Coset Codes Viewed as Terminated Convolutional Codes

Marc P. C. Fossorier, *Member, IEEE*, and Shu Lin, *Fellow, IEEE*

***Abstract*—In this paper, coset codes are considered as terminated convolutional codes. Based on this approach, three new general results are presented. First, it is shown that the iterative squaring construction can equivalently be defined from a convolutional code whose trellis terminates. This convolutional code determines a simple encoder for the coset code considered, and the state and branch labelings of the associated trellis diagram become straightforward. Also, from the generator matrix of the code in its convolutional code form, much information about the trade-off between the state connectivity and complexity at each section, and the parallel structure of the trellis, is directly available. Based on this generator matrix, it is shown that the parallel branches in the trellis diagram of the convolutional code represent the same coset code C_1 of smaller dimension and shorter length. Utilizing this fact, a two-stage optimum trellis decoding method is devised. The first stage decodes C_1 while the second stage decodes the associated convolutional code, using the branch metrics delivered by stage 1. Finally, a bidirectional decoding of each received block starting at both ends is presented. If about the same number of computations is required, this approach remains very attractive from a practical point of view as it roughly doubles the decoding speed. This fact is particularly interesting whenever the second half of the trellis is the mirror image of the first half, since the same decoder can be implemented for both parts.**

I. Introduction

ANY binary convolutional code of constraint length L (or memory order $L-1$) whose associated trellis diagram terminates after encoding x blocks of k information bits into x blocks of n transmitted symbols generates an $(n(x+L-1), kx)$ binary block code. Although the resulting code generally has a smaller minimum Hamming distance than the best $(n(x+L-1), kx)$ block codes, both its encoding and decoding can be realized very efficiently. The encoder consits of ν memory elements and binary adders whose outputs are serialized with a low-order multiplexer [1]. The Viterbi algorithm (VA) working on the associated 2^{ν}-state trellis diagram can be implemented at the decoding end [2]. On the other hand, while any good binary block code has a trellis structure and can also be decoded with the VA [3], the implementation of this approach is often not practical due to the large state and branch complexities. In general, the number of states in the trellis diagram associated with an (N, K) block code with large N, K and $N-K$ is very large and upper bounded by $2^{\min(K, N-K)}$. Finding good block codes with simple trellis structure remains an area of on-going research.

In [4], Forney proposed different decompositions for families of block codes, namely coset codes, for which the associated trellis diagrams have a much smaller number of states than $2^{\min(K, N-K)}$. These decompositions are referred as iterative squaring constructions and applied primarily to Reed–Muller (RM) codes. Implementing the Viterbi decoding algorithm for this class of decomposable codes becomes manageable for moderate code length. The associated trellis diagram consists of structurally identical parallel subtrellises without cross-connections between them, each subtrellis corresponding to a particular coset of the code. However, the coset decomposition and the associated trellis structure presented in [4] provide little information about the state and branch labelings of each subtrellis.

In this paper, we first propose an equivalent representation of the decomposition of RM codes based on the iterative squaring construction to that given in [4]. From this new representation, we show that each subtrellis corresponds to the same embedded convolutional code, which implicitly determines the state and branch labelings. Also, simple encoders for RM codes can be devised from this representation. Generalizations to decomposable codes other than RM codes are briefly discussed. The second part of the paper presents a two-stage optimum trellis decoding for coset codes. This decoding is based on the fact that each transition in the embedded convolutional code trellis represents the same smaller coset code. This smaller code is therefore first decoded and then the survivor determines the associated branch metric for the trellis decoding of the convolutional code. In the last part of the paper, we exploit the fact that the trellis diagram associated with any block code teminates and show that a bi-directional decoding of the received sequence is possible. This decoding method reduces the decoding delay by a factor of two. This approach is particularly interesting whenever the same trellis diagram can be used for searching from both ends. We show that such is the case for RM codes.

II. Iterative Squaring Constructions of RM Codes

A. Definitions

We denote by $RM(r, m)$ the rth order RM code with generator matrix $G_{(r,m)}$, length $N_{(r,m)} = 2^m$, dimension $K_{(r,m)} = \Sigma_{i=0}^{r} \binom{m}{i}$ and minimum distance $d_{(r,m)} = 2^{m-r}$. It is well known that $RM(r, m)$ is formed by $2^{K_{(r,m)} - K_{(r-1,m)}}$

Paper approved by T. M. Aulin, the Editor for Coding and Communications Theory of the IEEE Communications Society. Manuscript received March 15, 1995; revised July 21, 1995 and January 20, 1996. This work was supported in part by the National Science Foundation under Grants NCR 91-15400 and NCR 94-15374 and NASA Grant NAG 5-931.

The authors are with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822 USA.

Publisher Item Identifier S 0090-6778(96)07083-3.

cosets of $RM(r-1,m)$, so that [5]

$$G_{(r,m)} = \begin{bmatrix} C_{(r,m)} \\ G_{(r-1,m)} \end{bmatrix} \tag{1}$$

where $C_{(r,m)}$ generates the coset representatives of the partition $RM(r,m)/RM(r-1,m)$ [4]. $C_{(r,m)}$ is easily obtained from the Boolean representation of $G_{(r,m)}$ [5, p. 371]. By convention, we assume $G_{(-1,m)} = [0]$, which imposes $C_{(0,m)} = G_{(0,m)}$, and define for $r > m$, $C_{(r,m)} = [0]$.

Any convolutional code of rate k/n and constraint length L is defined by an infinite matrix of the form [1]

$$G = \begin{bmatrix} P_{L0} & P_{L1} & P_{L2} & \cdots & P_{LL-1} & 0 & 0\cdots \\ 0 & P_{L0} & P_{L1} & \ddots & P_{LL-2} & P_{LL-1} & 0 & \cdots \\ \vdots & \ddots & \cdots & & \ddots & \cdots & & \end{bmatrix} \tag{2}$$

where each P_{Li} represents an $k \times n$ matrix. We define

$$P_L = [P_{L0} P_{L1} P_{L2} \cdots P_{LL-1}] \tag{3}$$

as the generator pattern of length L associated with the convolutional code.

Utilizing these basic definitions, we explore the iterative squaring construction of RM codes, as defined in [4]. For each level l of decomposition, we show that different convolutional codes can be associated with the trellis diagrams described in [4]. For a given level l, each convolutional code is defined by a generating pattern of different length L, referred to as the l- level generating pattern of length L. Based on each embedded code, the branch and state labelings of the trellis become straightforward. Also, simple encoders for RM codes are devised from each associated convolutional code.

To keep the generality of the results presented in the following sections, many indexes and variables are used to describe different decompositions. According to the previous notations, G refers to the generator matrix of a RM code, C refers to coset selectors and P refers to a generator pattern. For a given decomposition, we express $G_{(r,m)}$ in terms of these three parameters. In general, P defines the convolutional code considered and C describes the number of such codes in parallel. Also, if the convolutional code contains parallel branches, the parameter G is used to represent them. This general guideline is used to describe the decompositions presented in the following sections.

B. l-level Generating Pattern of Length 1

The l-level generating pattern of length 1 is defined as

$$P_{1(r,m)}(l) = G_{(r-l,m-l)} \tag{4}$$

which defines a trivial convolutional code of zero memory order. Its associated trellis diagram contains a single state and the $2^{K_{(r-l,m-l)}}$ parallel branches compose the unique transition. These parallel branches define the $RM(r-l,m-l)$ code.

