mathematics

MDPI

*Article*

# Decoding of $\mathbb{Z}_{2^S}$ Linear Generalized Kerdock Codes

**Aleksandar Minja** *,† **and Vojin Šenk** †,‡

Department of Power, Electronic and Telecommunication Engineering, Faculty of Engineering (Technical Sciences), University of Novi Sad, 21000 Novi Sad, Serbia; vojin_senk@uns.ac.rs

* Correspondence: aminja@uns.ac.rs

† These authors contributed equally to this work.

‡ Retired.

**Abstract:** Many families of binary nonlinear codes (e.g., Kerdock, Goethals, Delsarte–Goethals, Preparata) can be very simply constructed from linear codes over the $\mathbb{Z}_4$ ring (ring of integers modulo 4), by applying the Gray map to the quaternary symbols. Generalized Kerdock codes represent an extension of classical Kerdock codes to the $\mathbb{Z}_{2^S}$ ring. In this paper, we develop two novel soft-input decoders, designed to exploit the unique structure of these codes. We introduce a novel soft-input ML decoding algorithm and a soft-input soft-output MAP decoding algorithm of generalized Kerdock codes, with a complexity of $\mathcal{O}(N^S \log_2 N)$, where $N$ is the length of the $\mathbb{Z}_{2^S}$ code, that is, the number of $\mathbb{Z}_{2^S}$ symbols in a codeword. Simulations show that our novel decoders outperform the classical lifting decoder in terms of error rate by some 5 dB.

**Keywords:** codes over rings; Kerdock codes; MAP decoding; Monte Carlo simulation

**MSC:** 94B35



**Citation:** Minja, A.; Šenk, V. Decoding of $\mathbb{Z}_{2^S}$ Linear Generalized Kerdock Codes. *Mathematics* **2024**, *12*, 443. https://doi.org/10.3390/math12030443

Academic Editor: Ivo Petráš

Received: 13 December 2023
Revised: 14 January 2024
Accepted: 19 January 2024
Published: 30 January 2024



## 1. Introduction

**Context and motivation.** It is well known that linear codes over the integer ring modulo $2^S$ (the $\mathbb{Z}_{2^S}$ ring) are the natural linear codes for $2^S$-ary phase modulation ($2^S$-PSK) [1]. Additionally, Hammons et al. demonstrated in [2] that certain notable binary nonlinear codes, recognized for their desirable characteristics, can be derived from quaternary codes (codes over $\mathbb{Z}_4$) through the application of the Gray map. These results have led to an increased interest in designing linear codes over finite rings. Several extensions of these codes to $\mathbb{Z}_{p^S}$ (where $p$ is a prime and $S$ is a positive integer) were also developed [1,3–12]. Solé [3] suggested in 1988 that $p$-adic cyclic codes should be investigated [4]. The generalization of binary duadic codes to the setting of Abelian codes over the ring $\mathbb{Z}_{2^S}$ was presented in [5]. Double circulant self-dual codes over $\mathbb{Z}_{2^S}$ were investigated in [6]. Generalized Kerdock and Delsarte–Goethals codes over the ring $\mathbb{Z}_{2^S}$ (an extension of $\mathbb{Z}_4$ linear Kerdock and Delsarte–Goethals codes to $\mathbb{Z}_{2^S}$) were introduced in [7]. Convolutional codes over rings were discussed in [1,8,9], and LDPC codes over rings were investigated in [10–12].

Codes over rings were used in classical coding systems because of their rate properties [13–15], while in today's systems, they are considered for applications with great secrecy and reliability [16,17] requirements. On the other hand, modern codes over rings are being used to improve bandwidth efficiency [10] and to better combat phase noise [11,12] in modern wireless systems. Quaternary Kerdock codes were also used to design MIMO codebooks (i.e., a finite set of precoders shared by the transmitter and the receiver) for limited-feedback precoded MIMO systems [18,19]. The proposed Kerdock codebook was shown to have systematic construction, reduced storage, and reduced online search computation [18,19]. It is also possible to develop quantum error correction and quantum communication codes from codes over rings, as discussed in [20,21].

Research in the field of ring-based code decoding has been a significant topic for many years, during which numerous algorithms for both hard and soft decision decoding,

designed to exploit the unique and rich structure of codes over rings, have been introduced. A detailed overview of different decoding algorithms for codes over rings is given in Section 2. Classical codes over rings are interesting as component codes in product and turbo coding schemes [16,22,23], but one of the main obstacles to their adoption in modern coding systems is the lack of efficient SISO decoding algorithms. As far as we are aware, generalized Kerdock codes do not yet have any soft decoders specifically designed to exploit their unique structure, so the main objective of this study is to address this gap by presenting maximum likelihood (ML) and maximum a posteriori (MAP) decoding algorithms for them. Furthermore, the MAP decoding algorithm is an SISO algorithm, so it allows the use of these codes in modern coding systems. These algorithms will also allow us to develop and evaluate reduced-complexity suboptimal decoding algorithms for generalized Kerdock codes and other related code families in the future.

**Contribution.** In our previous paper [24], we developed an SISO MAP decoding algorithm of linear $\mathbb{Z}_4$ Kerdock codes with the complexity of $\mathcal{O}(N^2 \log_2 N)$, where $N$ is the code length of a quaternary code (i.e., the number of quaternary symbols in a codeword). Here, we extend this result to the family of generalized Kerdock codes. In this paper, novel ML and MAP decoding algorithms with a complexity of $\mathcal{O}(N^s \log_2 N)$ (where $N$ is the code length of a $\mathbb{Z}_{2^s}$ code) are developed and presented. To the best of our knowledge, these are the first such decoding algorithms for generalized Kerdock codes.

**Paper organization.** The rest of this manuscript is organized into five individual sections. The literature overview is given in Section 2. Section 3 provides the necessary preliminaries. Section 4 introduces the novel decoding algorithms. Simulation results are presented in Section 5. Section 6 concludes the paper.

**Notation.** In this paper, we employ the same notation as in [24]. We use bold letters to denote module elements, and by extension vectors, as every vector space is by definition a free module. We use the standard function notation for polynomials. We denote the $i$-th component of a module element $\boldsymbol{x}$ or a polynomial $x(Z)$ (where $Z$ is a placeholder variable) as $x_i$. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two elements of equal length $N$. Then, the Hadamard (pointwise) product is defined as $\boldsymbol{x} \odot \boldsymbol{y} = (x_0 y_0, x_1 y_1, \ldots, x_{N-1} y_{N-1})$, and the inner product is defined as $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_n x_n y_n$. We use uppercase letters to represent matrices, with $\boldsymbol{x}_n$ indicating the $n$-th row of a matrix $X$. For convenience, let $\mathbf{1} = (1, \ldots, 1)$ represent an element of all ones. We denote the probability of $x$ as $P[x]$ and the conditional probability of $x$ given $y$ as $P[x|y]$. We use the standard blackboard bold letters to represent sets of numbers and their associated rings/fields, i.e., $\mathbb{C}$ represents the set of complex numbers, $\mathbb{R}$ represents the set of real numbers, and $\mathbb{Z}$ represents the set of integers. The set of integers modulo $n$ is denoted as $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$. Given a set $\mathbb{K}$, $\mathbb{K}^2 = \mathbb{K} \times \mathbb{K}$ represents the Cartesian square (i.e., the Cartesian product of $\mathbb{K}$ with itself), and $\mathbb{K}^N$ represents the $N$-ary Cartesian power of $\mathbb{K}$. We use cursive letters to represent codes and other sets. Additional notation will be introduced as and when it is utilized throughout the paper.

## 2. Literature Overview

A variety of strategies have been used in designing decoders for codes over rings, as mentioned in [25]. These include the algebraic (syndrome) decoding approach [26], the lifting decoder technique, introduced in [27] and extended in [28], the coset decomposition approach [29], and the permutation decoding [25,30,31]. Several algebraic decoders of codes over rings were presented in [2,32–35]. A lifting decoding scheme was introduced in [27,28]. This algorithm works by sequentially applying the hard decision decoding algorithm of a corresponding $\mathbb{Z}_p$ code to the appropriate $p$-ary projection of the input and canceling a part of the noise at each step. In this manner, the $\mathbb{Z}_p$ decoding algorithm is lifted to work with the corresponding $\mathbb{Z}_{p^s}$ code. This is a general hard decision decoding algorithm that can be applied to the generalized Kerdock codes. A simple bitwise APP decoding algorithm of $\mathbb{Z}_4$ linear codes, based on the lifting decoding scheme, was proposed in [24]. It was shown in [24] that in the case of the classical Kerdock and Preparata codes, this decoder has a complexity of $\mathcal{O}(N \log_2 N)$. The Chase decoding of $\mathbb{Z}_4$ codes

based on the lifting decoder technique was proposed in [36,37]. The coset decomposition approach consists of partitioning the code into the subcode and its coset and works for both linear and nonlinear codes. A SISO MAP decoding algorithm of linear $\mathbb{Z}_4$ Kerdock and Preparata codes, based on the coset decomposition technique, was introduced in [24]. A similar approach was used to develop an ML decoding algorithm, of the same complexity, for quaternary Kerdock codes in [2,38]. Following the coset decomposition strategy, Davis and Jedwab, in [13], introduced two distinct algorithms for the decoding of RM-like codes over rings of characteristic 2. The maximum a posteriori (MAP) decoding of linear codes is usually performed via the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm—a trellis-based MAP decoder, introduced in [39]. The structure and complexity of the trellis diagram of the classical Kerdock and Preparata codes were investigated in [40] and [41], respectively. The complexity of the trellis-based approach is higher than that of the MAP decoder in [24]. For LDPC codes over rings, a modified message passing is generally used [10,42].

