UNIVERSITY OF PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE:
ELETTRONICA, INFORMATICA, TELECOMUNICAZIONI

Doctorate in INFORMATION ENGINEERING

Curriculum in COMMUNICATION SYSTEMS

Iterative Message-Passing-Based Algorithms to Detect Spreading Codes

Presented by:
FABIO PRINCIPE .....

Supervisor:
Prof. Marco Luise .....

Accepted by: **Doctorate Council of Dip. di Ingegneria dell'Informazione**
President: **Prof. Lanfranco Lopriore**

Submission date: 26 February 2007 — *Acceptance date:* 20 March 2007

... to my father,

I miss you ...

Acknowledgements

I would like to thank *Margherita*, because during these three years she has been always with me. She has lovingly supported me in bad times and enjoyed happy moments with me.

I thank my *mother* and my *brother* because they have always believed in my capabilities and skills, so approving and supporting every my decisions and work experiences.

I thank *Prof. Marco Luise* for his precious suggestions and for the great working environment, that he have created with the collaboration of the *DSP-lab staff*. It has been a wonderful experience working together!

I would like to thank *Prof. Keith M. Chugg* and *his staff* for the opportunity that they gave me to visit USC and LA, their great hospitality, and the precious help and collaboration to perform the research topic, which is the base of this thesis.

Thank you very much!!!

Abstract

This thesis tackles the issue of the rapid acquisition of spreading codes in Direct-Sequence Spread-Spectrum (DS/SS) communication systems. In particular, a new algorithm is proposed that exploits the experience of the iterative decoding of modern codes (LDPC and turbo codes) to detect these sequences. This new method is a *Message-Passing-based algorithm*. In other words, instead of correlating the received signal with local replicas of the transmitted *linear feedback shift register* (LFSR) sequence, an iterative Message-Passing (iMP) algorithm is implemented to be run on a *loopy graph*. In particular, these graphical models are designed by manipulating the *generating polynomial* structure of the considered LFSR sequence. Therefore, this contribution is a detailed analysis of the detection technique based on Message-Passing (MP) algorithms to acquire *m-Sequences* and *Gold codes*. More in detail, a unified treatment to design and implement a specific set of graphical models for these codes is reported. A theoretical study on the acquisition time performance and their comparison to the standard algorithms (full-parallel, simple-serial, and hybrid searches) is done. A preliminary architectural design is also provided. Finally, the analysis is also enriched by comparing this new technique to the standard algorithms in terms of computational complexity and (missed/wrong/correct) acquisition probabilities as derived by simulations.

Contents

List of Figures

List of Tables

List of Abbreviations, Operators, and Symbols

Acronyms

AWGN	Additive White Gaussian Noise
BB	Base-Band
BGM	Basic Graphical Model
CD	Correct Detection
CDMA	Code Division Multiple Access
DLL	Delay Locked-Loop
DS/SS	Direct-Sequence/Spread-Spectrum
e.g.	Exempli Gratia (from Latin: For Example)
FA	False Alarm
FBA	Forward Backward Algorithm
FLL	Frequency Locked-Loop
FPA	Full-Parallel Algorithm
FSM	Finite State Machine
FSR	Feedback Shift Register

GF	Galois Field
GPS	Global Positioning System
HA	Hybrid Algorithm
HM	Hierarchical Model
IB	Input Buffer
iDU	Iterative Detection Unit
iMP	Iterative Message Passing
iPU	Iterative Processing Unit
LDPC	Low-Density Parity-Check (codes)
LSFR	Linear Feedback Shift Register
MD	Missed Detection
MGF	Moment Generating Function
ML	Maximum Likelihood
MP	Message Passing
MS	Min-Sum
NLSFR	Nonlinear Feedback Shift Register
PCU	Parity Control Unit
PLL	Phase Locked-Loop
PN	Pseudo Noise
RB	Refresh Buffer
RGM	Redundant Graphical Model
SBAS	Satellite Based Augmentation System
SNR	Signal-to-Noise Ratio
SP	Sum-Product
SR	Shift Register

SSA	Simple-Serial Algorithm
SS	Spread Spectrum
TG	Tanner Graph
UWB	Ultra Wide-Band
WD	Wrong Detection

Algebra

$\mathbf{a}$	Infinite sequence, whose elements $\{a_i\} \in F$, $i = 0, 1, \ldots$
$A(\mathbf{a})$	Set of all polynomials that satisfy $f(D)\mathbf{a} = 0$, on a prefixed $\mathbf{a}$
$F[x]$	Polynomial ring over $F = GF(2) = \{0, 1\}$
F	Finite field of order 2, $F = GF(2) = \{0, 1\}$
G^*	Multiplicative group with nonzero elements, for a finite field G
$G(f)$	Set of all sequences $\mathbf{a} \in V(f)$ such that $f(D)\mathbf{a} = 0$
$GF\,(q)$	Finite field of order q (q is a prime or a power of a prime)
$\mathbb{N}$	Set of natural numbers (positive integers)
$\mathbb{N}^*$	Set of natural numbers (positive integers), without 0
$V(F)$	Set of all infinite sequences whose elements are taken in F
$\mathbb{Z}$	Set of integers

Operators

D	Left-shift operator
$g(x) \mid f(x)$	The polynomial $g(x)$ is a divisor of the polynomial $f(x)$
$\mid G \mid$	Order of the finite group G
gcd	The greatest common divisor

ord (α)	Order of an element $\alpha \in G$, with G is a group
per (**a**)	Period of the sequence **a**
per $(f(x))$	Period of the polynomial $f(x)$
Q (.)	Complementary cumulative distribution function of a standard Gaussian random variable
$\mathcal{S}$ (.)	Sign function $\mathcal{S}$ (.) = sgn (.)
a $\backsim$ **b**	The sequences **a** and **b** are shift equivalent
$\oplus$	Modulo-2 addition or exclusive or-operator

Symbols

C_{Alg}	Complexity of FPA, SSA, HA, or iMP (respectively C_{FP}, C_{SS}, C_H, C_{iMP})
E_c	Chip energy
E_c/N_0	SNR
I_{MAX}	Maximum number of iterations
M	Number of observations at the receiver side
N	PN sequence period
$\mathcal{P}_{Case}$	Probability of WD, MD, FA, and CD (respectively $\mathcal{P}_{WD}$, $\mathcal{P}_{MD}$, $\mathcal{P}_{FA}$, $\mathcal{P}_{CD}$)
π	Its value is about 3.1415926535898
T_c	Chip time ($R_c = 1/T_c$ is the chip rate)
τ_d	Dwell Time
τ_{iPU}	Time delay of iPU
T_{it}	Iteration time ($R_{it} = 1/T_{it}$ is the iteration rate)
τ_{MD}	Time delay in case of MD
τ_{pt}	Penalty Time
τ_{RB}	Time delay to RB

Chapter 1

Introduction

1.1 Motivations

DIRECT-SEQUENCE/SPREAD-SPECTRUM (DS/SS) systems have been developed since the mid-1950's. They are widely used in wireless military and civil communications as well as in satellite positioning systems, because they provide low probability of interception, strong anti-jam protection, and low co-channel interferences. All these proprieties are basically due to use of *long* high-rate binary *pseudo-noise* (PN) sequences that spread the spectrum bandwidth and make it difficult to be detected and corrupted by jammers. Therefore, long period sequences are more desirable than shorter ones that make the link susceptible to repeat-back jamming or interception/detection via delay and correlate methods, [53].

At the receiver side, to correctly demodulate SS signals, a *despreading* operation is accomplished by correlating the incoming signal with local replicas of its PN sequence. Because of this, a precise code timing synchronization is necessary. This result is generally got in two receiver stages ([42], [44], [49], and [53]): the *acquisition stage*, that provides a preliminary coarse alignment between the received PN sequence and its local

replica, and the *tracking stage*, in which fine synchronization is realized and maintained by a *delay locked-loop* (DLL) unit ([12] and [13]), exploiting the previous rough alignment. Therefore, PN acquisition is the critical point to have rapid and correct synchronization between DS/SS transmitters and receivers. Of course, this condition is important to design a more dynamic receiver, that better tracks any change of the channel condition.

The standard and well-known acquisition techniques used to detect such sequences are ([23], [41], [44], [45], and [53]): full-parallel, simple-serial, and hybrid searches. The common denominator of all these techniques is that the received and local SS sequences are correlated and then processed by a suitable detector/decision rule to decide whether the two codes are in synchronism or not. Specifically, the first method implements a *maximum likelihood* (ML) estimation algorithm with a fully parallel search. Hence, it provides fast detection at price of a high implementation complexity, especially in case of long SS sequences. The opposite solution is the simple serial search, that has the lowest complexity, but its acquisition time is prohibitively long. The hybrid search mixes the two previous methods, resulting in a trade-off between these algorithms.

In this context, new techniques to acquire *linear feedback shift register* (LSFR) sequences have been, recently, presented in [10], [59], [61], and [62]. All these methods are based on the paradigm of *Message Passing* (MP) on graphical models, or, more specifically, on *iterative Message Passing* (iMP) algorithms to be run on loopy graphs ([1], [9], [34], [37], [56], and [60]). In other words, instead of correlating the received signal with a local PN replica (as in all standard methods), this algorithm uses all the information, provided by the incoming signal, as messages to be run on a predetermined graphical model with cycles, thus approximating the ML method. This results in a sub-optimal algorithm, that searches all possible code phases in parallel with a complexity typically lower than the full-parallel implementation, and an acquisition time shorter than that of the simple-serial algorithm. Furthermore, considering LFSR sequences characterized by sparse *generating polynomials*, it has been shown in [63] that significant improvements in terms of acquisition probability can be obtained using *redundant graphical models*

(RGMs) made up of a set of redundant parity check equations.

These motivations have induced us to investigate iMP algorithms to detect long LFSR sequences, so laying the bases of the present dissertation. More specifically, focusing on the approach presented in [10] and [63], we fully describe and analyze the iMP detector in terms of acquisition time, detection performance, and algorithm complexity. Furthermore, exploiting the algebraic description of LFSR sequences, based on *Galois Field* (GF) theory [20], we propose a unified treatment to detect *m*-sequences and Gold codes. All previous papers ([10], [59], [61], [62], and [63]) were only addressed to acquire *m*-sequences, and they showed that the acquisition performance decreases dramatically in case of *dense* primitive polynomials. Our methodology outperforms these results, showing better acquisition probability at low signal-to-noise-ratio (SNR) and low complexity, for a broader class of LFRS sequences that includes Gold codes.

We also remark that the analysis of iMP detectors is enriched with theoretical studies on the acquisition-time performance, based on the Holmes seminal work ([26] and [27]), and related measurements. Of course, all these results are compared to the corresponding parameters of the standard algorithms (full-parallel,[23] and [45], hybrid, and simple-serial search, [53]).

1.2 Key Points and Contributions

The equivalence between a SS acquisition problem and a decoding one is the cardinal point on which this thesis is based. Because of this, it is possible to exploit the iterative decoding techniques of modern codes (tubo codes, [6], and LDPC, [17]) to acquire LFRS sequences as demonstrated in [10], [59], [61], and [62]. More specifically, focusing on the techniques presented in [10] and [63], SS signals can be acquired running MP algorithms on loopy graphs, that are obtained manipulating generating polynomials of the considered LFSR sequences. In this way it is possible to perform a full parallel detection with low-complexity and rapid acquisition.

Nevertheless, we will see that to have good performance in case of LFSR sequences characterized by dense generating polynomials and also acquire Gold codes, it will be necessary to resort to GF properties and related algorithms typically applied in cryptography problems, [20], [21] and [38]. Such techniques allow us to easy manipulate generating polynomials, so producing RGMs, that guarantee very interesting performance in terms of acquisition probability at low-complexity.

In this context, the contributions of this thesis are listed below.

- The algebraic introduction of LFSR sequences (based on GF theory) allows to define m-sequences and Gold codes as sub-sets of the LFSR sequence family (see [18], [19], and [20]). This consideration lays the bases to have a unified treatment to acquire m-sequences and Gold codes, so adapting and improving the algorithms proposed in [10] and [63].

- About the design of graphical models used to be run iMP algorithms, we propose some simple algorithms (typically used in cryptographic applications, [30] and [24]) that easy manipulate dense primitive polynomials to obtain *sparse* polynomials multiple of them. These polynomials can be efficiently used to implement sparse TGs with redundancy (called RGMs), that offer very good performance at low SNR and low-complexity.

- A preliminary architectural design of an iMP detector is provided with a detailed description of the acquisition algorithm.

- We also present a theoretical study to evaluate the acquisition-time performance of an iMP detector (based on [27] and [26]) and related measurements, so comparing them to those of the standard algorithms (full-parallel, hybrid, and simple-serial searches), [46].

- Finally, to reduce the memory requirements, some different schedules are tested, and a distinct approach for acquiring Gold codes is also reported. This new method

basically runs an iMP algorithm on a hierarchical model (see also [47]) generated by the two primitive polynomials that comprise a Gold code.

1.3 Applications

As mentioned in Section 1.1, DS/SS techniques are widely applied in many communication systems for several applications, e.g.: civil/commercial, military, safety and rescue, satellite positioning systems, etc. Of course, each system is different from the others, so their requirements, in terms of performance, signal/data processing algorithms, etc., are different. Nevertheless, the common characteristic is the importance that the acquisition stage has in all these systems to have fast and reliable detections of PN sequences.

In this context, the acquisition algorithm, which is described in this dissertation, can find large applications, because it guarantees good performance and acquisition times that tend to that of a full-parallel search, but with a lower complexity. Furthermore, we remark that this algorithm has been successfully applied to acquire *m*-sequences in *Ultra Wide-Band* (UWB) systems (see [10] and [63]). Therefore, its extension to *Code Division Multiple Access* (CDMA) systems could be an important innovation to solve the critical issue of long spreading-sequence detection.

Taking into account these considerations, in this thesis the GPS/SBAS[1] ([2], [22], [33], and [51]) codes have been specially taken into consideration, in order to have a realistic context to evaluate the performance achievable using the iMP detector for the acquisition of Gold sequences.

1.4 Outline

The remainder of this thesis is structured as follows: **Chapter 2** introduces LFSR sequences in terms of finite field theory. More specifically m-sequence and Gold codes

[1] GPS stands for Global Positioning and SBAS stands for Satellite Based Augmentation Systems.

are defined as subsets of LFSR sequence family, providing the basis to have a unified treatment to acquire these codes using MP algorithms. **Chapter 3** gives a mathematical description of the communication system and the signal model that will be used to simulate the detection of SS signals. Architectural design of the *iterative detection unit* (iDU) is presented in **Chapter 4** with a description of MP algorithms, their characteristics, and implementations. **Chapter 5** shows some interesting results obtained by computer simulations which prove that MP algorithms can effectively acquire spreading codes. Finally conclusions and suggestions for future work are reported in **Chapter 6**.

Chapter 2

Linear Feedback Shift Register Sequences

Linear feedback shift register (LFSR) sequences have been widely used in many applications (as random number generators, scrambling codes, for white noise signals, etc.). Nevertheless, our interest is focused on the generation of *m-sequences* and *Gold codes*, that are commonly used in SS systems for their attractive properties of autocorrelation and cross-correlation ([14], [41], and [53]). Therefore, the outline of this chapter is reported below.

- **Section 2.1** provides a definition of *feedback shift register* (FSR) sequences, their basic concepts, and properties.
- **Section 2.2** presents LFSR sequences in terms of polynomial rings.
- **Section 2.3** gives an algebraic characterization of m-sequences and also provides their properties.
- **Section 2.4** contains an overview of Gold codes their definition and properties.

2.1 Feedback Shift Register Sequences

In this section, we define some basic concepts for FSR sequences. We denote the finite field $F = GF(2) = \{0, 1\}$ and

$$F^n = \{\mathbf{a} = (a_0, a_1, \ldots, a_{n-1}) | a_i \in F\}$$

a vector space over F of dimension n. A function with n binary inputs and one binary output is called a Boolean function of n variables, that is $f : F^n \rightarrow F$ which can be represented as

$$f(x_0, x_1, \ldots, x_{n-1}) = \sum_{i_1, i_2, \ldots, i_t} c_{i_1 i_2 \ldots i_t} x_{i_1} x_{i_2} \ldots x_{i_t}, \quad c_{i_1 i_2 \ldots i_t} \in F \tag{2.1}$$

where the sum runs through all subsets $\{i_1 i_2 \ldots i_t\}$ of $\{0, 1, \ldots, n-1\}$. This shows that there are 2^{2^n} different boolean functions of n variables.

2.1.1 Basic Concepts

An n-stage *shift register* (SR) is basically made up of n consecutive 2-state storage units (flip-flops) regulated by a single clock. At each clock pulse, the state (1 or 0) of each memory is shifted to the next stage in line. To have a code generator, a SR is integrated with a feedback loop, which computes a new term for the left-most stage, based on the n previous terms. So, a generic feedback shift register (FSR) generator is depicted in Fig. 2.1.

Typically, the n binary storage elements are called the stages of the SR, and their contents are a *state* of the SR. The initial state is $(a_0, a_1, \ldots, a_{n-1}) \in F^n$. The feedback function $f(x_0, \ldots, x_{n-1})$ is a boolean function of n variables, as defined in (2.1). At every clock pulse, there is a transition from one state to the next, so, to obtain a new value for the stage n, we compute $f(x_0, \ldots, x_{n-1})$ of all present terms in the SR and use this in the

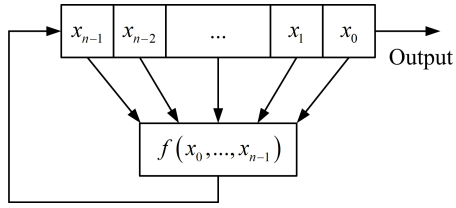

Figure 2.1: Generic configuration of a FSR generator.

stage n. Therefore, assuming that the FSR generator outputs a sequence

$$a_0, a_1, \ldots, a_n, \ldots, \tag{2.2}$$

at the generic time k the following recursive equation is satisfied

$$a_{k+n} = f(a_k, a_{k+1}, \ldots, a_{k+n-2}, a_{k+n-1}), \quad k = 0, 1, 2, \ldots \tag{2.3}$$

so, more in general, any n consecutive terms of the sequence (2.2)

$$a_k, a_{k+1}, \ldots, a_{k+n-2}, a_{k+n-1}, \quad k = 0, 1, 2, \ldots$$

represents a state of the shift register in Fig. 2.1. The output sequence is called a FSR sequence. Furthermore, if the feedback function, $f(x_0, \ldots, x_{n-1})$, is a linear function, then the output sequence is called *linear feedback shift register* (LFSR) sequence. Otherwise, it is called a *nonlinear feedback shift register* (NLFSR) sequence (see [20]). Over a binary field, *linear* means that the feedback function computes the modulo-2 sum of a subset of the stages of the SR.

We remark that this thesis is focused on LFSR sequences. In particular, we will see that their repetition periods are completely determined by their feedback functions in an easily predictable way. Furthermore, the design of LFSR sequences with desired properties requires us to understand the functionality of three components of an LFSR generator:

- the initial state;
- the feedback function;
- the output sequence.

More specifically, we will show how the behavior of the output sequences is determined by initial states and feedback functions. We mainly treat the binary case $F = GF(2)$, but a general analysis (with $F = GF(q)$, where $q \geqslant 2$) is reported in [20].

2.1.2 Periodic Property

In this section, the periodic property of a generic FSR sequence is defined, [20].

Definition 2.1. *The sequence* $a_0, a_1, \ldots$ *is denoted as* **a** *or* $\{a_i\}$. *If* $a_i \in F$, *then we say that* **a** *is a binary sequence or a sequence over F. If there exist two integers* $r > 0$ *and* $u > 0$ *such that*

$$a_{i+r} = a_i, \quad \forall i \geqslant u \tag{2.4}$$

then the sequence is said to be ultimately periodic *with parameters* (r, u) *and r is called a* period *of the sequence. The smallest number r satisfying (2.4) is called a* (least) period *of the sequence. if* $u = 0$, *then the sequence is said to be* periodic. *When the context is clear, we simply say the period of* **a** *instead of the least period of* **a**.

Let us see a simple example.

Example 2.1. Considering the output sequence 00011011011 ... of a 4-stage LFSR with feedback function $f(x_0, x_1, x_2, x_3) = x_2 + x_3$ and initial state $a_0 a_1 a_2 a_3 = 0001$, it is an *ultimately periodic* sequence, where $u = 2$ and the period r is 3.

The following theorem gives a general property for any q-ary FSR sequence[1] (with $F = GF(q)$ and $q \geqslant 2$).

[1] A binary FSR sequence can be considered as a particular case of a generic q-ary FSR sequence. In other word, let $F = GF(q)$ where q is a prime or a power of a prime (so $|F| = q$ and $q \geqslant 2$), and referring to Fig. 2.1, each stage is replaced by a q-state storage unit and the boolean feedback function (2.1) is replaced by a more general function from F^n to F. More details are given in [20].

Theorem 2.1. *Any q-ary FSR sequence is ultimately periodic with period $r \leqslant q^n$, where n is the number of the stages. In particular, if $q = 2$ then $r \leqslant 2^n$.*

Proof. In a q-ary FSR with n stages, there are q^n possible states. Each state uniquely determines its successor. Hence, the first time that a previous state is repeated, a period for the sequence is established. Thus, the maximum possible period is q^n, that is the maximum number of different states. □

2.1.3 Linear Feedback Shift Register Sequences

Assuming to have a linear feedback function

$$f(x_0, x_1, \ldots, x_{n-1}) = c_0 \cdot x_0 + c_1 \cdot x_1 + \ldots + c_{n-1} \cdot x_{n-1}, \quad c_i \in F = GF(2)$$

the recursive equation shown in (2.3) becomes

$$a_{k+n} = \sum_{i=0}^{n-1} c_i \cdot a_{k+i}, \quad k = 0, 1, \ldots. \tag{2.5}$$

Thus, an LFSR is also referred to as a *linear recursive sequence* over F. Note that it is possible to have only 2^n different n-stage LFSRs. On this consideration it is based the following general theorem.

Theorem 2.2. *Let* **a** *be a sequence generated by an n-stage LFSR over $F = GF(q)$. Then the period of* **a** *is less then or equal to $q^n - 1$. In particular, if $q = 2$, the period of any binary n-stage LFSR sequence is less then or equal to $2^n - 1$.*

Proof. Consider that the successor of state $00\ldots0$ (n times 0) of an n-stage LFSR is again $00\ldots0$. Using the same argumentations as in the proof of Theorem 2.1, we see that the period of **a** is $\leqslant q^n - 1$, since the state $00\ldots0$ cannot be a part of any period. □

We remark that in the rest of this chapter we restrict our attention to LFSR sequences.

2.2 LFSR Sequences in Terms of Polynomial Rings

This section is addressed to characterize the periodicity property of LFRS sequences, giving an equivalent definition of these sequences over F in terms of *polynomial ring*, $F[x]$. Therefore, we provide the following definition.

Definition 2.2. *The ring formed by the polynomials over $F = GF(2)$ with the classic modulo-2 sum and product operations is called the* polynomial ring *over F and denoted by $F[x]$*

$$F[x] = \left\{ f(x) = \sum_{i=0}^{n} c_i x^i | c_i \in F, n \geqslant 0 \right\}.$$

This analysis is conducted over $F = GF(2)$, but its extension to $F = GF(q)$, where q is prime or power of a prime, is sometimes obvious, as shown in [20].

2.2.1 Characteristic Polynomial

Let $V(F)$ be a set made up of all infinite sequences whose elements are taken from $F = GF(2)$,

$$V(F) = \{\mathbf{a} = (a_0, a_1, \ldots) | a_i \in F\}.$$

Assuming to have two generic sequences

$$\mathbf{a} = (a_0, a_1, a_2, \ldots) \in V(f)$$
$$\mathbf{b} = (b_0, b_1, b_2, \ldots) \in V(f)$$

and $c \in F$, we define the addition and scalar multiplication on $V(F)$ as follows

$$\mathbf{a} + \mathbf{b} = (a_0 + b_0, a_1 + b_1, a_2 + b_2, \ldots)$$
$$c \cdot \mathbf{a} = (c \cdot a_0, c \cdot a_1, c \cdot a_2, \ldots).$$

Introducing the zero sequence $\mathbf{0} = (000\ldots)$, it is easy to verify that $V(F)$ is a *vector space* over F under these two operations. Thus, an LFRS sequence is a sequence, $\mathbf{a} = (a_0, a_1, \ldots)$, in $V(F)$ whose elements satisfy the linear recursive equation

$$a_{n+k} = \sum_{i=0}^{n-1} c_i a_{k+i}, \quad k = 0, 1, \ldots. \tag{2.6}$$

For any sequence $\mathbf{a} = (a_0, a_1, a_2 \ldots) \in V(F)$, we define a *left shift* operator D as follows

$$D\mathbf{a} = (a_1, a_2, a_3, \ldots).$$

Note that D is a *linear transformation* of $V(F)$. Generally, for any positive integer i, we have

$$D^i\mathbf{a} = (a_i, a_{i+1}, a_{i+2}, \ldots).$$

By convention, we define $D^0\mathbf{a} = I\mathbf{a} = \mathbf{a}$, where I is the *identity* transformation on $V(F)$. Using the D operator, the recursive formula (2.6) becomes

$$D^n\mathbf{a} = \sum_{i=0}^{n-1} c_i D^i\mathbf{a}$$

or equivalently

$$D^n\mathbf{a} - \sum_{i=0}^{n-1} c_i D^i\mathbf{a} = 0. \tag{2.7}$$

So we can write

$$f(x) = x^n - \left(c_{n-1}x^{n-1} + \ldots + c_0\right)$$
$$f(D) = D^n - \left(c_{n-1}D^{n-1} + \ldots + c_0 I\right), \text{ and } f(D)\mathbf{a} = 0.$$

and, considering (2.7), we give the following definition.