Based on the one-level squaring construction or equivalently $|u|u+v|$ construction, $G_{(r,m)}$ is of the form [4]

$$G_{(r,m)} = \begin{bmatrix} G_{(r,m-1)} & G_{(r,m-1)} \\ 0 & G_{(r-1,m-1)} \end{bmatrix}. \tag{5}$$

By using (1), (4), and elementary row operations, we can rewrite (5) as

$$G_{(r,m)} = \begin{bmatrix} C_{(r,m-1)} & C_{(r,m-1)} \\ P_{1(r,m)}(1) & 0 \\ 0 & P_{1(r,m)}(1) \end{bmatrix}. \tag{6}$$

After repeating the same decomposition for each matrix of (5) and applying elementary row operations, we obtain

$$G_{(r,m)} = \begin{bmatrix} C_{(r,m-2)} & C_{(r,m-2)} & C_{(r,m-2)} & C_{(r,m-2)} \\ C_{(r-1,m-2)} & C_{(r-1,m-2)} & C_{(r-1,m-2)} & C_{(r-1,m-2)} \\ 0 & C_{(r-1,m-2)} & 0 & C_{(r-1,m-2)} \\ 0 & 0 & C_{(r-1,m-2)} & C_{(r-1,m-2)} \\ P_{1(r,m)}(2) & 0 & 0 & 0 \\ 0 & P_{1(r,m)}(2) & 0 & 0 \\ 0 & 0 & P_{1(r,m)}(2) & 0 \\ 0 & 0 & 0 & P_{1(r,m)}(2) \end{bmatrix}. \tag{7}$$

Equation (7) represents the two-level squaring construction [4]. This simple construction can be repeated iteratively. To the l-level squaring construction corresponds an l-level coset decomposition $RM(r,m-l)/RM(r-1,m-l)/\cdots/RM(r-l,m-l)$ which is described in detail in [4]. For each coset, we obtain 2^l independent sections representing the $RM(r-l,m-l)$ code. The advantage of this approach is the independence of the 2^l sections which can be exploited to speed up the decoding process [6]. However, the number of cosets increases rapidly.

C. l-level Generating Pattern of Length 2

The l-level generating pattern of length 2 is defined as

$$P_{2(r,m)}(l) = [P_{20(r,m)}(l) \quad P_{21(r,m)}(l)] \tag{8}$$

where

$$P_{20(r,m)}(l) = \begin{bmatrix} G_{(r-l,m-l)} \\ C_{r-l+1,m-l)} \end{bmatrix} \tag{9}$$

$$P_{21(r,m)}(l) = \begin{bmatrix} 0 \\ C_{r-l+1,m-l)} \end{bmatrix} \tag{10}$$

for $l \leq r$. For $l > r$, we simply define $P_{20(r,m)}(l) = P_{21(r,m)}(l) = G_{(0,m-l)}$. The pattern $P_{2(r,m)}(l)$ represents a rate $R = K_{(r-l+1,m-l)}/2^{m-l}$ convolutional code whose encoder contains $K_{(r-l+1,m-l)} - K_{r-l,m-l)}$ shift registers, each connected to a different input. Due to the repetition of the matrix $C_{(r-l+1,m-l)}$ in $P_{2(r,m)}(l)$, the convolutional encoding can also be viewed as a differential encoding of $K_{(r-l+1,m-l)} - K_{(r-l,m-l)}$ bits of the information sequence. This fact also implies that any $K_{(r-l+1,m-l)} \times K_{(r-l+1,m-l)}$ determinant obtained from $P_{2(r,m)}(l)$ has the common factor $1+D$. Thus, based on Theorem 2 of [7], $P_{2(r,m)}(l)$ describes a catastrophic encoder. In general, catastrophic encoders must be avoided when convolutional codes are decoded with the VA. Also a minimal encoder is implemented as it defines the convolutional code C with minimum complexity over all polynomial encoders that generate C [8], [9]. However, a catastrophic encoder can be used in our case as the trellis terminates. This encoder is not minimal but it is known that

for a given complexity, a catastrophic encoder may generate a code with a larger row distance d_{row} than the free distance d_{free} of any code with a minimum encoder [10].[1] These facts are observed in all decompositions presented in this paper.

When using (8)–(10), we can rewrite (6) as

$$G_{(r,m)} = \begin{bmatrix} P_{20(r,m)}(1) & P_{21(r,m)}(1) \\ 0 & G_{(r-1,m-1)} \end{bmatrix}. \tag{11}$$

The corresponding encoder is represented in Fig. 1. The associated trellis diagram has two sections corresponding to two encoding stages. At the first stage, we assume the shift registers are empty and we encode $K_{(r,m-1)}$ input bits. Then, at the second stage, only $K_{(r-1,m-1)}$ bits enter the encoder at the inputs without shift registers. After this second stage, the shift registers are all cleared, which terminates the trellis. Since at both stages, the $K_{(r-1,m-1)}$ inputs without shift registers are independent of the memory contents, each transition between two trellis states consists of $2^{K_{(r-1,m-1)}}$ parallel branches. This trellis diagram is identical to the diagram represented in Fig. 3 of [4]. However, based on the embedded convolutional code, both the state and branch labelings of this figure are immediate.

Similarly, for the two-level squaring construction, after elementary row operations, (7) becomes

$$G_{(r,m)} = \begin{bmatrix} C_{(r,m-2)} & C_{(r,m-2)} & C_{(r,m-2)} & C_{(r,m-2)} \\ P_{20(r,m)}(2) & P_{21(r,m)}(2) & 0 & 0 \\ 0 & P_{20(r,m)}(2) & P_{21(r,m)}(2) & 0 \\ 0 & 0 & P_{20(r,m)}(2) & P_{21(r,m)}(2) \\ 0 & 0 & 0 & G_{(r-2,m-2)} \end{bmatrix}. \tag{12}$$

Equation (12) indicates that $RM(r,m)$ can be viewed as the union of $2^{K_{(r,m-2)}-K_{(r-1,m-2)}}$ cosets of a convolutional code whose trellis terminates after encoding every block of $K_{(r,m)}$ bits. Within each coset, the associated encoder is similar to the encoder of Fig. 1, when substituting the appropriate generator matrices obtained from (8). The associated trellis diagram is equivalent to Fig. 6 of [4]. At the transmitting end, the coset representative selected by the first $K_{(r,m-2)} - K_{(r-1,m-2)}$ uncoded bits is added to each of the four encoded subblocks delivered by the convolutional encoder. At the receiving end, the $2^{K_{(r,m-2)}-K_{(r-1,m-2)}}$ possible contributions of each coset representative must be removed from the received sequence.[2] Then, each modified sequence is decoded independently using the VA working on the terminated trellis associated with the embedded convolutional code. Again, this construction can be repeated iteratively. Generalizing (12) to the l-level squaring construction, we obtain for each coset a 2^l-section trellis diagram corresponding to the terminated convolutional code with l-level generating pattern $P_{2(r,m)}(l)$ for the first $2^l - 1$ sections. Since the trellis terminates after the (2^l)th section, only the parallel branches corresponding to $G_{(r-l,m-l)}$ remain for this last section.

[1] In general $d_{\text{free}} \leq d_{\text{row}}$, and $d_{\text{free}} = d_{\text{row}}$ if the encoder is not catastrophic.

[2] For BPSK signaling normalized to ± 1, this is easily realized by inverting the sign at every position where the coset representative bit is one.

