Maximum Likelihood Decoding of the Leech Lattice

Alexander Vardy and Yair Be'ery

Abstract—**A new algorithm for maximum likelihood decoding of the Leech lattice is presented. The algorithm involves projecting the points of the Leech lattice directly onto the codewords of the (6,3,4) quaternary code—the hexacode. Projection on the hexacode induces a partition of the Leech lattice into four cosets of a certain sublattice Q_{24}. Such a partition into cosets enables maximum likelihood decoding of the Leech lattice with 3595 real operations in the worst case and only 2955 operations on the average. This is about half the worst case and the average complexity of the best previously known algorithm [3]. Moreover the proposed decoder is far simpler, both conceptually and structurally, than the state of the art decoder of [3].**

Index Terms— **Leech lattice, maximum likelihood decoding, Leech quarter-lattice, hexacode, binary Golay code, lattice codes.**

I. Introduction

Recently maximum likelihood decoding of lattices has gained a renewed interest [1]–[5], [9], [13], [14], [16], [18], [20], partly due to the fundamental work of Forney [13], [14] which showed that the vast majority of known good coded–modulation schemes may be regarded as coset codes based on lattice partitions. The *Leech lattice* Λ_{24}, which was first introduced in [19], is one of the most interesting and well studied lattices [10]. It is the unique laminated lattice in 24 dimensions. It is also an extremal (Type II [10, p. 194]) self–dual even unimodular lattice. Partitioning the space into the Voronoi regions of Λ_{24} produces the best known 24-dimensional quantizer, assuming high resolution source distribution. Taking the points of Λ_{24} to be the origins of equal nonoverlapping spheres yields a sphere packing of density 0.00193.., which is conjectured [10] to be the densest possible packing in 24 dimensions. When used as a code for a band-limited AWGN channel, the Leech lattice offers a nominal coding gain of 6.02 dB, while the effective coding gain is about 4 dB.

The problem of maximum likelihood decoding of the Leech lattice was intensively investigated in the last few years. New algorithms were introduced, more efficient than those previously known [8], [16]. In fact, to quote Forney [15], "the Leech lattice has become a kind of benchmark for decoding algorithms." Thus Conway and Sloane [9] devised a decoder that requires approximately 56 000 operations. Forney [4] used a 256-state trellis diagram to decode the Leech lattice with 15167 operations. In [18], Lang and Longstaff presented an algorithm that requires less than 10 000 operations. Yet the "current record" in the decoding of the Leech lattice belongs to Be'ery, Shahar, and Snyders [3] and it stands at 7929 operations using the algorithm described in [3], or 6129 operations using a more advanced algorithm. In the sequel we claim a more efficient decoder that requires only 3595 operations in the worst case and 2955 operations on the average.

To be precise, we note that in compliance with the convention established in [3], [9], [14], [18] the complexity of decoding is measured herein in terms of the total number of real arithmetic operations such as addition, subtraction, or comparison of two real values. Following [3], [9], [14], [18] such operations as memory addressing, negation, and taking the absolute value are neglected, though none of these is allowed to be excessive. As in [3], [14] we also assume that those values that can be computed directly from the geometry of the signal constellation are available. A special effort has been made to evaluate all the algorithms in a uniform manner. The complexity figures cited above follow those given in [3]. The study of [2] strongly supports the notion that these figures give approximate prediction of the relative VLSI complexity.

The close interrelationship between the Leech lattice and the (24, 12, 8) *binary Golay code* C_{24} is well known. Be'ery *et al.* [3] used the ideas developed in [24] for the soft decision decoding of C_{24} to design an efficient Leech lattice decoder. There also exists a close relationship between the binary Golay code and the (6, 3, 4) code over GF(4)—the hexacode. The *hexacode* H_6 was first studied by Assmus and Mattson [27] and Shaughnessy [28]. It was later employed by Conway [6], as a successor to the Miracle Octad Generator of Curtis [11], for the purpose of identifying the octads of the Steiner system $S(5, 8, 24)$. Subsequently, Conway [10, ch. 11] and Pless [21] have shown how the hexacode may be used for the hard decision decoding of C_{24}. In [26] we have employed the hexacode for the soft decision decoding of the binary Golay code, by means of projecting the 4096 codewords of C_{24} onto the 64 codewords of H_6. In the next section, we shall bridge the gap between the approaches of [3] and [26] and define the Leech lattice directly in terms of the hexacode. This will enable us to develop in Section III a Leech lattice decoder, which is not only computationally more efficient but also structurally simpler that the decoder of [3]. Finally, in Section IV we present a block diagram of the proposed decoder.

II. The Leech Lattice, the Binary Golay Code and the Hexacode

We shall generally use the notation of [3] and [26]. To enable the reader to follow, we now briefly restate some of this notation.

The hexacode H_6 is the unique (6, 3, 4) linear code over GF(4), or equivalently $(\mathbf{Z}_2)^2$. Following [21] we take

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \bar{\omega} & \omega \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \omega & \bar{\omega} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \tag{1}$$

as a generator matrix of H_6, where $\mathbf{0}$, $\mathbf{1}$, ω, $\bar{\omega}$ (hereafter called *characters*) are the elements of GF(4). The binary Golay code C_{24} is the unique (24, 12, 8) code generated by (cf. [14]):

$$\begin{bmatrix} 11111111 & 00000000 & 00000000 \\ 00000000 & 11111111 & 00000000 \\ 00000000 & 00000000 & 11111111 \\ 11110000 & 11110000 & 00000000 \\ 00000000 & 11110000 & 11110000 \\ 11001100 & 11001100 & 00000000 \\ 10101010 & 10101010 & 00000000 \\ 00000000 & 11001100 & 11001100 \\ 00000000 & 10101010 & 10101010 \\ 10011100 & 10011100 & 10011100 \\ 11111100 & 11111100 & 01010110 \\ 01111000 & 01111000 & 01111000 \end{bmatrix} \tag{2}$$

We shall represent binary vectors of length 24 by 4×6 arrays with

Manuscript received September 25, 1991; revised June 5, 1992.

A. Vardy was with the IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120. He is now with the Coordinated Science Laboratory, 1308 W. Main Street, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

Y. Be'ery is with the Department of Electrical Engineering-System, Tel-Aviv University, Ramat-Aviv, 69978, Israel.

IEEE Log Number 9209191.

$$\begin{array}{cccc} A_{000} & B_{000} & A_{110} & B_{110} \\ B_{100} & A_{010} & B_{010} & A_{100} \\ A_{111} & B_{111} & A_{001} & B_{001} \\ B_{011} & A_{101} & B_{101} & A_{011} \end{array}$$

Fig. 1. The 16-point configuration.

even representations

1	1	$(A_{111}, A_{110})^t$	$(A_{110}, A_{111})^t$	$(A_{111}, A_{000})^t$	$(A_{110}, A_{001})^t$	$(A_{101}, A_{100})^t$	$(A_{100}, A_{101})^t$	$(A_{100}, A_{011})^t$	$(A_{101}, A_{010})^t$
1	0	$(A_{110}, A_{110})^t$	$(A_{111}, A_{111})^t$	$(A_{111}, A_{001})^t$	$(A_{110}, A_{000})^t$	$(A_{100}, A_{100})^t$	$(A_{101}, A_{101})^t$	$(A_{100}, A_{010})^t$	$(A_{101}, A_{011})^t$
0	1	$(A_{001}, A_{000})^t$	$(A_{000}, A_{001})^t$	$(A_{001}, A_{110})^t$	$(A_{000}, A_{111})^t$	$(A_{011}, A_{010})^t$	$(A_{010}, A_{011})^t$	$(A_{011}, A_{100})^t$	$(A_{010}, A_{101})^t$
0	0	$(A_{000}, A_{000})^t$	$(A_{001}, A_{001})^t$	$(A_{000}, A_{110})^t$	$(A_{001}, A_{111})^t$	$(A_{010}, A_{010})^t$	$(A_{011}, A_{011})^t$	$(A_{010}, A_{100})^t$	$(A_{011}, A_{101})^t$

h-parity	k-parity	**0**	**1**	ω	$\bar{\omega}$
0	0	$(A_{010}, A_{110})^t$	$(A_{011}, A_{111})^t$	$(A_{010}, A_{000})^t$	$(A_{011}, A_{001})^t$	$(A_{000}, A_{100})^t$	$(A_{001}, A_{101})^t$	$(A_{000}, A_{010})^t$	$(A_{001}, A_{011})^t$
0	1	$(A_{011}, A_{110})^t$	$(A_{010}, A_{111})^t$	$(A_{011}, A_{000})^t$	$(A_{010}, A_{001})^t$	$(A_{001}, A_{100})^t$	$(A_{000}, A_{101})^t$	$(A_{001}, A_{010})^t$	$(A_{000}, A_{011})^t$
1	0	$(A_{100}, A_{000})^t$	$(A_{101}, A_{001})^t$	$(A_{100}, A_{110})^t$	$(A_{101}, A_{111})^t$	$(A_{110}, A_{010})^t$	$(A_{111}, A_{011})^t$	$(A_{110}, A_{100})^t$	$(A_{111}, A_{101})^t$
1	1	$(A_{101}, A_{000})^t$	$(A_{100}, A_{001})^t$	$(A_{101}, A_{110})^t$	$(A_{100}, A_{111})^t$	$(A_{111}, A_{010})^t$	$(A_{110}, A_{011})^t$	$(A_{111}, A_{100})^t$	$(A_{110}, A_{101})^t$

odd representations

Fig. 2. The 64-type-A representations of the characters $\mathbf{0}$, $\mathbf{1}$, ω, $\bar{\omega}$.

entries from GF(2). A row or a column of such array is said to be of *odd* or *even* parity according as it contains an odd or even number of nonzeros. Any binary column 4-tuple $\underline{a} = (a_1, a_2, a_3, a_4)^t$ satisfying $(\mathbf{0}, \mathbf{1}, \omega, \bar{\omega}) \cdot \underline{a} = x$, where $\cdot$ stands for the scalar product over GF(4) and $x \in$ GF(4), is said to be an *interpretation* of the character x. Conversely, x is said to be the *projection* of $\underline{a}$. A vector $\underline{x} \in$ GF(4) is said to be the projection of $\underline{v} \in$ GF(2)^{24} if the characters of $\underline{x}$ are the projections of the 6 columns in the array corresponding to $\underline{v}$. The binary Golay code may then be defined as the set of all $\underline{c} \in$ GF(2)^{24}, such that in the corresponding 4×6 array the parity of all the columns is equal to the parity of the top row, and the projection of the array is in H_6. For a more detailed treatment see [10], [21].

