Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

FINAL GRADE PROJECT

TITLE FGP : Characterization of BCH codes and its use in M2M healthcare applications

TITULATION: Telecommunications technical engineering, specializing in Telematics

AUTHOR: Albert Enrich de León

DIRECTOR: Joan Bas

TUTOR: Luis Alonso Zárate

DATE: 25th June, 2014

Title: Characterization of BCH codes and its use in M2M healthcare applications

Author: Albert Enrich de León

Director: Joan Bas

Tutor: Luis Alonso Zárate

Date: 25th June, 2014

Abstract

Machine-to-Machine (M2M) applications require reliable communications, especially in the medical field, at low-energy consumption to extend the battery life of the sensors. This fact makes BCH codes an interesting option for IEEE 802.15.6 communications since their algebraic nature permit them to obtain the positions of the erroneous bits with mathematical equations of reduced complexity. Moreover, the decoder can be enhanced to save energy through several ways that are exposed in the project.

M2M networks support a large number of interconnected devices. Given that the access to the gateways is through slotted protocols the increase in the demand lead to a reduction in the time-slot. Therefore is necessary to guarantee the reliability of the transmitted information as fast as possible. In addition medical applications are delay non-tolerant since a delay in the decisions could cause fatal consequences.

In this project BCH encoding and decoding algorithms will be implemented in a motta to determine the reduction of time that provides each improvement in the algorithm.

Título: Caracterización de códigos BCH y su desarrollo en aplicaciones médicas M2M.

Autor: Albert Enrich de León

Directores: Joan Bas

Tutor: Luis Alonso Zárate

Data: 25 de junio de 2014

Resumen

Las aplicaciones Machine-to-Machine (M2M) exigen comunicaciones seguras, especialmente en las médicas, y un bajo consumo de energía para alargar la vida de las baterías. Esto hace que los códigos BCH sean una buena opción para las comunicaciones IEEE 802.15.6 ya que su naturaleza algebraica les permite obtener las posiciones de los bits erróneos a partir de ecuaciones matemáticas poco complejas. Además, el decodificador puede ser mejorado para ahorrar energía de varias maneras las cuales se exponen en el proyecto.

Las redes M2M soportan un gran número de dispositivos interconectados. Dado que el acceso a las puertas de enlace se realiza mediante protocolos ranurados, el incremento de peticiones de acceso conlleva a la reducción del tiempo de ranura. Por tanto es necesario garantizar la validez de la información transmitida tan rápido como sea posible. Además, las aplicaciones médicas no son tolerantes a los retrasos ya que en tal caso las consecuencias podrían ser fatales.

En este proyecto los algoritmos de codificación y decodificación BCH serán implementados en motas para determinar la reducción de tiempo que aporta cada mejora en el algoritmo.

INDEX

1 Introduction

1.1. Motivation

M2M communications are characterized by requiring a large throughput, and reliable data at very low energy consumption [1]. In this scenario BCH and Convolutional codes could be perfectly used in M2M networks since they have efficient encoding and decoding algorithms. However, BCH codes can outperform the most energy efficient convolutional code by almost 15% due to the lower number of parity bits required (in average) [2]. In addition, in the next years communication systems will reduce the energy consumption and reduce its area. So, the necessity of more energy efficient communication systems without losing reliability will be a must.

For that reason short-range communications systems like Bluetooth and IEEE 802.15.6 resort to BCH codes to assess the reliability of their transmissions. In particular, this project is focused on improving the decoding process of BCH codes of IEEE 802.15.6 communications. This code will be used to benchmark the speed-up and WER of the different low-complexity BCH decoders that will be proposed based on hard and soft decisions. The soft BCH decoder is the main goal of this project. However, it is based on the hard-BCH decoder. Accordingly, a fast BCH soft decoder means to obtain a fast BCH hard decoder too. For that reason, we will start in the first chapter explaining how to build a hard-BCH decoder and next how to use it in a soft BCH decoder.

In order to speed up the decoding process of BCH codes, it is possible to take advantage of the algebraic nature of BCH codes to avoid iterative decoding. Specifically, it is possible to resort to mathematical equations of low complexity to determine the exact position of the erroneous bits of the codeword. This property has been applied in this project to obtain a low-complexity closed solution when there are two errors in the codeword. Simulation results show that the closed solution for the IEEE 802.15.6 code is 100 faster than the conventional decoder.

This project also evaluates the performance of selecting the correctness of the decoded frames by means of the use of CRC and Syndromes. The use of CRCs does not require Galois Field operations but implies a rate reduction compared to the Syndromes.

1.2. Objectives

The objectives for this final grade project are:

1) Implement BCH coders and decoders.

2) Obtain efficient soft-BCH decoders, improving the data reliability and decoding speed.

3) To obtain results, in terms of delay and word error rate, and compare the performance between hard-BCH decoders and efficient soft-BCH decoders.

4) Present all the pros and cons that the soft-BCH decoding has.

1.3. Document structure

The rest of this document is organized as follows. Section 2 describes the BCH coding structure and how it is implemented in the IEEE 802.15.6 standard. Section 3 describes the BCH decoding stages and how to improve them. Section 4 shows the results from the improved decoder. Section 5 presents the main conclusions of this work.

2 BCH Encoding

2.1. Channel encoding

The basic structure of a communications system is presented in this diagram.

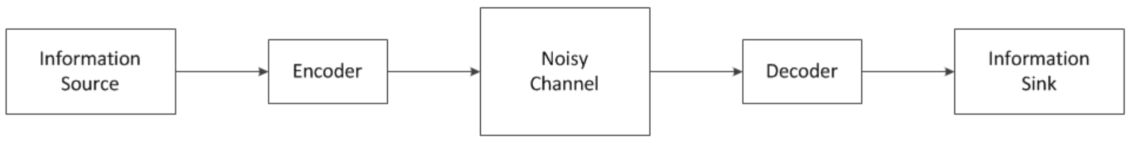

Fig 2.1 Structure of a communication System.

The purpose of the communication systems is to convey the information from one point to another with no degradation.

The communicationchannel adds noise to the information which degrades its reception. In order to overcome/reduce this drawback the transmitter is equipped with an encoder that add some extra bits, so called redundant bits, to protect the transmitted data. In this way, the receiver can detect or even correct the errors in the received codeword.

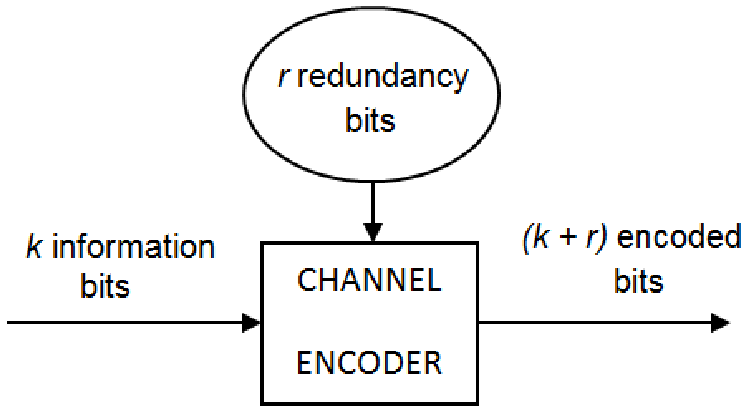

Fig 2.2 Encoder block

The addition of extra-bits to the message permits to define an information rate when a coding scheme is used which has the following expression:

$$\rho = \frac{k}{n} \qquad \textbf{(2.1)}$$

Being k and n the number of bits of the message and the codeword length respectively. On the other hand ρ is the code rate of the information due to the presence of a coding scheme.

2.2. Binary BCH codes

Binary BCH codes are identified by their codeword length *n*, their message length *k*, the maximum error capability of the code is *t*, and are represented as *BCH (n, k, t)*.

BCH codes belong to the class of linear cyclic algebraic codes and are defined over a Galois Field (GF) of *q* elements *GF(q)*, with *q=2^m^*. The parameter *m* corresponds to the degree of the GF, *q* is the number of states that takes each component of the GF elements, and they are related with the codeword length as *n=2^m^-1*. The *r-th* GF element is expressed as α^r, being its vector format equal to $\alpha^r = [\alpha_0{}^r \cdots \alpha^r{}_{m-1}]^T$ taking the *j-th* component the values $\alpha_j{}^r \in \{0,1,\cdots,m-1\}$.

This project is focused in the IEEE 802.15.6 standard, which uses the binary BCH code BCH (*n=63, k=51, t=2)*. So, the codeword spans n=63 bits, the message to transmit has a length of k=51 bits and the maximum error correction capability of the code is of t=2 bits.

To generate all the field elements a primitive polynomial in GF(64) is needed. In this case the primitive polynomial is

$$p(x) = 1 + x + x^6 \qquad \textbf{(2.2)}$$

The initial elements are 0, α^0, and α. Raising α to powers successively identifies α^2, α^3, α^4 and α^5 as members of the extension. When we get to α^6, we realize that $p(\alpha) = 0 = 1 + x + x^6$, so $\alpha^6 = 1 + \alpha$. Thus, we reduce each power greater than five using the identity $\alpha^6 = 1 + \alpha$.

GF(2^6^) with $p(\alpha) = 1 + \alpha + \alpha^6 = 0$		
0	0	(000000)
α^0	1	(100000)
α^1	α	(010000)
α^2	α^2	(001000)
α^3	α^3	(000100)
α^4	α^4	(000010)
α^5	α^5	(000001)
α^6	1 + α	(110000)
α^7	α + α^2	(011000)
α^8	α^2 + α^3	(001100)
α^9	α^3 + α^4	(000110)
α^{10}	α^4 + α^5	(000011)
α^{11}	1 + α + α^5	(110001)
α^{12}	1 + α^2	(101000)
α^{13}	α + α^3	(010100)

α^{14}					α^{2}			+	α^{4}			(001010)
α^{15}							α^{3}			+	α^{5}	(000101)
α^{16}	1	+	α					+	α^{4}			(110010)
α^{17}			α	+	α^{2}					+	α^{5}	(011001)
α^{18}	1	+	α	+	α^{2}	+	α^{3}					(111100)
α^{19}			α	+	α^{2}	+	α^{3}	+	α^{4}			(011110)
α^{20}					α^{2}	+	α^{3}	+	α^{4}	+	α^{5}	(001111)
α^{21}	1	+	α			+	α^{3}	+	α^{4}	+	α^{5}	(110111)
α^{22}	1			+	α^{2}			+	α^{4}	+	α^{5}	(101011)
α^{23}	1					+	α^{3}			+	α^{5}	(100101)
α^{24}	1							+	α^{4}			(100010)
α^{25}			α							+	α^{5}	(010001)
α^{26}	1	+	α	+	α^{2}							(111000)
α^{27}			α	+	α^{2}	+	α^{3}					(011100)
α^{28}					α^{2}	+	α^{3}	+	α^{4}			(001110)
α^{29}							α^{3}	+	α^{4}	+	α^{5}	(000111)
α^{30}	1	+	α					+	α^{4}	+	α^{5}	(110011)
α^{31}	1			+	α^{2}					+	α^{5}	(101001)
α^{32}	1					+	α^{3}					(100100)
α^{33}			α					+	α^{4}			(010010)
α^{34}					α^{2}					+	α^{5}	(001001)
α^{35}	1	+	α			+	α^{3}					(110100)
α^{36}			α	+	α^{2}			+	α^{4}			(011010)
α^{37}					α^{2}	+	α^{3}			+	α^{5}	(001101)
α^{38}	1	+	α			+	α^{3}	+	α^{4}			(110011)
α^{39}			α	+	α^{2}			+	α^{4}	+	α^{5}	(110100)
α^{40}	1	+	α	+	α^{2}	+	α^{3}			+	α^{5}	(111101)
α^{41}	1			+	α^{2}	+	α^{3}	+	α^{4}			(101110)
α^{42}			α			+	α^{3}	+	α^{4}	+	α^{5}	(010111)
α^{43}	1	+	α	+	α^{2}			+	α^{4}	+	α^{5}	(111011)
α^{44}	1			+	α^{2}	+	α^{3}			+	α^{5}	(101101)
α^{45}	1					+	α^{3}	+	α^{4}			(100110)
α^{46}			α					+	α^{4}	+	α^{5}	(010011)
α^{47}	1	+	α	+	α^{2}					+	α^{5}	(111001)
α^{48}	1			+	α^{2}	+	α^{3}					(101100)
α^{49}			α			+	α^{3}	+	α^{4}			(010110)
α^{50}					α^{2}			+	α^{4}	+	α^{5}	(001011)
α^{51}	1	+	α			+	α^{3}			+	α^{5}	(110101)
α^{52}	1			+	α^{2}			+	α^{4}			(101010)
α^{53}			α			+	α^{3}			+	α^{5}	(010101)
α^{54}	1	+	α	+	α^{2}			+	α^{4}			(111010)
α^{55}			α	+	α^{2}	+	α^{3}			+	α^{5}	(011101)

α^{56}	$1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4$	(111110)
α^{57}	$\alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5$	(011111)
α^{58}	$1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5$	(111111)
α^{59}	$1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5$	(101111)
α^{60}	$1 + \alpha^3 + \alpha^4 + \alpha^5$	(100111)
α^{61}	$1 + \alpha^4 + \alpha^5$	(100011)
α^{62}	$1 + \alpha^5$	(100001)
$\alpha^{63} = 1$		

Table 2.1 Galois Field *GF(2^6)* with p(α) = 1 + α + α^6

The table 2.1 defines the words of the code, BCH codes are cyclic since a shift of a word generates another word of the alphabet. In particular the sum of two Galois Field (GF) elements is determined by this table. In example the sum of the elements α^{10} and α^{14} will be:

$$\begin{aligned} &\alpha^{10} = (000011) \\ &\alpha^{14} = (001010) \\ &= (001001) \Rightarrow \alpha^{34} = \alpha^{10} + \alpha^{14} \end{aligned} \qquad \textbf{(2.3)}$$

The computation of the binary elements can be performed easily by using the logical XOR. As seen the result of the sum is α^{34} since the exclusive OR of the binary values α^{10} and α^{14} corresponds to the α^{34} binary value. Therefore, the sum of two GFelements does not match with the sum of their indexes (10+14=24). Then we need the table 2.1 defined by the primitive polynomial to be able to compute the sum of two GF elements. After obtaing the expressions of the elements that conform GF(2^6) it is necessary to explain how to obtain the generator polynomial of the BCH code for WBAN applications, the BCH (63, 51, 2) [3].

