutional code generated by a non-primitive feedback polynomial, the upper bound in Corollary 1 cannot be exceeded.

Proof. The proof is similar to the Proof of Theorem 5, but now $\sum w_i = (n-b)2^{m-1}$. ❐

IV. Turbo Decoding for Multiple Codes

In [9] we described a new turbo decoding scheme for q codes based on approximating the optimum bit decision rule. The scheme is based on solving a set of nonlinear equations given by ($q = 3$ is used to illustrate the concept)

$$\left.\begin{aligned}
\tilde{L}_{0k} &= 2\rho \mathbf{y}_{0k} \\
\tilde{L}_{1k} &= \log \frac{\sum_{\mathbf{u}:u_k=1} P(\mathbf{y}_1|\mathbf{u}) \prod_{j\neq k} e^{u_j(\tilde{L}_{0j}+\tilde{L}_{2j}+\tilde{L}_{3j})}}{\sum_{\mathbf{u}:u_k=0} P(\mathbf{y}_1|\mathbf{u}) \prod_{j\neq k} e^{u_j(\tilde{L}_{0j}+\tilde{L}_{2j}+\tilde{L}_{3j})}} \\
\tilde{L}_{2k} &= \log \frac{\sum_{\mathbf{u}:u_k=1} P(\mathbf{y}_2|\mathbf{u}) \prod_{j\neq k} e^{u_j(\tilde{L}_{0j}+\tilde{L}_{1j}+\tilde{L}_{3j})}}{\sum_{\mathbf{u}:u_k=0} P(\mathbf{y}_2|\mathbf{u}) \prod_{j\neq k} e^{u_j(\tilde{L}_{0j}+\tilde{L}_{1j}+\tilde{L}_{3j})}} \\
\tilde{L}_{3k} &= \log \frac{\sum_{\mathbf{u}:u_k=1} P(\mathbf{y}_3|\mathbf{u}) \prod_{j\neq k} e^{u_j(\tilde{L}_{0j}+\tilde{L}_{1j}+\tilde{L}_{2j})}}{\sum_{\mathbf{u}:u_k=0} P(\mathbf{y}_3|\mathbf{u}) \prod_{j\neq k} e^{u_j(\tilde{L}_{0j}+\tilde{L}_{1j}+\tilde{L}_{2j})}}
\end{aligned}\right\} \tag{14}$$

for $k = 1, 2, \cdots, N$. In Eq. (14), $\tilde{L}_{ik}$ represents extrinsic information and $\mathbf{y}_i$, $i = 0, 1, 2, 3$ are the received observation vectors corresponding to $\mathbf{x}_i$, $i = 0, 1, 2, 3$ (see Fig. 1), where $\rho = \sqrt{2rE_b/N_0}$, if we assume the channel noise samples have unit variance per dimension. The final decision is then based on

$$L_k = \tilde{L}_{0k} + \tilde{L}_{1k} + \tilde{L}_{2k} + \tilde{L}_{3k} \tag{15}$$

which is passed through a hard limiter with a zero threshold.

The above set of nonlinear equations is derived from the optimum bit decision rule, i.e.,

$$L_k = \log \frac{\sum_{\mathbf{u}:u_k=1} P(\mathbf{y}_0|\mathbf{u})P(\mathbf{y}_1|\mathbf{u})P(\mathbf{y}_2|\mathbf{u})P(\mathbf{y}_3|\mathbf{u})}{\sum_{\mathbf{u}:u_k=0} P(\mathbf{y}_0|\mathbf{u})P(\mathbf{y}_1|\mathbf{u})P(\mathbf{y}_2|\mathbf{u})P(\mathbf{y}_3|\mathbf{u})} \tag{16}$$

using the following approximation:

$$P(\mathbf{u}|\mathbf{y}_i) \approx \prod_{k=1}^{N} \frac{e^{u_k \tilde{L}_{ik}}}{1 + e^{\tilde{L}_{ik}}} \tag{17}$$

Note that, in general, $P(\mathbf{u}|\mathbf{y}_i)$ is not separable. The smaller the Kullback cross entropy [3,17] between right and left distributions in Eq. (17), the better is the approximation and, consequently, the closer is turbo decoding to the optimum bit decision.

We attempted to solve the nonlinear equations in Eq. (14) for $\tilde{\mathbf{L}}_1$, $\tilde{\mathbf{L}}_2$, and $\tilde{\mathbf{L}}_3$ by using the iterative procedure

$$\tilde{L}_{1k}^{(m+1)} = \alpha_1^{(m)} \log \frac{\sum_{\mathbf{u}:u_k=1} P(\mathbf{y}_1|\mathbf{u}) \prod_{j\neq k} e^{u_j(\tilde{L}_{0j}+\tilde{L}_{2j}^{(m)}+\tilde{L}_{3j}^{(m)})}}{\sum_{\mathbf{u}:u_k=0} P(\mathbf{y}_1|\mathbf{u}) \prod_{j\neq k} e^{u_j(\tilde{L}_{0j}+\tilde{L}_{2j}^{(m)}+\tilde{L}_{3j}^{(m)})}} \tag{18}$$