Another representation of (12) is

$$G_{(r,m)} = \begin{bmatrix} C_{(r,m-2)} & C_{(r,m-2)} & C_{(r,m-2)} & C_{(r,m-2)} \\ 0 & C_{(r-1,m-2)} & C_{(r-1,m-2)} & 0 \\ P_{20(r,m)}(2) & P_{21(r,m)}(2) & 0 & 0 \\ 0 & G_{(r-2,m-2)} & 0 & 0 \\ 0 & 0 & P_{20(r,m)}(2) & P_{21(r,m)}(2) \\ 0 & 0 & 0 & G_{(r-2,m-2)} \end{bmatrix}. \tag{13}$$

Based on (13), the same encoding procedure is used, with the exception that all shift registers are also cleared at the second stage. Therefore, the $K_{(r-1,m-2)} - K_{(r-2,m-2)}$ corresponding inputs are set to zero at the second encoding stage and the corresponding input bits now determine a second level coset decomposition based on the decomposition $RM(r-1, m-2)/RM(r-2, m-2)$. With respect to the previous four-section trellis, the number of cosets increases to $2^{K_{(r,m-2)}-K_{(r-2,m-2)}}$, but the first two sections become independent of the two last ones. Generalizing (13) to the l-level squaring construction, cosets based on the decomposition $RM(r-l+1, m-l)/RM(r-l, m-l)$ can be added to the decomposition of [4] to obtain independent subsections. This fact can be exploited in the implementation of the decoder.

1) Example 1 — RM(1, 3) and First Order RM Codes: For the two-level squaring construction of the (8, 4, 4) RM code, we identify the coset selector

$$C_{(1,1)} = [0 \quad 1] \tag{14}$$

and the two generator matrices

$$P_{20(1,3)}(2) = P_{21(1,3)}(2) = [1 \quad 1]. \tag{15}$$

Also, since $G_{(-1,1)} = [0], P_{20(1,3)}(2) = G_{(0,1)}$ so that only the second stage with shift registers remains with respect to Fig. 1. The corresponding rate 1/2 encoder and trellis diagram are shown in Fig. 2. We can verify easily that the encoder of the embedded convolutional code is catastrophic. However, terminating the trellis breaks the catastrophic behavior. Note finally that for this code, no additional bits are encoded during the terminating stage, or equivalently, the trellis diagram contains no parallel branches.

Generalization of this example to the two-level squaring construction of any first order RM code is straightforward. For $RM(1,m)$, the coset selector $C_{(1,m-2)}$ is an $(m-2) \times 2^{m-2}$ matrix which defines 2^{m-2} different cosets. Both generator matrices equal $[1_{2^{m-2}}]$, where 1_n represents the all-1 n-tuple. The corresponding rate $1/2^{m-2}$ encoder contains a single shift register whose input and output are summed at the 2^{m-2} outputs of the encoder.

2) Example 2 — RM(2, 4) and Second Order RM Codes: For the two-level squaring construction of the (16, 11, 4) RM code, the matrices in (8)–(10) are the coset selector

$$C_{(2,2)} = [0 \quad 0 \quad 0 \quad 1] \tag{16}$$

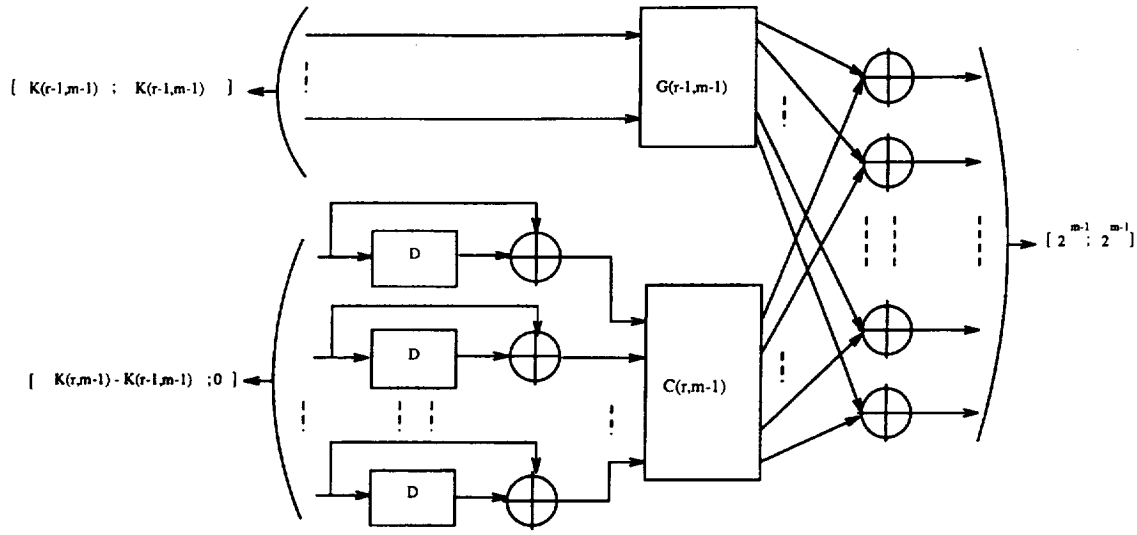


Fig. 1. Convolutional code encoder for one-level squaring construction.

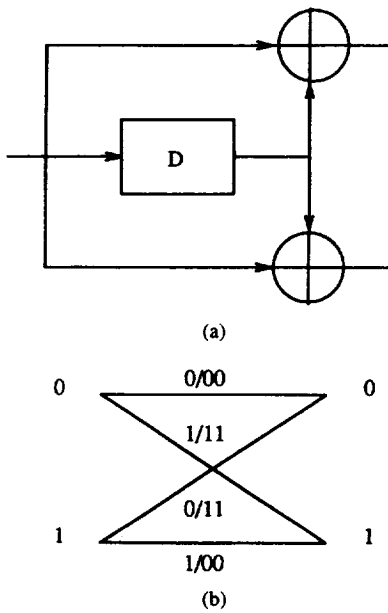


Fig. 2. (8, 4, 4) RM code. (a) Encoder and (b) trellis diagram.

and the two generator matrices

$$P_{20(2,4)}(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{17}$$

$$P_{21(2,4)}(2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}. \tag{18}$$

These matrices define a rate 3/4 convolutional code whose encoder and trellis diagram are represented in Fig. 3. Since the encoder contains only two shift registers, the trellis diagram has four states and each transition is represented by two parallel branches. As in the previous example, the convolutional code encoder is catastrophic and the catastrophic behavior is stopped at the fourth stage if the encoding is realized based on (12). However, at that last stage, $K_{(0,1)} = 1$ bit is encoded.

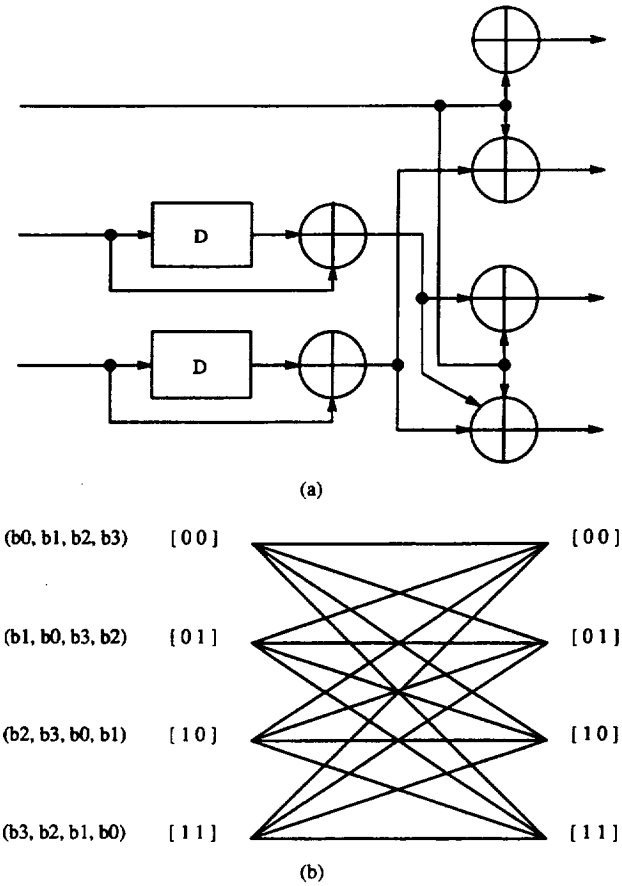


Fig. 3. (16, 11, 4) RM code. (a) Encoder and (b) trellis diagram with $b_0 = \{0000, 1111\}$, $b_1 = \{0101, 1010\}$, $b_2 = \{0011, 1100\}$, $b_3 = \{0110, 1001\}$.