2.3. Generator polynomial of the BCH (63, 51, 2) code

In order to obtain the generator polynomial of the BCH code we need and auxiliary polynomial called primitive polynomial. The generator polynomial is the polynomial of lowest degree over GF(2) with α , α^2, α^3, …, α^{2t} as roots.
Let m$_i$(x) be the minimal polynomial of α^i. Then, $g(x)$ must be the *least common multiple (LCM)* of $m_1(x)$, $m_2(x)$, …, $m_{2t}(x)$, that is,

$$g(x) = LCM\{m_1(x), m_2(x), \cdots, m_{2t}(x)\}. \qquad \textbf{(2.4)}$$

A simplification is possible because every even power of a primitive element has the same minimal polynomial as some odd power of the element, halving the number of factors in the polynomial. Then

$$g(x) = LCM\{m_1(x), m_3(x), \cdots, m_{2t-1}(x)\}. \qquad \textbf{(2.5)}$$

Hence, every even power of α in the sequence of (2.3) has the same minimal polynomial as some preceding odd power of α in the sequence. As a result, the generator polynomial $g(x)$ of the binary *t-error-correcting* BCH code of length $2^m - 1$ given by (2.4) can be reduced to

$$g(x) = LCM\{\phi_1(x), \phi_3(x), \cdots, \phi_{2t-1}(x)\}. \qquad \textbf{(2.6)}$$

The even powers minimal polynomials are duplicates of odd powers minimal polynomials, so we only use the first two minimal polynomials corresponding to odd powers of the primitive element.

We need first a primitive element. Well, α is a primitive element in GF(64). Next we need the minimal polynomials of the first two odd powers of α.

Elements	Minimal polynomials
$\alpha, \alpha^2, \alpha^4, \alpha^{16}, \alpha^{32}$	$1 + X + X^6$
$\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24}, \alpha^{48}, \alpha^{33}$	$1 + X + X^2 + X^4 + X^6$
$\alpha^5, \alpha^{10}, \alpha^{20}, \alpha^{40}, \alpha^{17}, \alpha^{34}$	$1 + X + X^2 + X^5 + X^6$
$\alpha^7, \alpha^{14}, \alpha^{28}, \alpha^{56}, \alpha^{49}, \alpha^{35}$	$1 + X^3 + X^6$
$\alpha^9, \alpha^{18}, \alpha^{36}$	$1 + X^2 + X^3$
$\alpha^{11}, \alpha^{22}, \alpha^{44}, \alpha^{25}, \alpha^{50}, \alpha^{37}$	$1 + X^2 + X^3 + X^5 + X^6$
$\alpha^{13}, \alpha^{26}, \alpha^{52}, \alpha^{41}, \alpha^{19}, \alpha^{38}$	$1 + X + X^3 + X^4 + X^6$
$\alpha^{15}, \alpha^{30}, \alpha^{60}, \alpha^{57}, \alpha^{51}, \alpha^{39}$	$1 + X^2 + X^4 + X^5 + X^6$
α^{21}, α^{42}	$1 + X + X^2$
$\alpha^{23}, \alpha^{46}, \alpha^{29}, \alpha^{58}, \alpha^{53}, \alpha^{43}$	$1 + X + X^4 + X^5 + X^6$
$\alpha^{27}, \alpha^{54}, \alpha^{45}$	$1 + X + X^3$
$\alpha^{36}, \alpha^{62}, \alpha^{61}, \alpha^{59}, \alpha^{55}, \alpha^{47}$	$1 + X^5 + X^6$

Table 2.2 Minimal polynomials of the elements

The first two odd power of α minimal polynomials are:

$$\begin{aligned} \alpha: &\quad m_1(x) = x^6 + x + 1 \\ \alpha^3: &\quad m_3(x) = x^6 + x^4 + x^2 + x + 1 \end{aligned}$$

Therefore, $g(x) = LCM\{m_1(x), m_3(x)\} = m_1(x) \cdot m_3(x)$ (since these are irreducible).

So $g(x) = (x^6 + x + 1)(x^6 + x^4 + x^2 + 1) = x^{12} + x^{10} + x^8 + x^5 + x^4 + x^3 + 1.$

The parity bits are determined by computing the remainder polynomial $r(x)$ [3].

In this project we focus in the BCH code *(n=63, k=51, t=2)*. That means that the codewords have a length of 63 bits composed of 51 information bits and 12 parity bits.

To encode a block of bits firstly we append a number of zeros equal to the degree of the generator polynomial to our message $k(x)$. This is the same as multiplying $k(x)$ by x^{12}. Next we divide by the generator polynomial using binary arithmetic.

$$r(x) = \sum_{i=0}^{(n-k-1)} r_i x^i = x^{(n-k)} k(x) \bmod g(x)$$
$$r(x) = \sum_{i=0}^{11} r_i x^i = x^{12} k(x) \bmod g(x) \tag{2.7}$$

Where *k(x)* is the message polynomial which it is expessed in a polynomial way as follows:

$$k(x) = \sum_{i=0}^{k-1} k_i x^i$$
$$k(x) = \sum_{i=0}^{50} k_i x^i \tag{2.8}$$

2.4. BCH applications

BCH codes are formed by two families: the binary and non-binary ones. The main difference between them is the number of states that can take each codeword α_i of the code. Specifically, for binary BCH codes the codewords α_i only can take the states 0 and 1 generating a Galois Field of 2^m elements, being m the degree of the GF. On the contrary, the non-binary BCH codes are formed by codewords that can take more than two states. Thus, generate a GF of a^m elements.In particular, the non-binary BCH codes so-called Reed–Solomon codes, are used in applications such as satellite communications, compact disc players, DVDs, disk drives, and two-dimensional bar codes. Moreover Facebook and Google drive use them to protect the stored data.

Also we find BCH codes in the Smart Cities, in fact this project is focused in the Body Area Network communications specified in the IEEE 802.15.6 standard.

2.4.1. IEEE 802.15.6 standard

The IEEE 802.15 Task Group 6 (BAN) has developed a communication standard optimized for low power devices and operation on, in or around the human body (but not limited to humans) to serve a variety of applications including medical, consumer electronics / personal entertainment and others.

This is the standard that we have chosen to analyze the performance of the decoders proposed in this project.

The physical layer is responsible for the following tasks:

- Activation and deactivation of the radio transceiver.
- Clear channel assessment (CCA) within the current channel.
- Data transmission and reception.

There is a method for transforming a physical-layer service data unit (PSDU) into a physical-layer protocol data unit (PPDU). During the transmission, the PSDU shall be pre-appended with a physical-layer preamble and a physical-layer header in order to create the PPDU. At the receiver, the physical-layer preamble and physical-layer header serve as aids in the demodulation, decoding and delivery of the PSDU.

2.4.1.1. Physical-layer protocol data unit (PPDU)

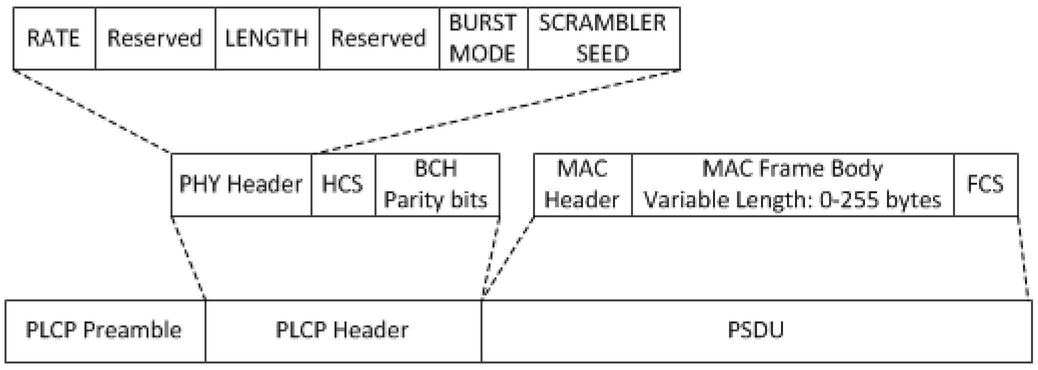

Fig 2.3 Standard PPDU structure

Figure 2.3 shows the format for the physical-layer protocol data unit (PPDU), which is composed of three main components: the physical layer convergence protocol (PLCP) preamble, the PLCP header, and the physical-layer service data unit (PSDU). The components are listed in the order of transmission. The PLCP preamble is the first component of the PPDU. The purpose of the preamble is to aid the receiver during timing synchronization and carrier-offset recovery.

The PLCP header is the second main component of the PPDU (see Fig 2.4). The purpose of this component is to convey the necessary information about the PHY parameters to aid in the decoding of the PSDU at the receiver. The PLCP header can be further decomposed into a RATE field, a LENGTH field, a

BURST MODE field, a SCRAMBLER SEED field, reserved bits, a header check sequence (HCS), and BCH parity bits. The BCH parity bits are added in order to improve the robustness of the PLCP header. The PLCP header shall be transmitted using the given header data rate in the operating frequency band.

The PSDU is the last component of the PPDU. This component is formed by concatenating the MAC header with the MAC frame body and frame check sequence (FCS).

2.4.1.2. PHY header

The PHY header contains information about the data rate of the MAC frame body, the length of the MAC frame body (which does not include the MAC header or the FCS) and information about the next packet whether it is being sent in a burst mode.

The PHY header field shall be composed of 15 bits, numbered from 0 to 14 as illustrated in Fig 2.4.

Bits 0–2 shall encode the RATE field, which conveys the information about the type of modulation, the information data rate, the pulse shaping, the coding rate, and the spreading factor used to transmit the PSDU. Bits 4–11 shall encode the LENGTH field, with the LSB being transmitted first.

Bit 13 shall encode whether or not the packet is being transmitted in the burst (streaming) mode. Bit 14 shall encode the scrambler seed. All other bits that are not defined in this clause shall be understood to be reserved for future use and shall be set to zero.

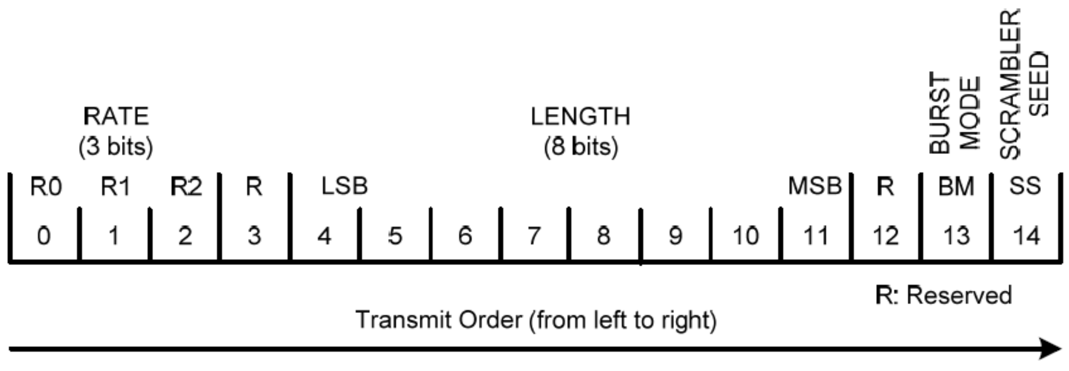

Fig 2.4 PHY header bit assignment

2.4.1.3. PLCP header

A PLCP header shall be added after the PLCP preamble to convey information about the PHY parameters that is needed at the receiver in order to decode the PSDU. The length of the PLCP header is 31 bits, and it shall be constructed as shown:

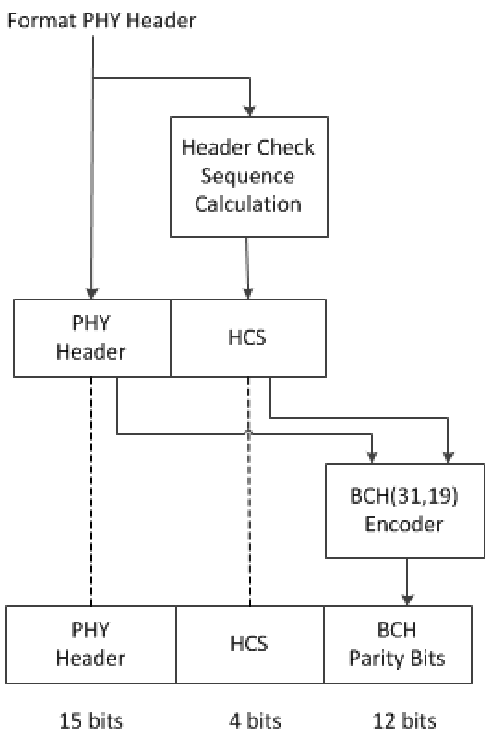

Fig 2.5 BCH encoding scheme for PLCP header construction

2.4.1.4. Header Check Sequence

The PHY header shall be protected with a 4-bit (CRC-4 ITU) header check sequence (HCS). The HCS shall be the ones complement of the remainder generated by the modulo-2 division of the PHY header by the polynomial. The HCS bits shall be processed in the transmit order. An example schematic of the processing order is shown in Fig 2.4. The registers shall be initialized to all ones.

