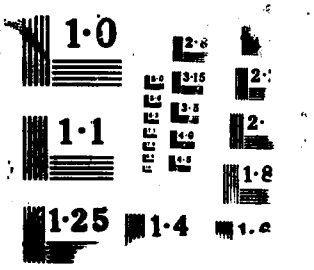
AD-A195 669 APPLICATIONS OF SIGNAL PROCESSING IN DIGITAL COMMUNICATIONS(U) POLITECNICO DI TORINO (ITALY) DEPT DI ELETTRONICA M ELIA ET AL. 30 APR 88 DAJA45-86-C-0044 1/1

UNCLASSIFIED F/G 12/3 NL

[redacted]


1·0
1·1
1·25
1·4

(2)

APPLICATIONS OF SIGNAL PROCESSING

IN DIGITAL COMMUNICATIONS

Principal Investigator: Michele Elia

Contractor: Politecnico di Torino
Corso Duca degli Abruzzi 24 - I-10129 TORINO (Italy)

Contract number DAJA45-86-C-0044

Fourth Interim Report
(November 1987 - April 1988)

The Research reported in this document has been made possible through the support and sponsorship of the U.S. Government through its European Research Office of the U.S. Army. ~~This report is intended only for the internal management use of the Contractor and the U.S. Government.~~

Our research activity during the period covered by this report was focused on the study of computational complexity related to error correcting codes and more in general to arithmetics over real and finite fields.

The importance of computational complexity in the design of any coding scheme either combined or not with modulations need not be explained. Our aim is to put together several scattered results and gather experience in manipulating such concepts in order to exhibit an organic view of computational problems that are present in the design of encoder decoder and demodulator with VLSI technologies.

Two papers are herewith enclosed. The first one concerns the complexity of power computations over any commutative ring with identity, i.e. finite fields, ring of matrices etc.
The second one concerns the decoding complexity of nonlinear codes and related topics connected with the computation of Hadamard Transforms, vector quantization and soft decoding of block codes.

The paper "A Note on Addition Chains and Some Related Conjectures" will be presented at the Advanced International Workshop on SEQUENCES, Combinatorics, Compression, Security and Transmission, to be held in Amalfi Coast, Salerno, Italy, June 1988.
The paper "A note on the Complete Decoding of Kerdock codes" will be presented at IEEE International Symposium on Information Theory to be held in Kobe, JAPAN, June 1988.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Applications of Signal Processing in Digital Communications		5. TYPE OF REPORT & PERIOD COVERED (Nov. 1987 - April 1988) 4th Interim Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ezio Biglieri and M. Elia		8. CONTRACT OR GRANT NUMBER(s) DAJA45-86-C-0044
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dipartimento di Elettronica Politecnico di Torino Corso D. Abruzzi 24 - I10129 Torino (I)		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S.Army Research, Development & Standardization GRoup - UK		12. REPORT DATE April 30, 1988 13. NUMBER OF PAGES 30
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Digital Communications, Group Codes, Computational Complexity		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) We consider the design of multidimensional signal sets and their combination with block or trellis codes. The goal is to achieve a high efficiency in the use of frequency spectrum for digital communications.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A Note on Addition Chains and some Related Conjectures *

M. Elia and F. Neri
Dipartimento di Elettronica
Politecnico di Torino – Italy

March 7th, 1988

Abstract

Addition chains are finite increasing sequences of positive integers, useful for the efficient evaluation of powers over rings. Many features of the addition chains are discussed, and some theorems connected to the still open Scholz-Brauer conjecture are presented.

1 Introduction

In many fields, such as computer science, applied number theory, cryptography, or numerical analysis, an efficient computation of

$$x^n = xx\ldots x \tag{1}$$

is required, where n is a positive integer ($n \in \mathcal{Z}$) and x can belong to any algebraic system $\mathcal{R}$ (usually a ring) in which an associative multiplication with identity is defined. This implies that sums and products can take a significantly different amount of time when applied to n (i.e. in $\mathcal{Z}$) or to x (i.e. in $\mathcal{R}$).

The problems typical of the evaluation of powers have been extensively discussed by Knuth [1] and by Borodin and Munro [2]. Several schemes have been proposed, in order to minimize the efforts (i.e. number of multiplications) for evaluating (1), but it seems that none can be definitely preferable in the general case. The choice of an approach instead of the other is affected by a number of constraints, aims or available resources, namely:

*This work was financially supported in part by the United States Army through its European Research Office, under grant n. DAJA45-86-C-0044.

- the order of magnitude of the exponent n;
- the availability of storage for precomputed tables;
- whether the situation calls for
  1. independent evaluations of the power (1)
  2. evaluations of several powers of the same base x
  3. evaluations of several powers to the same exponent n.

As an example, consider the generation of a pseudo-random sequence by purely multiplicative congruential methods, i.e. the desired sequence $< x_n >$ is obtained by the iterative relation

$$x_{n+1} = a x_n \bmod m. \tag{2}$$

A desirable feature for relation (2) is to generate a sequence with maximum period; if m is a prime number, this is achieved whenever the multiplier a is a primitive element in the finite field GF(m). The test for a number to be primitive can be a quite formidable task, however it can be obtained by raising the number being tested to quantities related to the factors of $m-1$, as shown in [3]. These exponents have the same order of magnitude of m, hence they are rather sizable for non trivial periods of interesting sequences $< x_n >$; moreover all of the operations must be done fully exploiting finite size registers if long periodicity is desired (see [3]), so that even the simple multiplication can be fairly costly. In this situation the efficiency in the computation of (1) is crucial.

2 Power Evaluations

At a first sight a very economical evaluation of (1) is obtained by the binary decomposition of the exponent n, which leads to a number of multiplications upper bounded by $2\log_2 n$. The same decomposition implies the simple but tight lower bound $\lceil \log_2 n \rceil$. Most considerations about the evaluation of powers concern the estimation of tighter upper bounds.

If we write

$$n = \sum_{i=0}^{t} b_i 2^i, \qquad b_i \in \{0,1\}, \tag{3}$$

where $t = \lfloor \log_2 n \rfloor$, the power (1) can be computed as

$$x^n = \prod_{i=0}^{t} \left(x^{2^i}\right)^{b_i}. \tag{4}$$

Given that the b_i's can be only 0 or 1, raising to b_i is straightforward. We shall call this approach *right to left binary method.*

In (4) $t-1$ multiplications are required to evaluate the powers

$$x^{2^i} \tag{5}$$

and one more multiplication is needed for every non zero b_i, leading to a total of

$$\lfloor \log_2 n \rfloor - 1 + \nu(n) \tag{6}$$

multiplications, where $\nu(n)$ is the number of 1's in the binary representation of n. The storage required by an implementation of the binary method (4) can be reduced to three memory cells: one to hold the successive powers (5), another to hold n during its decomposition, and an accumulator for the result.

The right to left binary method can be generalized to an m-ary method in the following way [4]. Assume the task to be the computation of (1) with the constraint

$$n < k, \qquad\qquad k \text{ fixed}, \tag{7}$$

and let

$$t = \lfloor \log_m k \rfloor. \tag{8}$$

The exponent n can be decomposed as

$$n = \sum_{i=0}^{t} d_i m^i, \qquad\qquad d_i \in (0, 1, \ldots, m-1). \tag{9}$$

This decomposition can be rewritten as

$$n = \sum_{i \in J_1} m^i + 2 \sum_{i \in J_2} m^i + \ldots + (m-1) \sum_{i \in J_{m-1}} m^i, \tag{10}$$

where J_j denotes the set of indices such that the coefficients d_i in (9) are equal to j.