Definition 2.3. *For any infinite sequence* $\mathbf{a} \in V(F)$*, if there exists a nonzero monic*[2] *polynomial* $f(x) \in F[x]$ *such that*

$$f(D)\mathbf{a} = 0,$$

then $\mathbf{a}$ *is called a* linear recursive sequence*, or equivalently* LFSR sequence*. The polynomial* $f(x)$ *is referred to as* characteristic polynomial *of* $\mathbf{a}$ *over* F*. The reciprocal polynomial is called the* feedback polynomial *of* $\mathbf{a}$*.*

More details on reciprocal polynomials are reported in [20]. Here, we just give its definition over $GF(2)$.

Definition 2.4. *Let* $f(x) = x^n + c_{n-1} \cdot x^{n-1} + \ldots + c_1 \cdot x + c_0$*,* $c_i \in GF(2)$ *and* $c_0 \neq 0$ *(so* $c_0 = 1$*), the* reciprocal polynomial *of* $f(x)$ *is defined as*

$$f^{-1}(x) \triangleq \frac{x^n}{c_0} \cdot f(x^{-1}) = x^n + c_1 \cdot x^{n-1} + \ldots + c_{n-1} \cdot x + 1.$$

One more definition is needed before proceeding further on.

Definition 2.5. *For any nonzero polynomial* $f(x) \in F[x]$*,* $G(f)$ *represents the set made up of all sequences in* $V(F)$ *with* $f(D)\mathbf{a} = 0$

$$G(f) = \{\mathbf{a} \in V(F) | f(D)\mathbf{a} = 0, f(x) \in F[x]\}.$$

Since $f(D)$ is a linear transformation, $G(f)$ is a subspace of $V(F)$. Furthermore, we remark that, by convention, the constant polynomial 1 is the characteristic polynomial of the zero sequence $000\ldots$.

Theorem 2.3. *Let* $f(x) \in F[x]$ *be a monic polynomial of degree* n*. Then* $G(f)$ *is a linear space of dimension* n*. Hence, it contains* 2^n *different binary sequences.*

[2] A monic polynomial is a polynomial in which the coefficient of the highest order term is 1, e.g.: $x^n + c_{n-1} \cdot x^{n-1} + \ldots + c_1 \cdot x + c_0$.

Proof. For a sequence

$$a = (a_0, a_1, \ldots, a_{n-1}, a_n, \ldots) \in G(f)$$

since $\deg(f) = n$, when the first n terms are given, all other terms of **a** can be determined by (2.6) starting from a_n. There are 2^n ways to choose and n-tuple $(a_0, a_1, \ldots, a_{n-1}) \in F$. Therefore $|G(f)| = 2^n$. □

Note that the sequences in $V(F)$ may or may not be periodic, in particular, applying Def. 2.3 of LFSR sequences, it is easy to check the periodicity of these sequences, as we will show in the next section.

Finally, we just remark the key points of this section. More specifically, if a sequence, $\mathbf{a} = (a_0, a_1, a_2, \ldots)$, is generated by an n-stage LFSR, the following three equivalent definitions are verified:

1. $\mathbf{a} = (a_0, a_1, a_2, \ldots)$ is an output sequence of an LFSR generator with the linear feedback function

   $$f(x_0, x_1, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} c_1 x_i, \quad c_i \in F$$

   and an initial state $(a_0, a_1, \ldots, a_{n-1})$, so all elements of **a** satisfy the following recursive relation

   $$a_{n+k} = \sum_{i=0}^{n-1} c_i a_{k+i}, \quad k = 0, 1, \ldots. \tag{2.8}$$

2. **a** is a linear recursive sequence that satisfies the above recursive equation, (2.8).

3. There exists a monic polynomial $f(x) = x^n - \sum_{i=0}^{n-1} c_i x^i$, with $f(x) \in F[x]$, of degree n such that $f(D)\mathbf{a} = 0$ or, equivalently, $\mathbf{a} \in G(f)$.

The polynomial $f(x)$ is referred to as *characteristic polynomial* of **a** and the reciprocal polynomial of $f(x)$ is called the *feedback polynomial* of **a**.

2.3 Minimal Polynomials and *M*-Sequences

In the previous section, LFSR sequences over F have been associated with polynomials over F. This result enables us to use the theory of the polynomial ring $F[x]$ to investigate the periods of LFSR sequences and discuss the *minimal polynomials*. In this way, it is possible to define m-sequences, or equivalently LFSR sequences with the maximum period.

2.3.1 Minimal Polynomials of LFSR Sequences

Let **a** be an LFSR sequence, so there is a nonzero monic polynomial $f(x)$ such that

$$f(D)\mathbf{a} = 0. \tag{2.9}$$

Nevertheless, for the fixed sequence **a**, there are many polynomials for which (2.9) is verified.

Example 2.2. Assuming to have the LFSR sequence $\mathbf{a} = 011011\ldots$, then both the polynomials $f_1(x) = x^2 + x + 1$ and $f_2(x) = x^3 + 1$ satisfy the property (2.9).

Hence, we want to find the relation among these polynomials, for a fixed LFSR sequence **a**. Therefore, we define

$$A(\mathbf{a}) = \{f(x) \in F[x] | f(D)\mathbf{a} = 0\}, \tag{2.10}$$

in other words, $A(\mathbf{a})$ is made up of all polynomials that verify the condition (2.9). The following theorem gives a set of properties that are verified by all characteristic polynomials of $A(\mathbf{a})$.

Theorem 2.4. *Let* **a** *be an LFSR sequence and* $A(\mathbf{a})$ *be defined as (2.10). Then* $A(\mathbf{a})$ *satisfies the following properties*

1. *The zero polynomial belongs to* $A(\mathbf{a})$.

2. *if* $f(x)$, $g(x) \in A(\mathbf{a})$, *then* $f(x) \pm g(x) \in A(\mathbf{a})$.

3. *if* $f(x) \in A(\mathbf{a})$ *and* $h(x) \in F[x]$, *then* $h(x) \cdot f(x) \in A(\mathbf{a})$.

Proof. We demonstrate all points in the following list.

1. $0\mathbf{a} = 0 \Longrightarrow 0 \in A(\mathbf{a})$.

2. Assuming $f(x), g(x) \in A(\mathbf{a})$, then

$$\begin{aligned} &f(D)\mathbf{a} = 0 \quad \text{and} \quad g(D)\mathbf{a} = 0 \\ &(f(D) \pm g(D))\,\mathbf{a} = f(D)\mathbf{a} \pm g(D)\mathbf{a} = 0 \\ &f(D) \pm g(D) \in A(\mathbf{a}). \end{aligned}$$

3. Assuming $f(x) \in A(\mathbf{a})$, then

$$\begin{aligned} &f(D)\mathbf{a} = 0 \\ &(h(D) \cdot f(D))\,\mathbf{a} = h(D)\,(f(D)\mathbf{a}) = h(D)0 = 0. \end{aligned}$$

Hence, all points are proved. □

Since $A(\mathbf{a})$ is closed with respect to all these operations, so $A(\mathbf{a})$ is an *algebra*. Now, we can give the definition of *minimal polynomial*.

Definition 2.6. *A monic polynomial of the lowest degree in* $A(\mathbf{a})$ *is referred to as* minimal polynomial *of* $\mathbf{a}$ *over* F.

According to Def. 2.6, we remark that the minimal polynomial of the zero sequence $000\ldots$ is 1, and the polynomial $x-1$ is the minimal polynomial of any constant sequence, $(1, 1, 1, \ldots)$. Furthermore, it is also possible to demonstrate the following theorem.

Theorem 2.5. *Let* $\mathbf{a} \in V(F)$ *and* $m(x)$ *be the minimal polynomial of* **a**. *Then the minimal polynomial of* **a** *is unique and satisfies the following two properties.*

1. $m(D)\mathbf{a} = 0$.
2. *For* $f(x) \in F[x]$, $f(D)\mathbf{a} = 0$ *iff m(x) is a divisor of f(x) (also pointed out* $m(x)|f(x)$*).*

The proof is shown in [20]. Of course, for any $\mathbf{a} \in G(f)$, $f(x)$ does not need to be the minimal polynomial of **a**, and so the following result is obvious.

Corollary 2.1. *If* $f(x) \neq 0$, $\mathbf{a} \in G(f)$, *then the minimal polynomial of* **a**, *called as* $m(x)$, *divides* $f(x)$, *or briefly* $m(x)|f(x)$.

Proof. According to the definition of $G(f)$ (Def. 2.5), $\mathbf{a} \in G(f) \Longrightarrow f(D)\mathbf{a} = 0$. So applying Theorem 2.5, $m(x)|f(x)$. □

we introduce now the concept of *irreducibility*[3] over F[x] to obtain another interesting result can be obtained.

Definition 2.7. *A polynomial* $f(x) \in F[x]$ *is referred to as* irreducible *over* F *if* $f(x)$ *has positive degree and* $f(x) = h(x) \cdot g(x)$ *with* $h(x), g(x) \in F[x]$ *implies that either* $h(x)$ *or* $g(x)$ *is a constant polynomial. Otherwise,* $f(x)$ *is called* reducible *over* F.

Corollary 2.2. *If* $f(x)$ *is irreducible, then* $f(x)$ *is the minimal polynomial of any nonzero sequence in G(f).*

Proof. Considering that $f(x)$ has only 1 and itself as its factor and the zero sequence has 1 as its minimal polynomial, then $f(x)$ is a minimal polynomial of any nonzero sequence in $G(f)$. □

According to Def. 2.6, an important result is that the degree of the minimal polynomial of **a** is equal to the *length* of the shorter LFSR generator that can *output* **a**. Furthermore, the degree of a minimal polynomial of a sequence is called the *linear span* of the sequence, as shown in the next subsection.

[3] More details and properties are reported in [20], [35], and [53].

2.3.2 Periodicity

For any periodic sequence, **a**, the following theorem is satisfied.

Theorem 2.6. *If* **a** *is an ultimately periodic sequence with parameters* (u, r)*, then the minimal polynomial of* **a** *is* $m(x) = x^u m_1(x)$ *with* $m_1(0) \neq 0$ *and* $m_1(x)|(x^r - 1)$*. Hence, it can be generated by an LFSR.*

Proof. Note that $a_{k+r} = a_k$, $k = u, u + 1, \ldots$, so

$$\Longrightarrow (D^r - 1)\, D^u(\mathbf{a}) = 0 \Longrightarrow m(x)|x^u\,(x^r - 1)\,.$$

Writing $m(x) = x^u m_1(x)$, then $m_1(0) \neq 0$ and $m_1(x)$ divides $x^r - 1$. Therefore, **a** can be outputted by an LFSR with characteristic polynomial $m(x)$. □

The following corollary immediately follows Theorem 2.6.

Corollary 2.3. *If* **a** *is a periodic sequence with period* r *(it means* $u = 0$*), then its minimal polynomial* $m(x)$ *divides* $(x^r - 1)$.

Proof. From Theorem 2.6, if **a** is ultimately periodic, then the minimal polynomial is $m(x) = x^u m_1(x)$, with $m_1(0) \neq 0$. In this case $u = 0$, so the minimal polynomial becomes $m(x) = m_1(x)$. □

Now, we give the following definition.

Definition 2.8. *Let* **a** *be an ultimately periodic sequence over* F*. Then the degree of the minimal polynomial of* **a** *is called* linear span *or* linear complexity *of* **a**.

Equivalently, the linear span of a periodic sequence is the length of the shortest LFSR generator that produce the sequence.

Lemma 2.1. *Let* r *be the least period of* **a***. If* l *is a period of* **a***, then* $r|l$.

Proof. Assuming that r and l are two periods of $\mathbf{a} \implies D^r\mathbf{a} = \mathbf{a}$ and $D^l\mathbf{a} = \mathbf{a}$. So, applying the division algorithm, we can write

$$l = q \cdot r + t, \quad 0 \leqslant t < r \quad \text{and} \quad q \in \mathbb{N},$$

because r is the smallest integer with this property ($r < l$). Considering that $D^{q \cdot r}\mathbf{a} = \mathbf{a}$, thus

$$D^l\mathbf{a} = D^{q \cdot r + t}\mathbf{a} = D^t\left(D^{q \cdot r}\right)\mathbf{a} = D^t\mathbf{a} = \mathbf{a}$$

so $t = 0$ and $r|l$. □

Now, we introduce the definition of the period of a polynomial (see also [20]), $f(x)$, over F. This step is fundamental to characterize *m*-sequences in terms of their generating polynomials.

Definition 2.9. *For a polynomial $f(x)$ over F, the* period *of $f(x)$ is the least positive integer r such that $f(x)|(x^r - 1)$.*

Hence, denoting by per (**a**) a period of a sequence **a** and per ($f(x)$) a period of a polynomial $f(x)$, we prove the following theorem.

Theorem 2.7. *Let* **a** *be an LFSR sequence with minimal polynomial $m(x)$.*

1. *If $m(0) \neq 0$, then* **a** *is periodic. In this case* per (**a**) = per ($m(x)$).
2. *If* **a** *is periodic, then $m(0) \neq 0$.*

Proof. We prove both these points.

1. Let **a** be an ultimately periodic sequence over F and with parameters (u, r) and $m(x)$ be its minimal polynomial. According to Def. 2.1, **a** is periodic if and only if $u = 0$. So, from Theorem 2.6, $m(x) = x^x m_1(x) = m_1(x)$ with $m_1(0) \neq 0$ and $m_1(x)|(x^r - 1)$. Thus, **a** is periodic if and only if $m(x) = m_1(x)$. According to Def. 2.9 and Corollary 2.3, it is proved per(**a**) = per ($m(x)$).

2. Conversely, if **a** is periodic, then $u = 0$ and $x^r - 1 \in A(\mathbf{a})$. According to Theorem 2.5, we have $m(x)|(x^r - 1) \Longrightarrow m(0) \neq 0$, which demonstrates the second assertion.

So, the theorem is demonstrated. □

Thus far, we have got a criterion for determining whether an ultimately periodic sequence is periodic evaluating its minimal polynomial at 0. Now, we want to show the relationships among the period of a sequence, the period of its minimal polynomial, and the *order* of a *root* of the minimal polynomial when the minimal polynomial is irreducible. Therefore, we first give a couple of definitions.

Definition 2.10. *An element $\alpha \in F$ is called a* root[4] *(or a* zero*) of the polynomial $f(x) \in F[x]$, if $f(\alpha) = 0$.*

Definition 2.11. *Let G be a group. For $\alpha \in G$, if r is the smallest positive integer such that $\alpha^r = 1$, then r is called* order *of α, denoted by* $\mathrm{ord}(\alpha) = r$.

Now, we prove the following important theorem.

Theorem 2.8. *Let* **a** *be an LFSR sequence with minimal polynomial $m(x)$. Assume that $m(x)$ is an irreducible polynomial over $F = GF(2)$ of degree n. Let α be a root of $m(x)$ in the extension field $GF(2^n)$[5]. Then*

$$\mathrm{per}\,(\mathbf{a}) = \mathrm{per}\,(m(x)) = \mathrm{ord}\,(\alpha)$$

in other words, the period of a sequence **a***, the period of its minimal polynomial, and the order of a root of the minimal polynomial of* **a** *are equal.*

[4] More details are contained in [20].

[5] The construction of an *extension field* $GF\,(2^n)$ is reported in App. A, but more details are contained in [20].

Proof. Noting that $m(x)$ is the minimal polynomial of α and according to Theorem A.2 in App. A, we have $\text{per}\,(m(x)) = \text{ord}\,(\alpha)$. Considering also Theorem 2.7

$$\text{per}\,(\mathbf{a}) = \text{per}\,(m(x)) = \text{ord}\,(\alpha)\,.$$

The assertion is established. □

This important result is emphasized here.

> *The period of an LFSR sequence,* **a***, is equal to the period of its minimal polynomial* $m(x)$*. If the minimal polynomial is irreducible, then the period of the sequence is equal to the order of a root*[6] α *of the minimal polynomial in the extension field. Briefly,* $\text{per}\,(\mathbf{a}) = \text{per}\,(m(x)) = \text{ord}\,(\alpha)$.

Thus, introducing the concept of *cyclic group*, it is possible to provide an algebraic definition of *primitive polynomial* (see also [20] and [35]).

Definition 2.12. *A multiplicative group* G *is said to be* cyclic *if there is an element* $a \in G$ *such that for any* $b \in G$ *there is some integer* i *with* $b = a^i$*. Such an element* a *is called a* generator *of the cyclic group, and we write* $G =< a >$.

Definition 2.13. *A generator of a cyclic group* $GF\,(2^n)^*$ *(this is a multiplicative group, without the zero element of* $GF\,(2^n)$*) is called a* primitive element *of* $GF\,(2^n)$*. An irreducible polynomial over* $GF\,(2)$ *having a primitive element in* $GF\,(2^n)$ *as a root is called a* primitive polynomial *over* $GF\,(2)$*. Equivalently, an irreducible polynomial* $f(x)$ *of degree* r *is said to be* primitive *if the smallest positive integer* p *for which* $f(x)$ *divides* $x^p + 1$ *(or* $x^p - 1$*) is* $p = 2^r - 1$.

Note that not all irreducible polynomials are primitive ([20] and [53]). We give an example.

[6] All roots have the same order [20].

Example 2.3. According to Def. 2.7, the polynomial $f(x) = x^4 + x^3 + x^2 + x + 1$ is irreducible over $GF(2)$, so it can be used to generate the field $GF\left(2^4\right)$. However, it is not a primitive polynomial. Indeed, let α be a root of $f(x)$, according to Def. 2.13, its order should be $2^4 - 1 = 15$, because α is a primitive element in $GF\left(2^4\right)$. Nevertheless, it possible to prove that $\text{ord}(\alpha) = 5 < 2^4 - 1 = 15$, so $f(x)$ is not a primitive polynomial over $GF(2)$.

2.3.3 *M*-Sequences

This section defines *m*-sequences in terms of the previous properties and theorems based on polynomial field theory, [20].

Definition 2.14. *Two periodic sequences* $\mathbf{a} = \{a_i\}$ *and* $\mathbf{b} = \{b_i\}$ *are called* shift equivalent *if there exists an integer* k *such that*

$$a_i = b_{i+k}, \quad \forall i \geqslant 0.$$

In this case, we write $\mathbf{a} = D^k\mathbf{b}$*, or* $\mathbf{a} \backsim \mathbf{b}$*. Otherwise, they are referred to as* shift distinct.

A set in which all sequences are shift equivalent is called a *shift-equivalent class*. One shift-equivalent class of $G(f)$ corresponds to one cycle of the states in the state diagram of the LFSR with $f(x)$. The number of shift-equivalent class is determined as follows.

Theorem 2.9. *Let* $f(x)$ *be an irreducible polynomial over* $GF(2)$ *of degree* n*. Then the number of shift-equivalent classes of nonzero LFSR sequences in* $G(f)$ *is given by*

$$\frac{2^n - 1}{\text{per}(f(x))}.$$

This theorem shows that for an LFSR with an irreducible polynomial there are $\frac{(2^n-1)}{\text{per}(f(x))}$ sequences with period equal to $\text{per}(f(x))$ and one sequence with period 1, that is the zero sequence. An example is useful to well understand this theorem.

Example 2.4. Considering the polynomial $f(x) = x^4 + x^3 + x^2 + x + 1 \in F[x]$ (see also Ex. 2.3), this is irreducible with $\text{per}(f(x)) = 5$. Hence, the number of shift-equivalent class is

$$\frac{2^n - 1}{\text{per}(f(x))} = 3, \quad (n = 4,\ \text{per}(f(x)) = 5),$$

or, equivalently, 3 sequences with period 5, as shown in Tab. 2.1, where each class is denoted by G_i, with $i = 1, 2, 3$. Thus we have

$$G(f) = \{0\} \cup G_1 \cup G_2 \cup G_3.$$

As consequence of Theorem 2.9, we have the following corollary.

Corollary 2.4. *If $f(x)$ is primitive over $GF(2)$ of degree n, then any nonzero $\mathbf{a} \in G(f)$ has period $2^n - 1$ and*

$$G(f) = \left\{D^i\mathbf{a} | 0 \leqslant i \leqslant 2^n - 2\right\} \cup \{0\}.$$

This result clearly means that if $f(x)$ is a primitive polynomial over $GF(2)$ of degree n, then the number of shift-equivalent class in $G(f)$ is always

$$\frac{2^n - 1}{\text{per}(f(x))} = 1$$

because $\text{per}(f(x)) = 2^n - 1$. So, *m*-sequences are defined as follows.

Definition 2.15. *A binary sequence, $\mathbf{a}$, generated by an n-stage LFSR is called a* maximal length *sequence if it has period $2^n - 1$, or, equivalently, if its polynomial, $f(x)$, is primitive, so satisfying the following equation*

$$\text{per}(\mathbf{a}) = \text{per}(f(x)) = \text{ord}(\alpha) = 2^n - 1.$$

where α is a polynomial root in the extension field $GF(2^n)$.

Table 2.1: Shift-equivalent class of $G(f)$.

G_1	G_2	G_3
00011	01010	11110
00110	10100	11101
01100	01001	11011
11000	10010	10111
10001	00101	01111

According to Corollary 2.4 and Def. 2.15, in order to generate an m-sequence of period $2^n - 1$ over $F = GF(2)$ by an LFSR, we only need to select a primitive polynomial over F of degree n as the *characteristic polynomial* (also referred to as *generating polynomial*) of this LFSR sequence. This result is summarized in the following theorem, [53].

Theorem 2.10. *An LFSR generator of a given memory, r, produces a sequence of elements from $GF(2)$, with the largest period, $2^r - 1$, iff its characteristic polynomial is primitive over $GF(2)$.*

2.3.3.1 *M*-Sequence Properties

A large bibliography is provided on m-sequence correlation properties and their applications ([3], [14], [18], [19], [20], [21], [35], [36], [41], and [53]). Therefore, here we just give an overview of the statistical properties of these codes ([14], [41], and [53]).

Property 2.1. *A maximal length sequence,* **a***, contains more ones than zeros. The number of ones is $\frac{1}{2} \cdot (N + 1)$, where $N = \text{per}(\mathbf{a}) = 2^r - 1$ and r is the degree of its characteristic polynomial.*

Property 2.2. *Let $\{a_n\}$ be an m-sequence over $GF(2)$ with linear span r (polynomial degree). Then for any τ, with $\tau \neq 0 \bmod 2^r - 1$, the difference of the m-sequence $\{a_n\}$*

and its τ-shift $\{a_{n+\tau}\}$ is another shift $\{a_{n+\tau'(\tau)}\}$ of the same m-sequence. That is

$$a_{n+\tau'(\tau)} = a_{n+\tau} \oplus a_n \quad \forall n,$$

where $\tau'(\tau)$ is defined for all $\tau \neq 0 \bmod 2^r - 1$ and $\oplus$ is modulo-2 sum. This property is referred to as shift-and-add property.

Property 2.3. *If a window of width r is slid along the m-sequence (with period $N = 2^r - 1$) for N shifts, each r-tuple, except the all zero r-tuple, appears exactly once.*

In data communication, binary sequences are often mapped on *polar* values (or BPSK values)

$$y_k = (-1)^{a_k}, \quad \forall k \in \mathbb{N} \quad \text{and} \quad a_k \in GF(2) = \{0, 1\}.$$

We introduce now the *periodic autocorrelation function* of the sequence **y** as

$$R(k) = \frac{1}{N} \cdot \left(\sum_{i=1}^{N} y_{i+k} \cdot y_i^* \right) \tag{2.11}$$

where N is the period. It is easy to show the following correlation properties of m-sequences.

Property 2.4. *The period autocorrelation function $R(k)$ of a BPSK-mapped m-sequence (with period N) is two-valued and is given by*

$$R(k) = \begin{cases} 1, & k = l \cdot N \\ \frac{-1}{N}, & k \neq l \cdot N \end{cases}$$

where $l \in \mathbb{Z}$[7].

Nevertheless, in realistic contexts, correlation calculations are carried out over M

[7] $\mathbb{Z}$ is the set of integers.

symbols, such that

$$r \ll M \ll N = 2^r - 1$$

where r is the polynomial degree and N is the sequence period. So, the *partial-period correlation* is defined as ([41] and [53])

$$r(M, n, \tau) = \frac{1}{M} \cdot \left(\sum_{i=0}^{M-1} y_{n+i+\tau} \cdot y_{n+i}^{*} \right).$$

The partial-period correlation value depends on the initial location, n, in the sequence, where the correlation computation begins as well as on the window length M. Therefore, this correlation function is statistically characterized by its first and second time-average moments, that are denoted by $\langle . \rangle$ and are calculated as follows respectively, [53]

$$\langle r(M, n, \tau) \rangle = \frac{1}{N} \cdot \left(\sum_{n=1}^{N} r(M, n, \tau) \right)$$
$$\langle |r(M, n, \tau)|^2 \rangle = \frac{1}{N} \cdot \left(\sum_{n=1}^{N} |r(M, n, \tau)|^2 \right)$$

where N is the sequence period. Hence, the following property is satisfied ([41] and [53]).