for $k = 1, 2, \cdots, N$, iterating on m. Similar recursions hold for $\tilde{L}_{2k}^{(m)}$ and $\tilde{L}_{3k}^{(m)}$. The gain $\alpha_1^{(m)}$ should be equal to one, but we noticed experimentally that better convergence can be obtained by optimizing this gain for each iteration, starting from a value less than 1 and increasing toward 1 with the iterations, as is often done in simulated annealing methods. We start the recursion with the initial condition[2] $\tilde{\mathbf{L}}_1^{(0)} = \tilde{\mathbf{L}}_2^{(0)} = \tilde{\mathbf{L}}_3^{(0)} = \tilde{\mathbf{L}}_0$. For the computation of Eq. (18), we use a modified MAP algorithm[3] with permuters (direct and inverse) where needed, as shown in Fig. 4. The MAP algorithm [1] always starts and ends at the all-zero state since we always terminate the trellis as described in [8]. We assumed $\pi_1 = I$ (identity); however, any π_1 can be used. The overall decoder is composed of block decoders connected as in Fig. 4, which can be implemented as a pipeline or by feedback. In [10] and [11], we proposed an alternative version of the above decoder that is more appropriate for use in turbo trellis-coded modulation, i.e., set $\tilde{L}_0 = 0$ and consider $\mathbf{y}_0$ as part of $\mathbf{y}_1$. If the systematic bits are distributed among encoders, we use the same distribution for $\mathbf{y}_0$ among the MAP decoders.

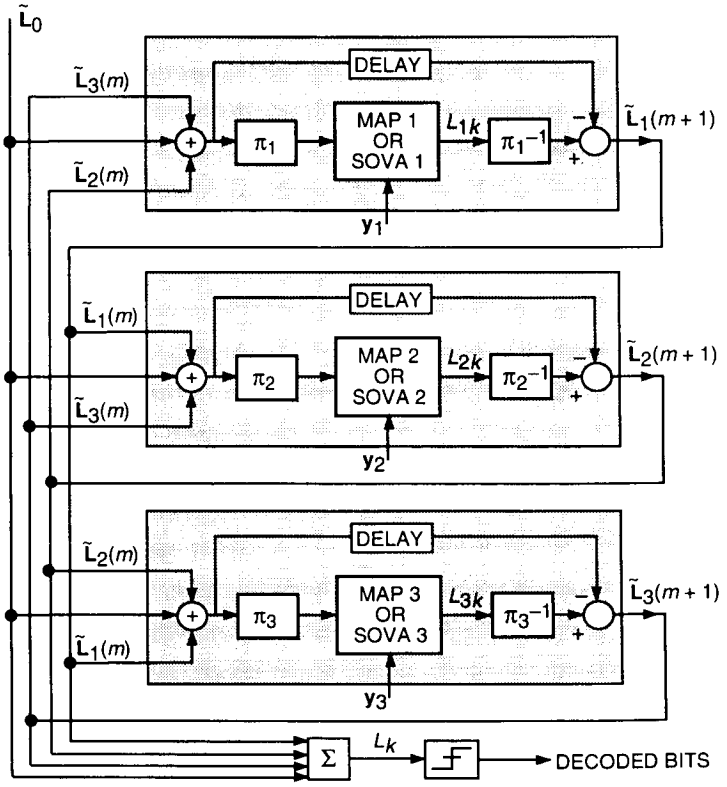

Fig. 4. Multiple turbo decoder structure.

[2] Note that the components of the $\tilde{\mathbf{L}}_i$'s corresponding to the tail bits, i.e., $\tilde{L}_{ik}$ for $k = N+1, \cdots, N+M_i$, are set to zero for all iterations.

[3] The modified MAP algorithm is described in S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-Output Decoding Algorithms in Iterative Decoding of Parallel Concatenated Convolutional Codes," submitted to *ICC '96*.

At this point, further approximation for turbo decoding is possible if one term corresponding to a sequence $\mathbf{u}$ dominates other terms in the summation in the numerator and denominator of Eq. (18). Then the summations in Eq. (18) can be replaced by "maximum" operations with the same indices, i.e., replacing $\sum_{\mathbf{u}:u_k=i}$ with $\max_{\mathbf{u}:u_k=i}$ for $i = 0, 1$. A similar approximation can be used for $\tilde{L}_{2k}$ and $\tilde{L}_{3k}$ in Eq. (14). This suboptimum decoder then corresponds to a turbo decoder that uses soft output Viterbi (SOVA)-type decoders rather than MAP decoders. Further approximations, i.e., replacing $\sum$ with max, can also be used in the MAP algorithm.[4]

A. Decoding Multiple Input Convolutional Codes

If the rate b/n constituent code is not equivalent to a punctured rate $1/n'$ code or if turbo trellis-coded modulation is used, we can first use the symbol MAP algorithm[5] to compute the log-likelihood ratio of a symbol $\mathbf{u} = u_1, u_2, \cdots, u_b$ given the observation $\mathbf{y}$ as

$$\lambda(\mathbf{u}) = \log \frac{P(\mathbf{u}|\mathbf{y})}{P(\mathbf{0}|\mathbf{y})}$$

where $\mathbf{0}$ corresponds to the all-zero symbol. Then we obtain the log-likelihood ratios of the jth bit within the symbol by

$$L(u_j) = \log \frac{\sum_{\mathbf{u}:u_j=1} e^{\lambda(\mathbf{u})}}{\sum_{\mathbf{u}:u_j=0} e^{\lambda(\mathbf{u})}}$$

In this way, the turbo decoder operates on bits, and bit, rather than symbol, interleaving is used.

V. Performance and Simulation Results

The BER performance of these codes was evaluated by using transfer function bounds [4,6,12]. In [12], it was shown that transfer function bounds are very useful for SNRs above the cutoff rate threshold and that they cannot accurately predict performance in the region between cutoff rate and capacity. In this region, the performance was computed by simulation.

Figure 5 shows the performance of turbo codes with m iterations and an interleaver size of $N = 16,384$. The following codes are used as examples:

(1) Rate 1/2 Turbo Codes.

