The VLSI Design of Error-Trellis Syndrome Decoding for Convolutional Codes

I. S. Reed and J. M. Jensen
University of Southern California, Los Angeles
T. K. Truong and I.S. Hsu
Communications System Research Section

In this article, a recursive algorithm using the error-trellis decoding technique is developed to decode convolutional codes (CCs). An example, illustrating the VLSI architecture of such a decoder, is given for a dual-K CC. It is demonstrated that such a decoder can be realized readily on a single chip with NMOS technology.

I. Introduction

Recently, the authors (Refs. 1, 2) developed a new error-trellis syndrome decoding scheme for convolutional codes (CCs). This new method involves finding minimum-error paths in an error-trellis. It was shown (Ref. 1) that the computation of the error-trellis is accomplished by finding the solution of the syndrome equations explicitly in terms of the actual error sequence. This syndrome decoding scheme was then applied to a rate 3/4, one-error-correcting systematic Wyner-Ash code (Refs. 4, 5).

In this article, the error-trellis decoding is applied to decode a rate $1/n$, dual-K nonsystematic CC. The special example of a rate 1/2, dual-3 nonsystematic CC is treated in this article.

It is demonstrated in Ref. 6 and this article that the real advantage of error-trellis decoding over Viterbi decoding of CCs is the reduction of the number of states and transitions between any two frames. A recursive algorithm for finding the path of minimum error in the error-trellis is found which realizes a rate 1/2, dual-3 nonsystematic CC. This recursive algorithm eliminates all paths in certain fixed frames except the path with minimum error. A VLSI chip architecture is developed to realize this new recursive algorithm for decoding the dual-K CC. The designs developed for this decoder are regular, simple, expandable and, therefore, naturally suitable for VLSI implementation.

II. The Properties of Convolutional Codes

In this section a brief review is presented of properties of CCs needed in the following sections.

The input $\underline{X}$ to a k-dimensional CC encoder is represented as an infinite sequence of vectors, X_j, of form,

$$\underline{X} = [X_0, X_1, X_2, \ldots] \tag{1}$$

where $X_j = [x_{1j}, x_{2j}, \ldots, x_{kj}]$ is a k-component vector of elements from the Galois field $GF(q)$, where $q = p^n$ with p a prime integer. Each vector X_j of k symbols for ($j = 0, 1, 2, \ldots$) is sometimes called an information or input frame (see Ref. 4, Sec. 12.1). Similarly, the output $\underline{Y}$ of a CC of length n is an infinite sequence of vectors, Y_j, of form

$$\underline{Y} = [Y_0, Y_1, Y_2, \ldots] \tag{2}$$

where $Y_j = [y_{1j}, y_{2j}, \ldots, y_{nj}]$ is an n-component vector of elements from $GF(q)$. Here vector Y_j is called a codeword frame or more simply, a code frame (Ref. 4, Sec. 12.1).

In a CC encoder the input $\underline{X}$ in Eq. (1) and output in Eq. (2) are linearly related; hence there exists what is called an infinite generator matrix $\underset{\sim}{G}$ such that

$$\underline{Y} = \underline{X} \cdot \underset{\sim}{G} \tag{3}$$

For the CC to have finite memory, G has the form

$$\underset{\sim}{G} = \begin{bmatrix} G_0 & G_1 & G_2 & \ldots & G_m & & \\ & G_0 & G_1 & G_2 & \ldots & G_m & \\ & & & \vdots & & & \\ & & & G_0 & G_1 & G_2 & \ldots & G_m \end{bmatrix} \tag{4a}$$

where the submatrices G_j are $k \times n$ matrices of form

$$G_j = [G_{rsj}] \tag{4b}$$

and the elements G_{rsj} belong to $GF(q)$ for $1 \leqslant r \leqslant k$, $1 \leqslant s \leqslant n$ and $0 \leqslant j \leqslant m$. Multiplying the subvectors of $\underline{x}$ in Eq. (1) by the matrix G in Eq. (4a) yields, by Eq. (3), the fundamental identity,

$$Y_j = \sum_{i=0}^{\min(j,m)} X_{j-i} \cdot G_i \tag{5}$$

which is the convolution of sequence $\{X_0, X_1, \ldots\}$ of information frames with the sequence $\{G_0, G_1, \ldots, G_m\}$ of matrix operators. The integer m in Eq. (4a) is the memory of the convolution Eq. (5). The value of m is the maximum number of past input frames X_j needed to compute Eq. (5), recursively.

The convolution property (Eq. [5]) of finitely generated CCs can be realized conveniently for some applications by the operational calculus over a finite field $GF(q)$. To accomplish this one defines first the generating functions or, what are sometimes called, the D-transform of the sequences $\{X_j\}$, and the $\{G_j\}$ and $\{Y_j\}$ matrices, as follows:

$$X(D) = \sum_{j=0}^{\infty} X_j D^j \tag{6a}$$

$$g(D) = \sum_{j=0}^{m} G_j D^j \tag{6b}$$

and

$$Y(D) = \sum_{j=0}^{\infty} Y_j D^j \tag{6c}$$

where D is an operator variable. It is not difficult to verify that by equating the coefficients in the matrix relationship

$$Y(D) = X(D) \cdot G(D) \tag{7}$$

the fundamental convolution property (Eq. [5]) of a convolutional code of memory m is derived. Hence identity (Eq. [7]) is precisely equivalent to the defining relationship (Eq. [3]) of a convolutional code. Finally if D is identified with a unit delay circuit element, it is not difficult to show that $G(D)$ maps directly onto an encoder circuit diagram.

By Eq. (5) the jth output of an n-vector or codeword frame, Y_j is dependent on at most the $m + 1$ present and past input k-vectors or information frames. Hence it is natural (as suggested by Blahut [Ref. 4, Section 12.1]) to define

$$k_1 = (m + 1)k \tag{8a}$$

to be the word length of the CC. Then the word length k, is extended by the encoding process of Eq. (5) to, what is called, the block length n_1 of the CC. The block length of CC is

$$n_1 = (m + 1) \cdot n = \frac{k_1}{R} \tag{8b}$$

where $R = k/n$ is the rate of the code. By Eq. (5) the block length $n_1 = (m + 1)n$ is the length of the subsequence of $\underline{Y}$ which, during encoding, can be influenced by a single information frame.

The minimum distance of the code is interpreted to be the Hamming weight of the smallest weight code word segment of $\ell = m + 1$ which is nonzero in the first frame. Suppose for some CC that at most t errors occur during transmission in the first ℓ code word frames, and that

$$2t + 1 \leqslant d$$

is satisfied by the code. Then those errors which occur in the first block length of CC can be corrected using feedback decoding. Such a CC is called a t-error-per-block-length-correcting CC or more simply a t-error-correcting CC (Ref. 4, Sec. 12.3).

Another distance between code words of a CC which is commonly used is the free distance d_{free}:

$$d_{\text{free}} = \min_{X(D)\neq 0} W_H(X(D)\cdot G(D))$$

Since clearly, $d \leqslant d_{\text{free}}$ (Ref. 4), designing a CC with minimum distance d guarantees that the code has a free distance of d or greater.

To find the minimum distance d of a CC, either the following $k_1 \times n_1$ submatrix is used

$$G = \begin{bmatrix} G_0 & G_1 & G_2 & \cdots & G_m & & \\ & G_0 & G_1 & \cdots & G_{m-1} & & \\ & & \ddots & & & & \\ & & & \ddots & G_0 & G_1 & \\ & & & & & G_0 & \end{bmatrix} \tag{9}$$

where $k_1 = (m+1)k$ and $n_1 = (m+1)n$, or its corresponding parity check matrix H. The techniques used to find the minimum distance for block codes apply also for finding d using matrix G in Eq. (9) or the associated parity-check matrix H. Sometimes (see Ref. 4, Sec. 3.3) matrix G in Eq. (9) is called the basic generator matrix of the CC.