Let D_2 be the two-dimensional *checkerboard lattice* consisting of all the points with integer coordinates, such that the sum of their coordinates is even. Partition D_2 into 16 subsets in a manner analogous to the Ungerboeck method [25], and arrange the labels of the 16 subsets in a configuration depicted in Fig. 1. Tiling the entire space with nonoverlapping copies of scaled and rotated version of this 16-point configuration establishes a correspondence between the labels of the 16 subsets and the points of D_2. For more details on this see [3].

Hereafter, we shall represent the points of Λ_{24} by 2×6 arrays whose entries are the points of D_2. The array will be called *type-A*, respectively *type-B*, if it contains only A_{ijk} points, respectively B_{ijk} points. The *overall k-parity* of the array is a modulo-2 sum of the k subscripts of the points that constitute the array. In the following, we shall consider only the arrays of type A. The discussion may be extended in an obvious way to include the *type-B* arrays as well. Let $(A_{i_1 j_1 k_1}, A_{i_2 j_2 k_2})$ be a 2-point sequence, where $i_1, i_2, j_1, j_2, k_1, k_2 \in$ GF(2) and $A_{i_1 j_1 k_1}$, $A_{i_2 j_2 k_2}$ are the labels corresponding to points of D_2. Thus $(A_{i_1 j_1 k_1}, A_{i_2 j_2 k_2})^t$, where superscript ^{t} denotes transposition, constitutes a column of a type-A array. This column is said to be *odd* or *even* according as the binary 4-tuple (i_1, j_1, i_2, j_2) contains an odd or even number of nonzeros. The binary index i_1 is called the *h-parity* of the column. The *k-parity* of the column is defined as $k_1 \oplus k_2$, where $\oplus$ stands for the modulo-2 summation. Hence, the overall *k-parity* of the 2×6 array is the modulo-2 sum of the individual k-parities of the columns that constitute the array. Similarly, the *overall h-parity* of the array is defined as the modulo-2 sum of the individual h-parities of the six columns. Now if the 4-tuple $(i_1, j_1, i_2, j_2)^t$ is a binary interpretation of a character $x \in$ GF(4), then the column $(A_{i_1 j_1 k_1}, A_{i_2 j_2 k_2})^t$ is said to be a *representation* of the character x. Conversely, x is said to be the *projection* of $(A_{i_1 j_1 k_1}, A_{i_2 j_2 k_2})^t$. Obviously any character $x \in$ GF(4) has exactly 16 different representations of type A (8 odd and 8 even). The 64 possible type-A representations of the four characters are listed in Fig. 2. The quaternary vector of length 6 that consists of the six projections of the columns that constitute the 2×6 array is called *projection of the array*. Given this notation the Leech lattice may be defined as follows.

Definition: The Leech lattice Λ_{24} consists of all the 2×6 arrays whose entries are points of D_2, such that each array satisfies the following conditions.

a) It is either type-A or type-B.
b) It consists either of only even columns or of only odd columns.
c) The overall k-parity of the array has to be even if the array is type-A; otherwise the overall k-parity is odd.
d) The overall h-parity of the array has to be even if the array consists of only even columns; otherwise the overall h-parity is odd.
e) The projection of the array is a codeword of H_6. □

For example, the arrays at the bottom of the next page are points of Λ_{24}. The equivalence of the foregoing definition to the formal definitions of Λ_{24} given in [10] readily follows by combining the discussions of [3] and [26].

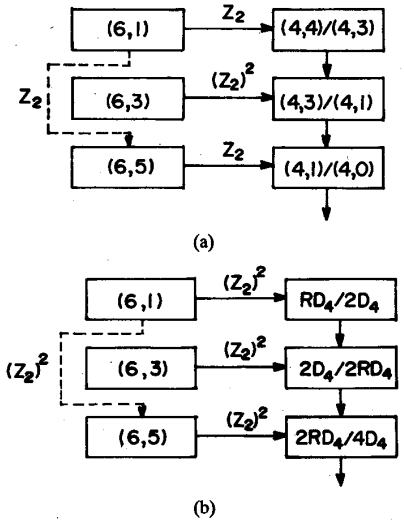

Fig. 3. Multilevel construction of (a) C_{24}, (b) Λ_{24}. Dotted lines indicate that the parity type (even or odd) employed by the (6, 5) single-parity-check encoders depends on the input to the (6, 1) repetition encoder.

It is noteworthy that taking only the type-A points in the foregoing definition produces the well-known *Leech half-lattice* H_{24}. The code formula (cf. [10, 13]) for H_{24} is

$$H_{24} = 4\mathbf{Z}^{24} + 2(24, 23, 2) + C_{24} = 2D_{24} + C_{24},$$

where D_{24} is the 24-dimensional checkerboard lattice. Taking only the type-B points produces the coset of H_{24}. Furthermore, by restricting the Leech lattice points to be type-A *and* to consist of only even columns we obtain a sublattice of H_{24}. We shall refer to this sublattice as the *Leech quarter-lattice* and denote it by Q_{24}. The code formula for Q_{24} is

$$Q_{24} = 4\mathbf{Z}^{24} + 2(24, 23, 2) + C'_{24} = 2D_{24} + C'_{24}$$

where C'_{24} is the subcode of C_{24} generated by the first 11 rows of (2). It may be also readily shown that Q_{24} is the intersection of Λ_{24} with $(2D_4)^6$, where $(2D_4)^6 = 4\mathbf{Z}^{24} + 2(4, 3, 2)^6$, and $(4, 3, 2)^6$ is the six-fold Cartesian product of the single-parity-check code of length 4.

The foregoing definitions of C_{24} and Λ_{24} may be stated more concisely in the notation of coset codes and multilevel constructions of Forney [13], [14]. The following construction of Λ_{24} also provides more insight onto the Leech quarter-lattice Q_{24}, and shows that Q_{24} is indeed a sublattice of $(2D_4)^6$. The multilevel constructions of C_{24} and Λ_{24} are illustrated in Fig. 3. The binary Golay code is constructed using the partition chain of binary codes of length four (4, 4)/(4, 3)/(4, 1)/(4, 0) where the corresponding Hamming distances are $1/2/4/\infty$. This implies representation of C_{24} as the union of two cosets of the (24, 11, 8) code C'_{24} obtained from a two-level Construction B of [10]. The resulting trellis diagram is the 64-state Cartesian product of a two-state trellis of the repetition code, a two-state single-parity-check code trellis, and a 16-state three-section trellis for H_6. Similarly, the Leech lattice may be constructed using the four-dimensional lattice partition chain $RD_4/2D_4/2RD_4/4D_4$ with Euclidean distances 4/8/16/32. This implies representation of Λ_{24} as the union of four cosets of the lattice Q_{24} that results from a two-level generalized Construction B. The corresponding trellis diagram for the Leech lattice is the 256-state Cartesian product of a four-state trellis for the repetition code over $(\mathbf{Z}_2)^2$, a four-state single-parity-check code trellis over $(\mathbf{Z}_2)^2$, and a 16-state three-section trellis for the hexacode.

Such representation of C_{24} and Λ_{24} provides insight for the development of an efficient decoding algorithm. In particular, the algorithm of [26] employs the Wagner decoding rule [22] for the single-

$\begin{bmatrix} A_{000} & A_{111} & A_{111} & A_{000} & A_{011} & A_{010} \\ A_{001} & A_{000} & A_{111} & A_{110} & A_{010} & A_{101} \end{bmatrix}$							$\begin{bmatrix} B_{100} & B_{011} & B_{011} & B_{100} & B_{101} & B_{101} \\ B_{100} & B_{101} & B_{010} & B_{011} & B_{100} & B_{011} \end{bmatrix}$					
0	1	1	0	0	0	h-parity	1	0	0	1	1	1
1	1	0	0	1	1	k-parity	0	0	1	1	1	0
0	**1**	**0**	**1**	ω	$\bar{\omega}$	projection	ω	$\bar{\omega}$	ω	$\bar{\omega}$	ω	$\bar{\omega}$
$\begin{bmatrix} A_{100} & A_{011} & A_{001} & A_{111} & A_{000} & A_{111} \\ A_{110} & A_{000} & A_{100} & A_{011} & A_{011} & A_{100} \end{bmatrix}$							$\begin{bmatrix} B_{111} & B_{001} & B_{100} & B_{101} & B_{010} & B_{010} \\ B_{100} & B_{101} & B_{110} & B_{000} & B_{110} & B_{001} \end{bmatrix}$					
1	0	0	1	0	1	h-parity	1	0	1	1	0	0
0	1	1	0	1	1	k-parity	1	0	0	1	0	1
1	**1**	ω	ω	$\bar{\omega}$	$\bar{\omega}$	projection	$\bar{\omega}$	ω	**1**	**0**	**0**	**1**

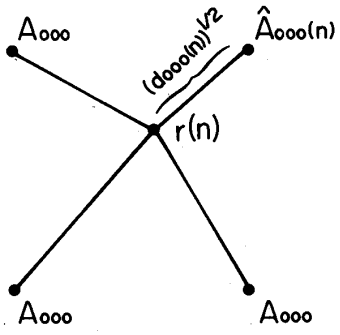

Fig. 4. Representative of the A_{000}-subset and the corresponding SED.

parity-check code over $\mathbf{Z}_2$. Thus, it is evident from the foregoing construction that finding an efficient algorithm for the decoding of Λ_{24} is mainly a matter of finding an efficient generalization of Wagner decoding for the single-parity-check code over $(\mathbf{Z}_2)^2$ rather than $\mathbf{Z}_2$.

