AN ABSTRACT OF THE THESIS OF

Luca Tallini for the degree of Master of Science in Computer Science presented on June 2, 1994.

Title: Design of Some New Efficient Balanced Codes

Abstract approved:_____ _____

Dr. Bella Bose

A balanced code with r check bits and k information bits is a binary code of length $k+r$ and cardinality 2^k such that each codeword is balanced; that is, it has $\lceil \frac{k+r}{2} \rceil$ 1's and $\lfloor \frac{k+r}{2} \rfloor$ 0's. This thesis contains new methods to construct efficient balanced codes based on the concept of tail-map. A tail-map is an injective function from the set of the very unbalanced words to the set of the balanced words. To design balanced codes, those information words with a low number of 1's or 0's are encoded using tail-maps, while those that have almost the same number of 1's and 0's are encoded using the single maps defined by Knuth's complementation method. Three different tail-map constructions are presented. Balanced codes with r check bits and k information bits with $k \leq 2^{r+1}-2$, $k \leq 3 \cdot 2^r - 8$ and $k \leq 5 \cdot 2^r - 10r + c(r)$ ($c(r) \in \{-15, -10, -5, 0, +5\}$) are given, improving the constructions found in the literature. The tail-maps used in the first two constructions can be computed using a parallel scheme.

Design of Some New Efficient Balanced Codes

by

Luca Tallini

A THESIS

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Master of Science

Completed June 2, 1994

Commencement June 1995

APPROVED:

Redacted for Privacy

Professor of Computer Science in charge of major

Redacted for Privacy

Head of Department of Computer Science

Redacted for Privacy

Dean of Graduate School

Date Thesis is Presented: June 2, 1994

Typed by Luca Tallini for Luca Tallini.

Table of Contents

List of Tables

Chapter 1
Introduction

1.1 General Introduction

When we manipulate information we need to encode it in order to match certain physical constraints. For this purpose, we often code information when we want to transmit it or store it. We want the coding and decoding process to be computationally simple. We also want our codes to have some other desirable properties like error detection, error correction (reliable communication), secretness (cryptography), compression (data compression), and so on. For example, in transmission on optical fibers, a transmitted 1 may be received as a 1 or a 0, whereas a transmitted 0 is always received as a 0 (this is because a photon can be missed during the interception but it can never be created from nothing). We would like to detect if an error (a sent 1 received as a 0) has been committed. If we had encoded the information in such a way that every transmitted word has an equal number of 0's and 1's (i.e. is balanced) then any received word with more 0's than 1's would be detected as erroneous and for example retransmitted. Notice that it doesn't matter how many errors have been committed during transmission, we can still detect the occurrence of errors. This is a simple application of what are called balanced codes. We will mention more applications later.

In this thesis, we will give some methods for constructing balanced codes. That is, given a set of words, find efficient methods for mapping these words to the balanced words which will represent them. There are several features which makes a method efficient. The encoding from the data words to codewords has to be computationally easy. The decoding from the codewords to data words has to be computationally easy. The length of the codewords shouldn't be much larger than the length of the data words.

For example, given a binary word

$$X = x_1 x_2 \ldots x_n$$

we can encode this as the balanced binary word

$$X\overline{X} = x_1 x_2 \ldots x_n \, \overline{x_1}\,\overline{x_2} \ldots \overline{x_n},$$

where $\overline{x}$ is the complement of the bit x. Clearly the encoding and decoding, for this example, are very easy. But the length of the codeword is relatively high.

In this thesis We will give methods in which the encoding and decoding are not much harder than those in the example, and in which the codewords are much smaller than those in this example.

1.2 Applications of balanced codes

Balanced codes have many applications. In this section we will briefly mention some of them.

- Balanced codes can **detect all unidirectional errors in a data word** [BER61, FRE62, SAI91]. In the case of unidirectional errors, both $1 \longrightarrow 0$ and $0 \longrightarrow 1$ errors can occur; however, in any particular transmitted word all the errors are of the same type.

- They can be used to maintain **data integrity in write-once memory**, such as digital optical disks, where a 0 can be changed to 1 but a written 1 cannot be changed to 0 [LEI84, KNU86].

- They are useful for **state assignments in fault-tolerant and fail-safe sequential circuit design** [TOH71].

- A binary code of length n is a **conservative code** if every codeword has $\left\lfloor \frac{n}{2} \right\rfloor$ transitions. A transition occurs when two adjacent bits are complementary. These codes are useful for **data synchronization** in some communication systems, especially in optical fibers [OFE90, ALB93, ALO88]. A balanced code $\mathcal{C}$ of length n can be transformed easily to a conservative code of length $n+1$ via the function $\sigma : \mathcal{C} \longrightarrow \mathbb{Z}_2^{n+1}$ defined as follows:

$$\sigma(b_1 b_2 \ldots b_n) = c_1 c_2 \ldots c_n c_{n+1},$$

where

$$\begin{cases} c_1 = 0, \\ \\ c_i = c_{i-1} + b_{i-1} \quad \text{for } i = 2, 3, \ldots, n+1, \end{cases}$$

and where the inverse transformation is defined by

$$b_i = c_i + c_{i+1} \qquad \text{for } i = 1, 2, \ldots, n.$$

- Balanced codes are useful for fiber optics and magnetic and optical storage media [TAK76, WID83, BEG86].
- They are useful to achieve **delay insensitive communication** [VER88].
- The cryptosystem proposed in [CHO85] requires the data words to be balanced.
- They are useful to accomplish **Noise reduction in VLSI chips** [TAB90].

1.3 Definitions

In this section we will give the necessary symbols and definitions that we will use in this thesis.

We believe it is a good idea to begin a thesis on coding theory making clear what is ment with code and encoding.

Let A be a finite non-empty set, which we will call the alphabet. Let:

$$A^n \stackrel{\text{def}}{=} \{a_1 a_2 \ldots a_n : \ a_i \in A \text{ for all } i = 1, \ldots, n\}, \qquad \text{for all } n = 1, 2, \ldots .$$

An element of A^n is called word of length n over the alphabet A, whereas a code $\mathcal{C}_A$ over A is nothing but a subset of

$$A^+ \stackrel{\text{def}}{=} \bigcup_{n=1}^{\infty} A^n$$

The elements of $\mathcal{C}_A$ are called codewords. A code is called a block code of length n if $\mathcal{C}_A \subseteq A^n$, otherwise it is called a variable length code.

A code is called q-ary if it is over the q-ary alphabet

$$A = \{0, 1, \ldots, q-1\} = \mathbb{Z}_q.$$

In particular, when $q = 2$ the code is called binary.

Given a q-ary code $\mathcal{C}$ of length n, the real number ($|S|$ indicates the cardinality of a set S)

$$\tau = \frac{\log_q |\mathcal{C}|}{n}$$

is called the information rate of $\mathcal{C}$. Having fixed a certain "service" (for example: error detection, error correction, synchronization, and so on) we want a code to accomplish, one of the major aim of coding theory is to design such a code with an information rate as close as possible to 1. The parameter

$$\rho = 1 - \tau$$

is called redundancy of $\mathcal{C}$.

Given a countable set S, an encoding of S is a couple $(\mathcal{C}_A, \chi)$, where $\mathcal{C}_A$ is a code and χ is a one-to-one mapping from S to $\mathcal{C}_A$ called the encoding function.

Note that, if $|S|, |\mathcal{C}_A| < \infty$ then there exist

$$|S|!\binom{|\mathcal{C}_A|}{|S|}$$

possible encodings of S. Usually the set S is chosen to be

$$S = \{0, 1, \ldots, p-1\}^k$$

for some natural numbers k and p and is called the set of information words. In this thesis S will be

$$S = \{0, 1\}^k.$$

Now we are ready to give the definition of balanced code.

Definition 1 *A binary code $\mathcal{C}$ is a* **balanced code with r check bits and k information bits** *(briefly a DC$(k+r, k)$ or DC-free code) if and only if:*

1. *$\mathcal{C}$ is a block code of length $n = k + r$,*
2. *each word $X \in \mathcal{C}$ is balanced, i.e. it has $\left\lceil \frac{n}{2} \right\rceil$ $\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$ 1's and $\left\lfloor \frac{n}{2} \right\rfloor$ $\left(\left\lceil \frac{n}{2} \right\rceil\right)$ 0's.*
3. *$|\mathcal{C}| = 2^k$.*

In this thesis the following notation is used:

k	number of information bits,
r	number of check bits,
n	$= k + r$, code length,
IN	set of natural numbers,

$\mathbb{R}$	set of real numbers,
$\mathbb{Z}_2$	$= \{0, 1\}$,
$[a, b]$	$= \{i \in \mathbb{N} : a \leq i \leq b\}$,
(a, b)	$= \{i \in \mathbb{N} : a < i < b\}$,
$w(X)$	= weight of $X \in \mathbb{Z}_2^+$, i.e. number of 1's in X
$l(X)$	= length of $X \in \mathbb{Z}_2^+$,
$\overline{X}$	complement of $X \in \mathbb{Z}_2^+$,
$\mathcal{S}_w^k$	$= \{X \in \mathbb{Z}_2^k : w(X) = w\}$,
$\lvert\mathcal{S}\rvert$	= number of elements in the set $\mathcal{S}$,
$a\mathbb{N} + b$	$= \{i \in \mathbb{N} : i = ah + b, \quad h \in \mathbb{N}\}$,
$\mathrm{DC}(k + r, k)$	balanced code with r check bits and k information bits.

Chapter 2
Previous work and Knuth's method

2.1 Previous work

In this section we will explain what was the state-of-the-art about the topic of balanced codes before our result.

Let us recall that a balanced code with r check bits and k information bits is a binary code $\mathcal{C}$ of length $k+r$ and cardinality 2^k such that each codeword is balanced; that is, it has $\left\lceil \frac{k+r}{2} \right\rceil$ 1's and $\left\lfloor \frac{k+r}{2} \right\rfloor$ 0's. One of the open research problems mentioned in [MAC77] is to find a code $\mathcal{C}$ and a one-to-one function

$$\mathcal{E} : \{0,1\}^k \longrightarrow \mathcal{C}$$

which, together with its inverse, is very easy to compute.

In [KNU86], Knuth showed that if a balanced code with r check bits and k information bits exists, then $r > \frac{1}{2}\log_2 k + 0.326$. He has designed serial encoding schemes and both parallel and serial decoding schemes. Using r check bits, the parallel decoding scheme can code

$$k = 2^r - r - 1$$

information bits. The serial decoding scheme can code

$$k = 2^r$$

information bits. In both methods, for each given information word, some appropriate number of bits, starting from the first bit, are complemented; then a check is assigned to this modified information word to make the entire word balanced. In the sequential decoding scheme the check represents the weight (i.e. the number of ones) of the original information word whereas in the parallel decoding scheme the check directly indicates the number of information bits complemented.

Al-Bassam and Bose in [ALB90] improved the parallel decoding scheme by presenting a construction with

$$k = 2^r - (r \bmod 2)$$

information bits. They showed this construction is optimal when Knuth's complementation method mentioned above is used.

The serial balanced coding scheme has been extended to

$$k = 2^{r+1} - r - 2$$

information bits by Bose in [BOS91] and further extended to

$$k = 2^{r+1} - \left\lceil 0.8\sqrt{r} \right\rceil - 2$$

information bits by Al-Bassam and Bose in [ALB94]. The latter result was also shown to be optimal when Knuth's complementation method is used.

This thesis contains new balanced code construction methods which give an improvement of 1 or 2 bits in the amount of redundancy in most practical cases. In particular, Chapter 3 contains three different constructions for balanced code with r check bits and

$$k \leq 2^{r+1} - 2,$$

$$k \leq 3 \cdot 2^r - 8,$$

$$k \leq 5 \cdot 2^r - 10r + C(r)$$

($C(r) \in \{-15, -10, -5, 0, +5\}$) information bits. Further we find a closed form for the function $k_{max}(r)$: the maximum integer such that there exists a balanced code with r check bits and k information bits. We show that

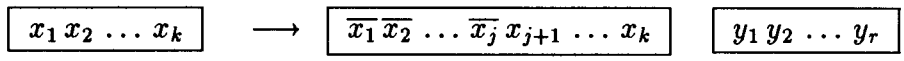
$$k_{max}(r) = 2\left\lfloor \frac{2^{2r}}{\pi} \right\rfloor - r - (1 \pm 1),$$

2.2 Knuth's method

Since our balanced code construction schemes partially relies upon Knuth's complementation method we will describe it here.

In 1986, Knuth proposed a very simple and efficient construction of balanced codes [KNU86]. These codes have high information rate as well as simple and fast encoding and decoding algorithms. In his construction, the encoding and decoding

require only a complementation of some appropriate number of bits in the information word, starting from the beginning. If $X \in \mathbb{Z}_2^k$, $X = x_1 x_2 \ldots x_k$, is a binary information word, the balanced codeword encoding of X, consists of X with the first j bits complemented (j may be 0) and a check symbol of length r, as depicted in the following diagram



information word balanced codeword

Serial and parallel decoding schemes were given in [KNU86]. In serial decoding, the check symbol specifies the original weight of the information word, so the decoder complements one bit at a time until it recovers the original information word. On the other hand, in parallel decoding the check symbol specifies the number of complemented bits, so the decoder simultaneously complements that many bits to recover the original information word.

Since we are mainly interested with the serial decoding scheme, we will describe it in the present and following sections. In order to do this, some notation is needed. Given $X = x_1 x_2 \ldots x_k \in \mathbb{Z}_2^k$, let $X^{(j)}$ be X with the first j bits complemented, i.e.

$$X^{(j)} \stackrel{\text{def}}{=} \overline{x_1}\,\overline{x_2} \ldots \overline{x_j}\, x_{j+1} \ldots x_k.$$

Further, let

$$\sigma_j(X) \stackrel{\text{def}}{=} w(X^{(j)}) = w(\overline{x_1}\,\overline{x_2} \ldots \overline{x_j}\, x_{j+1} \ldots x_k).$$

For example, if $X = 1001\,0000$ then $X^{(5)} = 0110\,1000$ and $\sigma_5(X) = 3$.

As a function of j, $\sigma_j(X)$ satisfies the following properties:

$$\sigma_0(X) = w(X),$$

$$\sigma_k(X) = k - w(X), \qquad k = l(X)$$

$$\sigma_j(X) = \sigma_{j-1}(X) \pm 1 \qquad \text{for } j = 1, 2, \ldots, k,$$

$$\sigma_j(X) \in [\sigma_i(X) - |j-i|, \sigma_i(X) + |j-i|] \qquad \text{for any } i, j \in [0, k].$$

Since as a function of j, $\sigma_j(X)$ goes from $\sigma_0(X) = w(X)$ to $\sigma_k(X) = k - w(X)$, by unit steps, it is possible to obtain a word of any weight between $w(X)$ and $k - w(X)$ by complementing the first j bits of X, for some $j \in [0, k]$. Of course, there may be several j's which give the same weight. In other words, given X and

$$a \in [\min\{w(X), k - w(X)\}, \max\{w(X), k - w(X)\}]$$

there exists a j such that $\sigma_j(X) = a$. In particular, there always exists a j such that $\sigma_j(X) = \left\lceil \frac{k}{2} \right\rceil$ (or $\left\lfloor \frac{k}{2} \right\rfloor$). In this sense, $\sigma_j(X)$ represents a "random walk" from $w(X)$ to $k-w(X)$. For example, if $X = 0101\,0000$ then $w(X) = 2$ and $k-w(X) = 8-2 = 6$ and so we can obtain words of weight 2, 3, 4, 5 and 6 by complementing the first 0, 1, 6, 7 and 8 bits respectively.

2.3 Map definitions and Code construction

Knuth's method with serial decoding scheme relies essentially upon the properties of $\sigma_j(X)$ mentioned in the previous section. A check symbol is an element of $\mathbb{Z}_2^r$. The general strategy of the encoding scheme is to partition the set of information words into 2^r sets, one for each check symbol, say $\{\mathcal{S}^Y\}_{Y \in \mathbb{Z}_2^r}$, and then consider 2^r one-to-one functions

$$<Y>: \mathcal{S}^Y \longrightarrow \mathcal{S}_v^k, \tag{2.1}$$

which map the words of $\mathcal{S}^Y$ to the words of weight

$$v = \left\lceil \frac{k+r}{2} \right\rceil - w(Y). \tag{2.2}$$

In order to encode the generic information word X, the encoder performs the following steps:

1) compute which set of the partition X belongs to, say $X \in \mathcal{S}^Y$,

2) compute $C = <Y> (X)$,

3) append the check symbol Y to C, obtaining as encoding of X the word

$$CY = <Y> (X)Y.$$

Since v is chosen so that relation (2.2) is satisfied, we have

$$w(CY) = w(C) + w(Y) = v + w(Y) = \left\lceil \frac{k+r}{2} \right\rceil - w(Y) + w(Y) = \left\lceil \frac{k+r}{2} \right\rceil,$$

i.e. the codeword associated with X is balanced. Decoding is straightforward. If CY is received, the decoder, reading the check symbol Y knows that the information word lies in the set $\mathcal{S}^Y$ and has been encoded by means of the function $<Y>$. So it computes $X = <Y>^{-1} (C)$.