The right to left M-ary method can be described by the following procedure.

Step 1. COMPUTE AND STORE (11)
$x^m, x^{m^2}, x^{m^3}, \ldots, x^{m^t}$; ($t$ as in (8))
Step 2. FOR EVERY $j \in 1 \ldots m-1$
COMPUTE $xx_j = x^{j \sum_{i \in J_j} m^i}$
Step 3. COMPUTE (1) AS
$\prod_{j=1}^{m-1} xx_j$

Step 1 of procedure (11) requires at most $tl(m)$ multiplications, if $l(m)$ is the minimum number of multiplications for raising a number to its m-th power: actually, in the average, not all the terms in Step 1 will be necessary. Raising to j in Step 2 requires $l(j)$ multiplications, while the remaining operations in Steps 2 and 3 can be carried out with no more than $t-1$ multiplications. The total number of multiplications is bounded by

$$t\,l(m) + t - 1 + \sum_{j=2}^{m-1} l(j) \qquad (t \text{ as in (8)}). \tag{12}$$

Another way of computing (1) is to rewrite the exponent n from (3) by Horner's rule for evaluating polynomials

$$n = b_0 + 2(b_1 + 2(b_2 + 2(b_3 + (\ldots + 2b_t)\ldots))).$$

We shall refer to this approach as *left to right binary method*, since a left to right scanning of n's binary representation is required.

The left to right binary method can be extended to an m-ary method, as described by the following procedure.

Step 1. COMPUTE AND STORE (13)
$x, x^2, x^3, \ldots, x^{m-1}$;
Step 2. LET $i = t$;
START WITH x^{d_t};
Step 3. REPEAT
LET $i = i - 1$;
RAISE TO THE m-TH POWER;
IF d_i IS NOT 0
MULTIPLY BY x^{d_i};
UNTIL $i = 0$;

Table 1: Number of multiplications in computing (1).

base m	right to left procedure (11)	left to right procedure (13)
2	$2\lfloor \log_2 n \rfloor - 1$	$2\lfloor \log_2 n \rfloor$
3	$3\lfloor \log_3 n \rfloor$	$3\lfloor \log_3 n \rfloor + 1$
4	$3\lfloor \log_4 n \rfloor + 2$	$3\lfloor \log_4 n \rfloor + 2$
5	$4\lfloor \log_5 n \rfloor + 4$	$4\lfloor \log_5 n \rfloor + 3$
6	$4\lfloor \log_6 n \rfloor + 7$	$4\lfloor \log_6 n \rfloor + 4$
7	$5\lfloor \log_7 n \rfloor + 10$	$5\lfloor \log_7 n \rfloor + 5$
8	$4\lfloor \log_8 n \rfloor + 14$	$4\lfloor \log_8 n \rfloor + 6$

Note that a certain amount of storage is necessary for the quantities computed in the first step of the above procedure; moreover the representation base m of n must be available in a left to right order.

Step 1 of procedure (13) requires at most $m-2$ multiplications; actually the x^{d_i} do not need to be computed for those values of d_i not present in the decomposition (9). Each iteration of Step 3 requires at most $l(m)+1$ multiplications, if $l(m)$ is the minimum number of multiplications for raising a number to its m-th power; the $+1$ is present only if the i-th d_i is not 0. In total the number of multiplications is bounded by

$$m - 2 + t\,(l(m)+1) \qquad (t \text{ as in } (8)). \tag{14}$$

By comparing the bounds (14) and (12) Table 1 can be built, where t is expressed as in (8). The order of magnitude of the exponent n (given by k in (7)) can be seen to affect the choice of the base m; the optimal m increases with k. Moreover, those bases that are powers of 2 appear somehow optimal, since they lead to comparatively small coefficients for $\lfloor \log_m n \rfloor$ in Table 1.

Note that all the bounds were derived neglecting the cost of decomposing the exponent n, and of related operations, such as the arrangement in (10) of n's terms in the sets J_j. This can be an acceptable approximation since the operations on x can be considered to be much more expensive than the operations on n.

Even if the left to right m-ary method seems to behave better for large

bases m, a careful inspection of the bounds (14) and (12) shows that the bound (12) is weaker, since Steps 1 and 2 of procedure (11) are open to several optimizations both in the case of few and the case of many terms in the decomposition (9).

When several powers of the same base x are to be done the precomputation Step 1 can be executed only once in both procedures (13) and (11); in this case the right to left method becomes extremely advantageous, since the precomputation in (11) is much heavier, and the bound

$$t\,l(m) + p\left(t - 1 + \sum_{j=2}^{m-1} l(j)\right)$$

can be obtained, where p is the number of powers to be computed.

3 Asymptotical Behavior: a Bound for l(n)

In this Section it is shown that both the upper and the lower bounds presented in the previous Section are asymptotically (for large exponents n) equivalent.

Let us consider the m-ary methods, and substitute

$$t = \lfloor \log_m n \rfloor$$

in (12); the minimum number of multiplications for raising to n, $l(n)$, is bounded by the number of multiplications required by the m-ary method (which is not optimal in general)

$$\log_2 n \leq \lceil \log_2 n \rceil \leq l(n) \leq \lfloor \log_m n \rfloor l(m) + \lfloor \log_m n \rfloor - 1 + \sum_{i=2}^{m-1} l(i); \quad (15)$$

m is assumed to be the optimal base for the given n. In the rigthmost inequality of relation (15), the $\lfloor \circ \rfloor$ function can be neglected for large n's; hence, after a base change in the logarithms, it can be rewritten as

$$l(n) \ln m \leq (l(m) + 1) \ln n - \ln m + \ln m \sum_{i=2}^{m-1} l(i)$$

and

$$\frac{\ln m}{l(m)} \frac{l(n)}{\ln n} \leq (1 + \frac{1}{l(m)}) - \frac{\ln m}{\ln n\, l(m)} + \frac{\ln m}{\ln n} \sum_{i=2}^{m-1} \frac{l(i)}{l(m)}. \quad (16)$$

Since the optimal base has been shown to increase when n increases, and m in (15) is chosen to be the optimal base, all the terms in the right hand part of (16) become negligible for large n, and we get

$$l(n) \approx \frac{\ln n}{k}, \qquad k = \frac{\ln m}{l(m)}.$$

The limiting values of the constant k can be obtained by taking

$$m = 2^s,$$

for which we have

$$\ln m = s \ln 2$$

$$l(m) = s$$

$$k \approx \frac{\ln m}{l(m)} \approx \ln 2$$

and the final result

$$l(n) \approx \log_2 n.$$

Note that Knuth [1, page 451, Theorem D] gives a totally different proof, due to Brauer [5], of this fact.

4 Addition Chains

Addition chains are the proper tool for solving the problem of computing (1) for a given n with the minimum number of multiplications, i.e. to

$$\textit{find } l(n). \textit{ minimum number of products for evaluating the } n\textit{-th power.} \tag{17}$$

Note that this problem is only a particular case of problem (1), in the sense that nothing is said about the cost of deriving $l(n)$; and this cost can exceed by far the cost of computing (1) by anyone of the previously quoted methods. Hence the addition chains' approach to the evaluation of powers is of interest either when several quantities need to be raised to a same fixed exponent, or in the case where enough storage is available to store the precomputed addition chains for all the possible values of the exponent n.

Let's see the formal definition of addition chains.
An *addition chain* for n is a sequence of integers

$$1 = a_0 < a_1 < a_2 < \ldots < a_r = n$$

with the property that, for every i, a couple (j,k) can be found, such that

$$a_i = a_j + a_k, \qquad i > j;\ i > k. \tag{18}$$

Without loss of generality, the a_i's are assumed to be sorted in ascending order, and with no duplications. Let

$$l(n) = \text{minimum } r \text{ for which there exists an addition chain of length } r \text{ for } n; \tag{19}$$

then this addition chain is a solution to problem (17).