III. The Decoding Algorithm

According to the construction of the Leech lattice outlined in the previous section, a Leech lattice decoder may be regarded to consist of four decoders for Q_{24} and its cosets. By comparing the outputs of the four Q_{24} decoders and choosing the most likely one, the final decision is reached. In the sequel only a decoder for Q_{24} is described. We assume that the channel output consists of a sequence of 12 two-dimensional symbols $\{r(n)\}_{n=1}^{12}$, where $r(n) \in \mathbf{R}^2$ for $n = 1, 2, \cdots, 12$. Given and individual received symbol $r(n)$, the decoder finds in each A_{ijk} subset a point $\hat{A}_{ijk}(n)$ which is the closest to $r(n)$. The squared Euclidean distance (SED) between $\hat{A}_{ijk}(n)$ and the received symbol $r(n)$ is denoted by $d_{ijk}(n)$. The definitions of $\hat{A}_{ijk}(n)$ and $d_{ijk}(n)$ are illustrated in Fig. 4 for the case of 64-point square QAM constellation. Note that if the channel noise is not Gaussian we can substitute a different metric in place of the squared Euclidean distance in the definition of $d_{ijk}(n)$, provided that this metric has the additive property.

Denote as in [3]

$$d_{ij}(n) = \min\{d_{ij0}(n), d_{ij1}(n)\}, \qquad \text{for } n = 1, 2, \cdots, 12, \quad (3)$$

$$\delta_{ij}(n) = d_{ij1}(n) - d_{ij0}(n), \qquad \text{for } n = 1, 2, \cdots, 12. \quad (4)$$

Computing both $\delta_{ij}(n)$ and $d_{ij}(n)$ requires a single comparison operation. Hence, the determination of the sequences $\{d_{ij}(n)\}_{n=1}^{12}$ and $\{\delta_{ij}(n)\}_{n=1}^{12}$ requires $4 \cdot 12 = 48$ comparisons for H_{24} and altogether 96 comparisons for Λ_{24}. However, all these values may be computed directly from the geometry of the constellation of symbols used for the transmission. Therefore, in order to be able to collate our algorithm with those of [3], [14], [18] we shall as in [3], [14], [18] exclude these 96 comparisons from the estimate of the total complexity of our algorithm. The 96 real values $\{d_{ij}(n)\}_{n=1}^{12}$ and $\{\delta_{ij}(n)\}_{n=1}^{12}$ are the input to our decoder. The decoding algorithm will be described in five steps.

1) Computing the Confidence Values, the Preferable Representations and the Penalties: At this step we essentially compute for each $\{r(2l-1), r(2l)\}$, $l = 1, 2, \cdots, 6$, the metrics of the 16 coset representatives for the 16 cosets of $4D_4$ in $2D_4$, and then categorize these metrics as the preferred metric and the three penalties for each of the four possible characters (the four cosets of $4RD_4$ in $2D_4$). We shall, however, introduce a notation which will be essential later in the algorithm. For each character $x \in$ GF(4) and for each of the six coordinates of H_6 we define the *confidence value in even interpretation* $\mu_l^e(x)$ and the *confidence value in odd interpretation* $\mu_l^o(x)$ as follows:

$$\begin{aligned}
\mu_l^e(\mathbf{0}) &= \min\{d_{00}(2l-1) + d_{00}(2l),\ d_{11}(2l-1) + d_{11}(2l)\}\\
\mu_l^e(\mathbf{1}) &= \min\{d_{00}(2l-1) + d_{11}(2l),\ d_{11}(2l-1) + d_{00}(2l)\}\\
\mu_l^e(\omega) &= \min\{d_{01}(2l-1) + d_{01}(2l),\ d_{10}(2l-1) + d_{10}(2l)\}\\
\mu_l^e(\overline{\omega}) &= \min\{d_{01}(2l-1) + d_{10}(2l),\ d_{10}(2l-1) + d_{01}(2l)\}\\
\\
\mu_l^o(\mathbf{0}) &= \min\{d_{10}(2l-1) + d_{00}(2l),\ d_{01}(2l-1) + d_{11}(2l)\}\\
\mu_l^o(\mathbf{1}) &= \min\{d_{01}(2l-1) + d_{00}(2l),\ d_{10}(2l-1) + d_{11}(2l)\}\\
\mu_l^o(\omega) &= \min\{d_{00}(2l-1) + d_{10}(2l),\ d_{11}(2l-1) + d_{01}(2l)\}\\
\mu_l^o(\overline{\omega}) &= \min\{d_{00}(2l-1) + d_{01}(2l),\ d_{11}(2l-1) + d_{10}(2l)\}.
\end{aligned} \quad (5)$$

Obviously the Q_{24} decoder will compute only the even interpretation confidence values $\mu_l^e(\mathbf{0})$, $\mu_l^e(\mathbf{1})$, $\mu_l^e(\omega)$, and $\mu_l^e(\overline{\omega})$, while a decoder for the coset of Q_{24} (namely the second coset of Q_{24} in H_{24}) will compute the odd interpretation confidence values $\mu_l^o(\mathbf{0})$, $\mu_l^o(\mathbf{1})$, $\mu_l^o(\omega)$, and $\mu_l^o(\overline{\omega})$. We now establish which of the eight even representations of the character $x \in$ GF(4) corresponds to its confidence value $\mu_l^e(x)$. Let $(i_1, j_1, i_2, j_2)^t$ be an even interpretation of x, and let

$$\begin{aligned}
\min\{d_{i_1j_1}(2l-1) + d_{i_2j_2}(2l),\ d_{\bar{i}_1\bar{j}_1}(2l-1) + d_{\bar{i}_2\bar{j}_2}(2l)\}\\
= d_{i_1j_1}(2l-1) + d_{i_2j_2}(2l),
\end{aligned}$$

where $\bar{a}$ denotes the binary complement of a. Define $k_1 = \text{sign}(\delta_{i_1j_1}(2l-1))$ and $k_2 = \text{sign}(\delta_{i_2j_2}(2l))$, where the sign$(\cdot)$ function is 1 whenever its argument is negative, and 0 otherwise. Then the total SED between $(\hat{A}_{i_1j_1k_1}(2l-1), \hat{A}_{i_2j_2k_2}(2l))^t$ and the pair of received symbols $\{r(2l-1), r(2l)\}$ is minimum among the eight even type-A representations of x. Therefore, $(\hat{A}_{i_1j_1k_1}(2l-1), \hat{A}_{i_2j_2k_2}(2l))^t$ is called the *preferable (even type-A) representation,* and the confidence value $\mu_l^e(x)$ is just the total SED between the preferable representation of x and $\{r(2l-1), r(2l)\}$. The h-parity and the k-parity of the preferable representation are given by, respectively, $u = i_1$ and $v = k_1 \oplus k_2$. For the sake of brevity we shall refer to the ordered duo (u, v) as the (h, k)-*parity* of the representation. Now if a Q_{24} decoder is to use in the lth coordinate a representation of x whose (h, k)-parity is (u, v), it will obviously choose the preferable representation. However, if the decoder is constrained to use a representation whose (h, k)-parity is either $(u, \overline{v})$, $(\overline{u}, v)$, or $(\overline{u}, \overline{v})$ is will not be able to employ the preferable representation. Using a representation whose (h, k)-parity differs from the (h, k)-parity of the preferable representation necessitates a penalty in the total SED (confidence value) with respect to the preferable representation. The *penalties* $\Delta_l^{u\overline{v}}(x)$, $\Delta_l^{\overline{u}v}(x)$, and $\Delta_l^{\overline{u}\,\overline{v}}(x)$ may be computed as follows:

$$\Delta_l^{u\overline{v}}(x) = \min\{|\delta_{i_1j_1}(2l-1)|,\ |\delta_{i_2j_2}(2l)|\}. \quad (6)$$

If $\text{sign}(\delta_{i_1j_1}(2l-1)) \oplus \text{sign}((\delta_{i_2j_2}(2l)) = \text{sign}(\delta_{\bar{i}_1\bar{j}_1}(2l-1)) \oplus \text{sign}((\delta_{\bar{i}_2\bar{j}_2}(2l))$, then

$$\begin{aligned}
\Delta_l^{\overline{u}v}(x) &= [d_{\bar{i}_1\bar{j}_1}(2l-1) + d_{\bar{i}_2\bar{j}_2}(2l)]\\
&\quad - [d_{i_1j_1}(2l-1) + d_{i_2j_2}(2l)], \qquad (7a)\\
\Delta_l^{\overline{u}\,\overline{v}}(x) &= \Delta_l^{\overline{u}v}(x) + \min\{|\delta_{\bar{i}_1\bar{j}_1}(2l-1)|,\ |\delta_{\bar{i}_2\bar{j}_2}(2l)|\}, \qquad (8a)
\end{aligned}$$

otherwise,

$$\begin{aligned}
\Delta_l^{\overline{u}\,\overline{v}}(x) &= [d_{\bar{i}_1\bar{j}_1}(2l-1) + d_{\bar{i}_2\bar{j}_2}(2l)]\\
&\quad - [d_{i_1j_1}(2l-1) + d_{i_2j_2}(2l)], \qquad (7b)\\
\Delta_l^{\overline{u}v}(x) &= \Delta_l^{\overline{u}\,\overline{v}}(x) + \min\{|\delta_{\bar{i}_1\bar{j}_1}(2l-1)|,\ |\delta_{\bar{i}_2\bar{j}_2}(2l)|\}. \qquad (8b)
\end{aligned}$$