Now, how do we define the maps (2.1)? The idea is to use Knuth's complementation method; i.e. we can consider the functions

$$f : \mathcal{S} \longrightarrow \mathcal{S}_v^k, \qquad S \subseteq \mathbb{Z}_2^k \tag{2.3}$$

which associates every information word $X \in \mathcal{S}$ with $X^{(j)}$; j being the smallest integer such that

$$w(X^{(j)}) = \sigma_j(X) = v.$$

However, in order for the coding scheme to work, we need the functions (2.3) to be well defined (i.e. total) and one-to-one. Note that such functions can be easily computed by complementing one bit of X at a time, starting from the first, until the weight becomes v.

The following theorems state necessary and sufficient conditions for correct coding.

Theorem 1 ([KNU86]) *Let $\sigma : \mathcal{S}_a^k \longrightarrow \mathcal{S}_v^k$ be the function which maps every word $X \in \mathcal{S}_a^k$ to $X^{(j)}$; j being the smallest integer such that*

$$w(X^{(j)}) = \sigma_j(X) = v.$$

Then, σ is well defined and injective iff

$$\min\{a, k-a\} \leq v \leq \max\{a, k-a\} \qquad (\iff |\mathcal{S}_a^k| \leq |\mathcal{S}_v^k|).$$

Theorem 2 ([ALB94]) *Let $\delta : \mathcal{S}_a^k \cup \mathcal{S}_b^k \longrightarrow \mathcal{S}_v^k$ (where $b > a$) be the function which maps every word $X \in \mathcal{S}_a^k \cup \mathcal{S}_b^k$ to $X^{(j)}$; j being the smallest integer such that*

$$w(X^{(j)}) = \sigma_j(X) = v.$$

Then, δ is well defined and injective iff

$$b - a > \max\{v, k-v\}.$$

The functions σ in Theorem 1 are called single maps, whereas the functions δ in Theorem 2 are called double maps.

For example Theorem 1 implies that the single map $\sigma : \mathcal{S}_5^{12} \longrightarrow \mathcal{S}_8^{12}$ is not well defined and injective, but the single map $\sigma : \mathcal{S}_5^{12} \longrightarrow \mathcal{S}_6^{12}$ is well defined and injective. Instead Theorem 2 implies that the double map $\delta : \mathcal{S}_3^{12} \cup \mathcal{S}_8^{12} \longrightarrow \mathcal{S}_7^{12}$ is not well defined and injective, but the double map $\delta : \mathcal{S}_2^{12} \cup \mathcal{S}_{10}^{12} \longrightarrow \mathcal{S}_6^{12}$ is well defined and injective.

From the above discussions it follows that to design a balanced code using Knuth's method, one needs to solve a combinatorial problem, i.e. find the best combination of check symbols $Y \in \mathbb{Z}_2^r$ and (well defined and injective) single or double maps such that each Y "encodes" a particular map. That is, each Y picks out either a single map or a double map and the appropriate domain and codomain for the map. The relation

$$v + w(Y) = \left\lceil \frac{k+r}{2} \right\rceil,$$

where v is the weight of elements in the codomain, must hold for all maps and the family of domains must define a partition of $\mathbb{Z}_2^k$.

In the literature, single and double maps are represented by the following notation

$$Y :: \mathcal{S}_a^k \longrightarrow \mathcal{S}_v^k$$

and

$$Y :: \mathcal{S}_a^k \cup \mathcal{S}_b^k \longrightarrow \mathcal{S}_v^k$$

respectively. Y is nothing but the check symbol which "encodes" them.

For example a balanced code with $r = 3$ check bits and $k = 12$ information bits (so $n = k + r = 15$) can be designed as follows

$111 :: \mathcal{S}_5^{12} \longrightarrow \mathcal{S}_5^{12}$ $\quad$ $000 :: \mathcal{S}_0^{12} \cup \mathcal{S}_9^{12} \longrightarrow \mathcal{S}_8^{12}$ $\quad$ $010 :: \mathcal{S}_3^{12} \cup \mathcal{S}_{11}^{12} \longrightarrow \mathcal{S}_7^{12}$

$110 :: \mathcal{S}_6^{12} \longrightarrow \mathcal{S}_6^{12}$ $\quad$ $101 :: \mathcal{S}_1^{12} \cup \mathcal{S}_8^{12} \longrightarrow \mathcal{S}_6^{12}$ $\quad$ $001 :: \mathcal{S}_4^{12} \cup \mathcal{S}_{12}^{12} \longrightarrow \mathcal{S}_7^{12}$

$100 :: \mathcal{S}_7^{12} \longrightarrow \mathcal{S}_7^{12}$ $\quad$ $011 :: \mathcal{S}_2^{12} \cup \mathcal{S}_{10}^{12} \longrightarrow \mathcal{S}_6^{12}$

Note that all the above maps satisfy the conditions of Theorem 1 and Theorem 2, so they are all well defined and injective. Now, assume the information word $X = 0101\,0000\,0000 \in \mathbb{Z}_2^k$ needs to be encoded. The encoder performs the following steps

1) compute $w(X) = 2$. Since $X \in \mathcal{S}_2^{12} \cup \mathcal{S}_{10}^{12}$, then

2) complement X one bit at a time until weight 6 is obtained. The computed word C is

$$C = 1010\,1111\,0000.$$

3) Append the check symbol $Y = 011$ to C, obtaining as encoding of X the word

$$CY = 1010\,1111\,0000\,011 \in \mathcal{S}_8^{15},$$

which is balanced.

On receiving $CY = 1010\,1111\,0000\,011$, the decoder, reading the check symbol $Y = 011$, knows that the original information word was encoded using the double map labeled with 011. It looks up and finds that the unencoded word came from $\mathcal{S}_2^{12}$ or $\mathcal{S}_{10}^{12}$. So it complements C one bit at a time until weight 2 or 10 is reached. Since $C = 1010\,1111\,0000$, weight 2 is reached first (complementing the first 8 bits), so it decodes C as $X = 0101\,0000\,0000$.

In [ALB94], Al-Bassam and Bose gave a method to obtain a best combination of check symbols and single or double maps, proving that with r check bits, using the complementation method, one cannot encode more than

$$k = 2^{r+1} - \left\lceil 0.8\sqrt{r} \right\rceil - 2$$

information bits.

Chapter 3
The Proposed Schemes

3.1 Overview

This chapter contains new balanced code construction methods which improve the constructions found in the literature. Our constructions save 1 or 2 bits. Further, in spite of an higher information rate, the codes presented here have the same encoding and decoding complexities as those known so far. In particular, we present three different constructions for balanced code with r check bits and k information bits, where

$$k \leq 2^{r+1} - 2,$$

$$k \leq 3 \cdot 2^r - 8,$$

$$k \leq 5 \cdot 2^r - 10r + c(r), \qquad c(r) \in \{-15, -10, -5, 0, +5\}.$$

The three constructions differ in redundancy as well as in circuit complexity and it is up to the system designer to choose which code is suitable for a specific application. In some cases the first two constructions have a parallel encoding scheme.

As in Section 2.3, the general encoding/decoding scheme of our methods is to partition the set of information words into 2^r sets, one for each check symbol, say $\{\mathcal{S}^Y\}_{Y \in \mathbb{Z}_2^r}$, and then consider 2^r one-to-one functions

$$<Y>: \mathcal{S}^Y \longrightarrow \mathcal{S}_v^k, \tag{3.1}$$

which map the words of $\mathcal{S}^Y$ to the word of weight

$$v = \left\lceil \frac{k+r}{2} \right\rceil - w(Y). \tag{3.2}$$

In order to encode the generic information word X, the encoder performs the following steps:

1) compute which set of the partition X belongs to, say $X \in \mathcal{S}^Y$,

2) compute $C = <Y> (X)$,

3) append the check symbol Y to C, obtaining as encoding of X the word

$$CY = <Y> (X)Y.$$

Since v is chosen so that relation (3.2) is satisfied, we have

$$w(CY) = w(C) + w(Y) = v + w(Y) = \left\lceil \frac{k+r}{2} \right\rceil - w(Y) + w(Y) = \left\lceil \frac{k+r}{2} \right\rceil,$$

i.e. the codeword associated with X is balanced. Decoding is straightforward. If CY is received, the decoder, reading the check symbol Y knows that the information word lies in the set $\mathcal{S}^Y$ and has been encoded by means of the function $<Y>$. So it computes $X = <Y>^{-1} (C)$.

According to the scheme, a balanced code is completely specified by defining for each of the 2^r check symbols $Y \in \mathbb{Z}_2^r$, a one-to-one function (3.1) for which (3.2) holds and the family $\{\mathcal{S}^Y\}_{Y \in \mathbb{Z}_2^r}$ is a partition of $\mathbb{Z}_2^k$; that is,

$$\mathbb{Z}_2^k = \bigcup_{Y \in \mathbb{Z}_2^r} \mathcal{S}^Y \qquad \text{and} \qquad \mathcal{S}^{Y_1} \cap \mathcal{S}^{Y_2} \neq \emptyset \iff Y_1 = Y_2.$$

In Section 2.3 the functions $<Y>$'s were single maps σ and double maps δ, here they are single maps and **tail-maps**, which we will name with τ.

All three constructions are based on the concept of tail-map. A tail-map is a one-to-one function from a set of words which are far from balanced to the set of the balanced words (or constant weight words). Now, those words which are far from balanced are encoded, using a small number p, independent on k, (say $p \leq 4$) of check symbols, by means of tail-maps. Whereas, the nearly balanced words are encoded, using the remaining $2^r - p$ check symbols, by means of single maps defined by Knuth's method. Given $t \in \mathbb{N}$, let

$$\mathcal{U}_t \stackrel{\text{def}}{=} \{X \in \mathbb{Z}_2^k : \ 0 \leq w(X) \leq t \text{ or } k - t \leq w(X) \leq k\}$$

and

$$\mathcal{B}_t \stackrel{\text{def}}{=} \mathbb{Z}_2^k - \mathcal{U}_t = \{X \in \mathbb{Z}_2^k : \ t < w(X) < k - t\}.$$

$\mathcal{U}_t$ is the set of far from balanced information words. $\mathcal{B}_t$ is the set of nearly balanced information words. In every construction presented in this thesis the set $\mathcal{B}_t$ is always partitioned as

$$\mathcal{B}_t = \mathcal{S}_{t+1}^k \cup \mathcal{S}_{t+2}^k \cup \ldots \cup \mathcal{S}_{k-t-1}^k$$

and each word $X \in \mathcal{S}_a^k \subseteq \mathcal{B}_t$ encoded by means of a single map $\sigma_a : \mathcal{S}_a^k \longrightarrow \mathcal{S}_v^k$. On the other hand, depending on k and the construction, the set $\mathcal{U}_t$ is partitioned in different ways, but it is never partitioned into more than 4 sets, say

$$\mathcal{U}_t = T_1 \cup T_2 \cup \ldots \cup T_p \qquad p \leq 4.$$

Analogously as before each word $X \in T_i \subseteq \mathcal{U}_t$ is encoded by means of a tail-map $\tau_i : T_i \longrightarrow \mathcal{S}_v^k$. Note that in order to minimize r (i.e. obtain less redundant codes) we need to maximize t.

In this chapter we give three different methods to design tail-maps. For each of these three methods we get the three announced balanced code constructions.

Now, let's focus on the tail map constructions. As we mentioned above a tail-map is a one-to-one function $\tau : T \longrightarrow \mathcal{S}_v^k$, T being a subset of the far from balanced information words U_t. In our case $v = \left\lfloor \frac{k}{2} \right\rfloor$ or $\left\lceil \frac{k}{2} \right\rceil$. The general idea of all the tail-map constructions presented here is the following: by means of a map $U : T \longrightarrow \mathbb{Z}_2^+$, we compress the information word $X \in T$ we need to balance, and then we use the saved space $k - l(U(X))$ to balance $U(X)$, the compressed version of X.

In the construction presented in Section 3.2, we identify the information word $X = x_1x_2x_3x_4 \ldots x_{k-1}x_k$ with the $\frac{k}{2}$ long sequence of the two bits words $b_1^X = x_1x_2$, $b_2^X = x_3x_4$, ..., $b_{\frac{k}{2}}^X = x_{k-1}x_k$ (assume k even). Considering each b_i^X as the binary encoding of a natural number, X can be identified with a sequence B^X of $\frac{k}{2}$ natural numbers between 0 and 3. We use, as compressing function U, the unary encoding of the sequence B^X. Note that, since every codeword of the unary encoding has exactly one 1 (we assume that the unary encoding of a natural number m is formed by m 0's followed by a 1), $U(X)$ has exactly $\frac{k}{2}$ 1's. Obviously we need the length of $U(X)$ to be less than or equal to $k = l(X)$. In this case, appending enough 0's to $U(X)$ we can get a balanced word (i.e. a word with $\frac{k}{2}$ 1's and length k). If the weight of X is less than or equal to $\left\lfloor \frac{k}{4} \right\rfloor$ then $l(U(X)) \leq k$ and so we are able to design a tail-map

$$\tau : \mathcal{S}_0^k \cup \mathcal{S}_1^k \cup \ldots \cup \mathcal{S}_{\left\lfloor \frac{k}{4} \right\rfloor}^k \longrightarrow \mathcal{S}_{\frac{k}{2}}^k.$$

In Section 3.3, we still use the unary encoding as above, but by counting the number of occurrences of 01's and 10's in X we are able to design a more powerful compressing function $U(X)$. In this case we are able to prove that if the weight of X is less than or equal to $\left\lfloor \frac{k}{3} \right\rfloor$ then $l(U(X)) \leq k$ and so we are able to design a tail-map

$$\tau : \mathcal{S}_0^k \cup \mathcal{S}_1^k \cup \ldots \cup \mathcal{S}_{\left\lfloor \frac{k}{3} \right\rfloor}^k \longrightarrow \mathcal{S}_{\frac{k}{2}}^k.$$

When $r \geq 3$ this construction is better, in terms of redundancy, than the previous one.

In Section 3.4, we compress the information word X using a variable length uniquely decodable binary code. In this case, however, $U(X)$ may not be balanced, but we use the saved space to make $U(X)$ balanced with Knuth's complementation method. In this way we are able to design a tail-map

$$\tau : \mathcal{S}_0^k \cup \mathcal{S}_1^k \cup \ldots \cup \mathcal{S}_t^k \longrightarrow \mathcal{S}_{\frac{k}{2}}^k, \qquad t \simeq \frac{2k}{5}.$$

When $r \geq 5$ this construction is better, in terms of redundancy, than the previous two.

Let $k_{max}(r)$ be the maximum integer such that a balanced code with $k(r)$ and r information bits exists. In Section 3.5 we give a closed form for the function $k_{max}(r)$, proving that

$$k_{max}(r) = 2\left\lfloor \frac{2^{2r}}{\pi} \right\rfloor - r - (1 \pm 1).$$

Further, we show some comparisons among our methods and the method proposed in [ALB94].

3.2 Construction I

In this section we will present a method to design balanced codes with r check bits and

$$k \leq 2^{r+1} - 2$$

information bits.

As we mentioned in Section 3.1, a balanced code construction method is completely specified by defining a partition of the set of information words, specifying how to map the words in each set of the partition and assigning a check symbol to every map defined in each such set. Depending upon the values of k, the balanced codes presented in this section are specified as follows. Assume $k \in 4\mathbb{N} + 2$ first. If $\{Y_i\}_{i=1,2,\ldots,2^r}$ is the set of check symbols $\mathbb{Z}_2^r$, then

$$<Y_1>: \bigcup_{i=0}^{t} \mathcal{S}_i^k \cup \bigcup_{i=k-t}^{k} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\frac{k}{2}}^k = \mathcal{S}_{v_1}^k,$$

$$<Y_2>: \mathcal{S}_{t+1}^k \longrightarrow \mathcal{S}_{v_2}^k,$$

$$\vdots$$

$$<Y_{2^r}>: \mathcal{S}^k_{k-t-1} \longrightarrow \mathcal{S}^k_{v_{2^r}},$$

where

$$t = \left\lfloor \frac{k}{4} \right\rfloor.$$

The function $< Y_1 >$ is a tail-map τ and it will be described later, whereas the functions $<Y_i>$, $i = 2, \ldots, 2^r$ are single maps σ (see Section 2.3). Note that if X is an information word and $X \in \mathcal{S}^k_w \subseteq \mathcal{B}_t = \bigcup_{i=t+1}^{k-t-1} \mathcal{S}^k_i$, then the encoding of X is $<Y_{w-t+1}> (X) Y_{w-t+1}$, whereas if $X \in \mathcal{U}_t = \bigcup_{i=0}^{t} (\mathcal{S}^k_i \cup \mathcal{S}^k_{k-i})$, the encoding of X is $<Y_1> (X) Y_1$. Further, in order for the code to be balanced, the relation

$$v_i + w(Y_i) = \left\lceil \frac{k+r}{2} \right\rceil$$

must hold for all $i = 1, 2, \ldots, 2^r$. In particular, since $v_1 = \frac{k}{2}$, then $w(Y_1) = \left\lceil \frac{r}{2} \right\rceil$.

