Optimal Soft Decision Block Decoders Based on Fast Hadamard Transform

YAIR BE'ERY AND JAKOV SNYDERS, MEMBER, IEEE

Abstract—**An approach for efficient utilization of fast Hadamard transform in decoding binary linear block codes is presented. Computational gain is obtained by employing various types of concurring codewords, and memory reduction is also achieved by appropriately selecting rows for the generator matrix. The availability of these codewords in general, and particularly in some of the most frequently encountered codes, is discussed.**

I. Introduction

IN THIS PAPER we present an approach for efficient utilization of the fast Hadamard transform (FHT) in soft decision (also named analog) maximum likelihood decoding of binary linear block codes. Among the relatively few existing results that involve application of FHT for decoding, perhaps the oldest one is the Hadamard transform decoding of first-order Reed–Muller codes proposed by Green [1]; see also [2]–[6] and [7, p. 419]. Application of FHT for decoding maximum length shift-register codes is presented in [8] and [9]. Utilization of the Green-machine concept for decoding binary block and convolutional codes is described by Clark and Davis in [10], but their procedure is efficient only for low-rate codes. An efficient maximum likelihood soft-decision decoding algorithm based on trellis description of high-rate block codes was conceived by Wolf [11].

The various alternative approaches investigated for obtaining optimal soft-decision decoders include the maximum likelihood symbol-by-symbol decoding rules [12], [13] and the minimum mean-square error decoding algorithm [14], [15] that uses Fourier and Hadamard transforms for computing certain parameters. Suboptimal soft-decision block decoding by implementing reduced trellis search is discussed in [16], and two different suboptimal soft-decision decoders [17], [18] apply erasures for reducing the number of candidate codewords. A maximum likelihood hard-decision decoding algorithm applicable to certain Reed–Muller codes is presented in [19]. Recently, a new method of maximum likelihood hard-decision decoding of binary codes was proposed [24].

The basic algorithm developed in Section II (Algorithm A) is slightly more general but essentially the same as the one described in [10]. However, the derivation we provide is formal even though straightforward. Algorithm A is applicable for decoding any binary block or truncated convolutional code and is therefore, in a sense, universal. However, it is inefficient unless the rate of the block code is extremely low or the (truncation) length of the code is short. In subsequent sections we develop somewhat more complex, but significantly more efficient decoding rules by taking into account the structure of the code under consideration. In Section III we use zero-concurring (i.e., orthogonal when viewed as real) codewords for reducing computational complexity, whereas in Section IV concurring (more commonly but misleadingly named orthogonal) codewords are employed for the same purpose. The availability of these codewords in general, and particularly in some of the most frequently encountered codes, is discussed, and the efficiency of the various algorithms is compared for five particular codes. In Section V an algorithm with reduced memory requirement is presented. A different approach for reducing computational complexity, resembling that of [11] but without relying on a trellis diagram, which is applicable also to convolutional and combined codes, is presented elsewhere [20].

II. The Basic Algorithm

Let $\mathscr{C}$ be an (n, k) binary linear block code of length n and dimension k. Assume for a while that a codeword is transmitted through a discrete memoryless channel with output alphabet $\mathbb{K}$ and transition probabilities $P_j(v) = P(v|j)$, $v \in \mathbb{K}$, $j \in \mathrm{GF}(2)$ and that the word $\boldsymbol{v} = (v_0, v_1, \cdots, v_{n-1})$, $v_i \in \mathbb{K}$, is observed at the output. Optimal decoding consists of finding a codeword $\boldsymbol{c} = (c_0, c_1, \cdots, c_{n-1}) \in \mathscr{C}$ that maximizes $P(\boldsymbol{v}|\boldsymbol{c}) = \prod_{i=0}^{n-1} P(v_i|c_i)$, that is, maximizes the a posteriori probability $P(\boldsymbol{c}|\boldsymbol{v})$ provided that $P(\boldsymbol{c}) = 2^{-k}$ for all $\boldsymbol{c} \in \mathscr{C}$. One may as well maximize with respect to all $\boldsymbol{c} \in \mathscr{C}$ the following expression:

$$M = \beta \sum_{i=0}^{n-1} \log P(v_i|c_i) + \gamma$$
$$= \beta \sum_{i \in \Lambda_0} \log p_0(v_i) + \beta \sum_{i \in \Lambda_1} \log p_1(v_i) + \gamma$$

Manuscript received October 30, 1984; revised September 30, 1985.
J. Snyders is with the Department of Electronic Communications, Control and Computer Systems, Tel-Aviv University, Tel-Aviv, Israel.
Y. Be'ery is with the Department of Electronic Communications, Control and Computer Systems, Tel-Aviv University, Tel-Aviv, Israel.
IEEE Log Number 8406948.

where β is any positive number, γ it any real number, $\Lambda_0 = \{i: c_i = 0\}$, and $\Lambda_1 = \{i: c_i = 1\}$. Setting $\beta = 2$ and

$$\gamma = -\sum_{i=0}^{n-1} [\log p_0(v_i) + \log p_1(v_i)]$$

(γ is indeed constant with respect to c), it readily follows that $M = \Sigma_{i \in \Lambda_0}\mu(v_i) - \Sigma_{i \in \Lambda_1}\mu(v_i)$, where $\mu(v) = \log[p_0(v)/p_1(v)]$, and consequently

$$M = \sum_{i=0}^{n-1} (-1)^{c_i}\mu(v_i). \tag{1}$$

Thus the quantity $M = M(c, v)$, a so-called metric, is computed by labeling each location i with the logarithm of the likelihood ratio of v_i and then summing all properly signed labels. In particular, for a binary symmetric channel with crossover probability η, $0 < \eta < 1/2$, we have $\mu(0) = -\mu(1) = \log[(1-\eta)/\eta] > 0$. Consequently, the log-likelihood ratio in (1) may be replaced, without loss of generality, by the label $\mu(v) = (-1)^v$. Therefore, an optimal c brings the Hamming distance between c and v to minimum, as expected.

In the case of a continuous-output memoryless channel characterized by two transition probability densities $f_j(v) = f(v|j)$, $v \in \mathbb{R}$, $j = 0, 1$, where $\mathbb{R}$ is the real line, optimal decoding consists of maximizing $f(v|c) = \Pi_{i=0}^{n-1} f(v_i|c_i)$, and a derivation similar to the preceding one yields (1), where now $\mu(v) = \log[f_0(v)/f_1(v)]$. In particular, for the case of additive Gaussian-noise channel and where the components of v take values from an appropriate analog alphabet, say $\{\sqrt{E}, -\sqrt{E}\}$, $\mu(v)$ is proportional to v, and consequently, we may set $\mu(v) = v$ in (1). Hereafter $\mu(v)$ stands for any one of the labels mentioned earlier.