Code A: Two 16-state, rate 2/3 constituent codes are used to construct a rate 1/2 turbo code as shown in Fig. 6. The (worst-case) minimum codeword weights, d_i, corresponding to a weight-i input sequence for this code are d_{ef}=14, d_3=7, d_4=8, d_5=5=d_{min}, and d_6=6.

[4] Ibid.

[5] Ibid.

Code B: A rate 1/2 turbo code also was constructed by using a differential encoder and a 32-state, rate 1/2 code, as shown in Fig. 7. This is an example where the systematic bits for both encoders are not transmitted. The (worst-case) minimum codeword weights, d_i, corresponding to a weight-i input sequence for this code are d_{ef}=19, d_4=6=d_{min}, d_6=9, d_8=8, and d_{10}=11. The output weights for odd i are large.

(2) Rate 1/3 Turbo Code.

Code C: Two 16-state, rate 1/2 constituent codes are used to construct a rate 1/3 turbo code as shown in Fig. 8. The (worst-case) minimum codeword weights, d_i, corresponding to a weight-i input sequence for this code are d_{ef}=22, d_3 = 11, d_4=12, d_5 = 9 = d_{min}, d_6=14, and d_7=15.

(3) Rate 1/4 Turbo Code.

Code D: Two 16-state, rate 1/2 and rate 1/3 constituent codes are used to construct a rate 1/4 turbo code, as shown in Fig. 9, with d_{ef} = 32, d_3 = 15 = d_{min}, d_4 = 16, d_5 = 17, d_6 = 16, and d_7 = 19.

(4) Rate 1/15 Turbo Code.

Code E: Two 16-state, rate 1/8 constituent codes are used to construct a rate 1/15 turbo code, $(1, g_1/g_0, g_2/g_0, g_3/g_0, g_4/g_0, g_5/g_0, g_6/g_0, g_7/g_0)$ and $(g_1/g_0, g_2/g_0, g_3/g_0, g_4/g_0, g_5/g_0, g_6/g_0, g_7/g_0)$, with $g_0 = (23)_{octal}$, $g_1 = (21)_{octal}$, $g_2 = (25)_{octal}$, $g_3 = (27)_{octal}$, $g_4 = (31)_{octal}$, $g_5 = (33)_{octal}$, $g_6 = (35)_{octal}$, and $g_7 = (37)_{octal}$. The (worst-case) minimum codeword weights, d_i, corresponding to a weight i input sequence for this code are d_{ef}=142, d_3=39=d_{min}, d_4=48, d_5=45, d_6=50, and d_7=63.

The simulation performance of other codes reported in this article is still in progress.

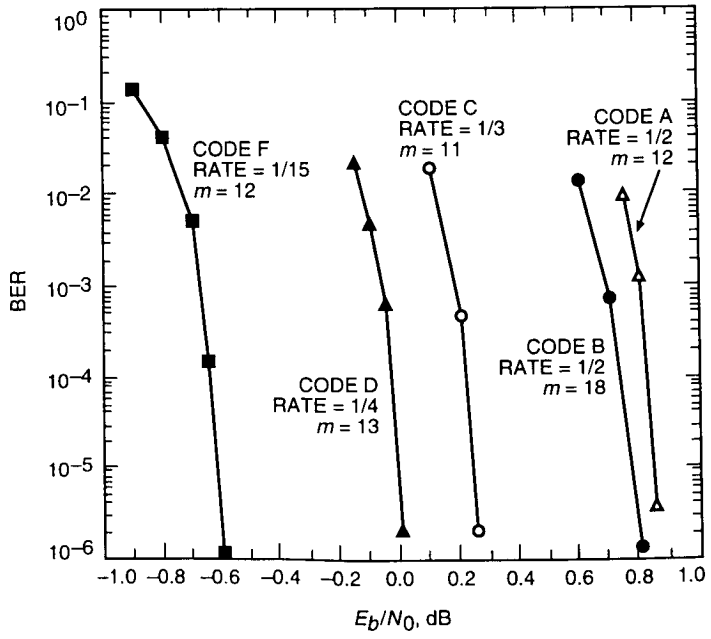

Fig. 5. Performance of turbo codes.

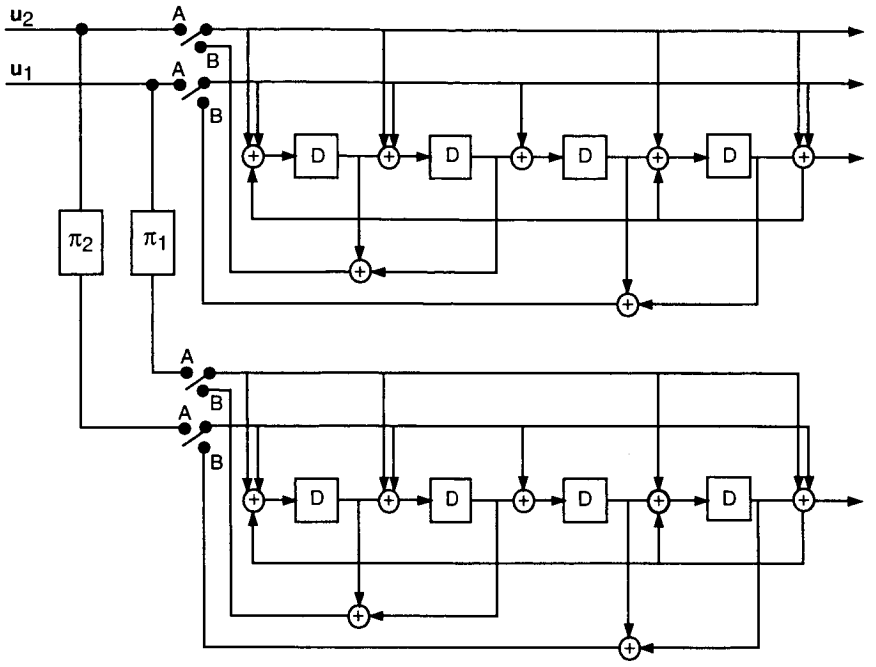

Fig. 6. Rate 1/2 turbo code constructed from two codes ($h_0 = 23$, $h_1 = 35$, $h_2 = 33$).