The embedded convolutional code and its corresponding encoder for the two-level squaring construction of any second order RM code can be determined from this example. For $RM(2, m)$, the $\binom{m-2}{2} \times 2^{m-2}$ matrix $C_{(2,m-2)}$ determines

the choice among $2^{\binom{m-2}{2}}$ coset candidates based on the first $\binom{m-2}{2}$ bits of the block to the encoded. Then, the remaining bits are encoded in four stages by a rate $(m-1)/2^{m-2}$ convolutional encoder with $m-2$ inputs connected to a single shift register. At the first three stages, the input and output of each shift register are summed. This resulting sum is then added to the encoder outputs, based on the matrix $C_{(1,m-2)}$. The last input without shift registers is also added at all encoder outputs. At the fourth stage, only this input is delivered at all encoder outputs, while the shift registers are cleared, which terminates the trellis. Due to this input, each transition in the trellis is represented by a pair of parallel branches.

D. l-level Generating Pattern of Length 3

The l-level generating pattern of length 3 is defined as

$$P_{3(r,m)}(l) = [P_{30(r,m)}(l) \quad P_{31(r,m)}(l) \quad P_{32(r,m)}(l) \quad P_{33(r,m)}(l)] \tag{19}$$

where

$$P_{30(r,m)}(l) = \begin{bmatrix} G_{(r-l,m-l)} \\ C_{(r-l+1,m-l)} \\ C_{(r-l+2,m-l)} \end{bmatrix} = \begin{bmatrix} P_{20(r,m)}(l) \\ C_{(r-l+2,m-l)} \end{bmatrix} \tag{20}$$

$$P_{31(r,m)}(l) = \begin{bmatrix} 0 \\ C_{(r-l+1,m-l)} \\ C_{(r-l+2,m-l)} \end{bmatrix} = \begin{bmatrix} P_{21(r,m)}(l) \\ C_{(r-l+2,m-l)} \end{bmatrix} \tag{21}$$

$$P_{32(r,m)}(l) = P_{33(r,m)}(l) = \begin{bmatrix} 0 \\ 0 \\ C_{(r-l+2,m-l)} \end{bmatrix} \tag{22}$$

for $l \le r$. If $l > r$, the conventions introduced in Section II-A are applied to the definitions of these matrices. Equation (19) defines a rate $K_{(r-l+2,m-l)}/2^{m-l}$ convolutional code. The encoder contains $K_{(r-l+2,m-l)} - K_{(r-l+1,m-l)}$ inputs connected to shift registers of length three, $K_{(r-l+1m-l)} - K_{(r-l,m-l)}$ inputs connected to a single shift register and $K_{(r-l,m-l)}$ without memory elements. The complete trellis diagram is composed of $2^{3K_{(r-l+2,m-l)}-2K_{(r-l+1,m-1)}-K_{(r-l,m-l)}}$ states. However, in the following decompositions, the inputs connected to the shift registers of length three are not activated at each stage, so that the number of states in the trellis diagram is less than this maximum number. The convolutional code generated by $P_{3(r,m)}(l)$ also corresponds to a differential encoding and its encoder is catastrophic since $1+D$ is a factor of each determinant for Theorem 2 of [7].

Using these definitions, (12) can be rewritten as

$$G_{(r,m)} = \begin{bmatrix} P_{30(r,m)}(2) & P_{31(r,m)}(2) & P_{32(r,m)}(2) & P_{33(r,m)}(2) \\ 0 & P_{20(r,m)}(2) & P_{21(r,m)}(2) & 0 \\ 0 & 0 & P_{20(r,m)}(2) & P_{21(r,m)}(2) \\ 0 & 0 & 0 & G_{(r-2,m-2)} \end{bmatrix}. \tag{23}$$

From (23), we verify that the inputs connected to the shift registers of length three are activated only during the first stage, so that the number of states in the trellis diagram remains constant up to termination. For each stages, the complete trellis diagram consists of $2^{K_{(r,m-2)}-K_{(r-1,m-2)}}$ parallel subtrellises with the same branch labeling, due to the repetition of $C_{(r,m-2)}$ in $P_{3(r,m)}(2)$. Therefore, this trellis is equivalent to the union of the trellises corresponding to (12). The inputs to the shift registers of length three determine the coset selection associated with (12).

Based on the three-level squaring construction of $G_{(r,m)}$, the matrix $G_{(r,m)}$ can be put into the following form (see matrix at the bottom of this page). This matrix defines $2^{K_{(r,m-3)}-K_{(r-2,m-3)}}$ parallel subtrellises. The inputs to the shift registers of length three are activated at stages 1, 3, and 5. Since stages 1 and 5 are independent, the number of states in the trellis increases only at stage 3. Also, parallel subtrellises are observed in stages 2, 4, and 6. A more regular trellis diagram is obtained when considering the previous matrix as shown at the bottom of the next page.

The corresponding number of cosets increases to $2^{K_{(r,m-3)}+K_{(r-1,m-3)}-2K_{(r-2,m-3)}}$ but the trellis diagram becomes much more regular, as shown in Fig. 4. Each stage from 1 to 7, except stage 5, contains $2^{K_{(r-1,m-3)}-K_{(r-3,m-3)}}$ states, while only $2^{K_{(r-2,m-3)}-K_{(r-3,m-3)}}$ states are present at stage 5 for which all blocks of three serial registers are cleared. We finally notice that stages 2, 3, 6, and 7 are composed of $2^{K_{(r-1,m-3)}-K_{(r-2,m-3)}}$ parallel subtrellises with the same branch labeling.

1) Example 3 — $RM(3,5)$ and $RM(3,6)$: For the (32, 26, 4) RM code, the last decomposition provides four subtrellises of the form depicted in Fig. 4. All inner stages but stage 5 contain two parallel sub-subtrellises of four states each, where a sub-subtrellis is defined as a subtrellis within another larger

$$\begin{bmatrix} C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} \\ 0 & C_{(r-1,m-3)} & C_{(r-1,m-3)} & 0 & 0 & C_{(r-1,m-3)} & C_{(r-1,m-3)} & 0 \\ P_{30(r,m)}(3) & P_{31(r,m)}(3) & P_{32(r,m)}(3) & P_{33(r,m)}(3) & 0 & 0 & 0 & 0 \\ 0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{30(r,m)}(3) & P_{31(r,m)}(3) & P_{32(r,m)}(3) & P_{33(r,m)}(3) & 0 & 0 \\ 0 & 0 & 0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P_{30(r,m)}(3) & P_{31(r,m)}(3) & P_{32(r,m)}(3) & P_{33(r,m)}(3) \\ 0 & 0 & 0 & 0 & 0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & G_{(r-3,m-3)} \end{bmatrix}.$$

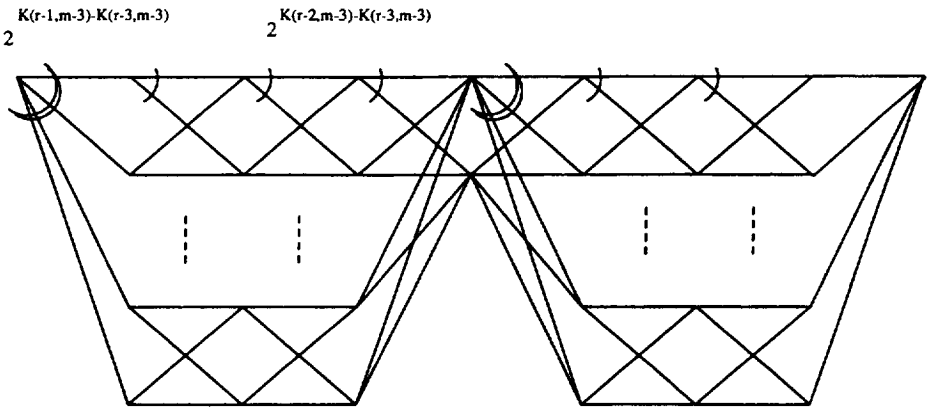


Fig. 4. Structure of each subtrellis associated with the convolutional code obtained from the second version of the generator matrix associated with the third order generating pattern of level three.

subtrellis. The same structure applies to the (64, 42, 8) RM code. The complete trellis contains 128 subtrellises formed of eight 8-state fully connected sub-subtrellises at all but the fifth inner stages.