## 3. System Model

**Integer Modulo Rings.** Let operators $+$ and $\cdot$ represent addition and multiplication in $\mathbb{Z}_{2^S}$. Every $z \in \mathbb{Z}_{2^S}$ has a unique dyadic expansion (representation)

$$z = \sum_{n=0}^{S-1} z_n 2^n, \; z_n \in \mathbb{Z}_2.$$

Using the notation introduced in [27], let $z^{(s-1)}$, $s \in \{1, \ldots, S-1\}$, represent the projection of $z \in \mathbb{Z}_{2^S}$ onto the ring $\mathbb{Z}_{2^s}$, defined as

$$z^{(s-1)} = z \bmod 2^s = \sum_{n=0}^{s-1} z_n 2^n \in \mathbb{Z}_{2^s}, \tag{1}$$

where the first equality follows from the fact that the remainder when dividing by $2^s$ is defined by the first $s$ terms of the dyadic expansion, i.e.,

$$z = \sum_{n=0}^{s-1} z_n 2^n + \sum_{n=s}^{S-1} z_n 2^n = \sum_{n=0}^{s-1} z_n 2^n + 2^s \sum_{n=s}^{S-1} z_n 2^{n-s}.$$

It is clear that $z^{(0)} = z_0$ and $z^{(S-1)} = z$ for all $z \in \mathbb{Z}_{2^S}$.

It is easy to see that for every positive integer $s \leq S$, every element $z \in \mathbb{Z}_{2^s}$ is also an element of $\mathbb{Z}_{2^S}$. Notice that multiplying some number $z \in \mathbb{Z}_{2^S}$ by $2^{S-s}$, $s \in \{0, 1, \ldots, S-1\}$, is equivalent to first projecting the number $z$ onto the $\mathbb{Z}_{2^s}$ and then multiplying it with $2^{S-s}$, where the multiplication is conducted in $\mathbb{Z}_{2^S}$, i.e., $2^{S-s} \cdot z = 2^{S-s} \cdot z^{(s-1)}$.

$$\begin{aligned} 2^{S-s} \cdot z = 2^{S-s} \cdot \sum_{n=0}^{S-1} z_n 2^n = \sum_{n=S-s}^{S-1} z_{n-S+s} 2^n = 2^{S-s} \sum_{n=0}^{s-1} z_n 2^n \\ = 2^{S-s} \cdot (z \mod 2^s) = 2^{S-s} \cdot z^{(s-1)}. \end{aligned} \tag{2}$$

Additionally, given $a, b \in \mathbb{Z}_{2^S}$ and some $s \in \{0, 1, \ldots, S-1\}$,

$$\begin{aligned} &2^{S-s} \cdot (a+b) = 2^{S-s} \cdot a + 2^{S-s} \cdot b = \\ &2^{S-s} \cdot a^{(s-1)} + 2^{S-s} \cdot b^{(s-1)} = \\ &2^{S-s} (a^{(s-1)} \oplus b^{(s-1)}), \end{aligned}$$

where $\oplus$ represents addition in $\mathbb{Z}_{2^s}$.

Similarly, we have

$$\begin{aligned} &2^{S-s} \cdot (a \cdot b) = 2^{S-s} \cdot (a \cdot b \mod 2^s) = \\ &2^{S-s} \cdot (((a \mod 2^s) \cdot (b \mod 2^s)) \mod 2^s) = \\ &2^{S-s} \cdot (a^{(s-1)} \otimes b^{(s-1)}), \end{aligned}$$

where $\otimes$ represents multiplication in $\mathbb{Z}_{2^s}$.

Using these relationships, we can formally define a natural ring epimorphism [27] as

$$\begin{array}{ccccc} \mathbb{Z}_{2^S} & \longrightarrow & \mathbb{Z}_{2^S}/2^s\mathbb{Z}_{2^S} & \longrightarrow & \mathbb{Z}_{2^s} \\ z & \mapsto & z + 2^s\mathbb{Z}_{2^S} & \mapsto & z^{(s-1)} \end{array}$$

for $1 \leq s \leq S \in \mathbb{N}$. Additionally, for $s \leq S \in \mathbb{N}$, there exists a group embedding

$$\begin{array}{ccc} \mathbb{Z}_{2^s} & \longrightarrow & \mathbb{Z}_{2^S} \\ z & \mapsto & 2^{S-s} \cdot z \end{array}$$

with the image $2^{S-s}\mathbb{Z}_{2^S}$. The unique preimage of an element $z \in \mathbb{Z}_{2^S}$ is denoted by $z/2^{S-s}$ [27].

**Galois Rings**. Let $GR(2^S, d)$ represent a Galois ring of characteristic $2^S$ and order $2^{dS}$. It can be defined as a quotient ring

$$GR(2^S, d) \cong \mathbb{Z}_{2^S}[Z]/\langle h_S(Z)\rangle,$$

where $\mathbb{Z}_{2^S}[Z]$ represents a polynomial ring over $\mathbb{Z}_{2^S}$ and $h_S(Z)$ is a monic polynomial of degree $d$ that is irreducible over $\mathbb{Z}_{2^S}$ (i.e., it does not factor as a product of nontrivial polynomials). This means that the elements of $GR(2^S, d)$ can be represented as polynomials over $\mathbb{Z}_{2^S}$, of degree at most $d$, and addition and multiplication are performed modulo the polynomial $h_S(Z)$. The polynomial $h_S(Z)$ may be found using Graeffe's method [2], which is used for finding a polynomial over $\mathbb{Z}_{2^S}$ whose roots are the $S$-th power of the roots of a corresponding polynomial over $\mathbb{Z}_2$.

Let $h(Z) \in \mathbb{Z}_2[Z]$ be a primitive irreducible polynomial of degree $d$. There exists a unique monic polynomial $h_S(Z) \in \mathbb{Z}_{2^S}[Z]$ of degree $d$ such that $h(Z) \equiv h_S(Z) \mod 2$, and $h_S(Z)$ divides $Z^{N-1} - 1 \mod 2^S$, where $N = 2^d$ [2]. Following Graeffe's method, we first divide the $h(Z)$ into polynomials $e(Z)$ and $o(Z)$, such that $h(Z) = e(Z) - o(Z)$, and $e(Z)$ contains only even powers, while $o(Z)$ contains only odd powers. Polynomial $h_2(Z)$ is given by

$$h_2(Z^2) = \pm(e^2(Z) - o^2(Z)).$$

We can repeat this process multiple times in order to obtain $h_S(Z)$ for some $S > 1$.

**Example 1.** *Consider the polynomial* $h(Z) = Z^5 + Z^2 + 1$. *We divide* $h(Z)$ *into*

$$e(Z) = Z^2 + 1,$$

*and*

$$o(Z) = Z^5.$$

*As* $h_2(Z^2) = \pm(e^2(Z) - o^2(Z))$, *we have*

$$h_2(Z^2) = \begin{cases} \frac{1}{3}(3Z^{10} + Z^4 + 2Z^2 + 1) \\ Z^{10} + 3Z^4 + 2Z^2 + 3 \end{cases},$$

*where* $\frac{1}{3}$ *is used to ensure that we have a monic polynomial. In both cases, we have* $h_2(Z) = Z^5 + 3Z^2 + 2Z + 3$. *We can repeat this procedure to obtain*

$$h_3(Z) = Z^5 + 4Z^3 + 7Z^2 + 2Z + 7.$$

Notice that $h_S(Z) \mod 2^s = h_s(Z)$, for any positive integer $s \leq S \in \mathbb{N}$. This follows directly from the construction method, as $h_S(Z)$ is obtained from $h_s(Z)$, which is obtained from $h(Z)$, i.e., $h_S(Z) \mod 2 = h_s(Z) \mod 2 = h(Z)$.

We can extend the previous integer modulo ring relationships to Galois rings. Given a ring element $\alpha \in GR(2^S, d)$, let

$$\alpha \mod 2^s = \alpha^{(s-1)} \in GR(2^s, d),$$

where $GR(2^s, d)$ is the Galois ring generated by the monic irreducible polynomial $h_s(Z)$. This is a trivial extension that follows from the fact that the modulus operation is applied elementwise. Similarly, we have

$$2^{S-s} \cdot \alpha = 2^{S-s} \cdot (\alpha \mod 2^s) = 2^{S-s} \cdot \alpha^{(s-1)}.$$

Moreover, for every fixed $d$, there is a ring homomorphism (i.e., a function that preserves the ring operations)

$$\mu_{S-1} : GR(2^S, d) \to GR(2^{S-1}, d),$$

for each $S$, having kernel $2^{S-1}GR(2^S, d)$ [43]. We call this homomorphism a natural projection of ring $GR(2^S, d)$ onto the ring $GR(2^{S-1}, d)$, and we define it as

$$\mu_{S-1}(\alpha) = \alpha^{(S-2)} \in GR(2^{S-1}, d),\ \alpha \in GR(2^S, d).$$

Notice that the ring

$$GR(2^{S-1}, d) = \{\mu_S(\alpha) \mid \alpha \in GR(2^S, d)\}$$

is equivalent to the polynomial ring $\mathbb{Z}_{2^{S-1}}[Z]$ modulo monic irreducible polynomial $h_{S-1}(Z) = h_S(Z) \mod 2^{S-1}$. This definition can easily be extended to the projection of the ring $GR(2^S, d)$ onto the ring $GR(2^{S-s}, d)$, denoted $\mu_{S-s}$, for some $s \in \{1, \ldots, S-1\}$.