It is convenient to define two special classes of addition chains. A *star chain* is defined as in (18) with the stronger constraint $j = i - 1$. An l^0-*chain* is an addition chain with some *marked* elements; the condition is that in (18) a_j is the largest marked element less that a_i. It can be shown that

$$l(n) \le l^0(n) \le l^*(n), \tag{20}$$

where $l^0(n)$ and $l^*(n)$ are defined as in (19), respectively for l^0-chains and star chains.

A lot has been written about addition chains (see [1] for a presentation of the main results), but the problem of finding $l(n)$ is not completely settled, in the sense that $l(n)$ is not known for all n's.

4.1 Functions Related To Addition Chains

Two interesting functions are related to $l(n)$ in (19); they are defined as follows.

$$c(r) = \text{minimum integer } n \text{ that } l(n) = r \tag{21}$$

$$d(r) = \text{number of solutions in } n \text{ to the equation } l(n) = r \tag{22}$$

From the previous sections, the function $l(n)$ is known to satisfy the following bounds

$$\lceil \log_2 n \rceil \le l(n) \le \lfloor \log_2 n \rfloor + \nu(n) - 1, \tag{23}$$

where $\nu(n)$ is defined in (6). Since $\nu(n) \le \lceil \log_2 n \rceil$, and

$$\lfloor \log_2 n \rfloor + \lceil \log_2 n \rceil \le 2\lfloor \log_2 n \rfloor + 1,$$

the bounds (23) can be rewritten as

$$\lceil \log_2 n \rceil \le l(n) \le 2\lfloor \log_2 n \rfloor. \tag{24}$$

For a generic n, for which $l(n) = r$, the following bounds hold

$$c(r) \leq n \leq 2^r; \tag{25}$$

the lower bound is straightforward from the definition (21) of $c(r)$, while the upper bound derives from the lower bound in (23). Substituting $n = c(r)$ in (24), we can write

$$\log_2 c(r) \leq r \leq 2 \log_2 c(r) \tag{26}$$

and, by combining the upper bounds in (25) and (26), we get

$$2^{r/2} \leq c(r) \leq 2^r. \tag{27}$$

From the lower bound in (27), the following holds for the function $d(\circ)$ defined in (22)

$$2^{r/2} \leq \sum_{t=1}^{r-1} d(t).$$

From (25), and from the definition of $d(r)$, the following inequality can be stated

$$d(r) < 2^r - c(r) + 1;$$

hence

$$d(r) + c(r) < 2^r + 1.$$

Another bound on $c(r)$ can be derived in the following way.

$$\sum_{t=1}^{r-1} (c(t) + d(t)) \leq \sum_{t=1}^{r-1} (2^t + 1) = 2^r - 2 + r - 1$$

$$\sum_{t=1}^{r-1} c(t) + 2^{r/2} \leq 2^r + r - 3$$

$$\sum_{t=1}^{r-1} 2^{t/2} \leq \sum_{t=1}^{r-1} c(t) \leq 2^r - 2^{r/2} + r - 3$$

$$(2^{1/2} + 1)(2^{r/2} - 1) - 1 \leq \sum_{t=1}^{r-1} c(t) \leq 2^r - 2^{r/2} + r - 3.$$

Asymptotically:

$$2^{.5r+1.3} \leq \sum_{t=1}^{r-1} c(t) \leq \left(2^{r/2} - \frac{1}{2}\right)^2 + r.$$

Table 2:

r	$2^{r/2}$	$d(r)$	$c(r)$	$S_d(r)$	$S_c(r)$	2^r
1	1.41	1	2	1	2	2
2	2	2	3	3	5	4
3	2.82	3	5	6	10	8
4	4	5	7	11	17	16
5	5.65	9	11	20	28	32
6	8	15	19	35	47	64
7	11.3	26	29	61	76	128
8	16	44	47	105	123	256
9	22.6	78	71	183	194	512
10	32	136	127	319	321	1024
11	45.2	246	191	565	512	2048
12	64	432	397	997	909	4096
13	90.5	772	607	1769	1516	8192
14	128		1087		2603	16384
15	181.0		1903		4506	32768

From this and the previous relations, a likely conjecture is that the functions $c(r)$ and $d(r)$ behave asymptotically as the r-th power of 2:

$$d(r) = \mathrm{O}\left(2^{d_r r}\right)$$

$$c(r) = \mathrm{O}\left(2^{c_r r}\right)$$

where d_r and c_r are constants between 0.5 and 1.

The values of $c(r)$ and $d(r)$ for some small values of r are shown in Table 2. The columns labeled as $S_d(r)$ and $S_c(r)$ show the quantities

$$S_d(r) = \sum_{t=1}^{r} d(t)$$

$$S_c(r) = \sum_{t=1}^{r} c(t).$$

5 The Scholz-Brauer Conjecture

A famous problem concerning addition chains is the Scholz-Brauer conjecture [6], referring to chains for $2^n - 1$, which are of special interest, since they are the worst case for the binary method (their binary representation is a stream of consecutive 1's). Let us call a number n satisfying the inequality

$$l(2^n - 1) \leq n - 1 + l(n), \tag{28}$$

where $l(n)$ is defined in (19), a *SB-number*. The longstanding Scholz-Brauer conjecture states that

all positive integers are SB-numbers.

In the following, it will be shown that (28) holds for infinitely many n's. Let us recall some of the properties of $l(n)$, reported from [1]; they will be useful in the sequel.

$$l(nm) \leq l(n) + l(m); \tag{29}$$

$$l(2^a) = a;$$

$$l(2^a + 2^b) = a + 1 \qquad \text{if } a > b \geq 0; \tag{30}$$

$$l(2^a + 2^b + 2^c) = a + 2 \qquad \text{if } a > b > c \geq 0$$

(this is Theorem B in [1]);

$$a + 2 \leq l(2^a + 2^b + 2^c + 2^d) \leq a + 3 \qquad \text{if } a > b > c > d \geq 0,$$

where $n = 2^a + 2^b + 2^c + 2^d$ is said to be *special* (see [1, p.449]) if the lower bound holds with equality (this is called Theorem C in [1]);

$$l^0(2^n - 1) \leq n - 1 + l^0(n); \tag{31}$$

this implies that the Sholz-Brauer conjecture holds for l^0-chains (the result, due to Hansen, is called Theorem G in [1]).

Lemma 1 *For every integer a, the following inequality holds*

$$l(2^{2^a} - 1) \leq 2^a - 1 + a. \tag{32}$$

Proof - It is direct that (32) holds for $a = 0$. Now let us suppose (32) to be satisfied for $a - 1$; thus, using (29) and (30), we have

$$l(2^{2^a} - 1) = l\left((2^{2^{a-1}} - 1)(2^{2^{a-1}} + 1)\right) \leq$$

$$\leq l\,(2^{2^{a-1}} - 1) + l(2^{2^{a-1}} + 1) \leq$$

$$\leq 2^{a-1} - 1 + a - 1 + 2^{a-1} + 1 \leq$$

$$\leq 2^a - 1 + a.$$

The validity of (32) for every a follows from the induction principle.
□

Note that the recursive argument used in the proof above also defines an addition chain which contains numbers of the form

$$2^k\,(2^{2^h} - 1) \qquad 0 \leq k \leq 2^h; \quad 1 \leq h \leq a - 1. \tag{33}$$

For later use, we state this point as a Corollary.