E. l-level Generating Pattern of Length L

Based on the previous section, a generalization to the l-level generating pattern of length L for $L \geq 4$, is possible. Generalizing (19)–(22), we have

$$P_{L(r,m)}(l) = P_{L(r+i,m+i)}(l+i). \tag{24}$$

This results in the following theorem which shows that the embedded convolutional code based on the l-level generating pattern of length L depends only on the differences $r-l$ and $m-l$.

Theorem 2.1: The embedded convolutional code associated with the l-level generating pattern of length L applied to $RM(r,m)$ is the same as the embedded convolutional code associated with the $(l+i)$- level generating pattern of length L applied to $RM(r+i, m+i)$.

F. Family of Codes Corresponding to the l-level Generating Pattern of Length L

As given in (3), P defines an infinite family of $(n(x+L-1), kx)$ block codes obtained by terminating the encoding of the associated convolutional code after x blocks of k information bits. The decoding complexity for each code of this family is fixed. Similarly, an infinite family of block codes can be associated with the l-level generating pattern of length L. For example, based on (12), the two-level generating pattern of length 2 obtained from $RM(r,m)$ defines a family of codes with

$$\begin{aligned} N &= 2^{m-2}x \\ K &= (x-1)K(r-1,m-2) + \lfloor x/4 \rfloor \\ &\quad \cdot (K(r,m-2) - K(r-1,m-2)) \\ &\quad + K(r-2,m-2) \\ d_H &= 2^{m-r}. \end{aligned} \tag{25}$$

To maintain the same decoding complexity as in (12), the coset selector inputs are activated independently every four blocks. Generalizations to other generating patterns follow in a similar way.

For $RM(2,5)$ in the previous example, we identify the $(8x, 4(x-1)+3\lfloor x/4 \rfloor + 1, 8)$ codes. This family contains the (8, 1, 8), (16, 5, 8) and (32, 16, 8) RM codes. Also, for $x = 8$, we verify that the corresponding (64, 35, 8) code is a subcode of the (64, 42, 8). Based on [4], we evaluate that 6,704 real operations are required for trellis decoding of this subcode. With respect to the (32, 16, 8) RM code, the average computational complexity per bit increases from 150 to 192 real operations for a 0.32 dB coding gain at the bit-error rate

$$\begin{bmatrix}
C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} & C_{(r,m-3)} \\
0 & C_{(r-1,m-3)} & C_{(r-1,m-3)} & 0 & 0 & C_{(r-1,m-3)} & C_{(r-1,m-3)} & 0 \\
0 & 0 & C_{(r-1,m-3)} & C_{(r-1,m-3)} & C_{(r-1,m-3)} & C_{(r-1,m-3)} & 0 & 0 \\
P_{30(r,m)}(3) & P_{31(r,m)}(3) & P_{32(r,m)}(3) & P_{33(r,m)}(3) & 0 & 0 & 0 & 0 \\
0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & P_{30(r,m)}(3) & P_{31(r,m)}(3) & P_{32(r,m)}(3) & P_{33(r,m)}(3) \\
0 & 0 & 0 & 0 & 0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & P_{20(r,m)}(3) & P_{21(r,m)}(3) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & G_{(r-3,m-3)}
\end{bmatrix}.$$

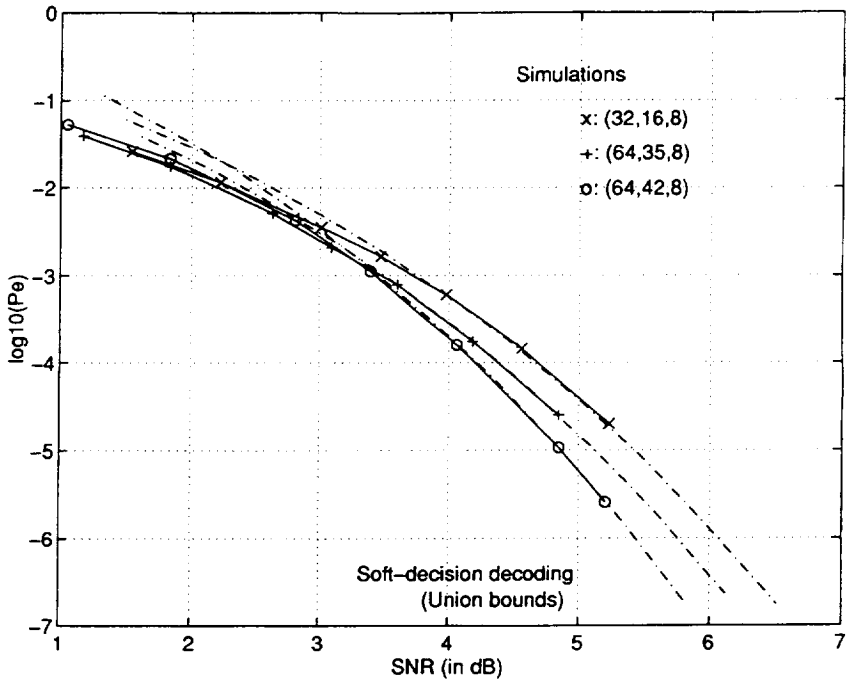


Fig. 5. Error performance comparison for the (32, 16, 8), (64, 35, 8), and (64, 42, 8) codes.

(BER) 10^{-6}, as shown in Fig. 5. At the same BER, 0.63 dB separates the bit error performance curves of the (32, 16, 8) and (64, 42, 8) RM codes as shown in Fig. 5. However, trellis decoding of the (64, 42, 8) RM code is realized with 12 968 real operations per bit [4]. By fixing the state complexity of the trellis, we define a subcode of the RM code of equivalent length. Trellis decoding of this subcode can provide a good trade-off between error performance and decoding complexity.

By considering the trellis diagram of the $(8x, 4(x-1)+3\lfloor x/4 \rfloor + 1, 8)$ codes, we compute the associated number of minimum weight codewords as

$$W_8(x) = 14x^2 - 13x + 448\lfloor x/4 \rfloor. \tag{26}$$

Therefore, $W_8(x)$ is proportional to x^2. For convolutional codes with a noncatastrophic encoder and row distance $d_{\text{row}}, W_{d_{\text{row}}}$ is proportional to x. As expected, since the codes considered have a catastrophic encoder, their number of minimum weight codewords increases faster than for convolutional codes with a noncatastrophic encoder. On the other hand, a larger row distance d_{row} can be achieved.