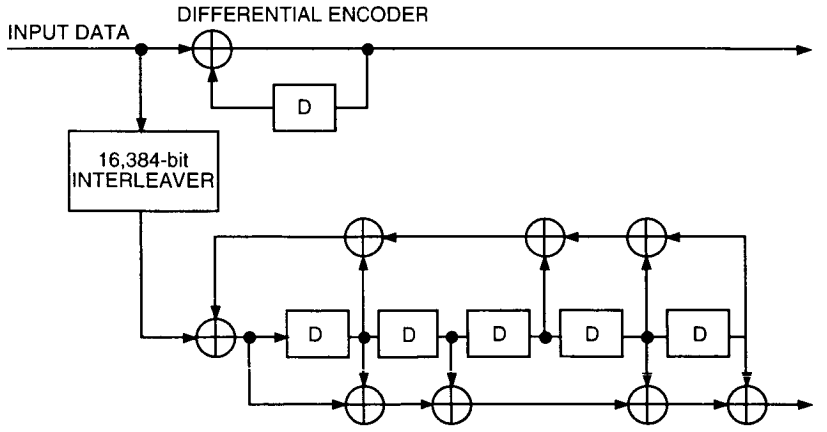

Fig. 7. Rate 1/2 turbo code constructed from a differential encoder and code ($g_0 = 67$, $g_1 = 73$).

VI. Turbo Trellis-Coded Modulation

A pragmatic approach for turbo codes with multilevel modulation was proposed in [16]. Here we propose a different approach that outperforms the results in [16] when M-ary quadrature amplitude modulation (M-QAM) or M-ary phase shift keying (MPSK) modulation is used. A straightforward method for the use of turbo codes for multilevel modulation is first to select a rate $b/(b+1)$ constituent code, where the outputs are mapped to a 2^{b+1}-level modulation based on Ungerboeck's set partitioning method [21] (i.e., we can use Ungerboeck's codes with feedback). If MPSK modulation is used, for every b bits at the input of the turbo encoder, we transmit two consecutive 2^{b+1} phase-shift keying (PSK) signals, one per each encoder output. This results in a throughput of $b/2$ bits/s/Hz. If M-QAM modulation is

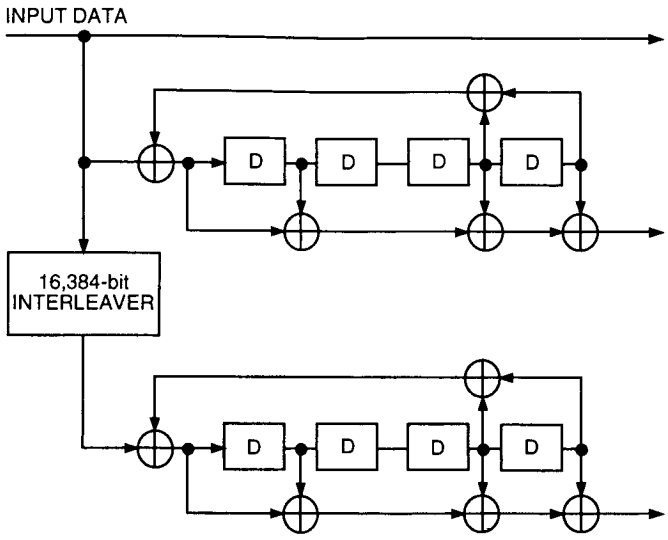

Fig. 8. Rate 1/3 turbo code constructed from two identical codes ($g_0 = 23$, $g_1 = 33$).

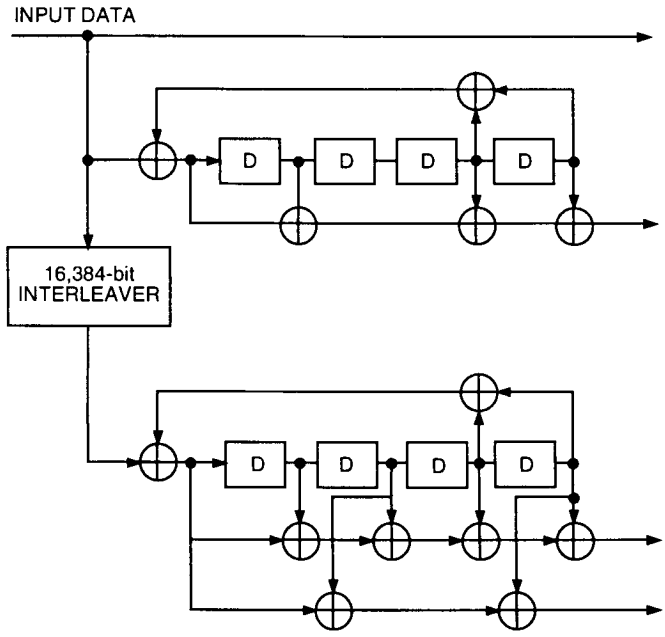

Fig. 9. Rate 1/4 turbo code constructed from two codes ($g_0 = 23$, $g_1 = 33$) and ($g_0 = 23$, $g_1 = 37$, $g_2 = 25$).

used, we map the $b + 1$ outputs of the first component code to the 2^{b+1} in-phase levels (I-channel) of a 2^{2b+2}-QAM signal set and the $b + 1$ outputs of the second component code to the 2^{b+1} quadrature levels (Q-channel). The throughput of this system is b bits/s/Hz.