Property 2.5. *Let* **y** *be a BPSK-mapped m-sequence of period N, then the first and second time-average moments of the partial-period correlation function of* **y** *are given by*

$$\langle r(M, n, \tau) \rangle = \begin{cases} \frac{-1}{N}, & \tau \neq 0 \bmod N \\ 1, & \tau = 0 \bmod N \end{cases}$$
$$\langle |r(M, n, \tau)|^2 \rangle = \begin{cases} \frac{1}{M} \cdot \left(1 - \frac{M-1}{N}\right), & \tau \neq 0 \bmod N \\ 1, & \tau = 0 \bmod N \end{cases}$$

for $M \leqslant N$.

Finally, an important class of m-sequences is characterized by generating polynomials

with only three coefficients

$$f(x) = x^n + x^k + 1, \quad \text{with} \quad n, k \in \mathbb{N}^* \quad \text{and} \quad n > k$$

referred to as *trinomials*. In this case, their implementation is very simple, because the LFSR generator is made up of a n-stage SR and a single exclusive-or gate (typically denoted by $\oplus$). Thus, it is more efficient to use a primitive trinomial for generation of an m-sequence than to use generating polynomials characterized by more nonzero coefficients.

2.4 Gold Codes

The main application of SS systems is to perform CDMA, so sharing the scarce channel resources. This is achieved by associating a code to each user. More in detail, each signal is spread by a pre-assigned code that identifies just one particular user. So, the overall SS signal is ideally a combination of all these spread signals and the *Additive White Gaussian Noise* (AWGN). In this way, all users can transmit/receive simultaneously using the same band of frequencies.

At the receiver side, a despreading operation is necessary to track and process the desired user signal. This operation is basically performed by a correlation between the incoming signal and local replicas of the desired spreading code. Of course, in this context, all other spread signals will not be despread and will cause interference in the considered signal (or user channel). Therefore, the main goal of a SS system designer, for a multiple-access system, is to search a set of spreading codes such that as many users as possible can share a band of frequencies with the minimum mutual interference.

The amount of interference from a user employing a different spreading code is related to the cross-correlation between the two different codes, thus the Gold codes[8], [18]

[8]Gold codes were invented in 1967 at the Magnavox Corporation.

and [19], were specifically introduced for multiple-access applications of SS systems. Relatively large sets of Gold codes exist which have well controlled cross-correlation properties. Furthermore, the large difference between the in-phase autocorrelation function $2^N - 1$ (where N is the sequence period) and the cross-correlation function of any two Gold sequences makes these codes useful in CDMA communication systems.

The full period cross-correlation between two spreading codes, a_n and b_n, with period N is defined

$$C(k) = \frac{1}{N} \cdot \sum_{n=0}^{N-1} a_{n+k} \cdot b_n^*. \tag{2.12}$$

Roughly speaking, this is a list of all possible correlation values $C(k)$ as a function of the temporal index k that yields that particular cross-correlation figure. In the case $a_n = b_n$ the cross-correlation coincides with the periodic autocorrelation and (2.12) becomes (2.11).

Definition and properties of Gold codes are reported in the following section. Further details can be found in [3], [18], [20], [41], and [52].

2.4.1 Definition and Properties

The Gold codes are large families of linear binary sequences with uniformly low cross-correlation values. More specifically, in [18] and [19], Gold described a class of pairs of m-sequence whose cross-correlation function have three low values which can be determined precisely.

Consider an m-sequence that is represented by a binary vector $\mathbf{a}$ of period N, and a second sequence $\mathbf{b}$ obtained sampling every q^{th} symbol of $\mathbf{a}$. The second sequence is said to be a decimation of the first, and the notation $\mathbf{b} = \mathbf{a}[q]$ is used to indicate that $\mathbf{b}$ is obtained sampling every q^{th} symbol of $\mathbf{a}$. The decimation of an m-sequence may or may not yield another m-sequence. When the decimation yields an m-sequence, the decimation is said to be a *proper decimation*. It has been proven in [52] that $\mathbf{b} = \mathbf{a}[q]$ has period N if and only if $\gcd(N, q) = 1$ (where gcd is the greatest common divisor),

and the proper decimation by odd integers, q, will give all the m-sequences of period N. Thus, any pair of m-sequences having the same period N, can be related by $\mathbf{b} = \mathbf{a}[q]$ for some q.

The cross-correlation of pairs of m-sequences computed using (2.12) can be three-valued, four-valued, or many-valued. In particular, certain special pairs of m-sequences, whose cross-correlation is three-valued (see also [41])

$$C(k) \in \left\{ -\frac{1}{N}\, t(r), -\frac{1}{N}, \frac{1}{N}\, [t(r) - 2] \right\},$$

where

$$t(r) = \begin{cases} 1 + 2^{(r+1)/2}, & \text{for } r \text{ odd} \\ 1 + 2^{(r+2)/2}, & \text{for } r \text{ even} \end{cases}$$

the code period is $N = 2^r - 1$, and r is their characteristic polynomial degree, are called *preferred pairs* of m-sequences. Finding preferred pairs is necessary to define a set of Gold codes. So, we give the following three conditions that are sufficient to define a preferred pair of m-sequences.

1. $r \neq 0 \bmod 4$, this means that r is odd or $r = 2 \bmod 4$;

2. $\mathbf{b} = \mathbf{a}[q]$ where q is odd and either $q = 2^k + 1$ or $q = 2^{2 \cdot k} - 2^k + 1$;

3. $\gcd(r, k) = \begin{cases} 1, & \text{for } r \text{ odd} \\ 2, & \text{for } r = 2 \bmod 4 \end{cases}$.

More details and an example to find a preferred pair of m-sequences are shown in [41].

Gold sequences are clearly defined by the following theorem, [3].

Theorem 2.11. *Let $f(x)$ and $g(x)$ be a preferred pair of primitive polynomials over GF (2) of degree r, $r \neq 0 \bmod 4$. The LFSR, with characteristic polynomial $f(x) \cdot g(x)$, will generate a set of $2^r + 1$ different sequences of period $N = 2^r - 1$. Any pair of*

sequences in this set has a three-valued cross-correlation, that is

$$C(k) \in \left\{ \frac{-1}{N}\, t(r), \frac{-1}{N}, \frac{1}{N}\, [t(r)-2] \right\}. \tag{2.13}$$

These sequences are commonly called Gold codes.

Indeed, let **a** and **b** represent a preferred pair of m-sequences having period $N = 2^r - 1$. The family of codes

$$\left\{ \mathbf{a}, \mathbf{b}, \mathbf{a} + D\mathbf{b}, \mathbf{a} + D^2\mathbf{b}, \ldots, \mathbf{a} + D^{N-1}\mathbf{b} \right\}$$

is called the set of Gold codes for the preferred pair **a** and **b**. The notation $D^j\mathbf{b}$ represent the cyclic shift of the m-sequence **b** by j units (or equivalently j *chips*). As shown in Theorem 2.11, any set of Gold codes has the property that any pair of sequences in the set have a three-valued cross-correlation which takes on the values defined in (2.13). Furthermore, the number of sequences in any family of Gold codes is $2^r + 1$.

Example 2.5. A typical implementation used to generate Gold codes is illustrated in Fig. 2.2. More in detail, the picture shows the configuration used to obtain the GPS/SBAS Gold code set. The two m-sequences are

$$\begin{aligned} P_{c'}(D) &= D^{10} + D^3 + 1 \\ P_{c''}(D) &= D^{10} + D^9 + D^8 + D^6 + D^3 + D^2 + 1 \end{aligned}$$

their degree is $r = 10$, so all the sequences of this set have period $N = 2^r - 1 = 1023$ and the total number of codes is $2^r + 1 = N + 2 = 1025$. The equivalent LFSR generator is characterized by the following polynomial of higher degree

$$\begin{aligned} P(D) &= P_{c'}(D) \cdot P_{c''}(D) \\ &= D^{20} + D^{19} + D^{18} + D^{16} + D^{11} + D^8 + D^5 + D^2 + 1 \end{aligned}$$

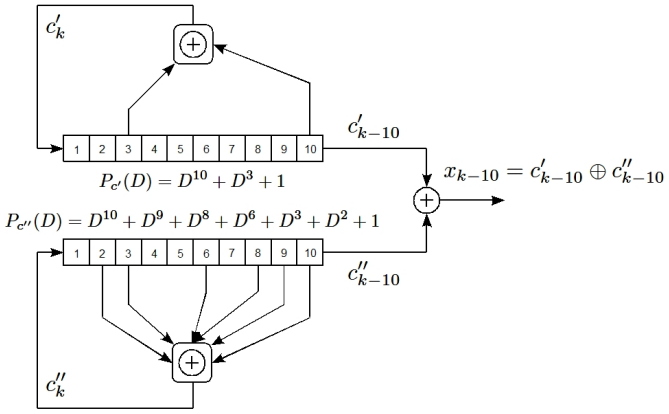

Figure 2.2: GPS/SBAS Gold code generator.

that provides the tap configuration of the LFSR generator of this Gold code set. This second implementation can also be used to generate the same set of $N + 2$ sequences, simply changing the initial word of its SR.

Chapter 3

Signal Model and Detection Algorithms

This chapter provides a mathematical description of the communication system and the signal model that will be considered to compare the iMP detector to the standard correlation-based acquisition algorithms (full-parallel, hybrid, and simple-serial searches). Furthermore, the second part of this chapter gives an overview of the standard detectors and presents the motivations that allow the application of MP algorithms to acquire spreading sequences. Thus, the outline is below.

- **Section 3.1** contains a description of a simplified DS/SS communication system, including a mathematical characterization of the signal model.

- **Section 3.2** is an overview of detection algorithms typically used to acquire SS codes. It also contains an introduction to MP algorithms and their application as detectors.

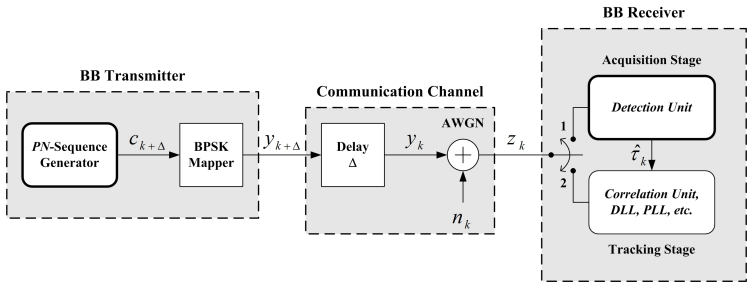

Figure 3.1: DS/SS communication system model.

3.1 Communication System

This section gives an overview of the communication system model that will be used to evaluate the performance of iMP algorithms to acquire PN sequences. The base-band (BB) representation of a DS/SS communication system is reported in Fig. 3.1. It is made up of: a BB transmitter, that outputs a predetermined spreading sequence[1], a channel that introduces a propagation delay ($\Delta = 0$), an additive white gaussian noise (AWGN), and finally a BB receiver that runs two important stages: *acquisition* and *tracking* stage. The first one is addressed to detect a SS signal, performing a rough estimation of its code delay. Then, the tracking stage exploits this coarse synchronization to run a DLL that improves the alignment between the transmitted spreading sequence and its local replica. This step is fundamental to avoid catastrophic SNR degradation and correctly process the incoming signal. The following sections provide more details on each stage of the proposed communication system model (Fig. 3.1).

3.1.1 Base-Band Transmitter

The BB transmitter is basically made up of a PN sequence generator that produces pseudo-random binary sequences, $\mathbf{c}$ (each element is $c_k \in \{0, 1\}$) and a BPSK mapper that outputs

[1] Only LFSR sequences are considered.

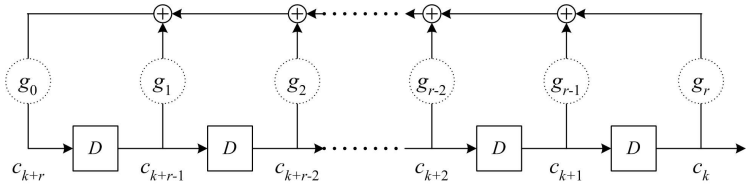

Figure 3.2: General representation of an r-stage LFSR generator.

the correspondent antipodal sequences, $\mathbf{y}$, where each component is $y_k = (-1)^{c_k}$. Only LFSR generators are taken into account in this thesis. A general representation of an LFSR generator is in Fig. 3.2. As shown in the picture, at the generic time k, assuming that c_k is the SR output and c_{k+i} (with $0 \leqslant i \leqslant r$) is the content of the i^{th} register, the following parity equation is verified

$$\begin{aligned} 0 &= g_r \cdot c_k \oplus g_{r-1} \cdot c_{k+1} \oplus g_{r-2} \cdot c_{k+2} \oplus \dots \\ &\quad \oplus g_2 \cdot c_{k+r-2} \oplus g_1 \cdot c_{k+r-1} \oplus g_0 \cdot c_{k+r} \\ &= \bigoplus_{i=0}^{r} g_{r-i} \cdot c_{k+i} \end{aligned} \tag{3.1}$$

where $\oplus$ is modulo-2 addition and $g_i \in \{0, 1\}$, $0 \leqslant i \leqslant r$, are the feedback coefficients (also referred to as *taps*). The most common way to represent an r-stage LFSR is providing its generating polynomial (that also gives the tap configuration of the code) as

$$\begin{aligned} P(D) &= g_0 + g_1 \cdot D + \dots + g_{r-1} \cdot D^{r-1} + g_r \cdot D^r \\ &= \sum_{i=0}^{r} g_i \cdot D^i \end{aligned} \tag{3.2}$$

where D is the unit delay operator[2], and r is the polynomial degree. For a given degree r, g_0 and g_r are always 1.

After the detailed introduction of m-sequences and Gold codes performed in Chapter

[2]It is mathematically defined left-shift operator (see Chap. 2 and [20]).

2, we highlight here that there also exits an equivalent way for producing Gold sequences using a single higher-order LFSR generator. Indeed, as demonstrated in [18], a Gold code can be generated by an r-stage LFSR unit (the scheme is in Fig. 3.2) with the tap configuration

$$P(D) = P_{c'}(D) \cdot P_{c''}(D) \tag{3.3}$$

where $P_{c'}(D)$ and $P_{c''}(D)$ are the primitive polynomials that specify the feedback connections of the two q-stage SRs, where $q = \frac{r}{2}$, that output the generating m-sequences $\mathbf{c}'$ and $\mathbf{c}''$ (as shows in Fig. 2.2).

Example 3.1. The two m-sequence polynomials of GPS/SBAS Gold sequences are

$$P_{c'}(D) = D^{10} + D^3 + 1 \tag{3.4a}$$

$$P_{c''}(D) = D^{10} + D^9 + D^8 + D^6 + D^3 + D^2 + 1 \tag{3.4b}$$

$q = 10$ implies $r = 2 \cdot q = 20$. From (3.3), the high-order LFSR (or Gold generating polynomial) is

$$P(D) = D^{20} + D^{19} + D^{18} + D^{16} + D^{11} + D^8 + D^5 + D^2 + 1 \tag{3.5}$$

this important result allows Gold codes to be treated as LFSR sequences.

Note that typically the equivalent LFSR for Gold sequences (e.g., Eq. (3.5) for GPS/SBAS codes) is not a *sparse* (*dense*) generator, because it has more then 4 coefficients.

3.1.2 Communication Channel

Referring to Fig. 3.1, the incoming BB spreading signal at the receiver side is found to be

$$z_k = \sqrt{E_c} \cdot y_k + n_k = \sqrt{E_c} \cdot (-1)^{c_k} + n_k \tag{3.6}$$

where z_k is the noisy sample received by detection unit at time $k\ T_c$ (T_c is the chip time), y_k is the antipodal modulation of the spreading sequence chip c_k (N is the sequence period), and n_k is an additive white gaussian noise (AWGN) with mean value 0 and variance $\frac{N_0}{2}$. No data modulation is shown, since we are assuming to acquire a *pilot* signal with *coherent* detection. This is admittedly a simplified representation, that we use here to "isolate" the issue we are concerned with as is customary done in the spread-spectrum literature (see also [10], [46], [47], [53], [59], [61], and [62]).

3.1.3 Base-Band Receiver

As mentioned, synchronization between the transmitter and the receiver is the key point to guarantee the correct functioning of a DS/SS communication system. More in detail, this synchronization is achieved when, at the receiver side, the transmitted code is aligned with its local replica, in such a way that it is possible to carry out despreading and then process the data. Of course, any misalignment can cause catastrophic degradations of the SNR, making fruitless the next data-processing.

As depicted in Fig. 3.1, to get a fine synchronization two receiver stages are necessary: the *acquisition* and *tracking* stages. The acquisitions stage is addressed to detect the incoming sequence performing a preliminary rough estimation of its code phase. This operation is generally performed by a *detection unit* that correlates the received signal with local replicas of its PN sequence. More specifically, the search is typically carried out shifting the local code until the maximum correlation peak is got or a fixed threshold is crossed. When this happens, the incoming sequence is acquired and the receiver goes into a *verification mode* ([26], [27], [44], [46], and [53]) that checks the correct alignment. This operation is commonly done executing a longer correlation. In both cases, the probability to a have a wrong decision during the verification mode can be neglected. Of course, if the test is not passed, a new acquisition try is carried out.

Assuming that the incoming sequence is acquired, the tracking stage is run. An ex-

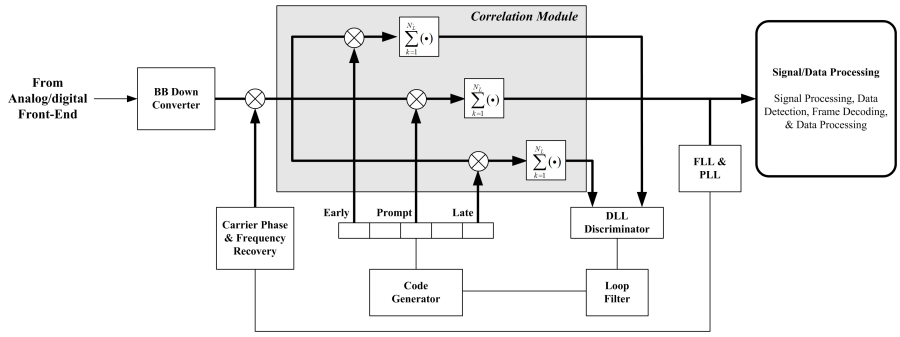

Figure 3.3: An example of a tracking stage in a DS/SS receiver.

ample of a typical architecture of a DS/SS receiver during this stage is shown in Fig. 3.3 (see [5], [11], [15], and [48]). During the tracking phase, the previous coarse synchronization[3] is exploited by a DLL to improve the code phase estimation and ultimately lock the received spreading code. In this way, the *correlation module* (Fig. 3.3) performs the despreading operation without any appreciable degradation of SNR, so recovering all code gain ([39], [53], and [54]). The estimations of carrier frequency and phase offsets are carried out by a Phase/Frequency Locked-Loop (PLL and FLL), [15], [39], and [48].

3.2 Detection Unit

This section contains an overview of the standard acquisition algorithms, used to detect SS sequences, and finally introduces the iMP detector. More specifically, full-parallel, hybrid, and simple-serial searches are described and characterized in terms of *correct detection* (CD), *missed detection* (MD), and *false alarm* (FA) probabilities ([32], [41], [44], and [53]), algorithm complexity, and acquisition time performance ([26], [27], and [44]). Then, the last part of this section gives the motivations that allow to use iMP-based

[3]Typically, the estimation error of a generic code phase is in modulo less than $T_c/2$, where T_c is the chip time. With this assumption, the estimation error is contained in the acquisition range of the DLL *S curve*, so guaranteeing the correct functioning of this device (see also [12], [13], [27], [28], [39], [53], and [54]).

algorithms to detect PN sequences, and a brief description of this new kind of detection unit is presented.

3.2.1 Single-Dwell Acquisition Algorithms

Correlation-based detectors, typically used to acquire PN sequences, are widely studied in literature: [7], [26], [27], [32], [41], [44], [53], [55], [57], and [64]. Therefore, this section just provides an overview of these techniques, showing their implementation and performance.

The common characteristic of all standard detectors is the correlation between the received signal and local replicas of the transmitted PN sequence. Thus, considering the assumption of a coherent pilot channel, that is done in Section 3.1.2, and the mathematical representation of the incoming signal, (3.6), a simplified representation of one correlation branch of an acquisition unit is shown in Fig. 3.4. In particular, the integration time is defined *dwell time* and its value is

$$\tau_d = M \cdot T_c \tag{3.7}$$

where M is the number of observations and T_c is the chip time. These algorithms are called *single-dwell* because they are characterized by just one integration stage (Fig. 3.4) with respect to the *multiple-dwell* algorithms in which more integration stages are sequentially performed in order to improve the performance (more details are given in [53]).

The code-phase search is carried out shifting the local code until a rough[4] alignment between the transmitted sequence and its local replica is achieved. When this happens, a suitable decision unit should detect the correlation peak and run a verification mode to check if a correct detection has been got ([27], [44], [46], and [53]). This last step is fundamental to avoid that a wrong decision which could cause huge delays due to a tracking stage that tries to process a misaligned incoming signal.

[4]Genarally, it means an error in modulo less then one-half chip.

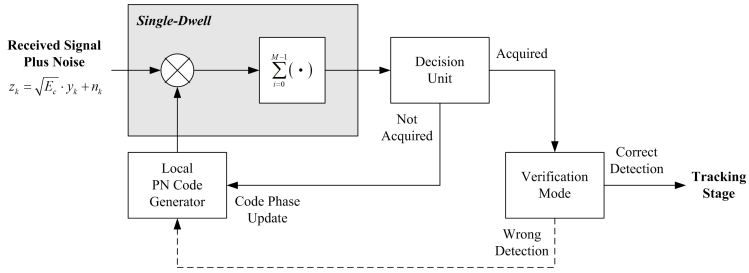

Figure 3.4: A general design of a coherent single-dwell detector.

The verification mode is typically implemented via a long correlation. In other wordS, the incoming signal is first despread and then the correlation is compared to a preset threshold. Of course, such solution adds a further delay to the acquisition time, that is called *penalty time*, τ_{pt}. Generally, this figure is assumed to be (using the dwell time definition (3.7))

$$\tau_{pt} \triangleq k \cdot \tau_d = k \cdot M \cdot T_c \tag{3.8}$$

where $k \in \mathbb{N}^*$ (integer larger than 0), and its value is a characteristic parameter of the receiver that depends on the implementation of the verification stage.

There are three main architectures to perform the code delay (also called *code phase*) search. Those are: full-parallel, hybrid, and simple-serial searches. More details on those algorithms are reported in the following sections.

3.2.1.1 Full-Parallel Search

The full-parallel search carries out the ML estimation of the code phase through an exhaustive search over all possible code delays, yielding the estimate

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}_i} \left[p\left(\mathbf{z}|\mathbf{y}_i\right) \right], \quad \text{with} \quad i = 0, 1, \ldots, N-1 \tag{3.9}$$

where $p(\mathbf{z}|\mathbf{y}_i)$ is the likelihood of $\mathbf{y}_i$ and $\mathbf{z}$ that are two vectors, respectively, made up of M chips[5], y_k, and M soft observations, z_k, as defined in (3.6).

Consider the coherent pilot channel reported in Fig. 3.1 and let N be the period of the transmitted PN sequence. The architectural design of a full-parallel detector is as in Fig. 3.5, where the number of correlation branches (also referred to as *fingers*) B_{FP} is N. Each finger is univocally associated to a code phase of the local replica of the considered PN sequence. The decision unit, selecting the finger with the maximum correlation figure, chooses the correspondent code phase estimation. This is in agreement with the ML estimate algorithm (3.9).

Assuming to process M observations (dwell time $\tau = M\ T_c$), the complexity C_{FP} is computed as the total number of sum operators per acquisition try

$$C_{FP} \cong B_{FP} \cdot M = N \cdot M \tag{3.10}$$

where $B_{FP} = N$ is the sequence period. In case of an m-sequence, the period depends on the degree, r, of its primitive polynomial. Thus, $N = 2^r - 1$. So, using (3.10), the complexity becomes

$$C_{FP} \cong N \cdot M = (2^r - 1) \cdot M.$$

The probabilities of correct and wrong acquisition, respectively called $\mathcal{P}_{CD}$ and $\mathcal{P}_{WD}$, for full parallel search are computed approximately using the model in (3.6). The results are ([10], [41], and [53])

$$\mathcal{P}_{CD} = \int_{-\infty}^{+\infty} \left[1 - \mathrm{Q}\left(\frac{w + \sqrt{2 \cdot M \cdot (E_c/N_0)}}{\sqrt{2 \cdot (E_c/N_0) + 1}}\right)\right]^{N-1} \cdot \frac{e^{-\frac{w^2}{2}}}{\sqrt{2 \cdot \pi}}\, dw$$

$$\mathcal{P}_{WD} = 1 - \mathcal{P}_{CD}$$

where $\mathrm{Q}(.)$ is the complementary cumulative distribution function of a standard Gaussian

[5] With an antipodal modulation, $y_k = (-1)^{c_k}$.