G. Extension to Other Related Constructed Codes

1) The (24, 12, 8) Golay Code and Related Codes Constructed from the Cubing Construction: The decomposition into cosets of a terminated convolutional code can be generalized to other families of block codes constructed from RM codes. We illustrate this fact with the (24, 12, 8) Golay code. Based on the cubing construction, the generator matrix $G_{(24,12)}$ of this code can be decomposed as [4]

$$G_{(24,12)} = \begin{bmatrix} C_{(2,3)} & C_{(2,3)} & C_{(2,3)} \\ P_{20(2,5)}(2) & P_{21(2,5)}(2) & 0 \\ 0 & P_{20(2,5)}(2) & P_{21(2,5)}(2) \\ 0 & 0 & G_{(0,3)} \end{bmatrix}. \tag{27}$$

Therefore, the encoder of the rate 4/8 convolutional code associated with $P_{20(2,5)}(2)$ and $P_{21(2,5)}(2)$ can be used to encode the (24, 12, 8) Golay code within any of its eight cosets. The VA working on the associated trellis diagram, terminated after the third section, can realize the decoding process and is described in [4]. Based on the rate 4/8 convolutional code encoder, the branch and state labelings of the trellis diagram are now immediate.

2) BCH Codes: Based on the construction summarized in Tables I and II of [11], the encoder and associated trellis diagram of certain BCH codes are determined as described in the previous sections.

III. Two-Stage Optimum Trellis Decoding of Block Codes

In Section II, we showed how to relate a particular coset decomposition to its corresponding embedded convolutional code for RM and related block codes. For many such decompositions, each transition in the trellis diagram associated with the corresponding embedded convolutional code consists of many parallel branches. In general, each transition corresponds to another shorter RM code of smaller dimension. Therefore, the trellis diagram of the corresponding code can be substituted for each transition of the trellis diagram of the embedded convolutional code. It results in a super-trellis diagram containing many more states. In this section, we propose to process this trellis diagram in two stages. At the first stage, all possible RM codes corresponding to parallel branches are processed. Then, each survivor and its metric are delivered to the corresponding transitions of the embedded convolutional code and the decoding is achieved based on the trellis diagram of this convolutional code only. The first stage of this two-stage decoding can be viewed as the branch

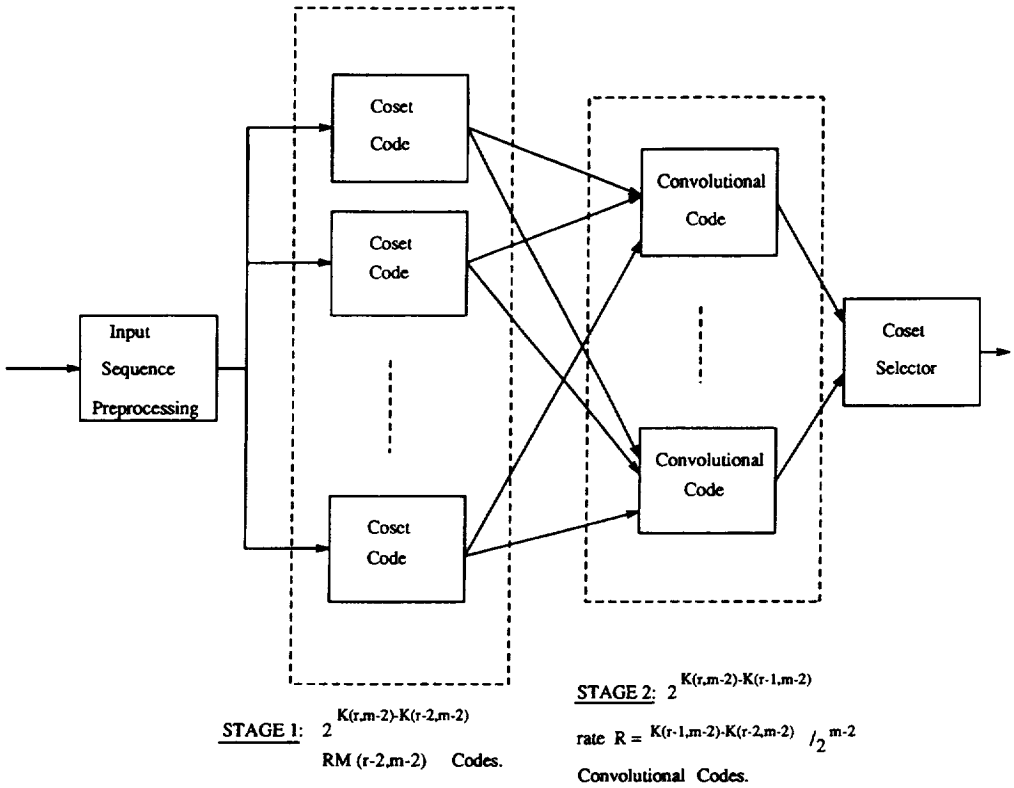


Fig. 6. Decoder structure for two-stages trellis decoding based on second order generating pattern of level two.

metric computation associated with the trellis decoding of the embedded convolutional code. Note that the decoding method presented in [4] is simply a special case of this two-stage decoding where the branch metric preprocessing corresponds to the first stage decoding of the trivial $RM(0, m-r)$ code.

We now present in detail the two-stage trellis decoding of the four-section decomposition obtained from (12). Generalization to any other decomposition follows the same lines. As described in Section II-C, a $2^{K_{(r-1,m-2)}-K_{(r-2,m-2)}}$-state rate $R = K_{(r-1,m-2)}/2^{m-2}$ convolutional code is associated with each coset of this representation. From (12), we observe that each parallel transition corresponds to a $RM(r-2, m-2)$ code. Based on this representation, we identify for each of the four sections $2^{K_{(r,m-2)}-K_{(r-2,m-2)}} RM(r-2, m-2)$ codes with different labeled trellises, or equivalently the same $RM(r-2, m-2)$ trellis working on $2^{K_{(r,m-2)}-K_{(r-2,m-2)}}$ different input sequences. Then, the second stage decoding is achieved based on the fully connected trellis diagram of the associated rate $R = K_{(r-1,m-2)} - K_{(r-2,m-2)}/2^{m-2}$ convolutional code for each coset. The corresponding implementation is shown in Fig. 6. The two-stage decoder represented in this figure is simply realized with a parallel implementation of respectively the same coset code and the same convolutional code. If the number of these component codes remains moderate, a low-cost decoder can be built.

1) Example 4 — $RM(3,5)$ and $RM(3,6)$: The first stage decoding of the (32, 26, 4) RM code requires the decoding of 16 (8, 4, 4) RM codes per section, each of them being achieved with 23 real operations when using conventional trellis decoding [4]. Thus stage 1 requires 1, 472 real operations. Then the trellis diagram of the rate $R = 3/8$ convolutional code is used by each of the two cosets at the second stage, resulting in 511 real operations, so that the complete two-stage trellis decoding is realized with 1, 983 real operations. Similarly, the first stage decoding of the (64, 42, 8) RM code decodes 1, 024 (16, 5, 8) RM codes per section, for a total of 389 120 real operations. Then the second stage decoding uses the trellis diagram of the fully connected $R = 6/16$ associated convolutional code in each of the 16 cosets of the decompositon. This second stage decoding is realized in 262 143 real operations. The total number of real operations for this codes becomes 651 263.