Let G be the generator matrix of $\mathscr{C}$ that represents the encoding, that is, a message $s \in GF(2)^k$ is mapped onto a codeword c according to $c = sG$, and let g_i for $i = 0, 1, \cdots, n-1$ denote the columns of G. Then (1) becomes

$$M(s) = \sum_{i=o}^{n-1} (-1)^{\langle s, g_i \rangle}\mu(v_i) \tag{2}$$

where $\langle \cdot , \cdot \rangle$ stands for inner product over GF(2) and the dependence of M on s is emphasized. The maximization is carried out with respect to all $s \in GF(2)^k$. Thus the message is straightforwardly decoded, whether the encoding is systematic or not, by adding n labels signed in each of 2^k prescribed ways and comparing the sums. More important, the decoding may be accomplished with practically the same computational complexity if in (2) the columns $\{g_i: i = 0, 1, \cdots, n-1\}$ of *any* generator matrix of $\mathscr{C}$ (or any of its bit-permuted equivalents) are used, by appropriately transforming the optimal s that results from maximizing (2). Hence in the sequel we shall adopt freely any convenient row-equivalent and column-permuted version of G.

It is rewarding to perform the summation in (2) according to increasing binary value of the columns of G, with all labels $\{\mu(v_i)\}$ that correspond to any common value being combined into a single label and $\mu = 0$ inserted whenever a value is missing: for any $y = (y_0, y_1, \cdots, y_{k-1}) \in GF(2)^k$ let $b(y) = \Sigma_{j=0}^{k-1} y_j 2^j$ and denote

$$u_j = \begin{cases} \sum_{i \in A_j} \mu(v_i), & A_j \neq \varnothing \\ 0, & A_j = \varnothing \end{cases}$$

where $A_j = \{i: b(g_i) = j\}$ for $j = 0, 1, \cdots, 2^k - 1$. Then

$$M(s) = \sum_{j=0}^{2^k-1} (-1)^{\langle s, g_j^* \rangle} u_j \tag{3}$$

where $g_j^* \in GF(2)^k$ is defined by $b(g_j^*) = j$. For $b(s) = i$ the right side of (3) is recognized as the ith component of $U = (U_0, U_1, \cdots, U_{2-1}^k)$ given by

$$U = uH_k \tag{4}$$

where

$$u = (u_0, u_1, \cdots, u_{2^k-1}) \tag{5}$$

and H_k is the $2^k \times 2^k$ Sylvester-type [7, p. 44] naturally ordered Hadamard matrix. This proves the following maximum likelihood decoding rule.

Algorithm A:

a) Locate the largest component of U given by (4), say U_i.
b) Set for s the k-bit radix-2 expansion of i.

The manipulations involved in the application of the algorithm are illustrated in Fig. 1. For convenience we phrased Algorithm A under the assumption that there is a unique optimal message word; we shall adhere to this assumption also in the sequel. Of course, it may happen—although rather rarely in true soft-decision decoding—that more than one component of U achieves the maximal value. In that case either one of them is selected arbitrarily or a decoding failure is announced. Also, the ratio of a uniquely attained maximum and the second-largest value attained by the components of U may serve as a measure of reliability of the decoded message vector.

Algorithm A, when implemented with the aid of FHT, requires only $k2^k$ additions [21] (excluding the search for the largest component of U and the usually negligible amount of additions needed for obtaining the modified labels u_j). In comparison with the straightforward computation based on (2), there is a gain of n/k, which is significant for low-rate codes. An obvious candidate for taking a full advantage of this gain is the class of simplex (or maximal length shift-register) codes [8], [9]. A closely related example is the class of first-order Reed–Muller codes; in this case

$$G = \left(\frac{G^T}{1\ 1\ 1 \cdots 1} \right)$$

where the columns of G^T are all the 2^{k-1} distinct binary numbers of length $k-1$, thus $A_j = \varnothing$ for $j = 0, 1, \cdots, 2^{k-1} - 1$. Let $U' = u'H_{k-1}$ where $u' = (u_{2^{k-1}},$

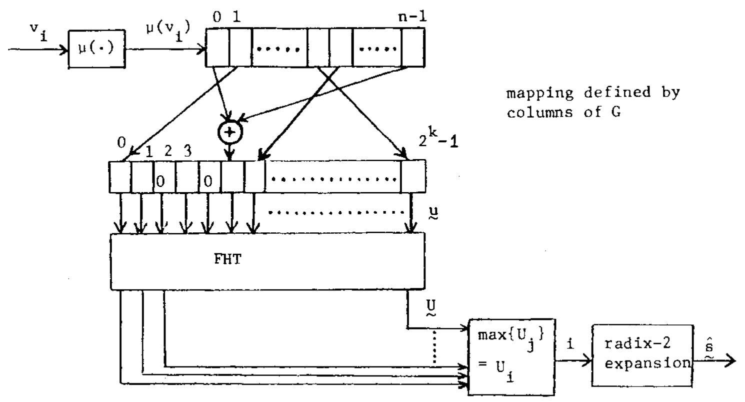

Fig. 1. Block diagram representation of Algorithm A.

$u_{2^{k-1}+1}, \cdots, u_{2^k-1})$, and note that [21]

$$H_k = H_1 \otimes H_{k-1} = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix}$$

where $\otimes$ stands for Kronecker product. It follows by (4) that decoding may be performed by locating the component of U' with the largest absolute value, say U_i', and taking for s the k bit radix-2 expansion of i or $i + 2^{k-1}$, depending on the sign of U_i'. This well-known result [1]–[6], [7, p. 419], derived here directly from Algorithm A, demonstrates the possibility of reducing the number of additions by taking into account the structure of the code. We shall pursue this matter in the following sections.

III. Computational Gain by Employing Zero-Concurring Codewords

Definition 1: Let L be a nonnegative integer. A set of J independent vectors in GF(2)^n is called *L-concurring* if

1) in each of some L positions all J vectors have component one, and
2) in all other positions at most one of the J vectors has component one.

In particular, at most one member of a set of *zero-concurring* vectors has component one in any of the positions. A set of *concurring* vectors is a set of L-concurring vectors where $L \geq 1$.

The terminology introduced stems from the word concurring, encountered in the literature on majority logic decoding [22]; the more commonly used term in that context is orthogonal. Evidently, a set containing one nonzero vector is zero-concurring. Also, a set of two independent vectors is L-concurring for some $L \geq 0$. For any matrix K over GF(2) let D_K be the size of a maximal collection of pairwise-different *nonzero* columns of K.

Lemma 1: Let K be a $J \times n$ matrix over GF(2). There exists a matrix row-equivalent to K and with all its rows zero-concurring iff $D_K = J$. Furthermore, assuming that all the rows of K are concurring, K is not row-equivalent to any matrix with all its rows zero-concurring iff $D_K = J + 1$.

Proof: Assuming $D_K = J$, let $K' = (A \quad B)$ be a column-permuted version of K such that A is invertible. Then any nonzero column of B is also a column of A and $A^{-1}K' = (I \quad A^{-1}B)$, where each column of $A^{-1}B$ contains at most a single 1. Hence by inverse column permutation of $A^{-1}K'$, the desired matrix is obtained. Conversely, any $J \times n$ matrix K' with zero-concurring rows satisfies $D_{K'} = J$, and a row operation does not affect the number of different columns. As for the second statement, either $D_K = J$ or $D_K = J + 1$, because the rows obtained by puncturing the positions where concurring occurs are either linearly dependent or zero-concurring. Hence the result follows by the first statement.