If $k \notin 4\mathbb{N} + 2$, the construction is exactly the same but, instead of using only one check symbol to encode the words of $\mathcal{U}_t$, we need to use two check symbols associated with the two different tail-maps

$$<Y_1>: \bigcup_{i=0}^{t} \mathcal{S}^k_i \longrightarrow \mathcal{S}^k_{\lceil \frac{k}{2} \rceil} = \mathcal{S}^k_{v_1},$$

$$<Y_2>: \bigcup_{i=k-t}^{k} \mathcal{S}^k_i \longrightarrow \mathcal{S}^k_{\lceil \frac{k}{2} \rceil} = \mathcal{S}^k_{v_2}.$$

Now that we outlined the code design, we are ready to look at the tail-map constructions. The idea is very simple. Let $k \in \mathbb{N}$ and

$$X = x_1 x_2 \, x_3 x_4 \ldots x_{k-2} x_{k-1} x_k$$

be an information word. Consider the correspondence which associates X with the sequence

$$B^X = \begin{cases} x_1x_2, x_3x_4, \ldots, x_{k-1}x_k & \text{if } k \text{ is even,} \\ \\ x_1x_2, x_3x_4, \ldots, x_{k-2}x_{k-1}, x_k & \text{if } k \text{ is odd.} \end{cases}$$

Note that such a correspondence is one-to-one. Each

$$b^X_i \stackrel{\text{def}}{=} \begin{cases} x_{2i-1}x_{2i} & \text{if } i \in \left[1, \left\lfloor \frac{k}{2} \right\rfloor\right], \\ \\ x_k & \text{if } i = \left\lceil \frac{k}{2} \right\rceil \neq \left\lfloor \frac{k}{2} \right\rfloor \end{cases} \tag{3.3}$$

can be considered as the binary encoding of an integer number between 0 and 3. In this way X can be identified by the sequence B^X of $\lceil \frac{k}{2} \rceil$ integer numbers between 0 and 3. Now, we could define a tail-map

$$\tau : \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k = \mathcal{S}_{v_1}^k,$$

where $\tau(X)$ is nothing but the unary representation of the sequence B^X followed by enough 0's to make the length equal to $k = l(X)$. Since the weight of every codeword of the unary encoding is equal to one and B^X is of length $\lceil \frac{k}{2} \rceil$ then $\tau(X)$ has $\lceil \frac{k}{2} \rceil$ 1's and length k, i.e. $\tau(X)$ is balanced. Note that if $w(X)$ is small enough then the length of the unary representation of B^X is small; namely less than or equal to k.

In the following we will make the ideas mentioned above more concrete. Let

$$\begin{aligned} u(00) &\stackrel{\text{def}}{=} 1, \\ u(01) &\stackrel{\text{def}}{=} 01, \\ u(10) &\stackrel{\text{def}}{=} 001, \\ u(11) &\stackrel{\text{def}}{=} 0001, \end{aligned} \tag{3.4}$$

and

$$U(X) \stackrel{\text{def}}{=} u(b_1^X)u(b_2^X)\ldots u(b_{\lceil \frac{k}{2} \rceil}^X). \tag{3.5}$$

Note that this encoding is the unary representation of the sequence B^X. Further, when k is odd, $b_{\lceil \frac{k}{2} \rceil}^X$ is only one bit long. In this case, U is completely defined by letting

$$\begin{aligned} u(0) &\stackrel{\text{def}}{=} 1, \\ u(1) &\stackrel{\text{def}}{=} 01. \end{aligned}$$

The following lemma it is useful for the construction.

Lemma 1 *The function $U : \mathbb{Z}_2^k \longrightarrow \mathbb{Z}_2^+$ defined by (3.5) is one-to-one and if $X \in \mathbb{Z}_2^k$ then*

1. $w(U(X)) = \lceil \frac{k}{2} \rceil$,
2. $l(U(X)) \leq 2w(X) + \lceil \frac{k}{2} \rceil$,
3. $2w(X) + \lceil \frac{k}{2} \rceil \leq k \iff w(X) \leq \lfloor \frac{k}{4} \rfloor$.

Proof: The function $U : \mathbb{Z}_2^k \longrightarrow \mathbb{Z}_2^+$ is one-to-one because the code $\{0^i1\}_{i=0,1,2,3}$ is uniquely decipherable.

If $X \in \mathbb{Z}_2^k$ then

1.
$$\forall b \in \mathbb{Z}_2^2 \qquad w(u(b)) = 1 \quad \Longrightarrow \quad w(U(X)) = \left\lceil \frac{k}{2} \right\rceil .$$

2. From the (3.4) it can be noticed that
$$\forall b \in \mathbb{Z}_2^2 \qquad l(u(b)) \leq 2w(b) + 1.$$
This implies that
$$l(U(X)) = \sum_{i=1}^{\lceil \frac{k}{2} \rceil} l(u(b_i^X)) \leq \sum_{i=1}^{\lceil \frac{k}{2} \rceil} [2w(b_i^X) + 1] =$$
$$2 \sum_{i=1}^{\lceil \frac{k}{2} \rceil} w(b_i^X) + \left\lceil \frac{k}{2} \right\rceil = 2w(X) + \left\lceil \frac{k}{2} \right\rceil .$$

3.
$$2w(X) + \left\lceil \frac{k}{2} \right\rceil \leq k \iff 2w(X) \leq k - \left\lceil \frac{k}{2} \right\rceil = \left\lfloor \frac{k}{2} \right\rfloor \iff$$
$$w(X) \leq \frac{1}{2} \left\lfloor \frac{k}{2} \right\rfloor \iff w(X) \leq \left\lfloor \frac{k}{4} \right\rfloor .$$

■

Let
$$t = t(k) \stackrel{\text{def}}{=} \left\lfloor \frac{k}{4} \right\rfloor .$$

From 2 and 3 of Lemma 1, it follows that
$$X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k \quad \Longrightarrow \quad l(U(X)) \leq 2 \left\lfloor \frac{k}{4} \right\rfloor + \left\lceil \frac{k}{2} \right\rceil = 2 \frac{k - k \bmod 4}{4} + \left\lceil \frac{k}{2} \right\rceil =$$
$$\left\lceil \frac{k - k \bmod 4}{2} + \frac{k}{2} \right\rceil = \begin{cases} k-1 & \text{if } k \in (4\mathbb{N}+2) \cup (4\mathbb{N}+3), \\ k & \text{if } k \in 4\mathbb{N} \cup (4\mathbb{N}+1). \end{cases}$$

In the case when $k \in 4\mathbb{N}+2$, this last relation implies that if $w(X) \leq \left\lfloor \frac{k}{4} \right\rfloor$ then $l(U(X)) \leq k-1$, i.e. we save one bit. In this case then, Lemma 1 and the fact that k is even guarantee that we can define the tail-map
$$\tau : \left(\bigcup_{i=0}^{t} \mathcal{S}_i^k \right) \cup \left(\bigcup_{i=k-t}^{k} \mathcal{S}_i^k \right) \longrightarrow \mathcal{S}_{\frac{k}{2}}^k$$
as follows
$$\tau(X) \stackrel{\text{def}}{=} \begin{cases} U(X) 0^{(k-1)-l(U(X))} 0 & \text{if } X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k, \\ \overline{U(\overline{X})} 1^{(k-1)-l(U(\overline{X}))} 1 & \text{if } X \in \bigcup_{i=k-t}^{k} \mathcal{S}_i^k. \end{cases} \tag{3.6}$$

Defining τ as above, we use the rightmost bit of $\tau(X)$ to store some information; namely, if $w(X) \leq t$ or $w(X) \geq k - t$. In the first case the rightmost bit of $\tau(X)$ is set equal to 0 and in the second case it is set equal to 1. In other words, once the encoder has established the weight of the information word $X \in \mathcal{U}_t$, if such weight is less than or equal to $t = \left\lfloor \frac{k}{4} \right\rfloor$, it converts X with U and appends 0's until the length becomes k. In this way the rightmost bit of $\tau(X)$ is always 0. If instead, $w(X) \geq k - \left\lfloor \frac{k}{4} \right\rfloor$ the encoder first complements X, obtaining a word of weight less than or equal to $\left\lfloor \frac{k}{4} \right\rfloor$, converts $\overline{X}$ with U and append 0's until the length becomes k and finally complements everything again. Note that in this way the rightmost bit of $\tau(X)$ is always 1. Now, the decoder, reading the rightmost bit of $\tau(X)$, knows how to invert the encoding procedure described above and recover the information word X. In fact, if the rightmost bit of $\tau(X)$ is 0, the decoder extracts the first part of $\tau(X)$, from the first bit to the last 1, and applies U^{-1}. If instead, the rightmost bit is 1, it first complements $\tau(X)$, then extracts the first part and applies U^{-1} as above, and then complements everything again.

In the case when $k \notin 4\mathbb{N} + 2$, in order to encode those information words in $\mathcal{U}_t$, we must define two tail-maps

$$\tau_1 : \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k,$$

$$\tau_2 : \bigcup_{i=k-t}^{k} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k$$

where

$$\tau_1(X) \stackrel{\text{def}}{=} U(X)0^{k-l(U(X))},$$
$$\tau_2(X) \stackrel{\text{def}}{=} U(\overline{X})0^{k-l(U(\overline{X}))}.$$

Note that all the functions defined above can be computed using a parallel scheme.

Example 1 *If* $r = 2$ *and* $k = 6 \in 4\mathbb{N} + 2$ *then* $t = \left\lfloor \frac{k}{4} \right\rfloor = 1$ *and* $k - t = 5$, *therefore the following tail-map*

$$\tau : \left(\mathcal{S}_0^6 \cup \mathcal{S}_1^6\right) \cup \left(\mathcal{S}_5^6 \cup \mathcal{S}_6^6\right) \longrightarrow \mathcal{S}_3^6$$

can be defined. Assume 10 *is the check symbol which encodes* τ*; i.e.* $\tau = <10>$.

If

$$X = 00\,00\,10 \in \mathbb{Z}_2^6$$

is an information word, the encoder performs the following steps:

1) compute $w(X) = 1$. *Since* $X \in (\mathcal{S}_0^6 \cup \mathcal{S}_1^6)$ *then*

2) compute $U(X) = U(\underline{00}\ \underline{00}\ \underline{10}) = \underline{1}\ \underline{1}\ \underline{001}$,

3) append 0*'s until length is k:* $U(X)0^{5-l(U(X))}0 = 1\,1\,001\,0$,

4) append 10 *as check symbol:* $<10>\,(X)\ 10 = \tau(X)\ 10 = 1\,1\,001\,0\ 10 \in \mathcal{S}_4^8$.

To decode the received word

$$\tilde{X}\tilde{Y} = 1\,1\,001\,0\ 10,$$

reading $\tilde{Y} = 10$, *the decoder knows that the information word* X *was encoded using the function* $\tau = <10>$. *Since the rightmost bit of* $\tilde{X}$ *is* 0, *it knows that* $X \in (\mathcal{S}_0^6 \cup \mathcal{S}_1^6)$ *and* U *was used for encoding. Therefore it computes*

$$U^{-1}(1\,1\,001) = X = 00\,00\,10.$$

Suppose now that

$$X = 11\,11\,10 \in \mathbb{Z}_2^6$$

is another information word. In this case the encoder performs the following steps:

1) compute $w(X) = 5$. *Since* $X \in (\mathcal{S}_5^6 \cup \mathcal{S}_6^6)$ *then*

2) complement X: $\overline{X} = 00\,00\,01 \in \mathcal{S}_1^6$,

3) compute $U(\overline{X}) = U(\underline{00}\ \underline{00}\ \underline{01}) = \underline{1}\ \underline{1}\ \underline{01}$,

4) append 0*'s until length is k:* $U(\overline{X})0^{5-l(U(\overline{X}))}0 = 1\,1\,01\,00$,

5) complement everything: $\overline{U(\overline{X})0^{5-l(U(\overline{X}))}0} = 0\,0\,10\,11$,

6) append 10 *as check symbol:* $<10>\,(X)\ 10 = \tau(X)\ 10 = 0\,0\,10\,11\ 10 \in \mathcal{S}_4^8$.

To decode the received word

$$\tilde{X}\tilde{Y} = 0\,0\,10\,11\ 10,$$

reading $\tilde{Y} = 10$, *the decoder knows that the information word* X *was encoded using the function* $\tau = <10>$. *Since the rightmost bit of* $\tilde{X}$ *is* 1, *it knows that* $X \in (\mathcal{S}_5^6 \cup \mathcal{S}_6^6)$, X *was complemented,* U *was used and then* $U(\overline{X})$ *was complemented during encoding. Therefore it computes*

1) $\overline{\tilde{X}} = 1\,1\,01\,00$,

2) $U^{-1}(1\,1\,01) = 00\,00\,01 = \overline{X}$,

3) $X = \overline{\overline{X}} = 11\,11\,10.$

To see how the tail-maps just defined can be used to design efficient balanced codes, let $k, r \in \mathbb{N}$ with $k \geq 6$ and

$$d(k) \stackrel{\text{def}}{=} |\{\mathcal{S}_i^k : i = t(k)+1, \ldots, k - t(k) - 1\}| = k - 2t(k) - 1. \qquad (3.7)$$

$d(k)$ is the number of different single maps needed to encode those information words that are close to balanced. From Lemma 1 it follows

$$d(k) = k - 2t(k) - 1 = k - 2\left\lfloor \frac{k}{4} \right\rfloor - 1 = k - 2\,\frac{k - k \bmod 4}{4} - 1 =$$

$$k - \frac{k - k \bmod 4}{2} - 1 = \frac{k + k \bmod 4 - 2}{2}.$$

Since $k \geq 6$, a balanced code can be constructed iff the following relations hold

$$d(k) = \frac{k + k \bmod 4 - 2}{2} \leq \begin{cases} 2^r - 1 & \text{if } k \in 4\mathbb{N} + 2, \\ 2^r - 2 & \text{otherwise.} \end{cases} \iff$$

$$k \leq \begin{cases} 2^{r+1} - 2 & \text{if } k \in 4\mathbb{N}, \\ 2^{r+1} - 3 & \text{if } k \in 4\mathbb{N} + 1, \\ 2^{r+1} - 2 & \text{if } k \in 4\mathbb{N} + 2, \\ 2^{r+1} - 5 & \text{if } k \in 4\mathbb{N} + 3. \end{cases} \iff$$

$$k \leq \begin{cases} 2^{r+1} - 4 & \text{if } k \in 4\mathbb{N}, \\ 2^{r+1} - 3 & \text{if } k \in 4\mathbb{N} + 1, \\ 2^{r+1} - 2 & \text{if } k \in 4\mathbb{N} + 2, \\ 2^{r+1} - 5 & \text{if } k \in 4\mathbb{N} + 3. \end{cases}$$

this implies that the greatest number of information bits that can be encoded using r check bits is $2^{r+1} - 2$.

Example 2 *If* $r = 2$ *then* $k = 2^{2+1} - 2 = 6$, $n = 8$ *and* $t(6) = \left\lfloor \frac{6}{4} \right\rfloor = 1$, *therefore a* $DC(8,6)$ *code can be constructed as follows*

$$<10>: (\mathcal{S}_0^6 \cup \mathcal{S}_1^6) \cup (\mathcal{S}_5^6 \cup \mathcal{S}_6^6) \longrightarrow \mathcal{S}_3^6,$$

$$<11>: \mathcal{S}_2^6 \longrightarrow \mathcal{S}_2^6,$$

$$<01>: \mathcal{S}_3^6 \longrightarrow \mathcal{S}_3^6,$$

$$<00>: \mathcal{S}_4^6 \longrightarrow \mathcal{S}_4^6,$$

where $<10>$ *is the tail-map defined in (3.6) and* $<11>$, $<01>$ *and* $<00>$ *are single maps* σ *which in this case are identity functions.*

3.3 Construction II

In this section we will design balanced codes with $r \geq 3$ check bits and

$$k \leq 3 \cdot 2^r - 8$$

information bits.