Corollary 1 *There exists an addition chain for $2^{2^a} - 1$ of length $2^a - 1 + a$, such that it contains the numbers (33). This addition chain has the form*

$$\ldots, (2^{2^h} - 1), 2\,(2^{2^h} - 1), \ldots, 2^{2^h}\,(2^{2^h} - 1), (2^{2^{h+1}} - 1), \ldots$$

Note that

$$2^{2^h}(2^{2^h} - 1) + (2^{2^h} - 1) = 2^{2^{h+1}} - 2^{2^h} + 2^{2^h} - 1 = 2^{2^{h+1}} - 1.$$

Theorem 1 *For every positive integer n the inequality*

$$l(2^n - 1) \leq n - 2 + \nu(n) + \lfloor \log_2 n \rfloor \tag{34}$$

holds.

Proof - By decomposing n into its binary representation as in (3), we can write

$$2^n - 1 = 2^{\sum_{j=0}^{t-1} b_i 2^i}(2^{b_t 2^t} - 1) + 2^{\sum_{j=0}^{t-2} b_i 2^i}(2^{b_{t-1} 2^{t-1}} - 1) + \ldots + (2^{b_0} - 1) =$$

$$= \sum_{j=0}^{t} 2^{\sum_{i=0}^{j-1} b_i 2^i}(2^{b_j 2^j} - 1). \tag{35}$$

Applying Corollary 1 it can be seen that all the $\nu(n)$ terms in the summation but the first are in the chain for $(2^{2^t}-1)$, whose length, according to Lemma 1, is bounded by 2^t-1+t. Since the first factor in the first term can be expressed as 2^{n-2^t}, it accounts for at most $n-2^t$ multiplications. Combining these two contributions with the $\nu(n)-1$ additional multiplications required by the $\nu(n)$ not zero terms in the decomposition (35), the Theorem is proved.
□

Lemma 2 *If $l(n)=l^*(n)$ then n is a SB-number.*

Proof - Straightforward from (20) and (31).

Lemma 3 *If $l(n)=\lfloor \log_2 n \rfloor + \nu(n) - 1$ then n is a SB-number.*

Theorem 2 *Every n such that $\nu(n)$ is not greater than 4 is a SB-number.*

Proof - The proof of Theorem 2 is given separately for the four cases $\nu(n)=1,\ldots,4$.

Case $\nu(n)=1$ - Proved in Lemma 1.

Case $\nu(n)=2$ - It must be shown that, for every integer a and b such that $a>b\geq 0$, the following inequality holds

$$l(2^{2^a+2^b}-1) \leq 2^a+2^b+a.$$

We can write

$$2^{2^a+2^b}-1 = 2^{2^b}(2^{2^a}-1)+2^{2^b}-1.$$

From Corollary 1 we know that $2^{2^b}-1$ belongs to the addition chain ending in $2^{2^a}-1$, so that, using Lemma 1 we have

$$l(2^{2^a+2^b}-1) \leq l(2^{2^a}-1)+2^b+1 \leq 2^a+2^b+a.$$

□

Case $\nu(n)=3$ - It must be shown that, for every a, b and c such that $a>b>c\geq 0$, the following inequality holds

$$l(2^{2^a+2^b+2^c}-1) \leq 2^a+2^b+2^c+a+1.$$

In a way similar to the case $\nu(n) = 2$, using Corollary 1 and Lemma 1, the proof stems from the equality

$$2^{2^a+2^b+2^c} - 1 = 2^{2^b+2^c}(2^{2^a} - 1) + 2^{2^c}(2^{2^b} - 1) + 2^{2^c} - 1.$$

□

Case $\nu(n) = 4$ - Two subcases must be considered: $l(n) = a+3$ and $l(n) = a+2$. In the first case the proof follows from Theorem 1. In the second case it follows from Exercise 13 in [1, p. 463] — showing that n has a star chain so that Lemma 2 applies — and (31). □

6 Conclusions

Knuth reports that $1 \leq n \leq 18$ and sporadic 20, 24 and 32 are SB-numbers. We have shown, as a consequence of Theorem 2, that all $1 \leq n \leq 30$ are SB-numbers. For $n = 31$ Lemma 1 gives

$$31 \leq l(2^{31} - 1) \leq 31 + 7,$$

being the upper bound slightly greater than $31 + 6$, which comes from (28). Therefore 31 is the smallest integer that might not be a SB-number.

For numbers of the form $2^n - 1$, since $\log_2 n \geq \nu(n) - 1$, the inequality (34) can be rewritten as

$$l(2^n - 1) \leq n - 1 + c \log_2 n, \tag{36}$$

where c is a convenient constant $1 \leq c \leq 2$. This result implies that the numbers of the form $2^n - 1$ behave better than Erdos' probabilistic bound [7]

$$l(n) = \log_2 n + O\left(\frac{\log_2 n}{\log_2 \log_2 n}\right). \tag{37}$$

In fact, the second term at the right hand side of (37), in this case, has the form

$$\frac{\log_2(2^n - 1)}{\log_2 \log_2(2^n - 1)} \approx \frac{n}{\log_2 n}$$

and, for large n's

$$c \log_2 n < \frac{n}{\log_2 n}.$$

As a consequence of the results presented in this paper, an interesting open question is to *find the smallest value of c such that (36) holds for every n*.

References

[1] **D. E. Knuth,** *The Art of Computer Programming,* vol. II, Addison-Wesley, Reading Massachussetts, 1981, pp. 441-466.

[2] **A. Borodin, I. Munro,** *The Computational Complexity of Algebraic and Numeric Problems,* American Elsevier Pub., New York, 1975.

[3] **M. Elia, F. Neri,** Generation of Pseudorandom Independent Sequences, *Proceedings of the IASTED International Symposium MIC '86,* Innsbruck (Austria), Feb. 18-21, 1986, M. H. Hanza ed., Acta Press, pp. 25-28.

[4] **A. Chi-Chih Yao,** On the Evaluation of Powers, *SIAM J. Comput.,* Vol. 5, No. 1, Mar. 1976, pp. 100-103.

[5] **A. Brauer,** *Bull. Amer. Math. Soc.,* 45 (1939), pp. 736-739.

[6] **A. Scholz,** *Jahresbericht der Deutschen Mathematiker-Vereinigung,* (II), 47 (1937), pp. 41-42.

[7] **P. Erdos,** Remarks on Number Theory, III: On Addition Chains, *Acta Arithm.,* 6 (1960), pp. 77-81.

A Note on the Complete Decoding of Kerdock Codes *†

M. Elia, C. Losana and F. Neri
Dipartimento di Elettronica
Politecnico di Torino – Italy

Abstract

A definition of the Kerdock code $K(m)$ is given that allows instantaneous encoding and the application of different complete decoding strategies. The particularly interesting code $K(4)$ is thoroughly described, with applications to error correction and to vector quantisation. In both cases the complexity in terms of number of arithmetical and logical operations is discussed. Finally the bit error rate for $K(4)$ on the binary symmetric channel is found in closed form.

1 Introduction

Kerdock codes $K(m)$ are nonlinear codes having many interesting properties, such as high error correcting capabilities, high symmetry and beautiful descriptions, combined with low rates for great m's. They may be viewed in some way as dual codes of Preparata codes $P(m)$, another noteworthy class of nonlinear codes. The code $K(4)$ is very interesting because besides the relatively high rate 1/2, it coincides with the Preparata code $P(4)$, so that it looks like a sort of self-dual nonlinear code.