Assume that $\mathscr{C}$ possesses a set of J zero-concurring codewords, and let

$$G = \begin{pmatrix} G^T \\ \hline G^B \end{pmatrix} \tag{6}$$

where G^B is a $J \times n$ matrix with those codewords listed as its rows. Denote the columns of G^T and G^B, respectively, by g_i^T and g_i^B for $i = 0, 1, \cdots, n-1$ and partition all message vectors accordingly: $s = (s^T | s^B)$. Then $\langle s, g_i \rangle = \langle s^T, g_i^T \rangle + \langle s^B, g_i^B \rangle$, and (2) becomes

$$M(s^T, s^B) = \sum_{i \in \Omega_\infty} (-1)^{\langle s^T, g_i^T \rangle} \mu(v_i) + \sum_{j=0}^{J-1} (-1)^{s_j^B} \sum_{i \in \Omega_j} (-1)^{\langle s^T, g_i^T \rangle} \mu(v_i) \tag{7}$$

where Ω_∞ is the (possibly empty) set $\{i\colon g_i^B = \mathbf{0}\}$ and $\Omega_j = \{i\colon b(g_i^B) = 2^j\}$ for $j = 0, 1, \cdots, J-1$ are nonempty sets. Rearrangement of the summation over Ω_∞ and over each Ω_j *separately* according to increasing value of $b(g_i^T)$ converts each sum into the ith component, where $i = b(s^T)$, of a 2^{k-J}-dimensional Hadamard transformed vector $U^{(\infty)}$, resp. $U^{(j)}$ given by $U^{(\infty)} = u^{(\infty)} H_{k-J}$ and $U^{(j)} = u^{(j)} H_{k-J}$ with $u^{(\infty)}$ and $u^{(j)}$ being defined similarly to u in (5). Thus the maximum of $M(s^T, s^B)$ with respect to all $s^T \in \mathrm{GF}(2)^{k-J}$ and $s^B \in \mathrm{GF}(2)^J$ is equal to the largest component of Y given by

$$Y = \max_{s^B \in \mathrm{GF}(2)^J} \left\{ U^{(\infty)} + \sum_{j=0}^{J-1} (-1)^{s_j^B} U^{(j)} \right\} \tag{8}$$

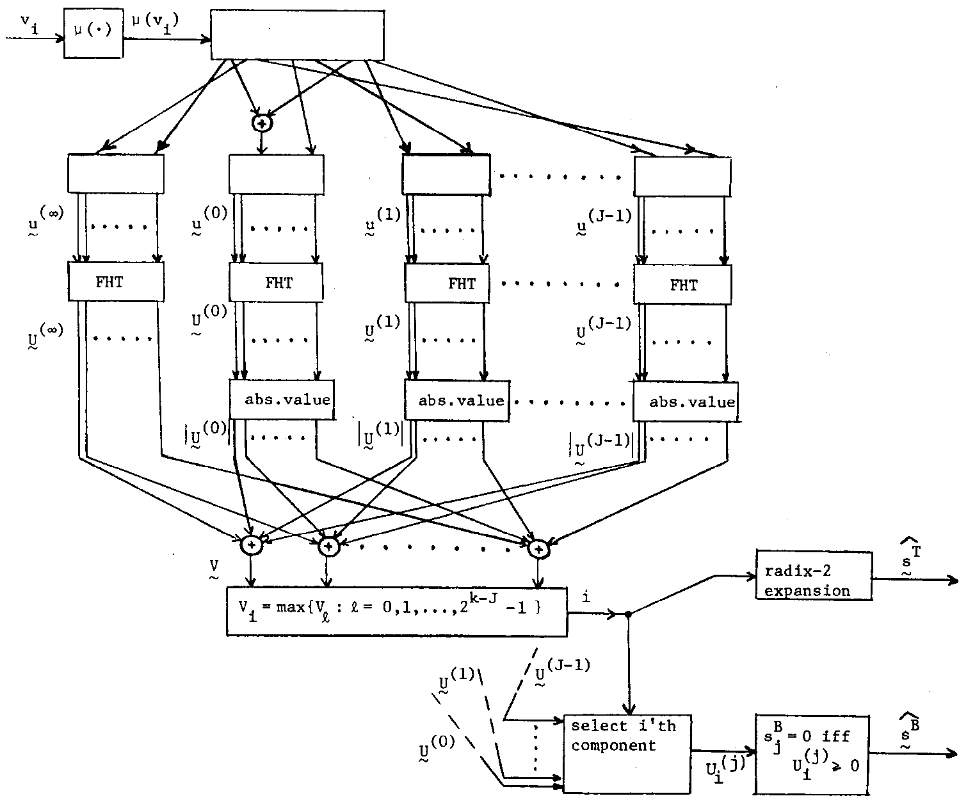

Fig. 2. Block diagram illustration of Algorithm B.

where the *maximum* is computed *componentwise*. For any vector X over the reals, let $|X|$ be the vector obtained by taking *componentwise absolute value*. Denote

$$V = U^{(\infty)} + \sum_{j=0}^{J-1} |U^{(j)}|. \tag{9}$$

It is easily seen that $Y = V$. This proves the following optimal soft decision decoding procedure.

Algorithm B:

a) Locate the largest component of V given by (9), say V_i.
b) Take for s^T-the k-J bit radix-2 expansion of i and set $s_j^B = 0$ whenever $U_i^{(j)}$, the ith component of $U^{(j)}$, is nonnegative; otherwise, set $s_j^B = 1$.

Fig. 2 provides block-diagram illustration of Algorithm B. The number N_B of additions required by the decoding procedure, including calculation of absolute values, is as follows:

$$\begin{aligned} N_B &= (J+1)(k-J)2^{k-J} + J2^{k-J} \\ &= \left[k(J+1) - J^2\right]2^{k-J}, \end{aligned} \tag{10}$$

and it is somewhat less if $\Omega_\infty = \varnothing$. Thus a significant improvement is obtained with large J. Further reduction of computational complexity is achievable if for some $j \in \{0, 1, \cdots, J-1\}$ the number n_j of elements of Ω_j satisfies $n_j \leq k - J$. For such j direct (rather than fast) transform evaluation of $|U^{(j)}|$ requires, due to sparseness of $u^{(j)}$ and repetition of entries in $|U^{(j)}|$, only $n_j 2^{n_j - 1}$ addition-equivalent operations. The case $n_\infty < k - J$ also allows a reduction of complexity. It may happen that $n_j \leq k - J$ for all $j = 0, 1, \cdots, J-1$ and also that $n_\infty \leq k - J$; Table I provides some examples. Then the total number N_B' of addition-equivalent operations, using direct Hadamard transform, is given by

$$N_B' = n_\infty 2^{n_\infty} + \sum_{j=0}^{J-1} n_j 2^{n_j - 1} + J2^{k-J}. \tag{11}$$

The $J+1$ inequalities $n_j \leq k - J$ can be satisfied only if $n/(J+1) \leq k - J$; this condition implies in turn $(k-1)^2 \geq 4(n-k)$ and sets both lower and upper bounds on J. By (11) we conclude that usually a large value of J is needed for securing high efficiency.

It is evident that

$$J \leq \left\lfloor \frac{n}{d} \right\rfloor \tag{12}$$

where d is the minimum distance of $\mathscr{C}$. For certain codes the foregoing bound is achievable. Maximal sets of zero-concurring codewords for some codes where the bound is met, for example, the Golay codes, and for some other codes where it is impossible to attain the bound are listed in Table I. Those lists were obtained by computer search. A result of a more general kind is the following.