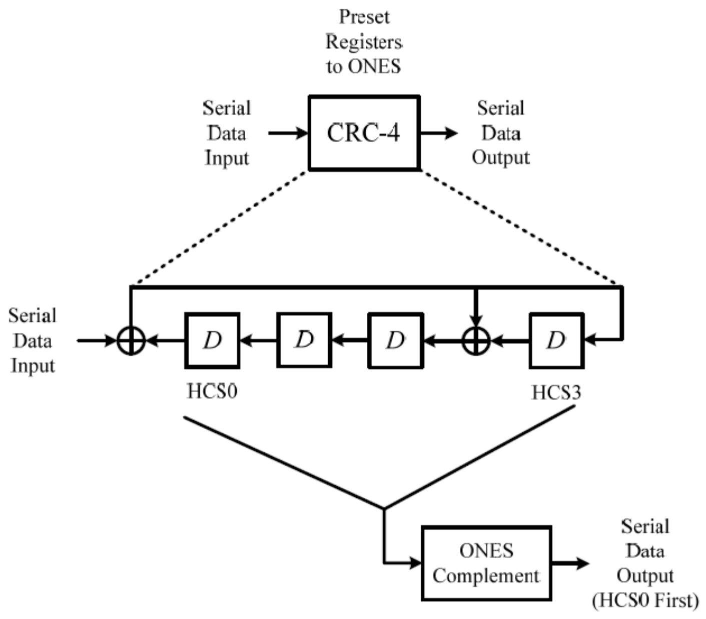

Fig 2.6 BCH encoding scheme for PLCP header construction

2.4.1.5. BCH encoder for PLCP header

The PLCP header shall use a systematic BCH (31, 19, t = 2) code, which is a shortened code derived from a BCH (63, 51, t = 2) code by appending 32 zero (or shortened) bits to the 19 information bits, to improve the robustness of the PLCP header. The shortened bits are removed prior to transmission [10].

3 Decoding

After the encoding process the codewords are modulated, RF converted and transmitted to the receiver. There, the incoming signal for the *q*-th modulated symbol, x_q, is:

$$y_q = \sqrt{\frac{E_b n_b}{N_0}} h_q x_q + w_q \tag{3.1}$$

where E_b / N_0 is the Energy per bit to Noise density ratio, n_b, is the number of bits per modulated symbol, h_q, is a flat fading channel, and w_q represents the Additive White Gaussian Noise (AWGN) signal of variance unity. The received signal is equalized and demapped by means of hard decision since we assume that M2M devices cannot support the complexity of demappers based on soft-decisions [5]. For that reason we have devised low-complexity soft-BCH decoders based on hard-decision BCH decoders of very small computational load. By doing so, we have optimized each one of the decoding stages of the BCH decoder. In particular these stages are: 1) Computation of the Syndromes; 2) Determination of the Error Locator Polynomial (ELP), 3) Solving the Error Locator Polynomial and 4) Bit Flipping.

The first stage of the decoder is to compute the syndromes, this stage is time-consuming since is necessary to determine several syndromes. For that reason some authors propose parallel implementations for computing the Syndromes of BCH codes [3][4]. Nevertheless, parallel architectures are not possible in sensor-based systems due to their limited-signal processing capability. Fortunately, it is possible to overcome this drawback by resorting to Galois Field properties in order to compute only the half of them [5]. If the sum of the syndromes is zero it means that the word does not contain errors otherwise the decoder proceeds to compute the ELP (Error Locator Polynomial) through the Berlekamp-Massey algorithm.

In order to find out the erroneous positions in the word the decoder searches the roots of the ELP using Chien's search method. It can take a long time to find the zeroes of the polynomial because it is a brute-force method. To reduce the time of the Chien's search this project proposes to update the polynomial every time that a solution is found. By doing so, the degree of the polynomial decreases which permits to speed up the BCH decoder. Another improvement is to avoid the Chien's search in the case that there are only two errors in the word, the positions can be directly found through the ELP and resorting to Galois Field properties.

3.1. Classical BCH decoding

This section describes how to decode and correct, if possible, the received codewords,. In particular, the BCH decoder is formed by the next stage: 1) Syndrome Computation, 2) Computation of the Error Locator Polynomial (ELP), 3) Finding the roots of the ELP, 4) Bit Flipping of the erroneous positions and 5) Assessment that the corrected codeword belongs to the codeword set.

3.1.1. Syndrome Computation

If we denote by, $c(x)$ the code polynomial we may define a *t*-error-correcting BCH code of length n = 2^m – 1 in the following manner: a binary *n*-tuple $c=(c_0,c_1,c_2,...,c_{n-1})$ is a codeword if and only if the polynomial $c(x)=c_0+c_1x+...+c_{n-1}x^{n-1}$ has $\alpha,\alpha^2,...,\alpha^{2t}$ as roots. This equality can be written as a product of vectors as follows:

$$(c_0,c_1,...,c_{n-1})\cdot\begin{bmatrix}1\\ \alpha^i\\ \alpha^{2i}\\ \vdots\\ \alpha^{(n-1)i}\end{bmatrix}=0 \qquad \textbf{(3.2)}$$

for $1\le i\le 2t$. The condition given by (3.2) simply says that the inner product of $(c_0,c_1,c_2,...,c_{n-1})$ and $(1,\alpha^i,\alpha^{2i},...,\alpha^{(n-1)i})$ is equal to zero. Now we form the following matrix:

$$H=\begin{bmatrix}1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1}\\ 1 & (\alpha^2) & (\alpha^2)^2 & (\alpha^2)^3 & \cdots & (\alpha^2)^{n-1}\\ 1 & (\alpha^3) & (\alpha^3)^2 & (\alpha^3)^3 & \cdots & (\alpha^3)^{n-1}\\ \vdots & & & & & \vdots\\ 1 & (\alpha^{2t}) & (\alpha^{2t})^2 & (\alpha^{2t})^3 & \cdots & (\alpha^{2t})^{n-1}\end{bmatrix} \qquad \textbf{(3.3)}$$

It follows from (3.2) that if $c=(c_0,c_1,c_2,...,c_{n-1})$ is a codeword in the *t*-error correcting BCH code, then

$$c\cdot H^T=0. \qquad \textbf{(3.4)}$$

Hence the code is the null space of the matrix H, and H is a parity-check matrix of the code.

As usual, the first step of decoding a code is to compute the syndrome from the received vector $c(x)$. For decoding a *t*-error-correcting primitive BCH code, the syndrome is a *2t*-tuple:

$$S = (S_1, S_2, \cdots S_{2t}) = c \cdot H^T \tag{3.5}$$

The expression of the *i-th* syndrome, *S_i,* is:

$$S_i = T_{GF \to D}\left(\sum_{k=0}^{n-1} \oplus \, \mathrm{c_k} T_{GF} (\alpha^{k \times i}) \right) \tag{3.6}$$

where $\alpha^{k \times i}$ is the $k \times i$-th GF element of the BCH code, c_k is the *k*-th bit/symbol of the estimated codeword whereas T_{GF} and $T_{GF \to D}$ are the functions that convert from the GF to integer and vice-versa, respectively. Given that there is no closed expression for the sum of two GF elements, it is necessary to build two auxiliary tables. The table of the GF elements T_{GF} , and the conversion table from GF elements to their power, $T_{GF \to D}$. Table 2.1 represents the polynomial expression of the GF elements for GF(64) expressed in a binary format, i.e., 6-tuple way (T_{GF}). However, the decimal value of T_{GF} does not correspond with the power index of the GF element. For this reason is necessary to build a table that associates the decimal magnitude of each GF element with its GF power index. That is the goal of $T_{GF \to D}$.

Because $\alpha, \alpha^2, \cdots, \alpha^{2t}$ are roots of each code polynomial, $c(\alpha^i) = 0$ for $1 \le i \le 2t$. When a syndrome is not zero we get the relationship between the syndrome and the error pattern:

$$S_i = e(\alpha^i) \tag{3.7}$$

For $1 \le i \le 2t$. From (3.7) we see that the syndrome *S* depends on the error pattern e only. Suppose that the error pattern *e*(x) has v errors at locations $x^{j1}, x^{j2}, \cdots, x^{jv}$; that is,

$$e(x) = x^{j1}, x^{j2}, \cdots, x^{jv}, \tag{3.8}$$

where $0 \le j_1 < j_2 < \cdots < j_v < n$.

3.1.2. Computation of the Error Locator Polynomial

The Error Locator Polynomial (ELP) can be determined by multiple methods: Peterson-Welch method, Berlekamp-Massey Algorithm, Euclidean Algorithm, etc. However, in this project the Berlekamp-Massey Algorithm is used.

Massey showed that to find the ELP $\sigma(X)$ is necessary to make equivalence with a shift-register synthesis problem. Namely, given a sequence of syndromes $S_1, S_2, \cdots, S_{2t}$ find the minimum-degree polynomial $\sigma_1, \sigma_2, \cdots, \sigma_v$ that generates $S_{v+1}, \cdots, S_{2t}$ from $S_1, \cdots, S_v$ in a shift-register of v degree.

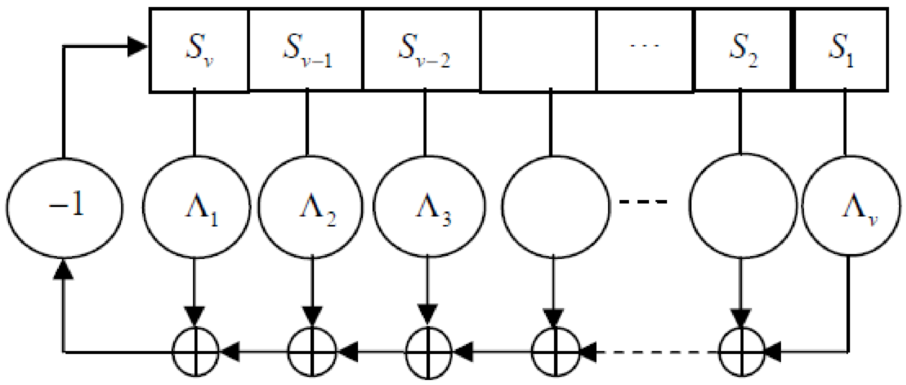

Fig 3.1 ELP from a shift-register view

From (3.7) and (3.8) we obtain the following set of equations:

$$\begin{aligned}
S_1 &= \alpha^{j1} + \alpha^{j2} + \cdots + \alpha^{jv} \\
S_2 &= (\alpha^{j1})^2 + (\alpha^{j2})^2 + \cdots + (\alpha^{jv})^2 \\
S_3 &= (\alpha^{j1})^3 + (\alpha^{j2})^3 + \cdots + (\alpha^{jv})^3 \\
&\vdots \\
S_{2t} &= (\alpha^{j1})^{2t} + (\alpha^{j2})^{2t} + \cdots + (\alpha^{jv})^{2t},
\end{aligned} \tag{3.9}$$

where $\alpha^{j1}, \alpha^{j2}, \cdots, \alpha^{jv}$ are unknown. *Any method for solving these equations is a decoding algorithm for the BCH codes.* Once we have found $\alpha^{j1}, \alpha^{j2}, \cdots, \alpha^{jv}$, the powers $j_1, j_2, \cdots, j_v$ tell us the error locations in $e(x)$. In general, the equations of (3.9) have many possible solutions(2^k of them). Each solution yields a different error pattern. If the number of errors in the actual error pattern $e(x)$ is *t* or fewer (i.e., $v \le t$), the solution that yields an error pattern with the *smallest number of errors* is the right solution; that is, the error pattern corresponding to this solution is the most probable error pattern $e(x)$ caused by channel noise. For large *t*, solving the equations of (3.9) directly is difficult and ineffective. In the following, we describe an effective procedure for determining α^{jl} for $l = 1, 2, \cdots, v$ from the syndrome components S_i's. For convenience, let

$$\beta_l = \alpha^{jl} \tag{3.10}$$

for $1 \le l \le v$. We call these elements *error location numbers*, since they tell us locations of errors.

Now we define the following polynomial:

$$\begin{aligned}
\sigma(X) &= (1+\beta_1 X)(1+\beta_2 X)\cdots(1+\beta_v X) \\
\sigma(X) &= \sigma_0 + \sigma_1 X + \sigma_2 X^2 + \cdots + \sigma_v X^v.
\end{aligned} \tag{3.11}$$

The roots of $\sigma(X)$ are $\beta_1^{-1}, \beta_2^{-1}, \cdots, \beta_v^{-1}$, which are the inverses of the error-location numbers. For this reason $\sigma(X)$ is called the *error-location polynomial*. Note that $\sigma(X)$ is an unknown polynomial whose coefficients must be determined. The coefficients of $\sigma(X)$ and the error-location numbers are related by the following equations:

$$\begin{aligned}
&\sigma_0 = 1 \\
&\sigma_1 = \beta_1 + \beta_2 + \cdots + \beta_v \\
&\sigma_2 = \beta_1\beta_2 + \beta_2\beta_3 + \cdots + \beta_{v-1}\beta_v \\
&\vdots \\
&\sigma_v = \beta_1\beta_2 \cdots \beta_v .
\end{aligned} \tag{3.12}$$

The σ_i's are known as *elementary symmetric functions of* β_l's. From (3.9) and (3.12), we see that the σ_i's are related to the syndrome components S_j's. In fact, they are related to the syndrome components by the following *Newton's identities*:

$$\begin{aligned}
&S_1 + \sigma_1 = 0 \\
&S_2 + \sigma_1 S_1 + 2\sigma_2 = 0 \\
&S_3 + \sigma_1 S_2 + \sigma_2 S_1 + 3\sigma_3 = 0 \\
&\vdots \\
&S_v + \sigma_1 S_{v-1} + \cdots + \sigma_{v-1} S_1 + v\sigma_v = 0 \\
&S_{v+1} + \sigma_1 S_v + \cdots + \sigma_{v-1} S_2 + \sigma_v S_1 = 0 \\
&\vdots
\end{aligned} \tag{3.13}$$

If it is possible to determine the elementary symmetric functions $\sigma_1, \sigma_2, \cdots, \sigma_v$ from the equations of (3.13), the error-location numbers $\beta_1, \beta_2, \cdots, \beta_v$ can be found by determining the roots of the error-location polynomial $\sigma(X)$. Again, the equations of (3.13) may have many solutions: however, we want to find the solution that yields a $\sigma(X)$ of minimal degree. This $\sigma(X)$ will produce an error pattern with a minimum number of errors. If $v \le t$, this $\sigma(X)$ will give the actual error pattern $e(X)$.