The most obvious application of Kerdock codes is their use as channel codes in communication systems. $K(4)$ may also be viewed as the Nordstrom-Robinson code $\mathcal{N}_{16}$, and used as a vector quantizer for encoding random waveforms such as in the case of speech Linear Predictive Coding

†This paper will be presented at IEEE International Symposium on Information Theory, Kobe, JAPAN, June 1988.

*This work was financially supported in part by the United States Army through its European Research Office, under grant n. DAJA45-86-C-0044.

i	A_i
0	1
$2^{m-1} - 2^{m/2-1}$	$2^m(2^{m-1} - 1)$
2^{m-1}	$2^{m+1} - 2$
$2^{m-1} + 2^{m/2-1}$	$2^m(2^{m-1} - 1)$
2^m	1

Table 1: Weight distribution for $K(m)$.

(LPC) at the rate of 1/2 bit per sample [3]. In a similar way Kerdock codes allow the decoding from data produced by soft demodulation. In both vector quantization and soft data decoding the problem is to minimize an objective function, which most frequently is taken to be the squared-error distortion.

In Section 2 a definition of $K(m)$ as systematic code is given that allows instantaneous encoding and the application of different strategies for a complete decoding, as it will be described in Section 3. The application of $K(4)$ to vector quantization will also be described in Section 3. A short analysis of the computational complexity pertaining to the above applications of $K(4)$ will be given in Section 4. Finally, Section 5 reports some results on the performance evaluation of $K(4)$ used as a channel code on the Binary Symmetric Channel (BSC).

2 Encoding

In this Section we briefly recall the formal definition of Kerdock codes in order to introduce a systematic encoding scheme. We also collect some of its general properties for easy reference.

The Kerdock code $K(m)$, m even, is a nonlinear code consisting of the Reed-Muller code of parameters $(2^m, m+1, 2^{m/2})$ and $2^{m-1} - 1$ cosets of $\mathcal{R}(1,m)$ in $\mathcal{R}(2,m)$. Usually $K(m)$ is denoted by $[2^m, 2^{2m}, 2^{m-1} - 2^{m/2-1}]$. Important features of any code are the weight and the distance distributions. The weight distribution of a $[n, M, d]$ code is the set $\{A_i\}_{i=0}^n$, where A_i denotes the number of codewords of weight i, while the distance distribution is the set $\{B_i\}_{i=0}^n$, where $M\,B_i$ is the number of ordered pairs of codewords such that the distance between them is i. Linear codes have $B_i = A_i$ and the same property is shown by Kerdock codes. The weight and distance distribution of $K(m)$, taken from [1], is given in Table 1.

Let $c_{\mathcal{R}}$ be a vector of $\mathcal{R}(1,m)$. For later use it is convenient to interpret

$c_{\mathcal{R}}$ according to the following decomposition

$$c_{\mathcal{R}} = (\underset{\leftarrow m+1 \rightarrow}{x_1} \mid \underset{\leftarrow m-1 \rightarrow}{a} \mid \underset{\leftarrow 2^m-2m \rightarrow}{b}). \qquad (1)$$

Let w_i be a coset leader that performs a translation of $\mathcal{R}(1,m)$ to generate a codeword c of $K(m)$, i.e.

$$c = w_i + c_{\mathcal{R}};$$

the code $K(m)$ can be written as the union of disjoint cosets of $\mathcal{R}(1,m)$ as follows

$$K(m) = [w_1 + \mathcal{R}(1,m)] \cup [w_2 + \mathcal{R}(1,m)] \cup \ldots \cup [w_{2^m} + \mathcal{R}(1,m)]. \qquad (2)$$

The definition of $K(m)$ strongly lies on the choice of the w_i's, $i = 1, \ldots, 2^m$, which may be obtained by means of primitive idempotents for length $2^{m-1} - 1$, or by using simplectic forms to define a convenient set of boolean functions. A very simple construction of $K(4)$ is given in [2] were the cosets leaders are defined through simplectic forms, very easy to obtain, on four variables.

For easy reference, it is convenient to introduce a binary matrix $W^{(m)}$, built with the coset leaders w_i written by rows, an example of which is given by (5).

Now we use a systematic $\mathcal{R}(1,m)$ code and its translates, given by coset leaders w_i of a special form, to generate $K(m)$.

Let $i = (\underset{\leftarrow m+1 \rightarrow}{i_1} \mid \underset{\leftarrow m-1 \rightarrow}{i_2})$ be a vector of $2m$ information bits. As a consequence of next Lemmas 1 and 2, w_i may be taken of the form

$$w_i = (o \mid i_2 \mid d)$$

and, referring to equation (1), we set $x_1 = i_1$, so that the codewords will result of the form

$$c = (\underset{\leftarrow m+1 \rightarrow}{i_1} \mid \underset{\leftarrow m-1 \rightarrow}{a + i_2} \mid \underset{\leftarrow 2^m-2m \rightarrow}{b + d}). \qquad (3)$$

The general properties that have been used to define $K(m)$ are formally stated in the next two Lemmas.

Lemma 1 *In every coset of a systematic linear (n,k,d) code there exists exactly one word with k consecutive zeros in information positions.*

Lemma 2 *In the matrix $W^{(m)}$ there exists a submatrix made of $m-1$ columns whose rows are the 2^{m-1} different binary sequences of $m-1$ bits.*

The proof of these two Lemmas is easy to obtain, and it will be omitted here.

From the form (3) it is seen that instantaneous encoding is possible. In fact the first $m+1$ bits can be transmitted while they enter the encoder. At the $(m+1)$-th bit the remaining parity check bits for the $\mathcal{R}(1,m)$ code, i.e. the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, can be computed. As the remaining $m-1$ information bits enter the encoder, they are summed with the entries of vector $\boldsymbol{a}$ and transmitted. At that point the coset leader $\boldsymbol{w}_i$ (hence the vector $\boldsymbol{d}$) is known, so that the remaining parity check bits can be computed as $\boldsymbol{b}+\boldsymbol{d}$ and transmitted.

The above results applied to $K(4)$ let the generating matrix of the underlying $\mathcal{R}(1,4)$ code be written in the form

$$G_{\mathcal{R}} = (G_1 \mid G_2 \mid G_3) = \begin{pmatrix} 10000 & 111 & 01101001 \\ 01000 & 110 & 11010101 \\ 00100 & 101 & 10110011 \\ 00010 & 011 & 10001111 \\ 00001 & 000 & 01111111 \end{pmatrix}; \tag{4}$$

and correspondently the matrix $W^{(4)}$ of coset leaders in the form

$$\begin{pmatrix} 00000 & 000 & 00000000 \\ 00000 & 001 & 11100101 \\ 00000 & 010 & 10111001 \\ 00000 & 100 & 11001011 \\ 00000 & 011 & 01010011 \\ 00000 & 101 & 00011101 \\ 00000 & 110 & 00100111 \\ 00000 & 111 & 11111110 \end{pmatrix}. \tag{5}$$

It must be noted that the $2m$ information positions in the code vectors $\boldsymbol{c}$ defined by equation (3) contain all the possible 2^{2m} binary sequences of such length, therefore the Kerdock code could be viewed as a strictly systematic code at the cost of loosing the orderly definition reported above.

3 Decoding and Quantization

In this Section some procedures for decoding Kerdock codes and some procedures for performing the vector quantization based on Kerdock codes are described. As a consequence of the definition given in Section 2, the problem of decoding Kerdock codes may be formulated as follows:

> *given a received word $\mathbf{r}$, find the pair $[\hat{\mathbf{w}}_i, \hat{\mathbf{c}}_R]$ made of a coset leader and a Reed-Muller codeword, such that the decoded codeword $\mathbf{c} = \hat{\mathbf{w}}_i + \hat{\mathbf{c}}_R$ satisfies the chosen decoding criterion.*

Several decision rules may be considered, their main difference lying in the manner adopted to resolve ties whenever more than $\lfloor \frac{d-1}{2} \rfloor$ errors are detected, since these codes are not perfect. In particular two strategies deserve special interest: the Maximum Likelihood (ML) and the Minimum Correction (MC) rules. They are defined as follows.