Theorem 1: In $\mathscr{C} = \mathrm{RM}(r, m)$, the Reed–Muller code of order r and length $n = 2^m$, there is a set of $J = n/d = 2^m/2^{m-r} = 2^r$ zero-concurring codewords.

Proof: Assume that $r < m$ since otherwise $\mathscr{C}$ contains all vectors of length $n = 2^m = 2^r$ and the conclusion triv-

TABLE I
MAXIMAL SETS OF ZERO-CONCURRING AND SETS OF CONCURRING CODEWORDS

Code	Generator polynomial	n	k	d	Zero-concurring		L-concurring	
					bound	code words	bound	code words
Golay	(0,1,5,6,7,9,11)	23	12	7	3	11001011001001010000000	5	11000111010100000000000
						00000000010010101001011		01100011101010000000000
						00110100100100000110100		01000011000000111001000
								01010011000001000010010
								01001011000000000100101
Expurgated Golay	(0,2,5,8,9,10,11,12)	23	11	8	2	10100100111110000000000	4	10100100111110000000000
						00001000000000111101101		01010010011111000000000
								00001000011110001001100
								00000000011110110100001
Hamming	(0,1,4)	15	11	3	5	100001000010000	9	110000000100001
						010000100001000		110010000000000
						001000010000100		110100010000000
						000100001000010		110001001000000
						000010000100001		111000000010000
								110000100000100
								110000000001010
Expurgated Hamming	(0,2,4,5)	15	10	4	3	101011000000000	6	101011000000000
						000100000001101		001010110000000
						000000110100010		011010001000000
								001110000000100
								001010000100010
								001010000001001
BCH	(0,4,6,7,8)	15	7	5	3	001001001001001	5	*
						010010010010010		same as zero-concurring
						100100100100100		
BCH	(0,2,4,6,7,9,10, 13,17,18,20)	31	11	11	2	any nonzero code word and its complement	4	* any two independent codewords
Expurgated BCH	(0,1,2,3,4,5,6,8,9, 11,13,14,17,19,20,21)	31	10	12	2	any nonzero codeword	4	* any two independent codewords
BCH	(0,1,2,4,6,7,8)	17	9	5	3	11101011100000000	5	* partially concurring set
						00010000011001100		11101011100000000
						00000100000110011		11010001011000000
								11000001000001100
								00000100000110011
BCH	(0,1,4,5,7,8,9)	21	12	5	4	100100100100100100100	7	*
								110011011100000000000
						010010010010010010010		011001101110000000000
						001001001001001001001		010101001100011001000
								010001001100100000110
								010001001101000110001

*Lengthy by nonexhaustive search

ially holds. The polynomials $(1+x)^i$, where i satisfies $i < 2^m$ and $w_2(i) \geq m-r$, $w_2(i)$ standing for the radix-2 weight of i, belong to $\mathscr{C}$ and are independent [23]. In particular, $i = 2^{m-r} - 1 + j2^{m-r}$ for all $j = 0, 1, \cdots, 2^r - 1$ are eligible because $w_2(2^{m-r} - 1) = m - r$. Let $a_j(x) = (1+x)^{2^{m-r}-1+j2^{m-r}}$ for $j = 0, 1, \cdots, 2^r - 1$, and let K be a $2^r \times 2^m$ matrix that has these codewords as its rows. Since

$$a_j(x) = (1+x)^{2^{m-r}-1}\left[(1+x)^{2^{m-r}}\right]^j$$
$$= \left(1 + x + x^2 + \cdots + x^{2^{m-r}-1}\right)\left(1 + x^{2^{m-r}}\right)^j, \quad (13)$$

it follows that the coefficients of each $a_j(x)$ are zero except for one or more all-one blocks of blocklength 2^{m-r} that start at a location which is some multiple of 2^{m-r}. Hence the size of a collection of pairwise-different nonzero columns of K is at most $2^m/2^{m-r} = 2^r$. However, since the rows of K are independent, K must possess at least 2^r independent columns. Hence the result follows by Lemma 1.

For example, the RM(2,5) code has dimension $k = 16$, and there exists a set of $J = 4$ zero-concurring codewords, whereas for RM(3,5) the corresponding numbers are $k =$

26 and $J = 8$, respectively. The computational gain is indeed high in these cases. In general, for any fixed value of m, $J = 2^r$ increases monotonically with k, but k increases more rapidly unless the rate is higher. Therefore, Algorithm B is practical only for either low-rate or short Reed–Muller codes. Presumably a similar conclusion applies to most if not all large classes of block codes. A decoding procedure that is more useful for some higher rate codes is developed in the next section, and an efficient algorithm for high-rate block codes is presented elsewhere [20].

By puncturing the set of 2^r zero-concurring codewords introduced in the proof of Theorem 1, we obtain, provided that $r < m$, a set of 2^r zero-concurring codewords for the punctured Reed–Muller code RM*(r, m) of length $n = 2^m - 1$. However, for RM*(r, m)

$$\frac{n}{d} - 2^r = \frac{2^m - 1}{2^{m-r} - 1} - 2^r = \frac{2^r - 1}{2^{m-r} - 1},$$

and consequently, $\lfloor n/d \rfloor > 2^r$ whenever $r \geq m/2$. For a subclass of punctured Reed–Muller codes and for many other cyclic codes, the following result provides large, sometimes bound-meeting, sets of zero-concurring codewords.

Let $\mathscr{C}$ be cyclic, denote by $g(x)$ and $h(x)$ its generator and check polynomial, respectively, and let J be a positive divisor of n. The polynomial

$$a_J(x) = 1 + x^J + x^{2J} + \cdots + x^{n-J} = \frac{1 + x^n}{1 + x^J} \quad (14)$$

belongs to $\mathscr{C}$ if and only if $1 + x^J$ is a divisor of $h(x)$. Indeed, if the last condition holds, then $a_J(x)h(x) = 0 \pmod{x^n + 1}$. Conversely, for some polynomial $z(x)$ we have $a_J(x) = z(x)g(x)$, but also $a_J(x) = g(x)h(x)/(1 + x^J)$; thus by uniqueness $z(x) = h(x)/(1 + x^J)$. This implies the following result.

Theorem 2: Let $\mathscr{C}$ be cyclic with composite blocklength n, let $J \geq 2$ be a divisor of n, and let $a_J(x)$ be given by (14). The set $\{x^j a_J(x)\colon j = 0, 1, \cdots, J - 1\}$ is a zero-concurring subset of $\mathscr{C}$ iff $1 + x^J$ is a divisor of $h(x)$. If $J = \lfloor n/d \rfloor$, where d is either the minimum distance or designed distance of $\mathscr{C}$, then there is no subset of zero-concurring codewords in $\mathscr{C}$ with more than J elements.