3.1.2.1. Berlekamp-Massey Algorithm

Here we present Berlekamp's iterative algorithm for finding the error-location polynomial.

The first step of iteration is to find a minimum-degree polynomial $\sigma^{(1)}(X)$ whose coefficients satisfy the first Newton's identity of (3.13). The next step is to test whether the coefficients of $\sigma^{(1)}(X)$ also satisfy the second Newton's identity of (3.13). If the coefficients of $\sigma^{(1)}(X)$ do satisfy the second Newton's identity of (3.13), we set

$$\sigma^{(2)}(X) = \sigma^{(1)}(X). \tag{3.14}$$

If the coefficients of $\sigma^{(1)}(X)$ do not satisfy the second Newton's identity of (3.13), we add a correction term to $\sigma^{(1)}(X)$ to form $\sigma^{(2)}(X)$ such that $\sigma^{(2)}(X)$ has a minimum degree and its coefficients satisfy the first two Newton's identities of (3.13). Therefore, at the end of the second step of iteration, we obtain a minimum-degree polynomial $\sigma^{(2)}(X)$ whose coefficients satisfy the first two Newton's identities of (3.13). The third step of iteration is to find a minimum-degree polynomial $\sigma^{(3)}(X)$ from $\sigma^{(2)}(X)$ such that the coefficients of $\sigma^{(3)}(X)$ satisfy the first three Newton's identities of (3.13). Again, we test whether the coefficients of $\sigma^{(2)}(X)$ satisfy the third Newton's identity of (3.13). If they do, we set $\sigma^{(3)}(X)=\sigma^{(2)}(X)$. If they not, we add a correction term to $\sigma^{(2)}(X)$ to form $\sigma^{(3)}(X)$. Iteration continues until we obtain $\sigma^{(2t)}(X)$. Then $\sigma^{(2t)}(X)$ is taken to be the error-location polynomial $\sigma(X)$, that is,

$$\sigma(X) = \sigma^{(2t)}(X). \tag{3.15}$$

This $\sigma(X)$ will yield an error pattern $e(X)$ of minimum weight that satisfies the equations of (3.9). If the number of errors in the received polynomial $c(X)$ is t or less, then $\sigma(X)$ produces the true error pattern.

Let

$$\sigma^{\mu}(X) = 1 + \sigma_1{}^{(\mu)}x + \sigma_2{}^{(\mu)}x^2 + \cdots + \sigma_{l\mu}{}^{(\mu)}x^{l\mu} \tag{3.16}$$

be the minimum-degree polynomial determined at the μth step of iteration whose coefficients satisfy the first μ Newton's identities (3.14). To determine $\sigma^{(\mu+1)}(X)$, we compute the following quantity:

$$d_{\mu} = S_{\mu+1} + \sigma_1{}^{(\mu)}S_{\mu} + \sigma_2{}^{(\mu)}S_{\mu-1} + \cdots + \sigma_{l\mu}{}^{(\mu)}S_{\mu+1-l\mu} \tag{3.17}$$

This quantity d_{μ} is called μth *discrepancy*. If $d_{\mu} = 0$, the coefficients of $\sigma^{(\mu)}(X)$ satisfy the $(\mu+1)$th Newton's identity. In this event, we set

$$\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X). \tag{3.18}$$

If $d_{\mu} \neq 0$, the coefficients of $\sigma^{(\mu)}(X)$ do not satisfy the $(\mu+1)$th Newton's identity, and we must add a correction term to $\sigma^{(\mu)}(X)$ to obtain $\sigma^{(\mu+1)}(X)$. To make this correction, we go back to the steps *prior to* the μth step and determine a polynomial $\sigma^{(\rho)}(X)$ such that the ρth discrepancy $d_{\rho} \neq 0$, and $\rho - l\rho$ [$l\rho$ is the degree of $\sigma^{(p)}(X)$] has the largest value. Then,

$$\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_\mu d_\rho^{-1} X^{(\mu-\rho)} \sigma^{(\rho)}(X), \qquad \textbf{(3.19)}$$

which is the minimum-degree polynomial whose coefficients satisfy the first $\mu+1$ Newton's identities.

The described iterative algorithm for finding $\sigma(X)$ applies to both binary and nonbinary BCH codes, in example Reed-Solomon codes; however, for binary BCH codes, this algorithm can be simplified to t-steps for computing $\sigma(X)$.

It is possible to prove that if the first, third, …, (2*t*-1)th Newton's identities hold, then the second, fourth, …, 2*t* th Newton's identities also hold. This implies that with the iterative algorithm for finding the ELP, the solution $\sigma^{(2\mu-1)}(X)$ at the $(2\mu-1)$th step of iteration is also the solution $\sigma^{(2\mu)}(X)$ at the 2μth step of iteration; that is,

$$\sigma^{(2\mu)}(X) = \sigma^{(2\mu-1)}(X). \qquad \textbf{(3.20)}$$

This suggests that the $(2\mu-1)$th and the 2μth steps of iteration can be combined. As a result, the foregoing iterative algorithm for finding $\sigma(X)$ can be reduced to *t* steps. Only the even steps are needed.

The Berlekamp-Massey algorithm builds the error locator polynomial iteratively. Using the notation of Lin and Costello, a $t+2$ line table may be used to handle the bookkeeping details of the error correction procedure for binary BCH decoding. It is described next.

First, make a table (using BCH(63,51,2) as our example):

μ	$\sigma^{(\mu)}(X)$	d_μ	l_μ	$2\mu - l\mu$
$-\frac{1}{2}$	1	1	0	-1
0	1	S_1	0	0
1				
t=2				

Table 3.1 Berlekamp procedure for finding the error-location polynomial

The BCH decoding algorithm follows.

1. Initialize the table as above. Set $\mu = 0$.

2. If $d_\mu = 0$, then $\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X)$.

3. If $d_\mu \neq 0$, then find a preceding row (row ρ) with the most positive $2\mu - l\mu$ and $d_\rho \neq 0$. Then $\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_\mu d_\rho{}^{-1} X^{2(\mu-\rho)} \sigma^{(\rho)}(X)$. If $\mu = t-1$, terminate the algorithm.

4. $l_{\mu+1} = \deg(\sigma^{(\mu+1)}(X))$.

5. $d_{\mu+1} = S_{2\mu+3} + \sigma_1{}^{(\mu+1)} S_{2\mu+2} + \sigma_2{}^{(\mu+1)} S_{2\mu+1} + \cdots + \sigma_{l\mu+1}{}^{(\mu+1)} S_{2\mu+3-l\mu+1}$. σ_i is the coefficient of the *i*-th term in $\sigma(X)$.

6. Increment μ and repeat from step 2.

For a better understanding of the Berlekamp-Massey algorithm we proceed to show an example. In this example a codeword contains 2 errors in the positions 55 and 12, since our BCH can correct up to two errors we must be able to find a valid ELP.

Firstly we compute the syndromes of the codeword, these are:

$$Sd_1 = \alpha^{51}$$
$$Sd_2 = \alpha^{39}$$
$$Sd_3 = \alpha^{5}$$
$$Sd_4 = \alpha^{15}$$

Using the algorithm, we fill the table.

μ	$\sigma^{(\mu)}(X)$	d_μ	l_μ	$2\mu - l\mu$
$-1/2$	1	1	0	-1
0	1	S_1	0	0
1	$\alpha^{51}X+1$	α^{27}	1	1
t=2	$\alpha^4X^2+\alpha^{51}X+1$	-	-	-

Set $\mu = 0$. We see that $d_\mu \neq 0$, so we chose $\rho = -1/2$, and

$$\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_\mu d_\rho{}^{-1} X^{2(\mu-\rho)} \sigma^{(\rho)}(X) = \sigma^{(0)}(X) + d_0 d_{-1/2}{}^{-1} X^{2(0+1/2)} \sigma^{(-1/2)}(X) =$$
$$= 1 + (\alpha^{51})(1)(X)(1) = \alpha^{51}X + 1.$$

Then, $l_{\mu+1} = \deg(\sigma^{(\mu+1)}(X)) = \deg(\alpha^{51}X + 1) = 1.$

Finally,

$$d_{\mu+1} = S_{2\mu+3} + \sigma_1{}^{(\mu+1)} S_{2\mu+2} + \sigma_2{}^{(\mu+1)} S_{2\mu+1} + \cdots + \sigma_{l\mu+1}{}^{(\mu+1)} S_{2\mu+3-l\mu+1} =$$
$$= S_3 + \sigma_1{}^{(1)} \cdot S_2 = (\alpha^5) + \alpha^{51}(\alpha^{39}) = \alpha^{27}.$$

Set $\mu = 1$. We see that $d_\mu \neq 0$, so we chose $\rho = 0$, and

$$\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_\mu d_\rho^{-1} X^{2(\mu-\rho)} \sigma^{(\rho)}(X) = \sigma^{(1)}(X) + d_1 d_0^{-1} X^{2(1-0)} \sigma^{(0)}(X) =$$
$$= \left(\alpha^{51} X + 1\right)\left(\alpha^{55}\right)\left(\alpha^{51}\right)^{-1} X^2 (1) = (\alpha^{51} X + 1) + \left(\alpha^4\right) X^2 = \alpha^4 X^2 + \alpha^{51} X + 1.$$

The final error locator polynomial is $\sigma^{(\mu)}(X) = \alpha^4 X^2 + \alpha^{51} X + 1.$

After deriving the Error Locator Polynomial (ELP) we have computed the average number of iterations of the Berlekamp-Massey algorithm and its deviation number. Next section plots if with further details.

3.1.2.2. Number of iterations and deviation for the Berlekamp-Massey Algorithm

The expression the number of iterations is:

$$\hat{n}_{it,v} = \frac{\sum_{l=v}^{t} \binom{l-1}{v-1} l}{\binom{t}{v}} = \frac{v}{v+1}(t+1). \qquad \textbf{(3.21)}$$

In case of considering bit error probabilities, the number of iterations is:

$$\overline{n}_{it} = \sum_{l=1}^{t} \hat{n}_{it,l} \binom{n}{l} P_b{}^{l} (1 - P_b)^{n-l}. \qquad \textbf{(3.22)}$$

The expression for deviation in the number of iterations is:

$$\hat{\sigma}_{\hat{n}\,it,v} = \frac{1}{v+1} \sqrt{\frac{v(t+1)(t-v)}{v+2}}. \qquad \textbf{(3.23)}$$

In case of considering bit error probabilities, the deviation in the number of iterations is:

$$\overline{\sigma}_{\bar{n}it} = \sum_{l=1}^{t} \hat{\sigma}_{\hat{n}\,it,l} \binom{n}{l} P_b{}^{l} (1 - P_b)^{n-l}. \qquad \textbf{(3.24)}$$

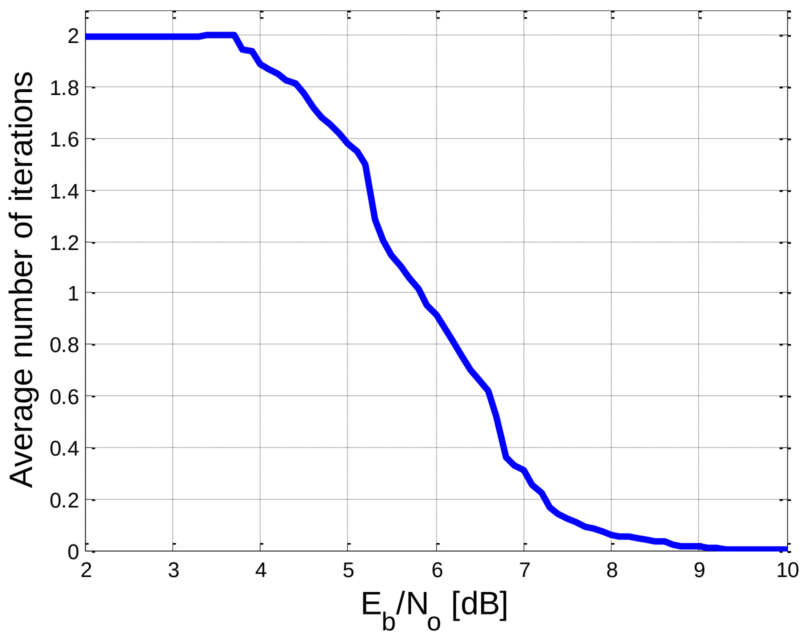

Fig 3.2 Average number of iterations Berlekamp-Massey algorithm

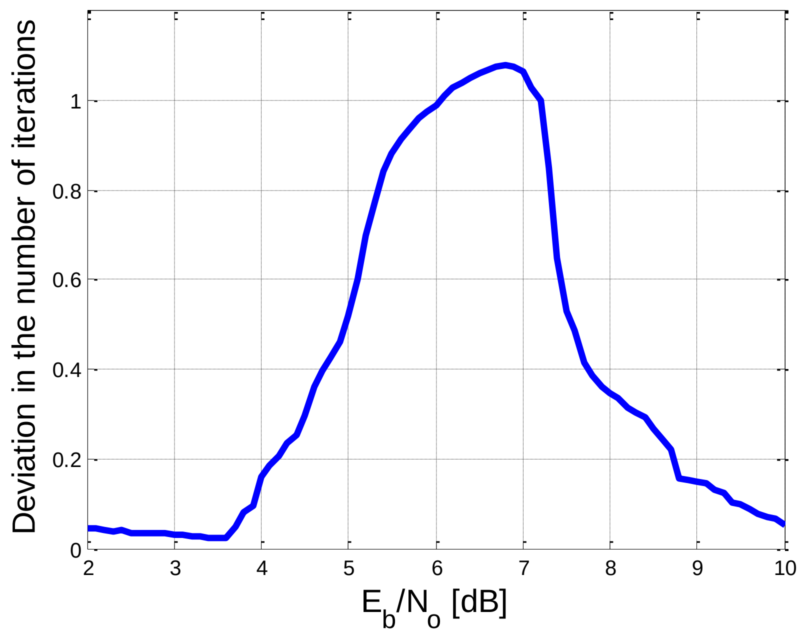

Fig 3.3 Deviation in the number of iterations Berlekamp-Massey algorithm

When the number of errors in the codeword is higher than *t*, the number of iterations of the Berlekamp-Massey algorithm is *t*, since it is not able to determine a polynomial to find the erroneous positions.