First, we note that these methods require more levels of modulation than conventional trellis-coded modulation (TCM), which is not desirable in practice. Second, the input information sequences are used twice in the output modulation symbols, which also is not desirable. An obvious remedy is to puncture the output symbols of each trellis code and select the puncturing pattern such that the output symbols of the turbo code contain the input information only once. If the output symbols of the first encoder are

punctured, for example as $101010\cdots$, the puncturing pattern of the second encoder must be nonuniform to guarantee that all information symbols are used, and it depends on the particular choice of interleaver. Now, for example, for 2^{b+1} PSK, a throughput b can be achieved. This method has two drawbacks: It complicates the encoder and decoder, and the reliability of punctured symbols may not be fully estimated at the decoder. A better remedy, for rate $b/(b+1)$ (b even) codes, is discussed in the next section.

A. A New Method to Construct Turbo TCM

For a $q = 2$ turbo code with rate $b/(b+1)$ constituent encoders, select the $b/2$ systematic outputs and puncture the rest of the systematic outputs, but keep the parity bit of the $b/(b+1)$ code (note that the rate $b/(b+1)$ code may have been obtained already by puncturing a rate 1/2 code). Then do the same to the second constituent code, but select only those systematic bits that were punctured in the first encoder. This method requires at least two interleavers: The first interleaver permutes the bits selected by the first encoder and the second interleaver those punctured by the first encoder. For MPSK (or M-QAM), we can use $2^{1+b/2}$ PSK symbols (or $2^{1+b/2}$ QAM symbols) per encoder and achieve throughput $b/2$. For M-QAM, we can also use $2^{1+b/2}$ levels in the I-channel and $2^{1+b/2}$ levels in the Q-channel and achieve a throughput of b bits/s/Hz. These methods are equivalent to a multidimensional trellis-coded modulation scheme (in this case, two multilevel symbols per branch) that uses $2^{b/2} \times 2^{1+b/2}$ symbols per branch, where the first symbol in the branch (which depends only on uncoded information) is punctured. Now, with these methods, the reliability of the punctured symbols can be fully estimated at the decoder. Obviously, the constituent codes for a given modulation should be redesigned based on the Euclidean distance. In this article, we give an example for $b = 2$ with 16-QAM modulation where, for simplicity, we can use the 2/3 codes in Table 1 with Gray code mapping. Note that this may result in suboptimum constituent codes for multilevel modulation. The turbo encoder with 16 QAM and two clock-cycle trellis termination is shown in Fig. 10. The BER performance of this code with the turbo decoding structure for two codes discussed in Section IV is given in Fig. 11. For permutations π_1 and π_2, we used S-random permutations [9] with $S = 40$ and $S = 32$, with a block size of 16,384 bits. For 8 PSK, we used two 16-state, rate 4/5 codes given in Section V to achieve throughput 2. The parallel concatenated trellis codes with 8 PSK and two clock-cycle trellis termination is shown in Fig. 12. The BER performance of this code is given in Fig. 13. For 64 QAM, we used two 16-state, rate 4/5 codes given in Section V to achieve throughput 4. The parallel concatenated trellis codes with 64 QAM and two clock-cycle trellis termination is shown in Fig. 14. The BER performance of this code is given in Fig. 15. For permutations π_1, π_2, π_3, and π_4 in Figs. 10, 12, and 14, we used random permutations, each with a block size of 4096 bits. As was discussed above, there is no need to use four permutations; two permutations suffice, and they may even result in a better performance. Extension of the described method for construction of turbo TCM based on Euclidean distance is straightforward.[6]

VII. Conclusions

In this article, we have shown that powerful turbo codes can be obtained if multiple constituent codes are used. We reviewed an iterative decoding method for multiple turbo codes by approximating the optimum bit decision rule. We obtained an upper bound on the effective free Euclidean distance of b/n codes. We found the best rate 2/3, 3/4, 4/5, and 1/3 constituent codes that can be used in the design of multiple turbo codes. We proposed new schemes that can be used for power- and bandwidth-efficient turbo trellis-coded modulation.

[6] This is discussed in S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Parallel Concatenated Trellis Coded Modulation," submitted to *ICC '96*.