The case $J = 1$ is intentionally excluded from Theorem 2 because by (10) the corresponding gain is negligible, a fact congruous with the observation that every code contains a set of one zero-concurring codeword. If n is odd, then the condition $h(x) = 0 (\bmod 1 + x^J)$ is equivalent to the requirement that $1 + x^J$ and $g(x)$ have no common zeros. Applying Theorem 2 to the case of $\mathscr{C} =$ RM*(r, m), all we need [7, p. 383] to do is to check that for a satisfactorily large divisor J of n $w_2(in/J) \geq m - r$ for all $i = 1, 3, \cdots, J$. Thus, for example, we find in RM*(3, 6) and RM*(4, 6), respectively, bound-meeting subsets of 9 and 21 zero-concurring codewords, compared with only 8 and 16 obtained by puncturing. Application of Theorem 2 to $\mathscr{C} =$ BCH(n, d), the primitive BCH code of length $n = 2^m - 1$ and designed distance d, consists only of checking that no cyclic shift of the m bit radix-2 expansion of in/J for all $i = 1, 3, \cdots, J$ coincides with the radix-2 expansion of any positive integer less than d. For example, we find in BCH(63, 9) with $k = 39$ a bound-meeting subset of $J = 7$ zero-concurring codewords, whereas for BCH(15, 5) with $k = 7$ the corresponding, also bound-meeting, number is $J = 3$, which implies a remarkably low computational complexity. In the nonprimitive (21, 12) BCH code, a set of $J = 3$ zero-concurring codewords is found by applying Theorem 2, and it was verified by computer search that no zero-concurring subset exists that meets the bound 4 (see also Table I).

It is noteworthy that a set of zero-concurring codewords derived either by Theorem 2 or (for RM(r, m)) by the proof of Theorem 1 is suited to increase the efficiency of Algorithm B through the use of direct Hadamard transform iff $n/J \leq k - J$; such J exists iff $k^2 \geq 4n$, and the corresponding complexity is given by

$$N_B' = n2^{n/J-1} + J2^{k-J}.$$

Theorem 2 is not applicable for a cyclic code with prime n. A possible approach in this case is to select an appropriate set of integers $\{j\}$ such that $\{x^j g(x) (\bmod 1 + x^n)\}$ is a zero-concurring subset of $\mathscr{C}$. For example, in the Hamming code of length 31 with $\lfloor n/d \rfloor = 10$ generated by $g(x) = 1 + x^2 + x^5$ we find a set of $J = 8$ zero-concurring codewords by setting $j = 0, 1, 7, 8, 14, 15, 21, 22$. Evidently, for high-rate cyclic codes multiples of $n - k + 1$ are always eligible, but they do not necessarily constitute the best choice. Another interesting case is a code with composite but odd n and $g(x)$ divisible by $1 + x$. Here one may try the set $\{x^{2j}(1 + x)a_J(x)\colon j = 0, 1, \cdots, (J - 3)/2\}$, where $a_J(x)$ is given by (14); it is a zero-concurring subset of $\mathscr{C}$ whenever $J \geq 3$, $n = 0 (\bmod J)$, and the zeros of $(1 + x^J)/(1 + x)$ are nonzeros of the code. For example, in the symmetric (63, 26) BCH code with $d = 14$ with zeros consisting of $\{\alpha^i\colon i = 0, \pm 1, \cdots, \pm 5\}$ and the conjugates of these, where α is a primitive element of GF(64), we find a bound-meeting set of four zero-concurring codewords ($J = 9$). However, for the expurgated Hamming code of length $n = 15$, the suggested procedure yields a set of only two codewords corresponding to $J = 5$, whereas a bound-meeting set is given by $\{x^j[(1 + x^3)/(1 + x)]g(x)\colon j = 0, 5, 10\}$. A generally useful alternative for the case of $g(x) = 0 \ (\bmod 1 + x)$ and odd n is to work with $\{(1 + x^j)a_J(x)\colon j = 1, 2, \cdots, J - 1\}$, where $a_J(x)$ is given by (14); it is an *n/J-concurring* subset of $\mathscr{C}$ for a divisor J of n provided that $(1 + x^J)/(1 + x)$ and $g(x)$ have no common zeros. For example, in the abovementioned symmetric (63, 26) BCH code we find a bound-meeting set of eight seven-concurring codewords.

IV. Computational Gain by Employing Concurring Codewords

Assuming that $\mathscr{C}$ has a set of J concurring codewords arranged as rows of G^B in (6), we obtain a modified version of (7) that contains one additional term in its right side, namely,

$$(-1)^{\underline{\Sigma}_{l=0}^{J-1} s_l^B} \sum_{i \in \Omega_J} (-1)^{\langle s^T, g^T \rangle} \mu(v_i) \tag{15}$$

where $\Omega_J = \{i\colon b(\boldsymbol{g}_i^B) = 2^J - 1\}$ and $\underline{\Sigma}$ stands for summation modulo-2. Also, one of the sets $\Omega_j, 0 \le j \le J-1$, is possibly empty. By Lemma 1 this occurs iff G^B is row-equivalent to a matrix with zero-concurring rows. Define $s_J^B = \underline{\Sigma}_{l=0}^{J-1} s_l^B$ and $\boldsymbol{s}^{BE} = (\boldsymbol{s}^B \;\; s_J^B)$. Then we have (7) with $J-1$ and s_j^B replaced by J and s_j^{BE}, respectively. Consequently, instead of (8) we write

$$\boldsymbol{Y} = \max_{\substack{\boldsymbol{s}^{BE} \in GF(2)^{J+1} \\ \underline{\Sigma}_j^J s_j^{BE} = 0}} \left\{ \boldsymbol{U}^{(\infty)} + \sum_{j=0}^{J} (-1)^{s_j^{BE}} \boldsymbol{U}^{(j)} \right\}. \tag{16}$$

Clearly, the difference between the zero-concurring and the concurring cases is essentially embodied in the constraint $\underline{\Sigma}_{j=0}^J s_j^{BE} = 0$ that appears in (16). Denote

$$\boldsymbol{V} = \boldsymbol{U}^{(\infty)} + \sum_{j=0}^{J} |\boldsymbol{U}^{(j)}| \tag{17}$$

and remove for a moment the constraint. Then the decoding may proceed as outlined in Algorithm B by locating first the largest component of $\boldsymbol{V}$, say the ith; if $\prod_{j=0}^J U_i^{(j)} \ge 0$, then the constraint is also satisfied. On the other hand, if $\prod_{j=0}^J U_i^{(j)} < 0$, then to comply with the constraint, we either keep i and change a single bit of the previously obtained $\boldsymbol{s}^{BE}$, one with location l such that $|U_i^{(l)}|$ is minimal among $\{|U_i^{(j)}|\colon j = 0,1,\cdots,J\}$, or perhaps select some other component of $\boldsymbol{V}$. Accordingly, denote

$$W_i = V_i - 2 \min_{j=0,1,\cdots,J} |U_i^{(j)}| \tag{18}$$

for $i = 0,1,\cdots,2^{k-J}-1$ and let

$$\boldsymbol{X} = (X_0, X_1, \cdots, X_{2^{k-J}-1}) \tag{19}$$

where

$$X_i = \begin{cases} V_i, & \prod_{j=0}^{J} U_i^{(j)} \ge 0 \\ W_i, & \text{otherwise.} \end{cases} \tag{20}$$

Then $\boldsymbol{Y} = \boldsymbol{X}$. This proves the following optimal decoding rule.

Algorithm C:

a) Locate the largest component of $\boldsymbol{X}$ given by (20), say the ith.

b) Take for s^T the $k-J$ bit radix-2 expansion of i and set $s_j^B = 0$ if $U_i^{(j)} \ge 0$; otherwise, set $s_j^B = 1$.

c) Check whether $X_i = W_i$. In case of a positive answer, change a single bit of $\boldsymbol{s}^B$ at a location l such that $|U_i^{(l)}| = \min\{|U_i^{(j)}|\colon j = 0,1,\cdots,J\}$, provided such l exists (it may happen that $|U_i^{(J)}| < |U_i^{(j)}|$ for all $j = 0,1,\cdots,J-1$).