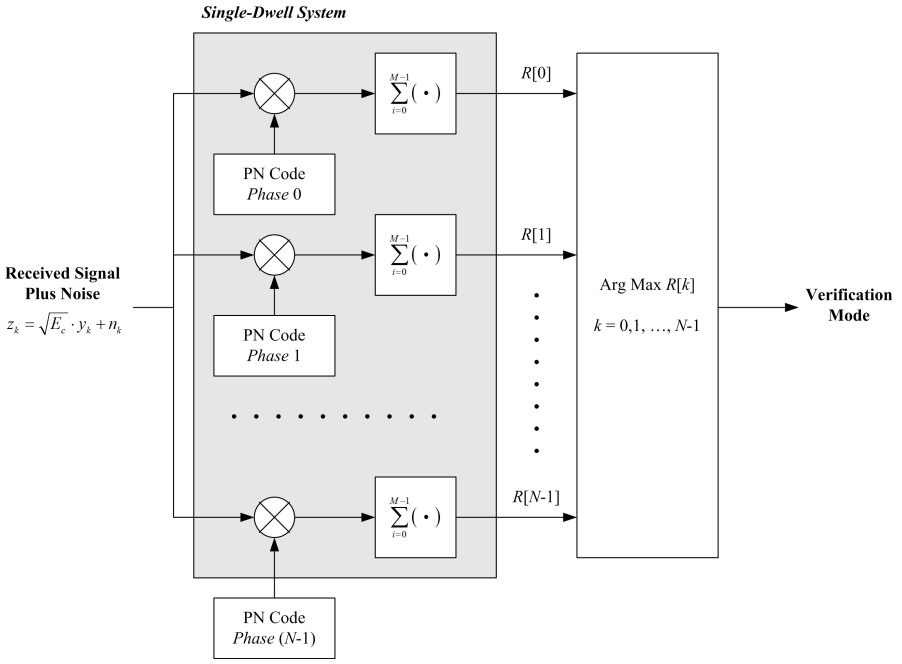

Figure 3.5: A simplified scheme of a full-parallel algorithm.

random variable.[6]

Finally, the mean and variance of its acquisition time can be evaluated using the following equation (see also [46])

$$\frac{\mu_{FP}}{T_c} = \frac{k+1}{\mathcal{P}_{CD}} \cdot M \tag{3.11a}$$

$$\frac{\sigma^2_{FP}}{T_c^2} = \frac{(k+1)^2}{\mathcal{P}^2_{CD}} \cdot M^2 \cdot (1 - \mathcal{P}_{CD}) \tag{3.11b}$$

The proof is easy got following the analysis in [26], [27], and [44].

[6]The Q-function is defined

$$\mathrm{Q}(x) \triangleq \int_x^{+\infty} \frac{e^{-\alpha^2/2}}{\sqrt{2\pi}}\, d\alpha.$$

3.2.1.2 Simple-Serial Search

With respect to the full-parallel search, the simple-serial search is the opposite solution. Indeed, it is characterized by one correlation branch (one finger) and the right alignment is obtained shifting the local code until the correlation values crosses a prefixed threshold. Its block scheme is shown in Fig. 3.6.

As in the previous section, assuming to collect M observations, the complexity of a simple-serial search, C_{SS}, is the total number of sum operators per acquisition try

$$C_{SS} \cong M. \tag{3.12}$$

The probabilities of correct, missed detection and false alarm, $\mathcal{P}_{CD}$, $\mathcal{P}_{MD}$, and $\mathcal{P}_{FA}$ respectively, for simple-serial search are

$$\begin{aligned}
\mathcal{P}_{CD} &= \mathrm{Q}\left(\left[\lambda - \sqrt{E_c}\right] \cdot \sqrt{\frac{2 \cdot M}{E_c} \cdot (E_c/N_0)}\right) \\
\mathcal{P}_{MD} &= 1 - \mathcal{P}_{CD} \\
\mathcal{P}_{FA} &= \mathrm{Q}\left(\lambda \cdot \sqrt{\frac{2 \cdot M \cdot (E_c/N_0)}{E_c \cdot [2 \cdot (E_c/N_0) + 1]}}\right).
\end{aligned}$$

where λ is the threshold, E_c/N_0 is the SNR, and E_c is the chip energy. These results are easily proved following the guidelines reported in [10], [44], and [53].

Finally, defining q as the number of cells, [26], to be searched (of course, it depends on sequence period, N, and the search step[7] of a simple-serial search), the mean and variance of the simple-serial acquisition time can be evaluated using the following equation (see

[7] The search step is typically one or one-half chip.

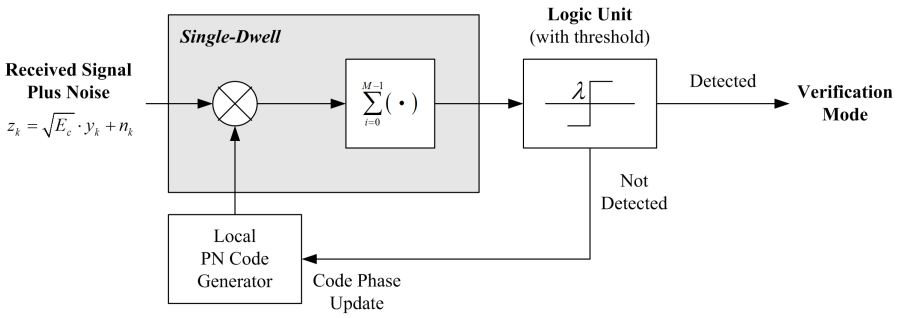

Figure 3.6: A simplified scheme of a simple-serial algorithm.

also [46])

$$\frac{\mu_{SS}}{T_c} = \frac{2 \cdot (k \cdot \mathcal{P}_{CD} + 1) + (q-1) \cdot (k \cdot \mathcal{P}_{FA} + 1) \cdot (2 - \mathcal{P}_{CD})}{2 \cdot \mathcal{P}_{CD}} \cdot M \tag{3.13a}$$

$$\frac{\sigma^2_{SS}}{T_c^2} \approx M^2 \cdot q^2 \cdot (k \cdot \mathcal{P}_{FA} + 1)^2 \cdot \left[\frac{1}{12} - \frac{1}{\mathcal{P}_{CD}} + \left(\frac{1}{\mathcal{P}_{CD}} \right)^2 \right] \tag{3.13b}$$

where $q \gg 1$ and $q \gg k \cdot (k \cdot \mathcal{P}_{FA} + 1)$. The proof is reported in [26], adding the penalty time (3.8) of the verification stage to both cases of correct detection and false alarm.

3.2.1.3 Hybrid Search

The hybrid search is a trade-off between the two previous algorithms. Its general design is shown in Fig. 3.7. It is characterized by B_H correlation branches (where $1 < B_H < N$ and N is the sequence period) each one associated with a particular code phase of the local code. The decision unit selects the maximum correlation value that crosses a prefixed threshold, λ (where $\lambda > 0$), otherwise all local replicas of the sequence are shifted.

Assuming to collect M observations, the complexity of a hybrid search, C_H, is the total number of sum operators per acquisition try

$$C_H \cong M \cdot B_H. \tag{3.14}$$

Of course

- if $B_H = B_{FP} = N$, where N is the sequence period, then $C_H = C_{FP}$;
- if $B_H = 1$, then $C_H = C_{SS}$.

Defining

$$\alpha \triangleq \mathrm{Q}\left(\left[\lambda - \sqrt{E_c}\right] \cdot \sqrt{\frac{2 \cdot M}{E_c} \cdot (E_c/N_0)}\right)$$
$$\beta(\nu) \triangleq \mathrm{Q}\left(\frac{\nu + \sqrt{2 \cdot M \cdot (E_c/N_0)}}{\sqrt{2 \cdot (E_c/N_0) + 1}}\right)$$
$$\xi \triangleq \mathrm{Q}\left(\lambda \cdot \sqrt{\frac{2 \cdot M \cdot (E_c/N_0)}{E_c \cdot [2 \cdot (E_c/N_0) + 1]}}\right)$$

the probabilities of correct, missed, wrong detection, and false alarm, $\mathcal{P}_{CD}$, $\mathcal{P}_{MD}$, $\mathcal{P}_{WD}$, and $\mathcal{P}_{FA}$[8] respectively, for the hybrid search are

$$\mathcal{P}_{CD} = \alpha \cdot \int_{-\infty}^{+\infty} [1 - \beta(\nu)]^{B_H - 1} \cdot \frac{e^{-\frac{\nu^2}{2}}}{\sqrt{2 \cdot \pi}}\, d\nu \tag{3.15a}$$

$$\mathcal{P}_{MD} = [1 - \alpha] \cdot [1 - \xi]^{B_H - 1} \tag{3.15b}$$

$$\mathcal{P}_{WD} = 1 - \mathcal{P}_{CD} - \mathcal{P}_{MD} \tag{3.15c}$$

$$\mathcal{P}_{FA} = \xi \cdot \int_{-\infty}^{+\infty} [1 - \mathrm{Q}(\nu)]^{B_H - 1} \cdot \frac{e^{-\frac{\nu^2}{2}}}{\sqrt{2 \cdot \pi}}\, d\nu. \tag{3.15d}$$

The proof can be achieved following the considerations reported in [44], [53], and [64]. It is easy to check that if $\lambda \to -\infty$ and $B_H = N$ then the hybrid performance tends to full-parallel one, and if $B_H = 1$ then the hybrid tends to simple-serial.

Finally, defining q as the number of cells to be searched, the mean and variance of its

[8] We assume to have a *wrong detection*, when one finger is aligned and a different finger is elected to be synchronized. While *false alarm* happens when none local replica is synchronized with the incoming sequence, but the threshold is crossed by a correlation peak. Thus, the finger, which generates that peak, is considered synchronized.

acquisition time can be evaluated using the following equation (see also [46])

$$\frac{\mu_H}{T_c} = \frac{[2\,(k\cdot\mathcal{P}_{CD}+1)+(q-1)\,(k\cdot\mathcal{P}_{FA}+1)\,(2-\mathcal{P}_{CD})+2\cdot k\cdot\mathcal{P}_{WD}]\;M}{2\cdot\mathcal{P}_{CD}} \tag{3.16a}$$

$$\frac{\sigma_H^2}{T_c^2} \approx \frac{\mu_H}{T_c} + \Gamma_1 + \Gamma_2 + \Gamma_3 \tag{3.16b}$$

where

$$\begin{aligned}
\Gamma_1 &\triangleq -M\cdot(k+1)\\
\Gamma_2 &\triangleq \frac{M}{\mathcal{P}_{CD}}\,\{(M-1)\,(1-\mathcal{P}_{CD})+k\,(M\,k+2\,M-1)\,\mathcal{P}_{WD}\}\\
&\quad+\frac{M}{\mathcal{P}_{CD}}\,\{[M-1+k\,(k\,M+2\,M-1)\,\mathcal{P}_{FA}]\,(1-\mathcal{P}_{CD})\,(q-1)\}\\
&\quad+\frac{M}{\mathcal{P}_{CD}}\,\{2\,M\,(q-1)\,(1+k\,\mathcal{P}_{FA})\,(1-\mathcal{P}_{CD}+k\,\mathcal{P}_{WD})\}\\
&\quad+\frac{M}{\mathcal{P}_{CD}}\,\left\{M\,(q-1)\,(q-2)\,(1-\mathcal{P}_{CD})\,(1+k\,\mathcal{P}_{FA})^2\right\}\\
&\quad+\left(\frac{M}{\mathcal{P}_{CD}}\right)^2\,\{k\,[\mathcal{P}_{WD}+\mathcal{P}_{FA}\,(1-\mathcal{P}_{CD})\,(q-1)]+q\,(1-\mathcal{P}_{CD})\}^2\\
\Gamma_3 &\triangleq \frac{1}{12}\,M\,(q-1)\,\left\{-6\,(1+k\,\mathcal{P}_{FA})+M\,\left[1+k^2\,\mathcal{P}_{FA}\,(6+\mathcal{P}_{FA}\,(q-5))\right]\right\}\\
&\quad+\frac{1}{12}\,M\,(q-1)\,\{q+2\,k\,\mathcal{P}_{FA}\,(q+1)\}\,.
\end{aligned}$$

3.2.2 Detection with MP-Based Algorithms

As shown in Section 3.2.1, all standard detection algorithms operate by correlating the received signal with shifted local replicas of the incoming PN code, until the right alignment is obtained ([53], [42], and [44]).

A new approach is proposed in [10], [47], [59], and [62], that is based on a generalization of the standard decoding problem. Consider the M-dimensional received vector,

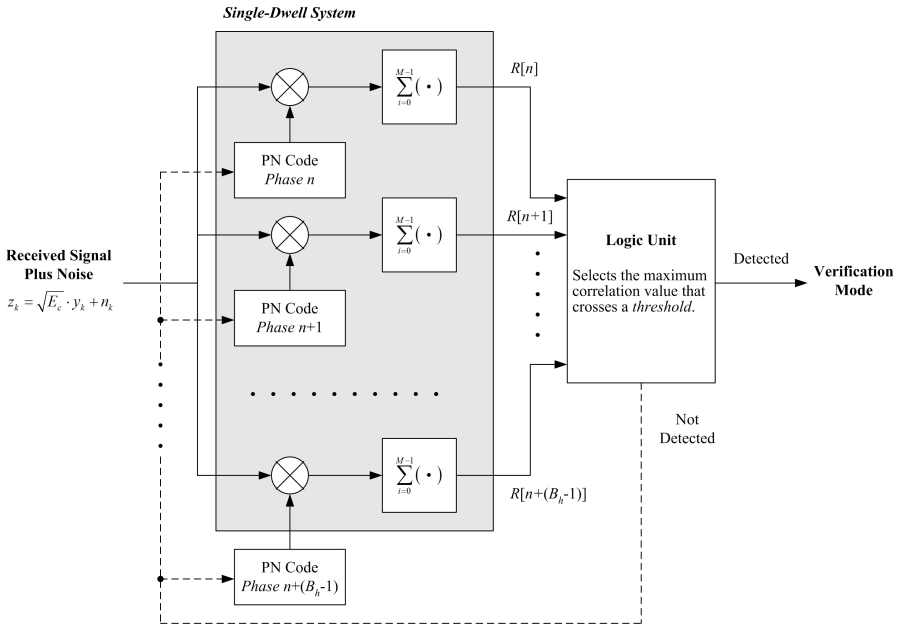

Figure 3.7: A simplified scheme of a hybrid search.

$\mathbf{z} = [z_0, z_1, \ldots, z_{M-1}]$, the ML detection algorithm can be formulated as

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}_i} \left[p\left(\mathbf{z}|\mathbf{y}_i\right) \right], \quad \text{with} \quad i = 0, 1, \ldots, N-1$$

where $\mathbf{y}_i$ is a vector that contains M shifted chips of the transmitted PN sequence, and $p\left(\mathbf{z}|\mathbf{y}_i\right)$ is the likelihood function of $\mathbf{y}_i$ and $\mathbf{z}$. Instead of casting our problem into one of delay estimation, we stick to a detection approach. Indeed, we have to search into a set of N different sequences corresponding to all possible shifts of the considered spreading code. This problem is definitely similar to decoding of a block code, in which the ML decoder selects the codeword $\mathbf{y}_i$ (among a set of N different codewords that could have been transmitted) that maximizes $p\left(\mathbf{z}|\mathbf{y}_i\right)$, [49] and [35]. So, the equivalence between acquisition and decoding is clearly proved.

Following the approach of iterative decoding of modern codes ([4], [6], and [17]), an ML algorithm can be implemented by a MP algorithm run on a graphical model without cycles, called *tree graph*. Unluckily, these optimal algorithms are often too complex to implement, so graphical models with cycles (e.g., Tanner graphs, TGs), [56] and [60], are commonly used. Indeed, these models with cycles yield sub-optimal solutions with lower complexity, and it has been experimentally observed that, iterating the MP on a proper model design, the performance can be close to that of the ML algorithm. These graphical models are, basically, made up of sets of *variable nodes*, directly associated to incoming soft information, and *check nodes*, that identify the parity equations (local constrains) verified by the transmitted code. Unfortunately, a systematic method for designing the best graphical model for a given specified code is not known. Complete treatments on standard MP algorithms are reported in [1], [9], [34], [35], [37], and [60]. Roughly speaking, an iMP algorithm passes soft information between nodes in its graph, and each iteration ends when all nodes are activated. Hence, in order to correctly implement an iMP algorithm, one must generate a *graphical model*, on which this algorithm is run, define its *activation schedule*, which is the order that is established to activate all variable and check nodes, including when the algorithm is terminated. Typically, these algorithms end either when their estimated vectors verify all parity checks or when the max number of iteration, I_{MAX}, is obtained. The last step is to select the processing used to perform the *message updating*. As reported in [9], [34], and [60], there are two main algorithms: *Sum-Product* (SP) and *Min-Sum* (MS) algorithms. We will consider the Min-Sum algorithm because it is simpler and does not require an estimate of the operating SNR. So, the key points to configure an iMP algorithm are summarized below.

- *Graphical Model* - that is typically a loopy graph, with a complexity lower than a tree graph.

- *Activation Schedule* - that is the set of rules that, for every iteration, establishes the activation order of all nodes in a graph. It also includes the end condition of the

iMP (e.g. I_{MAX}).

- *Message-Updating Algorithm* - Sum-Product or Min-Sum algorithms.

The next chapter is fully addressed to the iMP detector characterization. In particular, we will give a description of the algorithm, its implementation, and the complexity. We also evaluate its acquisition time performance, using the Markov chain theory ([26], [27], [40], [44], and [46]).

Chapter 4

Message Passing Algorithms to Detect Spreading Codes

This chapter introduces a new procedure to acquire spreading codes based on MP algorithms. Considering that a large bibliography is provided on MP and its applications in modern decoding theory ([1], [4], [6], [9], [17], [34], [35], [37], [56], and [60]), our attention will be mainly focused on acquisition aspects that characterize an iMP detector. More specifically, the outline of the chapter is the following.

- **Section 4.1** contains the architectural design of an iMP detector.
- **Section 4.2** gives a general overview of MP algorithms, focusing on the implementation and evaluation of the complexity of loopy graphs, [47] and [63].
- **Section 4.3** reports an analysis, based on the Holmes seminal work (reported in [26] and [27]), of the acquisition time performance of an iMP detector, [46].
- **Section 4.4** presents algorithms to search trinomial multiples of primitive polynomials that will be used to generate sparse graphical models.

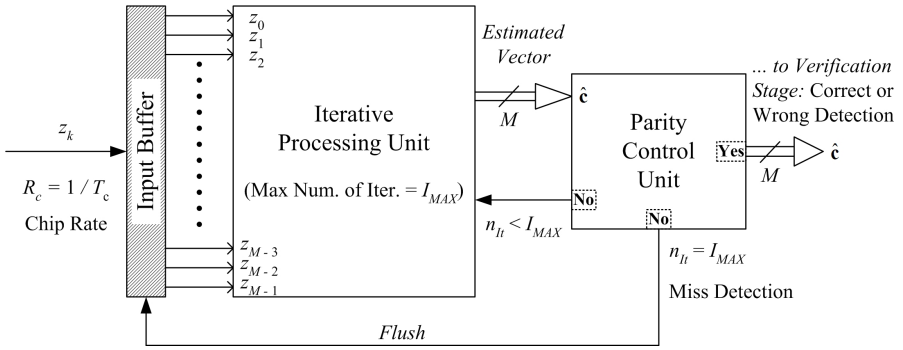

Figure 4.1: Iterative Detection Unit with an iMP algorithm.

4.1 Iterative Detection Unit

This section shows the architectural design of a coherent iMP detector. This unit is also referred to as *iterative detection unit* (iDU). All main phases of the iDU are shown and described, and their time delays are also provided. This last point is fundamental to perform the analysis of the acquisition time (Section 4.3).

4.1.1 Architectural Design

The basic architecture of a coherent iMP detector (also called iDU) is shown in Fig. 4.1. As the picture shows, it is made up of an *input buffer* (IB), to store the received vector $\mathbf{z}$, an *iterative processing unit* (iPU), that runs the iMP algorithm, which is the core of the iDU, and a *parity control unit* (PCU), that stops the acquisition procedure when the estimated binary vector verifies all the parity equations, otherwise a new iteration is carried out until the maximum number of iterations, I_{MAX}, is achieved.

More in detail, considering the communication system shown in Fig. 3.1, a predetermined LFSR sequence is transmitted through an AWGN channel. The mathematical representation of the signal is (3.6). At the receiver side, an incoming vector of M observations, $\mathbf{z}$, (each element, z_k, is characterized in (3.6)) is stored in the IB. So, let T_c

be the chip time of a generic z_k element (the chip rate is $R_c = 1/T_c$), the required time to fill the buffer is $M \cdot T_c$. This array of observations gets in the iPU that runs an iMP algorithm on a predetermined loopy graph (see [9], [34], [37], and [60]). Typically, at the end of each iteration, the iMP algorithm provides a soft-output information vector that is used to estimate the transmitted vector by hard decision, and, soon after, a parity control is executed on it ([47] and [63]). Because of this, the acquisition algorithm can end earlier than I_{MAX} iterations, as soon as the check is positive. Therefore, the time required by iPU, τ_{iPU}, to output a soft-output information vector (produced by the iMP algorithm) and perform a hard-decision on it, is

$$\tau_{iPU} \leqslant \frac{I_{MAX}}{\rho} \cdot T_c$$

where I_{MAX} is the maximum number of iterations of the iMP algorithm, T_c is the chip time, T_{it} is the time per iteration ($R_{it} = 1/T_{it}$ is the iteration rate), and $\rho = T_c/T_{it} = R_{it}/R_c$ is the iMP *time-factor*. Of course, the ρ-factor depends on the implementation technology of the receiver. Nevertheless, to simplify our analysis, we will over-bound the acquisition time, assuming that the iMP algorithm always ends when all I_{MAX} iterations are run. This means

$$\tau_{iPU} = \frac{I_{MAX}}{\rho} \cdot T_c.$$

When all I_{MAX} iterations are run, the estimate vector, $\hat{\mathbf{c}}$, is handed over to the PCU that checks the parity. If the parity control fails, a *missed detection* is achieved and a new received vector is processed by the iDU. In this case, the missed detection time, τ_{MD}, is

$$\tau_{MD} \triangleq m \cdot T_c, \quad \text{with} \quad m = \max\left(M, \frac{I_{MAX}}{\rho}\right). \tag{4.1}$$

In other words, τ_{MD} is the longest time between $\tau_{RB} \triangleq M \cdot T_c$, that is the time to refill the IB, and τ_{iPU}. If the check is passed, we can have either a *correct* or a *wrong acquisition.* Thus, the receiver goes into a verification mode, which may include a long correlation

test and/or a tracking loop. At the end of the verification, it is reasonable to assume that the probability of wrong decision is close to 0 (as mentioned in Section 3.1.3) and the *penalty-time* is $\tau_{pt} \triangleq k \cdot M \cdot T_c$ (see also [26] and [27]). Of course, in the case of wrong detection, a new acquisition try is run, otherwise, in the case of correct acquisition (often referred to as a *hit*), the signal is considered to be detected and tracking is started.

As previously illustrated, one acquisition attempt can end with just one of three mutually-exclusive possible outcomes: missed detection (MD), wrong detection (WD), or correct detection (CD). So, referring to the CD probability as $\mathcal{P}_{CD}$, to the WD probability as $\mathcal{P}_{WD}$, and to the MD probability as $\mathcal{P}_{MD}$, the following equality is verified

$$\mathcal{P}_{CD} + \mathcal{P}_{WD} + \mathcal{P}_{MD} = 1 \implies \frac{\mathcal{P}_{CD}}{1 - \mathcal{P}_{MD}} + \frac{\mathcal{P}_{WD}}{1 - \mathcal{P}_{MD}} = 1. \tag{4.2}$$

Fig. 4.2 summarizes all the main stages of this detection strategy, including their time delays.

4.2 Iterative Message Passing for PN Acquisition

As mentioned in the previous section, the iPU is the core of an iDU. Indeed, this device runs a MP algorithms on predetermined graphical models with cycles. Therefore, it becomes important to define the guidelines to correctly configure a MP algorithm. More in details, three cardinal points have been introduced in Section 3.2.2 and repeated here.

- *Graphical Model* - that is typically a loopy graph, because the complexity is lower than a tree graph.
- *Activation Schedule* - that is the set of rules which establishes, for every iteration, the activation order of all nodes in a graph. It also includes the end condition of a MP algorithm (e.g., I_{MAX}).
- *Message-Updating Algorithm* - Sum-Product (SP) or Min-Sum (MS).

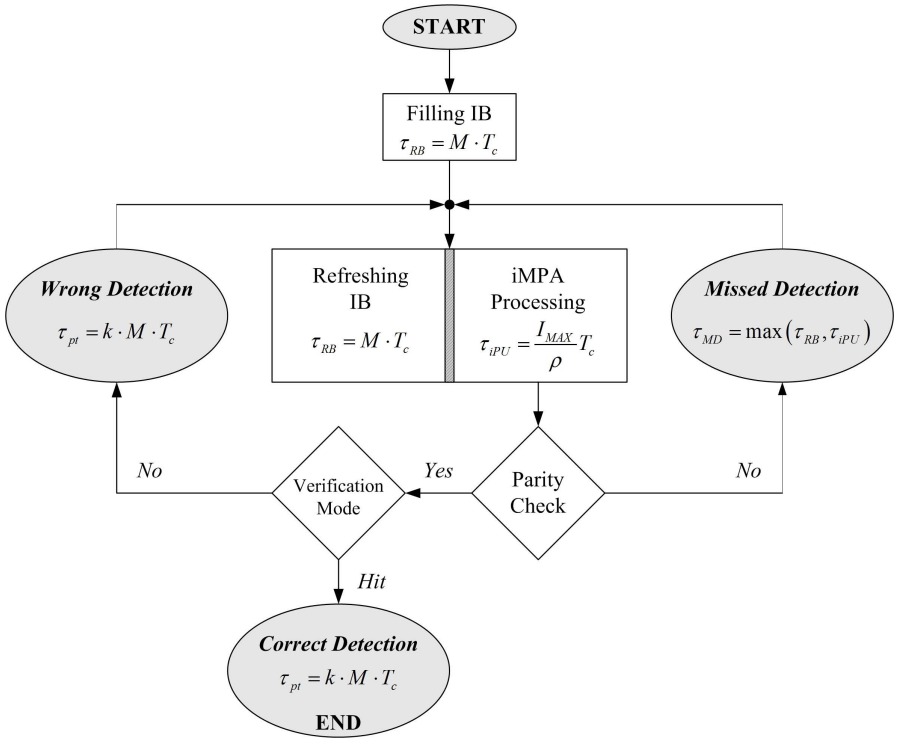

Figure 4.2: Main stages of the iDU.