By modifying only a little bit the method presented in Section 3.2, we are able to encode all the words whose weight is roughly less than $\frac{k}{3}$ (instead of $\frac{k}{4}$) or greater than $k - \frac{k}{3}$, using at most 4 check symbols. By using the unary encoding U (see (3.5)), it can be noticed that the maximum of $l(U(X))$ in $\mathcal{S}_w^k$ is equal to $2w + \left\lceil \frac{k}{2} \right\rceil$ and is reached for $X = (10)^w 0^{k-2w}$; i.e.

$$\max_{X \in \mathcal{S}_w^k} l(U(X)) = l(U(X))|_{X=(10)^w 0^{k-2w}} = 2w + \left\lceil \frac{k}{2} \right\rceil .$$

This means that, while encoding the substring b_i^X in unary, the worst case occurs when $b_i^X = 10$. Here we reduce the tight upper bound of $2w + \left\lceil \frac{k}{2} \right\rceil$ by encoding 10 as 001 and 01 as 01 if the number of occurrences of 10 in X is smaller than the number of occurrences of 01. In the other case 10 is encoded as 01 and 01 as 001. In other words, we encode the far from balanced information words by using two different encodings: the unary encodings as in Section 3.2 and the encodings which is exactly the same as the unary encoding but encodes 10 as 01 and 01 as 001. Further we use one or the other depending on which one gives a shorter encoding. This is the main strategy used in this section.

Given $k \in \mathbb{N}$ and $X \in \mathbb{Z}_2^k$, let b_i^X be as in (3.3). For $i = 1, 2$, let

$$\begin{aligned} u_i(00) &\stackrel{\text{def}}{=} 1, \\ u_i(11) &\stackrel{\text{def}}{=} 0001, \\ u_1(01) &\stackrel{\text{def}}{=} 01, \\ u_2(01) &\stackrel{\text{def}}{=} 001, \\ u_1(10) &\stackrel{\text{def}}{=} 001, \\ u_2(10) &\stackrel{\text{def}}{=} 01 \end{aligned}$$

and

$$U_i(X) \stackrel{\text{def}}{=} u_i(b_1^X) u_i(b_2^X) \ldots u_i(b_{\lceil \frac{k}{2} \rceil}^X). \tag{3.8}$$

U_1 is the unary encoding, whereas U_2 is the encoding which encodes 10 as 01 and 01 as 001. As in the previous section, when $b_{\lceil \frac{k}{2} \rceil}^X$ is only one bit long, U_i can be completely defined by letting

$$\begin{aligned} u_i(0) &\stackrel{\text{def}}{=} 1, \\ u_i(1) &\stackrel{\text{def}}{=} 01. \end{aligned}$$

For all $b \in \mathbb{Z}_2^2$ let

$$c_b(X) \stackrel{\text{def}}{=} \left| \left\{ i \in \left[1, \left\lceil \frac{k}{2} \right\rceil \right] : b_i^X = b \right\} \right|.$$

$c_b(X)$ is nothing but the number of occurrences of b in the sequence

$$B^X = b_1^X, b_2^X, \ldots, b_{\lceil \frac{k}{2} \rceil}^X$$

associated with X. For example $c_{01}(00\,01\,10\,01) = 2$.

Depending upon the values of k, the balanced codes presented in this section are specified as follows. Assume $k \in 6\mathbb{N} + 4$ first. If $\{Y_i\}_{i=1,2,\ldots,2^r}$ is the set of check symbols $\mathbb{Z}_2^r$, then

$$\begin{aligned}
<Y_1>&: \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\frac{k}{2}}^k = \mathcal{S}_{v_1}^k, \\
<Y_2>&: \bigcup_{i=k-t}^{k} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\frac{k}{2}}^k = \mathcal{S}_{v_2}^k, \\
<Y_3>&: \mathcal{S}_{t+1}^k \longrightarrow \mathcal{S}_{v_3}^k, \\
&\vdots \\
<Y_{2^r}>&: \mathcal{S}_{k-t-1}^k \longrightarrow \mathcal{S}_{v_{2^r}}^k,
\end{aligned} \tag{3.9}$$

where

$$t = \left\lfloor \frac{k}{3} \right\rfloor.$$

The functions $< Y_1 >$ and $< Y_2 >$ are tail-maps and they will be described later, whereas the functions $<Y_i>$, $i = 3, \ldots, 2^r$ are the single maps of Section 2.3. Recall that, in order for the code to be balanced, the relation

$$v_i + w(Y_i) = \left\lceil \frac{k+r}{2} \right\rceil$$

must hold for all $i = 1, 2, \ldots, 2^r$. In particular, since $v_1 = v_2 = \frac{k}{2}$, then $w(Y_1) = w(Y_2) = \left\lceil \frac{r}{2} \right\rceil$.

If $k \notin 6\mathbb{N} + 4$, the construction is exactly the same but, instead of using only two check symbol to encode the words of $\mathcal{U}_t$, we need to use four check symbols, one for each of the four different tail-maps

$$\begin{aligned}
<Y_1>&: \left\{ \bigcup_{i=0}^{t} \mathcal{S}_i^k : c_{01}(X) \geq c_{10}(X) \right\} \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k = \mathcal{S}_{v_1}^k, \\
<Y_2>&: \left\{ \bigcup_{i=0}^{t} \mathcal{S}_i^k : c_{01}(X) < c_{10}(X) \right\} \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k = \mathcal{S}_{v_2}^k,
\end{aligned} \tag{3.10}$$

$$<Y_3>: \left\{ \bigcup_{i=k-t}^{k} \mathcal{S}_i^k : c_{01}(\overline{X}) \geq c_{10}(\overline{X}) \right\} \longrightarrow \mathcal{S}^k_{\lfloor \frac{k}{2} \rfloor} = \mathcal{S}^k_{v_3},$$

$$<Y_4>: \left\{ \bigcup_{i=k-t}^{k} \mathcal{S}_i^k : c_{01}(\overline{X}) < c_{10}(\overline{X}) \right\} \longrightarrow \mathcal{S}^k_{\lfloor \frac{k}{2} \rfloor} = \mathcal{S}^k_{v_4}.$$

In this case

$$t = \begin{cases} \lceil \frac{k}{3} \rceil & \text{if } k \in 2\mathbb{N} \cap (3\mathbb{N}+2) = 6\mathbb{N}+2, \\ \lfloor \frac{k}{3} \rfloor & \text{otherwise.} \end{cases}$$

Now that we outlined the code design, we are ready to look in detail at the tail-map constructions. The constructions are based on the functions defined in the following lemma

Lemma 2 *The functions*

$$U_1 : \left\{ X \in \mathbb{Z}_2^k : c_{01}(X) \geq c_{10}(X) \right\} \longrightarrow \mathbb{Z}_2^+,$$

and

$$U_2 : \left\{ X \in \mathbb{Z}_2^k : c_{01}(X) < c_{10}(X) \right\} \longrightarrow \mathbb{Z}_2^+,$$

defined by (3.8) are one-to-one and if $X \in \mathbb{Z}_2^k$ *then*

1. *for* $i = 1, 2$

$$w(U_i(X)) = \left\lceil \frac{k}{2} \right\rceil,$$

2.

$$\min\{l(U_1(X)), l(U_2(X))\} = \begin{cases} l(U_1(X)) & \text{if } c_{01}(X) \geq c_{10}(X), \\ l(U_2(X)) & \text{if } c_{01}(X) < c_{10}(X) \end{cases}$$
$$\leq \left\lceil \frac{k}{2} \right\rceil + \left\lfloor \frac{3w(X)}{2} \right\rfloor,$$

3. $\lceil \frac{k}{2} \rceil + \lfloor \frac{3w(X)}{2} \rfloor \leq k \iff$

$$w(X) \leq \begin{cases} \lceil \frac{k}{3} \rceil & \text{if } k \in 2\mathbb{N} \cap (3\mathbb{N}+2) = 6\mathbb{N}+2, \\ \lfloor \frac{k}{3} \rfloor & \text{otherwise.} \end{cases}$$

Proof: The functions U_i are one-to-one because the code $\{0^i 1\}_{i=0,1,2,3}$ is uniquely decipherable.

If $X \in \mathbb{Z}_2^k$ then

1.

$$\text{for } i=1,2 \qquad \forall b\in\mathbb{Z}_2^2 \qquad w(u_i(b))=1 \quad\Longrightarrow\quad w(U_i(X))=\left\lceil\frac{k}{2}\right\rceil.$$

2. If $k\in 2\mathbb{N}$

$$X\in\mathcal{S}_w^k \quad\Longrightarrow\quad \min\{l(U_1(X)),l(U_2(X))\}=$$

$$\min_{i=1,2}\sum_{j=1}^{\frac{k}{2}} l(u_i(b_j^X))=\min_{i=1,2}\sum_{b\in\mathbb{Z}_2^2} c_b(X)\cdot l(u_i(b))=$$

$$c_{00}(X)+2\max\{c_{01}(X),c_{10}(X)\}+3\min\{c_{01}(X),c_{10}(X)\}+4c_{11}(X)=$$

$$\sum_{b\in\mathbb{Z}_2^2} c_b(X)+\sum_{b\in\mathbb{Z}_2^2} w(b)\cdot c_b(X)+\min\{c_{01}(X),c_{10}(X)\}+c_{11}(X)=$$

$$\frac{k}{2}+w(X)+(\min\{c_{01}(X),c_{10}(X)\}+c_{11}(X))\le$$

$$\max_{X\in\mathcal{S}_w^k}\left[\frac{k}{2}+w(X)+(\min\{c_{01}(X),c_{10}(X)\}+c_{11}(X))\right]=$$

$$\frac{k}{2}+w(X)+\max_{X\in\mathcal{S}_w^k}[\min\{c_{01}(X),c_{10}(X)\}+c_{11}(X)].$$

Now, if $X\in\mathcal{S}_w^k$ is such that the function

$$F(X)=\min\{c_{01}(X),c_{10}(X)\}+c_{11}(X)$$

assumes its maximum value in $\mathcal{S}_w^k$, then the following relation holds (See Lemma 3)

$$\min\{c_{01}(X),c_{10}(X)\}=\max\{c_{01}(X),c_{10}(X)\}-w(X)\bmod 2.$$

This implies

$$l(U(X))\le\frac{k}{2}+w(X)+$$

$$\max_{X\in\mathcal{S}_w^k:\,\min\{c_{01},c_{10}\}=\max\{c_{01},c_{10}\}-w\bmod 2}[\min\{c_{01}(X),c_{10}(X)\}+c_{11}(X)]=$$

$$\frac{k}{2}+w(X)+$$

$$\frac{\min\{c_{01}(X),c_{10}(X)\}+\max\{c_{01}(X),c_{10}(X)\}-w(X)\bmod 2}{2}+c_{11}(X)=$$

$$\frac{k}{2}+w(X)+\frac{c_{01}(X)+c_{10}(X)+2c_{11}(X)-w(X)\bmod 2}{2}=$$

$$\frac{k}{2}+w(X)+\frac{w(X)-w(X)\bmod 2}{2}=$$

$$\frac{k}{2}+\frac{3w(X)-w(X)\bmod 2}{2}=\frac{k}{2}+\left\lfloor\frac{3w(X)}{2}\right\rfloor.$$

If $k\in 2\mathbb{N}+1$, from the previous result

$$\min\{l(U_1(X)),l(U_2(X))\}=\min_{i=1,2}[l(u_i(b_1)u_i(b_2)\dots u_i(b_{\lceil\frac{k}{2}\rceil-1}))+l(u_i(x_k))]\leq$$

$$\frac{k-1}{2}+\left\lfloor\frac{3(w(X)-x_k)}{2}\right\rfloor+1+x_k=\left\lceil\frac{k}{2}\right\rceil+\left\lfloor\frac{3w(X)-3x_k}{2}\right\rfloor+x_k=$$

$$\left\lceil\frac{k}{2}\right\rceil+\left\lfloor\frac{3w(X)-x_k}{2}\right\rfloor-\not{x}_k+\not{x}_k=\left\lceil\frac{k}{2}\right\rceil+\left\lfloor\frac{3w(X)-x_k}{2}\right\rfloor\leq$$

$$\left\lceil\frac{k}{2}\right\rceil+\left\lfloor\frac{3w(X)}{2}\right\rfloor.$$

3.

$$\left\lceil\frac{k}{2}\right\rceil+\left\lfloor\frac{3w(X)}{2}\right\rfloor\leq k\iff\left\lfloor\frac{3w(X)}{2}\right\rfloor\leq\left\lfloor\frac{k}{2}\right\rfloor\iff$$

$$\frac{3w(X)-w(X)\bmod 2}{2}\leq\frac{k-k\bmod 2}{2}\iff$$

$$3w(X)-w(X)\bmod 2\leq \mathrm{k}-\mathrm{k}\bmod 2,$$

and this relation holds if and only if

$$w(X)\leq\begin{cases}\left\lceil\frac{k}{3}\right\rceil & \text{if } k\in 2\mathbb{N}\cap(3\mathbb{N}+2)=6\mathbb{N}+2,\\ \left\lfloor\frac{k}{3}\right\rfloor & \text{otherwise.}\end{cases}$$

■

Now we prove the lemma used in the proof of Lemma 2.

Lemma 3 *Given $k\in 2\mathbb{N}$ and $X\in\mathbb{Z}_2^k$, let*

$$F(X)\stackrel{\text{def}}{=}\min\{c_{01}(X),c_{10}(X)\}+c_{11}(X).$$

If $X\in\mathcal{S}_w^k$ is such that

$$F(X)=\max_{Z\in\mathcal{S}_w^k}F(Z),$$

then

$$\max\{c_{01}(X),c_{10}(X)\}=\min\{c_{01}(X),c_{10}(X)\}+w(X)\bmod 2.$$

Proof: Let $X \in \mathcal{S}_w^k$ be a string such that $F(X)$ is maximum in $\mathcal{S}_w^k$. Without loss of generality, it is possible to assume

$$X = (11)^a(10)^b(01)^c(00)^d$$

where

$$w = 2a + b + c \tag{3.11}$$

and

$$\frac{k}{2} = a + b + c + d.$$

Suppose $b \geq c$ (which implies $c = \min\{c, b\}$). If $b \geq c + 2$ then the string

$$X' = (11)^{a+1}(10)^{b-2}(01)^c(00)^{d+1}$$

is such that

$$F(X') = c + a + 1 > a + c = F(X),$$

contradicting the assumption that X is a maximum for F in $\mathcal{S}_w^k$. This means that only the following cases are left.

1. $b = c$. This implies $w \bmod 2 = 0$ (see (3.11)).

2. $b = c + 1$. This implies $w \bmod 2 = 1$ (see (3.11)).

In either case

$$b = c + w \bmod 2.$$

With a similar argument the lemma follows if $c \geq b$ (which implies $b = \min\{c, b\}$).

■

Lemma 2 is the basis for the tail-map design described below. Let

$$t = t(k) \stackrel{\text{def}}{=} \begin{cases} \left\lceil \frac{k}{3} \right\rceil & \text{if } k \in 6\mathbb{N} + 2, \\ \left\lfloor \frac{k}{3} \right\rfloor & \text{otherwise.} \end{cases}$$

From 2 and 3 of Lemma 2, it follows

$$X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k \implies l(U(X)) \leq \left\lceil \frac{k}{2} \right\rceil + \left\lfloor \frac{3w(X)}{2} \right\rfloor \leq \left\lceil \frac{k}{2} \right\rceil + \left\lfloor \frac{3t}{2} \right\rfloor =$$

$$\begin{cases} k - 1 & \text{if } k \in (6\mathbb{N} + 4) \cup (6\mathbb{N} + 5), \\ k & \text{otherwise.} \end{cases} \tag{3.12}$$

In the case when $k \in 6\mathbb{N}+4$, this last relation implies that if $w(X) \leq \left\lfloor \frac{k}{3} \right\rfloor$ then $l(U(X)) \leq k-1$, i.e. we save one bit. As in Section 3.2, we use this saved bit to encode some information; namely which function between U_1 and U_2 we use to encode X. In this case then, Lemma 1, (3.12) and the fact that k is even, guarantee that we can define the tail-map

$$\tau_1 : \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\frac{k}{2}}^k$$

as follows

$$\tau_1(X) \stackrel{\text{def}}{=} \begin{cases} U_1(X)0^{(k-1)-l(U_1(X))}0 & \text{if } c_{01}(X) \geq c_{10}(X), \\ \\ \overline{U_2(X)}1^{(k-1)-l(U_2(X))}1 & \text{if } c_{01}(X) < c_{10}(X). \end{cases} \tag{3.13}$$

Once the encoder has established that $w(X)$ is less than or equal to $t = \left\lfloor \frac{k}{3} \right\rfloor$, it computes $c_{01}(X)$ and $c_{10}(X)$. If $c_{01}(X) \geq c_{10}(X)$, it converts X with U_1 and appends 0's until the length becomes k. In this way the rightmost bit of $\tau_1(X)$ is always 0. If instead, $c_{01}(X) < c_{10}(X)$, the encoder converts X using U_2, appends 0's until the length becomes k and complements everything. In this way the rightmost bit of $\tau_1(X)$ is always 1. Now, reading the rightmost bit of $\tau(X)$, the decoder knows if X was encoded using U_1 or U_2, and hence it is able to invert $\tau_1(X)$.