If G^B in (6) is row-equivalent to a matrix with zero-concurring rows, then $X_i = V_i$ for all i and Algorithm C reduces to Algorithm B. Excluding that singular case from the discussion, the number of additions required for calculating $\boldsymbol{V}$ is $(J+2)(k-J)2^{k-J} + (J+1)2^{k-J}$, and not more than $(J+2)2^{k-J}$ additions are needed for obtaining $\{W_i\colon i = 0,1,\cdots,2^{k-J}-1\}$ (actually half of this number is enough on the average since not all W_i are used). To check the conditions in (20) we perform for each i a logic AND operation on one word of length $J+1$ that contains the most significant bit in twos-complement representation of $U_i^{(j)}$ for $j = 0,1,\cdots,J$. The total addition-equivalent computational complexity (for the case $\Omega_\infty \ne \varnothing$) is thus

$$N_C = [k(J+2) - J^2 + 4]2^{k-J}. \tag{21}$$

If $n_j \le k - J$ for all $j \in \{0,1,\cdots,J\}$ and $n_\infty < k - J$, then the reduced complexity N_C' of Algorithm C, which corresponds to calculation of $\boldsymbol{V}$ by direct Hadamard transform, is given by

$$N_C' = n_\infty 2^{n_\infty} + \sum_{j=0}^{J} n_j 2^{n_j - 1} + (2J+4)2^{k-J}. \tag{22}$$

A meaningful comparison of N_B and N_B' with N_C, N_C' has to be performed for some fixed code rather than for some J. In certain codes it may be easier to find a large set of concurring codewords than a zero-concurring subset of the same size (expurgated codes and symmetric BCH codes, mentioned in Section III are examples). Moreover, many codes contain considerably more concurring than zero-concurring codewords. A comparison of the efficiency of all the algorithms presented heretofore, as well as Wolf's method [11], is exhibited for a few codes in the following table. Values of J and n_j for computation of N_B, N_B', N_C, and N_C' were obtained from Table I. For all but the (15, 10) code, modified versions of (10) and (21), with the right side of each reduced by $(k-J)2^{k-J}$, were used. Based on the structure of the trellis and the number [11, p. 78] of its nodes, evaluation of all node values in Wolf's method requires

$$N_W = (2k - n + 2)2^{n-k+1} - 6$$

additions, except in case of the (15, 7) code where the expurgated trellis is not fully stretched.

Code	n	k	d	N_A	N_B	N_B'	N_C	N_C'	N_W
Golay	23	12	7	49152	15360	4032	7168	1964	12282
Hamming	15	11	3	22528	2240	380	800	317	282
Expurgated Hamming	15	10	4	10240	3968	504	768	286	442
BCH	15	7	5	896	240	288	240	288	634
BCH	21	12	5	49152	15360	2880	7168	1924	5114

The complexity of maximization, excluded from consideration up to here, is given for the algorithms developed earlier by $2^{k-J} - 1$ (with $J = 0$ for Algorithm A); this amounts to less than ten percent of N_C' and, for all but the (15, 10) code, less than 20 percent of N_B'. In Wolf's method partial maximizations, that is, selection of a survivor at each node, require altogether 127 addition-equivalent operations for the (15, 7) code (i.e., 20 percent of N_W) and $(2k - n + 1)2^{n-k} - 1$ comparisons, which represent an increase of ~ 40 percent in complexity, for the other codes.

Concurring sets were usually explored in the context of majority logic decoding. The following result is evident in view of (21).

Theorem 3: Suppose that $\mathscr{C}^\perp$, the dual code of $\mathscr{C}$, can correct $e = \lfloor J/2 \rfloor$ errors by (possibly multiple-step) majority logic decoding using J parity check equations. Then $\mathscr{C}$ can be optimally decoded by FHT with complexity of the order $O(2^{k-J})$.

Theorem 4: The number J_1 of one-concurring and the number J_2 of concurring codewords in $\mathscr{C}$ with minimum distance d satisfy

$$J_1 \le \left\lfloor \frac{n-1}{d-1} \right\rfloor \qquad J_2 \le \left\lfloor \frac{2n}{d} \right\rfloor - 1.$$

For a proof see [7, pp. 390–393]. The bounds presented so far are related as follows:

$$\frac{n}{d} \le \frac{n-1}{d-1} \le \frac{2n}{d} - 1.$$

Because $\mathrm{RM}(r, m) = \mathrm{RM}(m - r - 1, m)^\perp$, provided that $r < m$ [7, p. 375], we conclude by Theorem 3 and [7, p. 393] that $J_2 \ge 2^{r+1} - 1$ in $\mathrm{RM}(r, m)$. Thus by Theorem 4 the following result is implied.

Theorem 5: In $\mathrm{RM}(r, m)$ there exists a set of $J_2 = 2n/d - 1 = 2^{r+1} - 1$ concurring codewords.

By puncturing the set of concurring codewords introduced in Theorem 5, we obtain, provided that $r < m$, a set of $2^{r+1} - 1$ concurring codewords for $\mathrm{RM}^*(r, m)$. Since for the punctured code

$$\left(\frac{2n}{d} - 1\right) - (2^{r+1} - 1) = \frac{2(2^r - 1)}{2^{m-r} - 1},$$

it follows that $\lfloor 2n/d - 1 \rfloor = 2^{r+1} - 1$ whenever $r \le (m - 1)/2$. Because $\mathrm{RM}^*(r, m) \subset \mathrm{BCH}(2^m - 1, 2^{m-r} - 1)$ we also have a set of $2(n + 1)/(d + 1) - 1$ concurring codewords for $\mathrm{BCH}(n, d)$ with $n = 2^m - 1$ and designed distance $d = 2^{m-r} - 1$, and this set meets the bound $\lfloor 2n/d - 1 \rfloor$ if $n < d(d + 3)/2$. Similarly, because $\mathrm{RM}(r, m)$ is included in the extended BCH code with $n = 2^m$ and $d = 2^{m-r} - 1$, it follows that the latter code has a set of $2n/(d + 1) - 1$ concurring codewords which is bound-meeting if $n < d(d + 1)/2$.

An iterative version of Algorithm C, owning increased computational gain in case of high signal-to-noise ratio, is illustrated by a flowchart in Fig. 3. The idea is to economize in the computation of $\{W_i\}$ by replacing step a) by an iterative procedure.