Of course, a right selection and configuration of these parameters allows to correctly build a MP algorithm. Therefore, the following section provides a simplified treatment to define these cardinal parameters, so providing an immediate and practical approach to implement an iMP algorithm.

4.2.1 Iterative Message Passing Algorithms

Graphical modeling and MP algorithms are widely applied to inference problems in communications and signal processing, most notably decoding of modern error correction codes ([4], [6], and [17]). A graphical model captures constraints on variables by con-

necting *variable nodes* to *check nodes*, that constrain the configurations of the connected variables. For example, considering the set of m-sequence outputs $\{c_i\}_{i=0}^{M-1}$, one graphical model is a single check node with M binary variable connected. While there are 2^M possible combinations of these binary variables, the check node enforces the constraint that only $N = 2^r - 1$ (where r is the sequence polynomial degree) of these are allowable configurations. There are other graphical models that can enforce the same set of constraints. These are obtained by factoring this global constraint (involving all variables) into a sets of interdependent check nodes, each enforcing only local constraints (e.g., involving only a subset of variables). An example can help us.

Example 4.1. Consider the m-sequence generated by the following primitive polynomial

$$P(D) = D^6 + D + 1, \tag{4.3}$$

whose octal representation is $[103]_8$. The polynomial degree is $r = 6$ and the period $N = 2^r - 1 = 63$. A simple loopy graph to describe this LFSR is depicted in Fig. 4.3. Note that we use the convention that variable nodes are circles and check nodes are squares. Furthermore, each check node enforces the constraint $c_k \oplus c_{k-1} \oplus c_{k-6} = 0$ for any value of k. Of course this parity constraint is directly associated with the generating polynomial (4.3). This means that there are 4 valid local configurations, that are: $(c_k, c_{k-1}, c_{k-6}) \in \{(1,1,0),(1,0,1),(0,1,1),(0,0,0)\}$.

Note that graphical models like Fig. 4.3 are typically referred to as Tanner Graphs (TG), [56].

For a given graphical model with cycles, there is a well-defined MP algorithm that iteratively passes messages across edges in both directions (called iMP). The MP algorithm combines and marginalizes messages on variables over the constraints associated with the check nodes. Specifically, each check node will accept incoming messages, characterizing some form of soft-decision information, on the variables connected to it. These messages, which are sent from connected variable nodes, are then combined to obtain

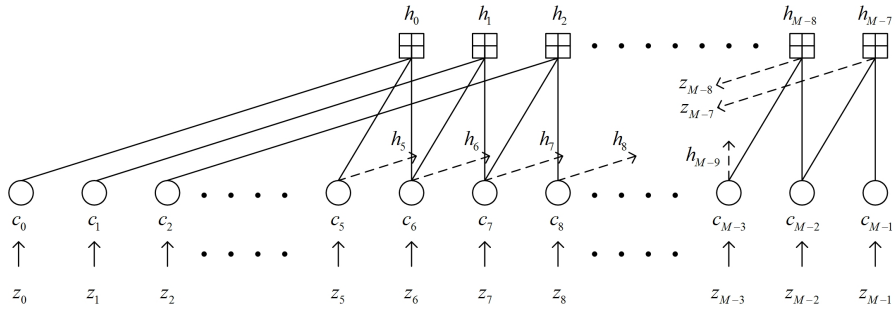

Figure 4.3: An example of TG.

soft-decision information (also called *metrics*) on all valid local configurations. Finally, these local configuration metrics are marginalized to produce output metrics.

Thus far, we have proposed a simple way to generate a graph model (a more complex construction will be presented in the next section) exploiting the generating polynomial structure (more examples are contained in [10], [47], and [63]). The next step is addressed to introduce the algorithm to process messages and the concept of the *activation schedule*. More specifically, regarding the first point, only the MS algorithm will be shown, because its complexity is lower than that of SP and does not require an estimate of E_c/N_0.

Referring to the communication model presented in Section 3.1 and to the TG reported in Fig. 4.3 and assuming to receive a set of M observations, represented by the vector $\mathbf{z} = [z_0, \ldots, z_{M-1}]$, the initial metrics (*soft-in* information) of an MS algorithm are defined

$$\Delta si_k \triangleq -\log\left[\frac{\mathcal{P}(z_k|c_k = 1)}{\mathcal{P}(z_k|c_k = 0)}\right] = z_i \quad 0 \leqslant k \leqslant M-1.$$

These become the initial input messages for all three check nodes which are connected to c_k. Under this convention, a message larger than 0 ($\Delta si_k > 0$) means a high confidence in $c_k = 0$, otherwise a $\Delta si_k < 0$ means that $c_k = 1$ is highly probable. Focusing on the generic couple variable-check node (c_i, h_j), the message from h_j to c_i is defined $\Delta\eta_{j,i}$, while the opposite message is pointed out $\Delta\mu_{i,j}$. So, the processing associated with the

message interchange can be viewed as a two-step process

$$\Delta\mu_{i,j} = \Delta si_i + \sum_{\substack{\forall n: h_n \to c_i \\ n \neq j}} \Delta\eta_{n,i} \tag{4.4}$$

$$\Delta\eta_{j,i} = \prod_{\substack{\forall n: c_n \to h_j \\ n \neq i}} \mathcal{S}\left(\Delta\mu_{n,j}\right) \cdot \min_{\substack{\forall n: c_n \to h_j \\ n \neq i}} \left|\Delta\mu_{n,j}\right| \tag{4.5}$$

where $h_n \to c_i$ (or $c_n \to h_j$) means the n^{th} check (variable) node connects to the i^{th} variable (j^{th} check) node[1] and $\mathcal{S}(.) = \text{sgn}(.)$. In other words, Eq. (4.4) computes messages from variable to check nodes, while check to variable messages are calculated by (4.5).

While the above defines the processing associated with message updating, to correctly carry out a MP algorithm, its *activation schedule* should be defined, too.

Definition 4.1. *An* activation schedule *is the order in which all variable and check nodes are activated, including when the condition upon which the algorithm is terminated.*

In particular, considering the graph in Fig. 4.3, a possible schedule[2] for the MP algorithm is to activate all variable nodes in parallel, then all check nodes in parallel, etc. One activation of all check and variable nodes will be defined as one *iteration*. Thus, at the end of each iteration, the *soft-out* metrics can be computed as follows

$$\Delta so_k = \Delta si_i + \sum_{\forall n: h_n \to c_i} \Delta\eta_{n,i}, \quad 0 \leqslant k \leqslant M-1 \tag{4.6}$$

and the hard decision performed on (4.6) is

$$\hat{c}_k = \begin{cases} 1, & \Delta so_k < 0 \\ 0, & \Delta so_k > 0 \end{cases} \quad \text{where } 0 \leqslant k \leqslant M-1.$$

In this way, we get the vector $\hat{\mathbf{c}}$ that is an estimate of the received PN sequence. Thus,

[1] The arrows also indicates the message direction.
[2] Many different activation schedules can be defined.

a parity control is performed to check that $\hat{\mathbf{c}}$ satisfies all the parity constraints provided by the graph. If the parity is verified the MP ends, otherwise a new iteration is run, until I_{MAX} iterations are got.

A basic result in this area is that if a graph has no cycles, then there is a schedule for which the MP algorithm is optimal. In other words, by repeatedly updating messages using simple local constraints, one can compute the same messages that would be computed using a single global constraint. The advantage is that the processing of many local constraints can be simpler than that associated with a single global constraint. Roughly, any activation schedule that passes messages from each node to all other nodes on a cycle-free graph is optimal and the MP algorithm converges to the same result that would have been obtained by processing the global constraint directly.

When the graphical model has cycles, the same message updating rules can be used, but the approaches are suboptimal heuristics, which we refer to as iMP algorithms. More specifically, little has been proven about the convergence properties and the long term evolution of the messages for these algorithms when cycles are present. However, it has been observed empirically that iMP algorithms are very effective and often yield performance near that of the optimal solution. Empirical results suggest that the iMP algorithm is most effective when there are no very short cycles and when the cycle structure is highly irregular. The advantage of using graphs with cycles is that the complexity of the resulting iMP algorithm can be significantly less than that of any MP algorithm associated with a cycle-free graphical model.

The graphical model associated with a particular set of constraints is not unique and selecting different models will yield a different MP algorithm. One way to alter a graph is to include *hidden variables*[3] that are neither the input nor the output of the system.

Example 4.2. The same m-sequence modeled in Fig. 4.3 can be modeled by the cycle-free graphical model in Fig. 4.4, in which the hidden variables σ_k, indexing all values

[3]The channel messages for these variables are taken to be zero for all conditional values.

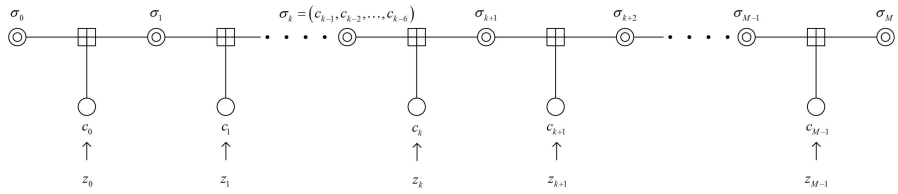

Figure 4.4: An example of a free-loop graph.

of $(c_{k-1}, c_{k-2}, \ldots, c_{k-6})$, have been added and are denoted with double-lined circles to distinguish them from the output variables.

These hidden variables are simply the state of the *finite state machine* (FSM) that represents that LFSR. An optimal MP algorithm on this graph is known as the *forward-backward algorithm* (FBA), [9]. In the FBA, messages are sent forward (left to right) starting at σ_0 and ending at σ_M, and then backward from σ_M to σ_0. This is one activation schedule that results in an optimal MP algorithm and further activations of the check nodes does not change the message values. It follows from the definition of the nonzero σ_k for an m-sequence that each state takes $2^r - 1$ values and each local check node has $2^r - 1$ valid configurations. Indeed, at the end of the forward recursion, the messages at σ_M are the $N = 2^r - 1$, which are all the correlations computed by the full-parallel search approach to acquire a PN sequence. This illustrates the importance of cycles in the graphical model to achieve low complexity iMP algorithms. Further details can be found in [1], [9], [10], [34], and [37].

4.2.2 Redundant Tanner Graphs

A graphical model (or TG) for a linear code can be mathematically described by its *parity matrix*, that contains all the edges between its variable and check nodes. In other words, assuming that the j^{th} column is directly associated with the j^{th} variable node and that the i^{th} row correspond to the i^{th} check node, the matrix element $h_{i,j}$ is 1 only if the i^{th} check node and the j^{th} variable node are connected, otherwise $h_{i,j} = 0$.

Now, in agreement with [10] (see also [34] and [37]), several TGs can be generated keeping in mind the particular LFSR sequence to be acquired. Nevertheless, the simplest and most general way to build graphical models is based on the generating polynomial of the LFSR sequence. Considering (3.1) and (3.2), it is clear that each generating polynomial identifies a set of parity checks, that can be used to construct a simple TG. The resulting parity matrix of the "code" is

$$\mathbf{H} = \begin{pmatrix} g_r & \cdots & g_0 & 0 & \cdots & \cdots & 0 \\ 0 & g_r & \cdots & g_0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & g_r & \cdots & g_0 & 0 \\ 0 & \cdots & \cdots & 0 & g_r & \cdots & g_0 \end{pmatrix}_{N_r \times N_c} \tag{4.7}$$

where $N_r = M - r$ is the number of rows (or number of parity equations), $N_c = M$ number of columns, r is the generating polynomial degree, M is the number of incoming observations (received soft information), and g_i (with $0 \leqslant i \leqslant r$) is the i^{th} polynomial coefficient. In (4.7), each row is a shifted repetition of the polynomial vector ($\mathbf{p} = [g_r \cdots g_0]$, associated to $P(D)$) of one column. We refer to this TG as a *basic graphical model* (BGM).

The regular structure of the BGM may cause the associated iMP algorithm to perform poorly [9], [10], and [34]. *Redundant graphical models* (RGMs) have been introduced to alleviate this effect. They are, roughly, made up of a set BGMs that are put together to form one big TG. Each BGM is based on one equivalent generating polynomial of the same LFSR sequence to be detected. The use of such redundancy has been shown to improve performance when each BGM has poor cycle structure.

Consider a set of $n+1$ equivalent generating polynomials that enforce the same LFSR sequence structure. A RGM can be defined by simply grouping all the BGMs generated

by these polynomials (as shown in (4.7)). The final parity matrix is

$$\mathbf{H_{RGM}} = \begin{pmatrix} \mathbf{H_0} \\ \mathbf{H_1} \\ \vdots \\ \mathbf{H_n} \end{pmatrix}_{N'_r \times N'_c} \tag{4.8}$$

where N'_r and N'_c are, respectively, the number of rows and columns. They are computed as

$$N'_r = [(n+1) \cdot M] - \sum_{j=0}^{n} r_j$$
$$N'_c = M$$

where M is the number of observations, and r_j (with $0 \leqslant j \leqslant n$) is the degree of the j^{th} equivalent polynomial.

An interesting family of RGMs was introduced in [63]. It is based on the following Galois field property (see [20], [35], and [53])

$$[P(D)]^{2^n} = P\left(D^{2^n}\right)$$

where $P(D)$ is a LFSR generating polynomial. Therefore, fixed n, a set of equivalent high-degree generating polynomial is identified by: $P\left(D^2\right)$, $P\left(D^4\right)$, $\cdots$, $P\left(D^{2^n}\right)$. Each polynomial can generate its own BGM (from (4.7)), and the union of all these BGMs produces a RGM (from the (4.8)) of order n ($n = 0$ clearly means a RGM made up of only one BGM based on $P(D)$). Furthermore, n is selected in agreement with

$$r_0 \cdot 2^n \leqslant M - 1 \;\Rightarrow\; n \leqslant K \cdot \log\left(\frac{M-1}{r_0}\right) \tag{4.9}$$

where r_0 is $P(D)$ degree, M is the number of received observations, and $K = [\log(2)]^{-1}$.

In other words, n is the largest integer that verifies the (4.9) inequality. We define these graphical models *Yeung*-RGM of order n (also pointed out YRGM$_n$).

4.2.3 Detection Algorithm Complexity

In case of the iMP detector, the complexity strictly depends on the particular graphical model that has been built. More specifically, considering the MS algorithm (presented in Section 4.2.1), its complexity can be easy measured counting the number of *sum* and *min* operations. Then, assuming that a *min*-operation is approximatively equivalent to a *sum*-operation ([43] and [50]), it is possible to compare the iMP complexity to that of the full-parallel, hybrid, and simple-serial searches (Sections 3.2.1.1, 3.2.1.3, and 3.2.1.2) simply comparing their total number of *sum*-operators per acquisition try.

The iMP complexity C_{iMP}, in case of a TG (e.g., Fig. 4.3), is

$$C_{iMP} \cong T_{\Sigma}^{vn} + T_{\Sigma}^{cn} \tag{4.10}$$

where T_{Σ}^{vn} and T_{Σ}^{cn} are respectively the numbers of *sum*-operators of all variable/check nodes, and they are computed using the following equations, [46]

$$T_{\Sigma}^{vn} \leqslant 2 \cdot \overline{N_{edg}^{vn}} \cdot N_{vn} \cdot I_{MAX} \tag{4.11a}$$

$$T_{\min}^{cn} \approx T_{\Sigma}^{cn} \leqslant \left(\overline{N_{edg}^{cn}} - 2\right) \cdot \overline{N_{edg}^{cn}} \cdot N_{cn} \cdot I_{MAX} \tag{4.11b}$$

where N_{vn} and N_{cn} are respectively the numbers of variable/check nodes of the TG, $\overline{N_{edg}^{vn}}$ and $\overline{N_{edg}^{cn}}$ are the mean numbers of edges per variable/check node, I_{MAX} is the max number of iterations, and $T_{\min}^{cn}$ is the total number of *min*-operations (a *min* is approximatively equivalent to a *sum*).

4.3 Acquisition Time

The acquisition time analysis of an iMP detector is performed in this section. More specifically, this analysis takes inspiration from the studies on the acquisition times of correlation-based detectors reported in [26], [27], [44], and [53]. Basically, building the Markov chain[4] that characterizes an iDU, it is possible to compute the *moment generating function* (MGF) of its acquisition time. Thus, deriving this result, it is possible to compute the first and the second order moments ([40] and [49]), so evaluating the mean and the variance of the acquisition time, as shown in [27].

4.3.1 Acquisition Time Analysis

Referring to Section 4.1, the architecture of iDU is reported Fig. 4.1. Considering that design, it is possible to build a state diagram of the acquisition unit, which is shown in Fig. 4.2. This picture summarizes the main steps that an iDU carries out to detect a transmitted PN sequences. More in detail, it also contains all delays that every phase introduces during a detection try. Exploiting this diagram, it is possible to generate the Markov chain which characterizes an iDU.

Assuming T_c as our time unit, the ad-hoc Markov chain in the z-transform domain is depicted in Fig. 4.5, where each node represents one of the stages of the iDU and each edge is labeled by a transition probability multiplied by its time delay. More specifically, starting from the $START$ node, the $A \mapsto B$ edge[5] is the IB filling stage, labeled z^M because its time interval is $\tau_{RB} = M \cdot T_c$. After that, the iPU provides an estimate vector on which parity checks are carried out by the PCU, so the missed detection is represented by the $B \mapsto B$ edge, labeled $\mathcal{P}_{MD} \cdot z^m$ (where m is (4.1)), while the right parity is the $B \mapsto C$ edge, labeled $(1 - \mathcal{P}_{MD}) \cdot z^{I_{MAX}/\rho}$. Following that line, the C node represents a

[4] A good introduction on Markov chain theory is in [40].

[5] $A \mapsto B$ means the edge that connects A and B nodes, and the direction is form A to B.

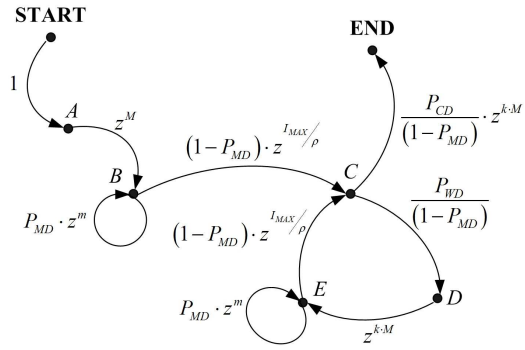

Figure 4.5: Markov chain of an iDU with all stages.

verification mode that can confirm the CD with the $C \mapsto END$ edge, labeled

$$\frac{\mathcal{P}_{CD}}{1-\mathcal{P}_{MD}} \cdot z^{k \cdot M},$$

because the probability is $\mathcal{P}_{CD}/(1-\mathcal{P}_{MD})$ (Eq. (4.2)) and the verification stage time is $\tau_{pt} = k \cdot M \cdot T_c$, or a WD can happen with

$$\frac{\mathcal{P}_{WD}}{1-\mathcal{P}_{MD}} \cdot z^{k \cdot M},$$

on the $C \mapsto D$ edge (Eq. (4.2)). In this last case, a new detection try is run: the $D \mapsto D$ edge represents the parity failure (MD), while $D \mapsto C$ is the case of correct parity. We remark that the D stage corresponds to the B stage, but it is split to simplify the next calculations.

Computing the MGF, we simplify the flow graph of Fig. 4.5 as shown in Fig. 4.6, where

$$Q_{AC}(z) = \frac{1-\mathcal{P}_{MD}}{1-P_{MD} \cdot z^m} \cdot z^{\frac{I_{MAX}}{\rho}+M} \tag{4.12a}$$

$$Q_C(z) = \frac{\mathcal{P}_{WD}}{1-P_{MD} \cdot z^m} \cdot z^{\frac{I_{MAX}}{\rho}+M \cdot k} \tag{4.12b}$$

so the MGF is (using (4.12) equations)

$$\begin{aligned} U(z) &= \frac{\mathcal{P}_{CD}}{1-\mathcal{P}_{MD}} \cdot Q_{AC}(z) \cdot \frac{1}{1-Q_c(z)} \\ &= \frac{\mathcal{P}_{CD}}{(1-\mathcal{P}_{MD}\cdot z^m) - \mathcal{P}_{WD}\cdot z^{\frac{I_{MAX}}{\rho}+k\cdot M}} \cdot z^{\frac{I_{MAX}}{\rho}+M\cdot(k+1)} \end{aligned} \tag{4.13}$$

we can check that (considering (4.2))

$$U(1) = \frac{\mathcal{P}_{CD}}{1-\mathcal{P}_{MD}-\mathcal{P}_{WD}} = \frac{\mathcal{P}_{CD}}{\mathcal{P}_{CD}} = 1,$$

as it should be. The mean of the acquisition time is derived from (4.13) (see also [27], [40], and [49])

$$\begin{aligned} \frac{\mu_{iDU}}{T_c} &= \left.\frac{\partial U(z)}{\partial z}\right|_{z=1} \xrightarrow{U(1)=1} \left.\frac{\partial \ln[U(z)]}{\partial z}\right|_{z=1} \\ &= \left[\frac{I_{MAX}}{\rho} + M\cdot(k+1)\right] + \frac{\mathcal{P}_{MD}\cdot m + \mathcal{P}_{WD}\cdot\left(\frac{I_{MAX}}{\rho}+k\cdot M\right)}{\mathcal{P}_{CD}} \end{aligned} \tag{4.14}$$

and the variance is (see also [27], [40], and [49])

$$\begin{aligned} \frac{\sigma^2_{iDU}}{T_c^2} &= \left[\frac{\partial^2 U(z)}{\partial z^2} + \frac{\partial U(z)}{\partial z} - \left(\frac{\partial U(z)}{\partial z}\right)^2\right]_{z=1} \\ &\xrightarrow{U(1)=1} \left[\frac{\partial^2 \ln[U(z)]}{\partial z^2} + \frac{\partial \ln[U(z)]}{\partial z}\right]_{z=1} \\ &= \left[\frac{\mathcal{P}_{MD}\cdot m + \mathcal{P}_{WD}\cdot\left(\frac{I_{MAX}}{\rho}+k\,M\right)}{\mathcal{P}_{CD}}\right]^2 \\ &\quad + \frac{\mathcal{P}_{MD}\cdot m^2 + \mathcal{P}_{WD}\cdot\left(\frac{I_{MAX}}{\rho}+k\cdot M\right)^2}{\mathcal{P}_{CD}}. \end{aligned} \tag{4.15}$$

Both these results ((4.14) and (4.15)) allow to evaluate the time performance of a coherent iDU with respect to the full-parallel, hybrid, and simple-serial search ones.

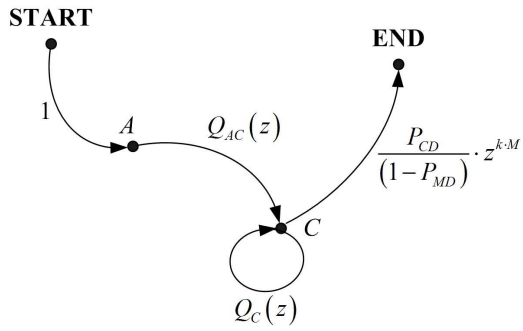

Figure 4.6: A simplified flow graph of an iDU.

4.4 Trinomial Multiples of Generating Polynomials

In Section 4.2, we have given an overview of MP algorithms to be run on graphical models with cycles. More specifically, we have introduced the MS algorithm and a typical activation schedule that is used to execute an iMP algorithm. Furthermore, we have provided a simple way to generate RGMs and a set of equations to evaluate the iMP-detector complexity. In particular, considering (4.10) and (4.11), it is clear that the iMP complexity depends on the number of connections of its graphical model. Therefore, to reduce the complexity is necessary have a *sparse* TG. Furthermore, from the decoding theory ([9] and [35]), it has been proved that iMP algorithms tends to ML performance with very sparse and random loopy graphs.

In this context, it is not sufficient to generate RGMs (Section 4.2.2) to have an iMP detector with good performance and low-complexity, but it is also important to reduce the number of connections in the graph. Referring to Section 4.2.2, a RGM is constructed using the generating polynomial of the considered LFSR sequence, and if that is a dense polynomial (more than 4 coefficients) its parity matrix results dense. Thus, in these circumstances, the solution is to search a *trinomial multiple*[6] of the generating polynomial, which provides a new set of parity equations, that can be used to generate a low-density

[6] A trinomial is a polynomial with 3 coefficients (see also Section 2.3.3.1 and [20]).

redundant TG (in agreement with construction strategy shown in Section 4.2.2).

The next section proposes four different algorithms to find trinomial multiples of an LFSR generating polynomial.