3.1.3. Chien's search

The Chien search is an algorithm for determining roots of polynomials defined over a finite field. The most typical use of the Chien search is in finding the roots of error-locator polynomials encountered in decoding Reed-Solomon codes and BCH codes.

It is well known that one of the most time-consuming stages of decoding process of BCH and some other codes is finding roots of the error-locator polynomial. The most widely known root finding algorithm is Chien search method, which is a simple substation of all elements of the field GF(64) into the polynomial, so it has very high time complexity for the case of large fields and polynomials of high degree.

The number of iterations of Chien's search is determined by the furthest position of the k-th erroneous bits. If we consider that the furthest erroneous position is in the bit of the codeword and that there are erroneous bits will be the combinatorial number $\binom{k-1}{v-1}$. If the total number of possibilities for having v errors in n bits is the combinatorial number of $\binom{n}{v}$ then the average number of iterations of the Chien's search when there are erroneous bits, denoted by $\hat{n}_{it,v}$, is

$$\hat{n}_{it,v} = \frac{\sum_{k=v}^{n} \binom{k-1}{v-1} k}{\binom{n}{v}} = \frac{v}{v+1}(n+1). \qquad \textbf{(3.25)}$$

Note that when the number of errors is zero, the number of iterations of Chien's search is also zero (there is no error in the codeword). On the contrary, if the number of errors tends to the codeword length $(v \to n)$, i.e. practically all the bits of the codeword are incorrect, then the average number of errors tends to the codeword length ($\hat{n}_{it,v} \to n$).

Given that the number errors are due to the channel impairments, the actual number of iterations of the Chien's search depends on the bit error probability after the demodulation process, which is denoted by P_b. Therefore, the number of iterations of Chien's method, n_{it}, considering the effect of the demapping process is

$$\bar{n}_{it,v} = \sum_{l=v}^{n} \hat{n}_{it,v} \binom{n}{l} P_b^{\,l} (1-P_b)^{n-l}. \qquad \textbf{(3.26)}$$

The expression for deviation in the number of iterations is:

$$\hat{\sigma}_{\hat{n}it,v} = \frac{1}{v+1}\sqrt{\frac{v(n+1)(n-v)}{v+2}}. \qquad \textbf{(3.27)}$$

In case of considering bit error probabilities, the deviation in the number of iterations is:

$$\overline{\sigma}_{\bar{n}it} = \sum_{l=v}^{n} \hat{\sigma}_{\hat{n}it,v} \binom{n}{l} P_b^{\,l} \left(1-P_b\right)^{n-l}. \qquad \textbf{(3.28)}$$

3.1.3.1. Number of iterations and deviation for the Chien's search

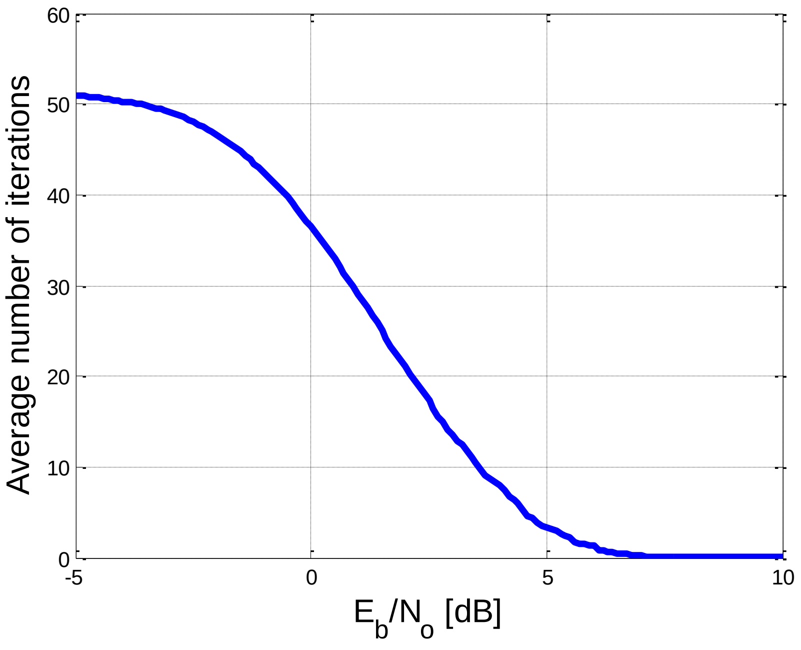

Fig 3.4 Average number of iterations for the Chien's search

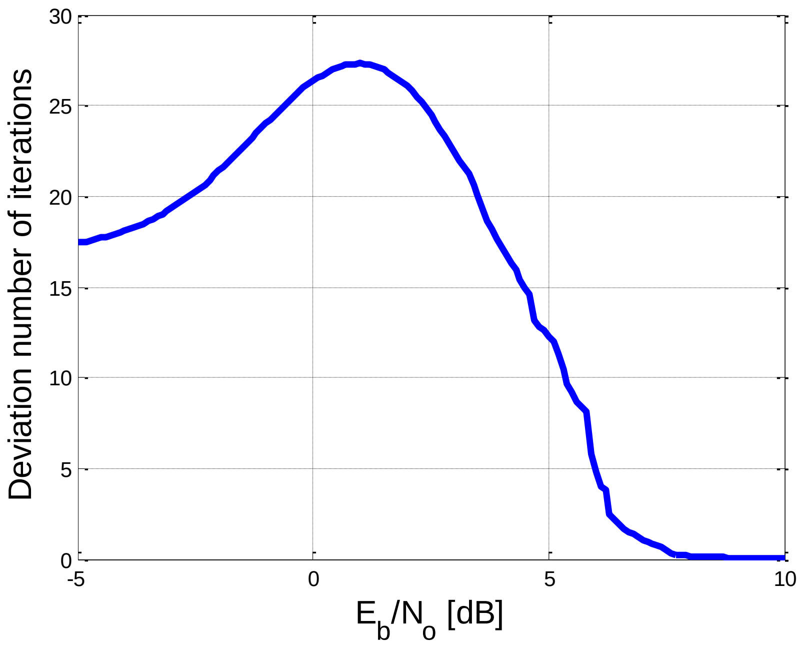

Fig 3.5 Deviation number of iterations for the Chien's search

The minimum distance of a BCH code is $d\min \geq 2t+1$. This means that flipping $2t+1$ bits or more may generate another codeword that belongs to the set of possible codewords, therefore the decoder considers there are no errors.

The reason why the Chien's search algorithm does not try all the 63 elements to solve the ELP is because the noisy channel and the FEC of the decoder flip $2t+1$ bits or more from the original codeword.

3.2. Proposed Low Energy BCH decoding

The decoding process of a channel coding algorithm represents one of the most time consuming parts of the communication receivers. For that reason it is necessary to introduce strategies for reducing its energy consumption. Specially, when parallel techniques cannot be used and very strict energy requirements have to be satisfied. That it is the case of medical applications, which demands very high reliable communication. In particular this work proposes to optimize BCH's decoding stages by resorting to a priori knowledge. This knowledge comes from:

1) The use of GF arithmetic.
2) Data frame information.
3) Cross-layer information and apply them to reduce of each stage.

The next sub-sections details the principles of each optimization.

3.2.1. Syndrome Computation

Binary GF has the nice property that the sum of two GF elements, denoted by x_1 and x_2, raised to a power of two is equal to the raised power two of the sum of the GF elements:

$$y = x_1^{2g} + x_2^{2g} = (x_1 + x_2)^{2g}, \tag{3.29}$$

being 2^g the *g*-th power two, taking g an integer non-negative value. This property is used for speeding up the computation of the syndromes, since it means that it is not necessary to compute all Syndromes:

$$\begin{aligned} S_1 &= \alpha^{j1} + \alpha^{j2} + \cdots + \alpha^{jv} \\ S_2 &= (\alpha^{j1})^2 + (\alpha^{j2})^2 + \cdots + (\alpha^{jv})^2 \\ S_3 &= (\alpha^{j1})^3 + (\alpha^{j2})^3 + \cdots + (\alpha^{jv})^3 \\ &\vdots \\ S_{2t} &= (\alpha^{j1})^{2t} + (\alpha^{j2})^{2t} + \cdots + (\alpha^{jv})^{2t}. \end{aligned} \tag{3.30}$$

If it is applied (3.26) to (3.27), then the even Syndromes can be expressed in terms of the odd ones as:

$$\alpha^{S_z} = \alpha^{zS_{z/2}} \tag{3.31}$$

where in (3.28) α^{S_z} represents the GF value of the *z-th* even Syndrome and its value is equal to the z shifts of the GF degree of the *z/2* odd Syndrome. As a result the even syndromes can be obtained from the odd ones by shifting [5].

3.2.2. Error Locator Polynomial Computation

It is not always necessary to apply the Berlekamp-Massey algorithm for obtaining the ELP. When the number of errors in the codeword is lower than four, it is possible to have closed-expressions of the Error Locator Polynomial of amenable complexity (Table 3.2).

Num. Errors	*Error Locator Polynomial σ(x)*
1	$\sigma(X) = 1 + \alpha^{S_1} X$
2	$\sigma(X) = 1 + \alpha^{S_1} X + \alpha^{d_1 - d_0} X^2$
3	$\sigma(X) = 1 + \alpha^{S_1} X + (\alpha^{d_2 - d_1} \alpha^{d_1 - d_0}) X^2 + \alpha^{d_2 - d_1 + d_0} X^3$

Table 3.2 Expressions of the Error Locator Polynomial σ(X).

It is known from the Berlekamp-Massey algorithm that the discrepancy is:

$$d_{\mu+1} = S_{2\mu+3} + \sigma_1^{(\mu+1)} S_{2\mu+2} + \sigma_2^{(\mu+1)} S_{2\mu+1} + \cdots + \sigma_{l\mu+1}^{(\mu+1)} S_{2\mu+3-l\mu+1} \tag{3.32}$$

Then,

$$\begin{aligned} d_0 &= S_1 \\ d_1 &= S_3 + \sigma_1^{(1)} S_2 = S_3 + S_1 \cdot S_2 \\ d_2 &= S_5 + \sigma_1^{(2)} \cdot S_4 + \sigma_1^{(2)} \cdot S_3 = S_5 + S_1 \cdot S_4 + \alpha^{d_1 - d_0} \cdot S_3 \end{aligned} \tag{3.33}$$

Applying these equations instead of using the iterative Berlekamp-Massey algorithm allows the decoder to obtain the ELP in a faster way.

But the Berlekamp-Massey algorithm also gives us information about the number of errors in the codeword, since it is the degree of the polynomial. So if we want to avoid the Berlekamp-Massey algorithm we need another way to know the number of errors.

Actually there is a way to know if the number of errors is zero, one or more through the Syndromes (Table 3.3):

- Zero errors: If the received codeword does not contain any error the value of Syndromes is zero.

- One error: If there is one error in the codeword the Syndromes follow a specific pattern:

$$\begin{aligned} &S_1 = \alpha^j \\ &S_2 = (\alpha^j)^2 = 2 \cdot S_1 \\ &S_3 = (\alpha^j)^3 = 3 \cdot S_1 \\ &S_4 = (\alpha^j)^4 = 4 \cdot S_1 \\ &\vdots \end{aligned} \tag{3.34}$$

  Being j the erroneous position of the codeword.

- Two or more errors: If the Syndromes do not satisfy any of the above cases there are two or more errors in the codeword.

Num. Errors	*Condition*
0	$S_1 = 0 \quad S_3 = 0$
1	$S_1 \neq 0 \quad S_3 = 3S_1$
2	$S_1 \neq 0 \quad S_3 \neq 3S_1$

Table 3.2 Number of errors in the codeword based on the Syndrome.