The iterative version has a pronounced advantage when the feedback path is used only a few times. Indeed, the number of additions required for calculating V is the same as in Algorithm C, and the number of addition-equivalent operations needed for each of the first few iterations is approximately $2^{k-J} + J + 3 \approx 2^{k-J}$ (actually the number decreases by one with each iteration). Altogether there are about

$$N_C'' = [(J + 2)(k - J) + J + 1 + N_{it}]2^{k-J}$$

operations, where N_{it} is the number of iterations. Upon comparing N_C'' with N_C, we conclude that the iterative

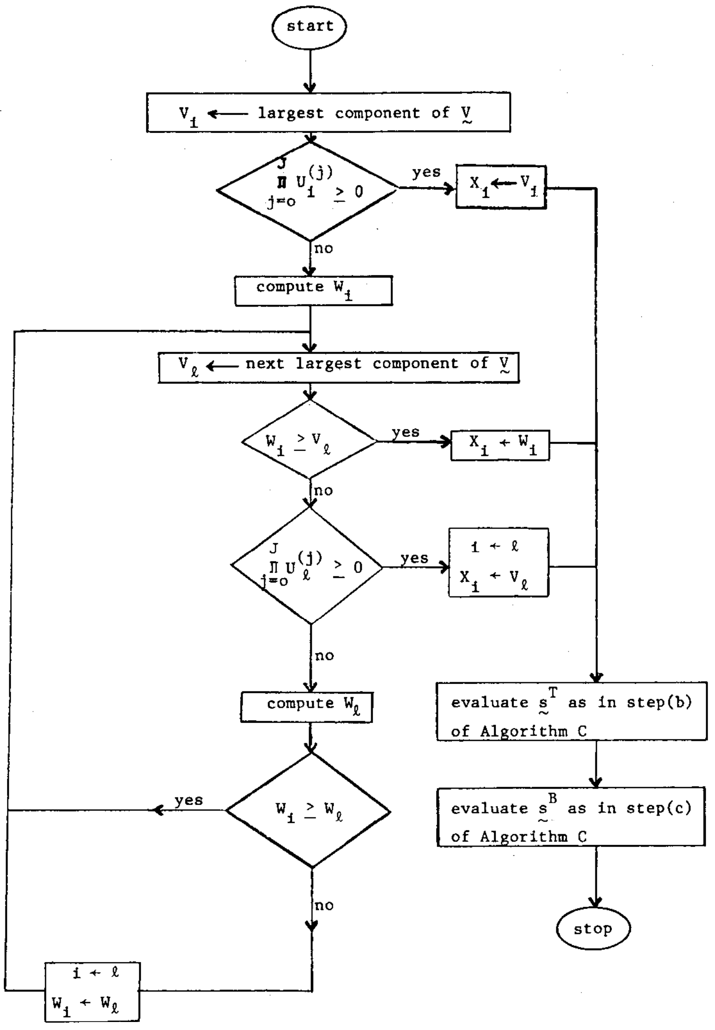
start
$V_i \longleftarrow$ largest component of $\underset{\sim}{V}$
$\prod_{j=0}^{J} U_i^{(j)} \ge 0$
yes
$X_i \longleftarrow V_i$
no
compute W_i
$V_\ell \longleftarrow$ next largest component of $\underset{\sim}{V}$
$W_i \ge V_\ell$
yes
$X_i \leftarrow W_i$
no
$\prod_{j=0}^{J} U_\ell^{(j)} \ge 0$
yes
$i \leftarrow \ell$
$X_i \leftarrow V_\ell$
no
compute W_ℓ
evaluate $\underset{\sim}{s}^T$ as in step(b) of Algorithm C
$W_i \ge W_\ell$
yes
evaluate $\underset{\sim}{s}^B$ as in step(c) of Algorithm C
no
stop
$i \leftarrow \ell$
$W_i \leftarrow W_\ell$

Fig. 3. Iterative version of Algorithm C.

algorithm is preferable whenever $N_{it} < J + 3$; this demands high signal-to-noise ratio. On the other hand, $N_C'' = O(2^{2(k-J)})$ in the worst case, that is, when decoding ends at iteration $N_{it} = 2^{k-J}$, implying that in case of a very noisy channel Algorithm C is faster.

An algorithm that suits a relaxed type of concurrence is obtainable by slightly modifying the derivation of Algorithm C.

Definition 2: A set of J independent vectors in $\mathrm{GF}(2^n)$ is partially concurring if

1) in each of some $L \ge 1$ positions at least one of the J vectors have component one and the rest (if any) have component zero, and
2) in all other positions at most one of the J vectors has component one.

A set of $J = 2$ partially concurring vectors that is not (strictly) concurring is necessarily zero-concurring. Also, by modifying the proof of the bound on J_2 in Theorem 4, it may be shown that the number J_3 of partially concurring codewords satisfies

$$J_3 \le \left\lfloor \frac{2n}{d} \right\rfloor - 1.$$

Nevertheless, $J_3 \geq J_2$ and in some codes $J_3 > J_2$. A proof similar to that of Lemma 1 yields the following result.

Lemma 2: A $J \times n$ matrix K over GF(2) with J partially concurring rows is not row-equivalent to any matrix with all its rows zero-concurring iff $D_K = J + 1$.

Assuming that $\mathscr{C}$ has a set of J partially concurring codewords, we again have (7) with one more term similar to that given by (15) added to its right side, but with Ω_J appropriately defined and $\Sigma_{l=0}^{J-1} s_l^B$ replaced by a partial sum of the bits of s^B, say $\Sigma_{l \in \mathscr{P}} s_l^B$. Denoting $s_J^B = \Sigma_{l \in \mathscr{P}} s_l^B$ and $s^{BE} = (s^B \; s_J)$, this leads to (16) with the modified constraint $\Sigma_{j \in \mathscr{F}} s_j^{BE} = 0$, where $\mathscr{F} = \mathscr{P} \cup \{J\}$. Subsequently, W_i and X_i are defined as before but with $\{0, 1, \cdots, J\}$ replaced by $\mathscr{F}$. The resulting decoding algorithm is therefore unchanged except for the replacement of $\{0, 1, \cdots, J\}$ by $\mathscr{F}$ in step c). The computational complexity in this case is upperbounded by the right side of (21) because $\{W_i\}$ and $\{X_i\}$ are more easily obtained if $\mathscr{F}$ contains less than $J + 1$ numbers. It is also possible to develop an iterative algorithm for the partially concurring case.

V. An Algorithm with Reduced Memory Requirement

Aside from the computational gains, Algorithms B and C incorporate a relaxed memory requirement owing to the decreased dimension of the Hadamard transformation. We describe here a more effective approach for reaching the latter goal. Consider an arbitrary partitioning (6) of G where G^T is a $Q \times n$ matrix and apply the subsequently introduced notation for rewriting (2) as follows:

$$M(s) = \sum_{l=0}^{n-1} (-1)^{\langle s^T, g_l^T \rangle} (-1)^{\langle s^B, g_l^B \rangle} \mu(v_l). \qquad (23)$$

Let

$$u_j^{(i)} = \begin{cases} \sum_{l \in A_j} (-1)^{\langle s^B, g_l^B \rangle} \mu(v_l), & A_j \neq \varnothing \\ 0, & A_j = \varnothing \end{cases} \qquad (24)$$

where $A_j = \{l\colon\ b(g_l^T) = j\}$ for $j = 0, 1, \cdots, 2^Q - 1$ and $i = b(s^B)$. Thus

$$M(s) = \sum_{j=0}^{2^Q - 1} (-1)^{\langle s^T, g_j^{T*} \rangle} u_j^{(i)} \qquad (25)$$

where $g_j^{T*} \in \mathrm{GF}(2)^Q$ is defined by $j = b(g_j^{T*})$ for $j = 0, 1, \cdots, 2^Q - 1$. Denote $u^{(i)} = (u_0^{(i)} \; u_1^{(i)} \; \cdots \; u_{2^Q-1}^{(i)})$ and

$$U^{(i)} = u^{(i)} H_Q, \qquad i = 0, 1, \cdots, 2^{k-Q} - 1. \qquad (26)$$

Evidently, the maximum of $M(s)$ given by (25) with respect to all $s \in \mathrm{GF}(2)^k$ is equal to the maximum among the largest components of all $U^{(i)}$ defined by (26). This implies the following optimal soft decision decoding procedure.