Complexity: We compute 96 real values $\mu_l^e(x)$, $\Delta_l^{u\overline{v}}(x)$, $\Delta_l^{\overline{u}v}(x)$, and $\Delta_l^{\overline{u}\,\overline{v}}(x)$ for all $l = 1, 2, \cdots, 6$ and $x \in \{\mathbf{0}, \mathbf{1}, \omega, \overline{\omega}\}$. Computing $\mu_l^e(x)$ as defined in (5) requires two additions and one comparison. Note that this comparison is equivalent to the subtraction in (7). Hence both $\mu_l^e(x)$ and either $\Delta_l^{\overline{u}v}(x)$ or $\Delta_l^{\overline{u}\,\overline{v}}(x)$ are computed by means of three real operations. The computation of $\Delta_l^{u\overline{v}}(x)$ as defined in (6) requires one comparison. Finally, in order to compute the remaining penalty (either $\Delta_l^{\overline{u}\,\overline{v}}(x)$ or $\Delta_l^{\overline{u}v}(x)$) as defined in (8) we need one comparison and one addition. Thus, the computation of the four values $\mu_l^e(x)$, $\Delta_l^{u\overline{v}}(x)$, $\Delta_l^{\overline{u}v}(x)$, and $\Delta_l^{\overline{u}\,\overline{v}}(x)$ requires six real operations per character per coordinate, and altogether $6 \cdot 4 \cdot 6 = 144$ real operations.

The next step in conjunction with the fourth step of the algorithm essentially constitute a generalization of the Wagner decoding rule [22] for the (6,5,2) single-parity-check code over $(\mathbf{Z}_2)^2$.

2) Sorting the Penalties: At this step of the algorithm we sort all the 24 penalties that have the same (h, k)-parity relative to the preferable representation. That is we rearrange the elements of the following three sets

$$\begin{aligned}
&\Big\{\Delta_1^{u\overline{v}}(\mathbf{0}), \Delta_1^{u\overline{v}}(\mathbf{1}), \Delta_1^{u\overline{v}}(\omega), \Delta_1^{u\overline{v}}(\overline{\omega});\\
&\Delta_2^{u\overline{v}}(\mathbf{0}), \Delta_2^{u\overline{v}}(\mathbf{1}), \Delta_2^{u\overline{v}}(\omega), \Delta_2^{u\overline{v}}(\overline{\omega});\\
&\cdots; \Delta_6^{u\overline{v}}(\mathbf{0}), \Delta_6^{u\overline{v}}(\mathbf{1}), \Delta_6^{u\overline{v}}(\omega), \Delta_6^{u\overline{v}}(\overline{\omega})\Big\},\\
&\Big\{\Delta_1^{\overline{u}v}(\mathbf{0}), \Delta_1^{\overline{u}v}(\mathbf{1}), \Delta_1^{\overline{u}v}(\omega), \Delta_1^{\overline{u}v}(\overline{\omega});\\
&\Delta_2^{\overline{u}v}(\mathbf{0}), \Delta_2^{\overline{u}v}(\mathbf{1}), \Delta_2^{\overline{u}v}(\omega), \Delta_2^{\overline{u}v}(\overline{\omega});\\
&\cdots; \Delta_6^{\overline{u}v}(\mathbf{0}), \Delta_6^{\overline{u}v}(\mathbf{1}), \Delta_6^{\overline{u}v}(\omega), \Delta_6^{\overline{u}v}(\overline{\omega})\Big\},\\
&\Big\{\Delta_1^{\overline{u}\,\overline{v}}(\mathbf{0}), \Delta_1^{\overline{u}\,\overline{v}}(\mathbf{1}), \Delta_1^{\overline{u}\,\overline{v}}(\omega), \Delta_1^{\overline{u}\,\overline{v}}(\overline{\omega});\\
&\Delta_2^{\overline{u}\,\overline{v}}(\mathbf{0}), \Delta_2^{\overline{u}\,\overline{v}}(\mathbf{1}), \Delta_2^{\overline{u}\,\overline{v}}(\omega), \Delta_2^{\overline{u}\,\overline{v}}(\overline{\omega});\\
&\cdots; \Delta_6^{\overline{u}\,\overline{v}}(\mathbf{0}), \Delta_6^{\overline{u}\,\overline{v}}(\mathbf{1}), \Delta_6^{\overline{u}\,\overline{v}}(\omega), \Delta_6^{\overline{u}\,\overline{v}}(\overline{\omega})\Big\}
\end{aligned}$$

in a nondecreasing order.

Complexity: Let $F(N)$ be the number of comparisons required to sort N real values. Then (cf.[17]):

$$\lceil \log N! \rceil \le F(N) \le \sum_{i=1}^{N} \left\lceil \log \left[\frac{3}{4} \cdot i \right] \right\rceil. \tag{9}$$

Substituting $N = 24$ in (9) yields $80 \le F(24) \le 81$. Thus the complexity of the second step is upper bounded by $3 \cdot 81 = 243$ real operations.

3) Computing the Confidence Values of the Blocks: If $\underline{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$ is a vector in GF(4)^6 the sets $\{x_1, x_2\}$, $\{x_3, x_4\}$, and $\{x_5, x_6\}$ are said to be the *blocks* of $\underline{x}$. For each of the 4096 vectors of GF(4)^6 and for each of the three blocks we determine the *confidence value of the block* in even interpretation $S_j^e(x_1, x_2)$ and in odd interpretation $S_j^o(x_1, x_2)$, where $j = 1, 2, 3$. For the first block these confidence values are defined by

$$\begin{aligned}
S_1^e(x_1, x_2) &= \mu_1^e(x_1) + \mu_2^e(x_2),\\
S_1^o(x_1, x_2) &= \mu_1^o(x_1) + \mu_2^o(x_2),
\end{aligned} \tag{10}$$

whereas for the other two blocks the corresponding confidence values are defined similarly. Evidently, the Q_{24} decoder will compute only $S_j^e(x_1, x_2)$, while a decoder for the coset of Q_{24} will compute only $S_j^o(x_1, x_2)$.

Complexity: Computing each of the confidence values defined in (10) requires 16 operations according to the 16 possibilities of choosing $\{x_1, x_2\}$. Thus for a Q_{24} decoder the complexity of the third step is 16 operations per block and altogther $3 \cdot 16 = 48$ real operations.

4) Finding the Images of the Hexacodewords: Once the representatives $\hat{A}_{ijk}(n)$ and $\hat{B}_{ijk}(n)$ of the 16 basic subsets in the two-dimensional constellations have been chosen, each *hexacodeword* $\underline{x} \in H_6$ may be regarded as the projection of exactly 2^{18} Leech lattice points, exactly 2^{16} of these being also points of Q_{24}. The task of the Q_{24} decoder is to determine for each $\underline{x} \in H_6$ which of these 2^{16} points is the closest to the received signal $\{r(n)\}_{n=1}^{12}$. The remaining $3 \cdot 2^{16}$ Leech lattice points that project on $\underline{x}$ are accounted for in one of the three decoders for the cosets of Q_{24}. The required procedure is best explained by an example. Consider for instance the following hexacodeword $(\mathbf{0101}\omega\overline{\omega})$ and distinguish between three different cases.

Case 1: Assume that the preferable even type-*A* representations of the characters of $(\mathbf{0101}\omega\overline{\omega})$ are given by (11) at the bottom of the page. The overall (h, k)-parity of the array in (11) is $(0, 0)$. Hence the array corresponds to a point of Q_{24} that projects on $(\mathbf{0101}\omega\overline{\omega})$. This point is obviously closer to the received signal than any other Q_{24} point that projects on $(\mathbf{0101}\omega\overline{\omega})$.