Now that we know how to encode the words whose weight is less than or equal to $\left\lfloor \frac{k}{3} \right\rfloor$, it is easy to see how to encode the words whose weight is greater than $k - \left\lfloor \frac{k}{3} \right\rfloor$. In fact this can be accomplished by means of the tail-map

$$\tau_2 : \bigcup_{i=k-t}^{k} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\frac{k}{2}}^k$$

defined as

$$\tau_2(X) \stackrel{\text{def}}{=} \tau_1(\overline{X}) = \begin{cases} U_1(\overline{X})0^{(k-1)-l(U_1(\overline{X}))}0 & \text{if } c_{01}(\overline{X}) \geq c_{10}(\overline{X}), \\ \\ \overline{U_2(\overline{X})}1^{(k-1)-l(U_2(\overline{X}))}1 & \text{if } c_{01}(\overline{X}) < c_{10}(\overline{X}). \end{cases} \tag{3.14}$$

Note that $w(X) \geq k-t$ implies $w(\overline{X}) \leq t$.

In the case when $k \notin 6\mathbb{N}+4$, in order to encode those information words in $\mathcal{U}_t$, we must define four tail-maps

$$\tau_1 : \left\{ X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k : c_{01}(X) \geq c_{10}(X) \right\} \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k$$

$$\tau_2 : \left\{ X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k : c_{01}(X) < c_{10}(X) \right\} \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k$$

$$\tau_3 : \left\{ X \in \bigcup_{i=k-t}^{k} \mathcal{S}_i^k : c_{01}(\overline{X}) \geq c_{10}(\overline{X}) \right\} \longrightarrow \mathcal{S}^k_{\lfloor \frac{k}{2} \rfloor}$$

$$\tau_4 : \left\{ X \in \bigcup_{i=k-t}^{k} \mathcal{S}_i^k : c_{01}(\overline{X}) < c_{10}(\overline{X}) \right\} \longrightarrow \mathcal{S}^k_{\lfloor \frac{k}{2} \rfloor}.$$

where

$$\tau_1(X) \stackrel{\text{def}}{=} U_1(X)0^{k-l(U_1(X))},$$
$$\tau_2(X) \stackrel{\text{def}}{=} U_2(X)0^{k-l(U_2(X))},$$
$$\tau_3(X) \stackrel{\text{def}}{=} \overline{U_1(\overline{X})}1^{k-l(U_1(\overline{X}))},$$
$$\tau_4(X) \stackrel{\text{def}}{=} \overline{U_2(\overline{X})}1^{k-l(U_2(\overline{X}))}.$$

Note that all the functions defined above can be computed using a parallel scheme.

Example 3 *If* $r = 3$ *and* $k = 16 \in 6\mathbb{N} + 4$ *then* $t = \left\lfloor \frac{k}{3} \right\rfloor = 5$ *and* $k - t = 11$, *therefore it is possible to define the following tail-maps*

$$\tau_1 : \bigcup_{i=0}^{5} \mathcal{S}_i^{16} \longrightarrow \mathcal{S}_8^{16},$$

$$\tau_2 : \bigcup_{i=11}^{16} \mathcal{S}_i^{16} \longrightarrow \mathcal{S}_8^{16}.$$

Assume 011 *and* 110 *are the check symbols which encode* τ_1 *and* τ_2 *respectively; i.e.* $<011>= \tau_1$ *and* $<110>= \tau_2$.

Suppose

$$X = 00\,10\,01\,10\,01\,01\,00\,00 \in \mathbb{Z}_2^k,$$

is the information word that needs to be encoded. The encoder performs the following steps

1) compute

$$w(X) = 5.$$

Since $X \in \bigcup_{i=0}^{5} \mathcal{S}_i^{16}$, *then*

2) compute

$$c_{01}(00\,10\,01\,10\,01\,01\,00\,00) = 3$$

and

$$c_{10}(00\,10\,01\,10\,01\,01\,00\,00) = 2.$$

Since $3 = c_{01} > c_{10} = 2$, *according to (3.13), the encoder does:*

3) compute

$$U_1(X) = U_1(\underline{00}\ \underline{10}\ \underline{01}\ \underline{10}\ \underline{01}\ \underline{01}\ \underline{00}\ \underline{00}) = \underline{1}\ \underline{001}\ \underline{01}\ \underline{001}\ \underline{01}\ \underline{01}\ \underline{1}\ \underline{1},$$

4) append 0*'s until length is* k*:*

$$U_1(X)0^{15-l(U_1(X))}0 = 1\,001\,01\,001\,01\,01\,1\,1\,0,$$

5) append 011 *as check symbol:*

$$<011>\,(X)\ 011 = \tau_1(X)\ 011 = 1\,001\,01\,001\,01\,01\,1\,1\,0\ \ 011.$$

To decode the received word

$$\tilde{X}\tilde{Y} = 1\,001\,01\,001\,01\,01\,1\,1\,0\ \ 011,$$

reading $\tilde{Y} = 011$*, the decoder knows that the information word* X *was encoded using the function* $\tau_1 =< 011 >$*. Since the rightmost bit of* $\tilde{X}$ *is* 0*, it knows that* $c_{01}(X) > c_{10}(X)$ *and that* U_1 *was used to encode* X*. Therefore it computes*

$$U_1^{-1}(1\,001\,01\,001\,01\,01\,1\,1) = 00\,10\,01\,10\,01\,01\,00\,00 = X.$$

If

$$X = 10\,01\,11\,11\,01\,01\,10\,11 \in \mathcal{S}_{11}^{16},$$

according to (3.14), the encoded word is

$$<110>\,(X)\ 110 = \tau_2(X)\ 110 = \tau_1(\overline{X})\ 110 =$$

$$\tau_1(\underline{01}\ \underline{10}\ \underline{00}\ \underline{00}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{00})\ 110 = \underline{001}\ \underline{01}\ \underline{1}\ \underline{1}\ \underline{01}\ \underline{01}\ \underline{001}\ \underline{1}\ \underline{0}\ \ 110.$$

To decode the received word

$$\tilde{X}\tilde{Y} = 001\,01\,1\,1\,01\,01\,001\,1\,0\ \ 110,$$

reading $\tilde{Y} = 110$*, the decoder knows that the information word* X *was encoded using the function* $\tau_2 =< 110 >$*, therefore it computes*

$$X = \tau_2^{-1}(\tilde{X}) = \overline{\tau_1^{-1}(\tilde{X})} = 10\,01\,11\,11\,01\,01\,10\,11.$$

Using the maps defined in this section, balanced codes can be constructed as follows. Let $r \in [3, +\infty]$, $k \in \{7, 9, 10, 11, 13\} \cup [15, +\infty]$ and $d(k)$ be defined as (3.7) of Section 3.2. From Lemma 2 it follows

$$d(k) = k - 2t(k) - 1 = \begin{cases} k - 2\left\lceil \frac{k}{3} \right\rceil - 1 & \text{if } k \in 6\mathbb{N} + 2, \\ k - 2\left\lfloor \frac{k}{3} \right\rfloor - 1 & \text{otherwise,} \end{cases}$$

$$= \begin{cases} k - 2\frac{k+1}{3} - 1 & \text{if } k \in 6\mathbb{N} + 2, \\ k - 2\frac{k-k \bmod 3}{3} - 1 & \text{otherwise,} \end{cases}$$

$$= \begin{cases} \frac{k-5}{3} & \text{if } k \in 6\mathbb{N} + 2, \\ \frac{k+2k \bmod 3-3}{3} & \text{otherwise.} \end{cases}$$

Since $k \in \{7, 9, 10, 11, 13\} \cup [15, +\infty]$ it is possible to design a balanced code if, and only if, the following relations hold (see (3.9) and (3.10))

$$d(k) = \begin{cases} \frac{k-5}{3} & \text{if } k \in 6\mathbb{N} + 2, \\ \frac{k+2k \bmod 3-3}{3} & \text{otherwise} \end{cases} \leq \begin{cases} 2^r - 2 & \text{if } k \in 6\mathbb{N} + 4, \\ 2^r - 4 & \text{otherwise} \end{cases} \iff$$

$$\begin{cases} \frac{k-5}{3} \leq 2^r - 4 & \text{if } k \in 6\mathbb{N} + 2, \\ \frac{k+2k \bmod 3-3}{3} \leq 2^r - 2 & \text{if } k \in 6\mathbb{N} + 4, \\ \frac{k+2k \bmod 3-3}{3} \leq 2^r - 4 & \text{otherwise} \end{cases} \iff$$

$$\begin{cases} k \leq 3 \cdot 2^r - 7 & \text{if } k \in 6\mathbb{N} + 2, \\ k + 2k \bmod 3 \leq 3 \cdot 2^r - 3 & \text{if } k \in 6\mathbb{N} + 4, \\ k + 2k \bmod 3 \leq 3 \cdot 2^r - 9 & \text{otherwise} \end{cases} \iff$$

$$k \leq \begin{cases} 3 \cdot 2^r - 7 & \text{if } k \in 6\mathbb{N} + 2, \\ 3 \cdot 2^r - 5 & \text{if } k \in 6\mathbb{N} + 4, \\ 3 \cdot 2^r - 9 & \text{if } k \in 6\mathbb{N}, \\ 3 \cdot 2^r - 11 & \text{if } k \in 6\mathbb{N} + 1, \\ 3 \cdot 2^r - 9 & \text{if } k \in 6\mathbb{N} + 3, \\ 3 \cdot 2^r - 13 & \text{if } k \in 6\mathbb{N} + 5 \end{cases} \iff$$

$$k \leq \begin{cases} 3 \cdot 2^r - 8 & \text{if } k \in 6\mathbb{N} + 4, \\ 3 \cdot 2^r - 9 & \text{if } k \in 6\mathbb{N} + 3, \\ 3 \cdot 2^r - 10 & \text{if } k \in 6\mathbb{N} + 2, \\ 3 \cdot 2^r - 11 & \text{if } k \in 6\mathbb{N} + 1, \\ 3 \cdot 2^r - 12 & \text{if } k \in 6\mathbb{N}, \\ 3 \cdot 2^r - 13 & \text{if } k \in 6\mathbb{N} + 5, \end{cases}$$

this implies that the greatest number of information bits that can be encoded with r check bits is $3 \cdot 2^r - 8$.

Example 4 *If* $r = 3$ *then* $k = 3 \cdot 2^3 - 8 = 16$, $n = 19$ *and* $t(16) = \left\lfloor \frac{16}{3} \right\rfloor = 5$; *therefore, a* $DC(19, 16)$ *code can be constructed as follows*

$$< 011 >: \bigcup_{i=0}^{5} \mathcal{S}_i^{16} \longrightarrow \mathcal{S}_8^{16},$$

$$<001>: \mathcal{S}_6^{16} \longrightarrow \mathcal{S}_9^{16},$$

$$<111>: \mathcal{S}_7^{16} \longrightarrow \mathcal{S}_7^{16},$$

$$<101>: \mathcal{S}_8^{16} \longrightarrow \mathcal{S}_8^{16},$$

$$<010>: \mathcal{S}_9^{16} \longrightarrow \mathcal{S}_9^{16},$$

$$<000>: \mathcal{S}_{10}^{16} \longrightarrow \mathcal{S}_{10}^{16},$$

$$<110>: \bigcup_{i=11}^{16} \mathcal{S}_i^{16} \longrightarrow \mathcal{S}_8^{16}.$$

In this case $< 011 >$ *is the tail-map* τ_1 *defined in (3.13),* $< 110 >$ *the tail-map* τ_2 *defined in (3.14) and* $< 001 >$, $< 111 >$, $< 101 >$, $< 010 >$ *and* $< 000 >$ *are single maps. Note that actually* $<111>$, $<101>$, $<010>$, $<000>$ *are identity functions.*

3.4 Construction III

In this section we will design balanced codes with $r \geq 3$ check bits and

$$k \leq 5 \cdot 2^r - 10r + c(r), \qquad c(r) \in \{-15, -10, -5, 0, +5\}.$$

information bits. How to determine $c(r)$ exactly, will be clear in the following.

The methods presented in the previous sections are based essentially on the compression property of the unary encoding U (see (3.5)) when applied to sequences of small integers. In particular, applying U to each word X whose weight is "small", results in a word $U(X)$ whose length is smaller than $k = l(X)$. Then the saved space is used to balance $U(X)$. Construction III follows this general idea. A one-to-one mapping

$$U : \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathbb{Z}_2^{k-e}$$

it is used to compress $X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k$ and save $e \in [1, \lceil \log_2 k \rceil]$ bits. In a second step, using these e saved bits, $U(X)$ it is made balanced by means of the single maps

$$Y_w^K :: \mathcal{S}_w^{k-e} \longrightarrow \mathcal{S}_{v_w}^{k-e} \qquad Y_w \in \mathbb{Z}_2^e, \quad w \in [0, k] \tag{3.15}$$

defined by Knuth's complementation method (see Section 2.3).

As regard the function U, instead of being defined via the variable length uniquely decodable code $\{0^i 1\}_{i=0,1,2,3}$, it is defined by means of the suitably chosen variable length uniquely decodable code $\mathcal{U}$ of Table 3.1. In particular, we consider

the information word X as a sequence of $\frac{k}{5}$ five bit binary words and we encode each such word $b \in \mathbb{Z}_2^5$, to a codeword of $\mathcal{U}$. The encoding is made so that the length of the codeword associated with b is a not decreasing function of the weight of b.

The balanced code design presented here is described as follows. Assume $k \in 5\mathbb{N}$. If $\{Y_i\}_{i=1,2,\ldots,2^r}$ is the set of check symbols $\mathbb{Z}_2^r$, then

$$\begin{aligned}
&<Y_1>: \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{v_1}^k = \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k \text{ (or } \mathcal{S}_{\lfloor \frac{k}{2} \rfloor}^k), \\
&<Y_2>: \bigcup_{i=k-t}^{k} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{v_2}^k = \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k \text{ (or } \mathcal{S}_{\lfloor \frac{k}{2} \rfloor}^k), \\
&<Y_3>: \mathcal{S}_{t+1}^k \longrightarrow \mathcal{S}_{v_3}^k, \qquad (3.16) \\
&\vdots \\
&<Y_{2^r}>: \mathcal{S}_{k-t-1}^k \longrightarrow \mathcal{S}_{v_{2^r}}^k,
\end{aligned}$$

where t is the biggest integer such that

$$t \leq \frac{2k}{5} - \left\lceil \log_2 \left\lfloor \frac{k+5t+10}{10} \right\rfloor \right\rceil.$$

For such t we have (see Lemma 5)

$$t \simeq \frac{2k}{5} - \left\lceil \log_2 \frac{k}{5} \right\rceil.$$

The functions $< Y_1 >$ and $< Y_2 >$ are tail-maps designed by using the variable length code of Table 3.1, and they will be described later, whereas the functions $<Y_i>$, $i = 3, \ldots, 2^r$ are the single maps of Section 2.3. Recall that, in order for the code to be balanced, the relation

$$v_i + w(Y_i) = \left\lceil \frac{k+r}{2} \right\rceil$$

must hold for all $i = 1, 2, \ldots, 2^r$.