Algorithm D:

a) Locate the largest component of $U^{(i)}$ given by (26) for each $i = 0, 1, \cdots, 2^{k-Q} - 1$.

b) Identify the index corresponding to the largest of all numbers obtained in step a), say i; then denote by j the location of the largest component of $U^{(i)}$.

c) Take for s^T the Q-bit radix-2 expansion of j. For s^B take the $k - Q$ bit radix-2 expansion of i.

Except for the computations prescribed by (24) and the maximizations (with total negligible complexity equaling that of the single overall maximization in previous decoding procedures), Algorithm D performed by FHT requires $2^{k-Q} \cdot Q2^Q = Q2^k$ additions. However, in contrast to previous cases the computation of the modified labels $U_j^{(i)}$ may be quite cumbersome unless proper care is taken. If G^T is selected so that its columns are pairwise-different, then, with (24) entailing merely sign changes of negligible complexity, a computational gain of nearly k/Q relative to Algorithm A results. However, compliance of G^T with the foregoing condition confines Q to sufficiently large values; certainly $Q < \log_2 n$ is prohibited. An important feature of Algorithm D is the significant memory reduction achievable by excecuting the Hadamard transforms (26) sequentially; this consideration limits Q to reasonably small values. As a compromise one may take $Q \approx \log_2 n$, usually enabling the choice of G^T that has only a few columns listed in more than one location, though for the purpose of hardware realization it seems to be preferable to provide more regularity by setting Q large enough such that no additions occur in the precomputation of modified labels.

Consider $\mathscr{C} = \mathrm{RM}(r, m)$, and take $Q = m + 1$. Then $Q > m = \log_2 n$, and indeed a submatrix G^T exists, the generator of RM(1, m), whose columns are all the 2^m distinct m-tuples with a 1-bit annexed at a fixed location [7, p. 373]. Therefore, RM(r, m) is decodable either by operating 2^{k-m-1} RM(1, m) decoders (e.g., Green machines) in parallel or by performing RM(1, m) decoding 2^{k-m-1} times by a single apparatus, followed by steps b) and c). The sequential processing accommodates a memory reduction by a factor of approximately $2^k/2^{m+1} = 2^{k-m-1}$ and also a computational gain of $k/(m+1)$ relative to Algorithm A. By taking advantage of the row-equivalence of G^T to a matrix with two zero-concurring rows (implied by Theorem 1, also obvious from the presence of the all-one row), the memory and computational gains become nearly 2^{k-m+1} and $4k/(3m-1)$, respectively. Further improvement is obtained by taking into account the existence of three concurring codewords (Theorem 5) in RM(1, m), yielding reduction of memory requirement by a factor close to 2^{k-m+2} and computational gain of $9k/(5m)$. In contrast, by adopting G^T whose columns are all the 2^m distinct m-tuples, the resulting numbers are 2^{k-m} and k/m, respectively, and no further improvement is evident.

As illustrated earlier, combination of Algorithm D with Algorithm B or C may yield fruitful results. It is also noteworthy that a column of G^T that appears in several locations but always above a zero column of G^B induces only a slight computational complexity, because for the corresponding value of j the labels $u_j^{(i)}$ are all equal, irrespective of the value of i.

Acknowledgment

The authors are indebted to Nathan Zimerman and Aner Shalev for their contribution to the development of Algorithm A, Amir Dembo for several helpful discussions, and a reviewer for constructive comments. Jakov Snyders wishes to acknowledge with thanks the hospitality of McGill University, where part of this paper was written.

References

[1] R. R. Green, "A serial orthogonal decoder," *JPL Space Program Summary*, vol. IV, no. 37-39, pp. 247-253, 1966.
[2] ——, "Analysis of a serial orthogonal decoder," *JPL Space Program Summary*, vol. III, no. 37-53, pp. 185-187, 1968.
[3] J. O. Duffy, "Detailed design of a Reed-Muller (32, 6) block encoder," *JPL Space Program Summary*, vol. III, no. 37-47, pp. 263-267, 1967.
[4] E. C. Posner, "Combinatorial structure in planetary reconaissance," in *Error Correcting Codes*, H. B. Mann, Ed. New York: Wiley, 1969, pp. 15-46.
[5] C. K. Rushforth, "Fast Fourier-Hadamard decoding of orthogonal codes," *Inform. Contr.*, vol. 15, pp. 33-37, 1969.
[6] H. J. Manley, H. F. Mattson Jr., and J. R. Schatz, "Some applications of Good's theorem," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 475-476, 1980.
[7] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.
[8] M. Cohn and A. Lempel, "On fast *M*-sequence transforms," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 135-137, 1977.
[9] V. V. Losev and V. D. Dvornikov, "Fast Walsh decoding of maximum length codes," *Radio Tek. El.*, vol. 24, pp. 630-632, 1979.
[10] G. C. Clark, Jr., and R. C. Davis, "A decoding algorithm for group codes and convolution codes based on the fast Fourier-Hadamard transform," presented at the IEEE 1969 Int. Symp. Information Theory, *Ellenville, NY.*
[11] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 76-80, 1978.
[12] C. R. P. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 514-517, 1976.
[13] L. D. Rudolph, C. R. P. Hartmann, T. Y. Hwang, and N. Q. Duc, "Algebraic analog decoding of linear binary codes," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 430-440, 1979.
[14] G. R. Redinbo, "Optimum soft decision decoding with graceful degradation," *Inform. Contr.*, vol. 41, pp. 165-185, 1979.
[15] ——, "Optimal symbol-by-symbol mean-square error channel coding," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 387-405, 1979.
[16] K. R. Matis and J. W. Modestino, "Reduced-search soft-decision trellis decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 349-355, 1982.
[17] E. R. Berlekamp, "The construction of fast, high-rate, soft decision block decoders," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 372-377, 1983.
[18] H. Tanaka and K. Kakigahara, "Simplified correlation decoding by selecting possible codewords using erasure information," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 743-748, 1983.
[19] G. Seroussi and A. Lempel, "Maximum likelihood decoding of certain Reed-Muller codes," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 448-450,1983.
[20] Y. Be'ery and J. Snyders, "A recursive Hadamard transform optimal soft decision decoding algorithm," to be published.
[21] D. F. Elliot and K. R. Rao, *Fast Transforms, Algorithms, Analyses, Applications*. New York: Academic, 1982, pp. 313-317.
[22] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983, p. 392.
[23] J. L. Massey, D. J. Costello, Jr., and J. Justesen, "Polynomial weights and code construction," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 101-110, 1973.
[24] L. B. Levitin and C. R. P. Hartmann, "A new approach to the general minimum distance decoding problem: the zero-neighbors algorithm," *IEEE Trans. Inform. Theory*, vol. IT-31, no. 3, pp. 378-384, May 1985.