III. Error-Trellis Decoding

In this section error-trellis decoding as developed in Refs. 3 and 6 is reviewed briefly. First in order to avoid catastrophic error propagation the D-transform $G(D)$ in Eq. (6b) is restricted to have the Smith normal form

$$G(D) = A(D)\,[I_k, 0]\,B(D) \tag{10}$$

where $A = A(D)$ and $B = B(D)$ are, respectively, $k \times k$ and $n \times n$ invertible matrices with elements in $F[D]$, the ring of polynomials in D over $GF(g)$. The elements of the inverses A^{-1} and B^{-1} of matrices A and B, respectively, are also in $F[D]$ or are polynomials in D. For descriptive brevity the D-transform $G(D)$ is called the generator matrix.

If $B = B(D)$ in Eq. (10), let

$$B = [B_1, B_2]^T \text{ and } B^{-1} = [\overline{B_1}, \overline{B_2}] \tag{11}$$

where the first k rows of B constitute submatrix B_1 and the remaining $n-k$ rows are B_2. Similarly the first k columns of B^{-1} constitute submatrix $\overline{B_1}$ and the other $n-k$ columns are $\overline{B_2}$. It was shown (Refs. 1, 2, 6) that

$$G \cdot H^T(D) = G \cdot \overline{B_2} = 0 \tag{12}$$

where $H(D)$ is a parity-check matrix for $G(D)$.

Let $Z(D) = [Z_1(D), \ldots, Z_n(D)]$ be the vector D-transform of received sequence $\underline{Z}$. Then the D-transforms of the transmitted and received sequences are related by

$$Z(D) = Y(D) + e(D) \tag{13}$$

where $e(D) = [e_1(D), \ldots, e_n(D)]$ is D-transform of error sequence, henceforth called, simply, the error sequence.

From Eq. (12) the syndrome of the received code word is

$$S = Z \cdot H^T = e \cdot \overline{B_2} \tag{14}$$

This is a nonhomogeneous system of linear equations for the unknown error sequence $e(D)$. The problem of syndrome decoding CCs is to solve this system of equations for $e(D)$. It was shown (Refs. 1, 6) that the general solution of Eq. (14) is given by

$$e = uG + ZR \tag{15a}$$

where

$$R = \overline{B_2} \cdot B_2 \tag{15b}$$

with B_2 and $\overline{B_2}$ defined in Eq. (11).

Let e_a replace e in Eq. (13) as the actual error sequence; then a substitution of Eq. (13) into Eq. (15a) gives by Eq. (12)

$$e = uG + ZR = uG + (Y + e_a)\,\overline{B_2}\,B_2 = uG + e_a R$$

which is independent of the transmitted codeword Y.

Using the solution (Eq. (15a)) of the syndrome equation an estimate of the error sequence can be obtained from

$$\|\hat{e}\| = \min_u \|uG + ZR\| \tag{16}$$

where $\|\ \|$ denotes Hamming distance norm or weight and the minimization is taken over all k-vectors u over $F[D]$. By Eq. (16) the minimum weight error sequence is

$$\hat{e} = \hat{u}G + ZR = \hat{u}G + e_a R \tag{17}$$

where $\hat{u}$ is the k-vector with elements in $F[D]$ for which the minimum weight in Eq. (16) is obtained. It was shown (Ref. 1)

that $\hat{u}$ in Eq. (17) is a correction factor such that the original message is estimated by

$$\hat{X} = Z \cdot G^{-1} - \hat{u} \tag{18}$$

Substituting $Z = Y + e_a$ into Eq. (17) yields

$$\hat{X} = (Y + e_a) G^{-1} - \hat{u} = X + e_a G^{-1} - \hat{u} \tag{19}$$

Let E be the set of all error sequences which can be decoded correctly. Then, if $e_a \in E$, the most likely error sequence found by the minimization in Eq. (16) is equal to e_a, and therefore, by Eq. (19), $u = e_a G^{-1}$. Thus, the minimization in Eq. (16) has only to be taken over those sequences u which belongs to $E^{(-1)} = \{\hat{u} = e G^{-1} : e \in E\}$. Hence

$$\|\hat{e}\| = \min_{u \in E^{(-1)}} \|uG + ZR\| \tag{20}$$

Note that if $e_a \in E$, the most likely error sequence found by either Eq. (16) or (20) is identical and equal to e_a.

In order to actually perform the minimization in Eq. (20) over $E^{(-1)}$, the sets E and $E^{(-1)}$ must be identical. This is generally impossible. However for systematic CCs, it was shown (Ref. 6) that $E^{(-1)}$ can be approximated by the set

$$E_1^{(-1)} = \{u : W_H(u_j, \ldots, u_{j+m}) \leqslant t, \text{ for all } j \geqslant 0\} \tag{21}$$

where $t = [(d_{\text{free}} - 1)/2]$ and m is the length of memory. For a more detailed discussion of the relation between $E^{(-1)}$ and $E_1^{(1)}$, see Ref. 6. Thus, for systematic CCs, Eq. (20) becomes

$$\|\hat{e}\| = \min_{u \in E_1^{(-1)}} \|uG + ZR\| \tag{22}$$

In order to take the minimization in Eq. (22) over $E_1^{(-1)}$, a specific procedure was found to "prune" the error-trellis (Ref. 6). Also it was shown (Ref. 6) that the number of states S and transitions T needed for error-trellis decoding of an arbitrary systematic CC is

$$S = \sum_{i=0}^{\min\{t, mk\}} \binom{mk}{i} (q-1)^i \tag{23}$$

and

$$T = \sum_{i=0}^{\min\{t, (m+1)k\}} \binom{(m+1)k}{i} (q-1)^i \tag{24}$$

respectively. Note that the standard Viterbi decoding (Ref. 3, Sec. 4.17) requires q^{mk} states and $q^{(m+1)k}$ transitions within a frame time.

In the next section error-trellis decoding is developed for an important class of nonsystematic CCs, called dual-K CCs. The dual-K convolutional codes were invented and developed by Viterbi and Odenwalder. These CCs are nonbinary codes over the field $GF(2^K)$ and are used in practice in channels which experience fading such as UHF tropospheric scatter channels, etc.

IV. Error-Trellis Decoding of Dual-K CCs

Dual-K $(n, 1)$ convolutional codes are of rate $1/n$, of memory $m = 1$, and with symbols in the finite or Galois field $GF(2^K)$ (see Ref. 7). The generating matrix G is a special case of Eq. (4a), namely,

$$G^{(1)} = \begin{bmatrix} G_0 & G_1 & & \\ & G_0 & G_1 & \\ & & G_0 & G_1 \\ & & & \ddots \end{bmatrix} \tag{25a}$$

where

$$G_0 = [1, 1, 1, \ldots, 1]$$
$$G_1 = [g_{11}, g_{12}, \ldots, g_{1n}] \tag{25b}$$

with $g_{1j} \neq 0$ and $g_{1j} \in GF(2^K)$ and the g_{1j}'s are all distinct, for $1 \leqslant j \leqslant n$.