Case 2: Assume that the preferable even type-*A* representations of the characters of $(\mathbf{0101}\omega\overline{\omega})$ are given by (12) at the bottom of the page. The overall (h, k)-parity of the array in (12) is $(1, 1) \neq (0, 0)$

$$\begin{array}{lcccccc}
 & \begin{bmatrix}\hat{A}_{000}(1)\\ \hat{A}_{001}(2)\end{bmatrix} & \begin{bmatrix}\hat{A}_{111}(3)\\ \hat{A}_{000}(4)\end{bmatrix} & \begin{bmatrix}\hat{A}_{111}(5)\\ \hat{A}_{111}(6)\end{bmatrix} & \begin{bmatrix}\hat{A}_{000}(7)\\ \hat{A}_{110}(8)\end{bmatrix} & \begin{bmatrix}\hat{A}_{011}(9)\\ \hat{A}_{010}(10)\end{bmatrix} & \begin{bmatrix}\hat{A}_{010}(11)\\ \hat{A}_{101}(12)\end{bmatrix}\\
h\text{-parity:} & 0 & 1 & 1 & 0 & 0 & 0\\
k\text{-parity:} & 1 & 1 & 0 & 0 & 1 & 1
\end{array} \tag{11}$$

$$\begin{array}{lcccccc}
 & \begin{bmatrix}\hat{A}_{110}(1)\\ \hat{A}_{111}(2)\end{bmatrix} & \begin{bmatrix}\hat{A}_{000}(3)\\ \hat{A}_{110}(4)\end{bmatrix} & \begin{bmatrix}\hat{A}_{001}(5)\\ \hat{A}_{000}(6)\end{bmatrix} & \begin{bmatrix}\hat{A}_{111}(7)\\ \hat{A}_{001}(8)\end{bmatrix} & \begin{bmatrix}\hat{A}_{101}(9)\\ \hat{A}_{100}(10)\end{bmatrix} & \begin{bmatrix}\hat{A}_{010}(11)\\ \hat{A}_{100}(12)\end{bmatrix}\\
h\text{-parity:} & 1 & 0 & 0 & 1 & 1 & 0\\
k\text{-parity:} & 1 & 0 & 1 & 0 & 1 & 0
\end{array} \tag{12}$$

and therefore it does not correspond to a point of Q_{24} (or to any Leech lattice point). In order to obtain a point of Q_{24}, the decoder must employ columns whose (h, k)-parity differs from the (h, k)-parity of the preferable representations. This necessitates a penalty in the total squared Euclidean distance. Let the alternative representations together with the corresponding penalties be given by the following:

$$\begin{matrix}\begin{bmatrix}\hat{A}_{111}(1)\\ \hat{A}_{111}(2)\end{bmatrix} & \begin{bmatrix}\hat{A}_{000}(3)\\ \hat{A}_{111}(4)\end{bmatrix} & \begin{bmatrix}\hat{A}_{001}(5)\\ \hat{A}_{001}(6)\end{bmatrix}\\ \Delta_1^{u\overline{v}}(\mathbf{0}) = 0.54 & \Delta_2^{u\overline{v}}(\mathbf{1}) = 0.99 & \Delta_3^{u\overline{v}}(\mathbf{0}) = 1.21\\ \begin{bmatrix}\hat{A}_{110}(7)\\ \hat{A}_{001}(8)\end{bmatrix} & \begin{bmatrix}\hat{A}_{101}(9)\\ \hat{A}_{101}(10)\end{bmatrix} & \begin{bmatrix}\hat{A}_{011}(11)\\ \hat{A}_{100}(12)\end{bmatrix};\\ \Delta_4^{u\overline{v}}(\mathbf{1}) = 0.25 & \Delta_5^{u\overline{v}}(\omega) = 1.02 & \Delta_6^{u\overline{v}}(\overline{\omega}) = 0.05\end{matrix}$$

$$\begin{matrix}\begin{bmatrix}\hat{A}_{000}(1)\\ \hat{A}_{001}(2)\end{bmatrix} & \begin{bmatrix}\hat{A}_{111}(3)\\ \hat{A}_{001}(4)\end{bmatrix} & \begin{bmatrix}\hat{A}_{110}(5)\\ \hat{A}_{111}(6)\end{bmatrix}\\ \Delta_1^{\overline{u}v}(\mathbf{0}) = 0.27 & \Delta_2^{\overline{u}v}(\mathbf{1}) = 0.18 & \Delta_3^{\overline{u}v}(\mathbf{0}) = 0.82\\ \begin{bmatrix}\hat{A}_{001}(7)\\ \hat{A}_{111}(8)\end{bmatrix} & \begin{bmatrix}\hat{A}_{010}(9)\\ \hat{A}_{011}(10)\end{bmatrix} & \begin{bmatrix}\hat{A}_{100}(11)\\ \hat{A}_{010}(12)\end{bmatrix};\\ \Delta_4^{\overline{u}v}(\mathbf{1}) = 0.32 & \Delta_5^{\overline{u}v}(\omega) = 0.45 & \Delta_6^{\overline{u}v}(\overline{\omega}) = 1.28\end{matrix}$$

$$\begin{matrix}\begin{bmatrix}\hat{A}_{000}(1)\\ \hat{A}_{000}(2)\end{bmatrix} & \begin{bmatrix}\hat{A}_{110}(3)\\ \hat{A}_{001}(4)\end{bmatrix} & \begin{bmatrix}\hat{A}_{110}(5)\\ \hat{A}_{110}(6)\end{bmatrix}\\ \Delta_1^{\overline{u}\,\overline{v}}(\mathbf{0}) = 0.42 & \Delta_2^{\overline{u}\,\overline{v}}(\mathbf{1}) = 0.95 & \Delta_3^{\overline{u}\,\overline{v}}(\mathbf{0}) = 0.67\\ \begin{bmatrix}\hat{A}_{000}(7)\\ \hat{A}_{111}(8)\end{bmatrix} & \begin{bmatrix}\hat{A}_{011}(9)\\ \hat{A}_{011}(10)\end{bmatrix} & \begin{bmatrix}\hat{A}_{100}(11)\\ \hat{A}_{011}(12)\end{bmatrix}.\\ \Delta_4^{\overline{u}\,\overline{v}}(\mathbf{1}) = 0.50 & \Delta_5^{\overline{u}\,\overline{v}}(\omega) = 0.35 & \Delta_6^{\overline{u}\,\overline{v}}(\overline{\omega}) = 0.78\end{matrix}$$

Evidently to attain a point that is closest to the received signal the minimum penalty possibility has to be selected. Recall that all the 24 penalties that share the same (h, k)-parity relative to the preferable representations have been presorted at step 2. Thus at step 4 the decoder has *a priori* information that,

$$\begin{aligned}\Delta_6^{u\overline{v}}(\overline{\omega}) \le \Delta_4^{u\overline{v}}(\mathbf{1}) \le \Delta_1^{u\overline{v}}(\mathbf{0}) \le \Delta_2^{u\overline{v}}(\mathbf{1})\\ \le \Delta_5^{u\overline{v}}(\omega) \le \Delta_3^{u\overline{v}}(0),\end{aligned}$$

$$\begin{aligned}\Delta_2^{\overline{u}v}(\mathbf{1}) \le \Delta_1^{\overline{u}v}(\mathbf{0}) \le \Delta_4^{\overline{u}v}(\mathbf{1}) \le \Delta_5^{\overline{u}v}(\omega)\\ \le \Delta_3^{\overline{u}v}(\mathbf{0}) \le \Delta_6^{\overline{u}v}(\overline{\omega}),\end{aligned}$$

$$\begin{aligned}\Delta_5^{\overline{u}\,\overline{v}}(\omega) \le \Delta_1^{\overline{u}\,\overline{v}}(\mathbf{0}) \le \Delta_4^{\overline{u}\,\overline{v}}(\mathbf{1}) \le \Delta_3^{\overline{u}\,\overline{v}}(\mathbf{0})\\ \le \Delta_6^{\overline{u}\,\overline{v}}(\overline{\omega}) \le \Delta_2^{\overline{u}\,\overline{v}}(\mathbf{1}).\end{aligned}$$

The decoder may therefore conclude that the minimum penalty required to change the overall (h, k)-parity of the array in (12) from $(1, 1)$ to $(0, 0)$ is either $\Delta_5^{\overline{u}\,\overline{v}}(\omega) = 0.35$ or $\Delta_6^{u\overline{v}}(\overline{\omega}) + \Delta_2^{\overline{u}v}(\mathbf{1}) = 0.23$. By comparing between the two possibilities and choosing the one with the lower penalty a decision is reached.

Case 3: Assume that the preferable even type-A representations of the characters of $(\mathbf{0101}\omega\overline{\omega})$ are given by (13) at the botom of the page and the alternative representations along with the corresponding penalities are

$$\begin{matrix}\begin{bmatrix}\hat{A}_{111}(1)\\ \hat{A}_{111}(2)\end{bmatrix} & \begin{bmatrix}\hat{A}_{111}(3)\\ \hat{A}_{001}(4)\end{bmatrix} & \begin{bmatrix}\hat{A}_{000}(5)\\ \hat{A}_{001}(6)\end{bmatrix}\\ \Delta_1^{u\overline{v}}(\mathbf{0}) = 0.25 & \Delta_2^{u\overline{v}}(\mathbf{1}) = 0.39 & \Delta_3^{u\overline{v}}(\mathbf{0}) = 0.55\\ \begin{bmatrix}\hat{A}_{110}(7)\\ \hat{A}_{001}(8)\end{bmatrix} & \begin{bmatrix}\hat{A}_{010}(9)\\ \hat{A}_{010}(10)\end{bmatrix} & \begin{bmatrix}\hat{A}_{011}(11)\\ \hat{A}_{100}(12)\end{bmatrix};\\ \Delta_4^{u\overline{v}}(\mathbf{1}) = 0.88 & \Delta_5^{u\overline{v}}(\omega) = 0.11 & \Delta_6^{u\overline{v}}(\overline{\omega}) = 0.76\end{matrix}$$

$$\begin{matrix}\begin{bmatrix}\hat{A}_{001}(1)\\ \hat{A}_{000}(2)\end{bmatrix} & \begin{bmatrix}\hat{A}_{001}(3)\\ \hat{A}_{110}(4)\end{bmatrix} & \begin{bmatrix}\hat{A}_{111}(5)\\ \hat{A}_{111}(6)\end{bmatrix}\\ \Delta_1^{\overline{u}v}(0) = 1.10 & \Delta_2^{\overline{u}v}(1) = 0.98 & \Delta_3^{\overline{u}v}(0) = 0.52\\ \begin{bmatrix}\hat{A}_{000}(7)\\ \hat{A}_{110}(8)\end{bmatrix} & \begin{bmatrix}\hat{A}_{100}(9)\\ \hat{A}_{101}(10)\end{bmatrix} & \begin{bmatrix}\hat{A}_{100}(11)\\ \hat{A}_{010}(12)\end{bmatrix};\\ \Delta_4^{\overline{u}v}(1) = 0.37 & \Delta_5^{\overline{u}v}(\omega) = 0.28 & \Delta_6^{\overline{u}v}(\overline{\omega}) = 0.76\end{matrix}$$