Since we are working with a BCH (63, 51, t = 2) code the decoder does not need the Berlekamp-Massey algorithm to compute the ELP, the expressions of the ELP are enough.

3.2.3. Chien’s Search

In order to find out the erroneous positions in the word the decoder searches the roots of the ELP using the Chien’s search method. It can take a long time to find the zeroes of the polynomial because it is a brute-force method. This project shows several ways to reduce the time of the Chien’s search:

1) Knowing the data frame structure is possible to avoid some iterations in the Chien’s search.

2) Knowing the structure of the error-locator polynomial coefficients is possible to update the polynomial every time that a solution is found. By doing so, the degree of the polynomial decreases.

3) Avoid the Chien’s search in the case that there are only two errors in the word.

3.2.3.1. Knowing Data Frame Structure to reduce the load of Chien’s search

If the number of erroneous bits is higher than two, then Chien’s search has to test, in a serial way, the GF elements of the BCH code into the Error Locator Polynomial.

However, the information of the data frame can be used in order to:

1) Improve the error control process.
2) Speed-up the Chien’s search.

Note that if some fields or bits of the received codeword are known then it is possible to pre-fix them before starting the classical BCH decoding process. That assumption is reasonable since some fields of the frames are known due to:

1) They are reserved.
2) Identify the type of frame.
3) They are forbidden.
4) Used for extending the code.

In this way the WER is reduced since the aided information increase the error correction capacity of the code. In addition the number of iterations of the Chien’s search is also reduced since there are some positions in the codeword known at the receiver side.

As example case Fig 2.3 shows the frame structure for the IEEE 802.15.6 Physical Layer Convergence Procedure (PLCP). This frame is protected by BCH(63,51,2) code and has a CRC of 4 bits allocated in the Header Control Sequence (HCS). The message length of the PHY layer is $k = 31$ whereas the codeword length is $n = 63$.

This frame has 2 reserved bits and 32 bits that are zero due to the BCH code is extended from the BCH(31,19,2). So, all these 34 bits correspond to GF positions that are not necessary to test in the Chien's search since they are known at the receiver side. As a result, the introduction of aided information in the BCH's decoder permits to reduce its load if it is compared to the conventional non-data aided BCH decoder. For instance the rate spans three bits but some of its combinations are not used since there are less code rates than possible combinations.

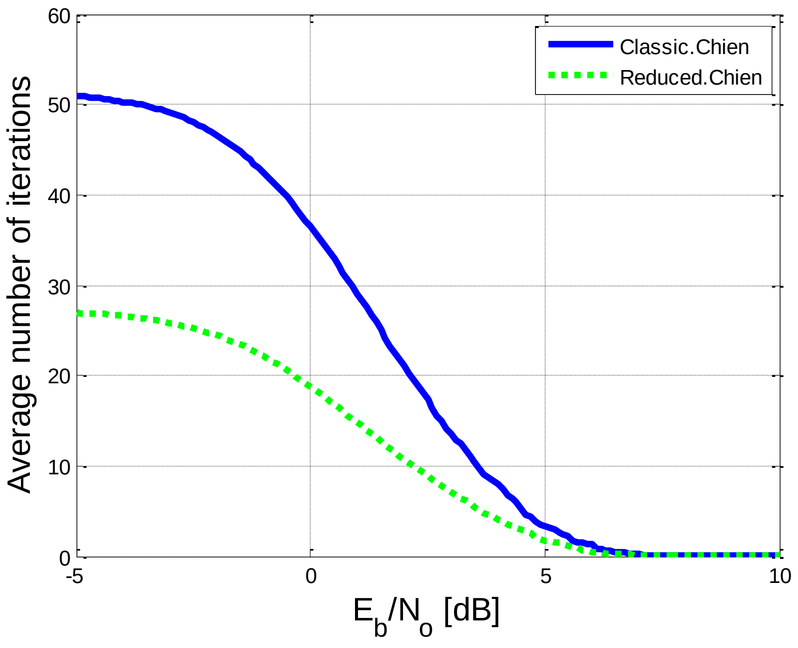

Fig 3.6 Average number of iterations for the Chien's search

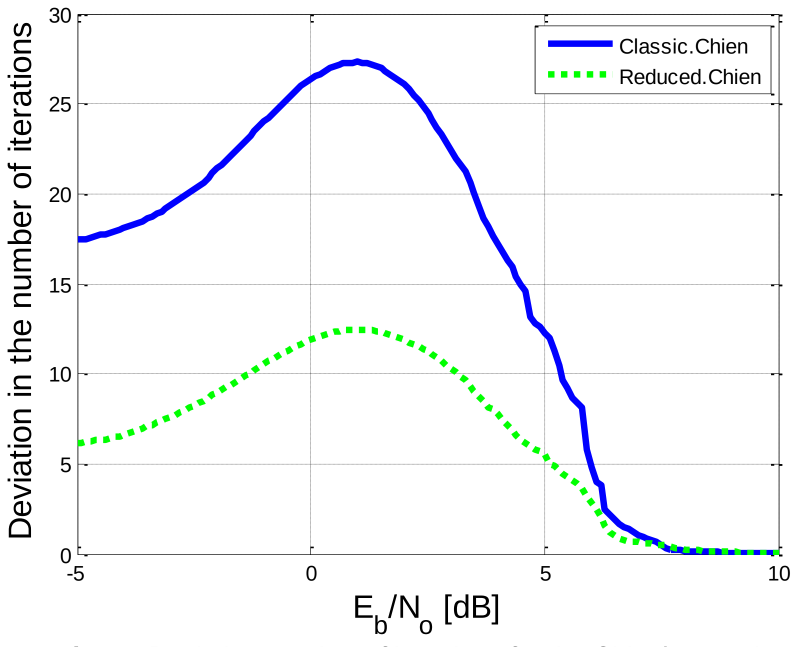

Fig 3.7 Deviation number of iterations for the Chien's search

Due to the reduced Chien algorithm just tests 29 out of 63 elements, the consumed energy in this stage is reduced to more than a half.

3.2.3.2. Structure of the Error Locator Polynomial Coefficients

The Error Locator Polynomial has the next structure when there are v erroneous positions in the codeword.

$$\sigma(X) = 1 + \alpha^{g_{1,v}} X. + \ldots + \alpha^{g_{1v,v}} X^{v}. \tag{3.35}$$

Being X the GF field element to test, whereas $\alpha^{g_{j,v}}$ is the j-th GF coefficient of the ELP when there are v errors in the codeword. The coefficients of the Error Locator Polynomial have a structure that depends on the positions of the v roots. In particular, the general expression for the j-th coefficient of an ELP of degree v is:

$$\alpha^{g_{j,v}} = \sum_{u_1=1}^{v-j} \alpha^{p_{u_1}} \sum_{u_2=u_1+1}^{v-1} \alpha^{p_{u_2}} \cdots \sum_{u_j=u_{j-1}+1}^{v} \alpha^{p_{u_j}}, \tag{3.36}$$

being P_{uj} the u-th erroneous position of the codeword.

Basically, from (3.34) we observe that the j-th GF coefficient of the ELP is formed by the sum of the erroneous positions taken in groups of j elements. Next, if separate the term of an erroneous positions, for instance, the first one p_1, the expression (3.34) results:

$$\begin{aligned} \alpha^{g_{j,v}} = \alpha^{p_1} \sum_{u_2=2}^{v-j} \alpha^{p_{u_2}} \cdots \sum_{u_j=u_{j-1}+1}^{v} \alpha^{p_{u_j}} + \\ \sum_{u_1=2}^{v-j} \alpha^{p_{u_1}} \sum_{u_2=u_1+1}^{v-1} \alpha^{p_{u_2}} \cdots \sum_{u_j=u_{j-1}+1}^{v} \alpha^{p_{u_j}}, \end{aligned} \tag{3.37}$$

However, if we compare (3.33) and (3.32), then we obtain that the two groups of summatories correspond to the values of the *j-1-th* and *j-th* coefficients of an ELP with *v-1* erroneous positions, denoted as $\alpha^{g_{j-1,v}}$ and $\alpha^{g_{j-1,v-1}}$ respectively

$$\begin{aligned} \alpha^{g_{j-1,v-1}} &= \sum_{u_2=2}^{v-j} \alpha^{p_{u_2}} \cdots \sum_{u_j=u_{j-1}+1}^{v} \alpha^{p_{u_j}} \\ \alpha^{g_{j,v-1}} &= \sum_{u_1=2}^{v-j} \alpha^{p_{u_1}} \sum_{u_2=u_1+1}^{v-1} \alpha^{p_{u_2}} \cdots \sum_{u_j=u_{j-1}+1}^{v} \alpha^{p_{u_j}}, \end{aligned} \tag{3.38}$$

In this way if we plug (3.34) into (3.33) then, the recursive relationship of the ELP's GF coefficients is:

$$\alpha^{g_{j,v}} = \alpha^{p_1}\alpha^{g_{j-1,v-1}} + \alpha^{g_{j,v-1}}, 1 \le j \le v \qquad \textbf{(3.39)}$$

This relationship is quite useful in the Chien's search since it permits to updates the ELP polynomial once a root has been estimated. In this case, the value of p_1 would represent the root that has been detected in the Chien's search after testing a GF element. In this way, it is matched the degree of the ELP with the number of roots to search which reduces the load of the Chien's search. In this case there are two techniques for obtaining the immediate ELP polynomial of lower degree. Starting from the lowest degree coefficient from the highest one as follows:

$$\begin{cases} \alpha^{g_{j,v-1}} = \alpha^{p_1} + \alpha^{g_{j,v}}, j = 1 \\ \alpha^{g_{j,v-1}} = \alpha^{g_{j,v}} + \alpha^{p_1}\alpha^{g_{j-1,v-1}}, 2 \le j \le v \end{cases} \qquad \textbf{(3.40)}$$

Or starting by the highest degree coefficient and descending to the lowest one as:

$$\begin{cases} \alpha^{g_{j-1,j-1}} = \alpha^{g_{j,j}-p_1}, j = v \\ \alpha^{g_{j,v-1}} = \alpha^{g_{j,v}} + \alpha^{p_1}\alpha^{g_{j-1,v-1}}, v-1 \ge j \ge 1 \end{cases} \qquad \textbf{(3.41)}$$

Both strategies of updating the ELP polynomial are quite similar since only differ in the initial condition. However, if it is started from the highest GF degree is a bit easier to get the GF degree of the ELP's polynomial.

3.2.3.3. Solving the Error Locator Polynomial when there are two errors

The terms $\alpha^{g_{1,2}}$ and $\alpha^{g_{2,2}}$ represent the coefficients of first and second degree of the ELP. In order to detect an erroneous position is necessary to evaluate the GF value of the possible erroneous position into the ELP. If the value of ELP is null, then the tested GF value corresponds to an erroneous position of the codeword.

$$\sigma(X = \alpha^{-p_1}) = 1 + \alpha^{g_{1,2}}\alpha^{-p_1} + \alpha^{g_{2,2}}\alpha^{-2p_1} = 0. \qquad \textbf{(3.42)}$$

Next, if it is multiplied (3.38) by α^{p_1}:

$$\alpha^{p_1} + \alpha^{g_{2,2}-p_1} = \alpha^{g_{1,2}}. \qquad \textbf{(3.43)}$$

At that point, the strategy is to get the addition of two complementary GF elements in the left side part of (3.39). In order to obtain it, is necessary to raise to power two all the coefficients of (3.39) by taking advantage of GF properties. In this way, after some posterior manipulations with $\alpha^{g_{2,2}}$ is obtained:

$$\alpha^{-g_{2,2}+2p_1} + \alpha^{g_{2,2}-2p_1} = \alpha^{2g_{1,2}-g_{2,2}} \qquad \textbf{(3.44)}$$

However, from (3.13) the expression of $\alpha^{g_{2,2}}$ is equal to $\alpha^{g_{2,2}} = \alpha^{p_1+p_2}$, being p_1 and p_2 the two erroneous positions of the codeword. Next, if the distance between the two erroneous positions is r,

$$p_2 = p_1 + r, \qquad \textbf{(3.45)}$$

and (3.39) is introduced into (3.38), the next expression results:

$$\alpha^{-r} + \alpha^{r} = \alpha^{2g_{1,2}-g_{2,2}} \qquad \textbf{(3.46)}$$

This equation tells us that the key parameter for decoding two errors is the distance between them not their respective positions. Although the terms $\alpha^{g_{1,2}}$ and $\alpha^{g_{2,2}}$ depends on the erroneous positions of the codeword, if their relative distance r is the same, the expression (3.40) produces identical value. Consequently, it is possible to match the value of r with the magnitude $2g_{1,2} - g_{2,2}$ to obtain r. This matching can be done in a Look-Up-Table (LUT) or in logical gates. The number of entries of the LUT is reduced to $(n-1)/2$, since it is possible to apply the symmetry of the two erroneous positions in the GF. Recall that in GF a negative GF degree is converted to a positive one by adding the GF order, $\alpha^{-x} = \alpha^{n-x}$.Note that if the value of $2g_{1,2} - g_{2,2}$ belongs to a forbidden combination, it is possible to conclude that a decoding failure has happened. So, it is not necessary to compute any CRC or Syndrome which speeds up the decoding process. Moreover, take into account that for the case of two errors the use of (3.39) only requires the computation of 2 syndromes instead of *2t* ones required for solving the key-equation of BCH decoder **¡Error! No se encuentra el origen de la referencia.**.