From the above definition of a dual-K CC, it is readily verified that the minimum distance of the code is $d = (2n - 1)$ and the free distance is

$$d_{\text{free}} = 2n \tag{25c}$$

Hence if no more than t symbol errors occur in the first 2 code word frames and $2t + 1 \leqslant d = 2n - 1$ or $t \leqslant n - 1$, then those errors which occur in the first frame can be corrected. In other words, the dual-K CC is a t-error-per-block-length-correcting CC where

$$t = \left[\frac{(d_{\text{free}} - 1)}{2}\right] = n - 1$$

and $[x]$ denotes the greatest integer less than x.

If error-trellis decoding is applied to the dual-K CCs, then from Eq. (20) the most likely error sequence $\hat{e}$ is found as

$$\|\hat{e}\| = \min_{u \in E^{(-1)}} \|uG + ZR\| \tag{25d}$$

In Appendix A it is shown that $E^{(-1)}$ can be approximated by the set

$$\tilde{E}^{(-1)} = \{u : W_H(u_j, u_{j+1}) \leqslant t, \text{ for all } j \geqslant 0\} \tag{25e}$$

Thus error-trellis decoding of dual-K CCs is performed by taking the minimum in Eq. (25d) over the set $\tilde{E}^{(-1)}$ in Eq. (25e). But by Eq. (21) this set is equal to $E_1^{(-1)}$ which is used in Eq. (6) also for systematic CCs. Therefore, the trellis can be "pruned" using the procedure in Eq. (6), and also the number of states and transitions within a frame time is as given in Eqs. (23) and (24). Consider the example by Odenwalder (Ref. 7, Fig. 1).

Example 1. Let the Galois field $GF(2^3)$ be generated by the polynomial $x^3 + x^2 + 1$, irreducible over $GF(2^3)$. If α is a root of this polynomial, then

$$\alpha, \alpha^2, \alpha^3, \alpha^4 = 1 + \alpha + \alpha^2, \alpha^5 = 1 + \alpha, \alpha^6$$

$$= 1 + \alpha, \alpha^7 = 1, \text{and } 0$$

are the eight elements of $GF(2^3)$. The generating matrix of type Eq. (6b) for a rate 1/2, dual-3 CC is

$$G = [1 + D, 1 + \alpha D] \tag{26a}$$

The output of encoder in terms of input is

$$Y = [Y_1, Y_2] = X[1 + D, 1 + \alpha D]. \tag{26b}$$

If one applies elementary column operations to G in Eq. (25b), it is not difficult to show that

$$G = [1, 0] \begin{bmatrix} 1 + D, & 1 + \alpha D \\ 1, & \alpha \end{bmatrix}$$

is the Smith normal from Eq. (10). Hence

$$B = \begin{bmatrix} 1 + D, & 1 + \alpha D \\ 1, & \alpha \end{bmatrix} \tag{27a}$$

$$B^{-1} = \begin{bmatrix} \alpha^3, \alpha^2 + \alpha^3 D \\ \alpha^2, \alpha^2 + \alpha^2 D \end{bmatrix} \tag{27b}$$

are the matrices needed in Eqs. (11a) and (15b).

$$R = \overline{B_2} B_2 = \begin{bmatrix} \alpha^2 + \alpha^3 D \\ \alpha^2 + \alpha^2 D \end{bmatrix} [1, \alpha]$$

$$= \begin{bmatrix} \alpha^2 + \alpha^3 D, \alpha^3 + \alpha^4 D \\ \alpha^2 + \alpha^2 D, \alpha^3 + \alpha^3 D \end{bmatrix} \tag{28}$$

is the matrix R needed in the error-trellis solution, Eq. (15b), of the syndrome Eq. (14). From Eq. (25c), $d_{\text{free}} = 4$ and hence, the present dual-3 code will correct at most 1 symbol per block length of 2. From Eq. (24), the number of transitions in one frame time needed in the error-trellis is

$$T = \sum_{i=0}^{1} \binom{(m+1)k}{1} (q-1)^i = \sum_{i=0}^{1} \binom{2}{1} \cdot (8-1)^i = 15$$

For the standard decoding trellis $q^{(m+1)k} = 8^2 = 64$ transitions are required. This yields a fractional reduction of $15/64 \cong 1/4$ in the number of transitions needed for error-trellis decoding between that required for standard Viterbi hard decoding. Also from Eq. (23), the number of states is

$$S = \sum_{i=0}^{1} \binom{mk}{i} (q-1)^i = \sum_{i=0}^{1} \binom{1}{i} (8-1)^i = 8$$

which is equal to the number of states of Viterbi decoding, i.e., $q^{mk} = 8$.

The "pruned" error-trellis is shown in Fig. 1. And the construction of the trellis is described in the following. The labels on the pruned error-trellis shown in Fig. 1 correspond to the solution, Eq. (15a), of the syndrome equation for the actual error equal to the all-zero sequence. That is,

$$e = [e_1, e_2] = uG = u\{1 + D, 1 + \alpha D\}$$

$$= \{u + Du, u + \alpha Du\} \tag{30}$$

is the output of the trellis. For example, at frame time j and state 0 if $u = \alpha^4$, then $e = [\alpha^4 + 0, \alpha^4 + \alpha \cdot 0] = [\alpha^4, \alpha^4]$ is

the label on transition from state 0 to state α^4. Such a transition represents an attempt to "cancel" a simple error in the error-trellis equation, Eq. (15a). If such an error does, in fact, occur at frame j, then no further errors are allowed to occur at frame $j+1$. Thus a transition to other than state zero must be followed by a transition back to state 0 in the next frame as shown in Fig. 1.

Next suppose a transition to state α^4 occurs, i.e., $Du = \alpha^4$. Then since $u = 0$, the transition from state $Du = \alpha^4$ back to 0 is given, using Eq. (30) by $e = [0 + \alpha^4, 0 + \alpha \cdot \alpha^4] = [\alpha^4, \alpha^5]$. The remaining labels to the "pruned" error-trellis are obtained in a similar manner.

To illustrate error-trellis decoding of the dual-3 CC let the generating sequence be

$$X(D) = 1 + \alpha D \tag{31}$$

Then the code word sequence is by Eqs. (7) and (26b)

$$Y(D) = X(D)\,G(D) = [1 + \alpha^5 D + \alpha D^2, 1 + \alpha^2 D^2]$$

Next, let the actual error sequence be $e_a(D) = [D^2, \alpha]$ so that

$$Z(D) = Y(D) + e_a(D) = [1 + \alpha^5 D + \alpha^5 D^2, \alpha^5 + \alpha^2 D^2] \tag{32}$$

Hence by Eq. (28),

$$ZR = [1 + \alpha^5 D + \alpha^5 D^2, \alpha^5 D^2] \begin{bmatrix} \alpha^2 + \alpha^3 D, \alpha^3 + \alpha^4 D \\ \alpha^2 + \alpha^2 D, \alpha^3 + \alpha^3 D \end{bmatrix}$$

$$= [\alpha^3, \alpha^4] + [\alpha^3, \alpha^4]\,D + [\alpha^2, \alpha^3]\,D + [\alpha^3, \alpha^4]\,D^3 \tag{33}$$

The finding of the minimum weight error-path $\hat{e}(D)$ in terms of $u(D)$ as given by Eq. (17) is equivalent by Eq. (15a) to finding the code word $u(D)\,G(D)$ which is closest to $Z(D) \cdot R(D)$ as given in Eq. (33). Hence the minimum-weight error-path can be found by applying the Viterbi decoding algorithm (Ref. 3) to the pruned error-trellis in Fig. 1. To accomplish this, the frames of ZR in Eq. (33) are added to the output uG in the pruned error-trellis in Fig. 1 as shown in Fig. 2.