$$\begin{matrix}\begin{bmatrix}\hat{A}_{001}(1)\\ \hat{A}_{001}(2)\end{bmatrix} & \begin{bmatrix}\hat{A}_{001}(3)\\ \hat{A}_{111}(4)\end{bmatrix} & \begin{bmatrix}\hat{A}_{110}(5)\\ \hat{A}_{111}(6)\end{bmatrix}\\ \Delta_1^{\overline{u}\,\overline{v}}(\mathbf{0}) = 1.27 & \Delta_2^{\overline{u}\,\overline{v}}(\mathbf{1}) = 0.59 & \Delta_3^{\overline{u}\,\overline{v}}(\mathbf{0}) = 0.92\\ \begin{bmatrix}\hat{A}_{000}(7)\\ \hat{A}_{111}(8)\end{bmatrix} & \begin{bmatrix}\hat{A}_{101}(9)\\ \hat{A}_{101}(10)\end{bmatrix} & \begin{bmatrix}\hat{A}_{101}(11)\\ \hat{A}_{010}(12)\end{bmatrix}.\\ \Delta_4^{\overline{u}\,\overline{v}}(\mathbf{1}) = 0.85 & \Delta_5^{\overline{u}\,\overline{v}}(\omega) = 1.28 & \Delta_6^{\overline{u}\,\overline{v}}(\overline{\omega}) = 0.51\end{matrix}$$

From step 2, we have

$$\begin{aligned}\Delta_5^{u\overline{v}}(\omega) \le \Delta_1^{u\overline{v}}(\mathbf{0}) \le \Delta_2^{u\overline{v}}(\mathbf{1}) \le \Delta_3^{u\overline{v}}(0)\\ \le \Delta_6^{u\overline{v}}(\overline{\omega}) \le \Delta_4^{u\overline{v}}(\mathbf{1}),\end{aligned}$$

$$\begin{aligned}\Delta_5^{\overline{u}v}(\omega) \le \Delta_4^{\overline{u}v}(\mathbf{1}) \le \Delta_3^{\overline{u}v}(\mathbf{0}) \le \Delta_6^{\overline{u}v}(\overline{\omega})\\ \le \Delta_2^{\overline{u}v}(\mathbf{1}) \le \Delta_1^{\overline{u}v}(\mathbf{0}),\end{aligned}$$

$$\begin{aligned}\Delta_6^{\overline{u}\,\overline{v}}(\overline{\omega}) \le \Delta_2^{\overline{u}\,\overline{v}}(\mathbf{1}) \le \Delta_4^{\overline{u}\,\overline{v}}(\mathbf{1}) \le \Delta_3^{\overline{u}\,\overline{v}}(\mathbf{0})\\ \le \Delta_1^{\overline{u}\,\overline{v}}(\mathbf{0}) \le \Delta_5^{\overline{u}\,\overline{v}}(\omega).\end{aligned}$$

As in the previous case there are basically two possibilities to change the overall (h, k)-parity of the array in (13) from $(1, 1)$ to $(0, 0)$. One of these is to employ a single alternative representation whose (h, k)-parity is $(\overline{u}, \overline{v})$. The minimum penalty associated with this possibility is obviously $\Delta_6^{\overline{u}\,\overline{v}}(\overline{\omega}) = 0.51$. The other possibility is to use two alternative representations whose (h, k)-parities are $(u, \overline{v})$ and $(\overline{u}, v)$. However, the decoder cannot use the minimum penalty

$$\begin{array}{lcccccc}
 & \begin{bmatrix}\hat{A}_{110}(1) & \hat{A}_{110}(3) & \hat{A}_{000}(5) & \hat{A}_{110}(7) & \hat{A}_{010}(9) & \hat{A}_{011}(11)\\ \hat{A}_{111}(2) & \hat{A}_{001}(4) & \hat{A}_{000}(6) & \hat{A}_{000}(8) & \hat{A}_{011}(10) & \hat{A}_{101}(12)\end{bmatrix} & & & & & \\
h\text{-parity:} & 1 & 1 & 0 & 1 & 0 & 0\\
k\text{-parity:} & 1 & 1 & 0 & 0 & 1 & 0
\end{array} \tag{13}$$

$(u, \overline{v})$-parity representation and the minimum penalty $(\overline{u}, v)$-parity representation as they both fall in the same coordinate. Hence, the latter possibility involves comparing between the minimum $(\overline{u}, v)$-penalty plus the next to minimum $(\overline{u}, v)$-penalty and the minimum $(u, \overline{v})$-penalty plus the next to minimum $(u, \overline{v})$-penalty, namely between $\Delta_5^{u\overline{v}}(\omega) + \Delta_4^{\overline{u}v}(\mathbf{1}) = 0.48$ and $\Delta_5^{\overline{u}v}(\omega) + \Delta_1^{u\overline{v}}(\mathbf{0}) = 0.53$. The decision is reached by choosing the minimum among $\{\Delta_6^{\overline{u}\,\overline{v}}(\overline{\omega}), \Delta_5^{u\overline{v}}(\omega) + \Delta_4^{\overline{u}v}(\mathbf{1}), \Delta_5^{\overline{u}v}(\omega) + \Delta_1^{u\overline{v}}(\mathbf{0})\}$.

Complexity: In the first case, no operations on real values are required, the second case involves one addition and one comparison, while the third case requires two additions and two comparisons. These computations are performed for each of the 64 codewords of H_6. Thus the fourth step of our algorithm requires $64 \cdot 4 = 256$ real operations in the worst case and no operations at all in the most favorable case.

5) Computing the Metrics of the Hexacodewords and Final Minimization: This step of the algorithm is similar to the corresponding step in [26] and amounts to trellis decoding of a 16-state, three-section (three-block) trellis for the hexacode. For each $\underline{x} = (x_1, x_2, x_3, x_4, x_5, x_6) \in H_6$ we calculate the even type-A metric of $\underline{x}$, which is given by

$$M^e(\underline{x}) = S_1^e(x_1, x_2) + S_2^e(x_3, x_4) + S_3^e(x_5, x_6), \qquad (14)$$

provided that the overall (h, k)-parity of the preferable representations of the characters of $\underline{x}$ is $(0, 0)$. If the overall (h, k)-parity of the array composed of the even type-A preferable representations differs from $(0, 0)$, then one or two alternative representations must be used for the characters of $\underline{x}$ and the corresponding penalty must be added to (14). Now let $\hat{\underline{x}}$ be the codeword of H_6 that has the minimum (even type-A) metric among the 64 hexacodewords, and let $M^e(\hat{\underline{x}})$ be the squared Euclidean distance between the received signal and the point of Q_{24} that projects on $\hat{\underline{x}}$, and is the closest to the received signal (the image of $\hat{\underline{x}}$). This point of Q_{24} along with $M^e(\hat{\underline{x}})$ constitute the output of our decoder. Note that the image of $\hat{\underline{x}}$ may be reconstructed from $\hat{\underline{x}}$ by using the preferable (even type-A) representations for all but possibly one or two of the characters of $\hat{\underline{x}}$.

Complexity: As in [26] we partition H_6 into 16 disjoint sets of four codewords each, so that all the codewords in the same set share a common block. Finding the codeword with the minimum metric in each of the 16 sets requires at most 12 real operations (i.e. four additions to compute the four partial sums of two blocks, then at most four additions to account for the penalties if any, then three comparisons to find the minimum among the four sums obtained, and finally one more addition to add the confidence value of the shared block). Hence, the complexity of the fifth step is at most $16 \cdot 12 + 15 = 207$ real operations (at least 143 operations). □

The total number of real addition-equivalent operations required by the Q_{24} decoder is $144 + 243 + 48 + 256 + 207 = 898$ in the worst case, and $144 + 243 + 48 + 143 = 578$ in the most favorable case. Since the decoder for the coset of Q_{24} is essentially identical to the Q_{24} decoder, it requires exactly the same number of operations. Thus, maximum likelihood decoding of the Leech lattice is accomplished by at most $4 \cdot 898 + 3 = 3595$ real addition-equivalent operations, as compared to 55 968 operations in [9], 15 167 operations in [14], 10 000 operations in [18], and 6129 operations in [3].

We now estimate the average complexity of our algorithm. Be'ery *et al.* [3] use the following estimate of the average decoding complexity:

$$\text{average complexity} = \frac{\text{worst case complexity} + \text{most favorable case complexity}}{2}.$$

We shall employ a slightly more elaborate estimate that reflects the probabilities of the various cases. Define the event $\mathfrak{U}$ as follows:

event $\mathfrak{U}$: the overall (h, k) parity of the array composed of the preferable even type-A representations of the characters of a given hexacodeword is $(0, 0)$.

Assuming uniform distribution of the (h, k)-parities of the preferable even type-A representations, we have $P(\mathfrak{U}) = 0.25$ and $P(\overline{\mathfrak{U}}) = 0.75$, where $\overline{\mathfrak{U}}$ is the complement of $\mathfrak{U}$. Clearly event $\mathfrak{U}$ corresponds to the first case of step 4 of our algorithm. Now assume that $\overline{\mathfrak{U}}$ occurs and let the overall (h, k)-parity of the array composed of the preferable even type-A representations be $(a, b) \neq (0, 0)$. The events $\overline{\mathfrak{B}}$ and $\mathfrak{B}$ that correspond, respectively, to the second and the third cases of step 4 may be defined by

event $\mathfrak{B}$: $(a, b) = (1, 1)$ and the minimum $(u, \overline{v})$ and $(\overline{u}, v)$ penalties fall in the same coordinate; or $(a, b) = (1, 0)$ and the minimum $(u, \overline{v})$ and $(\overline{u}, \overline{v})$ penalties fall in the same coordinate; or $(a, b) = (0, 1)$ and the minimum $(\overline{u}, v)$ and $(\overline{u}, \overline{v})$ penalties fall in the same coordinate.

event $\overline{\mathfrak{B}}$: the complement of $\mathfrak{B}$.