After estimating the distance between the two erroneous positions, the detection of the erroneous positions p_1 is carried out by means of the next expression:

$$\alpha^{-g_{2,2}+2p_1} = \alpha^{-r} \Rightarrow p_1 = \frac{-r + g_{2,2}}{2}, \qquad \textbf{(3.47)}$$

which corresponds to a bit shifting of the distance of r from $g_{2,2}$. So, it is not necessary any division. After computing, p_1 and r the second erroneous positions comes from (3.39). Note that the symmetry properties permit to apply (3.39) and (3.41) with independence of knowing what root is the biggest one. In case that having a number of errors higher than two, then it is applied Chien's search with updating the coefficients of the ELP once the roots of the ELP are detected. Once, the degree of the ELP is equal to two is applied the reduced load strategy for the two erroneous positions. This solution is even valid for ELP of higher degree if they have the next pattern:

$$\Lambda(X) = 1 + \alpha^{g_{1,2\times 2^{2u}}} X^{2^u} + \alpha^{g_{2\times 2^{2u},2\times 2^{2u}}} X^{2\times 2^{2u}}, \qquad \textbf{(3.48)}$$

Following the same example that we used to prove the Berlekamp-Massey algorithm we proceed to prove the two errors algorithm. In example the received codeword contains 2 errors in the positions 55 and 12 and its Error Locator Polynomial is $\sigma(X) = 1 + \alpha^{51}X + \alpha^4 X^2$.

$$\alpha^{-r} + \alpha^{r} = \alpha^{2g_{1,2} - g_{2,2}}$$
$$\alpha^{-r} + \alpha^{r} = \alpha^{2\cdot 51 - 4} = \alpha^{35}$$

Using the LUT we find out that $r = 20$. Then $\alpha^{-g_{2,2}+2p_1} = \alpha^{-r} \Rightarrow p_1 = \dfrac{-r + g_{2,2}}{2}$,

$p_1 = \dfrac{-20+4}{2} = -8 \Rightarrow -8 + 63 = 55.$ Therefore $p_2 = p_1 + r = 55 + 20 = 75 \Rightarrow 75 - 63 = 12.$

Finally, after obtaining Chien's search, comes the correction of the received codeword. Given that the studied BCH code is the binary one, the correction of the received codeword consists on the bit flipping of the erroneous positions obtained in the Chien's search. Nevertheless, it does not mean that the corrected codeword be the true one. There are several techniques to carry out this process. The next section details them.

3.2.4. Checking the Validness of the Corrected Codeword

In order to assess the validness of the corrected codeword two strategies are proposed:

1) Fast Computation of the Syndromes of the corrected codeword.

2) Resort to an additional Cyclic Redundancy Code (CRC) to check if the corrected message codeword is the true one.

3.2.4.1. Fast syndrome update

After the Chien's search, BCH's decoder knows the number of erroneous bits in the codeword, their positions and the syndromes of the received data. Therefore, if it is not necessary to compute the Syndromes since its load depends on the codeword length *n.* The proposed alternative consists on updating the Syndromes of the received data by adding the GF elements of the erroneous positions normalized by the number of Syndrome. Thus, their updating law is:

$$S_{j,new} = S_{j,old} \oplus \sum_{q=1}^{v} \alpha^{j\times p_q} \qquad \textbf{(3.49)}$$

Where $S_{j,new}$ $S_{j,old}$ represent the j-th syndrome after and before the decoding process, v is the number of erroneous bits and p_q is the q-th position of the received codeword with erroneous bit, whereas the symbol $\oplus$ expresses sum in modulo two, i.e. or-exclusive. The syndromes that have to be recomputed are only the odd ones, and if there were one syndrome that after the updating process was different from zero, then the updating process would finish. In this stage, the WER given by Syndrome strategy tends to be:

$$WER = 1 - \sum_{q=0}^{t} \binom{n}{q} p_b^q (1 - p_b)^{n-q} \qquad \textbf{(3.50)}$$

Which it is the BCH bound for hard decoding. The larger is the codeword length the closer is the approximation[5].

3.2.4.2. CRC for determining the validness of the codeword

An additional degree of information comes from the use of a CRC code in the frame structure. Its main its use pursues to determine if the message contained in the codeword is correct. This strategy is still valid although the syndromes are not zero after the decoding process. This strategy does not require carry out any Galois Field operation but reduces the rate. However, this strategy has also an error probability in determining the correct codeword [10]:

$$WER = \sum_{q=t+1}^{n} A_q p_b^{d_q} (1 - p_b)^{n-d_q}, \qquad \textbf{(3.51)}$$

being p_b the erroneous bit probability, d_q is the q-th Hamming distance of the code whereas A_q represents the number of codewords with Hamming distance equal to d_q.

At that point it has been detailed how to design an efficient BCH decoder guided by hard-decisions .The next step uses the additional information to obtain a soft BCH decoder of reduced load. The following section details it.

3.2.5. Soft decoder

In information theory, a soft-decision decoder is a class of algorithm used to decode data that has been encoded with an error correcting code. Whereas a hard-decision decoder operates on data that take on a fixed set of possible values (typically 0 or 1 in a binary code), the inputs to a soft-decision decoder may take on a whole range of values in-between. This extra information indicates the reliability of each input data point, and is used to form better estimates of the original data. Therefore, a soft-decision decoder will typically perform better in the presence of corrupted data than its hard-decision counterpart.

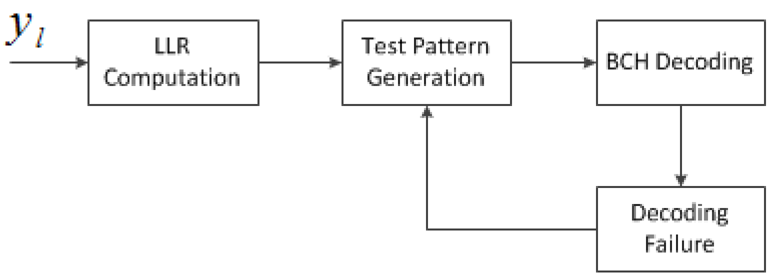

Fig 3.8 Soft-Decision scheme.

Soft BCH decoders are designed in order to i) compute the Log-Likelihood Ratios (LLRs) of the received codeword, ii) determine the p-lowest LLRs, iii) modify the least p-reliable positions of the codeword until a maximum set of 2^p hard-decided codewords and iv) evaluate by each candidate to codeword the BCH hard-decoder. Given that the soft-BCH decoder introduces an iterative mechanism for selecting the candidate codewords, the elimination of the iterations in the BCH-Hard decoder permits to obtain practical soft-BCH decoders. This strategy is faster than the use of a hard BCH code with the same codeword length and higher protection level. Faster in the sense that the increase of the protection level of the soft-BCH does not imply the reduction in the transmission rate of BCH hard with higher protection level and in the number of iterations. If only consider the average number of iterations of Chien's search [8], the trade-off for selecting the number of errors that can correct the soft-decoder will be:

$$2^{n_e}\left(\frac{t_1-1}{t_1}\right)(n_1-1)\leq\left(\frac{t_2-1}{t_2}\right)(n_2-1), \qquad \textbf{(3.52)}$$

being n_e the number of positions that the soft-decoder can consider as no reliable, whereas n_1, t_1 and n_2, t_2 correspond to the codeword length and the maximum error correction capabilities of the BCH code with soft-decisions and hard ones. In our case, t_1 is equal to two and the BCH soft decoder follows the Chase-3 Algorithm with $n_e = t_1$. [9] The expression (3.52) assumes the worst case criterion for designing the soft-decoder since considers that it is necessary to test all combinations of n_e possible error patterns to successfully decode the BCH's codeword [9]. After presenting the hard and soft BCH decoders, next section presents the results.

4 Results

This section shows the results obtained in the decoder about the Word Error Rate and the decoding time.

Firstly we show the WER obtained with the strategies submitted above. Then we show the improvement of the WER when a soft decoder is implemented.

Next we show the results about the decoding time, we compare the time obtained in each stage and the total decoding time between a classic BCH decoder and a fast BCH decoder.

4.1. Word Error Rate

4.1.1. Hard decoder

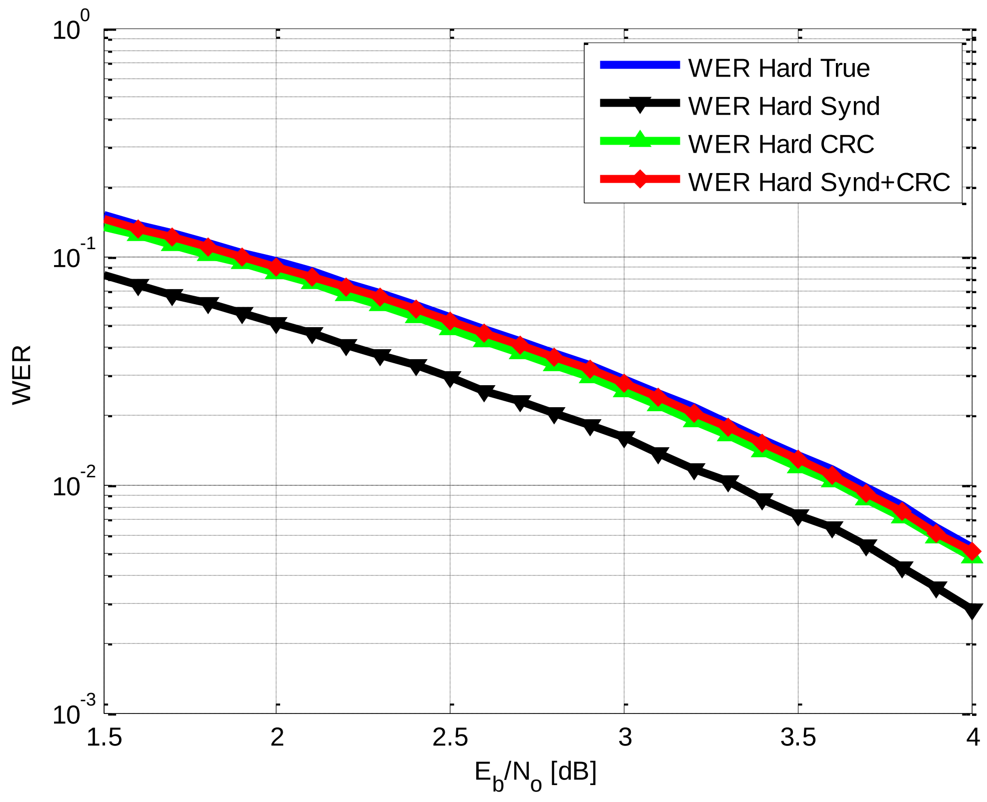

Fig 3.9 WER comparison of BCH(63,51,2) assuming that receiver uses CRC and Syndrome information for estimating the true WER.

This graph shows the WER obtained when we use Syndromes, CRC, and both to check the validness of the codewords.

The Syndromes strategy has a WER away from the True WER because when the decoder uses the FEC find other codewords that does not correspond with the original one.

But the CRC in the message helps the decoder to decide if the final codeword contains the same message than the original codeword.

Using both Syndromes and CRC the decoder is able to obtain a WER similar to the true WER.

4.1.2. Soft decoder

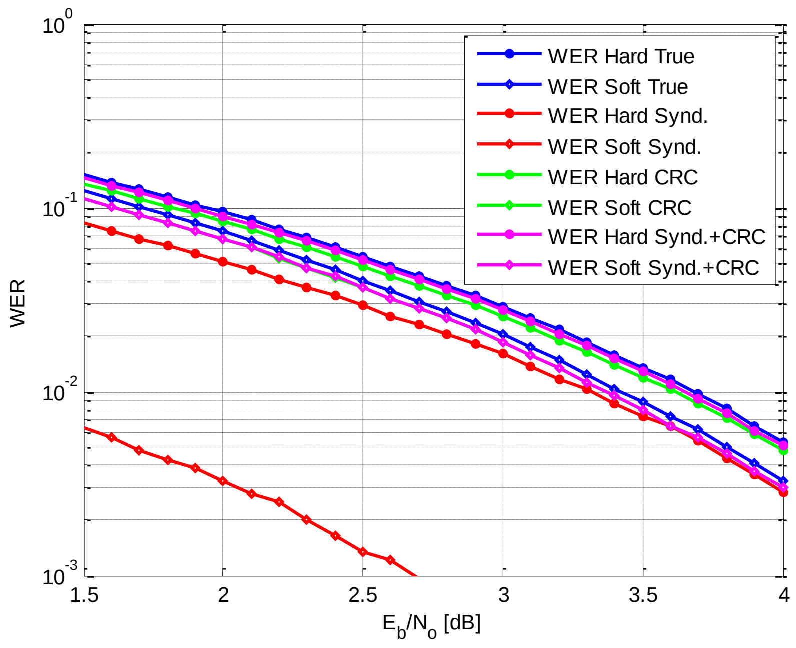

Fig 3.10 WER comparison for Hard and Soft decoding of BCH(63,51,2) assuming that receiver uses CRC and Syndrome information for estimating the true WER.

The soft decoder allows the decoder to improve the true WER, therefore if our BCH code is capable to correct up to t errors, using a soft decoder most of the times the decoder will be able to correct more than t errors.

The WER of the CRC is closer to the true WER than the syndromes WER, this is because the syndromes is some cases consider that the corrected codeword is the original codeword although it has erroneous positions compared to the original. This can happen if the corrected codeword overcomes the minimum distance of the code.

Moreover a soft decoder increases the probability of overcoming the minimum distance of the code because it flips more bits than the hard decoder. So it is easier to obtain a codeword of the code which is different from the original. That is the reason why the WER of the syndromes when using a soft decoder is so low, because sometimes it obtains the original codeword and sometimes generates another one.

The next graph shows how the soft decoder performance depends on the rate of the code.