To define the compressing function U mentioned above, we will develop a general technique (which could have other applications besides the present one). We will design a class of encodings to variable length uniquely decodable codes which compress binary strings of low weight; i.e. if U is one of such encodings, for every binary string X of weight less than a certain fraction of $l(X)$, it is guaranteed that $l(U(X)) < l(X)$. To define such class of codes, let $s \in \mathbb{N}$ and $a, b \in \mathbb{R}$ such that $a + b = 2$. Then using the binomial formula, we have

$$2^s = (a+b)^s = \sum_{w=0}^{s} \binom{s}{w} a^w b^{s-w} \iff \sum_{w=0}^{s} \binom{s}{w} \frac{a^w b^{s-w}}{2^s} = 1 \iff$$

$$\sum_{w=0}^{s} \frac{1}{\frac{2^s}{a^w b^{s-w}}} \binom{s}{w} = 1,$$

but

$$\frac{2^s}{a^w b^{s-w}} = 2^{\log_2 \frac{2^s}{a^w b^{s-w}}} =$$

$$2^{s-w\log_2 a-(s-w)\log_2 b} = 2^{s\log_2 \frac{2}{b}+w\log_2 \frac{b}{a}},$$

therefore

$$\forall a, b \in \mathbb{R} : a+b=2 \qquad \sum_{w=0}^{s} \binom{s}{w} 2^{-(s\log_2 \frac{2}{b}+w\log_2 \frac{b}{a})} = 1.$$

Choosing $a, b \in \mathbb{R}$ such that $\log_2 \frac{b}{a} = d \in \mathbb{N}$

$$\sum_{w=0}^{s} \binom{s}{w} 2^{-\left(\left\lceil s\log_2 \frac{2^d+1}{2^d}\right\rceil + d\,w\right)} \leq \sum_{w=0}^{s} \binom{s}{w} 2^{-(s\log_2 \frac{2^d+1}{2^d}+d\,w)} = 1 \qquad (3.17)$$

holds for all $s, d \in \mathbb{N}$.

Relation (3.17), which is Kraft's inequality, implies that for each $s, d \in \mathbb{N}$ there exists a binary prefix code $\mathcal{C}_{s,d}$ with $\binom{s}{w}$ words of length $\left\lceil s\log_2 \frac{2^d+1}{2^d}\right\rceil + d\,w$, $\forall w \in [0, s]$ (see [GAL68]). Let

$$u_{s,d} : \mathbb{Z}_2^s \longrightarrow \mathcal{C}_{s,d}$$

be any encoding to $\mathcal{C}_{s,d}$ such that

$$\forall b_1, b_2 \in \mathbb{Z}_2^s \qquad w(b_1) \leq w(b_2) \iff l(u_{s,d}(b_1)) \leq l(u_{s,d}(b_2))$$

holds.

Now, given $k = ms$ $(m \in \mathbb{N})$ let

$$X = x_1 x_2 \ldots x_s \;\; x_{s+1} x_{s+2} \ldots x_{2s} \;\; \ldots \;\; x_{(m-1)s+1} x_{(m-1)s+2} \ldots x_{ms},$$

$$b_i^X = x_{(i-1)s+1} x_{(i-1)s+2} \ldots x_{is}, \qquad i = 1, \ldots, m,$$

and $U_{s,d} : \mathbb{Z}_2^k \longrightarrow \mathbb{Z}_2^+$ be the one-to-one function defined as follows.

$$U_{s,d}(X) \stackrel{\text{def}}{=} u_{s,d}(b_1^X) u_{s,d}(b_2^X) \ldots u_{s,d}(b_m^X).$$

The following lemma relates the weight of X with the length of $l(U_{s,d}(X))$ and will be useful for the construction.

Lemma 4 *If* $X \in \mathbb{Z}_2^k$ *then*

1.

$$l(U_{s,d}(X)) = \left\lceil s\log_2 \frac{2^d+1}{2^d}\right\rceil m + d\,w(X),$$

2. $\forall e \in \mathbb{N}$

$$l(U_{s,d}(X)) = \left\lceil s \log_2 \frac{2^d+1}{2^d} \right\rceil m + d\, w(X) \leq k - e \iff$$

$$w(X) \leq \left\lfloor \frac{\left\lfloor s\left(1 - \log_2 \frac{2^d+1}{2^d}\right)\right\rfloor m - e}{d} \right\rfloor .$$

Proof:

1.

$$l(U_{s,d}(X)) = \sum_{i=1}^{m} l(u_{s,d}(b_i^X)) = \sum_{i=1}^{m} \left(\left\lceil s \log_2 \frac{2^d+1}{2^d} \right\rceil + d\, w(b_i^X) \right) =$$

$$\left\lceil s \log_2 \frac{2^d+1}{2^d} \right\rceil m + d \sum_{i=1}^{m} w(b_i^X) = \left\lceil s \log_2 \frac{2^d+1}{2^d} \right\rceil m + d\, w(X).$$

2. It is easy to prove once it is noticed that $k = ms$.

■

Now, since our aim is to design tail-maps whose domain is as big as possible (that is to make t as big as possible), we need to choose the parameters s and d, in such a way that the quantity

$$\left\lfloor \frac{\left\lfloor s\left(1 - \log_2 \frac{2^d+1}{2^d}\right)\right\rfloor m - e}{d} \right\rfloor$$

of 2 of Lemma 4 is as big as possible. This lead us to choose as compressing function, the function $U = U_{5,1}$ defined by Table 3.1. This particular function is the basis for our tail-map design. Further among all the possible choices of $\mathcal{C}_{5,1}$ of Table 3.1, we choose one which guarantees $U(X)$ to be not so far from being balanced. The following lemma states some properties of this function U which are fundamental for the tail-map design.

Lemma 5 *If $k = 5m$, with $m \in \mathbb{N}$, then*

1. *if $X \in \mathbb{Z}_2^k$ then*

$$l(U(X)) = 3m + w(X).$$

2. *If $X \in \bigcup_{i=0}^{2m} \mathcal{S}_i^k$ then*

$$w(U(X)) \in \left[\left\lceil \frac{5m - w(X)}{2} \right\rceil, 3m \right],$$

i.e. $U(X)$ is not so far from being balanced.

b	$u(b)$	$l(u(b))$	$w(u(b))$	$\min w(u(b))$	$\max w(u(b))$
00000	111	3	3	3	3
10000	1101		3		
01000	1100		2		
00100	1011	4	3	2	3
00010	1010		2		
00001	0111		3		
11000	10011		3		
10100	10010		2		
10010	10001		2		
10001	01101		3		
01100	01100	5	2	2	3
01010	01011		3		
01001	01010		2		
00110	01001		2		
00101	00111		3		
00011	00110		2		
11100	100001		2		
11010	100000		1		
11001	010001		2		
10110	001011		3		
10101	001010	6	2	1	3
10011	001001		2		
01110	000111		3		
01101	000110		2		
01011	000101		2		
00111	000011		2		
11110	0100001		2		
11101	0010001		2		
11011	0001001	7	2	2	2
10111	0000101		2		
01111	0000011		2		
11111	00000011	8	2	2	2

Table 3.1: Definition of a particular $u_{5,1} : \mathbb{Z}_2^5 \longrightarrow \mathbb{Z}_2^+$.

3. *If* $t \in [0, 2m]$ *then, the number of bits saved compressing the information words* X *of weight less than or equal to* t *using the function*

$$U : \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathbb{Z}_2^+$$

is

$$e(t) \stackrel{\text{def}}{=} k - \max_{X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k} l(U(X)) = 2m - t.$$

4. *If* $t \in [0, 2m]$ *then, the number of bits needed to balance the set of binary words of length* $k - e(t)$*:*

$$\{U(X)0^{(3m+t)-l(U(X))} : X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k\} \subseteq \mathbb{Z}_2^{k-e},$$

using single maps (3.15) is

$$c(t) \stackrel{\text{def}}{=} \left\lceil \log_2 \left(\left\lfloor \frac{m+t}{2} \right\rfloor + 1 \right) \right\rceil.$$

5. *Following our strategy, we need to save enough space to balance, by means of the complementation method, the compressed versions of every information word* $X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k$*. This implies that* $e(t)$ *must be greater than or equal to* $c(t)$*. Further, since we want* t *to be as big as possible, it is fundamental to determine the biggest* $t \in \mathbb{N}$ *such that*

$$e(t) = 2m - t \geq \left\lceil \log_2 \left(\left\lfloor \frac{m+t}{2} \right\rfloor + 1 \right) \right\rceil = c(t). \tag{3.18}$$

Let $t(m)$ *be the biggest* t *such that relation (3.18) is true, then the following tight estimation of* $t(m)$ *holds.*

$$2m - \lceil \log_2 m \rceil - 1 \leq t(m) \leq 2m - \lceil \log_2 m \rceil. \tag{3.19}$$

Proof:

1. This follows from Lemma 4, letting $s = 5$ and $d = 1$.

2. $\forall b \in \mathbb{Z}_2^5$, let

$$c_b(X) \stackrel{\text{def}}{=} |\{i \in [1, m] : b_i^X = b\}|.$$

Looking at Table 3.1 it can be noticed that

$$X \in \mathcal{S}_w^k \implies w(U(X)) = \sum_{i=1}^{m} w(u(b_i^X)) = \sum_{b \in \mathbb{Z}_2^5} c_b(X) w(u(b)) \geq$$

$$c_{00000}(X) - c_{11010}(X) + 2 \sum_{b \in \mathbb{Z}_2^5} c_b(X) = c_{00000}(X) - c_{11010}(X) + 2m \geq$$

$$2m + \min_{X \in \mathcal{S}_w^k} [c_{00000}(X) - c_{11010}(X)] .$$

Now, if $w \in [0, m]$, a minimum is reached when (see Lemma 6)

$$X = (10000)^w (00000)^{m-w}.$$

On the other hand, if $w \in (m, 2m]$, a minimum is reached when

$$X = (11010)^{\left\lceil \frac{w-m}{2} \right\rceil} (10000)^{w-3\left\lceil \frac{w-m}{2} \right\rceil} (00000)^{(w-m) \bmod 2}.$$

This implies

$$w(U(X)) \geq 2m + \begin{cases} m - w(X) & \text{if } w(X) \in [0, m] \\ (w(X) - m) \bmod 2 - \left\lceil \frac{w(X)-m}{2} \right\rceil & \text{if } w(X) \in (m, 2m] \end{cases} =$$

$$\begin{cases} 3m - w(X) & \text{if } w(X) \in [0, m] \\ \left\lceil \frac{5m - w(X)}{2} \right\rceil & \text{if } w(X) \in (m, 2m] \end{cases} \geq \left\lceil \frac{5m - w(X)}{2} \right\rceil .$$

On the other hand

$$w(U(X)) = \sum_{b \in \mathbb{Z}_2^5} c_b(X) w(u(b)) \leq 3 \sum_{b \in \mathbb{Z}_2^5} c_b(X) = 3m.$$

3. Since

$$\max_{X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k} l(U(X)) = 3m + t$$

and $k = 5m$, then

$$e(t) = k - \max_{X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k} l(U(X)) = 5m - (3m + t) = 2m - t.$$

4. Let

$$w_1 = \left\lceil \frac{5m - t}{2} \right\rceil$$

and

$$w_2 = 3m.$$

From 2, the number of different weights of words in

$$W \stackrel{\text{def}}{=} \{U(X)0^{(3m+t)-l(U(X))} : X \in \bigcup_{i=0}^{t} \mathcal{S}_i^k\}$$

is $w_2 - w_1 + 1$. Since we encode W using single maps (3.15), then

$$c(t) = \lceil \log_2(w_2 - w_1 + 1) \rceil = \left\lceil \log_2 \left(\left\lfloor \frac{m+t}{2} \right\rfloor + 1 \right) \right\rceil .$$

5. From the definition of $t(m)$, the following two relations must be satisfied

$$t(m) \leq 2m - \left\lceil \log_2 \left(\left\lfloor \frac{m + t(m)}{2} \right\rfloor + 1 \right) \right\rceil , \tag{3.20}$$

$$t(m) + 1 > 2m - \left\lceil \log_2 \left(\left\lfloor \frac{m + (t(m)+1)}{2} \right\rfloor + 1 \right) \right\rceil . \tag{3.21}$$

Since $m \leq t(m) \leq 2m - 1$ the result

$$2m - \lceil \log_2 m \rceil - 1 \leq t(m) \leq 2m - \lceil \log_2 m \rceil$$

is true.

■

Now we prove the lemma used in the proof of Lemma 5.

Lemma 6 *Given $m \in \mathbb{N}$ and $X \in \mathbb{Z}_2^{5m}$, let*

$$F(X) \stackrel{\text{def}}{=} c_{00000}(X) - c_{11010}(X).$$

A minimum for F in $\mathcal{S}_w^{5m}$ is reached when

$$X = (10000)^w (00000)^{m-w}$$

and $w \in [0, m]$ or when

$$X = (11010)^{\lceil \frac{w-m}{2} \rceil} (10000)^{w-3\lceil \frac{w-m}{2} \rceil} (00000)^{(w-m) \bmod 2}.$$

and $w \in (m, 2m]$.

Proof: Let $X \in \mathcal{S}_w^{5m}$ be a string such that $F(X)$ is minimum in $\mathcal{S}_w^{5m}$. Without loss of generality, it is possible to assume

$$X = (11111)^a (11110)^b (11010)^c (11000)^d (10000)^e (00000)^f$$

where

$$w = 5a + 4b + 3c + 2d + 1e + 0f$$

and

$$m = a + b + c + d + e + f.$$

Since $w \leq 2m$, then

$$a \geq 1 \quad \Longrightarrow \quad e + f \geq 1.$$

Replacing each subsequence (11111)(00000) or (11111)(10000) with (11010)(11000) or $(11010)^2$ respectively, it is possible to assume $a = 0$.

Since $w \leq 2m$, then

$$b \geq 1 \quad \Longrightarrow \quad e + f \geq 1.$$

Replacing each subsequence (11110)(10000) or (11110)(00000) with (11010)(11000) or $(11000)^2$ respectively, it is possible to assume $b = 0$.

By replacing each subsequence $(11000)^2$ with (11010)(10000) it is possible to assume $d = 0$ or $d = 1$. However,

$$a = 0, b = 0, d = 1 \text{ and } w \leq 2m \qquad \Longrightarrow \qquad e + f \geq 1.$$

Therefore, by replacing either the subsequence (11000)(10000) or (11000)(00000) with (11010)(00000) or $(10000)^2$ respectively, it is actually possible to assume $d = 0$.

From the above it is possible then to assume

$$X = (11010)^c(10000)^e(00000)^f$$

where

$$\begin{cases} m = c + e + f, \\ \\ w = 3c + e. \end{cases} \tag{3.22}$$

The last relations imply

$$w - m = 2c - f \qquad \Longrightarrow \qquad f \bmod 2 = (\mathrm{w} - \mathrm{m}) \bmod 2. \tag{3.23}$$

Now, if $c \neq 0$ and $f \geq 2$ then the string

$$X' = (11010)^{c-1}(10000)^{e+3}(00000)^{f-2}$$

is such that

$$F(X') = (f - 2) - (c - 1) = f - c - 1 < f - c = F(X),$$

contradicting the assumption that X is a minimum for F in $\mathcal{S}_w^k$. This means that only the following cases are left.

1. $c = 0$. This implies $w \in [0, m]$, $e = w$ and $f = m - w$ (see (3.22)).

2. $c \neq 0$ and $f = (w - m) \bmod 2 = 0$. This implies $w \in (m, 2m]$, $c = \frac{w-m}{2}$ and $e = w - 3\frac{w-m}{2}$ (see (3.22) and (3.23)).

3. $c \neq 0$ and $f = (w - m) \bmod 2 = 1$. This implies $w \in (m, 2m]$, $c = \frac{w-m+1}{2}$ and $e = w - 3\frac{w-m+1}{2}$ (see (3.22) and (3.23)).

■

Now we can give the tail-map constructions and the encoding/decoding procedures. Given $k = 5m$, $m \in \mathbb{N}$, let $t = t(m)$ be the number defined by 5 of Lemma 5, i.e. the biggest $t \in \mathbb{N}$ such that

$$e(t) = 2m - t \geq \left\lceil \log_2 \left(\left\lfloor \frac{m+t}{2} \right\rfloor + 1 \right) \right\rceil = c(t).$$

5 of Lemma 5 implies that

$$t \simeq 2m - \lceil \log_2 m \rceil = \frac{2k}{5} - \left\lceil \log_2 \frac{k}{5} \right\rceil.$$

Further, let

$$w_1 \stackrel{\text{def}}{=} \min_{X : w(X) \leq t} w(U(X)) = \left\lceil \frac{5m - t}{2} \right\rceil,$$

$$w_2 \stackrel{\text{def}}{=} \max_{X : w(X) \leq t} w(U(X)) = 3m,$$

$$k^* \stackrel{\text{def}}{=} k - e(t)$$

and

$$U^* : \mathcal{S}_0^k \cup \mathcal{S}_1^k \cup \ldots \cup \mathcal{S}_t^k \longrightarrow \mathcal{S}_{w_1}^{k^*} \cup \mathcal{S}_{w_1+1}^{k^*} \cup \ldots \cup \mathcal{S}_{w_2}^{k^*}$$

be the function defined as follows

$$U^*(X) \stackrel{\text{def}}{=} U(X) 0^{k^* - l(U(X))}.$$

Note that $U^*(X)$ is nothing but $U(X)$ followed by enough 0's to make $U^*(X)$ of length equal to $k^* = k - e(t)$. Further, the weight of $U^*(X)$ is between w_1 and w_2.