Maximum Likelihood rule: $\mathbf{r}$ is decoded as the codeword $\hat{\mathbf{c}}$ that maximizes the conditional probability $p\{\mathbf{c} \mid \mathbf{r}\}$. On BSC this rule coincides with the minimum distance decoding, i.e. $\mathbf{r}$ is decoded as the codeword $\hat{\mathbf{c}}$ corresponding to the minimum distance $d(\mathbf{c}, \mathbf{r})$.

Minimum Correction rule: $\mathbf{r}$ is decoded as the codeword at the minimum distance if the distance is less or equal to $\lfloor \frac{d-1}{2} \rfloor$; otherwise the information bits are extracted from the received word without any correction attempt.

Four decoding algorithms will be now described, based on the arithmetic in GF(2). Let us remind that the Hamming weight $\mathrm{wt}(\mathbf{x})$ of a vector $\mathbf{x}$ is the number of its nonzero components and let us introduce the vector $\mathbf{r}$, decomposed as in (1)

$$\mathbf{r} = (\mathbf{r}_1 \mid \mathbf{r}_2 \mid \mathbf{r}_3),$$

that will be referred to as the received vector.

Algorithm 1 (Minimum distance).

Store $\mathbf{c}_i$ in a table T.

- Input $\mathbf{r}$.
- Compute the 2^{2m} Hamming distances

$$y_i = \mathrm{wt}(\mathbf{r} - \mathbf{c}_i), \qquad i = 1, \ldots, 2^{2m}.$$

- Find the minimum $\hat{y}_i$. Ties are resolved by random equiprobable choices.

i	L_i
0	1
1	16
2	120
3	112
4	7

Table 2: Weight distribution of ML correctable error patterns for $K(m)$.

- Decode $\boldsymbol{r}$ as the codeword $\hat{\boldsymbol{c}}_i$.
- Recover $\hat{\boldsymbol{i}}$ from $\hat{\boldsymbol{c}}_i$.

□

The next algorithm is restricted to $K(4)$, as it takes advantage from the fact that Standard Array decoding is possible. In fact by computer search 256 correctable error patterns $\boldsymbol{\ell}_i$ have been found such that the translates $\boldsymbol{\ell}_i + K(4)$ do not overlap and cover the whole space $\mathrm{GF}(2)^{16}$. The weight distribution $\{L_i\}_{i=0}^{2^m}$ of the correctable error patterns is reported in Table 2, where L_i denotes the number of error patterns of weight i. From this Table, the fact that $K(4)$ is not quasi-perfect can be observed.

Algorithm 2 (Syndrome decoding - ML rule).

Store $H_{\mathcal{R}}$, the parity check matrix of $\mathcal{R}(1,4)$.
Store the vectors $\boldsymbol{z}_i = H_{\mathcal{R}}\boldsymbol{w}_i$, $i = 1, \ldots, 2^8$.
Store the vectors $\boldsymbol{u}_j = H_{\mathcal{R}}\boldsymbol{\ell}_j$, $j = 1, \ldots, 2^8$.

- Input $\boldsymbol{r}$.
- Compute the syndrome $\boldsymbol{s} = H_{\mathcal{R}}\boldsymbol{r}$.
- Find the pair $[\hat{\boldsymbol{u}}_j, \hat{\boldsymbol{z}}_i]$ that sums to $\boldsymbol{s}$.
- Recover $\hat{\boldsymbol{\ell}}_j$ from $\hat{\boldsymbol{u}}_j$.
- Decode $\boldsymbol{r}$ as $\hat{\boldsymbol{c}} = \boldsymbol{r} + \hat{\boldsymbol{\ell}}_j$.
- Recover $\hat{\boldsymbol{i}}$ from $\hat{\boldsymbol{c}}$.

□

There are good reasons to conjecture that this decoding algorithm may also be applied to decode $K(m)$, for every m.
Algorithm 2 can be adapted to the MC decoding rule as follows.

Algorithm 3 (Syndrome decoding - MC rule).

Store a table $\mathcal{T}$ of 8×137 syndrome vectors associated to error pattern of weight not greater than 2.
Store $H_{\mathcal{R}}$, the parity check matrix of $\mathcal{R}(1,4)$.
Store $G_{\mathcal{R}}$, the generating matrix of $\mathcal{R}(1,4)$.

- Input $\boldsymbol{r}$.
- Compute the syndrome $\boldsymbol{s} = H_{\mathcal{R}}\boldsymbol{r}$.
- Search for the error pattern $\hat{\boldsymbol{e}}$ in $\mathcal{T}$, using the entry $\boldsymbol{s}$.
- IF successful
  THEN decode $\boldsymbol{r}$ as $\hat{\boldsymbol{c}} = \boldsymbol{r} + \hat{\boldsymbol{e}}$
  recover $\hat{\boldsymbol{i}}$ from $\hat{\boldsymbol{c}}$
  ELSE take the first $m+1$ information bits unmodified: $\hat{\boldsymbol{i}}_1 = \boldsymbol{r}_1$,
  compute $\hat{\boldsymbol{a}} = G_2^T\,\hat{\boldsymbol{i}}_1$
  and take the remaining $m-1$ information bits as $\hat{\boldsymbol{i}}_2 = \boldsymbol{r}_2 + \hat{\boldsymbol{a}}$.

□

Algorithm 4 (Tabular Decoding).

Store a Table $\mathcal{T}_1$ of the indices j's associated to the error patters $\boldsymbol{\ell}_j$'s, for every $\boldsymbol{r} \in \mathrm{GF}(2)^{16}$.
Store a Table $\mathcal{T}_2$ of the error patterns $\boldsymbol{\ell}_j$'s, $j = 1, \ldots, 2^8$.

- Input $\boldsymbol{r}$.
- Get j from $\boldsymbol{r}$ using Table $\mathcal{T}_1$.
- Get $\boldsymbol{\ell}_j$ from j using Table $\mathcal{T}_2$.
- Compute $\hat{\boldsymbol{e}} = \boldsymbol{r} + \boldsymbol{\ell}_j$.
- Recover $\hat{\boldsymbol{i}}$ from $\hat{\boldsymbol{e}}$.

□

Vector quantization is a field where $K(4)$ has found a valuable application. Let us formally recall the vector quantization problem with minimum squared-error distortion. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in R^n$ be the input to the vector quantizer and let $\{\boldsymbol{c}_i\}_{i=1}^N$ be the set of codewords. The problem may be formulated as follows:

find the codeword c_i among the N possible ones which minimizes the squared error

$$\| x - c_i \|^2 = x^T x - 2x^T c_i + c_i^T c_i, \tag{6}$$

where if $c_i^T c_i$ is independent of i then the minimum distance is achieved by the codeword c_i yielding the largest scalar product $y_i = x^T c_i$.

The most efficient algorithms known today for performing the vector quantization using $K(4)$ are based on the Hadamard Transform (HT), whose definition, for easy reference, will now be recalled.

Let H_n denote an Hadamard matrix, which is a $n \times n$ matrix of $+1$'s and -1's with the property that the scalar product of any two distinct rows is 0. Thus H_n must satisfy the relation

$$H_n H_n^T = nI,$$

where I is the $n \times n$ identity matrix.