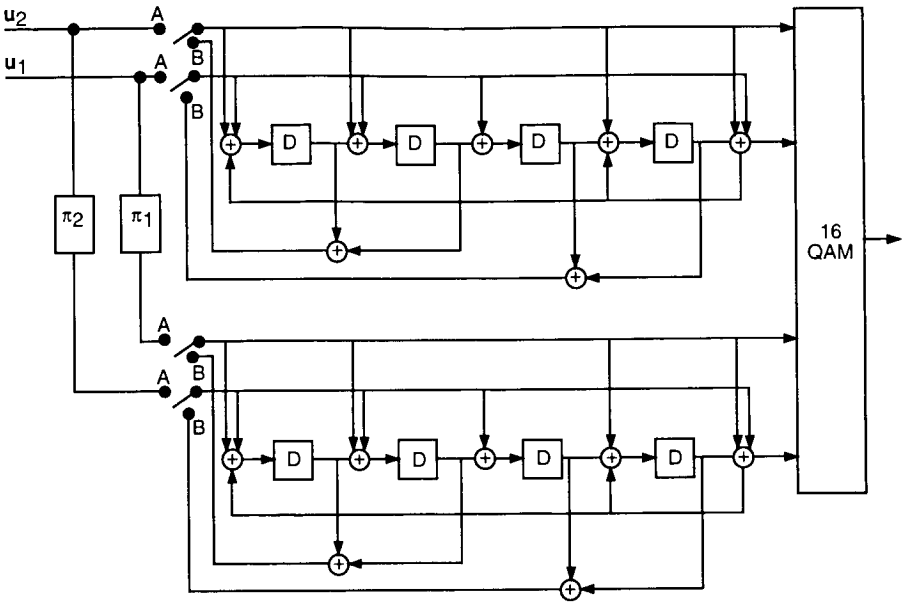

Fig. 10. Turbo trellis-coded modulation, 16 QAM, 2 bits/s/Hz.

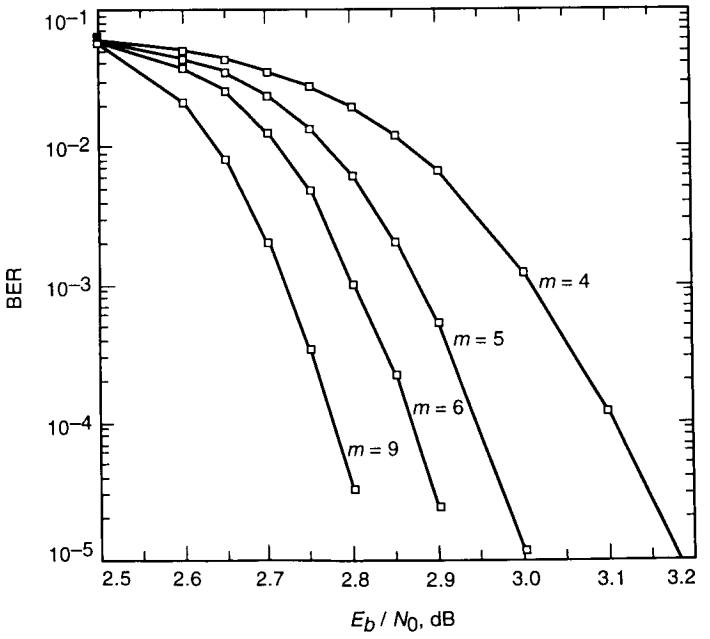

Fig. 11. BER performance of turbo trellis-coded modulation, 16 QAM, 2 bits/s/Hz.

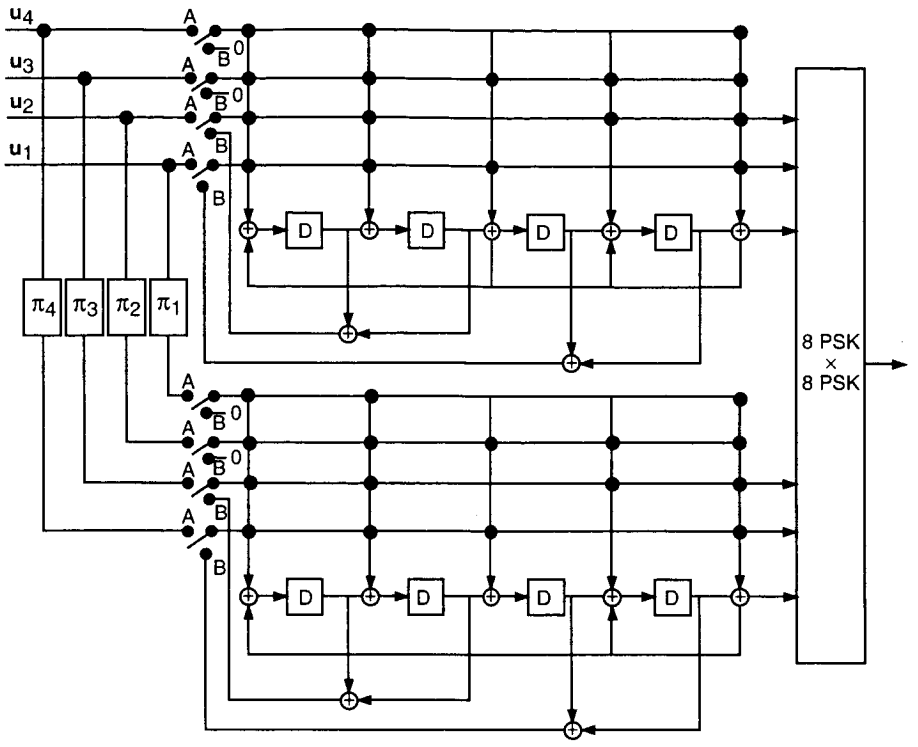

Fig. 12. Parallel concatenated trellis-coded modulation, 8 PSK, 2 bits/s/Hz.

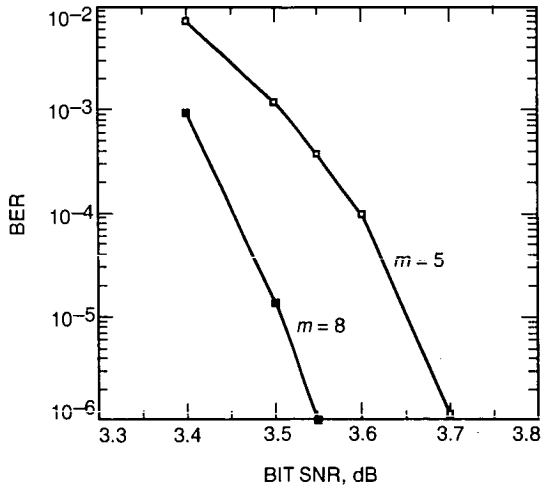

Fig. 13. BER performance of parallel concatenated trellis-coded modulation, 8 PSK, 2 bits/s/Hx.