In order to illustrate the Viterbi algorithm as applied to the pruned error-trellis suppose the decoder has reached frame 4. The output of the transition from state α^3 to 0 is

$$\text{Coef}_{D^3}[u(D) \cdot G(D) + Z(D)\,R(D)] = [\alpha^3, \alpha^4] + [\alpha^3, \alpha^4] = [0, 0]$$

with Hamming weight 0. A similar calculation for the other 7 possible transitions shows that the transition from α^3 to 0 is the only one with Hamming weight 0. The path segment from α^3 to 0 is chosen since it has minimum weight.

At frame 5, Fig. 2, the minimum weight estimate of the D-transform of the error sequence is $\hat{e}(D) = [0, \alpha] + [1, 0]\,D^2$. Hence the estimate $u(D)$ of the message correction factor which achieves $e(D)$ is

$$\hat{u} = \alpha^3 + \alpha^3 D^2 \tag{34}$$

Finally, using Eqs. (27), (32), (34) in Eq. (18) yields, by Table 1,

$$\hat{X} = ZG^{-1} - \hat{u}$$

$$= Z \begin{bmatrix} \alpha^3, \alpha^2 + \alpha^3 D \\ \alpha^2, \alpha^2 + \alpha^2 D \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \hat{u}$$

$$= [1 + \alpha^5 D + D^2, \alpha^5 + \alpha^2 D^2] \begin{bmatrix} \alpha^3 \\ \alpha^2 \end{bmatrix}$$

$$- [\alpha^3 + \alpha^3 D^2] = 1 + \alpha D$$

the original encoded message in Eq. (31).

V. Recursive Algorithm for Error-Trellis Syndrome Decoding of Convolutional Codes

For the dual-3 CC described in Sec. IV, error-trellis decoding of CC is used to correct one error in every ℓ frames. For this example a recursive algorithm is developed to determine the path with minimum Hamming weight for every ℓ frames. This recursive algorithm for the error-trellis syndrome decoding of dual-3, rate 1/2, one-error-correcting nonsystematic convolutional codes is described with a flowchart as shown in Fig. 3.

The recursive algorithm is illustrated in example 1 of the last section. In this example, $u = 1$ and $\ell = 2$. By Eq. (33)

$$ZR = ([\alpha^3, \alpha^4], [\alpha^3, \alpha^4], [\alpha^2, \alpha^3], [\alpha^3, \alpha^4], [0, 0]) \tag{35}$$

In Eq. (35), the first value of ZR is not equal to zero. Thus it is assumed that one error occurs in the first two frames. To find this error, the pruned error-trellis with a no-error output $u(D)\ G(D)$ in $\ell = 2$ frames is computed. This partial trellis is shown in Fig. 4. An error-trellis over these two frames is created by adding the vectors $[\alpha^3, \alpha^4]$ and $[\alpha^3, \alpha^4]$, successively, to all labels, in the pruned trellis in Fig. 4. This resulting error-trellis for these two frames is shown in Fig. 5.

In Fig. 5 one needs only to find the path with minimum weight which ends up in state $\sigma = 0$ at the end of $\ell = 2$ frames. Using Viterbi decoding the path with minimum weight is $([\alpha, 0], [0, 0])$.

Next, the input value following $[\alpha^3, \alpha^4]$ and $[\alpha^3, \alpha^4]$ again is not equal to zero. Again it is assumed that only one error occurs in the next two consecutive values of $[\alpha^2, \alpha^3]$ and $[\alpha^3, \alpha^4]$. These values are added again to the pruned trellis of two frames given in Fig. 4 for generating an error-trellis. Again using Viterbi decoding one finds the path with minimum weight to be $([1, 0], [0, 0])$. Finally, the input value following $[\alpha^2, \alpha^3]$ and $[\alpha^3, \alpha^4]$ is $[0, 0]$. Hence the estimated error for these two frame times is $[0, 0]$. Thus the overall path with minimum weight is

$$\hat{e} = ([0, \alpha], [0, 0], [1, 0], [0, 0], [0, 0])$$

As a consequence, the correction factor $\hat{u}(D)$ is $\hat{u}(D) = \alpha^3 + \alpha^2 D^2$. Hence, from Eq. (18), the estimated message is $\hat{X} = 1 + \alpha D$. The same procedure applies similarly to a systematic one-error-correcting Wyner-Ash CC presented in Ref. 5. In this code, $\ell = m + 1 = 2 + 1 = 3$.

VI. A VLSI Design for Error-Trellis Syndrome Decoding of Convolutional Codes

In this section, a VLSI architecture is developed for the recursive algorithm for error-trellis decoding of convolutional codes presented in Fig. 3. This VLSI processor for selecting the path with minimum Hamming weight is composed first of d basic cells, where d is the number of paths in the error-trellis. These d cells are followed then by a weight comparison circuit. A basic cell computes the path weight incrementally. That is, if symbol A is not equal to symbol B, then the weight of that particular path increases by one; otherwise, the weight remains unchanged. The VLSI architecture of the error-trellis syndrome decoder is illustrated in the following example.

The calculations used in the present example were given in the last two sections. The VLSI architecture for this convolutional code is illustrated in Fig. 6. In this figure there are 8 basic cells corresponding to the eight possible paths in the error-trellis. The function of each basic cell is described as follows.

The ith basic cell corresponds to the ith path in 2 frame times in the error-trellis. Thus if A is not equal to B, then the weight of the ith path increases by one (otherwise it remains the same) where A is the input value and B is the precalculated value stored in the ith basic cell. In this example, A is $(ZR)_k$, for $(k = 1, 2, 3, 4)$, and B is $(u \cdot G)_j$, for $(j = 1, 2, 3, 4)$, where Z is the received code sequence, R is defined in Eq. (28), u is an arbitrary k-vector of elements in $F[D]$, and G is the generator matrix.

First the received code sequence Z is multiplied by matrix R and G^{-1}, as shown in Fig. 7, to form both the input sequence to the basic cells and the inverse of the received message, i.e., $Z \cdot G^{-1}$. The input sequence ZR is then sent to all the 8 basic cells as well as a zero detector simultaneously. The inverse of the received message ZG^{-1} is sent then to a delay line to wait for the completion of the set of operations needed to estimate the correction factor $\hat{u}$. Then $Z \cdot G^{-1}$ is added to $\hat{u}$ to obtain the estimated message $\hat{x}$ by Eq. (18). The purpose of this zero-detector is to check if the input vector ZR is zero or not. If the two components are zero, then all the outputs of the weight comparator, which are described in the following, are also equal to zero. This indicates that the estimated correction factor u is zero, i.e., no error has occurred in the received code sequence Z. If ZR is not equal to zero, then the two components of ZR, i.e., $[ZR_1, ZR_2]$, in the first time frame are sent to the equality check circuit sequentially as shown in Fig. 8. The TG_i's (for $i = 1, 2, 3, 4$) shown in Fig. 8 are 3-bit registers. They are used to store the precalculated $u_j \cdot G$ values for the jth path. Since it requires only 2 frame times to choose a minimum Hamming weight path, four registers are needed to store $u_j \cdot G$. The reason for the use of 4 registers instead of one in this design is to avoid a more complex sequential computation of $u_j \cdot G$.

The loading of the TG_i's into the equality check circuit is controlled by a 2-bit counter which is capable of generating the required 4 different states. Because only 4 pairs of values need to be checked in every 2 frame times, four clock cycles are needed to finish the loading operation.

At the first clock cycle, ZR_1 and TG_1 are loaded into the equality check circuit. At the next clock cycle, ZR_2 and TG_2 are fed into the same circuit in sequence. The equality check circuit is implemented by the XOR arrays and an OR gate as shown in Fig. 9. For example, if ZR_1 is equal to TG_1,

then the output of the equality check circuit has logic level zero; otherwise it is one.