Let $+$ and $\cdot$ denote addition and multiplication in $GR(2^S, d)$, and let $\oplus$ and $\otimes$ denote addition and multiplication in $GR(2^s, d)$ for some $s \in \{1, \ldots, S\}$. Then, the following relationships naturally follow from the definition of projection:

$$2^{S-s}(\alpha + \beta) = 2^{S-s}(\alpha^{(s-1)} \oplus \beta^{(s-1)}),$$

$$2^{S-s}(\alpha \cdot \beta) = 2^{S-s}(\alpha^{(s-1)} \otimes \beta^{(s-1)}).$$

The natural projection $\mu \equiv \mu_1$ maps the ring $GR(2^S, d)$ to the Galois field $GR(2, d) = GF(2^d)$, generated by the primitive polynomial $h(Z)$.

Every Galois ring has a primitive element, $\xi$, such that $\xi^{N-1} = 1$. Given the Teichmüller set $\mathscr{T} = \{0, 1, \xi, \xi^2, \ldots, \xi^{N-2}\}$, every element $\alpha \in GR(2^S, d)$ has a unique "multiplicative" representation [7]

$$\alpha = \sum_{s=0}^{S-1} 2^s \tau_s,\ \tau_s \in \mathscr{T}. \tag{3}$$

The Frobenius map $f_S : GR(2^S, d) \to GR(2^S, d)$ is the ring automorphism defined as [2,7]

$$f_S : \sum_{s=0}^{S-1} 2^s \tau_s \to \sum_{s=0}^{S-1} 2^s \tau_s^2,\ \tau_s \in \mathscr{T}.$$

The relative trace from $GR(2^S, d)$ to $\mathbb{Z}_{2^s}$ is defined as [2,7]

$$T_S(\alpha) = \sum_{i=0}^{d-1} \alpha^{f_S^i}, \ \alpha \in GR(2^S, d).$$

Notice that $T \equiv T_1$ represents the usual trace from $GF(2^d)$ to $\mathbb{Z}_2$, defined as

$$T(\alpha_0) = \alpha_0 + \alpha_0^2 + \alpha_0^{2^2} + \cdots + \alpha_0^{2^{d-1}}.$$

Some useful properties of the relative trace are the following [44]:

- $T_S(\alpha + \beta) = T_S(\alpha) + T_S(\beta)$, for all $\alpha, \beta \in GR(2^S, d)$;
- $T_S(z\alpha) = zT_S(\alpha)$ for all $z \in \mathbb{Z}_{2^s}$, $\alpha \in GR(2^S, d)$;
- $T_S(\alpha_S^f) = (T_S(\alpha))_S^f = T_S(\alpha)$, for all $\alpha \in GR(2^S, d)$.

The projection of $\alpha \in GR(2^S, d)$ also has a unique "multiplicative" representation, given by

$$\alpha^{(s-1)} = \mu_s(\alpha) = \mu_s\left(\sum_{n=0}^{S-1} 2^n \tau_n\right) = \sum_{n=0}^{s-1} 2^n \mu_s(\tau_n) \ \tau_n \in \mathscr{T},$$

where $\mu_s(\xi) = \xi^{(s-1)}$ is the root of the monic irreducible polynomial $h_s(Z)$ and $2^{S-s}\xi = 2^{S-s}\xi^{(s-1)}$. Given the canonical projection homomorphism $\mu$, defined as the mod-2 reduction, the following commutative relationships can easily be verified [2,44]:

$$\mu \circ f_S = f \circ \mu, \tag{4}$$
$$\mu \circ T_S = T \circ \mu. \tag{5}$$

**Generalized Kerdock Codes.** The generalized Kerdock code over $\mathbb{Z}_{2^s}$, of dimension $K = d + 1$ and length $N = 2^d$, introduced in [7], is defined as a set $\mathcal{K}$ of all valid codewords. A sequence $\boldsymbol{c} = (c_{-\infty}, \ldots, c_{N-2})$ is a codeword in $\mathcal{K}$ if and only if, for some $\lambda \in GR(2^S, d)$ and $\epsilon \in \mathbb{Z}_{2^s}$,

$$c_n = T_S(\lambda \xi^n) + \epsilon, \ n \in \{-\infty, \ \ldots, N-2\}, \tag{6}$$

with a standard convention of $\xi^{-\infty} = 0$.

The information pair $(\lambda, \epsilon)$ represents the information sequence of length $d + 1$ (consisting of $d$ coefficients of the polynomial $\lambda$ plus symbol $\epsilon$).

**Theorem 1.** *The binary image of the generalized Kerdock code is the first-order Reed–Muller code of length* $2^d$*, the* $RM(1, d)$ *code.*

**Proof.** The quaternary projection of the code $\mathcal{K}$ is given by

$$\mathcal{K}^{(1)} = \mu_2(\mathcal{K}) = \{\mu_2(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathcal{K}\},$$

where $\mu_2$ is applied componentwise. Every code symbol $c_n$, $n \in \{-\infty, \ldots, N-2\}$, is given by

$$c_n^{(1)} = \mu_2(c_n) = \mu_2(T_S(\lambda \xi^n) + \epsilon). \tag{7}$$

Using the properties of the projection map and the relationship between the natural projection and the relative trace, Equation (7) can be rewritten as

$$c_n^{(1)} = T_2(\mu_2(\lambda) \otimes \mu_2(\xi)^n) + \mu_2(\epsilon) = T_2(\lambda^{(1)} \otimes (\xi^{(1)})^n) + \epsilon^{(1)},$$

where $\xi^{(1)}$ is the root of $h_2(Z) \in \mathbb{Z}_4[Z]$, $\lambda^{(1)}$ is an element of the Galois ring $GR(4, d)$, generated by the monic irreducible polynomial $h_2(Z)$, and $\epsilon^{(1)} \in \mathbb{Z}_4$. The operator $\otimes$ represents multiplication in $GR(4, d)$. This is the definition of the quaternary Kerdock code, as presented in [2].

The projection of the generalized Kerdock code onto $GR(4,d)$ is the corresponding classical quaternary Kerdock code. It is well known that the binary image of the Kerdock code (the $\mu(\mathcal{K}^{(1)})$ projection) is the $RM(1,d)$ code [2,24]. This is also true for the generalized Kerdock code, i.e., $\mu(\mathcal{K}^{(1)}) = \mu(\mu_2(\mathcal{K})) = \mu(\mathcal{K}) = RM(1,d)$. □

Permuting the coordinates of a code produces an equivalent code [26]. Consider a permutation $\pi : \Delta \to \Delta$, where $\Delta = \{-\infty, 0, 1, \ldots, N-2\}$. For any codeword $\boldsymbol{b} \in \mathcal{K}$, we can define a codeword $\boldsymbol{c} = \pi(\boldsymbol{b}) = (b_{\pi_{-\infty}}, b_{pi_0}, \ldots, b_{\pi_{N-2}})$ that belongs to the equivalent generalized Kerdock code defined by $\pi$. Let $\boldsymbol{b} \in \mathcal{K}$ be generated by an information pair $(\lambda, \epsilon)$. Then, $\boldsymbol{c} = \pi(\boldsymbol{b})$ is defined as

$$c_n = T_S(\lambda \xi^{\pi_n}) + \epsilon, \; n \in \Delta.$$

If the $RM(1,d)$ code is constructed by taking the rows of the binary Sylvester-type Hadamard matrix (a binary matrix obtained from a regular Sylvester-type Hadamard matrix, by replacing the 1's with binary 0's and $-1$'s with binary 1's) and their complements as codewords, it is possible to use the fast Hadamard transform for decoding [24,26]. In the remainder of the paper, we will assume that the permutation $\pi$ is chosen so that the associated binary code of the Kerdock code is the $RM(1,d)$ code, constructed in this way. This makes applying the fast Hadamard transform possible, significantly reducing the decoding complexity. Permutation $\pi$ can be found beforehand using the Hungarian algorithm [45] and does not affect the decoding complexity of the proposed algorithms.

**Channel model and decoding.** Let $\boldsymbol{x} = \phi(\boldsymbol{c})$, $\boldsymbol{c} \in \mathcal{K}$, be a random $2^S$-PSK-modulated codeword of a generalized Kerdock code $\mathcal{K}$, transmitted over the AWGN channel. The modulation mapping is defined as

$$\phi(\alpha) = \exp\left\{ I \frac{2\pi\alpha}{2^S} \right\}, \; I = \sqrt{-1}, \alpha \in \mathbb{Z}_{2^S}, \tag{8}$$

and it naturally extends to vectors. Furthermore,

$$\phi(\alpha + \beta) = \phi(\alpha)\phi(\beta). \tag{9}$$

Let $\boldsymbol{y}$ be the output of the complex AWGN channel, defined by the conditional probability

$$P[\boldsymbol{y} \mid \boldsymbol{x}] = \prod_{n \in \Delta} P[y_n \mid x_n] = \prod_{n \in \Delta} \frac{1}{\pi\sigma^2} \exp\left\{ -\frac{|y_n - x_n|^2}{\sigma^2} \right\},$$

where $|\cdot|$ represents the modulus of a complex number.