An n-dimensional column vector y is called the HT of the vector x if it is obtained multiplying the vector x by an Hadamard matrix, i.e.

$$y = H_n x.$$

In this context we shall consider the i-th binary codeword of $K(m)$ as a vector of $+1$'s and -1's, with $+1$'s replacing 0's and -1's replacing 1's, this will replace the usual vector sum over GF(2) with the Hadamard dot component-wise product of real vectors, hereafter denoted $\odot$. The scalar and Hadamard products are compatible in the sense that the following property holds

$$x^T(y \odot z) \;=\; (x \odot y)^T z. \tag{7}$$

Vector quantization with the $K(4)$ code, requires the computation of 256 scalar products

$$y_i = x^T c_i, \qquad\qquad i = 1, \ldots, 256, \tag{8}$$

and 256 comparisons to search the minimum $\hat{y}_i$. Applying the property (7), y_i may be computed as

$$y_i = x^T(w_j \odot c_R) \;=\; (x \odot w_j)^T c_R. \tag{9}$$

As noted in [1,2,3], the 32 codewords of $R(1,4)$ can be grouped to form a Hadamard matrix H_{16} and its negative $-H_{16}$. Therefore the y_i's can

be computed as HT's of the 8 vectors $(x \odot w_j)$. Moreover only 128 scalar products are to be computed because $K(4)$ in the ± 1 representation contains both c_i and $-c_i$. Finally only 128 comparisons are necessary to find the maximum scalar product; the search can be limited to the absolute values $| x^T c_i |$ and the proper codeword can then be chosen according to the sign of $x^T c_i$.

The above observations can be also applied to the minimum distance decoding of soft data. The computation of the Algorithm 1 may be performed by executing the HT's of the received vector r and the seven companion vectors $r \odot w_j$, $j = 2, \ldots, 8$. In the following, two algorithms that implement the decoding along these lines are described.

Algorithm 5 (HT - ML rule).

Store $W^{(4)}$.
Store H_{16}.

- Input r.
- Compute the 2^8 scalar products $y_i = r^T c_i$, $i = 1, \ldots, 2^8$, by performing the 16 HT's

$$H_{16}(r \odot w_i) \qquad i = 2, \ldots, 8$$
$$-H_{16}(r \odot w_i) \qquad i = 2, \ldots, 8.$$

- Find the maximum $\hat{y}_i$. Ties are resolved by random equiprobable choices.
- Decode r as the codeword $\hat{c}_i$.
- Recover $\hat{i}$ from $\hat{c}_i$.

□

Algorithm 6 (HT - MC rule).

Store $W^{(4)}$.
Store H_{16}.
Store $G_{\mathcal{R}}$.

- Input r.

- Compute the 2^8 scalar products $y_i = r^T c_i$, $i = 1, \ldots, 2^8$, by performing the 16 HT's

$$H_{16}(r \odot w_i) \qquad i = 1, \ldots, 8$$
$$-H_{16}(r \odot w_i) \qquad i = 1, \ldots, 8.$$

- Find the maximum $\hat{y}_i$.

- IF no ties
  THEN decode r as the codeword $\hat{c}_i$
  recover $\hat{s}$ from $\hat{c}_i$

  ELSE take the first $m+1$ information bits unmodified: $\hat{s}_1 = r_1$,
  compute $a = G_2^T \hat{s}_1$
  and take the remaining $m-1$ information bits as $\hat{s}_2 = r_2 + a$.

□

4 Computational Complexity

Every dissertation on decoding complexity suffers the lacking of suitable measures of complexity. However for most practical applications the number of arithmetical operations (in any field), the number of logical operations and the amount of storage required can be taken as meaningful figures. In the following the complexity for decoding $K(4)$ with both the ML and the MC rules is estimated. The complexity of vector quantization using $K(4)$ is also analyzed and a significant improvement over [3] is obtained.

Decoding . The complexity for decoding Algorithm 2 of the previous Section is briefly discussed in the following. Similar considerations can be applied to all the algorithms presented. Complexity results are summarized in Table 3.

Complexity of Algorithm 2. We have 2^{11} syndromes of 11 bits which correspond to the $2^3 \times 2^8$ pairs (w_i, ℓ_j); they can be precomputed as

$$H_R w_i + H_R \ell_j \qquad i = 1, \ldots, 8; \;\; j = 1, \ldots, 256. \tag{10}$$

The decoding operations require the computation of s, a table look up for finding w_i and ℓ_j, and finally the computation of $r + \ell_j$ to obtain the correct information bits. The complexity results in

- the storing of H_R, requiring 11×16 bits;

- the storing of $2^{11} \times 8$ bits, i.e. the reference to the suitable ℓ_j for every s;
- the storing of $2^8 \times 16$ bits, i.e. the ℓ_j's;
- 15×11 sums in GF(2) for the evaluation of $s = H_R r$;
- a direct access search in a table of 2^{11} entries to get j and then in a second table of 2^8 entries to get ℓ_j;
- the computation of the correct information bits requiring either
  - the storing of the w_i's, requiring 8×16 bits, and the computation of 5 sums in GF(2) to find the i_2 bits;
  - the storing of the i_2 bits for every codeword, requiring 256×3 bits, and a direct table access.

This complexity is the same for ML and MC rules, the only difference being different choices of the correctable error patterns.

Operating in real fields according to Algorithms 5 and 6 is convenient only for minimum distance decoding; the complexity is the same as in the case of vector quantization, as analyzed in the following, but it does not compare favorably with the decoding in GF(2) (see Table 3).

Quantization. In [3], by referring to a definition of $K(4)$ as Λ_{16}, it is shown that the nearest neighbor codeword can be found with 304 additions and 128 comparisons. By using the same argument introduced in [3], based on a variant of the Fast Hadamard Transform (FHT), and using our definition of $K(4)$, we will show that 288 additions are sufficient. The proof stems on the following observations, motivated in [1,2,3].

1. The HT of dimension 16 may be computed by evaluating HT's of smaller dimension. The matrix H_{16} may always be written as

$$H_{16} = \begin{pmatrix} H_8 & H_8 \\ H_8 & -H_8 \end{pmatrix} = \begin{pmatrix} H_4 & H_4 & H_4 & H_4 \\ H_4 & -H_4 & H_4 & -H_4 \\ H_4 & H_4 & -H_4 & -H_4 \\ H_4 & -H_4 & -H_4 & H_4 \end{pmatrix};$$

   from the above observations it follows that a HT of dimension 16 may be computed by performing two HT's of dimension 8 and operating 16 sums, and every HT of dimension 8 may by obtained from two HT's of dimension 4 and 8 sums. The FHT of dimension 4 requires 8 additions.

2. It is direct to verify that the HT's of a 4-dimensional vector a with an even number of components having the sign changed, may be obtained from the HT of a by simple permutations and changes of signs. In fact let $a^T = (a_1, a_2, a_3, a_4)$ be a 4-dimensional vector whose HT is the vector $b^T = (b_1, b_2, b_3, b_4)$ given by $b = H_4 a$, where the matrix H_4 has the structure

$$H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

If all four a_i's change the sign then the four b_i's do the same. If only two a_i's change the sign then the four b_i's are permuted and/or changed of sign according to the following correspondence

$$\begin{aligned}
(-a_1, -a_2,\ a_3,\ a_4) &\leftrightarrow (-b_3, -b_4, -b_1, -b_2) \\
(\ a_1,\ a_2, -a_3, -a_4) &\leftrightarrow (\ b_3,\ b_4,\ b_1,\ b_2) \\
(-a_1,\ a_2, -a_3,\ a_4) &\leftrightarrow (-b_2, -b_1, -b_4, -b_3) \\
(\ a_1, -a_2,\ a_3, -a_4) &\leftrightarrow (\ b_2,\ b_1,\ b_4,\ b_3) \\
(-a_1,\ a_2,\ a_3, -a_4) &\leftrightarrow (-b_4, -b_3, -b_2, -b_1) \\
(\ a_1, -a_2, -a_3,\ a_4) &\leftrightarrow (\ b_4,\ b_3,\ b_2,\ b_1).
\end{aligned}$$

3. By permuting some columns in the matrix (5) we get

$$\begin{pmatrix}
0000\ 0000 & 0000\ 0000 \\
0000\ 0001 & 1010\ 1101 \\
0000\ 0011 & 1111\ 0000 \\
0000\ 0101 & 1100\ 1010 \\
0000\ 0011 & 0001\ 1011 \\
0000\ 0101 & 0101\ 0101 \\
0000\ 0111 & 0010\ 0110 \\
0000\ 0110 & 1111\ 1111
\end{pmatrix}. \qquad (11)$$

4. Due to the particular partition of the vectors w_i, as shown by the rows of matrix (11), many of the HT's of dimension 4 need not to be computed 8 times. In fact if both vectors x and w_i are partitioned into four parts of the same size

$$(x_1 \mid x_2 \mid x_3 \mid x_4)$$

and

$$(w_{1i} \mid w_{2i} \mid w_{3i} \mid w_{4i}),$$

their dot product may be performed independently in each single part

$$x \odot w_i = (x_1 \odot w_{1i} \mid x_2 \odot w_{2i} \mid x_3 \odot w_{3i} \mid x_4 \odot w_{4i}).$$

Therefore, the above observation, together with the property reported in the previous item 2, yields the conclusion that only the HT's of the following seven vectors must be computed

$$x_1$$

$$x_2 \text{ and } x_2 \odot (1,1,1,-1)$$

$$x_3 \text{ and } x_3 \odot (1,1,1,-1)$$

$$x_4 \text{ and } x_4 \odot (1,1,-1,1).$$

5. In combining the HT's pertaining to x_3 and x_4 to get the HT's of dimension 8, only 5 out of 8 combinations are necessary. This comes form the fact that two vector sums are to be performed

$$z_{1i} = x_3 \odot w_{3i} + x_4 \odot w_{4i}$$

$$z_{2i} = x_3 \odot w_{3i} - x_4 \odot w_{4i}.$$

In computing z_{1i} and z_{2i}, for $i = 1, \ldots, 8$, the cases $i = 3, 6, 8$ do not need any addition, since

$$z_{13} = -z_{21}$$

$$z_{23} = -z_{11}$$

$$z_{18} = -z_{11}$$

$$z_{28} = -z_{21}$$

z_{16} is a permutation of $-z_{11}$

z_{26} is a permutation of $-z_{21}$.

Alg.	real sums	GF(2) sums	memory	comparisons
1	3540	4096	4096 bit	255
2	–	144	4352 bit	–
3	–	140	12312 bit	–
4	–	16	528384 bit	–
5	288	–	384 bit	128
6	288	16	512 bit	128

Table 3: Complexity figures for decoding $K(4)$.

6. In conclusion the total number of additions is

128 $= 8 \times 16$; number of sums required for combining 8 pairs of HT's of dimension 8;
56 $= 7 \times 8$; number of sums required for the evaluation of seven HT's of dimension 4;
64 $= 8 \times 8$; number of sums required for combining the HT's pertaining to x_1 and x_2
40 $= 5 \times 8$; number of sums required for combining the HT's pertaining to x_3 and x_4

288

5 Bit Error Probability

In general it is very hard to compute either bit error rate or word error rate for nonlinear codes. For $K(4)$, however, such a computation is feasible because its structure is very similar to that of linear codes. As previously observed, the decoding can be organized as a Standard Array, since the translates of $K(4)$ by the correctable error patterns do not overlap and cover the whole vector space of dimension 16 over GF(2).

The polynomial expression of the bit error probability p_b of $K(4)$ after complete decoding on the BSC, in terms of p, raw bit error rate of the BSC, has the form

$$p_b = \frac{1}{8} \sum_{i=0}^{16} E_i \, p^i.$$

where the coefficients E_i are reported in Table 4. They have been computed from the Standard Array according to a counting scheme proposed in [1].

i	E_i	
	ML	MC
0	0	0
1	0	0
2	0	0
3	1464	1329
4	−12635	−10856
5	52116	40497
6	−130242	−81309
7	211196	65510
8	−218250	110310
9	117176	−409012
10	22836	675936
11	−99288	−704544
12	88496	496376
13	−43680	−233184
14	12480	66912
15	−1664	−8960
16	0	0

Table 4: Coefficients for BER computation of $K(4)$.

Interesting are the asymptotic expressions $p_b \asymp \frac{1464}{8}p^3$ and $p_b \asymp \frac{1329}{8}p^3$ for the ML and MC decoding respectively, as p tends to zero. From these relations it follows that, at least asymptotically, the MC rule is superior to the ML rule. These theoretical results have been checked by simulation using the TOPSIM package [7].

6 Conclusions

In this paper we have dealt with many different properties of Kerdock codes. Disparate results have been put together because they represents the manifold aspects of this very interesting class of nonlinear codes.

This final Section summarizes the many results that were obtained. First of all a description of Kerdock codes that allows instantaneous encoding is given. This approach leads to the application of two different decoding strategies, i.e. the well known Maximum Likelihood criterion and another one that we have called Minimum Correction rule. Referring to $K(4)$ it

has been shown that a Standard Array can be built by translating the set of codewords without overlapping. From the inspection of this Standard Array it turns out that K(4) is not quasi-perfect (see also Table 2). The same Standard Array allows the computation of the bit error rate for $K(4)$ on the binary symmetric channel, with respect to both ML and MC decoding strategies: in this particular case MC is asymptotically superior.

Finally it has been shown that a vector quantization scheme based on $K(4)$ can be performed with only 18 sums and 8 comparisons per sample.

References

[1] **F. J. MacWilliams and N. A. J. Sloane,** *The Theory of Error Correcting Codes,* North Holland, Amsterdam, 1977.

[2] **J. H. vanLint,** *Coding, Decoding and Combinatorics,* Applications of Combinatorics, R. J. Wilson editor, Shiva Pub., 1982, pp. 67-74.

[3] **J. Adoul,** Fast ML Decoding Algorithm for the Nordstrom-Robinson Code, *IEEE Trans. on Inform. Th.,* vol. IT-33, N. 6, Nov. 1987, pp. 931-933.

[4] **A. M. Kerdock,** A class of low-rate nonlinear codes, *Info. and Control,* 20 (1972), pp. 182-187.

[5] **J.H.Conway and N.J.A.Sloane,** Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice, *IEEE Trans. Inform. Theory,* vol. IT-32, Jan. 1986, pp. 41-50.

[6] **M.Elia,** A note on the computation of bit error rate for binary block codes, *Journal of Linear Algebra and its Applic.,* Wisconsin, Jan. 1988.

[7] **M. Ajmone-Marsan et al.,** Digital Simulation of Communication Systems with TOPSIM, *IEEE Journal Select. Areas in Communications,* vol. SAC-2, n. 1, Jan. 1984, pp. 42-50.

[8] **D. E. Knuth,** *The Art of Computer Programming,* vol. II, Addison-Wesley, Reading Massachussetts, 1981.

[9] **A. Borodin, I. Munro,** *The Computational Complexity of Algebraic and Numeric Problems,* American Elsevier Pub., New York, 1975.

END

DATE
FILMED

8 8