The output of the equality check circuit is sent to a 3-bit counter which accumulates the weight of a path in the error-trellis. After 4 clock-time, all the 4 pairs of the input sequence $Z \cdot R$ and $u_j \cdot G$ of each path on the error-trellis are compared. The output of the 3-bit counter which is the calculated weight of each path on the error-trellis is then sent to a weight comparator circuit. The weight comparator consists of Programmable Logic Array (PLA array). This is denoted by PLA1 in Fig. 10.

Also shown in Fig. 10 is an array of control gates and a table-lookup PLA. The inputs to each PLA1 in Fig. 10 are two 3-bit registers, W_i and W_j, which denote the weights of ith and jth path, respectively, in the error-trellis. The outputs of PLA1 are W_i or W_j depending on which is smaller, and a 1-bit signal, LR, to indicate if W_i is smaller than W_j. If W_i is smaller than W_j, then LR is zero; otherwise LR is one. The PLA1's are configured in a tree structure. The outputs of the first level PLA1's are sent to the second level PLA1's as their inputs and so forth.

For example, if W_1 is the smallest of all the weights, as indicated in Fig. 10 at point A, it is logic zero. This will turn on gate T_1 and turn off gate T_2. Then the value of C which is zero will be transferred through gate T_1 to point B. Since W_1 is the smallest value, the value at point D is zero. This turns on gates T_3 and T_4 and turns off gates T_5 and T_6. Therefore the values at points A and B, which are zero, together with the value at D which is zero as well, are transferred to the inputs of another type of PLA, labeled by PLA2 in Fig. 10.

The function of PLA2 is to form a mapping between the path and correction factor $\hat{u}$. Since there are eight different paths in the error-trellis, there are, correspondingly eight different $\hat{u}$'s. Finally, the correction factor $\hat{u}$ is added back to ZG^{-1}, by Eq. (18), as the estimated information $\hat{X}$. The estimated information $\hat{X}$ is then shifted out of this circuit sequentially.

References

1. Reed, I. S., and Truong, T. K., "Error-Trellis Decoding of Convolutional Code" to be published in *Proceedings IEE*.
2. Reed, I. S., and Truong, T. K., "New Syndrome Decoding for $(n, 1)$ Convolutional Codes," *Electronic Letters*, Vol. 19, No. 9, April 1983, pp. 344–346.
3. Viterbi, A. J., and Omura, J. K., *Principles of Digital Communication and Coding*. New York: McGraw-Hill, 1979.
4. Blahut, R. E., *Theory of Practice of Error Control Codes*. New York: Addison-Wesley, 1983.
5. Wyner, A. D., and Ash, R. B., "Analysis of Recurrent Codes," *IEEE Trans. Info. Theor. IT-9*, pp. 143–156, 1963.
6. Jensen, J. M., and Reed, I. S., "Error-Trellis Decoding of Convolutional Codes," submitted to *IEEE Trans. on Information Theory*.
7. Odenwalder, O. P., "Dual-K Convolutional Codes for Noncoherent Demodulated Channels," *Proceedings of International Telemetry Conference (ITC)*, Vol. 12, pp. 165–174, 1978.

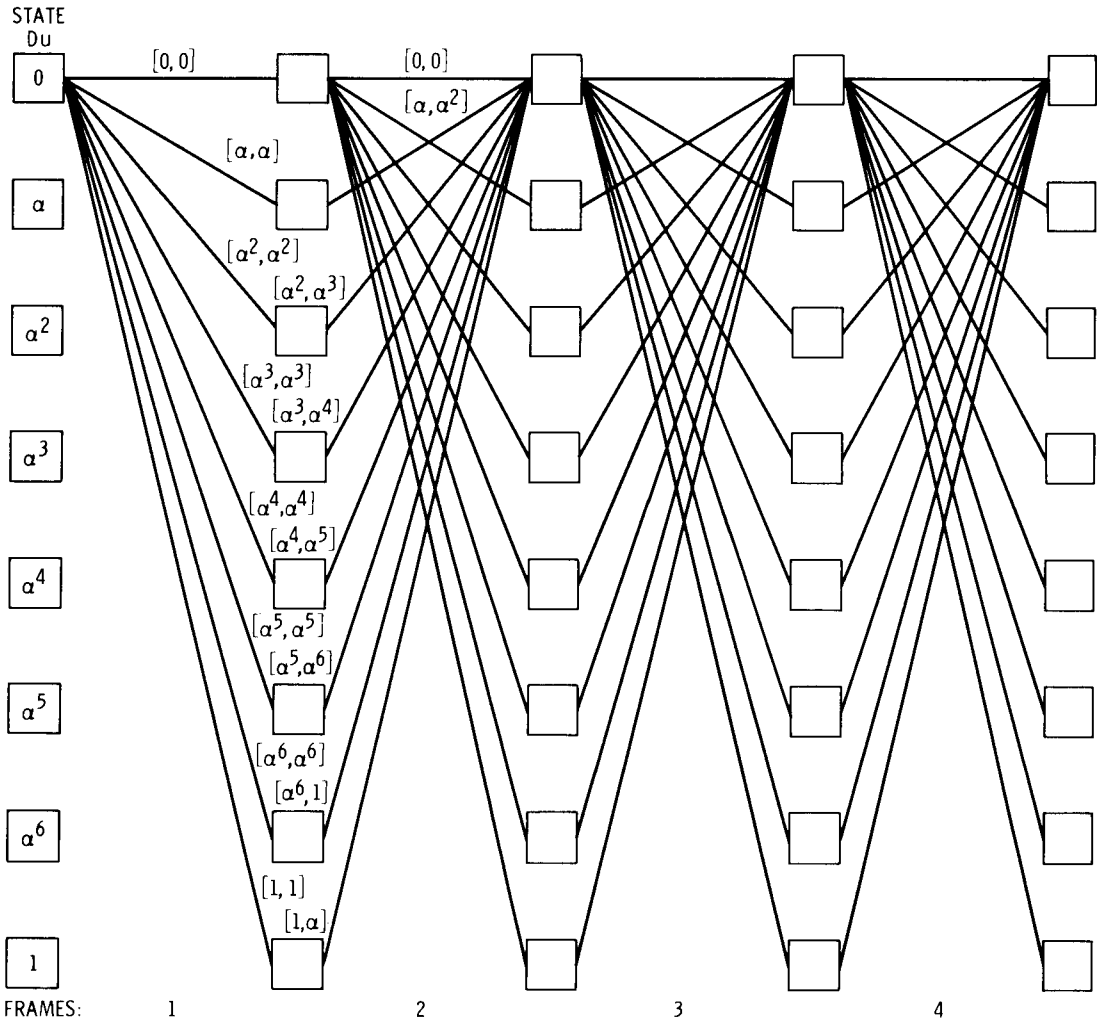

Fig. 1. Pruned error-trellis with no error outputs *u*(*D*) *G*(*D*)