If all codewords are equally likely and the channel is memoryless, the ML decoder is optimal in terms of minimizing the word-error probability. Given the channel output $\boldsymbol{y}$, the ML decoder proceeds to find the codeword $\boldsymbol{c} \in \mathcal{C}$ that maximizes the $P[\boldsymbol{y} \mid \boldsymbol{x}]$, where $\boldsymbol{x} = \phi(\boldsymbol{c})$, i.e.,

$$\hat{\boldsymbol{c}} = \underset{\substack{\boldsymbol{x}=\phi(\boldsymbol{c}) \\ \boldsymbol{c} \in \mathcal{K}}}{\arg\max} P[\boldsymbol{y} \mid \boldsymbol{x}] = \underset{\substack{\boldsymbol{x}=\phi(\boldsymbol{c}) \\ \boldsymbol{c} \in \mathcal{K}}}{\arg\max} \prod_{n \in \Delta} P[y_n \mid x_n].$$

It is well known that the ML decoder can be implemented as the minimum distance decoder,

$$\hat{\boldsymbol{c}} = \underset{\substack{\boldsymbol{x}=\phi(\boldsymbol{c}) \\ \boldsymbol{c} \in \mathcal{K}}}{\arg\max} \prod_{n \in \Delta} \exp\left\{ -\frac{|y_n - x_n|^2}{\sigma^2} \right\} = \underset{\substack{\boldsymbol{x}=\phi(\boldsymbol{c}) \\ \boldsymbol{c} \in \mathcal{K}}}{\arg\min} \sum_{n \in \Delta} |y_n - x_n|^2 = \underset{\substack{\boldsymbol{x}=\phi(\boldsymbol{c}) \\ \boldsymbol{c} \in \mathcal{K}}}{\arg\min} \|\boldsymbol{y} - \boldsymbol{x}\|^2, \tag{10}$$

where $\|\cdot\|$ represents the norm of a complex vector.

Consider the term $|y_n - x_n|^2$ in Equation (10),

$$|y_n - x_n|^2 = (y_n - x_n)(y_n - x_n)^* = |y_n|^2 + |x_n|^2 - 2\Re(y_n x_n^*), \tag{11}$$

where $(\cdot)^*$ represents the complex conjugate and $\Re$ represents the function that extracts the real part of a complex number. By substituting (11) in Equation (10), we obtain

$$\hat{\boldsymbol{c}} = \underset{\substack{\boldsymbol{x}=\phi(\boldsymbol{c}) \\ \boldsymbol{c}\in\mathcal{K}}}{\arg\min} \sum_{n\in\Delta} |y_n - x_n|^2 = \underset{\substack{\boldsymbol{x}=\phi(\boldsymbol{c}) \\ \boldsymbol{c}\in\mathcal{K}}}{\arg\min} \|\boldsymbol{y}\|^2 + \|\boldsymbol{x}\|^2 - 2\sum_{n\in\Delta} \Re(y_n x_n^*).$$

The terms $\|\boldsymbol{y}\|^2$ and $\|\boldsymbol{x}\|^2$ can be ignored ($\|\boldsymbol{y}\|^2$ is fixed, while $\|\boldsymbol{x}\|^2$ is the same for all codewords because of the PSK modulation), so the ML decoder can be implemented as the correlation decoder, where we compute the correlation between the complex conjugate of all possible modulated codewords and the channel output and we select the codeword $\hat{\boldsymbol{c}}$ that corresponds to the highest correlation, i.e.,

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}\in\mathcal{K}}{\arg\min} - \sum_{n\in\Delta} \Re(y_n \phi(-c_n)) = \underset{\boldsymbol{c}\in\mathcal{K}}{\arg\max}\, \Re\left(\sum_{n\in\Delta} y_n \phi(-c_n)\right),$$

where $x_n^* = \phi(-c_n)$, and the last equality follows from the fact that the sum of the real part of complex numbers is equivalent to the real part of the sum of complex numbers.

**Example 2.** *Consider the transmission of a single quaternary symbol $c = 2$ over the AWGN channel. As $c \in \mathbb{Z}_4$, we select the QPSK modulation scheme, shown in Figure 1, for transmission.*

*For simplicity, the complex numbers are represented as points in the Euclidean plane, where the first coordinate corresponds to the real part of a complex number, and the second coordinate corresponds to the imaginary part of a complex number.*

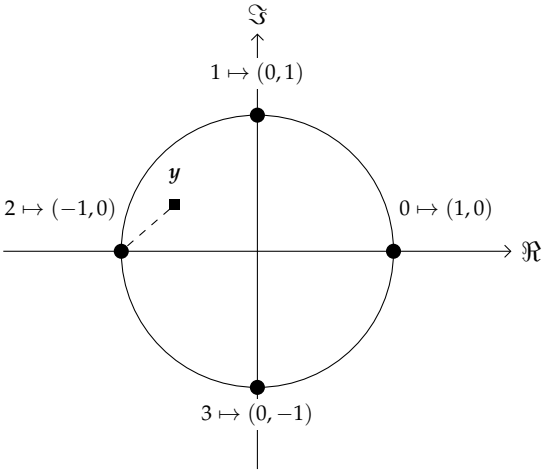


**Figure 1.** The constellation diagram of the used QPSK modulation

*The modulated symbol $\boldsymbol{x} = \phi(c) = (-1, 0)$ is sent over the AWGN channel, where Gaussian noise is added to every coordinate independently, and the channel output $\boldsymbol{y}$ is received. At the receiver, we compute the squared Euclidean distance between the channel output and every constellation point and select the closest one, i.e., we compute*

$$\hat{c} = \underset{c\in\mathbb{Z}_4}{\arg\min} \|\boldsymbol{y} - \phi(c)\|^2 = \underset{c\in\mathbb{Z}_4}{\arg\max} \langle \boldsymbol{y}, \phi(c)\rangle.$$

*Notice that when using the complex notation, this decision rule is given by the real part of the product of the complex channel output and the conjugate of a complex constellation symbol.*

The MAP decoder operates on a bitwise basis and is optimal in terms of minimizing the bit-error probability. Given the corresponding channel output $\boldsymbol{y}$, the MAP decoder proceeds to find [24,46]

$$P[c_j = \alpha|\boldsymbol{y}] = \sum_{\boldsymbol{b}\in\mathcal{C}_j^\alpha} P[\boldsymbol{b}|\boldsymbol{y}] \propto \sum_{\boldsymbol{b}\in\mathcal{C}_j^\alpha} \prod_{n=0}^{N-1} P[y_n|b_n],$$

where $\mathcal{C}_j^\alpha \subset \mathcal{K}$ represents the set of all codewords that have symbol $\alpha \in \mathbb{Z}_{2^S}$ in position $j$, for all $j = 0, \ldots, N-1$. The MAP decision is by definition

$$\hat{c}_n = \arg\max_{\alpha\in\mathbb{Z}_{2^S}} P[c_n = \alpha|\boldsymbol{y}].$$

The MAP decoder is well suited for use in a concatenated coding scheme, where we exchange soft messages between different coding blocks.

## 4. Decoding Algorithms

Let $\mathcal{C} = \mathcal{K} \setminus \{\mathbf{1}\}$ be the linear subcode of the generalized Kerdock code that does not contain the all-one codeword. The size of the code $\mathcal{C}$ is $M = |\mathcal{C}| = N^S$.

Let $\boldsymbol{c}_m = (c_{m,-\infty}, c_{m,0}, \ldots, c_{m,N-2}) \in \mathcal{C}$ be the codeword corresponding to the information sequence given by the components of the polynomial $\lambda \in GR(2^S, d)$. Then,

$$c_{m,n} = T_S(\lambda\xi^{\pi_n}) = T_S(\lambda^{(S-2)}\xi^{\pi_n}) + 2^{S-1}\cdot T(\xi_0^{r_{S-1}+\pi_n}) = c_{m,n}^{(S-2)} + 2^{S-1}\cdot c_{m,n}^{(0)}, \tag{12}$$

where $r_{S-1}, n \in \Delta$. Notice that the vector $\boldsymbol{c}_m^{(0)}$ is a codeword of the $RM(1,d)$-associated code of $\mathcal{K}$, i.e., any codeword of $RM(1,d)$, scaled by $2^{S-1}$, is also a codeword of $\mathcal{K}$. As all codewords of the $\mathbb{Z}_{2^S}$ linear generalized Kerdock code form an additive Abelian group, then

$$\boldsymbol{c}_l^{(S-2)} + 2^{S-1}\cdot\boldsymbol{c}_n^{(0)} \in \mathcal{K}, \tag{13}$$

for all $l \in \{0, \ldots, N^{S-1}\}$ and $n \in \{0, \ldots, N-1\}$.

For convenience, we define three auxiliary matrices, $A_0 \in \mathbb{Z}_{2^S}^{2^S\times N}$, $A_1 \in \mathbb{Z}_{2^S}^{N^{S-1}\times N}$, and $A_2 \in \mathbb{Z}_{2^S}^{N\times N}$, together with their row sets (sets of row elements),

$$\mathcal{A}_0 = \{\alpha\mathbf{1} \mid \alpha \in \mathbb{Z}_{2^S}\},\ |\mathcal{A}_0| = 2^S,$$

$$\mathcal{A}_1 = \{\boldsymbol{c}^{(S-2)} \mid \boldsymbol{c} \in \mathcal{C}\},\ |\mathcal{A}_1| = N^{S-1},$$

and

$$\mathcal{A}_2 = \{2^{S-1}\cdot\boldsymbol{c}^{(0)} \mid \boldsymbol{c} \in \mathcal{C}\}$$

Note that

$$\mathcal{K} = \{\boldsymbol{a}^0 + \boldsymbol{a}^1 + \boldsymbol{a}^2 \mid \boldsymbol{a}^0 \in \mathcal{A}^0, \boldsymbol{a}^1 \in \mathcal{A}^1, \boldsymbol{a}^2 \in \mathcal{A}^2\},$$

and

$$\mathcal{C} = \{\boldsymbol{a}^1 + \boldsymbol{a}^2 \mid \boldsymbol{a}^1 \in \mathcal{A}^1, \boldsymbol{a}^2 \in \mathcal{A}^2\}.$$

The rows of the matrix $A_2$ are generated as codewords of the $RM(1,d)$ code, scaled by $2^{S-1}$.