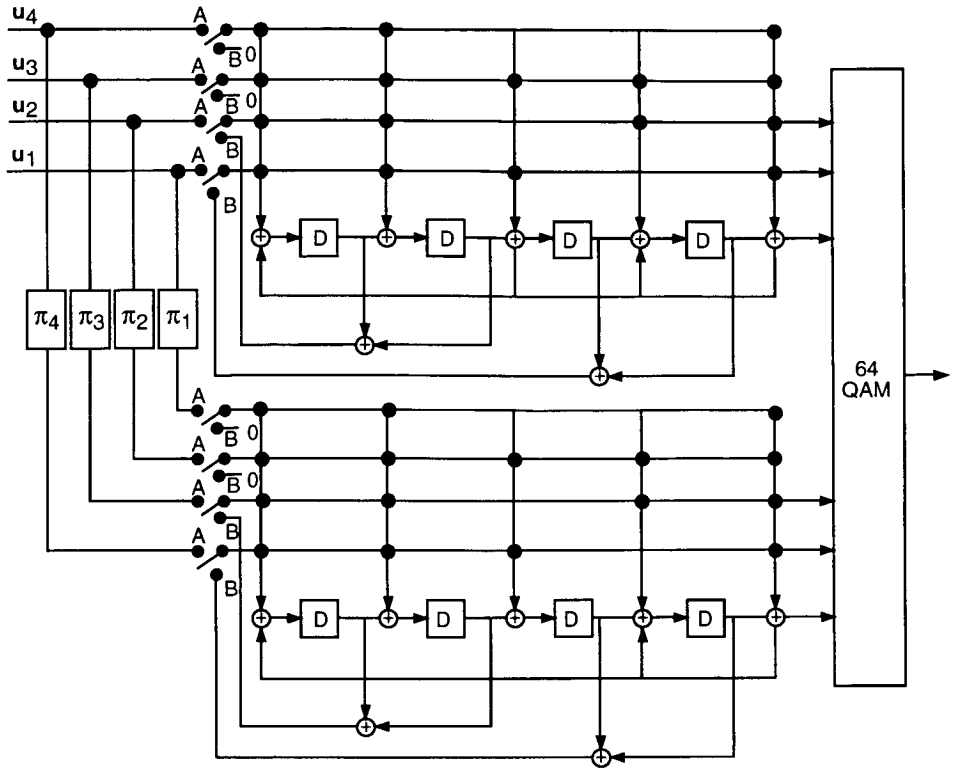

Fig. 14. Parallel concatenated trellis-coded modulation, 64 QAM, 4 bits/s/Hz.

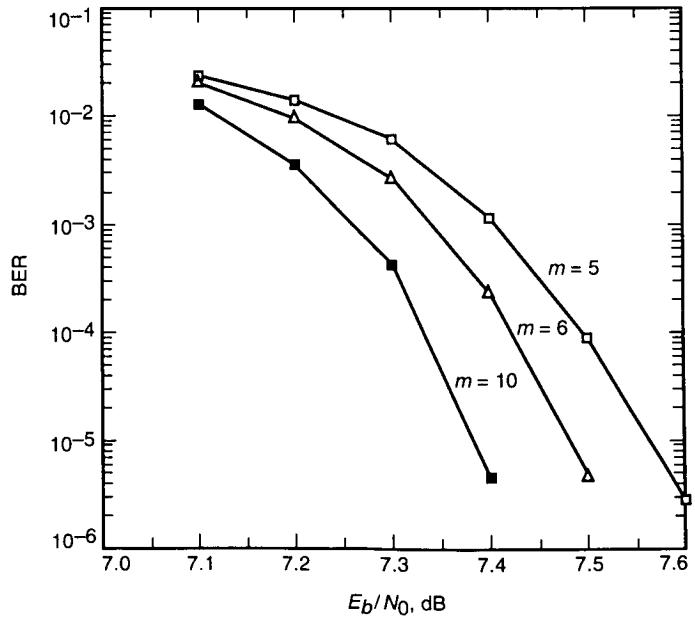

Fig. 15. BER performance of parallel concatenated trellis-coded modulation, 64 QAM, 4 bits/s/Hz.

Acknowledgments

The authors are grateful to S. Dolinar and R. J. McEliece for their helpful comments throughout this article, to S. Benedetto and G. Montorsi for their helpful comments on the turbo trellis-coded modulation section, and special thanks to S. W. Golomb for his contribution, as reported in the Appendix.

References

[1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, 1974.

[2] G. Battail, C. Berrou, and A. Glavieux, "Pseudo-Random Recursive Convolutional Coding for Near-Capacity Performance," *Comm. Theory Mini-Conference, GLOBECOM '93*, Houston, Texas, December 1993.

[3] G. Battail and R. Sfez, "Suboptimum Decoding Using the Kullback Principle," *Lecture Notes in Computer Science*, vol. 313, pp. 93–101, 1988.

[4] S. Benedetto, "Unveiling Turbo Codes," IEEE Communication Theory Workshop, Santa Cruz, California, April 23–26, 1995.

[5] S. Benedetto and G. Montorsi, "Design of Parallel Concatenated Convolutional Codes," to be published in *IEEE Transactions on Communications*, 1996.

[6] S. Benedetto and G. Montorsi, "Performance Evaluation of Turbo-Codes," *Electronics Letters*, vol. 31, no. 3, pp. 163–165, February 2, 1995.