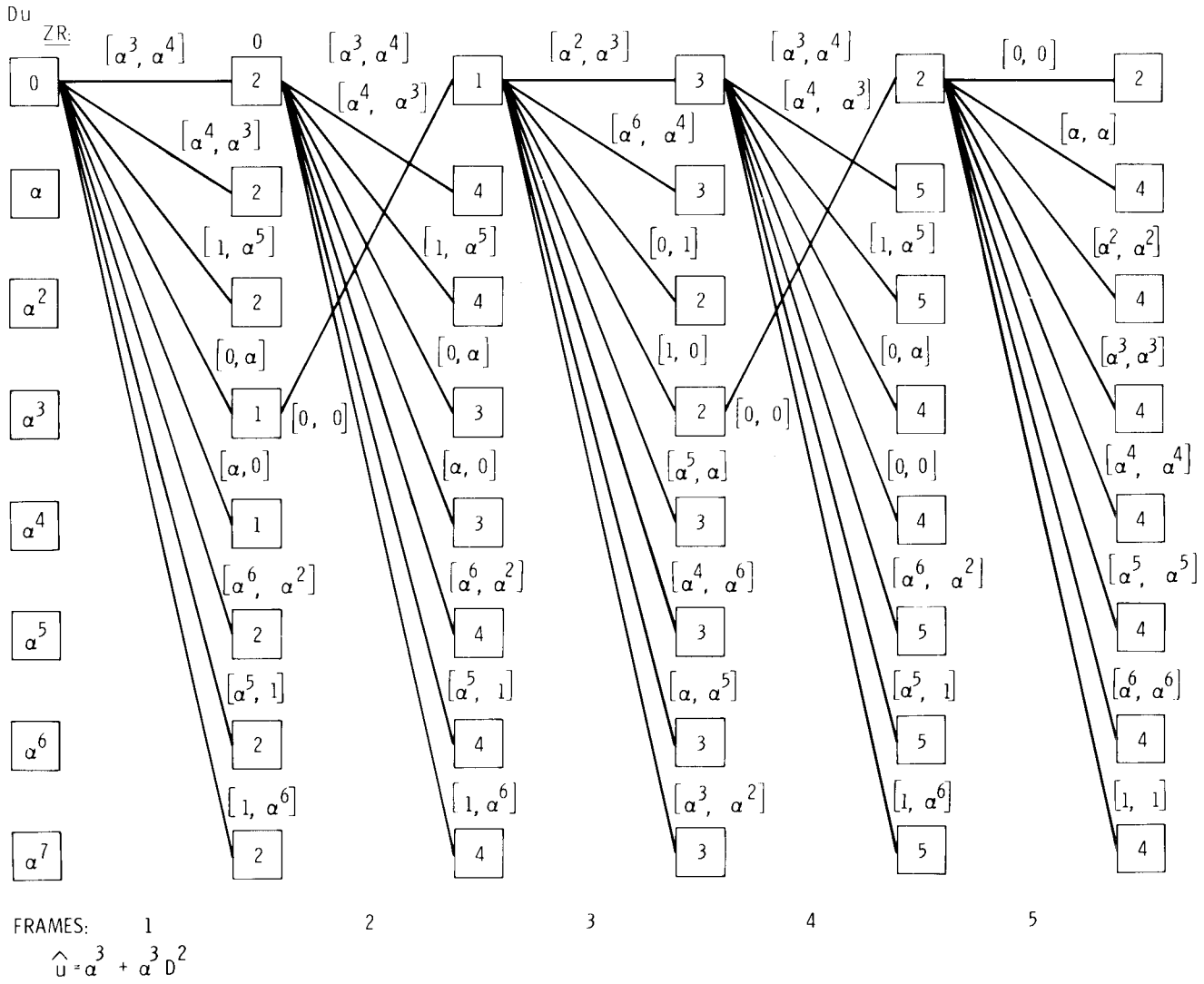

Fig. 2. Minimum-error path $\hat{u}$ in pruned error trellis

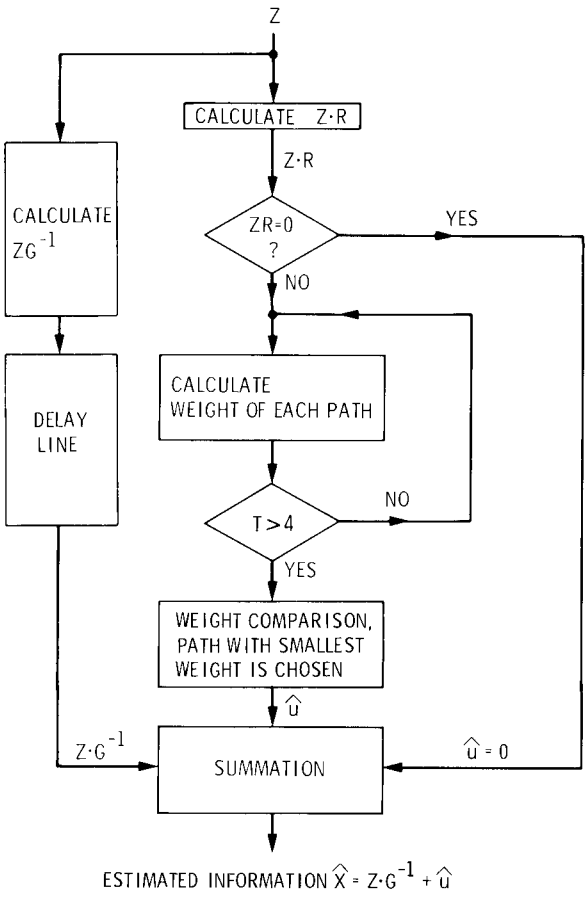

Fig. 3. Flow chart of error-trellis syndrome decoding of dual-3, rate 1/2, convolutional code