### *4.1. ML Decoding Algorithm*

The ML decoder begins by calculating the correlation coefficient of the received channel output, $\boldsymbol{y} = (y_{-\infty}, y_0, \ldots, y_{N-2})$, with the complex conjugate of each possible modulated codeword. After that, the decoder selects the codeword with the largest correlation coefficient.

For every possible information pair $(\lambda, \epsilon)$, let the correlation coefficient between the corresponding codeword $(\boldsymbol{c})$ and the channel output be defined as

$$\rho(\lambda, \epsilon) = \sum_{n \in \Delta} y_n \phi(-c_n). \tag{14}$$

There are $N^S$ possible information pairs, and the computation of the correlation coefficient, using Equation (14), has complexity $\mathcal{O}(N)$. The total complexity, when using the brute-force approach, is $\mathcal{O}(N^{S+1})$.

After substituting Equation (6) in Equation (14), we obtain

$$\rho(\lambda, \epsilon) = \sum_{n \in \Delta} y_n \phi(-T_S(\lambda \xi^{\pi_n}) - \epsilon) = \phi(-\epsilon) \sum_{n \in \Delta} y_n \phi(-T_S(\lambda \xi^{\pi_n})),$$

where the last equality follows from (9). Using the definition in (12), we obtain

$$\rho(\lambda, \epsilon) = \phi(-\epsilon) \sum_{n \in \Delta} y_n \phi(-c_n) = \phi(-\epsilon) \sum_{n \in \Delta} y_n \phi(-c_n^{(S-2)}) \phi(-2^{S-1} \cdot c_n^{(0)}),$$

Let $x_n = \phi(-c_n^{(S-2)})$ and

$$a_n = \phi\left(-2^{S-1} \cdot c_n^{(0)}\right) = \exp\left\{-I \frac{2\mu}{2^S} 2^{S-1} \cdot c_n^{(0)}\right\} = (-1)^{c_n^{(0)}}.$$

Finally, the correlation coefficient is given by

$$\rho(\lambda, \epsilon) = \phi(-\epsilon) \sum_{n \in \Delta} y_n x_n a_n = \phi(-\epsilon) \sum_{n \in \Delta} y_n x_n (-1)^{c_n^{(0)}}.$$

The correlation coefficient can be viewed as $\phi(-\epsilon)$ times the fast Hadamard transform (FHT) [2,24] of the vector $\boldsymbol{y} \odot \boldsymbol{x}$, $\boldsymbol{x} = (x_{-\infty}, x_0, \ldots, x_{N-2})$. The complexity of applying the FHT to a vector of length $N$ is $\mathcal{O}(N \log_2 N)$ (note that we compute $N$ correlation coefficients in parallel using the FHT), and we need to compute the FHT for every possible vector $\boldsymbol{y} \odot \boldsymbol{x}$.

After this, we search for the correlation coefficient with the largest real part [2], and we output the corresponding codeword. This concludes the algorithm.

**Implementation of the ML decoder.** Here, we provide a brief rundown of the primary steps in the ML algorithm. Let $X_1 = \phi(-A_1)$ be a complex matrix of size $N^{S-1} \times N$, with rows $\boldsymbol{x}_l$, $l \in \{0, \ldots, N^{S-1}\}$. Furthermore, let $H_N = \phi(-A_2)$ be the $N \times N$ Sylvester-type Hadamard matrix [26].

Given the channel output $\boldsymbol{y}$, the ML decoder first generates the correlation matrix $R \in \mathbb{R}^{M \cdot N^{S-1} \times N}$, with rows

$$\boldsymbol{r}_i = \Re(\phi(-\epsilon) \cdot \boldsymbol{y} \odot \boldsymbol{x}_l \cdot H_m) = \Re(\phi(-\epsilon) \cdot \mathrm{FHT}(\boldsymbol{y} \odot \boldsymbol{x}_l)),$$

where $\Re$ is applied componentwise and the row index $i$ is calculated as $i = \epsilon N^{S-1} + l$, $\epsilon \in \mathbb{Z}_{2^S}$, $l \in \{0, \ldots, N^{S-1}\}$. Matrix $R$ can be generated with complexity $\mathcal{O}(N^S \log_2 N)$, as there are $N^{S-1}$ vectors for which we have to compute $N$ correlation coefficients using the FHT. Let $r_{j,k}^*$ be the largest correlation coefficient, which can be found with complexity $\mathcal{O}(N^S)$. Then, the ML decoder outputs the codeword

$$\hat{\boldsymbol{c}} = \boldsymbol{a}^0_{j \operatorname{div} N^{S-1}} + \boldsymbol{a}^1_{j \bmod N^{S-1}} + \boldsymbol{a}^2_k,$$

where $\boldsymbol{a}^0_{j \operatorname{div} N^{S-1}} \in \mathcal{A}_0$, $\boldsymbol{a}^1_{j \bmod N^{S-1}} \in \mathcal{A}_1$, and $\boldsymbol{a}^3_k \in \mathcal{A}_2$. It is clear that the complexity of this decoder is $\mathcal{O}(N^S \log_2 N)$.

*4.2. MAP Decoding Algorithm*

The MAP decoding algorithm follows the group algebra description of the MAP decoder introduced in [46] and used in [24] to develop the MAP decoder of the quaternary Kerdock and Preparata codes.

Similarly as in [24], let $\mathbb{S} = \mathbb{R}[Z]/\langle Z^{2^S} - 1\rangle$ be a set of all polynomials over $\mathbb{R}$, modulo $Z^{2^S} - 1$, where $Z$ is a dummy variable. Given a polynomial $f(Z) = \sum_{\alpha\in\mathbb{Z}_{2^S}} f_\alpha Z^\alpha \in \mathbb{S}$ and a monomial $Z^{-\beta} \in \mathbb{S}$, $\beta \in \mathbb{Z}_{2^S}$, using a simple change of variable, we have

$$f(Z)\cdot Z^{-\beta} = \sum_{\alpha\in\mathbb{Z}_{2^S}} f_\alpha Z^{\alpha-\beta} = \sum_{\alpha\in\mathbb{Z}_{2^S}} f_{\alpha+\beta} Z^\alpha \in \mathbb{S}.$$

We start the decoding process by calculating the vector of log-likelihoods $\boldsymbol{w} \in \mathbb{S}^N$, with

$$w_n(Z) = \sum_{\alpha\in\mathbb{Z}_{2^S}} \ln P[y_n|\alpha] Z^\alpha, \tag{15}$$

where $y_n$ is the $n$-th component of the channel output $\boldsymbol{y}$.

For every codeword $\boldsymbol{c}_m \in \mathcal{C}$, $m \in \{0,\ldots,M-1\}$, we compute

$$t_m(Z) = \sum_{n\in\Delta} w_n(Z)\cdot Z^{-c_{m,n}}$$

Notice that

$$\begin{aligned} t_m(Z) &= \sum_{n\in\Delta} Z^{-c_{m,n}} \sum_{\alpha\in\mathbb{Z}_{2^S}} \ln P[y_n|\alpha] Z^\alpha = \sum_{\alpha\in\mathbb{Z}_{2^S}} \sum_{n\in\Delta} \ln P[y_n|\alpha] Z^{\alpha-c_{m,n}} \\ &= \sum_{\alpha\in\mathbb{Z}_{2^S}} \sum_{n\in\Delta} \ln P[y_n|\alpha+c_{m,n}] Z^\alpha = \sum_{\alpha\in\mathbb{Z}_{2^S}} Z^\alpha \ln \prod_{n\in\Delta} P[y_n|\alpha+c_{m,n}] \\ &= \sum_{\alpha\in\mathbb{Z}_{2^S}} Z^\alpha \ln P[\boldsymbol{y}|\boldsymbol{c}_m+\alpha\boldsymbol{1}], \end{aligned}$$

where $(\boldsymbol{c}_m + \alpha\boldsymbol{1}) \in \mathcal{K}$. Alternatively, using the definition in (12), we have

$$\begin{aligned} t_m(Z) &= \sum_{n\in\Delta} \sum_{\alpha\in\mathbb{Z}_{2^S}} \ln P[y_n|\alpha] Z^{\alpha-(c_{m,n}^{(S-2)}+2^{S-1}c_{m,n}^{(0)})} \\ &= \sum_{n\in\Delta} \sum_{\alpha\in\mathbb{Z}_{2^S}} \ln P[y_n|\alpha] Z^{\alpha-c_{m,n}^{(S-2)}} Z^{-2^{S-1}c_{m,n}^{(0)}} \\ &= \sum_{n\in\Delta} \left( \sum_{\alpha\in\mathbb{Z}_{2^S}} \ln P[y_n|\alpha+c_{m,n}^{(S-2)}] \right) Z^{-2^{S-1}c_{m,n}^{(0)}}. \end{aligned} \tag{16}$$