Again assuming a completely random channel we have $P(\mathfrak{B}) = 1/6$ and $P(\overline{\mathfrak{B}}) = 5/6$. In terms of the probabilities of the events $\mathfrak{U}$, $\overline{\mathfrak{U}}$, $\mathfrak{B}$, and $\overline{\mathfrak{B}}$, the average complexity of the algorithm may be expressed as follows:

$$144 + 243 + 48 + P(\overline{\mathfrak{U}}) \cdot [P(\mathfrak{B}) \cdot 256 + P(\overline{\mathfrak{B}}) \cdot 128] + [P(\mathfrak{U}) \cdot 143 + P(\overline{\mathfrak{U}}) \cdot 207] = 738.$$

Hence, maximum likelihood decoding of the Leech lattice is accomplished with only $4 \cdot 738 + 3 = 2955$ operations on the average. Note that on a high SNR channel the probability of $\mathfrak{U}$ would be greater than 0.25 and consequently the average complexity of the algorithm would be less than 2955 operations.

IV. Conclusion and Implementation Considerations

A new algorithm for decoding Λ_{24} with about 3000 operations on the average has been presented. It is noteworthy that the proposed decoder is not only computationally more efficient but also simpler, both conceptually and structurally, than the decoder of [3]. In fact the technique proposed herein for decoding of Λ_{24} is in a sense merely repetitive soft-decision decoding of the hexacode. This approach enables us to avoid the extensive use of elaborate tables such a Tables I, II, and III of [3]. Moreover, since we decode H_6 four times (once for each (h, k)-parity) our decoder consists of four independent computation paths operating concurrently. This structure of the proposed decoder is illustrated in Fig. 5. Each of the four computation paths consists of the five decoding steps that constitute a single Q_{24} decoder. Note that every individual step is independent of its successors. This property of the decoder makes it amenable to pipelined implementation. A block diagram of the Q_{24} decoder is presented in Fig. 6.

The algorithm described herein shows how to find the point in the Leech lattice that is closest to an arbitrary point of $\mathbf{R}^{24}$. In practice, a communication system with a finite peak power constraint may utilize only a finite subset of lattice points. Such a subset is called a Leech-lattice code. In case of the conventional quadrature amplitude modulation (QAM) signaling scheme this subset is usually defined in terms of a finite two-dimensional constellation. In this case, the proposed algorithm obviously becomes a maximum

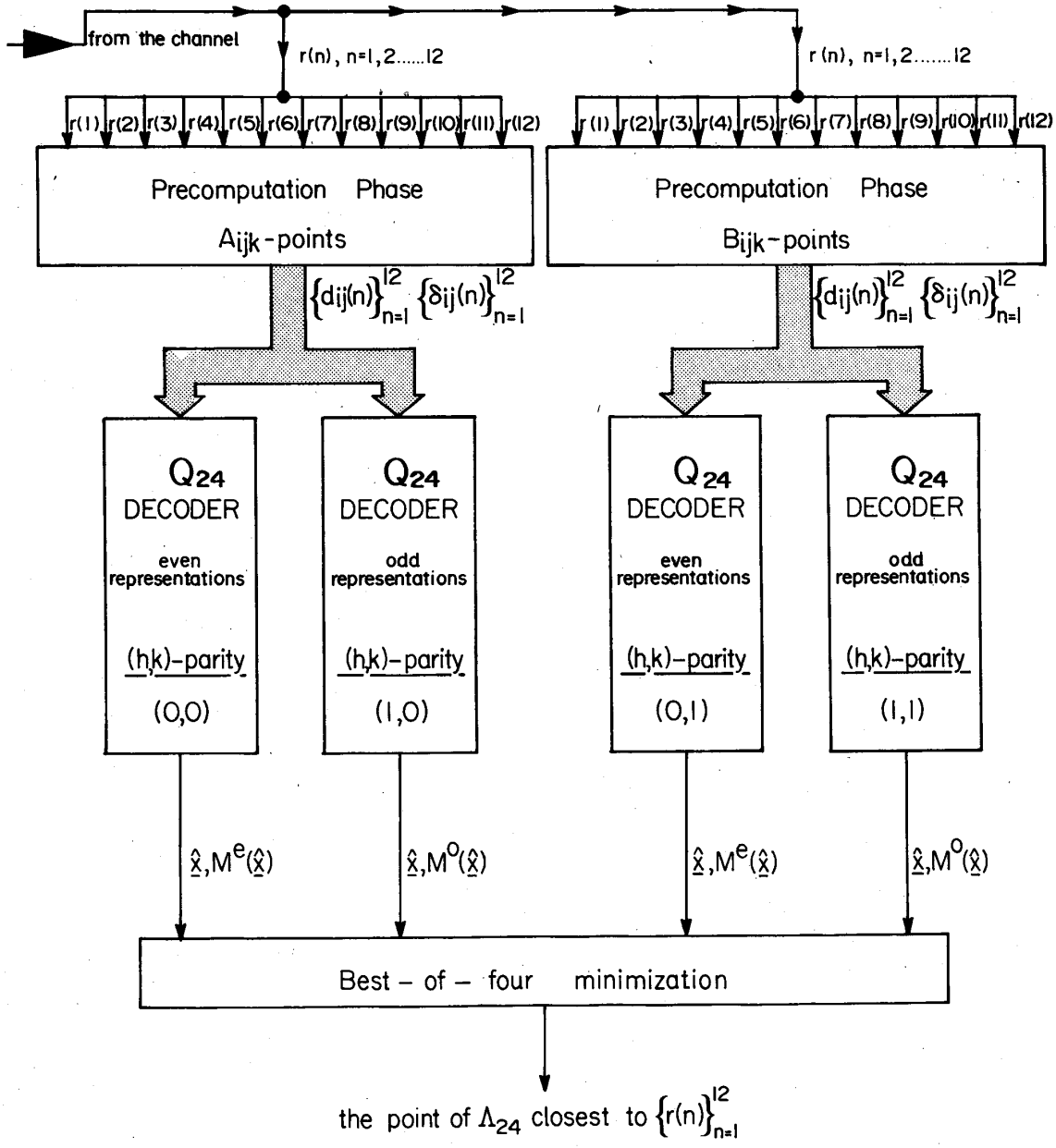

Fig. 5. Four computation paths of the Λ_{24} decoder.

likelihood decoding algorithm for the Leech-lattice code as well. Since the precomputation stage is performed in two dimensions, it would suffice to note that the point $\hat{A}_{ijk}(n)$ should be chosen from the finite set of all possible $A_{ijk}(n)$ points defined by the two-dimensional constellation (cf. Fig. 4). Owing to the fact that Q_{24} is a sublattice of $(2D_4)^6$ the proposed algorithm can be made to handle any Leech-lattice code that is defined in terms of four-dimensional constellations as well. This may be readily accomplished by performing the precomputation in four dimensions and including the first step of the algorithm in the precomputation stage. However, if the Leech-lattice code cannot be derived using six independent four-dimensional constellations, there is no way to ensure that the algorithm in choosing the closest Leech lattice point will also choose a point that belongs to the Leech-lattice code. In particular, the spherical codes of Adoul and Bàrth [1] and the codes contained in a thick shell, as discussed by de Buda [12], cannot be described in terms of independent four-dimensional constellations. Adapting the algorithm presented herein for decoding these Leech-lattice codes remains an open problem.

Finally, we should mention that while our algorithm achieves maximum likelihood decoding of the Leech lattice, there exist slightly suboptimal but more efficient decoding algorithms [15]. Thus using the algorithm of Forney [15] in conjunction with the algorithm of [26], bounded-distance decoding of the Leech lattice can be