For comparison, the trellis decoding based on [4] uses an eight-section trellis decomposition and requires respectively 1, 271 real operations for the (32, 26, 4) RAM code and 544 640 operations for the (64, 42, 8) RM code, which is smaller in both cases. However, for these two codes, for a slightly larger computation complexity, a decoder of lower cost can be built by considering the two-stage decoding. Also, trellis decoding of first order RM codes can be efficiently improved without significant decoding delay as shown in [12] by considering metric differences only. This scheme is referred to as differential trellis decoding (DTD). With this method, trellis decoding requires 21 operations for the (8, 4, 4) code and 59 instead of 95 real operations for the (16, 5, 8) code. Similarly, DTD of the embedded convolutional codes can be processed with 335 and 164 863 real operations for the rates $R = 3/8$ and $R = 6/16$, respectively. The corresponding total number of operations for two-stage trellis decoding becomes 1, 679 for the (32, 26, 4) RM code and 406 527 for the (64,

42, 8) RM code. We finally mention that the computational complexity associated with trellis decoding of a given block code is generally higher than the decoding complexity of maximum likelihood decoding algorithms based on reliability information (see [13] for a recent bibliography of contributions with this approach). However, these algorithms generally required a partial or total ordering of the received sequence and do not offer as much parallel structure and regularity as trellis decoding, resulting in a larger decoding delay. Nevertheless, this additional delay may not be as important when considering only the code of smaller dimension decoded at the first stage. This suggests a mixed decoding method. For example, the best decoding algorithm of the (8, 4, 4) code requires only 17 real operations [14] and can be directly obtained from the trellis decoding method of [12], so that the (32, 26, 4) is now decoded in 1, 423 real operations with two-stage trellis decoding. These particular examples illustrate the difficulty of choosing the best decoder implementation for a particular code due to the trade-offs between decoding speed, computational complexity and implementation cost.

IV. Two-Way Trellis Decoding of Block Codes

A. Properties

Further speedup of trellis decoding of block codes can be obtained by exploiting the fact that for each block, the trellis terminates. We obtain the following theorem.

Theorem 4.1: Consider a trellis T from the original node O to the terminating node S, with $N(s)$ nodes at the end of each section s. Then, the minimum metric $M_{\min}$ in state S is equal to the minimum of the $N(s)$ sums $M^-(i)+M^+(i), i \in [1, N(s)]$, where $M^-(i)$ represents the minimum metric from state O to the ith node of the end of section s, and $M^+(i)$ represents the minimum metric from the ith node of the end of section s to state S.

Proof: By definition

$$M_{\min} = \min_{\overline{p} \in \pi} \{M(\overline{p})\} \tag{28}$$

where π represents the set of all paths from node O to node S in T, and $M(\overline{p})$ is the metric associated with path $\overline{p}$. If $\pi(i)$ denotes the set of paths $\overline{p} \in \pi$ containing state i, then, for section s

$$\bigcup_{i \in [1, N(s)]} \pi(i) = \pi \tag{29}$$

which implies

$$M_{\min} = \min_{i \in [1, N(s)]} \{\min_{\overline{p} \in \pi(i)} \{M(\overline{p})\}\}. \tag{30}$$

For each state $i \in [1, N(s)]$ of section s, we define $\pi(i)^-$ as the set of paths from state O to state i, and $\pi(i)^+$ as the set of paths from state i to S. Then

$$\forall \overline{p} \in \pi(i), \exists (\overline{p}^-, \overline{p}^+) \in \pi(i)^- \times \pi(i)^+ : \overline{p} = [\overline{p}^- | \overline{p}^+]. \tag{31}$$

Therefore, (30) can be rewritten as

$$M_{\min} = \min_{i \in [1, N(s)]} \{\min_{\overline{p}^- \in \pi(i)^-} \{M(\overline{p}^-)\} + \min_{\overline{p}^+ \in \pi(i)^+} \{M(\overline{p}^+)\}\} \tag{32}$$

which completes the proof. □

Theorem 4.1 implies that the trellis decoding of the received sequence $(y_1, y_2, \cdots, y_N)$ can be processed independently based on the trellis T^- from the origin node to section s for $(y_1, y_2, \cdots, y_s)$ and based on the mirror image T^+ of the trellis from section s to the final node S for the reverse sequence $(y_N, y_{N-1}, \cdots, y_{s+1})$. Then, at each node of section s, the two metrics are summed and the minimum of these sums determines the decoded sequence. With this approach, for $s > 2$, we no longer compute the metric candidates for the last two stages of the complete trellis, which correspond to the first stage of the mirror trellis, but we compute twice the metric candidates at each node of section s. In general, we choose $s = N/2$ and for faster decoding, the encoded sequence $(x_1, x_2, \cdots, x_N)$ can be interleaved to $(x_1, x_N, x_2, x_{N-1}, \cdots x_{N/2}, x_{N/2+1})$ before transmission. Also, for all but the merging section s, smaller cumulative metric values are carried out when decoding the second part of the received sequence.

B Application to Decoding of $RM(r, m)$ Based on the r-level Squaring Construction

In general, the trellises T^- and T^+ defined in Section IV-A are different. In this section, we show that the same trellis diagram can be used to decode $(y_1, y_2, \cdots, y_{N/2})$ and $(y_N, y_{N-1}, \cdots, y_{N/2+1})$ for the r-level squaring construction of $RM(r, m)$.

For any $A \times B$ matrix $M = [m_{i,j}], i \in [1, A], j \in [1, B]$, we define the operators $\alpha(\cdot)$ and $\beta(\cdot)$ as

$$\begin{aligned} \alpha(M) &= [m_{A-i+1,j}] \\ \beta(M) &= [m_{i,B-j+1}] \end{aligned} \tag{33}$$

and the binary matrix

$$M^* = M \oplus [1_{A,B}] \tag{34}$$

where $[1_{A,B}]$ represents the $A \times B$ all-1 matrix. If $C_{(1,m)}$ is expressed in its conventional Boolean form, we can easily show that

$$\beta(C_{(1,m)}) = C^*_{(1,m)}. \tag{35}$$

We consider the decomposition obtained when generalizing (12). Evaluating (8) for $r = l$ implies that $RM(r, m)$ can be generated either by the generating pattern

$$P_{(r,m)}(r) = \begin{bmatrix} 1_{2^{m-l}} & 0 \\ C_{(1,m-r)} & C_{(1,m-r)} \end{bmatrix} \tag{36}$$

or after adding $1_{2^{m-l}}$ to each row of $C_{(1,m-r)}$, by the generating pattern

$$P'_{(r,m)}(r) = \begin{bmatrix} 1_{2^{m-l}} & 0 \\ C^*_{(1,m-r)} & C_{(1,m-r)} \end{bmatrix}. \tag{37}$$

If $P'_{(r,m)}(r)$ is the chosen generating pattern, we observe that $(y_N, y_{N-1}, \cdots, y_{N/2+1})$ can be decoded within each coset from the trellis diagram associated with the generating pattern

$$\alpha\left(\beta\left(\begin{bmatrix} C^*_{(1,m-r)} & C_{(1,m-r)} \\ 0 & 1_{2^{m-l}} \end{bmatrix}\right)\right) = \begin{bmatrix} 1_{2^{m-l}} & 0 \\ \alpha(\beta(C_{(1,m-r)})) & \alpha(\beta(C^*_{(1,m-r)})) \end{bmatrix} = \begin{bmatrix} 1_{2^{m-l}} & 0 \\ \alpha(C^*_{(1,m-r)}) & \alpha(C_{(1,m-r)}) \end{bmatrix}. \quad (38)$$

From (37) and (38), we conclude that the same trellis diagram can be used when searching the closest paths to $(y_1, y_2, \cdots, y_{N/2})$ and $(y_N, y_{N-1}, \cdots, y_{N/2+1})$ with the VA. However, each subblock of the decoded sequence corresponding to $(y_N, y_{N-1}, \cdots, y_{N/2+1})$ must be permuted according to (38).