As $\boldsymbol{c}_m^{(0)}$ is a codeword of the $RM(1,d)$ code, the expression in (16) can be interpreted as applying a modified FHT to every vector $\boldsymbol{b}_m = \boldsymbol{w} \odot \boldsymbol{x}_m$, where

$$\boldsymbol{x}_m = (Z^{-c_{m,-\infty}}, Z^{-c_{m,0}}, Z^{-c_{m,1}}, \ldots, Z^{-c_{m,N-2}}).$$

This allows us to compute $N$ different values of $t_m$ in parallel, i.e., instead of computing $N^S$ values, where every computation requires $\mathcal{O}(N)$ operations in $\mathbb{S}$, we apply the modified FHT (with complexity $\mathcal{O}(N\log_2 N)$) to $N^{S-1}$ vectors, so the total complexity is reduced from $\mathcal{O}(N^{S+1})$ to $\mathcal{O}(N^S \log_2 N)$. By the modified FHT, we assume a fast Hadamard transform applied in $\mathbb{S}$, which is defined in Algorithm 1. In this pseudocode, $h$ is used as a control variable for the loop, representing the current step size in the modified FHT algorithm. Variable $h$ starts at 1 and doubles each time, determining the pairs of elements to be combined at each stage of the transform. This ensures that first the adjacent elements

are paired (the distance between them is $h = 1$), and as it progressively increases, so does the distance between the pairs of elements that are combined.

**Algorithm 1** Modified fast Hadamard transform (in-place implementation)

**Input:** $\boldsymbol{x}$,
**Output:** $\boldsymbol{x}$,

**for** $h = 1; h < N; h \leftarrow 2 \cdot h$ **do**
  **for** $i = 0; i < N; i \leftarrow i + 2 \cdot h$ **do**
    **for** $j = i$ **to** $i + h$ **do**
      $a \leftarrow x_j$, $b \leftarrow x_{j+h}$, {Save current values}
      $x_j \leftarrow a + b$, {Calculate new $x_j$ value}
      $x_{j+h} \leftarrow a + b \cdot Z^{2^{S-1}}$, {Calculate new $x_{j+h}$ value}
    **end for**
  **end for**
**end for**

Next, for every $t_m$, we recompute the sums of logarithms of probabilities into products of probabilities, as follows:

$$v_m(Z) = \sum_{\alpha \in \mathbb{Z}_{2^S}} \exp\{t_{m,\alpha}\} Z^{\alpha} = \sum_{\alpha \in \mathbb{Z}_{2^S}} P[\boldsymbol{y} | \boldsymbol{c}_m + \alpha \mathbf{1}] Z^{\alpha}.$$

Finally, we compute

$$\begin{aligned} s_n(Z) &= \sum_{m=0}^{M-1} Z^{c_{m,n}} \sum_{\alpha \in \mathbb{Z}_{2^S}} v_m(Z) Z^{\alpha} = \sum_{\alpha \in \mathbb{Z}_{2^S}} \sum_{m=0}^{M-1} P[\boldsymbol{y} | \boldsymbol{c}_m + \mathbf{1}\alpha] Z^{\alpha + c_{m,n}} \\ &= \sum_{\alpha \in \mathbb{Z}_{2^S}} \sum_{m=0}^{M-1} \left( \prod_{n \in \Delta} P[y_n | \alpha + c_{m,n}] \right) Z^{\alpha + c_{m,n}} = \sum_{\alpha \in \mathbb{Z}_{2^S}} \sum_{\boldsymbol{c} \in C_n^{\alpha}} P[\boldsymbol{y} | \boldsymbol{c}] Z^{\alpha}, \end{aligned}$$

where the last equality follows from the fact that

$$\begin{aligned} &P[y_{-\infty} | \alpha + c_{m,-\infty}] \cdots P[y_n | \alpha + c_{m,n}] \cdots P[y_{N-2} | \alpha + c_{m,N-2}] Z^{\alpha + c_{m,n}} = \\ &P[y_{-\infty} | \alpha + c_{m,-\infty}] \cdots P[y_n | \alpha] \cdots P[y_{N-2} | \alpha + c_{m,N-2}] Z^{\alpha}. \end{aligned}$$

The complexity of computing one of the $N$ components of $\boldsymbol{s}$ is $\mathcal{O}(N^S)$, so the total complexity of computing $\boldsymbol{s}$ is $\mathcal{O}(N^{S+1})$. We can reduce the complexity of this step in a similar way as before. Notice that

$$s_n(Z) = \sum_{\alpha \in \mathbb{Z}_{2^S}} \sum_{m=0}^{M/N-1} \sum_{l=0}^{N-1} P[\boldsymbol{y} | \boldsymbol{c}_m + \mathbf{1}\alpha] Z^{c_{m,n}^{(S-2)}} Z^{2^{S-1} c_{l,n}^{(0)}},$$

which follows from (13). Notice that the computation of $\boldsymbol{s}(Z)$ can now be interpreted as applying the modified FHT $N^{S-1}$ times and then performing a componentwise addition of the results. The complexity of this approach is now $\mathcal{O}(N^S \log_2 N)$. This completes the algorithm.

**Implementation of the MAP decoder.** We now present a summary of the essential steps in the MAP algorithm. For convenience, we define a mapping $\zeta : \mathbb{Z}_{2^S} \to \mathbb{S}$, such that $\zeta(a) = Z^a$. This function can be extended to elements and matrices, by applying $\zeta(\cdot)$ to every component independently [24]. Let $D_1 = \zeta(A_1)$ and $\overline{D}_1 = \zeta(A_1)$.

We begin the decoding procedure by computing the log-likelihood vector $\boldsymbol{w} \in \mathbb{S}^N$, using Equation (15). This can be accomplished with complexity $\mathcal{O}(N)$.

Next, we compute matrix $T \in \mathbb{S}^{N^{S-1} \times N}$ with rows

$$\boldsymbol{t}_l = \mathrm{FHT}(\boldsymbol{w} \odot \overline{\boldsymbol{d}}_l),\ l \in \{0, \ldots, N^{S-1}\},$$

where $\overline{\boldsymbol{d}}_l$ represents the $l$-th row of matrix $\overline{D}$. Matrix $T$ can be computed in $\mathcal{O}(N^S \log_2 N)$ steps, as we need to apply the modified FHT to $N^{S-1}$ vectors.

Next, we compute the likelihood matrix $V \in \mathbb{S}^{N^{S-1} \times N}$, with components

$$v_{l,n} = \sum_{\alpha \in \mathbb{Z}_{2^S}} Z^{\alpha} \exp\{t_{l,n,\alpha}\}.$$

This can be accomplished in $\mathcal{O}(N^S)$ steps.

Let $B$ be an $N^{S-1} \times N$ matrix, with rows

$$\boldsymbol{b}_l = \mathrm{FHT}(\boldsymbol{v}) \odot \boldsymbol{d}_l,$$

where $l \in \{0, \ldots, N^{S-1}\}$. This can be accomplished with complexity $\mathcal{O}(N^S \log_2 N)$.

Finally, we form vector $\boldsymbol{s} \in \mathbb{S}^N$ by summing all rows of $B$,

$$s_n = \sum_{l=0}^{N^{S-1}} b_{l,n},\ n \in \{0, \ldots N-1\}.$$

This can be performed in $\mathcal{O}(N^S)$ steps. With this, we finish the algorithm description. It is important to note that all the operations here are conducted over the polynomial ring $\mathbb{S}$. As the polynomials in $\mathbb{S}$ can be realized as real vectors of the fixed length of $2^S$ (where $2^S$ is less than $N$ in most practical use cases), the additional complexity can be treated as a constant term, so the total complexity of this algorithm is $\mathcal{O}(N^S \log_2 N)$ [24].

## 5. Simulation Results

In this section, we present the simulation results of our novel decoding algorithms and compare them to the existing techniques in terms of their error-correcting performance. As these decoders are polynomial algorithms, we limit ourselves to short-length (where code length is given as the number of $\mathbb{Z}_{2^S}$ symbols) codes over $\mathbb{Z}_4$, $\mathbb{Z}_8$, and $\mathbb{Z}_{16}$. Let $\mathcal{K}_S^d$ represent a generalized Kerdock code of length $2^d$, over the ring $\mathbb{Z}_{2^S}$, defined in terms of the irreducible monic polynomial ($h_S(z)$) used to design an extension ring, as presented in Section 3. We also include some classical Kerdock codes, as they also fall into the family of the generalized Kerdock code. These codes are the $\mathcal{K}_2^3$ code defined by $h_2(Z) = 3 + Z + 2Z^2 + Z^3$, the $\mathcal{K}_2^5$ code defined by $h_2(Z) = 3 + 2Z + 3Z^2 + Z^5$, the $\mathcal{K}_2^7$ code defined by $h_2(Z) = 3 + Z + 2Z^4 + Z^7$, the $\mathcal{K}_3^3$ code defined by $h_3(Z) = 7 + 5Z + 6Z^2 + Z^3$, the $\mathcal{K}_3^5$ code defined by $h_3(Z) = 7 + 2Z + 7Z^2 + 4Z^3 + Z^5$, and the $\mathcal{K}_4^3$ code defined by $h_4(Z) = 15 + 5Z + 6Z^2 + 1$.