Given $Y_{w_1}^K, Y_{w_1+1}^K \ldots, Y_{w_2}^K \in \mathbb{Z}_2^e$, let

$$< Y_w^K > : \mathcal{S}_w^{k^*} \longrightarrow \mathcal{S}_{v_w}^{k^*} \qquad \forall w \in [w_1, w_2]$$

be single maps (3.15) such that

$$v_w + w(Y_w^K) = \left\lceil \frac{k^* + e}{2} \right\rceil = \left\lceil \frac{k}{2} \right\rceil \qquad \forall w \in [w_1, w_2].$$

Now it is possible to define the tail-map

$$\tau_1 : \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k, \tag{3.24}$$

as follows

$$\tau_1(X) \stackrel{\text{def}}{=} \begin{cases} <Y_{w_1}^K> (U^*(X))\, Y_{w_1}^K & \text{if } w(U^*(X)) = w_1, \\ <Y_{w_1+1}^K> (U^*(X))\, Y_{w_1+1}^K & \text{if } w(U^*(X)) = w_1 + 1, \\ \vdots & \\ <Y_{w_2}^K> (U^*(X))\, Y_{w_2}^K & \text{if } w(U^*(X)) = w_2. \end{cases} \tag{3.25}$$

Note that, by complementing $\tau_1(X)$ it is possible to define an equivalent tail-map

$$\tau_1' : \bigcup_{i=0}^{t} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\lfloor \frac{k}{2} \rfloor}^k.$$

Further, in order to encode those words whose weight is greater or equal than $k-t$, we can use the tail-map

$$\tau_2 : \bigcup_{i=k-t}^{k} \mathcal{S}_i^k \longrightarrow \mathcal{S}_{\lceil \frac{k}{2} \rceil}^k \text{ (or } \mathcal{S}_{\lfloor \frac{k}{2} \rfloor}^k) \tag{3.26}$$

defined as

$$\tau_2(X) \stackrel{\text{def}}{=} \tau_1(\overline{X}) \text{ (or } \overline{\tau_1(\overline{X})}). \tag{3.27}$$

Given an information word

$$X = b_1^X\, b_2^X\, \ldots\, b_m^X \in \mathbb{Z}_2^k,$$

the algorithm to compute $\tau_1(X)$ can be described as follows. Assume $w(X) \leq t$ (otherwise X is encoded either by single maps or by means of τ_2). The encoder performs the following steps:

1) compute (u is the function defined in Table 3.1)

$$C = U(X) = u(b_1^X)\, u(b_2^X)\, \ldots\, u(b_m^X) \in \mathbb{Z}_2^+.$$

5 and 3 of Lemma 5 imply that $l(C) \leq k^* = k - e(t)$.

2) Add 0's until $l(C)$ becomes k^*; i.e. compute

$$C = C\, 0^{k^* - l(C)} = U^*(X).$$

3) Compute $w = w(C)$. 2 of Lemma 5 implies that $w \in [w_1, w_2]$.

4) Balance C using Knuth's single maps; i.e. compute

$$\tau_1(X) = < Y_w^K > (C)\, Y_w^K.$$

Note that

$$l(\tau_1(X)) = l(< Y_w^K > (C)) + l(Y_w^K) = k^* + e(t) = k.$$

Example 5 *If* $r = 5$ *and* $m = 21$ *then*

$$\begin{aligned}
&k = 5m = 105,\\
&t(21) = 37,\\
&k - t(21) = 68,\\
&e(37) = 5,\\
&k^* = 100,\\
&w_1 = 34,\\
&w_2 = 63,\\
&k^*/2 = 50,\\
&\lceil (k^* + e)/2 \rceil = \lceil k/2 \rceil = 53,\\
&\lceil (k + r)/2 \rceil = 55.
\end{aligned}$$

therefore it is possible to define the following tail-map

$$\tau_1 : \mathcal{S}_0^{105} \cup \mathcal{S}_1^{105} \cup \ldots \cup \mathcal{S}_{37}^{105} \longrightarrow \mathcal{S}_{53}^{105},$$

Assume 00101 *is the check symbol which encodes* τ_1 *(i.e.* $< 00101 >= \tau_1$*) and that* τ_1 *has been defined using the single maps* $Y_w^K :: \mathcal{S}_w^{100} \longrightarrow \mathcal{S}_{v_w}^{100}$ *shown in Table 3.2. If*

$$X = (11010)^5 (11100)^3 (01000)^4 (10000)^8 (00000)^1 \in \mathbb{Z}_2^{105},$$

is the information word that needs to be encoded, the encoder does

1) compute $w(X) = 36$. *Since* $X \in \bigcup_{i=0}^{37} \mathcal{S}_i^{105}$, *then*

2) compute

$$C = U(X) = (100000)^5 (100001)^3 (1100)^4 (1101)^8 (111)^1,$$

3) add 0*'s until* $l(C)$ *becomes* $k^* = 100$*; i.e. compute*

$$C = U^*(X) = (100000)^5 (100001)^3 (1100)^4 (1101)^8 (111)^1 0,$$

Y_w^K	w	v_w	Y_w^K	w	v_w
00111	34	50	10101	43	50
11000	35	51	01010	57	51
01011	36	50	11001	44	50
01111	37	49	00110	56	51
10000	63	52	01110	45	50
10111	38	49	10001	55	51
01000	62	52	10110	46	50
11011	39	49	01001	54	51
00100	61	52	11010	47	50
11101	40	49	00000	53	53
00010	60	52	11111	48	48
10011	41	50	00001	52	52
01100	59	51	11110	49	49
01101	42	50	00011	51	51
10010	58	51	11100	50	50

Table 3.2: Choice of Y_w^K and v_w defining single maps $Y_w^K :: \mathcal{S}_w^{100} \longrightarrow \mathcal{S}_{v_w}^{100}$, for $w \in [w_1, w_2]$.

4) compute $w = w(C) = 46$. *Since* $C \in \mathcal{S}_{46}^{100}$, *then*

5) balance C *using Knuth's single maps; i.e. compute*

$$\tau_1(X) = <10110> (C)\,10110 =$$

$$011111(100000)^4(100001)^3(1100)^4(1101)^8(111)^10\ 10110.$$

6) append the check 00101 *which encodes* τ_1. *The encoding of* X *is then* $\mathcal{E}(X) =$

$$011111(100000)^4(100001)^3(1100)^4(1101)^8(111)^10\ 10110\ 00101,$$

which is a balanced word.

Let

$$\tilde{X}\tilde{Y}^K\tilde{Y} = 011111(100000)^4(100001)^3(1100)^4(1101)^8(111)^10\ 10110\ 00101,$$

be the received word. Reading $\tilde{Y} = 00101$, *the decoder knows that the information word* X *was encoded using the function* $\tau_1 = < 00101 >$. *Reading* $\tilde{Y}^K = 10110$ *it knows that* $U^*(X)$ *was encoded using the single map* $< 10110 >$; *therefore, it computes*

1)

$$<10110>^{-1} (\tilde{X}) = (100000)^5(100001)^3(1100)^4(1101)^8(111)^10,$$

2)

$$U^{*-1}(<10110>^{-1} (\tilde{X})) = (11010)^5(11100)^3(01000)^4(10000)^8(00000)^1 = X.$$

Before giving an example of the whole balanced code design, we will estimate how powerful is the method presented in this section; namely we will estimate the maximum number of information bits that it is possible to balance using r check bits. We have the following

Theorem 3 *Given* $r \in [3, +\infty]$ *let* $k(r)$ *be the greatest integer* $k \in 5\mathbb{N}$ *such that a* $DC(k + r, k)$ *code can be constructed using tail-maps (3.24) and (3.26). Then*

$$5 \cdot 2^r - 10r - 15 \leq k(r) \leq 5 \cdot 2^r - 10r + 5.$$

Proof: Let $m = m(r) \stackrel{\text{def}}{=} \frac{k(r)}{5}$, $t = t(m)$ be the numbers defined by 5 of Lemma 5 and $d(k)$ be defined as (3.7) of Section 3.2.

From the definition of $k(r)$

$$d(5 \cdot m) = 5 \cdot m - 2t(m) - 1 \leq 2^r - 2, \tag{3.28}$$

and

$$d(5 \cdot (m+1)) = 5 \cdot (m+1) - 2t(m+1) - 1 > 2^r - 2. \tag{3.29}$$

From (3.21) and (3.19) it follows

$$t(m+1) \geq 2(m+1) - \left\lceil \log_2 \left(\left\lfloor \frac{(m+1) + [t(m+1)+1]}{2} \right\rfloor + 1 \right) \right\rceil \iff$$

$$-2t(m+1) \leq -4(m+1) + 2 \left\lceil \log_2 \left(\left\lfloor \frac{m + t(m+1) + 2}{2} \right\rfloor + 1 \right) \right\rceil \leq$$

$$-4(m+1) + 2 \left\lceil \log_2 \left(\left\lfloor \frac{m + 2(m+1) - \lceil \log_2(m+1) \rceil + 2}{2} \right\rfloor + 1 \right) \right\rceil =$$

$$-4(m+1) + 2 \left\lceil \log_2 \left(\left\lfloor \frac{3m - \lceil \log_2(m+1) \rceil + 4}{2} \right\rfloor + 1 \right) \right\rceil \leq$$

$$-4(m+1) + 2 \left\lceil \log_2 \left(\left\lfloor \frac{3m - \lceil \log_2(m+1) \rceil + 4}{2} \right\rfloor + 1 \right) \right\rceil \leq$$

$$-4(m+1) + 2 \left\lceil \log_2 \left(\left\lfloor \frac{3m + (6 - \lceil \log_2(m+1) \rceil)}{2} \right\rfloor \right) \right\rceil .$$

Since, for $m > 3$, $6 - \lceil \log_2(m+1) \rceil \leq m$, it follows

$$-2t(m+1) \leq -4(m+1) + 2 \lceil \log_2 m \rceil + 2.$$

Substituting this relation in (3.29)

$$5(m+1) - 4(m+1) + 2 \lceil \log_2 m \rceil + 2 > 2^r - 1 \iff$$

$$m + 1 + 2 \lceil \log_2 m \rceil + 2 > 2^r - 1 \iff$$

$$m \geq 2^r - 2 \lceil \log_2 m \rceil - 3. \tag{3.30}$$

On the other hand, from (3.20) and (3.19)

$$t(m) \leq 2m - \left\lceil \log_2 \left(\left\lfloor \frac{m + t(m)}{2} \right\rfloor + 1 \right) \right\rceil \iff$$

$$-2t(m) \geq -4m + 2 \left\lceil \log_2 \left(\left\lfloor \frac{m + t(m)}{2} \right\rfloor + 1 \right) \right\rceil \geq$$

$$-4m + 2 \left\lceil \log_2 \left(\left\lfloor \frac{m + 2m - \lceil \log_2 m \rceil - 1}{2} \right\rfloor + 1 \right) \right\rceil =$$

$$-4m + 2\left\lceil \log_2 \left(\left\lfloor \frac{3m + 1 - \lceil \log_2 m \rceil}{2} \right\rfloor \right) \right\rceil \geq$$
$$-4m + 2\left\lceil \log_2 \left(\left\lfloor \frac{2m}{2} \right\rfloor \right) \right\rceil \iff$$
$$-2t(m) \geq -4m + 2\lceil \log_2 m \rceil .$$

Substituting this relation in (3.28)

$$5m - 4m + 2\lceil \log_2 m \rceil \leq 2^r - 1 \iff$$
$$m \leq 2^r - 2\lceil \log_2 m \rceil - 1. \tag{3.31}$$

Relation (3.30), together with (3.31), gives

$$2^r - 2\lceil \log_2 m(r) \rceil - 3 \leq m(r) \leq 2^r - 2\lceil \log_2 m(r) \rceil - 1. \tag{3.32}$$

Given

$$\forall r \in [5, +\infty] \qquad 2^{r-1} \leq m(r) \leq 2^r$$

and (3.32), it follows

$$\forall r \in [5, +\infty] \qquad 2^r - 2r - 3 \leq m(r) \leq 2^r - 2r + 1,$$

which means (for $r = 3$ or 4 the previous relations hold as well)

$$\forall r \in [3, +\infty] \qquad 5 \cdot 2^r - 10r - 15 \leq k(r) \leq 5 \cdot 2^r - 10r + 5.$$

■

Example 6 *For $r = 5$, looking at Table 3.3, the reader can see that $m(5) = 21$ and $t(21) = 37$, which means $k = 5m = 105$. A DC$(110, 105)$ code can be constructed as follows*

$$<00101>: \bigcup_{i=0}^{37} \mathcal{S}_i^{105} \longrightarrow \mathcal{S}_{53}^{105}, \qquad <11010>: \bigcup_{i=68}^{105} \mathcal{S}_i^{105} \longrightarrow \mathcal{S}_{52}^{105},$$

$$\begin{array}{lll}
<01111>: \mathcal{S}_{38}^{105} \longrightarrow \mathcal{S}_{51}^{105}, & <01110>: \mathcal{S}_{48}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <00110>: \mathcal{S}_{58}^{105} \longrightarrow \mathcal{S}_{53}^{105}, \\
<10111>: \mathcal{S}_{39}^{105} \longrightarrow \mathcal{S}_{51}^{105}, & <10110>: \mathcal{S}_{49}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <01010>: \mathcal{S}_{59}^{105} \longrightarrow \mathcal{S}_{53}^{105}, \\
<11011>: \mathcal{S}_{40}^{105} \longrightarrow \mathcal{S}_{51}^{105}, & <11111>: \mathcal{S}_{50}^{105} \longrightarrow \mathcal{S}_{50}^{105}, & <10010>: \mathcal{S}_{60}^{105} \longrightarrow \mathcal{S}_{53}^{105}, \\
<11101>: \mathcal{S}_{41}^{105} \longrightarrow \mathcal{S}_{51}^{105}, & <11110>: \mathcal{S}_{51}^{105} \longrightarrow \mathcal{S}_{51}^{105}, & <01100>: \mathcal{S}_{61}^{105} \longrightarrow \mathcal{S}_{53}^{105}, \\
<00111>: \mathcal{S}_{42}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <11100>: \mathcal{S}_{52}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <10100>: \mathcal{S}_{62}^{105} \longrightarrow \mathcal{S}_{53}^{105}, \\
<01011>: \mathcal{S}_{43}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <00011>: \mathcal{S}_{53}^{105} \longrightarrow \mathcal{S}_{53}^{105}, & <11000>: \mathcal{S}_{63}^{105} \longrightarrow \mathcal{S}_{53}^{105}, \\
<10011>: \mathcal{S}_{44}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <00001>: \mathcal{S}_{54}^{105} \longrightarrow \mathcal{S}_{54}^{105}, & <00010>: \mathcal{S}_{64}^{105} \longrightarrow \mathcal{S}_{54}^{105}, \\
<01101>: \mathcal{S}_{45}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <00000>: \mathcal{S}_{55}^{105} \longrightarrow \mathcal{S}_{55}^{105}, & <00100>: \mathcal{S}_{65}^{105} \longrightarrow \mathcal{S}_{54}^{105}, \\
<10101>: \mathcal{S}_{46}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <01001>: \mathcal{S}_{56}^{105} \longrightarrow \mathcal{S}_{53}^{105}, & <01000>: \mathcal{S}_{66}^{105} \longrightarrow \mathcal{S}_{54}^{105}, \\
<11001>: \mathcal{S}_{47}^{105} \longrightarrow \mathcal{S}_{52}^{105}, & <10001>: \mathcal{S}_{57}^{105} \longrightarrow \mathcal{S}_{53}^{105}, & <10000>: \mathcal{S}_{67}^{105} \longrightarrow \mathcal{S}_{54}^{105}.
\end{array}$$

In this case $< 00101 >= \tau_1$ is defined by (3.24), $< 11010 >= \tau_2$ is defined by (3.26) and $< 11111 >$, $< 11110 >$, $< 11100 >$, $< 00011 >$, $< 00001 >$, $<00000>$ are identity functions. The remaining $<Y>$'s are proper single maps.

r	$t(m(r))$	$m(r)$
3	4	3
4	10	7
5	37	21
6	91	49
7	214	111
8	465	237
9	972	491
10	1991	1001
11	4034	2023
12	8125	4069
13	16312	8163

Table 3.3: Values of $m(r)$ and $t(m(r))$ for $r \in [3, 13]$.

3.5 Comparisons

In this section, before showing the comparisons among our methods with the method proposed in [ALB94], we give a closed form for the function $k_{max}(r)$: the maximum integer such that there exists a balanced code with r check bits and $k(r)$ information bits.