4.4.1 Algorithms to search Trinomial Multiple

This section contains four algorithms that we adapted from the literature to search trinomials which are multiple of a given generating polynomial. They are:

- *Algebraic Manipulation*.
- *Zech's Logarithm Table*.
- *Division Algorithms*.
- *Exhaustive Search*.

We remark that the only limit that we have in the search of trinomials is their degree, r_{tr}, that should be $r_{tr} \ll M$, where M is the number of observations at the receiver side. This condition is fundamental to perform RGMs as described in Section 4.2.2. When the condition $r_{tr} \ll M$ cannot be satisfied, 4-nomial multiples with degree $r \ll M$ will be searched.

4.4.1.1 Algebraic Manipulation

The simplest method to find a trinomial multiple of an LFSR generating polynomial is the algebraic manipulation of the initial polynomial till an equivalent one of higher degree with only 3 coefficients is achieved. We give an example.

Example 4.3. Consider the m-sequence identified by the following dense generating polynomial

$$P(D) = D^{10} + D^9 + D^8 + D^6 + D^3 + D^2 + 1$$

its octal representation is $[3515]_8$. An equivalent sparse polynomial can be computed as

$$P_{tr}(D) = P(D) \cdot [D^3 + D^2 + 1] = D^{13} + D^4 + 1$$

the octal representation is $[20021]_8$.

It is evident that $P_{tr}(D)$ is sparser than $P(D)$ and its degree is higher. So, $P_{tr}(D)$ can be used as BGM to generate a YRGM of order n (YRGM$_n$). Nevertheless, not all cases are so simple to process. Indeed, some LFSR sequences are characterized by equivalent sparse polynomials with very high degrees that require a quite complex computational search to be identified and evaluated with this method.

Note that this method can be also used to find any t-nomial[7] multiple of a prefixed generating polynomial.

4.4.1.2 Zech's Logarithm Table

Exploiting the GF theory, it is possible to provide an elegant way to search trinomial multiples of a primitive generating polynomial. The first important result is given by the following theorems ([21], [24], and [29]).

Theorem 4.1. *Let $f(x)$ be a primitive polynomial of degree r. Then there exists a trinomial $g(x) = x^i + x^j + 1$ which is divisible by $f(x)$.*

Theorem 4.2. *Let $f(x)$ be a primitive polynomial over $GF(2)$ of even degree r. Then the trinomial $x^{\frac{2}{3}(2^r-1)} + x^{\frac{1}{3}(2^r-1)} + 1$ is divisible by $f(x)$.*

Theorem 4.1 guarantees that a trinomial multiple of a primitive polynomial always exists, and Theorem 4.2 provides a simple formula to compute a trinomial that is divisible by any primitive polynomial, $f(x)$, with even degree r. Then, the number of all trinomial multiples of a given primitive polynomial is given by the following corollary (see also [24]).

[7] A t-nomial is a polynomial with t coefficients.

Corollary 4.1. *Given a primitive polynomial $f(x)$ of degree r, there will be $2^{r-1}-1$ distinct trinomial multiples (of degree $< 2^r - 1$) of $f(x)$.*

Thus, this number is $2^{r-1}-1$, where $r = \deg(f(x))$ and $f(x)$ is a primitive polynomial. Now, the issue is to find these trinomials, and one possible way is to exploit *Zech's Logarithm table*.

Let $f(x)$ be a primitive polynomial over $GF(2)$ of degree r and let α be a root of $f(x)$, in the extension $GF(2^r)$, so $\alpha^{2^r-1} = 1$ ([20] and [30]). Thus, if $g(x) = x^i + x^j + 1$ is a multiple of $f(x)$, then $g(\alpha) = \alpha^i + \alpha^j + 1 = 0$, so we have

$$\alpha^i = 1 + \alpha^j.$$

This result means that, given the value of j, such that $\alpha^i = 1 + \alpha^j$, the value of i is fixed depending on the primitive element α, which in turn depends on the specific primitive polynomial $f(x)$. Therefore, we can write

$$\alpha^{Z(j)} = 1 + \alpha^j \tag{4.16}$$

this equation defines the *Zech's Logarithm* $Z(j)$, and it can be directly associated to the trinomial $g(x) = x^{Z(j)} + x^j + 1$, which is a multiple of the primitive polynomial $f(x)$.

It is possible now to compute the Zech's Logarithm table which yields all couples $(j, Z(j))$ that satisfies (4.16). Therefore, this table also provides all trinomial multiples of a given primitive polynomial. Referring to [30], we assume that α is a root of a primitive polynomial

$$P(D) = D^r + p_{r-1}\, D^{r-1} + \ldots + p_1\, D + p_0$$

where $p_i \in F = GF(2)\ \forall i \in [0, r-1]$ and $\alpha \in GF(2^r)$. Since $P(\alpha) = 0$, we have

$$\alpha^r = -(p_{r-1}\, \alpha^{r-1} + \ldots + p_1\, \alpha + p_0). \tag{4.17}$$

By successive application of (4.17), a generic α^j can be written in the form of a polynomial in α of degree at most $r-1$ ([20] and [30])

$$\alpha^j = b_{r-1}\,\alpha^{r-1} + \ldots + b_1\,\alpha + b_0 \tag{4.18}$$

where $j \in \{\infty, 0, \ldots, 2^r-2\}$ and $b_i \in F$. Since α is a primitive element, j and $(b_{r-1}, \ldots, b_0)$ are in one-to-one correspondence for any $j \in \{\infty, 0, \ldots, 2^r-2\}$.[8] So, we can define N_j by

$$N_j = b_{r-1}\,2^{r-1} + \ldots + b_1\,2 + b_0 \tag{4.19}$$

where j and N_j, they are in one-to-one correspondence for any $j \in \{\infty, 0, \ldots, 2^r-2\}$. From (4.16) and (4.18), we have

$$\alpha^{Z(j)} = b_{r-1}\,\alpha^{r-1} + \ldots + b_1\,\alpha + \widetilde{b_0} \tag{4.20}$$

where $\widetilde{b_0} = b_0 + 1$. Thus, from (4.19) and (4.20), we obtain

$$N_{Z(j)} = b_{r-1}\,2^{r-1} + \ldots + b_1\,2 + \widetilde{b_0} = \begin{cases} N_j - 1, & \text{if } |N_j|_2 = 1 \\ N_j + 1, & \text{if } |N_j|_2 = 0 \end{cases}. \tag{4.21}$$

To sum up, the algorithm to generate a Zech's Logarithm table is reported in the following three steps.

Step 1 Tabulate N_j for $j \in \{\infty, 0, \ldots, 2^r-2\}$ by using (4.18) and (4.19).

Step 2 Tabulate $N_{Z(j)}$ for $j \in \{\infty, 0, \ldots, 2^r-2\}$ by using (4.21).

Step 3 Decide $Z(j)$ from $N_{Z(j)}$ by looking $N_{Z(j)}$ up in the table of N_j.

An example is useful to illustrate this procedure (more details are contained in [30]).

[8] ∞ is a symbol defined by $\alpha^\infty = 0$.

Example 4.4. Let $P(D) = D^5 + D^4 + D^3 + D^2 + 1$ and $\deg(P(D)) = 5$. By successive application of $\alpha^5 = \alpha^4 + \alpha^3 + \alpha^2 + 1$, the third column of Tab. 4.1 is achieved. Then, we compute N_j in the fourth column by using (4.19), and $N_{Z(j)}$ is got by (4.21) and reported in the fifth column. Finally, $Z(j)$ is given by taking the elements of the first column that are in the same rows where N_j assumes the same value of $N_{Z(j)}$ ($N_j \equiv N_{Z(j)}$).

Now, all trinomial multiples of $P(D)$ can be read in the Zech's Logarithm table and written as

$$g(D) = D^{Z(j)} + D^j + 1.$$

We remark that the total number of rows in a Zech's Logarithm table is $2^r - 1$, but the effective number of distinct trinomials is $(2^r - 2)/2 = 2^{r-1} - 1$. This result is definitely in agreement with Corollary 4.1. The following example lists all trinomials provided by Tab. 4.1.

Example 4.5. Referring to Ex. 4.4 and Tab. 4.1, all trinomials, that are multiple of $P(D)$, are listed in Tab. 4.2. The number of these trinomials is $2^{r-1} - 1 = 15$. In particular, the trinomial with the lowest degree is (to satisfy the condition $r_{tr} \ll M$)

$$g(D) = D^8 + D^5 + 1.$$

This can be used to generate a RGM as proposed in Section 4.2.2.

4.4.1.3 Division Algorithms

Several studies were done to compute multiples of primitive polynomials over $GF(2)$ in [24], [38], and [58]. In this section, we report an example of algorithm based on division between polynomials. In particular, let $g(D) = D^{r_{tr}} + D^j + 1$ be a trinomial multiple of $f(D)$ of degree r. Then, the search of couples (r_{tr}, j) is done fixing a value of $r_{tr} \in [r + 1, 2^r - 1]$, and searching j such that $f(D)|g(D)$. Algorithm 1 show the pseudo-code of the proposed procedure. Other algorithms based on the same criteria and

j	α^j	$b_4, \ldots, b_0$	N_j	$N_{Z(j)}$	$Z(j)$
∞	0	00000	0	1	0
0	1	00001	1	0	∞
1	α	00010	2	3	20
2	α^2	00100	4	5	9
3	α^3	01000	8	9	26
4	α^4	10000	16	17	18
5	α^5	11101	29	28	8
6	α^6	00111	7	6	21
7	α^7	01110	14	15	29
8	α^8	11100	28	29	5
9	α^9	00101	5	4	2
10	α^{10}	01010	10	11	16
11	α^{11}	10100	20	21	12
12	α^{12}	10101	21	20	11
13	α^{13}	10111	23	22	17
14	α^{14}	10011	19	18	27
15	α^{15}	11011	27	26	25
16	α^{16}	01011	11	10	10
17	α^{17}	10110	22	23	13
18	α^{18}	10001	17	16	4
19	α^{19}	11111	31	30	30
20	α^{20}	00011	3	2	1
21	α^{21}	00110	6	7	6
22	α^{22}	01100	12	13	24
23	α^{23}	11000	24	25	28
24	α^{24}	01101	13	12	22
25	α^{25}	11010	26	27	15
26	α^{26}	01001	9	8	3
27	α^{27}	10010	18	19	14
28	α^{28}	11001	25	24	23
29	α^{29}	01111	15	14	7
30	α^{30}	11110	30	31	19

Table 4.1: The Zech's Logarithm table of $P(D) = D^5 + D^4 + D^3 + D^2 + 1$.

$g(D) = D^{Z(j)} + D^j + 1$
$D^{20} + D + 1$
$D^9 + D^2 + 1$
$D^{26} + D^3 + 1$
$D^{18} + D^4 + 1$
$D^8 + D^5 + 1$
$D^{21} + D^6 + 1$
$D^{29} + D^7 + 1$
$D^{16} + D^{10} + 1$
$D^{12} + D^{11} + 1$
$D^{17} + D^{13} + 1$
$D^{27} + D^{14} + 1$
$D^{25} + D^{15} + 1$
$D^{30} + D^{19} + 1$
$D^{24} + D^{22} + 1$
$D^{28} + D^{23} + 1$

Table 4.2: List of trinomial multiples of $P(D) = D^5 + D^4 + D^3 + D^2 + 1$.

more details are also reported in [24] and [58].

4.4.1.4 Exhaustive Search

Let $f(D)$ be the generating polynomial of the PN sequence $\mathbf{a}$ with period N, the property $f(D)\mathbf{a} = 0$ is always satisfied, as proved by (2.7). Thus, assuming that $g(D)$ is a polynomial multiple of $f(D)$, the equality $g(D)\mathbf{a} = 0$ is also satisfied (Theorem 2.4). This property can be exploited to search trinomial multiples of a generating polynomial by performing an exhaustive search. Thus, defining $g(D) = D^i + D^j + 1$ and $\deg(f(D)) = r$, Algorithm 2 shows again the pseudo-code.

This algorithm can be also applied to search any t-nomial multiple of the initial generating polynomial.

Algorithm 1: Division algorithm.

Input: A primitive polynomial $f(D)$ and its degree r.
Output: A trinomial $g(D) = D^{r_{tr}} + D^j + 1$.

```
1  begin
2      repeat
3          Random selection of an integer r_tr ∈ [r + 1, 2^r − 1];
4          for (1 ⩽ j ⩽ r_tr − 1) do
5              g(D) ⟵ D^{r_tr} + D^j + 1;
                   // Computing the reminder of g(D)/f(D)
6              rem ⟵ Rem(f(D), g̃(D));
7              if (rem = 0) then
8                  return g(D);
9              end
10         end
11     until (rem ≠ 0);
12 end
```

Algorithm 2: Exhaustive search algorithm.

Input: A generating polynomial $f(D)$ and its degree r.
Output: A trinomial $g(D) = D^i + D^j + 1$.

```
1  begin
       // Generating the LFSR sequence and computing its period
2      a ⟵ GenerateSeq(f(D));
3      N ⟵ ComputePeriod(a);
       // Searching i
4      for (r < i ⩽ N) do
           // Searching j
5          for (0 < j < i) do
               // Testing the parity provided by g(D) = D^i + D^j + 1
6              if (g(D)a = 0) then
7                  return g(D);
8              end
9          end
10     end
11 end
```

Chapter 5

Performance Evaluation

This chapter shows the performance of iMP detectors and compares it to that of all standard algorithms presented in Chapter 3. This comparison will be done in terms of wrong, missed, and correct detection probabilities and acquisition-time performance. We will also show that RGMs, generated by using trinomial multiples of dense primitive polynomials, offer better performance in terms of correct detection probability with respect to that of RGMs generated by dense primitive polynomials. Finally, a new activation schedule and a different graph-model implementation for Gold codes are also proposed and tested. The outline is listed here.

- **Section 5.1** compares different ways to implement loopy graphs in terms of performance and complexity.
- **Section 5.2** shows a hierarchical implementation to generate loopy graph in case of Gold sequences.
- **Section 5.3** compares the acquisition time performance of an iMP algorithm to that of all standard detectors.

5.1 Equivalent Sparse Polynomials with High-Degree

Using *equivalent sparse polynomials with high-degree* (e.g., trinomials and 4-nomials), that are multiple of dense primitive polynomial, to generate RGMs, it is possible to get significant improvements in terms of detection probabilities and low-complexity. This section reports some simulation results to prove this.

Furthermore, an alternative activation schedule is also shown in this section, to reduce the memory required to store all node messages during each MP iteration.

5.1.1 Simulation Results and Performance

As demonstrated in [63], YRGMs offer great benefits in terms of acquisition probability at low-SNR, in the case of *sparse* generating polynomials (e.g., with only 3 or 4 nonzero coefficients). However, for *dense* generating polynomials (e.g., more than 4 nonzero coefficients) experiments suggest that poor performance is obtained using TG models. This problem is common to many m-sequences and Gold codes that, having a dense polynomial, cannot be efficiently acquired using iMP algorithms on YRGMs.

In order to address this problem, the key idea is to find equivalent higher degree generating polynomials that are sparse and use these as BGMs to build new RGMs, following (4.7) and (4.8). These models will be denoted by adding the superscript *esp*, to identify RGMs generated by equivalent sparse polynomials of higher degree – e.g., RGM^{esp}. Furthermore, we will show these RGMs^{esp} provide better performance and lower complexity than YRGMs generated by initial dense polynomials. Now the problem is to search and identify these high degree equivalent polynomials with 3 or 4 nonzero coefficients (respectively called trinomials and 4-nomials). This result can be achieved exploiting the algorithms presented in Section 4.4.

An example is useful to well explain this procedure and illustrates the potential improvements. Consider the m-sequence identified by the following dense primitive poly-

nomial

$$P(D) = D^{12} + D^{11} + D^9 + D^7 + D^6 + D^5 + 1$$

its octal representation is $[15341]_8$ and the period is $N = 2^r - 1 = 4\,095$. An equivalent sparse polynomial can be easy computed as

$$\begin{aligned} P_{esp}(D) &= P(D) \cdot [D^{19} + D^{18} + D^{17} + D^{15} + D^{14} + D^{13} \\ &\quad + D^{12} + D^{10} + D^8 + D^6 + D^5 + D^2 + 1] \\ &= D^{31} + D^2 + 1 \end{aligned}$$

whose octal representation is $[20000000005]_8$. It is evident that $P_{esp}(D)$ is more sparse than $P(D)$ and its degree is higher. So, $P_{esp}(D)$ can be used as BGM to generate a YRGM^{esp} of order n (YRGM$_n^{esp}$). Assuming the signal model of Chapter 3, and considering 1024 observations, a comparison between the YRGM$_5^{esp}$, YRGM$_6$, BGM, and ML algorithm is reported in Fig. 5.1. Here, we show the values of the detection probability obtained by simulation as a function of the signal-to-noise ratio (SNR) E_c/N_0. In particular, the YRGM$_5^{esp}$ gains about 7 to 8 dB with 30 iterations on the YRGM$_6$ that requires 100 iterations (the gain is larger than 10 dB if it is compared to the BGM). The complexity depends on the number of iterations run and number of edges per variable/check node. Assuming one *min*-operation is equivalent to a *sum*-operation, it is possible to evaluate the following complexity factors:[1]

$$\frac{C_{YRGM_5^{esp}}}{C_{YRGM_6}} < \frac{1}{24} \quad \text{and} \quad \frac{C_{YRGM_5^{esp}}}{C_{ML}} \leqslant \frac{1}{4}$$

where C_{Mod} points out the complexity of one model (Mod is YRGM$_5^{esp}$, or YRGM$_6$, or ML[2]). These two factors evaluate the complexity of the YRGM$_5^{esp}$ with respect to the others. In the both cases, they demonstrate that the complexity is lower than that of the

[1]The complexity is measured counting the number of *sum*-operators per iteration and multiplying it by the number of iterations (see also Section 4.2.3).

[2]Full-parallel implementation of the ML algorithm.

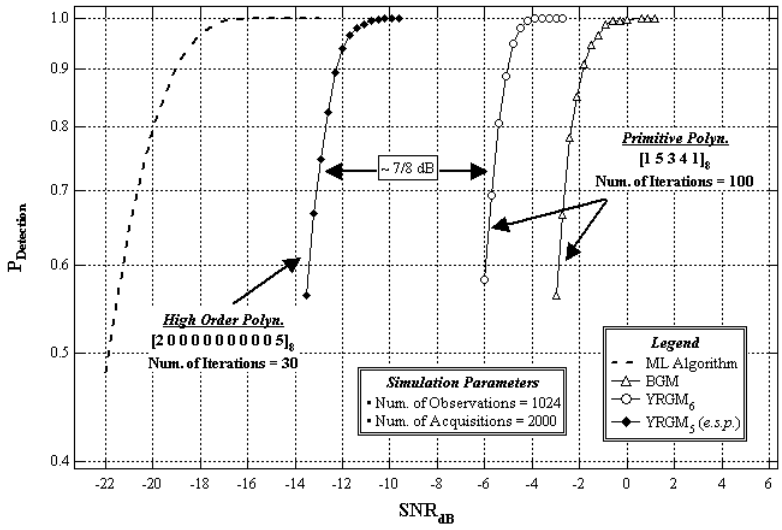

Figure 5.1: Performance in case of the m-sequence $[15341]_8$ ($r = 12$).

YRGM_6 (based on the primitive polynomial) and of the ML algorithm.

Nevertheless, not all cases are so simple to process. Indeed, some LFSR sequences are characterized by equivalent sparse polynomials with very high degrees that require a quite complex computational search to be identified and evaluated. In these cases, one of the algorithms reported in Section 4.4 (e.g., the exhaustive search or the division algorithm) can be performed to find all equivalent sparse polynomials that will be used to build RGMs^{esp}. An example is provided by GPS/SBAS Gold codes. Indeed, they are generated by the dense generating polynomial showed in (3.5) that has equivalent sparse polynomials with very high degree. Carrying out an exhaustive search (on one period, 1 023 chips), it is possible to identify 341 equivalent sparse polynomials: only 1 with 3 coefficients, and 340 with 4 coefficients, that offer the possibility to generate a large number of different RGMs^{esp} with only $3 \div 4$ edges per check node and without 4-cycles. A pair of RGMs^{esp} are compared to the YRGM_5, the BGM, and the ML algorithm in Fig. 5.2. In particular, the RGM_M^{esp} is the largest RGM achievable with all the 341 GPS/SBAS equivalent sparse

polynomials, and the RGM^{esp} is generated grouping the BGMs obtained by the following sparse equivalent polynomials

$$P_{Eq,1}(D) = D^{682} + D^{341} + 1$$
$$P_{Eq,2}(D) = D^{111} + D^{46} + D^{5} + 1$$
$$P_{Eq,3}(D) = D^{222} + D^{92} + D^{10} + 1$$
$$P_{Eq,4}(D) = D^{444} + D^{184} + D^{20} + 1$$
$$P_{Eq,5}(D) = D^{888} + D^{368} + D^{40} + 1.$$

As in the previous example, both these models (RGM$_M^{esp}$ and RGM^{esp}) present benefits in terms of detection probability with respect to the YRGM$_5$ and BGM, with fewer iterations. Furthermore, the complexity is lower than that of the YRGM$_5$ (generated by (3.5)), as follows

$$\frac{C_{RGM_M^{esp}}}{C_{YRGM_5}} < \frac{1}{3}, \quad \frac{C_{RGM^{esp}}}{C_{YRGM_5}} < \frac{1}{90}, \quad \frac{C_{RGM^{esp}}}{C_{ML}} < \frac{1}{2}.$$

$C_{RGM_M^{esp}}$ and $C_{RGM^{esp}}$ are smaller than C_{YRGM_5}. Furthermore, $C_{RGM^{esp}}$ is smaller than C_{ML}, but it is possible to verify that $C_{RGM_M^{esp}} > C_{ML}$. Indeed, this method is most suitable for longer Gold sequences.

The examples above demonstrate that equivalent sparse polynomials can be efficiently used to generate RGMs, on which low-complexity iMP algorithms are run, achieving good performance at low-SNR.

Another parameter to be considered, when iMP algorithms are implemented, is the memory required to store all node messages during each iteration. The memory requirements depends on the selected TG and, more specifically, on the number of edges. For this reason, large RGMs typically have large memory requirements. To address this problem, a different activation schedule is proposed. In the previous examples, all variable or check nodes were activated in parallel at the same time implying that all messages along edges had to be stored. The memory requirements for large RGMs can be reduced by

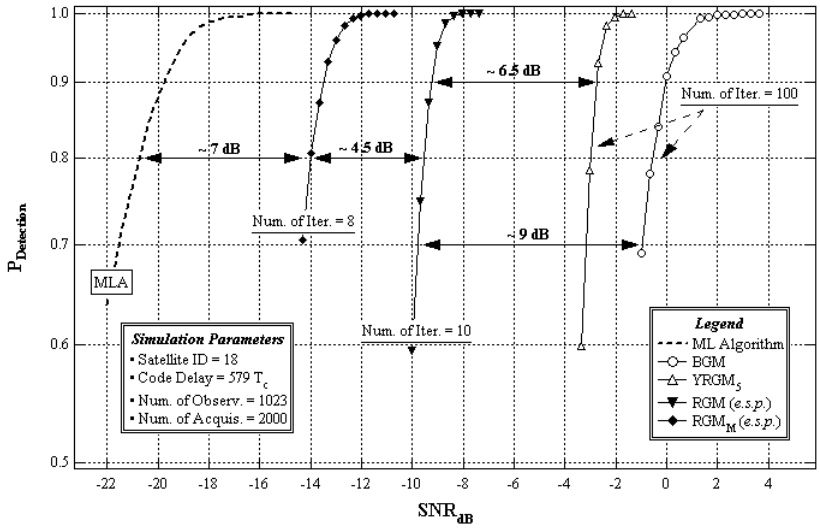

Figure 5.2: Performance in case of the GPS/SBAS codes.

using a different activation schedule and a modified message passing algorithm. The activation schedule is based on breaking the RGM into a set of smaller sub-TGs, each one containing a portion of the parity checks (e.g., two different sub-TGs do not share common parity checks), and running the iMP sequentially on all sub-TGs (see also [31] and [25]). This yields a reduction in memory requirements if the messages between variable and check nodes in a given sub-TG are not stored while iterating other sub-TGs. This, in fact, is not standard iMP since these internal messages would normally be required for the next activation of the iMP algorithm on the sub-TG. Nevertheless, this yields a significant decrease in memory requirements with a small performance degradation. More precisely, assume there are I_2 sub-TGs, one iteration is made for each sub-model (*inner iterations* = 1), and its soft output metrics become the input metrics in the next sub-model. The MP ends either when all check nodes of one sub-TG are verified or when I_1 outer iterations are performed. In this way, the computational complexity is the same as in the previous examples, but the required memory is only that needed to store all messages of the largest

sub-TG and, therefore, it is reduced. A pictorial representation of this schedule is reported in Fig. 5.3 and the algorithm is shown in App. B. In addition to not storing the internal messages between sub-TG iterations, we use a MS algorithm with damping factor α (see also App. B, [31], [25], and [8]).

A performance comparison between these two methods is displayed in Fig. 5.4. Indeed, the RGM_M^{esp} acquisition is compared to the results achievable splitting this graph in 2 and 10 sub-TGs, with a damping factor $\alpha = 0.1$. It is quite evident that the split models maintain the same rapid convergence of RGM_M^{esp}, but their performance can change in function of α and the characteristics of sub-TGs in which the initial RGM is separated. In this case, the required memory, it is about 1/2 in case of 2 sub-TGs and 1/10 in case of 10 sub-TGs, because the initial RGM was split in sub-TGs with about the same dimensions. Furthermore, all the models have the same complexity.