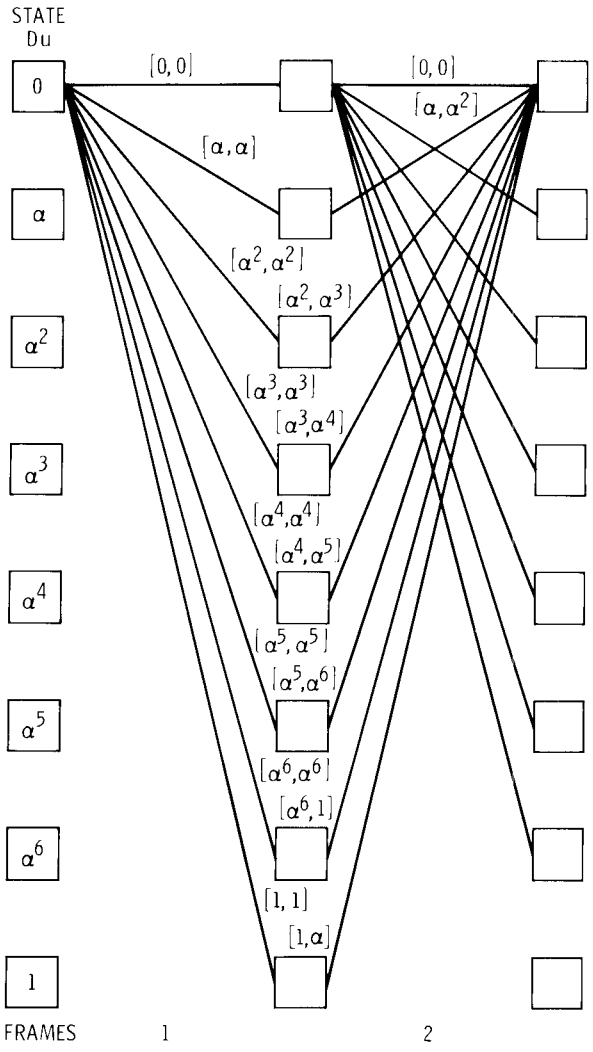

Fig. 4. A basic cell of a pruned error-trellis with no error outputs *u(D) G(D)*

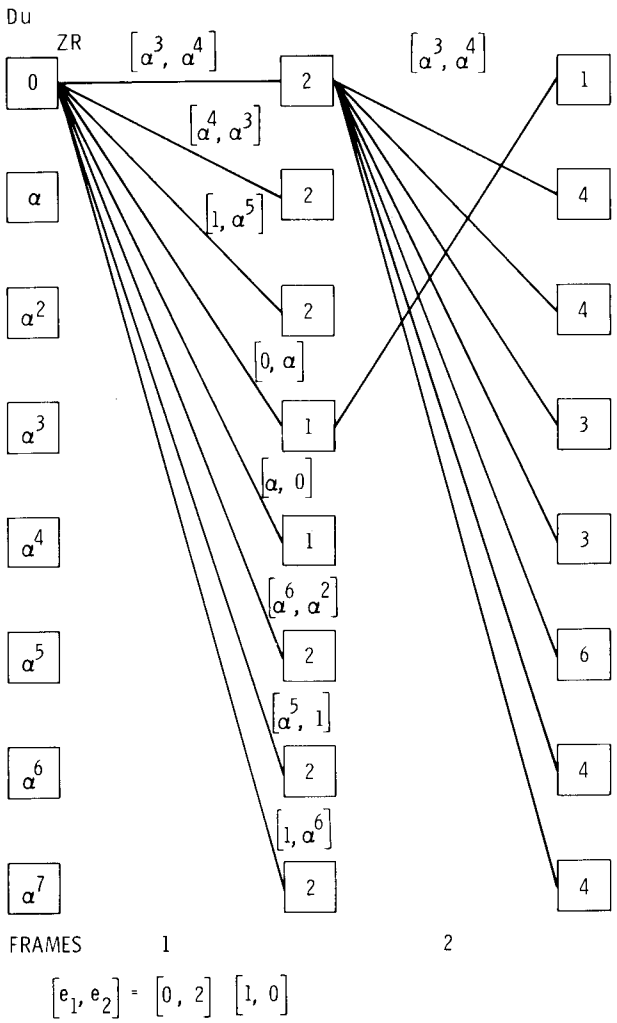

Fig. 5. Minimum-error path $\hat{e}$ in pruned error trellis in two frames

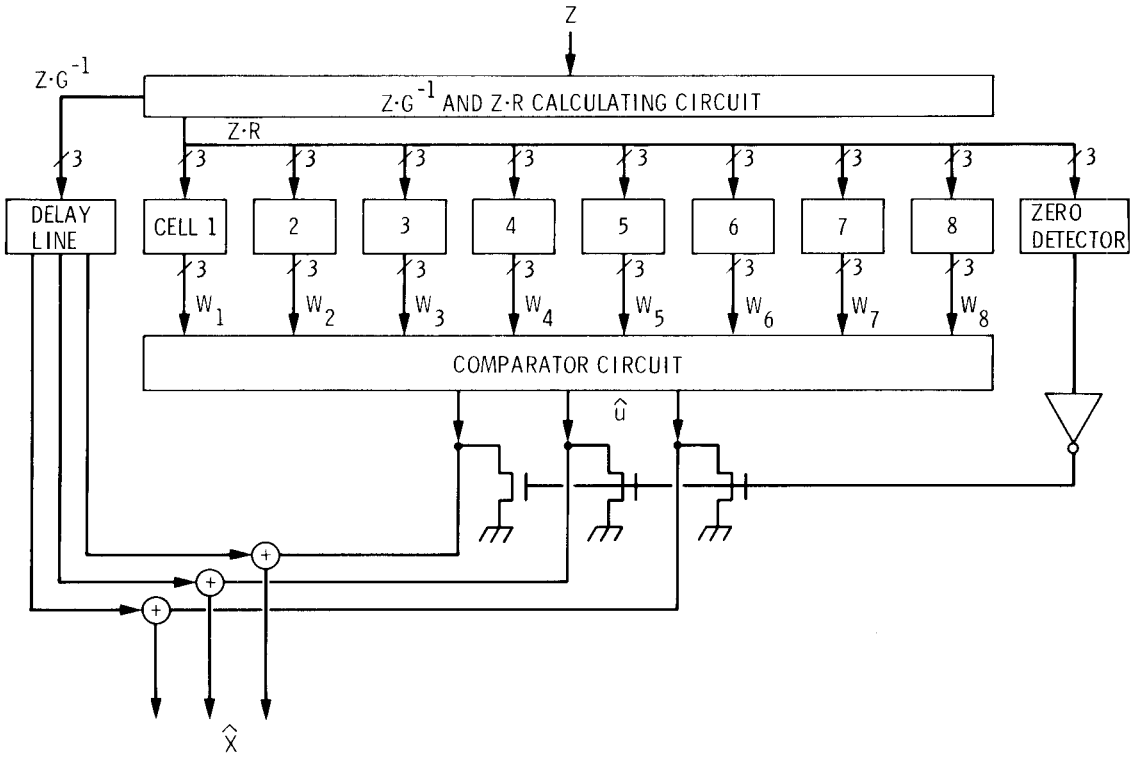

Fig. 6. Block diagram of a dual-3, rate I/2, convolutional code decoder

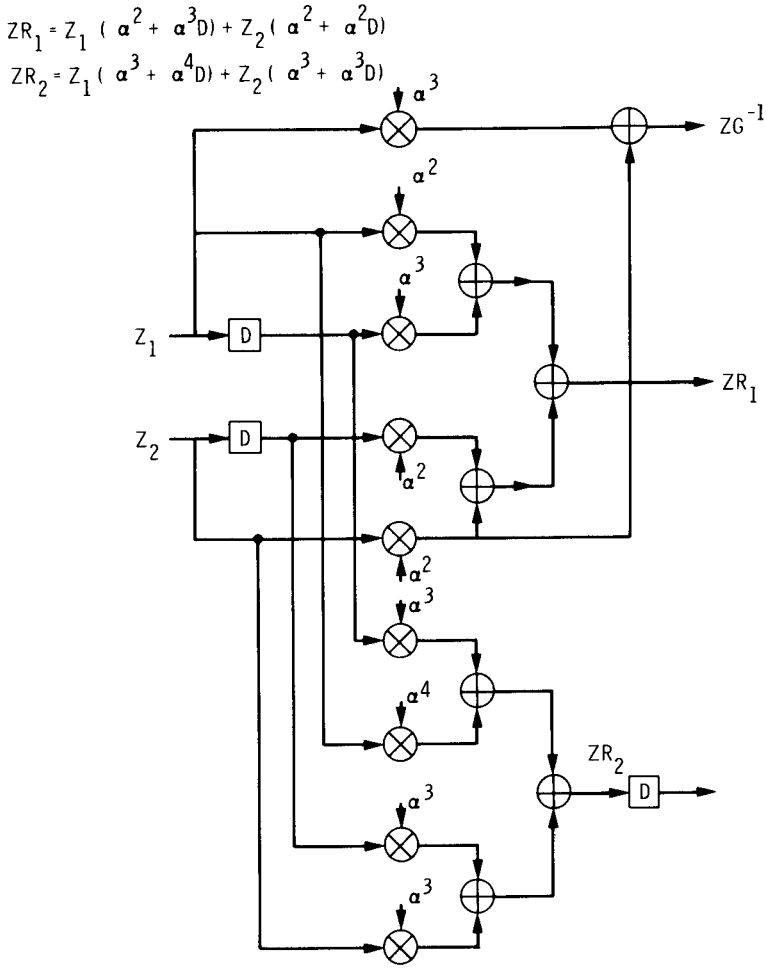

Fig. 7. Block diagram of circuit for calculating $Z \cdot R$ and $Z \cdot G^{-1}$