[7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding: Turbo Codes," *Proc. 1993 IEEE International Conference on Communications*, Geneva, Switzerland, pp. 1064–1070, May 1993.

[8] D. Divsalar and F. Pollara, "Turbo Codes for Deep-Space Communications," *The Telecommunications and Data Acquisition Progress Report 42-120, October–December 1994*, Jet Propulsion Laboratory, Pasadena, California, pp. 29–39, February 15, 1995, URL http://edms-www.jpl.nasa.gov/tda/progress_report/42-120/120D.pdf.

[9] D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications," *The Telecommunications and Data Acquisition Progress Report 42-121, January–March 1995*, Jet Propulsion Laboratory, Pasadena, California, pp. 66–77, May 15, 1995, URL http://edms-www.jpl.nasa.gov/tda/progress_report/42-121/121T.pdf.

[10] D. Divsalar and F. Pollara, "Turbo Codes for PCS Applications," *Proceedings of IEEE ICC'95*, Seattle, Washington, pp. 54–59, June 1995.

[11] D. Divsalar and F. Pollara, "Turbo Codes for Deep-Space Communications," IEEE Communication Theory Workshop, Santa Cruz, California, April 23–26, 1995.

[12] D. Divsalar, S. Dolinar, R. J. McEliece,and F. Pollara, "Transfer Function Bounds on the Performance of Turbo Codes," *MILCOM 95*, San Diego, California, November 5–8, 1995.

[13] S. Dolinar and D. Divsalar, "Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations," *The Telecommunications and Data Acquisition Progress Report 42-122, April–June 1995*, Jet Propulsion Laboratory, Pasadena, California, pp. 56–65, August 15, 1995, URL http://edms-www.jpl.nasa.gov/tda/progress_report/42-122/122B.pdf.

[14] S. W. Golomb, *Shift Register Sequences*, Revised Edition, Laguna Beach, California: Aegean Park Press, 1982.

[15] J. Hagenauer and P. Robertson, "Iterative (Turbo) Decoding of Systematic Convolutional Codes With the MAP and SOVA Algorithms," *Proc. of the ITG Conference on Source and Channel Coding*, Frankfurt, Germany, pp. 1–9, October 1994.

[16] S. LeGoff, A. Glavieux, and C. Berrou, "Turbo Codes and High Spectral Efficiency Modulation," *Proceedings of IEEE ICC'94*, New Orleans, Louisiana, pp. 645–651, May 1–5, 1994.

[17] M. Moher, "Decoding Via Cross-Entropy Minimization," *Proceedings GLOBECOM '93*, Houston, Texas, pp. 809–813, December 1993.

[18] A. S. Barbulescu and S. S. Pietrobon, "Terminating the Trellis of Turbo-Codes in the Same State," *Electronics Letters*, vol. 31, no. 1, pp. 22–23, January 1995.

[19] P. Robertson, "Illuminating the Structure of Code and Decoder of Parallel Concatenated Recursive Systematic (Turbo) Codes," *Proceedings GLOBECOM '94*, San Francisco, California, pp. 1298–1303, December 1994.

[20] G. D. Forney, Jr., "Convolutional Codes I: Algebraic Structure," *IEEE Transactions on Information Theory*, vol. IT-16, pp. 720–738, November 1970.

[21] G. Ungerboeck, "Channel Coding With Multi-Level Phase Signals," *IEEE Transactions on Information Theory*, vol. IT-28, pp. 55–67, January 1982.

Appendix

A Bound on the Weights of Shift Register Cycles[1]

I. Introduction

A maximum-length linear shift register sequence—a pseudonoise (PN)-sequence or a maximal length (m)-sequence—of degree m has period $p = 2^m - 1$, with 2^{m-1} ones and $2^{m-1} - 1$ zeroes in each period. Thus, the weight of a PN cycle is 2^{m-1}. From a linear shift register whose characteristic polynomial is reducible, or irreducible but not primitive, in addition to the "zero-cycle" of period 1, there are several other possible cycles, depending on the initial state of the register, and each of these cycles has a period less than $2^m - 1$.

The question is whether it is possible for any cycle, from any linear shift register of degree m, to have a weight greater than 2^{m-1}. We shall show that the answer is "no" and that this result does not depend on the shift register being linear.

II. The Main Result

Let S be any feedback shift register of length m, linear or not. We need not even specify that the shift register produce "pure" cycles, without branches. We will use only the fact that each state of the shift register has a unique successor state. For any given initial state, we define the length L of the string starting from that state to be the number of states, counting from the initial state, prior to the second appearance of any state in the string. (In the case of branchless cycles, this is the length of the cycle with the given initial state.)

The string itself is this succession of states of length L. The corresponding string sequence is the sequence of 0's and 1's appearing in the right-most position of the register (or any other specific position of the register that has been agreed upon) as the string goes through its succession of L states.

Theorem 1. From a feedback shift register S of length m, the maximum number of 1's that can appear in any string sequence is 2^{m-1}.

Proof. There are 2^m possible states of the shift register S altogether. In any fixed position of the shift register, 2^{m-1} of these states have a 0 and 2^{m-1} states have a 1. In a string of length L, all L of the states are distinct, and in any given position of the register, neither 0 nor 1 can occur more than 2^{m-1} times. In particular, the weight of a string sequence from a register of length m cannot exceed 2^{m-1}. ❐

Corollary 1. No cycle from a feedback shift register of length m can have weight exceeding 2^{m-1}.

[1] S. W. Golomb, personal communication, University of Southern California, Los Angeles, California, 1995.