5.2 Hierarchical Model

In this section, we propose a distinct approach to acquire Gold codes using *hierarchical models* (HMs) built by their two generating m-sequences. In other words, introducing a set of suitably defined hidden variables (see Section 4.2, [9], [34], and [60]), it is possible to generate two graphical models serially connected. Thus, the result is a hierarchical TG on which an iMP algorithm can be run. The following section gives an example using GPS/SBAS Gold sequences.

5.2.1 Simulation Results and Performance

Considering the primitive polynomials in (3.4) (GPS/SBAS LFSR generators), (3.3) can be expressed as

$$P(D) = P_{c'}\, D^{10} + P_{c'}\, D^{9} + P_{c'}\, D^{8} + P_{c'}\, D^{6} + P_{c'}\, D^{3} + P_{c'}\, D^{2} + P_{c'}$$

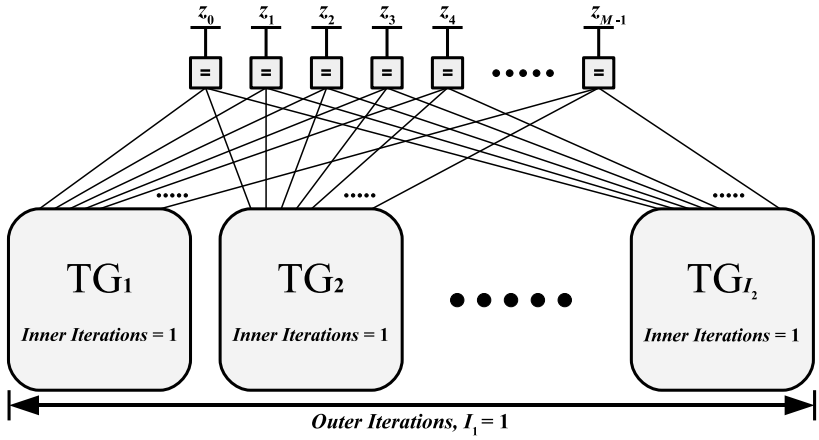

Figure 5.3: Multi-TGs activation schedule.

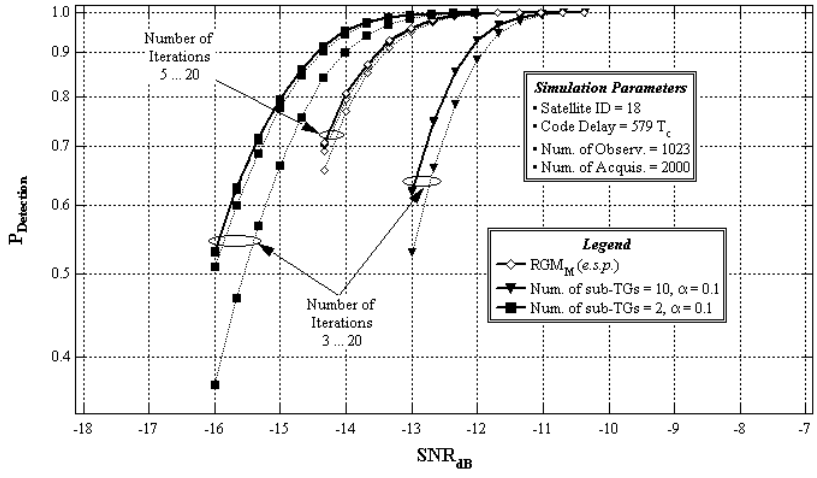

Figure 5.4: Comparison between the two activation schedules.

which leads to the system of equations

$$\sigma_{k-10} \triangleq z_k \oplus z_{k-3} \oplus z_{k-10} \tag{5.1a}$$

$$0 = \sigma_k \oplus \sigma_{k-2} \oplus \sigma_{k-3} \oplus \sigma_{k-6} \oplus \sigma_{k-8} \oplus \sigma_{k-9} \oplus \sigma_{k-10} \tag{5.1b}$$

where (5.1a) defines a generic hidden variable σ_k. It is evident that the system (5.1) identifies a concatenated structure because (5.1b) (*slave*) directly depends on (5.1a) (*master*). Furthermore, (5.1a) provides a first set of local constraints depending of the index k that are useful to generate a first preliminary model, $\mathbf{H}'$. A second set of constraints is got by (5.1b), delivering the second graphical model $\mathbf{H}''$. These two models are related by their hidden variables. An example of this hierarchical TG is displayed in Fig. 5.5, in case of 23 observations. Specifically, a generic h_i' corresponds to a $\mathbf{H}'$ check node, while a generic h_j'' is a check nodes of the $\mathbf{H}''$ matrix.

We also remark that this method can be applied to a generic set of Gold codes and that the example in Fig. 5.5 is not the only way to realize a HM for GPS/SBAS sequences. Indeed, many different HMs can be constructed manipulating their generating m-sequence polynomials and inverting the master and slave polynomials. Of course, if we group together more HMs, graphical models with redundancy are obtained. The message updating algorithm for HMs is described in App. C.

Some preliminary results, obtained using this technique, are displayed in Fig. 5.6. Specifically, HM_1 and HM_2 are both generated by $P_{c'}$ and $P_{c''}$, from (3.4), by inverting master and slave. The performance is better in the case of $P_{c''}$-master and $P_{c'}$-slave (HM_2). Furthermore, manipulating $P_{c''}(D)$ yields

$$P_{c''}^{esp}(D) = P_{c''}(D) \cdot \left[D^3 + D^2 + 1\right] = D^{13} + D^4 + 1$$

where $P_{c''}^{esp}$-master and $P_{c'}$-slave are used to generate HM_3 and HM_4. In particular, in the case of HM_4, the $\mathbf{H}''$ is constructed as YRGM_6. So, from results in Fig. 5.6, it

is evident that more sparse polynomials can generate models (HM_3) that provide better performance with fewer iterations than more dense models (HM_1 and HM_2) and, adding more redundancy (HM_4), this performance can be improved. Therefore, these results confirm the previous section conclusions.

5.3 Acquisition time

In this section, the iMP detector is compared to the standard correlation-based algorithms in terms of acquisition time performance. This analysis is also enriched comparing their wrong, missed, and correct detection probabilities and algorithm complexity.

5.3.1 Simulation Results and Performance

This section compares the iDU to the full-parallel (FPA), hybrid (HA), and simple-serial (SSA) search algorithms. This comparison is performed in terms of detection performance, acquisition time, and implementation complexity. More specifically, the detection performance of each method is measured in terms of $\mathcal{P}_{WD}$, $\mathcal{P}_{MD}$, and $\mathcal{P}_{CD}$ as a function of the signal-to-noise ratio (SNR = E_c/N_0). These curves are obtained by simulating the DS/SS communication system described in Chapter 3. The time performance of each algorithm is characterized by the mean and the variance of its acquisition time. In particular, for the iDU, these parameters are evaluated by (4.14)-(4.15). While in case of:

- FPA, we have Equations (3.11),
- SSA, we have Equations (3.13),
- HA, we have Equations (3.16).

To complete the comparison of these algorithms, an analysis in terms of implementation complexity is necessary too. This can be done by counting the number of *sum*-operators of each method per detection try, and comparing these figures. In particular, in case of FPA,

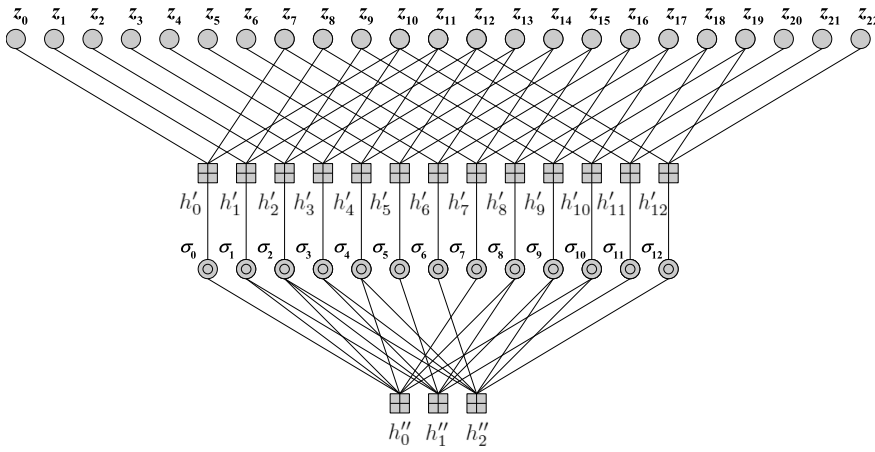

Figure 5.5: Example of hierarchical model for GPS/SBAS codes.

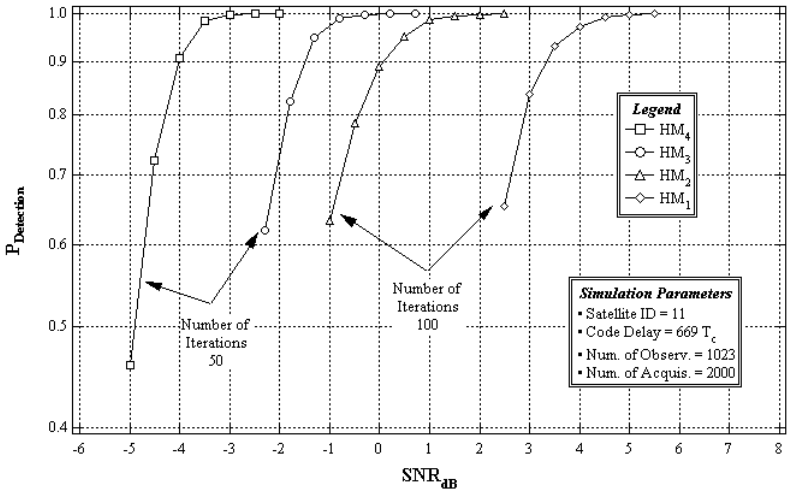

Figure 5.6: Hierarchical model performance.

SSA, and HA, the complexity is respectively measured with (3.10), (3.12), and (3.14), while the iMP complexity is (4.11) and (4.10).

We will perform this comparison on the specific example of the m-sequence $g_{18} = [1000201]_8$, with degree $r = 18$ and period $N = 262\,143$. For the iMP algorithm, at the receiver side (see Fig. 3.1), 1 024 observations (M) are collected. Furthermore, the SSA threshold is $\lambda = 0.85$. About the graphical model used by the iDU, we refer to Section 4.2.2 ([47] and [63]). In this particular example a YRGM_5 is implemented and the number of iteration is $I_{MAX} = 30$.

Some simulation results are shown in Fig. 5.7. In particular, the $\mathcal{P}_{FA}$ of the SSA is lower than the $\mathcal{P}_{WD}$ and $\mathcal{P}_{MD}$ of the iDU. Furthermore, it can be clearly neglected for SNR > -16 dB ($\mathcal{P}_{FA} < 10^{-9}$). The detection probabilities of each algorithm (SSA, FPA, and iMP algorithm) are given in Fig. 5.8. Of course, the best performance is provided by the ML algorithm. The cross-over value SNR $\cong -13.8$ dB splits the chart in two regions in which the iDU outperforms the SSA (SNR > -13.8 dB) and vice versa (SNR < -13.8 dB). A comparison between the acquisition times of the SSA and the iMP detector is contained in Tab. 5.1. To provide a realistic scenario, we can suppose a chip time $T_c \approx 0.1 \div 1\,\mu$sec, that is typical of satellite positioning systems (as GPS, [2], and Galileo System, [16]), and an iteration time $T_{it} \approx 1\,\mu$sec that can be considered a reasonable figure for the state-of-the-art of LDPC decoders. Therefore, $\rho = T_c / T_{it} \approx 0.1 \div 1$, so we can consider as the worst case: $\rho = 0.1$. Furthermore, we assume that the penalty time (defined in (3.8)) is the same for all the detectors and its value is proportional to 10 times the number of observations M, so $k = 10$. Finally, for the SSA two typical values of q are considered: N and $2\,N$ (it respectively means a search step of one chip or half a chip). All results contained in Tab. 5.1 are obtained considering the curves in Fig. 5.7 and Fig. 5.8 and Equations (4.14), (4.15), and (3.13). This table shows the huge gap in terms of the acquisition time between the iDU and the SSA. In particular, the iMP detector has a mean time and a standard deviation about 10^5 times smaller than the SSA ones. This result is basically due to the q-factor that depends on the selected search

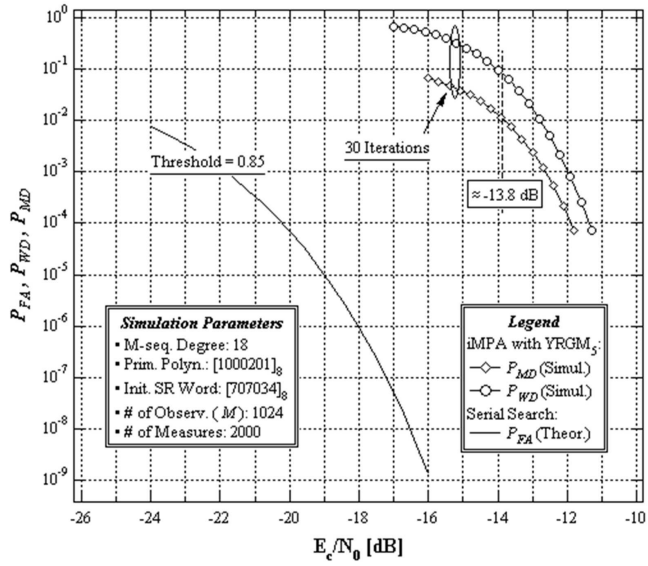

Figure 5.7: The SSA $\mathcal{P}_{FA}$ vs iMP $\mathcal{P}_{WD}$ and $\mathcal{P}_{MD}$.

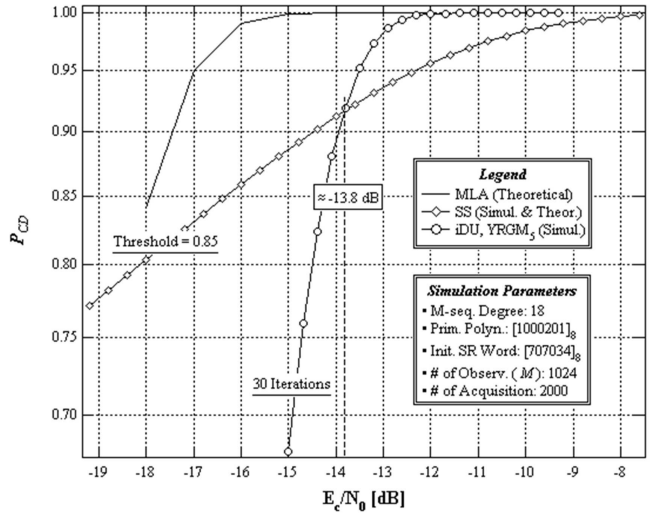

Figure 5.8: $\mathcal{P}_{CD}$ of the iDU, the SSA, and the FPA.

step and on the sequence length. A similar comparison between the iMP and the FPA is performed in Tab. 5.2 (considering Fig. 5.8 and (4.14), (4.15), and (3.11)). In this case, the FPA implementation is optimal in terms of the acquisition time performance, but the iDU tends to its performance. More specifically, it is evident the difference between the iDU standard deviation and that of FPA, due to the highest $\mathcal{P}_{WD}$ and $\mathcal{P}_{MD}$ of the iMP detector. Nevertheless, increasing the SNR, both the iDU acquisition time parameters (the mean and the standard deviation) tend to reduce the gap with respect to FPA. This result still demonstrates that the iMP algorithm is a sub-optimal solution of the ML algorithm in terms of time performance.

Coming to the issue of complexity, let us call C_{Alg} the complexity of one detection algorithm (Alg is a FP, SS, H, or iMP algorithm). Thus, in agreement with (3.10) and (3.12), $C_{FP} = M\,N = M\,(2^r - 1) = 268\,434\,432$ and $C_{SS} \approx M = 1\,024$. In case of the iMP detector, the complexity strictly depends on the particular graphical model that has been built, and it is measured in agreement with the equations (4.11) and (4.10). Hence, C_{iMP} can be computed and reported in Tab. 5.3 in comparison with the other algorithms. Our analysis shows that the iDU complexity is considerably smaller than that of the full-parallel implementation of the ML algorithm, and, of course, considerably higher than that of the SSA. Therefore the final conclusion is that the iMP detector is a good trade-off between the FPA and SSA, because it allows to have a rapid detection (that tends to the FPA acquisition time) and good performance, in terms of correct detection probability at low SNR, with a complexity lower than a full parallel implementation.

The last step is to compare the hybrid search to the iMP detector. To do this, we refer to the architectural design of an hybrid algorithm presented in Section 3.2.1.3. The analysis is conducted identifying the hybrid detector that is equivalent to the iDU in terms of acquisition time, and then, its complexity is evaluated and compared to that of the iMP algorithm.

Considering the previous m-sequence $g_{18} = [1000201]_8$ and assuming that the threshold λ of the hybrid unit is 0.85, Equations (3.15) and (3.16) can be used to evaluate the

SNR (dB)	SSA		iDU			$q = N$		$q = 2 \cdot N$	
	$\mathcal{P}_{FA}$	$\mathcal{P}_{CD}$	$\mathcal{P}_{WD}$	$\mathcal{P}_{MD}$	$\mathcal{P}_{CD}$	$\frac{\mu_{iMP}}{\mu_{SS}}$	$\frac{\sigma_{iMP}}{\sigma_{SS}}$	$\frac{\mu_{iMP}}{\mu_{SS}}$	$\frac{\sigma_{iMP}}{\sigma_{SS}}$
-12	$< 10^{-9}$	≈ 0.956	≈ 0.0008	≈ 0.0002	≈ 0.999	$7.89\,10^{-5}$	$3.07\,10^{-6}$	$3.95\,10^{-5}$	$1.53\,10^{-6}$
-13.8	$< 10^{-9}$	≈ 0.92	≈ 0.07	≈ 0.01	≈ 0.92	$7.86\,10^{-5}$	$2.67\,10^{-5}$	$3.93\,10^{-5}$	$1.33\,10^{-5}$
-14.4	$< 10^{-9}$	≈ 0.902	≈ 0.155	≈ 0.021	≈ 0.824	$8.31\,10^{-5}$	$4.12\,10^{-5}$	$4.15\,10^{-5}$	$2.06\,10^{-5}$

Table 5.1: Comparison of the acquisition time between the SSA and the iDU.

SNR (dB)	FPA	iDU			$\frac{\mu_{iMP}}{\mu_{FP}}$	σ_{iMP}	σ_{FP}	$\frac{\sigma_{iMP}}{\sigma_{FP}}$
	$\mathcal{P}_{CD}$	$\mathcal{P}_{WD}$	$\mathcal{P}_{MD}$	$\mathcal{P}_{CD}$				
-12	1	≈ 0.0008	≈ 0.0002	≈ 0.999	≈ 1.027	298.742	0	–
-13.8	≈ 1	≈ 0.07	≈ 0.01	≈ 0.92	≈ 1.099	3 020.783	≈ 0	–
-14.4	≈ 0.9995	≈ 0.155	≈ 0.021	≈ 0.824	≈ 1.204	4 995.884	≈ 251.997	19.825

Table 5.2: Comparison of the acquisition time between the FPA and the iDU.

YRGM$_5$	$N_{vn} \equiv M$	N_{cn}	$\overline{N_{edg}^{vn}}$	$\overline{N_{edg}^{cn}}$	C_{iMP}	$\frac{C_{iMP}}{C_{SS}}$	$\frac{C_{iMP}}{C_{FP}}$
$I_{MAX} = 30$	1 024	5 010	14.68	3	1 352 839.2	1 321.132	$\frac{1}{198.42}$

Table 5.3: Comparison of the implementation complexity of the iDU, the FPA, and the SSA.

mean of the acquisition time at SNR $\cong -13.8\,\text{dB}$ as a function of the number of correlation branches. More specifically, let B_H is the number of correlation fingers of a hybrid system, we compute the following function (from Eq. (3.16))

$$\Psi\left(B_H\right) \triangleq \log_{10}\left[\frac{\mu_H\left(B_H\right)}{\mu_{FP}}\right]_{\substack{\lambda=0.85 \\ E_c/N_0=-13.8\,\text{dB}}}$$

and the result is plotted in Fig. 5.9. From Tab. 5.2, we have

$$\log_{10}\left[\frac{\mu_{iMP}}{\mu_{FP}}\right]_{E_c/N_0=-13.8\,\text{dB}} = 0.040997692 \longrightarrow \Psi\left(\tilde{B}_H\right)$$

so, using (3.15) and (3.16), and by numerical approximations, we get

$$\Psi\left(\tilde{B}_H\right) \approx 0.040997692 \Longrightarrow \tilde{B}_H \approx 131\,071.$$

It is now possible to evaluate the complexity of the hybrid detector and compare it to that of the iMP algorithm. Therefore, using (3.14) and considering Tab. 5.3, we find that

$$C_H = B_H \cdot M = 134\,216\,704 \Longrightarrow \frac{C_{iMP}}{C_H} \approx \frac{1}{99.211} \Longrightarrow C_{iMP} \ll C_H.$$

Thus, to get the same mean of the iMP acquisition time, the hybrid search needs to have a number of correlation branches which tends to the sequence period N ($B_H \rightarrow N$). Therefore, its complexity tends to that of a full-parallel implementation, and so it is higher than the iDU complexity. Like this, many other examples can be done (with sparse/dense generating polynomials) that provide similar results and conclusions, so definitely proving the effectiveness of MP algorithms to acquire LFSR sequences.

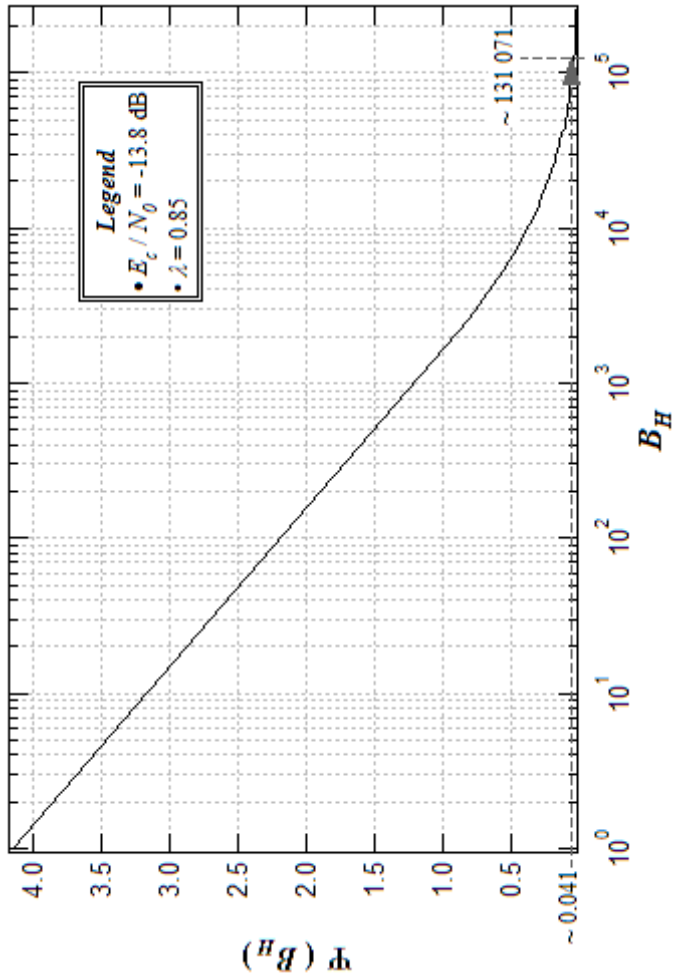

Figure 5.9: The mean of the acquisition time of a hybrid detector.

Chapter 6

Conclusions

In this thesis, a novel detection technique that exploits iMP algorithms to perform initial acquisition of a spreading code is analyzed. Such technique basically uses the channel soft information as messages to be exchanged within a graphical model with cycles to estimate the transmitted LFSR sequence, thus implicity evaluating its code delay. In particular, the graphical model can be implemented exploiting the generating polynomial structure of the sequence, as shown in Chapter 4 and in [47] and [63].

The main feature that makes this algorithm very attractive is fast acquisition of long SS sequences. More specifically, the standard algorithms are not adequate to acquire these codes, because the full-parallel implementation features fast detection at a price of a high complexity, and a simple-serial search results in a low complexity algorithm but has a prohibitively long acquisition time. In this context, the iMP detector is a good trade-off between these two techniques, because its correct detection probability is equivalent to that of the simple-serial algorithm, but its performance in terms of acquisition time tends to that of a full-parallel implementation with a much lower complexity. Comparing the iDU to an hybrid search detector, it can also be shown that an hybrid algorithm tends to the iMP acquisition time when the number of its correlation fingers tends to the sequence

period. This means that the complexity of an hybrid detector tends to that of a full-parallel search, and so it is higher than that of the iMP algorithm.