All simulations were conducted for the additive white Gaussian noise (AWGN) channel. Additionally, every code was coupled with a corresponding $2^S$-PSK modulation, as there exists a natural straightforward mapping of code symbols onto the PSK symbols (Equation (8)). We assessed the performance of our decoding algorithms by estimating the frame error rate (FER) and symbol error rate (SER) as a function of energy per bit to noise power spectral density ratio ($E_b/N_0$) and comparing it to the classical lifting decoder [27]. The classical lifting decoder can be applied to generalized Kerdock codes without modification, but it has a higher error probability, as it is a suboptimal hard-input hard-output (HIHO) decoder. This decoder was implemented as an $S$-stage decoder, where each stage uses the minimum Hamming distance decoder of the associated binary $RM(1, d)$ code. As the minimum Hamming distance decoder of the $RM(1, d)$ code can be implemented with a complexity of $\mathcal{O}(N \log_2 N)$, the total complexity of the classical lifting decoder is $\mathcal{O}(SN \log_2 N)$. This complexity is significantly lower than that of the novel algorithms developed in this paper, but so is its error-correcting performance. FER and SER were

estimated using the Monte Carlo simulation method with a relative precession $\delta = 0.05$, for a range of $Eb/N_0$ points, with a step of 0.5 dB.

We assume that the ML decoder will make a mistake if the correct codeword is further away from the channel output than some other codeword (e.g., the output of some not-necessarily-optimal decoding algorithm) in terms of the Euclidean distance [24]. We can simulate the lower bound on the ML decoding by counting the number of times the Euclidean distance between the channel output and the correct codeword is greater than the distance between the channel output and the decoded codeword. This bound becomes tight as $E_b/N_0$ increases. We compare the FER performance of our novel ML and MAP decoders with the ML bound (Figure 2) and see that they perfectly coincide. This indicates that our novel algorithms are optimal in terms of FER, as was expected.

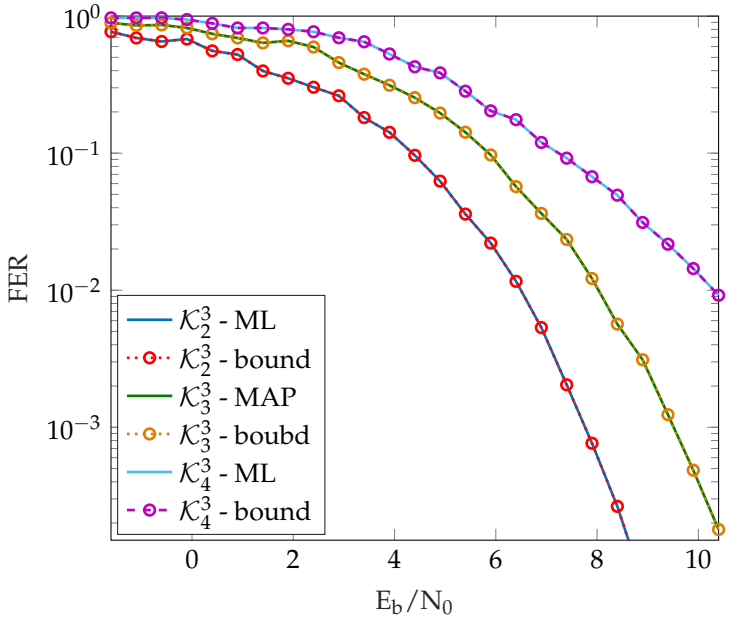


**Figure 2.** FER comparison of the ML decoding algorithm with an ML lower bound for different generalized Kerdock codes.

Figure 3 presents the FER of various generalized Kerdock codes using the novel ML decoder, and Figure 4 shows the SER of different generalized Kerdock codes using the novel MAP decoder. We notice that the error-correcting performance improves with code length, while it decreases with base ring size. This is expected behavior, as the error-correcting performance of the code should increase with $N$, while the increase in the constellation size means that the modulation symbols are closer together and are more susceptible to noise, and this leads to an increase in the probability of error.

Figure 5 shows the SER of different generalized Kerdock codes using the novel MAP decoder and the classical lifting decoder. We see in Figure 5 that the MAP decoder exhibits a gain of some 5 dB when compared to the classical lifting decoder. This is expected, as the novel MAP decoder is an optimal SISO decoding algorithm, while the lifting decoder is a suboptimal HIHO decoding algorithm.

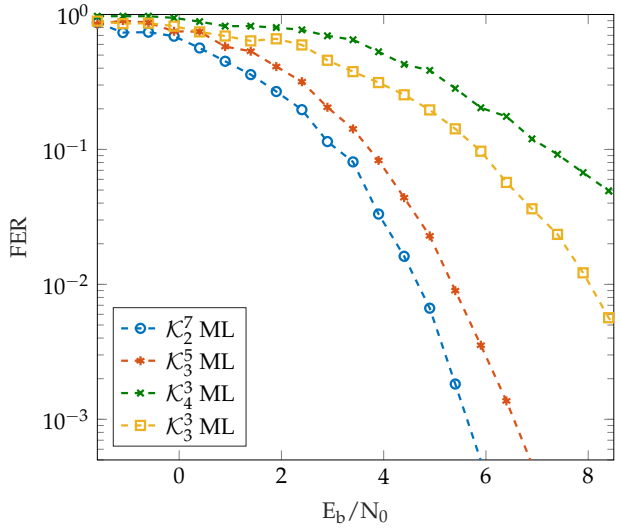


**Figure 3.** FER comparison of the ML decoding for different generalized Kerdock codes.

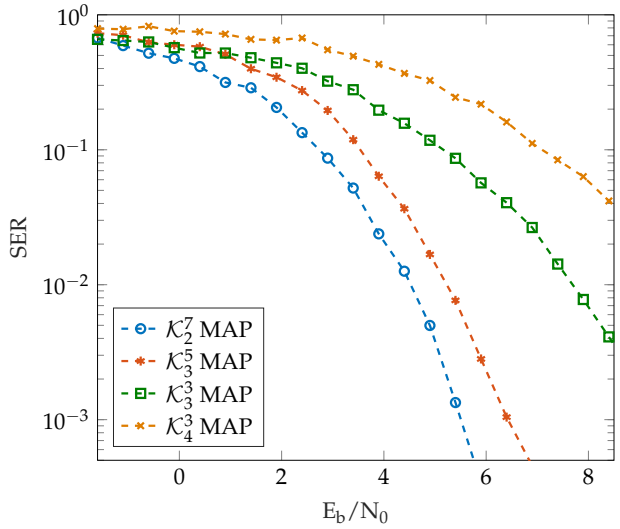


**Figure 4.** SER comparison of the MAP decoding for different generalized Kerdock codes.

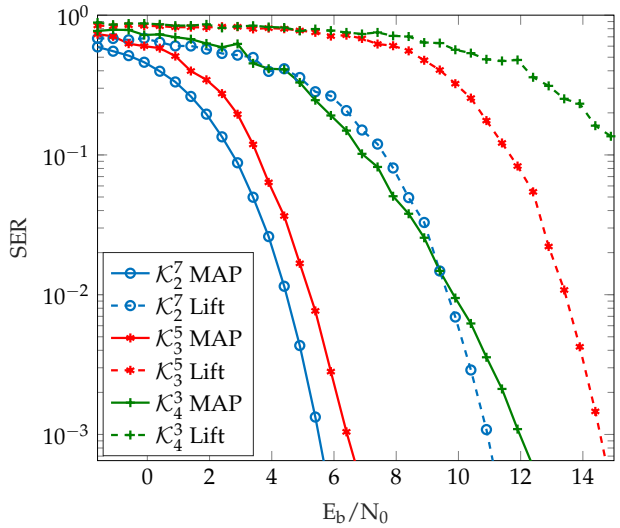


**Figure 5.** SER comparison of the MAP and the lifting decoder algorithms for different generalized Kerdock codes.

## 6. Conclusions

In this manuscript, we presented two novel optimal algorithms for the decoding of generalized Kerdock codes. Their complexity is $\mathcal{O}(N^S \log_2 N)$, and although they are polynomial complexity algorithms, they can be used as a starting point for developing and a benchmark for evaluating the performance of suboptimal decoders. Furthermore, as the MAP decoder is an SISO algorithm, it allows the use of small-length generalized Kerdock codes in modern coding systems as component codes. The novel decoding algorithms were compared with the classical lifting decoding algorithm in terms of error-correcting performance, and it was shown that they achieved a gain of about 5 dB. In our future work, we will focus on reducing the complexity below that of the MAP decoder without significantly compromising its error-correcting efficiency. The lifting technique is a powerful technique that can be combined with the MAP decoder and used to develop novel suboptimal SISO decoders. We will also use generalized Kerdock codes (and other related codes) as components in modern coding schemes, such as the turbo-product scheme, braided construction, coded modulation schemes, and many others.

**Author Contributions:** All authors have contributed equally to the paper. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was in part supported by the European Union Horizon Europe research and innovation program under the grant agreement No. 101086387 and the Secretariat for Higher Education and Scientific Research of the Autonomous Province of Vojvodina through the project "Visible light technologies for indoor sensing, localization and communication in smart buildings" (142-451-3511/2023).