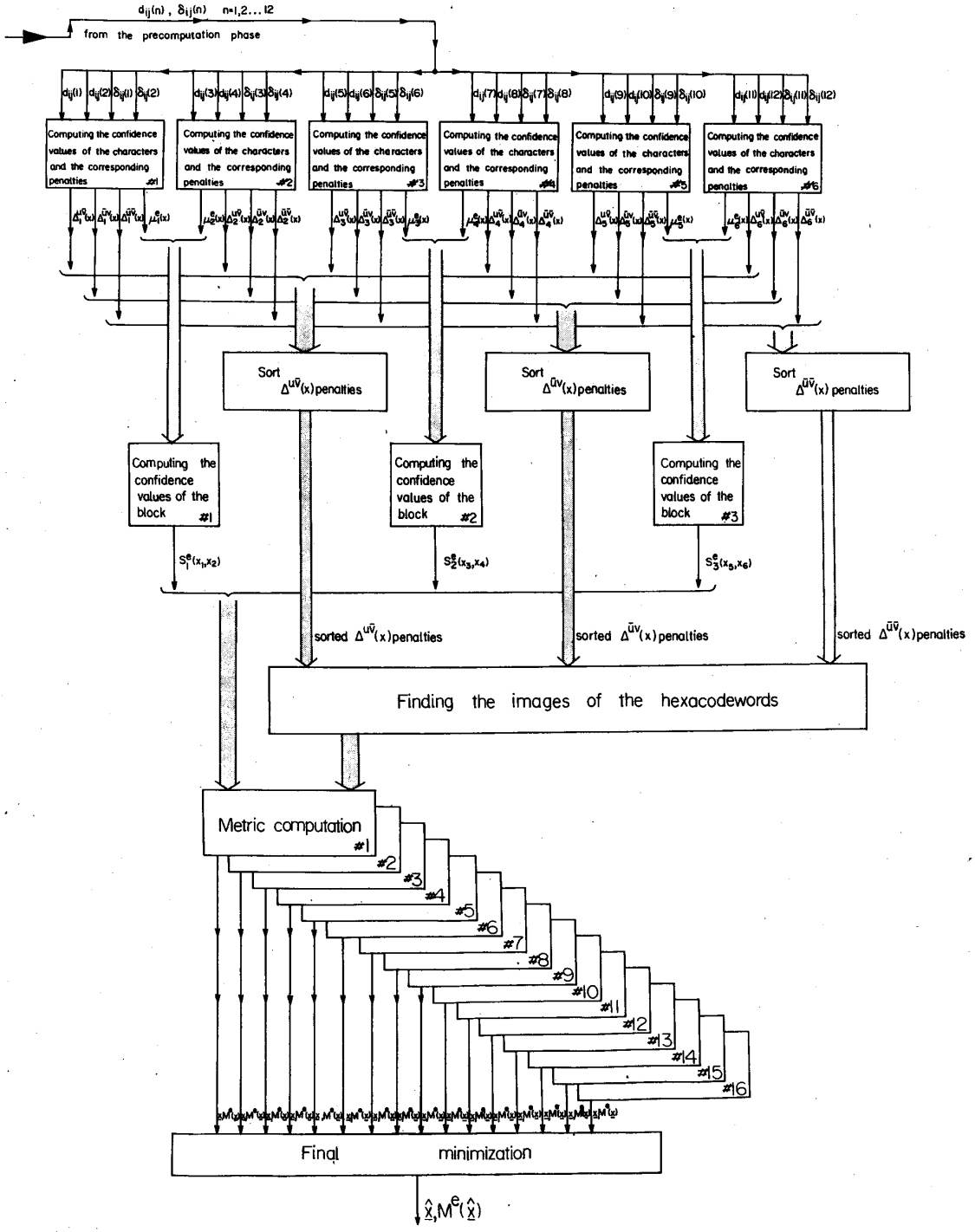

Fig. 6. Block diagram of the Q_{24} decoder.

performed with 1500 operations, while the SNR penalty is only about 0.1 dB. Furthermore the construction of the Leech lattice given in Section II provides the ground for the development of even more efficient bounded-distance decoding algorithms along the lines of [15].

ACKNOWLEDGMENT

The authors are indebted to S. Mattson for pointing out [27], [28] and to one of the referees for the contribution of the multilevel constructions of Λ_{24} and C_{24}, (which are illustrated in Fig. 3), and for

many valuable comments. A. Vardy wishes to thank Dr. S. Rubinstein, Dr. H. Rinsky, and H. Itzkowitz.

REFERENCES

[1] J. P. Adoul and M. Barth, "Nearest neighbor algorithm for spherical codes from the Leech lattice," *IEEE Trans. Inform. Theory*, vol. 34, pt. II, pp. 1188–1202, Sept. 1988.

[2] Y. Be'ery and B. Shahar, "VLSI Architectures for soft decoding of the Leech lattice and the Golay codes," *Proc. IEEE Int. Workshop on Microelectronics in Commun.*, Interlaken, Switzerland, Mar. 1991.

[3] Y. Be'ery, B. Shahar, and J. Snyders "Fast decoding of the Leech lattice," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 959–967, 1989.

[4] J. W. M. Bergmans, S. A. Rajput, and F. A. M. Van de Lear, "Co-designed coding, modulation and equalization," preprint.

[5] A. R. Calderbank and N. J. A. Sloane, "New trellis codes based on lattices and cosets," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 177–195, Mar. 1987.

[6] J. H. Conway, "Hexacode and tetracode—MOG and MINIMOG," in New York: Academic Press, *Computational Group Theory*, 1984, pp. 359–365.

[7] J. H. Conway and N. J. A. Sloane, "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 211–226, 1982.

[8] —— "On the Voronoi regions of certain lattices", *SIAM J. Algebraic Discrete Methods*, vol. 5, pp. 294–305, 1984.

[9] —— "Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 41–50, Jan. 1986.

[10] —— *Sphere Packings, Lattices and Groups.* New York: Springer-Verlag, 1988.

[11] R. T. Curtis, "A new combinatorial approach to M_{24}," *Math Proc. Camb. Phil. Soc.*, vol. 79, pp. 25–41, 1976.

[12] R. de Buda, "Some optimal codes have structure," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 959–967, 1989.

[13] G. D. Forney, Jr., "Coset codes I: Introduction and geometrical classification," *IEEE Trans. Inform. Theory*, vol. 34, Pt. II, pp. 1123–1151, Sept. 1988.

[14] —— "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pt. II pp. 1152–1187, Sept. 1988.

[15] —— "A bounded distance decoding algorithm for the Leech lattice, with generalizations," *IEEE Trans. Inform. Theory*, vol. 35, pp. 906–909, July 1989.

[16] G. D. Forney, Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Qureshi, "Efficient modulation for band-limited channels," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 632–647, 1984.

[17] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching.* Reading, MA: Addison-Wesley, 1973.

[18] G. R. Lang and F. M. Longstaff, "A Leech lattice modem," *IEEE J. Select Areas Commun.*, vol. 7, pp. 968–973, 1989.

[19] J. Leech, "Notes on sphere packing," *Can. J. Math.*, vol. 19, pp. 251–267, 1967.

[20] R. E. Peile, Final report on INTEL-963: "Co-designed coding, modulation and equilization," preprint.

[21] V. S. Pless, "Decoding the Golay codes," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 561–567, July 1986.

[22] R. A. Silverman and M. Balser, "Coding for a constant data rate source," *IRE Trans. Inform. Theory*, vol. IT-4, pp. 50–63, 1954.

[23] N. J. A. Sloane, "Tables of sphere packings and spherical codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 327–338, 1981.

[24] J. Snyders and Y. Be'ery, "Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 963–975, Sept. 1989.

[25] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.

[26] A. Vardy and Y. Be'ery, "More efficient soft-decision decoding of the Golay codes," *IEEE Trans. Inform. Theory*, vol. 37, pt. I, pp. 667–672, May 1991.

[27] E. F. Assmus, Jr. and H. F. Mattson, Jr., "Algebraic theory of codes II," *Scientific Rep. No. 1*, Contract No. F19628-69-C-0068, Air Force Cambridge Research Labs., 1969.

[28] E. P. Shaughnessy, "Codes with simple automorphism groups," *Arch. Math. (Basel)*, vol. 22, pp. 459–466, 1971.

A Coding Scheme for Single Peak-Shift Correction in (d, k)-Constrained Channels

A. V. Kuznetsov and A. J. Han Vinck

***Abstract*—A two step coding scheme for peak-shift correction in (d, k)-constrained sequences is described. The first step is based on q-ary ($q=k-d+1$ is a prime) block codes that allow correction of specific types of double errors caused by single peak-shifts. The second step is a simple conversion of q-ary symbols to binary strings of the type $00\cdots01$. The concatenation of these strings satisfies the (d, k)-constraint within the codeword and in concatenation with neighboring words. The length of the codewords is controlled and, if necessary, can be fixed. The rate R_t of the overall encoding approaches $(2\log_2(k-d+1))/(k+d+2)$ for large codeword lengths. Codes for correction of peak-shift, deletions, and insertions of zeros are presented as well. Encoding and decoding are done by simple algorithms without using look-up tables, enumeration or denumeration procedures and, therefore, the codelength may be large.**

***Index Terms*—Recording, peak-shift correction, (d, k)-constraint.**

I. INTRODUCTION

In digital magnetic and optical storage, binary sequences are used as modulation codes, and for the standard modulation scheme known as NRZI the "1's" of the recorded codeword determine the positions on the track of storage media where the magnetization is changed from one state to another one. When data bits are directly used as modulation codes, the number of bits per unit of track length (linear density) is equal to $n = 1/\delta$, where δ is the minimum acceptable distance between two consecutive transitions on the track. It was noted already in the sixties that the same minimum gap δ between transitions can be obtained with modulation codes that have $(d+1)$ times more bits per unit of track length but that satisfy the condition that the number of zeros between any two consecutive ones is not less than d. Since the number of such sequences equals 2^N, $N \approx (d+1)C(d)n$, where $C(d)$ is the Shannon capacity of the channel with d-constrained input, then the ratio $\gamma = N/n$ is equal to $(d+1)C(d)$ and grows with d [1]. It characterizes an improvement in the linear data density we gain by coding, and for this reason may be called the coding gain. For clock recovery the number of zeros between any two consecutive ones of the modulation code is upper bounded by another integer parameter k ($\geq d+1$). For small values of $d = 1, 2$ many good recording codes have been constructed in

Manuscript received April 22, 1991; revised December 7, 1992. A. V. Kuznetsov wa supported by Eindhoven University and Essen University. This work was presented at the IEEE International Symposium on Information Theory, San Antonio, TX, January 17–22, 1993. This work was done while A. V. Kuznetsov was with Eindhoven University in 1990 and with Essen University in 1991 on leave from the Institute for Problems of Information Transmission (IPPI) of the Academy of Sciences of Russia.

A. V. Kuznetsov is with the Institute for Problems of Information Transmission, Academy of Sciences of Russia, Ermolovoy str. 19, Moscow 101447, Russia.

A. J. Han Vinck is with the University of Essen, Institute for Experimental Mathematics, Ellernstr. 29, 4300 Essen, Germany.

IEEE Log Number 9209661.