We also focused on rapid acquisition of LFSR sequences with dense generating polynomials and Gold codes, using iMP algorithms. Exploiting the theorems reported in [18], we showed that these two cases are closely related, because every set of Gold sequences can be described by a high-order LFSR (typically dense) generator. Thus, in order to detect these sequences, iMP algorithms are run on RGMs generated by high degree equivalent polynomials that are very sparse (trinomials and 4-nomials). These polynomials can be algebraically computed by manipulating dense primitive polynomials of LFSR sequences, as shown in Section 4.4. Simulations results demonstrate that such technique, called RGMs^{esp}, offer benefits in terms of performance at low-SNR and low complexity respect graphical models based on dense polynomials.

Addressing the problem of large memory requirements, a different activation schedule and modified MP rules were proposed using the MS algorithm with a damping factor ([8] and [25]). This approach yields significant memory savings without a change in computational complexity as compared to the initial iMP algorithms on RGMs. This modification can provide good performance, but requires care in selecting the sub-model partition and the damping factor.

In order to acquire Gold codes, HMs are also proposed. Our preliminary results demonstrate the richness of the available HMs for Gold codes, but the performance is not as good as that obtained with redundant TG models. Thus, an interesting future direction is to further explore these HMs in search for better performance/complexity compromises.

Our results, take along with those in [10] and [63], demonstrate that iMP algorithms can yield low-complexity, full-parallel search for rapid PN acquisition that approximates the ML method. Many topics still remain to be investigated, such as: the SS acquisition with the joint coarse estimation of code timing and carrier phase and frequency, the improvement of HM or RGM^{esp} techniques, the search of other graphical models that could introduce more benefits, more detailed hardware implementation considerations, and de-

sign of high-performance PN sequences, other than m-sequences and Gold codes, which are studied to be acquired by iDU.

Appendix A

Extension of Finite Fields

A.1 Extension Field $GF(p^n)$

Here, we define the procedure to construct the finite field $GF(p^n)$, [20]. Let $GF(p) = \mathbb{Z}_p$ be the finite field of order p, where p is a prime, the elements of $GF(p)$ are $\{0, 1, \ldots, p-1\}$, and the addition + and the product $\cdot$ are carried out modulo p. Considering the following fact, [20].

Fact A.1. *For every prime p and every degree $n > 1$, there is at least one irreducible polynomial of degree n over $\mathbb{Z}_p$.*

Assuming n a positive integer, thus to construct the finite field $GF(p^n)$ of order p^n, we choose $f(x)$ to be an irreducible polynomial over $GF(p)$ of degree n. Let α be a formal symbol that satisfies $f(\alpha) = 0$ (α is a root of $f(x)$), we define

$$GF(p^n) = \{a_0 + a_1\,\alpha + \ldots + a_{n-1}\,\alpha^{n-1} | a_i \in GF(p)\}.$$

We also define two operations: + and $\cdot$ on $GF(p^n)$ as follows. Let $g(\alpha), h(\alpha) \in GF(p^n)$

be

$$g(\alpha) = \sum_{i=0}^{n-1} a_i \, \alpha^i \quad \text{and} \quad h(\alpha) = \sum_{i=0}^{n-1} b_i \, \alpha^i.$$

- **Addition**: $g(\alpha) + h(\alpha) = \sum_{i=0}^{n-1} (a_i + b_i) \, \alpha \in GF(p^n)$;
- **Multiplication**: $g(\alpha) \cdot h(\alpha) = r(\alpha)$;

where $r(\alpha)$ is computed as follows.

- **Step 1** - Multiply $g(\alpha)$ and $h(\alpha)$ according to the polynomial multiplication

$$g(\alpha)\, h(\alpha) = \sum_{k=0}^{2(n-1)} c_k \, \alpha^k \triangleq c(\alpha)$$

$$c_k \triangleq \sum_{i+j=k} a_i \, b_j$$

- **Step 2** - Dividing $c(\alpha)$ by $f(\alpha)$, we can get two polynomials $q(\alpha)$ and $r(\alpha)$ such that $c(\alpha) = q(\alpha)\, f(\alpha) + r(\alpha)$ with $\deg(r(\alpha)) < n$.

Since, $f(\alpha) = 0$, we have $r(\alpha) = c(\alpha) \in GF(p^n)$. In other words, $r(x)$ is the reminder of the $g(x)\, f(x)$ divided by $f(x)$.

Theorem A.1. *The set $GF(p^n)$ together with the two operations defined above ($+$ and $\cdot$) forms a finite field and the order of this field is p^n.*

The polynomial $f(x)$ is called a *defining polynomial* of $GF(p^n)$ and α is a *defining element* of $GF(p^n)$ over $GF(p)$. From the construction of $GF(p^n)$, $f(\alpha) = 0$. Therefore, α is a root of $f(x)$ in $GF(p^n)$, so we say that $GF(p^n)$ is a *finite extension* of $GF(p)$.

A.2 Periods of Minimal Polynomials

Theorem A.2. *For any $0 \neq \alpha \in GF(p^n)$, the period of the minimal polynomial of α is equal to the order of α*

$$\text{per}(m(x)) = \text{ord}(\alpha).$$

Appendix B

Multi-TG Model

B.1 Message-Updating Algorithm with Damping Factor

Referring to Fig. 5.3, the processing that we propose to update messages in a multi-TG model is characterized by three loops, which are connected in a pyramid structure, as shows in Algorithm 3. More specifically, the *outer loop* (with I_1 iterations) counts the number of global iterations[1], the *TG-loop* is addressed to scan all sub-graphs (their number is I_2), and finally the *inner loop* (with I_3 iterations) carries out a MS algorithm with damping factor for a specific sub-TG. The algorithm is reported below.[2]

- **STEP 0** - Initialization, $i_3 = 0$ (line 7 of Algorithm 3):

$$\Delta so_i = so_i[1] - so_i[0] = \Delta si_i$$

$$\Delta\mu_{i,j} = 0, \quad \forall i \rightarrow j \text{ (Messages from Variable to Check Nodes)}$$

$$\Delta\eta_{j,i} = 0, \quad \forall j \rightarrow i \text{ (Messages from Check to Variable Nodes)}$$

[1] One global iteration ends when all sub-TGs are activated.
[2] $\mathcal{S}(.) = \text{sgn}(.)$.

- **Step 1** - Iteration i_3^{th}, $0 \leqslant i_3 < I_3$ (line 9 of Algorithm 3):

$$\Delta\mu_{i,j} = \mu_{i,j}[1] - \mu_{i,j}[0] = \Delta so_i - \left(\alpha \cdot \Delta\eta_{j,i}\right)$$
$$\Delta\eta_{j,i} = \eta_{j,i}[1] - \eta_{j,i}[0] = \prod_{\substack{\forall k \to j \\ k \neq i}} \left[\mathcal{S}\left(\Delta\mu_{k,j}\right)\right] \cdot \min_{\substack{\forall k \to j \\ k \neq i}} \left(\left|\Delta\mu_{k,j}\right|\right)$$
$$\Delta so_i = so_i[1] - so_i[0] = \Delta si_i + \left(\alpha \cdot \sum_{\forall k \to i} \Delta\eta_{k,i}\right).$$

Algorithm 3: Message-updating with damping factor α.

Input: I_1, I_2, I_3, initial damping factor α_0, soft-input vector **si** (dim. M).
Output: soft-output vector **so** (dim. M), hard decision vector **hd** (dim. M).

```
1  begin
       // Initializing
2      α^(0) ⟵ α_0;
3      s ⟵ si ;
       // Outer Loop:  Iteration Loop
4      for (0 ≤ i_1 < I_1) do
           // Tanner Graph Loop:  Loop to Scan Sub-TG
5          for (0 ≤ i_2 < I_2) do
6              H_TG ⟵ H_{i_2};
7              InitMinSumALg(s);
               // Inner Loop:  Min-Sum with Damping Factor
8              for (0 ≤ i_3 < I_3) do
9                  so ⟵ MinSumAlg(s, H_TG, α^(i_1));
10                 hd ⟵ HardDecision(so);
11                 if (H_TG · hd = 0) then
12                     return [hd];
13                 else
14                     s ⟵ so;
15                 end
16             end
17         end
18         α^(i_1) ⟵ [α_0 + (1 − α_0) · (i_1 / (I_1 − 1))];
19     end
20 end
```

Appendix C

Hierarchical Model

C.1 Message-Updating Algorithm

To describe the message updating proposed for HMs, a generic path of these graphs is extracted and shown in Fig. C.1. Let **z** be an observation vector of M elements, the *soft-in* information (Δsi_i), in negative *log*-domain, is defined (see also [9]) as $\Delta si_i \triangleq -\log\left[\frac{\Pr(z_i|c_i=1)}{\Pr(z_i|c_i=0)}\right] = z_i$, $0 \leqslant i \leqslant M-1$. The main steps of our algorithm are below.

- **Step 0** - Initialization ($n = 1$):

$$\Delta so_i \triangleq so_i[1] - so_i[0] = \Delta si_i$$

  all other messages are zero.

- **Step 1** - Iteration n^{th}, $1 \leqslant n \leqslant I_{MAX}$:

$$\begin{aligned} \Delta\mu'_{i,j} &= \Delta so_i - \Delta\eta'_{j,i}, && \forall i : z_i \to h'_j \\ \Delta\alpha_j &= \prod_{\forall i: z_i \to h'_j} \left[\mathcal{S}(\Delta\mu'_{i,j})\right] \cdot \min_{\forall i: z_i \to h'_j} \left(\left|\Delta\mu'_{i,j}\right|\right), && \forall j : h'_j \to \sigma_j \end{aligned}$$

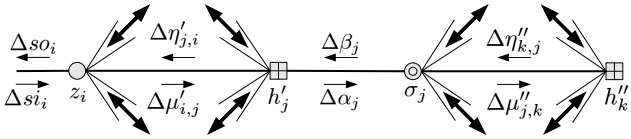

Figure C.1: One hierarchical model path.

$$\Delta\mu''_{j,k} = \Delta\alpha_j + \Delta\beta_j - \Delta\eta''_{k,j}, \qquad \forall j : \sigma_j \to h''_k$$

$$\Delta\eta''_{k,j} = \prod_{\substack{\forall l:\sigma_l \to h''_k \\ l \neq j}} \left[\mathcal{S}(\Delta\mu''_{l,k})\right] \cdot \min_{\substack{\forall l:\sigma_l \to h''_k \\ l \neq j}} \left(\left|\Delta\mu''_{l,k}\right|\right), \qquad \forall k : h''_k \to \sigma_j$$

$$\Delta\beta_j = \sum_{\forall k: h''_k \to \sigma_j} \Delta\eta''_{k,j} = S_{\beta_j} \cdot M_{\beta_j}, \qquad \forall j : \sigma_j \to h'_j$$

$$\Delta\eta'_{j,i} = S_{\beta_j} \cdot \prod_{\substack{\forall m: \\ z_m \to h'_j \\ m \neq i}} \mathcal{S}(\Delta\mu'_{m,j}) \cdot \min_{\substack{\forall m: \\ z_m \to h'_j \\ m \neq i}} \left[M_{\beta_j}, \left|\Delta\mu'_{m,j}\right|\right], \qquad \forall j : h'_j \to z_i$$

$$\Delta so_i = \Delta si_i + \sum_{\forall j: h'_j \to z_i} \left(\Delta\eta'_{j,i}\right), \qquad \forall i \in [0, M-1]$$

where $S_{\beta_j} = \mathcal{S}\left(\Delta\beta_j\right)$ and $M_{\beta_j} = |\Delta\beta_j|$. The hard decision is made on the *soft-out* information (Δso_i). If the estimated vector verifies all the parity checks, the algorithm will end, otherwise $n = n + 1$ and the **Step 1** will restart. The algorithm definitely ends when the last iteration is performed ($n = I_{MAX}$).

Bibliography

[1] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 325–343, March 2000.

[2] M. Aparicio and *et al.*, *Global Positioning Systems: Theory and Applications*, B. W. Parkinson and J. J. Spilker, Eds. 370 L'Enfant Promenade, SW, Washington, DC 20024-2518: American Institute of Aeronautics and Astronautics, Inc., 1996, vol. I and II.

[3] N. E. Bekir, "Bounds on the distribution of partial correlation for PN and Gold sequences," Ph.D. dissertation, University of Southern California, Los Angeles, CA 90007, January 1978.

[4] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European Transactions on Telecommunications*, vol. 9, no. 2, pp. 155–172, March-April 1998.

[5] S. Berberich and S. Fischer, "AGGA-3 functional specification," EADS Astrium, Tech. Rep. 1, 16 October 2003.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes(1)," in *Proc. Int. Conf. Communications*, Geneva, Switzerland, 23-26 May 1993, pp. 1064–1070.

[7] K. K. Chawla and D. V. Sarwate, "Parallel acquisition of PN sequences in DS/SS systems," *IEEE transactions on communications*, vol. 42, no. 5, pp. 2155–2164, May 1994.

[8] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity-check codes," *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 406–414, March 2002.

[9] K. M. Chugg, A. Anastasopoulos, and X. Chen, *Iterative Detection: Adaptivity, Complexity Reduction, and Applications*. Norwell, Massachusetts 02061: Kluwer Academic Publishers, 2001.

[10] K. M. Chugg and M. Zhu, "A new approach to rapid PN code acquisition using iterative message passing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 5, pp. 884–897, May 2005.

[11] R. De Gaudenzi, L. Fanucci, F. Giannetti, M. Luise, and M. Rovini, "Satellite mobile communications spread-spectrum receiver," *IEEE Aerospace and Electronic Systems Magazine*, vol. 18, no. 8, pp. 23–30, August 2003.

[12] R. De Gaudenzi, M. Luise, and R. Viola, "Chip timing synchronization in an all-digital band-limited DS/SS modem," in *Proc. IEEE International Conference on Communications 91*, vol. 3, Denver CO, USA, 23-26 June 1991, pp. 1688–1692.

[13] ——, "A digital chip timing recovery loop for band-limited direct-sequence spread-spectrum signals," *IEEE Transactions on Communications*, vol. 41, no. 11, pp. 1760–1769, November 1993.

[14] R. C. Dixon, *Spread Spectrum Systems*, ser. Wiley-Interscience Publication. John Wiley & Sons, Inc., 1976.

[15] P. Fenton, B. Falkenberg, T. Ford, K. Ng, and A. J. Van Dierendonck, "NovAtel's GPS receiver, the high performance OEM sensor of the future,," in *Proc. of GPS 1991*, T. I. of Navigation, Ed., 1991, pp. 49–58.

[16] Galileo JU (Joint Undertaking), "Galileo open service signal in space interface control document (OS SIS ICD) - draft 0," ESA, Tech. Rep. GAL-OS-SIS-ICD/D.0, May 23 2006.

[17] R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, pp. 21–28, January 1962.

[18] R. Gold, "Optimal binary sequences for spread spectrum multiplexing," *IEEE Transactions on Information Theory (Correspondence)*, vol. 13, pp. 619–621, October 1967.

[19] ——, "Maximal recursive sequences with 3-valued recursive cross-correlation functions," *IEEE Transactions on Information Theory (Correspondence)*, vol. 14, pp. 154–156, January 1968.

[20] R. W. Golomb and G. Gong, *Signal Design for Good Correlation for Wireless Communications, Cryptography, and Radar*. New York, NY 10011-4211: Cambridge University Press, 2005.

[21] S. W. Golomb and P. F. Lee, "Which irreducible polynomials divide trinomials over GF(2)," *SETA 2004*, no. 3486, pp. 414–424, 2004.

[22] GPS JTO (Joint Program Office), "NAVSTAR GPS space segment/navigation user interfaces," ARINC Research Corporation, Fountain Valley, California, Tech. Rep. ICD-GPS-200, Rev. C-PR (Public Release Version), October 1993.

[23] C. Gumacos, "Analysis of an optimal synch search procedure," *IEEE Transactions on Communications Systems*, vol. CS-11, pp. 89–99, March 1963.

[24] K. C. Gupta and S. Maitra, "Primitive polynomials over GF(2) - a cryptologic approach," *ICICS 2001*, no. 2229, pp. 23–34, November 2001.

[25] T. R. Halford and K. M. Chugg, "Random redundant soft-in soft-out decoding of linear block codes," in *Proc. IEEE International Symp. on Information Theory*, Seattle WA, USA, July 2006.

[26] J. K. Holmes, "Acquisition time performance of PN spread-spectrum systems," *IEEE Transactions on Communications*, vol. COM-25, no. 8, pp. 778–784, August 1977.

[27] ——, *Coherent Spread Spectrum Systems*. New York, NY: John Wiley & Sons, Inc., 1982.

[28] J. K. Holmes and L. Biederman, "Delay-lock-loop mean time to lose lock," *IEEE Transactions on Communications*, vol. COM-26, no. 11, pp. 1549–1557, November 1978.

[29] K. Huber, "Some comments on Zechs logarithms," *IEEE Transactions on Information Theory*, vol. 36, no. 4, pp. 946–950, July 1990.

[30] K. Imamura, "A method for computing addition table in GF(p^n)," *IEEE Transactions on Information Theory*, vol. IT-26, no. 3, pp. 367–369, May 1980.

[31] J. Jiang and K. R. Narayanan, "Iterative soft decision decoding of Reed Solomon codes," *IEEE Communications Letters*, vol. 8, no. 4, pp. 244–246, April 2004.

[32] B.-J. Kang and I.-K. Lee, "A performance comparison of code acquisition techniques in DS-CDMA system," *Wireless Personal Communications*, vol. 25, no. 2, pp. 163–176, May 2003.

[33] E. D. Kaplan, *Understanding GPS: Principles and Applications*, ser. Mobile Communications Series, E. D. Kaplan, Ed. Norwood, MA 02062: Artech House, Inc., 1996.

[34] F. R. Kschichang, B. J. Frey, and H. A. Loeliger, "Factor graph and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, February 2001.

[35] S. Lin and D. J. Costello, *Error Control Coding Fundamentals and Applications*, 2nd ed. Upper Saddle River, NJ 07458: Pearson Prentice-Hall, 2003.

[36] J. H. Lindholm, "An analysis of the pseudo-randomness properties of subsequences of long m-sequences," *IEEE Transactions on Information Theory*, vol. IT-14, no. 4, pp. 569–576, July 1968.

[37] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, pp. 28–41, March 2004.

[38] S. Maitra, K. C. Gupta, and A. Venkateswarlu, "Multiples of primitive polynomials and their products over GF(2)," in *Proc. SAC 2002*, August 2002, pp. 218–234.

[39] H. Meyr and G. Ascheid, *Synchronization in Digital Communications: Phase-, Frequency-Locked Loops, and Amplitude Control*, ser. Wiley Series in Telecommunications. John Wiley & Sons, Inc., January 1990, vol. 1.

[40] A. Papoulis and U. Pillai, *Probability, Random Variables and Stochastic Processes*, 4th ed. McGraw-Hill, 2002.

[41] R. L. Peterson, R. E. Ziemer, and D. E. Borth, *Introduction to Spread-Spectrum Communications*. Prentice-Hall, Inc., 1995.

[42] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, "Theory of spread spectrum communications - a tutorial," *IEEE Transactions on Communications*, vol. COM-30, no. 5, pp. 855–884, May 1982.

[43] L. Ping, X. Huang, and N. Phamdo, "Zigzag codes and concatenated zigzag codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 800–807, February 2001.

[44] A. Polydoros and C. L. Weber, "A unified approach to serial search spread-spectrum code acquisition," *IEEE Transactions on Communications*, vol. 32, no. 5, pp. 542–560, May 1984.

[45] E. C. Posner, "Optimal search procedure," *IEEE Transactions on Information Theory*, vol. IT-11, pp. 157–160, July 1963.

[46] F. Principe, K. M. Chugg, and M. Luise, "Performance evaluation of message-passing-based algorithms for fast acquisition of spreading codes with application to satellite positioning," in *Proc. ESA Workshop on Satellite Navigation User Equipment Technologies NAVITEC 2006*. Noordwijk, The Netherlands: ESTEC, 11-13 December 2006.

[47] ——, "Rapid acquisition of Gold codes and related sequences using iterative message passing on redundant graphical models," in *Proc. IEEE Military Communications Conference*, Washington DC, USA, 23-25 October 2006.

[48] F. Principe, C. Terzi, M. Luise, and M. Casucci, "SOFT-REC: a GPS/EGNOS software receiver," in *Proc. 14th IST Mobile & Wireless Communication Summit*. Dresden, Germany: EURASIP, 19-23 June 2005.

[49] J. G. Proakis, *Digital Communications*, 4th ed., S. W. Director, Ed. McGraw-Hill, 2001.

[50] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," in *Proc. ICC 1995*, vol. 2, Seattle, WA, USA, 18-22 June 1995, pp. 1009–1013.

[51] RTCA JTO (Joint Program Office), "Minimum operational performance standards for global positioning system/wide area augmentation system airborne equipment," RTCA Inc., Washington DC, Tech. Rep. RTCA/DO-299C, 2001.

[52] D. V. Sarwate and M. B. Pursley, "Crosscorrelation properties of pseudorandom and related sequences," in *Proc. of the IEEE*, vol. 68, no. 5, May 1980, pp. 593– 619.

[53] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, *Spread Spectrum Communications Handbook*. McGraw-Hill TELECOM, 2002.

[54] J. J. Spilker, *Digital Communications by Satellite*, ser. Prentice-Hall Information Theory Series, T. Kailath, Ed. Prentice-Hall, Inc., 1977.

[55] M. Srinivasan and D. V. Sarwate, "Simple schemes for parallel acquisition of spreading sequences in DS/SS systems," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 3, pp. 593–598, August 1996.

[56] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, September 1981.

[57] A. Van Der Meer and R. Liyana-Pathirana, "Performance analysis of a hybrid acquisition system for DS spread spectrum," in *Proc. TENCON 2003*, vol. 1, 15-17 October 2003.

[58] A. Venkateswarlu and S. Maitra, "Further results on multiples of primitive polynomials and their products over GF(2)," *ICICS 2002*, no. 2513, pp. 231–242, 2002.

[59] B. Vigoda, J. Dauwels, M. Frey, N. Gershenfeld, T. Koch, H. A. Loeliger, and P. Merkli, "Synchronization of pseudorandom signals by forward-only message passing with application to electronic circuits," *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3843–3852, August 2006.

[60] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, S-581 83 Linköping, Sweden, 1996.

[61] L. Yang and L. Hanzo, "Iterative soft sequential estimation aided differential acquisition of m-sequences," in *Proc. Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, vol. 3, 22-25 April 2003, pp. 1629–1633.

[62] ——, "Acquisition of m-sequences using recursive soft sequential estimation," *IEEE Transactions on Communications*, vol. 52, no. 2, pp. 199–204, February 2004.

[63] O. W. Yeung and K. M. Chugg, "An iterative algorithm and low complexity hardware architecture for fast acquisition of long PN codes in UWB systems," *Springer J. VLSI and Signal Processing (Special Issue on UWB Systems)*, vol. 43, no. 1, pp. 25–42, April 2006.

[64] W. Zhuang, "Noncoherent hybrid parallel PN code acqusition for CDMA mobile communications," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 4, pp. 643–656, November 1996.

List of Publications

International Conferences

IC01 F. Principe, M. Luise, and K. M. Chugg, "Performance Evaluation of Message-Passing-Based Algorithms for Fast Acquisition of Spreading Codes with Application to Satellite Positioning," in *Proc. NAVITEC 2006*, ESTEC Noordwijk (The Netherlands), December 11-13, 2006.

IC02 F. Principe, K. M. Chugg, and M. Luise, "Rapid Acquisition of Gold Codes and Related Sequences Using Iterative Message Passing on Redundant Graphical Models," in *Proc. MILCOM 2006*, Washington DC (USA), October 23-25, 2006.

IC03 G. Bacci, F. Principe, M. Luise, C. Terzi, and M. Casucci, "SOFT-REC: a GPS Real Time Software Receiver with EGNOS Augmentation," in *Proc. Workshop on EGNOS Performance and Applications 2005*, Gdynia (Poland), October 27-28, 2005.

IC04 F. Principe, C. Terzi, M. Luise, and M. Casucci, "SOFT-REC: a GPS/EGNOS Software Receiver," in *Proc. 14th IST Mobile & Wireless Communication Summit*, Dresden (Germany), June 19-23, 2005.

IC05 F. Principe, C. Terzi, M. Luise, and M. Casucci, "SOFT-REC: a Low-Cost GPS Receiver Following the Software Radio Paradigm," in *Proc. NAVITEC 2004*, ESTEC Noordwijk (The Netherlands), December 8-10, 2006.

Technical Reports

TR01 G. Bacci, F. Principe, and M. Luise, *SOFT-REC: GPS/EGNOS Software Receiver. Technical Note: Verification of User Requirement 8 [UR8].* Dip. di Ingegneria dell'Informazione - University of Pisa, Pisa (Italy), May 6, 2005.

TR02 F. Principe, G. Bacci, and M. Luise, *SOFT-REC: GPS/EGNOS Software Receiver. Technical Note: Signal Processing and Navigation Algorithms*. Dip. di Ingegneria dell'Informazione - University of Pisa, Pisa (Italy), July 15, 2005.

TR03 F. Principe, G. Bacci, and M. Luise, *SOFT-REC: GPS/EGNOS Software Receiver. Technical Note: Signal Processing and Navigation Algorithms Software Description*. Dip. di Ingegneria dell'Informazione - University of Pisa, Pisa (Italy), July 15, 2005.

TR04 F. Principe, M. Luise, and L. Fanucci, *GREAT (Galileo Receiver Advanced Testbed). GARDA Receiver Architecture*. Dip. di Ingegneria dell'Informazione - University of Pisa, Pisa (Italy), January 10, 2005.