**Data Availability Statement:** No new data were created or analyzed in this study. Data sharing is not applicable to this article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Massey, J.; Mittelholzer, T. Convolutional codes over rings. In Proceedings of the 4th joint Swedish-Soviet International Workshop on Information Theory, Gotland, Sweden, 27 August–1 September 1989.
2. Hammons, A.R.; Kumar, P.V.; Calderbank, A.R.; Sloane, N.J.A.; Sole, P. The Z/sub 4/-linearity of Kerdock, Preparata, Goethals, and related codes. *IEEE Trans. Inf. Theory* **1994**, *40*, 301–319. [CrossRef]
3. Solé, P. Open problem 2: Cyclic codes over rings and p-adic fields. In *Coding Theory and Applications, Proceedings of the Coding Theory 1988, Toulon, France, 2–4 November 1988*; Cohen, G., Wolfmann, J., Eds.; Springer: Berlin/Heidelberg, Germany, 1989; p. 329.
4. Calderbank, A.R.; Sloane, N.J.A. Modular and p-adic cyclic codes. *Des. Codes Cryptogr.* **1995**, *6*, 21–35. [CrossRef]
5. Ling, S.; Sole, P. Duadic codes over Z/sub 2k/. *IEEE Trans. Inf. Theory* **2001**, *47*, 1581–1588. [CrossRef]
6. Gulliver, T.; Harada, M. Double circulant self-dual codes over Z/sub 2k/. *IEEE Trans. Inf. Theory* **1998**, *44*, 3105–3123. [CrossRef]
7. Carlet, C. $\mathbb{Z}_{2^k}$-linear codes. *IEEE Trans. Inf. Theory* **1998**, *44*, 1543–1547. [CrossRef]
8. Mittelholzer, T. Convolutional codes over rings and the two chain conditions. In Proceedings of the IEEE International Symposium on Information Theory, Ulm, Germany, 29 June–4 July 1997; p. 285. [CrossRef]
9. Napp, D.; Pinto, R.; Toste, M. Column Distances of Convolutional Codes Over $\mathbb{Z}_{p^r}$. *IEEE Trans. Inf. Theory* **2019**, *65*, 1063–1071. [CrossRef]
10. Sridhara, D.; Fuja, T. LDPC codes over rings for PSK modulation. *IEEE Trans. Inf. Theory* **2005**, *51*, 3209–3220. [CrossRef]
11. Ninacs, T.; Matuz, B.; Liva, G.; Colavolpe, G. Non-binary LDPC coded DPSK modulation for phase noise channels. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6. [CrossRef]
12. Ninacs, T.; Matuz, B.; Liva, G.; Colavolpe, G. Short Non-Binary Low-Density Parity-Check Codes for Phase Noise Channels. *IEEE Trans. Commun.* **2019**, *67*, 4575–4584. [CrossRef]
13. Davis, J.A.; Jedwab, J. Peak-to-mean power control in OFDM, Golay complementary sequences, and Reed-Muller codes. *IEEE Trans. Inf. Theory* **1999**, *45*, 2397–2417. [CrossRef]
14. Schmidt, K. Quaternary Constant-Amplitude Codes for Multicode CDMA. *IEEE Int. Symp. Inf. Theory* **2007**, *55*, 1824–1832. [CrossRef]
15. Shakeel, I. Performance of Reed-Muller and Kerdock Coded MC-CDMA System with Nonlinear Amplifier. In Proceedings of the 2005 Asia-Pacific Conference on Communications, Perth, Australia, 3–5 October 2005; pp. 644–648.
16. Amrani, O. Nonlinear Codes: The Product Construction. *IEEE Trans. Commun.* **2007**, *55*, 1845–1851. [CrossRef]
17. Karp, B.; Amrani, O.; Keren, O. Nonlinear Product Codes for Reliability and Security. In Proceedings of the 2019 IEEE 4th International Verification and Security Workshop (IVSW), Rhodes, Greece, 3 October 2019; pp. 13–18.

18. Inoue, T.; Heath, R.W. Kerdock codes for limited feedback MIMO systems. In Proceedings of the 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, USA, 31 March–4 April 2008; pp. 3113–3116.
19. Inoue, T.; Heath, R.W., Jr. Kerdock Codes for Limited Feedback Precoded MIMO Systems. *IEEE Trans. Signal Process.* **2009**, *57*, 3711–3716. [CrossRef]
20. Güzeltepe, M.; Sarı, M. Quantum codes from codes over the ring $\mathbb{F}_q + \alpha\mathbb{F}_q$. *Quantum Inf. Process.* **2019**, *18*, 365. [CrossRef]
21. Güzeltepe, M.; Aytaç, N. Quantum Codes from Codes over the Ring Rq. *Int. J. Theor. Phys.* **2023**, *62*, 26. [CrossRef]
22. Kim, H.; Markarian, G.; Rocha, V.C.d., Jr. Nonlinear Product Codes and Their Low Complexity Iterative Decoding. *ETRI J.* **2010**, *32*, 588–595. [CrossRef]
23. Kim, H.; Markarian, G.; da Rocha, V.C., Jr. Nonlinear turbo product codes. In Proceedings of the XXV Simpósio Brasileiro de Telecomunicações, Recife, PE, Brazil, 3–6 September 2007.
24. Minja, A.; Šenk, V. SISO Decoding of $\mathbb{Z}_4$ Linear Kerdock and Preparata Codes. *IEEE Trans. Commun.* **2022**, *70*, 1497–1507. [CrossRef]
25. Barrolleta, R.; Pujol, J.; Villanueva, M. Comparing decoding methods for quaternary linear codes. *Electron. Notes Discret. Math.* **2016**, *54*, 283–288. [CrossRef]
26. MacWilliams, F.; Sloane, N. *The Theory of Error-Correcting Codes*; Mathematical Studies; Elsevier Science: Amsterdam, The Netherlands, 1977.
27. Greferath, M.; Vellbinger, U. Efficient decoding of Z/sub p/k-linear codes. *IEEE Trans. Inf. Theory* **1998**, *44*, 1288–1291. [CrossRef]
28. Babu, N.S.; Zimmermann, K. Decoding of linear codes over Galois rings. *IEEE Trans. Inf. Theory* **2001**, *47*, 1599–1603. [CrossRef]
29. Conway, J.; Sloane, N. Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice. *IEEE Trans. Inf. Theory* **1986**, *32*, 41–50. [CrossRef]
30. Barrolleta, R.D.; Villanueva, M. Partial permutation decoding for binary linear and $Z_4$-linear Hadamard codes. *Des. Codes Cryptogr.* **2018**, *86*, 569–586. [CrossRef]
31. Barrolleta, R.D.; Villanueva, M. Partial Permutation Decoding for Several Families of Linear and $Z_4$-Linear Codes. *IEEE Trans. Inf. Theory* **2019**, *65*, 131–141. [CrossRef]
32. Helleseth, T.; Kumar, P.V. The algebraic decoding of the Z/sub 4/-linear Goethals code. *IEEE Trans. Inf. Theory* **1995**, *41*, 2040–2048. [CrossRef]
33. Byrne, E.; Fitzpatrick, P. Hamming metric decoding of alternant codes over Galois rings. *IEEE Trans. Inf. Theory* **2002**, *48*, 683–694. [CrossRef]
34. Rong, C.; Helleseth, T.; Lahtonen, J. On algebraic decoding of the Z/sub 4/-linear Calderbank-McGuire code. *IEEE Trans. Inf. Theory* **1999**, *45*, 1423–1434. [CrossRef]
35. Ranto, K. On algebraic decoding of the Z/sub 4/-linear Goethals-like codes. *IEEE Trans. Inf. Theory* **2000**, *46*, 2193–2197. [CrossRef]
36. Armand, M.A. Chase decoding of linear Z/sub 4/ codes. *Electron. Lett.* **2006**, *42*, 1049–1050. [CrossRef]
37. Armand, M.A.; Halim, A.; Nallanathan, A. Chase Decoding of Linear Z4 Codes at Low to Moderate Rates. *IEEE Commun. Lett.* **2007**, *11*, 811–813. [CrossRef]
38. Elia, M.; Losana, C.; Neri, F. Note on the complete decoding of Kerdock codes. *IEE Proc. I Commun. Speech Vis.* **1992**, *139*, 24–28. [CrossRef]
39. Bahl, L.; Cocke, J.; Jelinek, F.; Raviv, J. Optimal decoding of linear codes for minimizing symbol error rate (Corresp.). *IEEE Trans. Inf. Theory* **1974**, *20*, 284–287. [CrossRef]
40. Shany, Y.; Reuven, I.; Be'ery, Y. On the trellis representation of the Delsarte-Goethals codes. *IEEE Trans. Inf. Theory* **1998**, *44*, 1547–1554. [CrossRef]
41. Shany, Y.; Be'ery, Y. The Preparata and Goethals codes: Trellis complexity and twisted squaring constructions. *IEEE Trans. Inf. Theory* **1999**, *45*, 1667–1673. [CrossRef]
42. Davey, M.; MacKay, D. Low density parity check codes over GF(q). In Proceedings of the 1998 Information Theory Workshop (Cat. No.98EX131), Killarney, Ireland, 22–26 June 1998; pp. 70–71. [CrossRef]
43. Bini, G.; Flamini, F. *Finite Commutative Rings and Their Applications*; The Springer International Series in Engineering and Computer Science; Springer: New York, NY, USA, 2002.
44. Sison, V. Bases of the Galois Ring $GR(p^r, m)$ over the Integer Ring $Z_{p^r}$. *arXiv* **2014**, arXiv:cs.IT/1410.0289.
45. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
46. Ashikhmin, A.; Litsyn, S. Simple MAP decoding of first-order Reed-Muller and Hamming codes. *IEEE Trans. Inf. Theory* **2004**, *50*, 1812–1818. [CrossRef]