Given $n \in \mathbb{N}$, the following relations hold [LON80]

$$\sqrt{2\pi n} n^n e^{-n} e^{\frac{1}{12n+1}} < n! < \sqrt{2\pi n} n^n e^{-n} e^{\frac{1}{12n}}. \tag{3.33}$$

These relations could be used to define Stirling's famous approximation formula

$$n! \simeq \sqrt{2\pi n} n^n e^{-n}.$$

The following theorem holds.

Theorem 4 *Given $r \in \mathbb{N}$, if $k = k(r)$ is the greatest integer such that a*

$$DC(k + r, k)$$

code exists, then

1.

$$n = k + r \in 2\mathbb{N}, \tag{3.34}$$

2.
$$n = 2\left\lfloor \frac{2^{2r}}{\pi} e^{-\frac{1}{2n}} \right\rfloor,$$

3.
$$\frac{2^{2r}}{\pi} e^{-\frac{\pi}{2^{2r+1.94}}} - 1 < \frac{n}{2} < \frac{2^{2r}}{\pi} e^{-\frac{\pi}{2^{2r+2}}} < \frac{2^{2r}}{\pi}, \tag{3.35}$$

4.
$$\frac{2^{2r}}{\pi} - \left[\frac{2^{2r}}{\pi} e^{-\frac{\pi}{2^{2r+1.94}}} - 1 \right] < 1.27, \tag{3.36}$$

5.
$$n = 2\left\lfloor \frac{2^{2r}}{\pi} \right\rfloor - (1 \pm 1).$$

Proof:

1. From the definition of k, the inequalities
$$2^k \leq \binom{k+r}{\left\lceil \frac{k+r}{2} \right\rceil} < \binom{(k+1)+r}{\left\lceil \frac{(k+1)+r}{2} \right\rceil} < 2^{k+1} \tag{3.37}$$
hold. If
$$n = k + r \in 2\mathbb{N} + 1$$
then
$$2^{k+1} \leq 2\binom{k+r}{\left\lceil \frac{k+r}{2} \right\rceil} = \binom{(k+1)+r}{\frac{(k+1)+r}{2}},$$
contradicting the third relation of (3.37). It is possible then to write (3.37) and (3.34) in the following way
$$2^k \leq \binom{k+r}{\frac{k+r}{2}} < \binom{(k+1)+r}{\left\lceil \frac{(k+1)+r}{2} \right\rceil} < 2^{k+1}. \tag{3.38}$$

2. If $n \in 2\mathbb{N}$, then
$$\binom{n}{\frac{n}{2}} = \frac{n!}{\left[\left(\frac{n}{2}\right)!\right]^2},$$
therefore, from (3.33), it follows
$$\frac{\sqrt{2\pi n} n^n e^{-n} e^{\frac{1}{12n+1}}}{\left[\sqrt{\pi n}\left(\frac{n}{2}\right)^{\frac{n}{2}} e^{-\frac{n}{2}} e^{\frac{1}{12\left(\frac{n}{2}\right)}}\right]^2} < \binom{n}{\frac{n}{2}} < \frac{\sqrt{2\pi n} n^n e^{-n} e^{\frac{1}{12n}}}{\left[\sqrt{\pi n}\left(\frac{n}{2}\right)^{\frac{n}{2}} e^{-\frac{n}{2}} e^{\frac{1}{12\left(\frac{n}{2}\right)+1}}\right]^2} \iff$$
$$\sqrt{\frac{2}{\pi n}} 2^n e^{\left(\frac{1}{12n+1} - \frac{1}{3n}\right)} < \binom{n}{\frac{n}{2}} < \sqrt{\frac{2}{\pi n}} 2^n e^{\left(\frac{1}{12n} - \frac{1}{6n+1}\right)} \iff$$

$$\sqrt{\frac{2}{\pi n}}2^n e^{-\frac{9n+1}{3n(12n+1)}} < \binom{n}{\frac{n}{2}} < \sqrt{\frac{2}{\pi n}}2^n e^{-\frac{18n-1}{12n(6n+1)}}. \tag{3.39}$$

If $n \in 2\mathbb{N}+1$, since

$$\binom{n}{\lceil \frac{n}{2} \rceil} = \frac{1}{2}\binom{n+1}{\frac{n+1}{2}}$$

and (3.39), it follows

$$\sqrt{\frac{2}{\pi(n+1)}}2^n e^{-\frac{9n+10}{3(n+1)(12n+13)}} < \binom{n}{\lceil \frac{n}{2} \rceil} < \sqrt{\frac{2}{\pi(n+1)}}2^n e^{-\frac{18n+17}{12(n+1)(6n+7)}}. \tag{3.40}$$

A lower bound for k can be determined as follows. (3.40) and (3.38) imply

$$2^{k+1} > \binom{(k+1)+r}{\lceil \frac{(k+1)+r}{2} \rceil} > \sqrt{\frac{2}{\pi(k+r+2)}}2^{k+r+1}e^{-\frac{9n+19}{3(n+2)(12n+25)}} \iff$$

$$\not{k}+\not{1} > \not{k}+r+\not{1} - \frac{1}{2}\log_2(k+r+2) + \frac{1}{2}\log_2\frac{2}{\pi} + \log_2 e^{-\frac{9n+19}{3(n+2)(12n+25)}} \iff$$

$$\frac{1}{2}\log_2(k+r+2) > r + \frac{1}{2}\log_2\frac{2}{\pi} + \log_2 e^{-\frac{9n+19}{3(n+2)(12n+25)}} \iff$$

$$\log_2(k+r+2) > 2r + \log_2\frac{2}{\pi} + \log_2 e^{-\frac{18n+38}{3(n+2)(12n+25)}} \iff$$

$$k+r+2 > \frac{2}{\pi}2^{2r}e^{-\frac{18n+38}{3(n+2)(12n+25)}} \iff$$

$$k > \frac{2^{2r+1}}{\pi}e^{-\frac{18n+38}{3(n+2)(12n+25)}} - r - 2. \tag{3.41}$$

An upper bound for k can be determined as follows. (3.39) and (3.38) imply

$$2^k \le \binom{k+r}{\frac{k+r}{2}} < \sqrt{\frac{2}{\pi(k+r)}}2^{k+r}e^{-\frac{18n-1}{12n(6n+1)}} \iff$$

$$\not{k} < \not{k}+r - \frac{1}{2}\log_2(k+r) + \frac{1}{2}\log_2\frac{2}{\pi} + \log_2 e^{-\frac{18n-1}{12n(6n+1)}} \iff$$

$$\frac{1}{2}\log_2(k+r) < r + \frac{1}{2}\log_2\frac{2}{\pi} + \log_2 e^{-\frac{18n-1}{12n(6n+1)}} \iff$$

$$\log_2(k+r) < 2r + \log_2\frac{2}{\pi} + \log_2 e^{-\frac{18n-1}{6n(6n+1)}} \iff$$

$$k+r < \frac{2}{\pi}2^{2r}e^{-\frac{18n-1}{6n(6n+1)}} \iff$$

$$k < \frac{2^{2r+1}}{\pi}e^{-\frac{18n-1}{6n(6n+1)}} - r. \tag{3.42}$$

(3.41) and (3.42) give

$$\frac{2^{2r+1}}{\pi}e^{-\frac{18n+38}{3(n+2)(12n+25)}} - 2 < n < \frac{2^{2r+1}}{\pi}e^{-\frac{18n-1}{6n(6n+1)}}. \tag{3.43}$$

Since

$$e^{-\frac{1}{2(n+1)}} \leq e^{-\frac{18n+38}{3(n+2)(12n+25)}}$$

and

$$e^{-\frac{18n-1}{6n(6n+1)}} \leq e^{-\frac{1}{2n}}$$

then

$$\frac{2^{2r+1}}{\pi}e^{-\frac{1}{2(n+1)}} - 2 < n < \frac{2^{2r+1}}{\pi}e^{-\frac{1}{2n}}. \tag{3.44}$$

The statement follows by noticing that

$$\frac{2^{2r+1}}{\pi}e^{-\frac{1}{2n}} - \left[\frac{2^{2r+1}}{\pi}e^{-\frac{1}{2(n+1)}} - 2\right] < 2.$$

3. (3.44) is equivalent to

$$\frac{1}{\pi}2^{2r+1-\frac{\log_2 e}{2(n+1)}} - 2 < n < \frac{1}{\pi}2^{2r+1-\frac{\log_2 e}{2n)}}.$$

Now

$$\forall r \in [3, +\infty] \qquad 0.98 \leq 1 - \frac{\log_2 e}{2(n(3)+1)} = 1 - \frac{\log_2 e}{2(40+1)} \leq 1 - \frac{\log_2 e}{2(n(r)+1)}$$

and

$$\forall r \in [1, +\infty] \qquad 1 - \frac{\log_2 e}{2n(r)} \leq 1$$

therefore

$$\frac{2^{2r+0.94}}{\pi} - 1 \leq \frac{2^{2r+0.98}}{\pi} - 2 < n < \frac{2^{2r+1}}{\pi}.$$

The statement follows by substituting these relations in (3.44) and noticing that (3.35) is true when $r = 1$ or 2.

4.

$$\frac{2^{2r}}{\pi} - \left[\frac{2^{2r}}{\pi}e^{-\frac{\pi}{2^{2r+1.94}}} - 1\right] =$$

$$\frac{2^{2r}}{\pi}\left[1 - e^{-\frac{\pi}{2^{2r+1.94}}}\right] + 1 \leq$$

$$\frac{2^{2r}}{\pi} \cdot \frac{\pi}{2^{2r+1.94}} + 1 = \frac{1}{2^{1.94}} + 1 < 1.27.$$

5. It follows from (3.34), (3.35) and (3.36), once it is noticed that there exist at most two integers in an interval of length less than or equal to 1.27.

■

Some comparisons are shown in Tables 3.4 and 3.5.

r	Con. in [ALB94], $k =$ $2^{r+1}-\lceil 0.8\sqrt{r}\rceil-2$	Con. I, $k =$ $2^{r+1}-2$	Con. II, $k =$ $3\cdot 2^{r}-8$	Con. III, $k =$ $5\cdot 2^{r}-10r+c(r)$	$k_{max} =$ $2\left\lfloor\frac{2^{2r}}{\pi}\right\rfloor-r-(1\pm 1)$
1	-	-	-	-	1
2	4	6	-	-	6
3	12	14	16	15	37
4	28	30	40	35	158
5	60	62	88	105	645
6	124	126	184	245	2600
7	251	254	376	555	10421
8	507	510	760	1185	41712
9	1019	1022	1528	2455	166875
10	2043	2046	3064	5005	667532
11	4091	4094	6136	10115	2670165
12	8187	8190	12280	20345	10680694
13	16379	16382	24568	40815	42722815

Table 3.4: Comparisons of various constructions. r and k are the number of check and information bits respectively. In the fifth column $c(r)\in\{-15,-10,-5,0,+5\}$.

$k = 2^p$	Con. in [ALB94], $r =$	Con. I, $r =$	Con. II, $r =$	Con. III, $r =$	$r_{min} =$
4	2	2	2	-	2
8	3	3	3	3	3
16	4	4	3	4	3
32	5	5	4	4	3
64	6	6	5	5	4
128	7	7	6	6	4
256	8	8	7	7	5
512	9	9	8	7	5
1024	10	10	9	8	6
2048	11	11	10	9	6
4096	12	12	11	10	7
8192	13	13	12	11	7
16384	14	14	13	12	8
32768	15	15	14	13	8

Table 3.5: Comparisons of the number of check bits required when k is a power of 2. r_{min} is the minimum number of check bits required to have a DC$(k + r, k)$ code.

Chapter 4
Future Research

The research can proceed in two directions. The first one is to improve the new scheme presented in this thesis finding some other functions like Knuth's map and tail-maps to further reduce the redundancy. In order to do this, the new maps should 1) be very easy to compute, 2) be very "adherent", 3) have a domain as big as possible, and 4) have as codomain a constant weight code $\mathcal{S}_v^k$ with v close to $\frac{k}{2}$. Given a function $f : D \longrightarrow C$, we say that f is adherent iff f is one-to-one and the cardinality of the domain is almost equal to the cardinality of the codomain. In more precise terms we can define a parameter which gives an indication of how adherent is the function f, as follows

$$\alpha_f \stackrel{\text{def}}{=} \frac{\log_2 |D|}{\log_2 |C|} \leq 1.$$

For example, it is easy to prove that the adherence of the tail maps defined in Section 3.4 is greater than or equal to 0.9710, which is very close to 1.

A second direction of future research is to find efficient design methods for parallel encoding and decoding of balanced codes. All the methods known so far have sequential encoding. However in some VLSI applications (Noise Reduction in VLSI Systems) it is required to have parallel encoding and parallel decoding schemes for balanced codes [TAB90, STO91]. To our knowledge no efficient method is available in the literature for this problem.

Bibliography

[ALB94] S. Al-Bassam and B. Bose, *Design of efficient balanced codes*, IEEE Trans. Comput., vol. 43, pp. 362–365, March 1994.

[ALB90] S. Al-Bassam and B. Bose, *On balanced codes*, IEEE Trans. Inform. Theory, vol. 36, pp. 406–408, March 1990.

[ALB93] S. Al-Bassam and B. Bose, *Conservative and balanced codes*, Proc. IEEE 1993 Int. Symp. Inform. Theory, page 9, Jan 1993.

[ALO88] N. Alon, E. E. Bergmann, D. Coppersmith and A. M. Odlyzko, *Balancing sets of vectors.* IEEE Trans. Inform. Theory, vol. 34, no. 1, pp. 128–130, Jan. 1988.

[BEG86] E. E. Bergmann, A. M. Odlyzko and S. H. Sangani, *Half weight block codes for optical communications*, AT&T Technical Journal, vol. 65, pp. 85–93, May 1986.

[BOS91] B. Bose, *On unordered codes*, IEEE Trans. Comput., vol 40, pp. 125–131, Feb. 1991.

[BER61] J. M. Berger, *A note on error detecting codes for asymmetric channels*, Inform. Contr., vol 4, pp. 68–73, March 1961.

[CHO85] B. Z. Chor, *Two Issues in Public Key Cryptography: RSA Bit Security and a New Knapsack Type System*, MIT Press, Cambridge, Massachusetts, 1985.

[FRE62] C. V. Freiman, *Optimal error detecting codes for completely asymmetric binary channels*, Inform. Contr., vol 5, pp. 64–71, March 1962.

[GAL68] R. G. Gallager *Information Theory and Reliable Communication*, New York, Wiley 1968.

[KNU86] D. E. Knuth, *Efficient balanced codes*, IEEE Trans. Inform. Theory, vol. IT-32, pp. 51–53, Jan. 1986.

[LEI84] E. L. Leiss, *Data integrity in digital optical Disks*, IEEE Trans. Comput., vol c-33, pp. 818–827, Sept. 1984.

[LON80] G. Longo, *Teoria dell'Informazione*, Serie di Informatica, Boringhieri, 1980.

[MAC77] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, 21, page 530, North-Holland, 1977.

[OFE90] Y. Ofek, *The conservative code for bit synchronization*, IEEE Trans. Comm., vol 38, pp. 1107–1113, July 1990.

[SAI91] Y. Saitoh and H. Imai, *Multiple unidirectional byte error-correcting codes*, IEEE Trans. Inform. Theory, vol. 37, pp. 903–908, May 1991.

[SPE28] E. Sperner, *Ein satz über untermenge einer endlichen menge*, Math. Zhais., vol. 27, pp. 544–548, 1928.

[STO91] H. S. Stone and J. Cocke, *Computer architecture in the 1990s*, Computer, pp. 30–38, Sept. 1991.

[TAB90] J. Tabor, *Noise reduction using low weight and constant weight coding techniques*, Technical Report AI-TR 1232, MIT Artificial Intelligence Laboratory. June 1990.

[TAK76] Y. Takasaki, M. Tanaka, N. Maeda, K. Yamashita and K. Nagano, *Optical pulse formats for fiber optic digital communications*, IEEE Trans. Comm., vol 24, pp. 404–413, April 1976.

[TAL93] L. Tallini, R. M. Capocelli and B. Bose, *Design of some new balanced codes*, Proc. IEEE 1993 Int. Symp. Inform. Theory, page 7, Jan 1993.

[TOH71] Y. Tohma, R. Sakai and R. Ohyama, *Realization of fail-safe sequential machines by using k-out-of-n code*, IEEE Trans. Comput., vol. c-20, pp. 1270–1275, Nov. 1971.

[VER88] T. Verhoeff, *Delay-insensitive codes-an overview*, Distributed Computing, vol. 3, pp. 1–8, 1988.

[WID83] A. X. Widmer and P. A. Franaszek, *A DC-balanced, partitioned-block, 8B/10B transmission code*, IBM J. Res. Develop., vol 27, pp. 440–451, Sept. 1983.