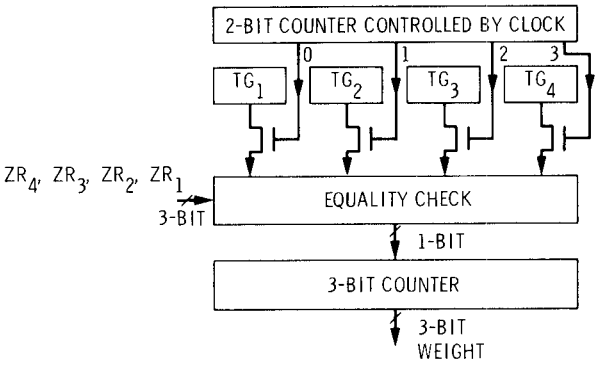

Fig. 8. Block diagram of each cell for calculating weight

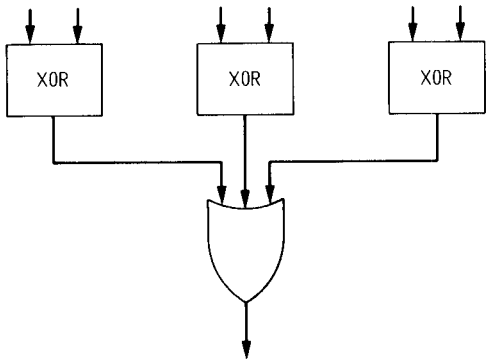

Fig. 9. Logic diagram of equality check circuit

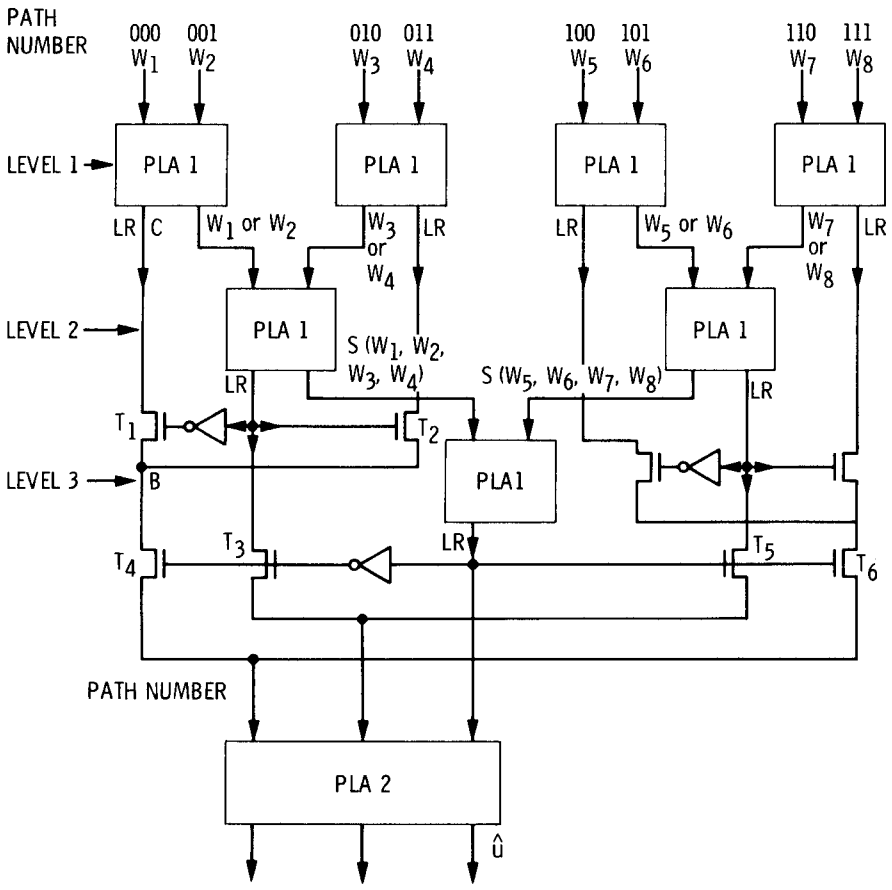

Fig. 10. Block diagram of weight comparator circuit. The inputs to this circuit are weights of each path. The outputs are estimated corrector factors.

Appendix A

Approximation of the Set $E^{(-1)}$ for Dual-K CCs

Let $G^{-1}(D)$ be a right inverse to the generator matrix (25b) of a dual-K CC of length n. Then $E^{(-1)} = \{eG^{-1} : e \epsilon E\}$. Define

$$\tilde{E}_1^{(-1)} = \{u : W_H(u_j, u_{j+1}) \leqslant t, \text{ for all } j \geqslant 0\} \tag{A-1}$$

In the following it will be shown that $\tilde{E}_1^{(-1)}$ is a suitable approximation to $E^{(-1)}$. Since for $t \geqslant 2$, $E^{(-1)} \leqslant \tilde{E}_1^{(-1)}$, the only nontrivial case is for $t = 1$, that is for $n = 2$. It is readily verified that

$$G^{-1} = \left[\frac{g_{12}}{(g_{12}+g_{11})}, \frac{g_{11}}{(g_{12}+g_{11})}\right]^T = [g_1, g_2]^T \tag{A-2}$$

is a right inverse matrix to the generator matrix of a dual-K CC of length 2. Note that g_1 and g_2 are nonzero elements in $GF(2^K)$.

Since the dual-K CCs of length 2 are 1-error-per-block-length-correcting CCs, the set of sequences

$$\tilde{E} = \{v : W_H(v_j, v_{j+1}) \leqslant 1, \text{ for all } j \geqslant 0\} \tag{A-3}$$

is a desirable approximation to E. Thus

$$\tilde{E}^{(-1)} = \{v = vG^{-1} : v \epsilon \tilde{E}\} \tag{A-4}$$

is an approximation to $E^{(-1)}$. Also from the following lemma, one has that $\tilde{E}^{(-1)} = \tilde{E}_1^{(-1)}$.

Lemma. Let $n = 2$, then $\tilde{E}^{(-1)} = \tilde{E}_1^{(-1)}$.

Proof: Let $u(D) \epsilon \tilde{E}^{(-1)}$. Then

$$u(D) = \sum_{j=0}^{\infty} u_j \cdot D^j = v(D) \cdot G^{-1}$$

where $v(D) \epsilon \tilde{E}$. And from (A-2)

$$u(D) = \sum_{j=0}^{\infty} (v_{1j} - v_{2j}) D^j [g_1, g_2]^T$$

$$= \sum_{j=0}^{\infty} (v_{1j} g_1 + v_{2j} g_2) D^j$$

Thus,

$$\begin{aligned} w_H(u_j, u_{j+1}) &= w_H(v_{1j} g_1 + v_{2j} g_2, v_{1j+1} g_1 + v_{2j+1} g_2) \\ &\leqslant w_H(v_{1j} g_1, v_{1j+1} g_1, v_{2j} g_2, v_{2j+1} g_2) \\ &= w_H(v_{1j}, v_{1j+1}, v_{2j}, v_{2j+1}) \\ &= w_H(v_1, v_2) \leqslant 1 \end{aligned}$$

From (A-3), since $v(D) \epsilon E_1$, this implies that $E^{(-1)} \subseteq E_1^{(-1)}$.

To show that $\tilde{E}_1^{(-1)} \subseteq \tilde{E}^{(-1)}$, let $u(D) \epsilon \tilde{E}_1^{(-1)}$, construct the sequence $v(D) = (v_1(D), v_2(D)) = (g_1^{-1} u(D), 0)$. Since $g_1^{-1} \cdot u(D) \epsilon \tilde{E}_1^{(-1)}$, it follows from (A-3) that $v(D)$ is in $\tilde{E}$. Hence $v(D) \cdot G^{-1} = v(D) \cdot [g_1, g_2]^T = u(D)$ is in $\tilde{E}^{(-1)}$.