Another solution is to keep the generating pattern of (36), and therefore the general encoder structure depicted in Fig. 1 which is somehow easier to implement. The generating pattern for the sequence $(y_N, y_{N-1}, \cdots, y_{N/2+1})$ then becomes

$$\alpha\left(\beta\left(\begin{bmatrix} C_{(1,m-r)} & C_{(1,m-r)} \\ 0 & 1_{2^{m-l}} \end{bmatrix}\right)\right) \begin{bmatrix} 1_{2^{m-l}} & 0 \\ \alpha(\beta(C_{(1,m-r)})) & \alpha(\beta(C_{(1,m-r)})) \end{bmatrix}. \quad (39)$$

Equation (39) expresses that each subblock of length 2^{m-l} must be permuted before being processed by the VA. Therefore, the first subblock to be processed becomes $(y_{N-2^{m-l}+1}, y_{N-2^{m-l}+2}, \cdots, y_{N-1}, y_N)$. Note however that this permutation can equivalently be realized before transmission. Then, the same permutation on the decoded sequence as for (38) also applies to this version. Finally we mention that the use of the same trellis diagram for bidirectional decoding of $RM(r, m)$ also holds for any l-level squaring construction.

1) Example 5 — RM(2, 4): As an example, we consider bidirectional decoding of the (16, 11, 4) RM code based on the second version. The corresponding encoder and decoding trellis diagram are represented in Fig. 3. Let $(x_1, x_2, \cdots, x_{16})$ be the transmitted encoded BPSK sequence representing the information sequence $(c_0, c_1, \cdots, c_{10})$ and $(y_1, y_2, \cdots, y_{16})$ the corresponding noisy received sequence. Based on (8) and (39), within each of the two coset candidates, the VA processes independently the subsequences corresponding to $(y_1, y_2, \cdots, y_8)$ and $(y_{13}, y_{14}, y_{15}, y_{16}, y_9, y_{10}, y_{11}, y_{12})$ with the trellis diagram of Fig. 3. At each merging state, both metrics are added and the minimum determines the surviving path for each coset. Assuming c_0 determines the coset selection, the decoded sequence delivered by the VA corresponds to $(c_1, c_2, \cdots, c_5, c_6)$ for the forward sequence and $(c_{10}, c_8, c_9, c_7, c_5, c_6)$ for the backward sequence, as dictated in (39). We then permute the decoded sequence and decode c_0 by choosing the coset with minimum survivor metric. Next we compare the number of computations required for bidirectional and conventional trellis decoding of each coset of the (16, 11, 4) RM code. For both decoding cases, we first preprocess each of the values $\pm y_{4i+1} \pm y_{4i+2} \pm y_{4i+3} \pm y_{4i+4}$ for $i \in [0, 3]$, for a total of $4 \cdot 12 = 48$ additions [4]. At each trellis state, four additions to compute the cumulative metric candidates and three comparisons are processed by the VA. In conventional trellis decoding, sections two and three have four states and section four has 1 state, so a total of 63 real operations are required for this method. In bi-directional decoding, 56 real operations are processed since both second sections have four states. Then four additions and three comparisons are performed at the merging stage, so that the same number of 63 real operations is also required for this approach. From this example, we conclude that in general, neither significant computation gain, nor significant computation loss is achieved by bi-directional trellis decoding with respect to conventional trellis decoding. However, the decoding process reduces the decoding delay by a factor of two, as well as the metric memory requirements.

V. Conclusion

In this paper, we have shown that the trellis representation corresponding to any iterative squaring construction of RM codes as described in [4] can be viewed as the union of cosets of an embedded terminated convolutional code. For a given RM code and a given decomposition, different convolutional codes can be defined. We provided the generator polynomials of each of these convolutional codes by associating with each code a generating pattern. In each case, the same general structure of the embedded convolutional code is observed. The choice of the code requires a balance in the number of shift registers connected to each different input of the encoder with the number of cosets. A short generating pattern determines only a few shift registers connected to each input, which minimizes the code rate loss due to the termination of the trellis. However, since many rows of the generator matrix are not included in the generating pattern, the number of cosets is very large. The corresponding trellis diagrams are loosely connected with a high level of parallelism, but also a large total number of states whenever the number of cosets is important. The number of cosets can be reduced by choosing longer generating patterns. However, the encoder inputs with many memory elements have to be regularly set to zero to prevent a large expansion of the number of states in the trellis and maintain an efficient termination of the trellis. The complete trellis diagram contains few parallel subtrellises, but a high level of connectivity within subtrellises. In all considered cases, the convolutional code has a nonminimal catastrophic encoder and the catastrophic behavior is stopped by terminating the trellis. This guarantees a large row distance for the number of trellis states considered. Generalization to other codes related to RM codes has also been discussed. Based on the embedded convolutional code, the branch and state labeling of the trellis representations introduced in [4] becomes straightforward. Also, the general structure of this trellis is easily obtained from the generating pattern of the code.

We then showed that for a particular representation, the associated trellis diagram of each coset can be viewed as the trellis diagram of the associated convolutional code, with each

transition representing the same coset code C_1 of a smaller dimension. Therefore, this structure allows a two-stage trellis decoding where the first stage computes the branch metrics of the associated convolutional code trellis by trellis decoding of the code C_1. A wide choice of trellis decoding implementation is offered by this method.

Using the fact that for any block code the trellis terminates, we also showed that bidirectional decoding of each received sequence is possible. Further decoding speed-up is achievable with this approach when processing in parallel both ends of the received sequence, one forward and one backward. Also, this method allows one to carry smaller cumulative metrics during the decoding process at all but the merging states for the second part of trellis. For the r-level squaring construction of any $RM(r, m)$ code, the VA search for both subsequences can be realized with the same trellis diagram. However, the order of the decoded sequence must be modified for the backward search.

REFERENCES

[1] S. Lin and D. J. Costello Jr, *Error Control Coding Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[2] G. D. Forney Jr, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[3] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 76–80, Jan. 1978.

[4] G. D. Forney Jr, "Coset codes II: binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1152–1187, Sept. 1988.

[5] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland Mathematical Library, 1977.

[6] D. Rhee and S. Lin. "A low complexity and high performance concatenated scheme for high speed satellite communications," NASA Tech. Rep., Oct. 1993.

[7] J. L. Massey and M. K. Sain. "Inverse of linear sequential circuits," *IEEE Trans. Comput.*, vol. COMP-17, pp. 330–337, Apr. 1968.

[8] G. D. Forney Jr, "Convolutional Codes I: Algebraic structure," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 720–738, Nov. 1970.

[9] P. Piert, *Convolutional Codes, An Algebraic Approach*. Cambridge, MA: M.I.T. Press, 1988.

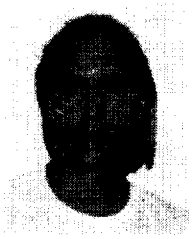
[10] D. J. Costello Jr., "Free distance bounds for convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 356–365, May 1974.

[11] A. Vardy and Y. Be'ery, "Maximum likelihood soft decoding decoding of BCH codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 546–554, Mar. 1994.

[12] M. P. C. Fossorier and S. Lin, "Differential trellis decoding of convolutional codes," submitted *IEEE Trans. Commun.*, Jan. 1995.

[13] ——, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inform. Theory*, vol. IT-41, pp. 1379–1396, Sept. 1995.

[14] M. Ran and J. Snyders, "On maximum likelihood decoding of some binary self-dual codes," *IEEE Trans. Commun.*, vol. 41, pp. 439–446, Mar. 1993.



Marc P. C. Fossorier (S'90–M'95) was born in Annemasse, France, on March 8, 1964. He received the B.E. degree from the National Institute of Applied Sciences (I.N.S.A.) Lyon, France, in 1987, and the M.S. and Ph.D. degrees from the University of Hawaii, Manoa in 1991 and 1994, all in electrical engineering.

He is currently with the Department of Electrical Engineering of the University of Hawaii, where he has been an Assistant Professor since January 1996, and worked as a Postdoctoral Fellow in 1995. His research interests include decoding techniques for linear codes, code constructions, communication algorithms, magnetic recording, and statistics.

Shu Lin (F'90), for a photograph and biography, see p. 42 of the January 1996 issue of this TRANSACTIONS.