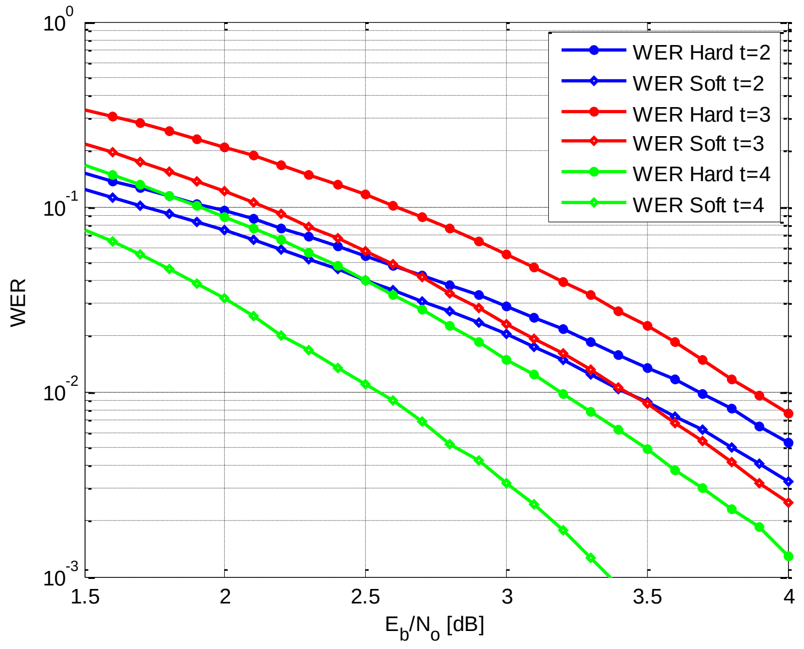

Fig 3.11 WER comparison for Hard and Soft decoding for BCH(63,51,2), BCH(63,45,3) and BCH(63,39,4).

The soft decoder performs better for high values of t, because it is harder to overcome the minimum distance of a codeword.

4.2. Decoding time

4.2.1. Syndromes

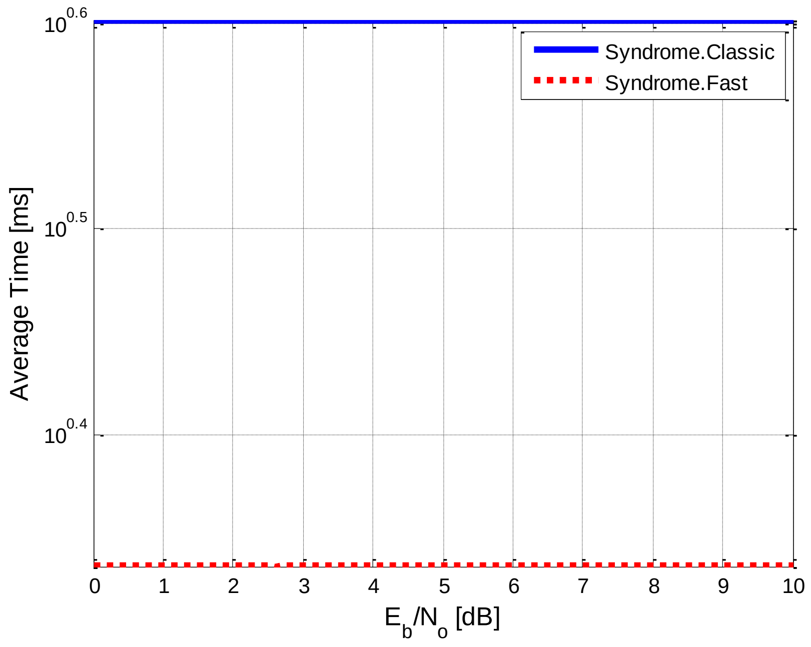

Fig 3.12 Syndromes time comparison between the classic and the fast decoder.

The fast decoder is two times faster in this stage compared to the classic one since only the half of the Syndromes are computed.

4.2.2. Error-Locator polynomial

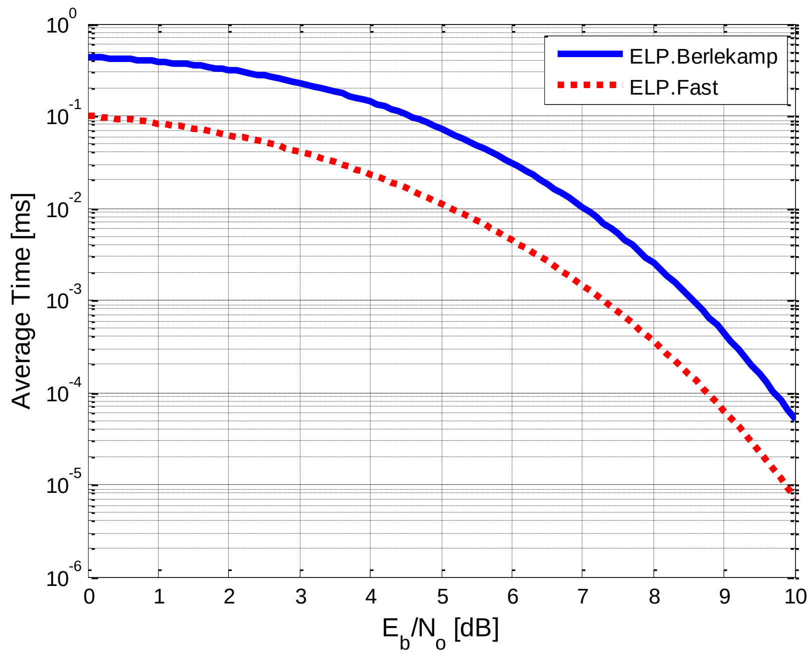

Fig 3.13 ELP time comparison between the classic and the fast decoder.

The fast decoder computes the error-locator polynomial much faster than the classic one because the iterative Berlekamp-Massey algorithm is not needed, unlike the classic decoder.

4.2.3. Chien's search

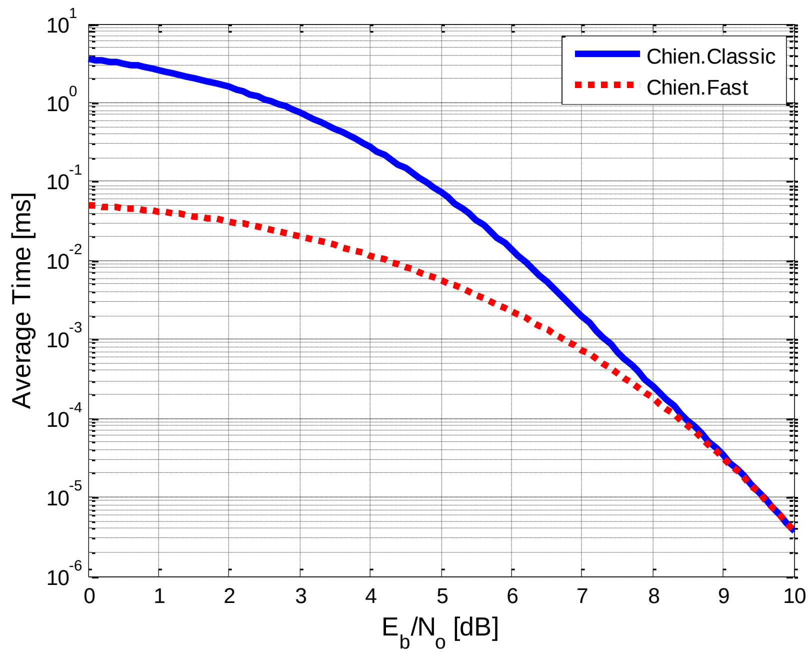

Fig 3.14 Chien's search time comparison between the classic and the fast decode.

The fast decoder does not need to use the Chien's search algorithm to find the zeroes of the error-locator polynomial, the coefficients of the polynomial and a Look-Up Table are enough to find the erroneous positions of the codeword. While the classic decoder searches the roots of the ELP using the Chien's search method. It can take a long time to find the zeroes of the polynomial because it is a brute-force method.

4.2.4. Checking the validness of the corrected codeword

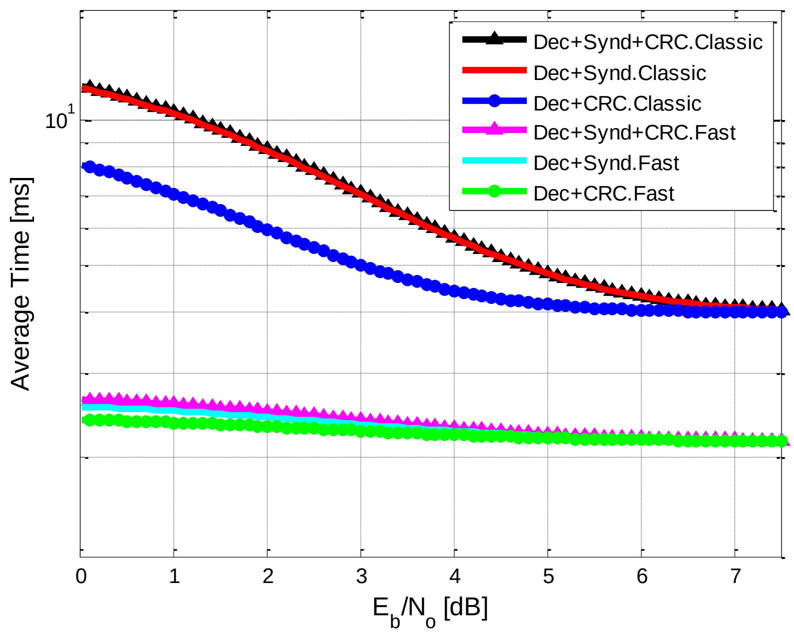

Fig 3.15 Decoding time comparison between the classic and the fast decode.

The classic decoder computes all the Syndromes of the corrected codeword from zero to determine if it is valid, while the fast decoder updates the Syndromes of the received codeword by adding the GF elements of the erroneous positions normalized by the number of Syndrome.

5 Conclusions

BCH codes belong to the class of linear cyclic codes supported by an algebra de Galois Field, GF. In particular, the decoding processes of BCH codes have the following stages: i) Computation of the Syndromes; ii) Determination of the Error Locator Polynomial; iii) Solving the roots of the ELP; and iv) Validating the corrected codeword.

From all these stages, the Computation of the Syndromes and the Solving process of the ELP are the most time consuming stages since they depend on the codeword length. In order to speed up the Solving Process of the ELP this project proposes to express the coefficients of the ELP in terms of the erroneous positions of the codeword. In this way, it is possible to i) update the ELP once a root has been detected; and ii) obtain decoding architectures of reduced complexity.

In particular, it is shown that if there are two erroneous bits separated a power of two then it is possible to obtain a low complexity solution for the two erroneous positions. That solution can be implemented in reduced size LUT or in a reduced number of logical gates.

In addition, for this case it is shown that only is required to compute the odd syndromes which reduce the complexity of the Computation of the Syndromes' stage to the half. Finally, in the validating process of the corrected codeword is provided a fast updating strategy of the syndromes that avoids their dependence from the codeword length.

6 Future lines of research

- Join design of the Berlekamp-Masey algorithm and the Chien's search algorithm.
- Extend the fast solution of two errors to three and more.
- Obtain the fast solution of two errors without using a Look-Up Table.

7 Bibliography

[1] D.Chase,"A class of algorithms for decoding block codes with channel measurement information," IEEE Trans. Inf. Theory, vol.IT-18,nº1,pp. 170-182,Jan. 1972.

[2] R.Koetter and A.Vardy,"Algebraic soft-decision decoding of Reed Solomon Codes," IEEE Trans. Inf. Theory, vol. 49,nº 11,pp.2809-2825,Nov. 2003.

[3] S..Lin and D.J.Costello, Error Control Coding: Fundamentals and Applications. Englewood Cliffs, NJ:Prentice-Hall, 2nd. Edition, 2004.

[4] Y.Lee,H.Yoo,I.Park,"Low-Complexity Parallel Chien Search Structure Using Two-Dimensional Optimization",In Proc. IEEE Trans. On Circuits and Systems-II:Express Briefs,vol.58,nº8,August 2011,pp.522-526.

[5] N.Ahmadi,M.Hasan,A.Dipta,T.Adiono,"An Optimal Architecture of BCH Decoder", In Proc. Application of Information and Communication Technologies (AICT), October 2010, pp.1-5, Tashkent, Uzbekistan.

[6] C.Chu,Y.Lin,C.Yang,H. Chang,"A Fully Parallel BCH Codec with Double Error Correcting Capability for NOR Flash Applications", In Proc. Of IEEE Acoustics, Speech and Signal Processing (ICASSP) Conference, 2012,pp.1605-1608, Kyoto, Japan.

[7] J.Cho,W.Sung,"Strength-Reduced Parallel Chien Search Architecture for Strong BCH Codes", In Proc. IEEE Trans. On Circuits and Systems II-Express Briefs, vol. 55,nº5,May 2008,pp.427-431.

[8] W.Xueqiang,P.Liyang,W.Dong,H. Chaohong,and Z. Runde,”A High-Speed Two-Cell BCH Decoder for Error Correcting in MLC NOR Flash Memories”, In Proc. IEEE Trans. On Circuits and Systems, II: Express Briefs, vol. 56,nº11,November 2009, pp. 865-869.

[9] H.Yoo,Y.Lee,I.Park,”Area Efficient Syndrome calculation for strong BCH decoding”, In Proc. IEEE Electronic Letters, January 2011, vol.47,nº2,pp.107-108.

[10] 802.15.1 IEEE standard for Information technology –Telecommunications and information exchange between systems- Local and metropolitan area networks- Specific requirements Part 15.1 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs).