Resource-efficient parallel acquisition architectures for modernized GNSS signals

THÈSE N^{O} 6190 (2014)

PRÉSENTÉE LE 26 JUIN 2014

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE D'ÉLECTRONIQUE ET TRAITEMENT DU SIGNAL
PROGRAMME DOCTORAL EN MICROSYSTÈMES ET MICROÉLECTRONIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jérôme LECLÈRE

acceptée sur proposition du jury:

Prof. H. P. Herzig, président du jury
Prof. P.-A. Farine, Dr C. Botteron, directeurs de thèse
Prof. R. Garello, rapporteur
Dr C. Macabiau, rapporteur
Prof. Y. Perriard, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2014

Chaque fois que j'utilise la raison,
chaque fois que j'utilise la logique,
je suis très pessimiste.
Quand j'écoute mon coeur,
quand j'écoute ma foi
- et j'ai la foi en l'humanité -
alors je deviens très optimiste.

— Jacques-Yves Cousteau

Acknowledgements

If this Ph.D. thesis has been completed, it is thanks to many people, that I would like to thank today.

I first thank Prof. Pierre-André Farine, for giving me the opportunity to work in the ESPLAB and offering very good conditions of work.

I sincerely thanks my supervisor, Cyril Botteron, for his time, for his advice, his suggestions, his reviews, and for giving me the freedom to explore my ideas. It was a real pleasure to work with him.

I would like to thank the colleagues from ESPLAB, especially the GNSS group, Youssef, Aleksandar, Vincenzo, Miguel, Marcel and Grégoire, for all the interesting discussion, and my first officemate Patrick Stadelmann, for all the music I discovered thanks to him and for the many questions I asked him during these years.

I spare a thought for Paulo, Sébastien, Myriam, Leslie and Sébastien that made the ION GNSS 2012 a so nice conference. And I must confess that I regret not meeting you again in other conferences. I have a special thought for Myriam, because I have really appreciated all the discussions, the exchanges, the questions, and I thank her to have review this thesis (and also some of my papers). I am really happy to have met her, and to have been able to discuss very technical points about GNSS and acquisition. I am also pleased that our common work will materialize in a near future.

I would like to thank also Henri Nussbaumer for his kindness, his time, and his useful advices. In a certain way, it's thanks to him that I am now raring to explore the fascinating worlds of modular arithmetic and number theory. I also thanks Rick Lyons for his kindness, and I am impatient to share a drink during my next travel to USA.

I also thank the people from the Altera forum, which helped me a lot during the development of my GPS receiver.

I thank Virginie, because her love and her sweet nothings were really important during these last months far from each other.

Et pour finir, je remercie ma mère, car c'est grâce à elle si j'en suis là aujourd'hui. Qui l'aurait crû il y a 10 ou 15 ans ?

Abstract

The acquisition of global navigation satellite system (GNSS) signals is an extremely computationally intensive task. This explains that the first GPS receivers needed a very long time to obtain a position. Thanks to technological advances, it is now possible to use highly parallel implementations and efficient algorithms based on the fast Fourier transform (FFT), and thus reduce significantly the processing time. Indeed, today, it takes less than a second to detect a GPS L1 C/A signal with a clear view of the sky. However, this case does not correspond to all the situations, and does not mean that the research on this topic is completed. For example, it is always necessary to reduce the power consumption of GNSS devices embedded in portable electronics equipment to further improve their battery autonomy. Sometimes, it is necessary to detect very weak signals, such as in space applications because of the long distances and the bad geometry, or with receivers embedded in smartphones where the antenna must meet aesthetic criteria, which leads to poor performance in terms of gain. And finally, the recent introduction of new GNSS signals will lead to better performance, but at the same time will require a much more complex signal processing. Therefore, it is still necessary to find new algorithms in order to meet the current needs of the society and scientists.

The goal of this Ph.D. thesis has been to search algorithms to reduce the complexity of the acquisition, in order to reduce the processing time or the resources used, depending on the context. The research has focused on the computation of the correlation using FFTs, and on reducing its complexity by exploiting the characteristics of the GNSS signals. This research has been performed with a hardware implementation in mind rather than a software implementation, because this Ph.D. thesis started with a project where the goal was to develop a GPS receiver on a programmable circuit (more specifically on a field programmable gate array, or FPGA).

In this thesis, first we show simple methods to reduce the complexity of the FFT or of a correlation computed by FFT (or of a convolution) on Altera FPGAs. In particular the methods we propose allow a significant reduction of the memory resources. Moreover, the application of these methods is not restricted to GNSS signals, but in fact apply to any other systems computing FFTs, convolutions, or correlations, since no assumption is made on the signals.

Afterwards, we focus on the acquisition of modern signals having a secondary code using the parallel code search, and especially on the acquisition of the GPS L5 signal. Two cases are considered :

Abstract

First, we discuss the acquisition where the correlation is performed over one period of the primary code. This case is useful if we want to receive a modern GNSS signal and a fast processing is preferred to a high sensitivity for example, or if we want to use one period of the secondary code (which is longer than a period of the primary code), but it is not possible to compute directly so large FFTs. Starting from an existing solution, two algorithms are proposed in order to reduce the complexity under some conditions (e.g. the FFT length must be a power of two).
Second, we discuss the case where the correlation is performed over one period of the secondary code. We seek to reduce the complexity by exploiting the fact that the tiered code contains a repetition of the primary code, and that the secondary code is short and common to all the satellites (for the considered signal). In search for an exact algorithm, we found an approximation that reduces the complexity in exchange of a reduction of the signal-to-noise ratio (SNR).

Keywords : Acquisition, Algorithm, Complexity, Convolution, Correlation, fast Fourier transform, FFT, FPGA, Hardware receiver, Implementation, Galileo, Global navigation satellite system, Global positioning system, GNSS, GPS, Parallel code search, Sensitivity.

Résumé

L'acquisition des signaux GNSS est une tâche extrêmement gourmande en calculs. Ceci explique que les premiers récepteurs GPS obtenaient une position après un temps très long. Grâce aux progrès technologiques, il est devenu possible d'utiliser des implémentations hautement parallèles et des algorithmes efficaces basés sur la transformée de Fourier rapide (FFT), et donc de réduire le temps de calcul de manière significative. En effet, à l'heure actuelle, détecter un signal GPS L1 C/A avec une vue dégagée du ciel prend moins d'une seconde. Cependant, ce cas ne s'applique pas à tous les contextes, et ne signifie pas que la recherche sur ce sujet soit terminée. Entre autres, il y a toujours la nécessité de réduire la consommation des appareils GNSS intégrés aux équipements portables afin d'améliorer l'autonomie. Il est parfois nécessaire de détecter des signaux très faibles, comme pour les applications spatiales à cause des longues distances et de la géométrie défavorable, ou avec les récepteurs intégrés aux téléphones portables où l'antenne doit répondre à des exigences esthétiques, ce qui est fortement préjudiciable pour le gain de l'antenne. Et pour finir, l'introduction récente de nouveaux signaux GNSS décuple les possibilités et les performances mais rend également les traitements beaucoup plus complexe. Il est donc toujours nécessaire de trouver de nouveaux algorithmes afin de répondre aux besoins actuels de la société et des scientifiques.

Le but de ce doctorat a donc été de rechercher des algorithmes qui permettent de réduire la complexité de l'acquisition, afin de réduire le temps de calcul ou les ressources utilisées, selon le contexte. La recherche s'est articulée principalement autour de l'opération de corrélation calculée par FFT, et des moyens d'exploiter les caractéristiques des signaux GNSS afin d'en réduire la complexité. Cette recherche a été effectuée en ayant à l'esprit plutôt une implémentation matérielle que logicielle des algorithmes, car ce doctorat a débuté avec un projet dont le but était de développer un récepteur GPS sur un circuit programmable (FPGA).

Dans cette thèse, dans un premier temps des méthodes simples sont présentées pour réduire la complexité d'une FFT ou d'une corrélation par FFT (ou d'une convolution) sur des FPGAs de chez Altera. En particulier, ces méthodes permettent une réduction significative de la mémoire nécessaire. De plus, l'application de ces méthodes n'est pas limitée aux signaux GNSS, mais s'applique à tout système calculant des FFT, des convolutions ou des corrélations, car elles n'utilisent aucune propriété concernant les signaux.

Ensuite, nous nous concentrons sur l'acquisition des nouveaux signaux comportant un code secondaire en utilisant la recherche parallèle du code, et en particulier sur l'acquisition du signal GPS L5. Deux cas sont étudiés :
Tout d'abord, celui où la corrélation est calculée sur une période du code primaire. Ce cas est utile si l'on souhaite par exemple recevoir un des nouveaux signaux GNSS et que la rapidité est privilégiée sur la performance en matière de sensibilité, ou si l'on souhaite utiliser une période du code secondaire (qui est plus longue qu'une période du code primaire) mais qu'il n'est pas possible de calculer des FFT de si grande longueur directement. Partant d'une solution existante, deux algorithmes sont proposés afin de réduire la complexité sous certaines conditions (par exemple que la longueur des séquences pour les FFT soit une puissance de deux).
Ensuite, nous étudions le cas où la corrélation est calculée sur une période du code secondaire directement. En exploitant le fait que le code local complet contient une répétition du code primaire et que le code secondaire est court et commun à tous les satellites (pour le signal considéré), nous essayons de réduire la complexité. À défaut d'avoir trouvé une méthode de calcul exact, nous proposons une approximation qui permet de réduire la complexité en contrepartie d'une baisse du rapport signal sur bruit.

Mots-clés : Acquisition, Algorithme, Complexité, Convolution, Corrélation, FFT, FPGA, Implémentation, Galileo, GNSS, GPS, Récepteur matériel, Recherche parallèle du code, Sensibilité, système de positionnement par satellites, transformée de Fourier rapide.

Contents

List of Figures

List of Tables

List of acronyms

ADC	Analog-to-digital converter
AGC	Automatic gain control
AWGN	Additive white Gaussian noise
BOC	Binary offset carrier
BPSK	Binary phase shift keying
CAF	Cross ambiguity function
CBOC	Composite binary offset carrier
CDMA	Code division multiple access
CRT	Chinese remainder theorem
DFT	Discrete Fourier transform
DLL	Delay locked-loop
FFT	Fast Fourier transform
FLL	Frequency locked-loop
FPGA	Field programmable gate array
GEO	Geostationary Earth orbit
GNSS	Global navigation satellite system
GPS	Global positioning system
IDFT	Inverse discrete Fourier transform
IFFT	Inverse fast Fourier transform
LEO	Low Earth orbit
LNA	Low noise amplifier
MEO	Medium Earth orbit
NCO	Numerically controlled oscillator
QPSK	Quadrature phase shift keying
PCS	Parallel code search
PFS	Parallel frequency search

PLL	Phase locked-loop
PRN	Pseudo-random noise
SNR	Signal-to-noise ratio
SS	Serial search
TMBOC	Time modulated binary offset carrier
TTFF	Time to first fix

Mathematical notations

Notations of signals

A continuous signal is denoted $x(t)$, where t represents the time. A discrete signal is denoted $x(nT_S)$, where n is an integer and T_S is the sampling period. If the knowledge of the sampling frequency is not needed, a discrete signal is then denoted x_n. The same notation is used to denote the sequence x_n and the nth sample of this sequence, since the context is usually clear enough to prevent any confusion. The transform of a sequence is denoted with a capital letter, for example the discrete Fourier transform of x_n is denoted X_k, and the z transform of x_n is denoted $X(z)$. Vectors are denoted with small letters in boldface, and matrices are denoted with capital letters in boldface. Below is a summary of the notations used.

$x(t)$	Continuous signal
$x(nT_S)$	Discrete signal sampled with a period T_S
x_n	Discrete signal, or nth sample of the sequence
X_k	Discrete Fourier transform of the sequence x_n
$X(z)$	z transform of the sequence x_n
$\mathbf{x}$	Column vector corresponding to the sequence x_n
$\mathbf{X}$	Matrix generated from the sequence x_n
$\mathbf{x}^*$	Conjugate of the vector $\mathbf{x}$
$\mathbf{X}^*$	Conjugate of the matrix $\mathbf{X}$
$\mathbf{X}^{\mathrm{T}}$	Transpose of the matrix $\mathbf{X}$

Rule for subscripts

A subscript is written in capital letters if :

1. the symbol corresponds to an element fixed by design (e.g. f_{L1} for the L1 carrier frequency, f_S for the sampling frequency, f_{REF} for the reference frequency).

2. the subscript corresponds to an acronym (e.g. f_{LO} for the local oscillator frequency).

A subscript is written in lower case if :

1. the symbol describes a time-varying element (e.g. s_e for the signal emitted, s_b for the signal in baseband, c_p for the primary code).
2. the symbol describes an element whose actual value differs from the theoretical one (e.g. f_s for the actual sampling frequency, f_{ref} for the actual reference frequency).

List of symbols

α	Normalized relative velocity between the emitter and the receiver
β	Accuracy of the reference oscillator
δf	Frequency step in the acquisition (Hz)
$\delta\tau$	Code step in the acquisition (chip)
Δf_i	Difference between theoretical and actual intermediate frequency (Hz)
η	Noise
π	Ratio between the circumference and the diameter of a circle (≈ 3.14)
σ	Standard deviation of a noise (σ^2 corresponding to the noise power)
τ	Delay (s or chip)
$\widehat{\tau}$	Estimation of the delay (s or chip)
φ	Phase of a carrier (rad)
B	Bandwidth of the front-end filter (Hz)
c	Speed of light, equal to 299 792 458 m/s
$c(t)$	Spreading code
$c_i(t)$	Code used for the data channel
$c_q(t)$	Code used for the pilot channel
$c_p(t)$	Primary code
$c_{p,i}(t)$	Primary code used for the data channel
$c_{p,q}(t)$	Primary code used for the pilot channel
$c_s(t)$	Secondary code
$c_{s,i}(t)$	Secondary code used for the data channel
$c_{s,q}(t)$	Secondary code used for the pilot channel
$d(t)$	Data signal, which can take as value $+1$ or -1
e	Base of the natural logarithm (≈ 2.72)
f_b	Baseband frequency (Hz)
$\widehat{f}_b$	Estimation of the baseband frequency (Hz)
f_d	Doppler frequency (Hz)

f_I	Theoretical intermediate frequency in a GNSS receiver (Hz)
f_i	Actual intermediate frequency in a GNSS receiver (Hz)
f_L	Carrier frequency in the L band (1 to 2 GHz)
f_{L1}	Carrier frequency of the GPS L1 and Galileo E1 signals, equal to 1575.42 MHz
f_{L2}	Carrier frequency of the GPS L2C signal, equal to 1227.60 MHz
f_{L5}	Carrier frequency of the GPS L5 and Galileo E5a signals, equal to 1176.45 MHz
f_{LO}	Local oscillator frequency in a GNSS receiver (Hz)
f_{REF}	Theoretical reference frequency in a GNSS receiver (Hz)
f_{ref}	Actual reference frequency in a GNSS receiver (Hz)
f_S	Theoretical sampling frequency in a GNSS receiver (Hz)
f_s	Actual sampling frequency in a GNSS receiver (Hz)
h_n	Local code used for a correlation
j	Imaginary number, equal to $\sqrt{-1}$
k	Index used for discrete signals
k_B	Boltzmann constant, approximately equal to 1.380 648 8 10^{-23} J/K
m	Index used for discrete signals
n	Index used for discrete signals
N_0	One-sided noise power spectral density
N_A	Number of samples used by the accumulator in the PFS method
N_B	Number of branches in the implementation of an acquisition channel
N_C	Number of samples corresponding to a coherent integration time T_C
N_{CB}	Number of code bins in the acquisition
N_{FB}	Number of frequency bins in the acquisition
N_{FFT}^{PCS}	Length of the FFT in the parallel code search
N_{FFT}^{PFS}	Length of the FFT in the parallel frequency search
N_{NC}	Number of non-coherent accumulations
N_P	Number of code periods during the coherent integration time
p_n	Local sequence corresponding to the primary code
P_e	Power of an emitted signal (W)
P_r	Power of a received signal (W)
P_{PCS}	Parallelization of the parallel code search implementation
P_{PFS}	Parallelization of the parallel frequency search implementation
P_{SS}	Parallelization of the serial search implementation
$r(\widehat{f}_b, \widehat{\tau})$	Cross ambiguity function
s_n	Local sequence corresponding to the secondary code

$s_b(t)$	Signal in baseband
$s_e(t)$	Signal emitted by a GNSS satellite
$s_r(t)$	Signal at the receiver antenna
$sc_i(t)$	Sub-carrier used for the data channel
$sc_q(t)$	Sub-carrier used for the pilot channel
t	Time (s)
T_C	Coherent integration time (s)
T_{EFF}	Effective temperature of the entire front-end (K)
T_E	Time to explore the entire acquisition search space (s)
T_S	Theoretical sampling period in a GNSS receiver (s)
T_s	Actual sampling period in a GNSS receiver (s)
$.^u$	Specific to the satellite u
v	Relative velocity between a satellite and the receiver
x_n	Incoming signal used for a correlation
y_n	Output signal of a correlation
z	Variable used with polynomials

Introduction

Genesis of this thesis

I started my Ph.D. thesis working on a European project, where my objective was to develop a high-sensitivity GPS receiver with a fast time-to-first-fix (TTFF) implemented into an FPGA.

Since one of the target was to obtain a fast TTFF, I focused my attention on the acquisition, the most time consuming operation when we turn on a GPS receiver. What I was looking for, a comparison between different acquisition methods for an FPGA implementation, was not available in the literature, thus I performed this comparison myself.

After this comparison, I chose one of the best methods, with the intuition that it was possible to obtain more efficient implementations. This was the starting point of the *real* research.

Motivations

As mentioned above, the acquisition is a computational intensive task. The technology improvements have allowed the use of efficient and highly parallel algorithms, so that it is now possible to acquire a sufficient number of satellite signals in less than a second. However, there are still many reasons to perform additional research on this topic :

1. Although the power consumption is not so critical for GNSS receivers for cars, boats, or planes, this is highly critical for embedded systems, such as mobile phones, satellites, or robots.

2. There are applications where the power of the GNSS signals received is very low, which requires much more processing to detect the signals. This can be due to the environment, like for indoor positioning where there are obstacles, or like for space navigation where the main lobe of the GNSS satellite antenna may not be in the direction of the receiver and where the distances may be much longer. It can also be due to the receiver components, like with mobile phones where the antenna must meet aesthetic requirements that lead to poor performance (the gain can be 10 dB lower compared to an ideal 3 dBi antenna, which means that 90 % of the power is lost before reaching the front-end (van Diggelen [2009], p. 216)).

3. Several new signals are now available, and although they offer better performance, the complexity for the acquisition is much higher, especially because of a higher chipping rate, the use of longer codes, and the presence of a secondary code (van Diggelen [2014]).

4. With four GNSS constellations in the sky soon, the performance of GNSS receiver will be improved significantly (e.g. the number of satellites in view will increase and the geometry will be improved, leading to a better positioning accuracy and availability). However, each constellation provides signals with different characteristics, it is thus a challenging task to find implementations that can process several signals in an efficient way (e.g. by sharing some processing units).

For all these reasons, the research of more efficient algorithms for the acquisition is still necessary. Here, by more efficient, we mean that the complexity of the algorithms should be reduced, which can lead to different improvements according to the objective. For instance, it can result in a reduction of the processing time (thus allowing to get a position more quickly), or in a reduction of the resources used (e.g. memory), leading to a reduction of the energy consumption.

Therefore, during this Ph.D. thesis, I focused my research on the acquisition, and on how to reduce its complexity. The acquisition includes additional topics, such as ways to increase the sensitivity (coherent vs. non-coherent vs. differential integration, combination of data and pilot channels), or the methods of detection. Here, these topics are not addressed, and the focus is only on the computation algorithms, and more specifically on the ways to compute the correlation of two signals through FFTs in the GNSS context.

I specify in the GNSS context because the aim was to use the specificities of the GNSS signals and the application context to find more efficient algorithms, and not to find the solution of general problems, because the question of the fast computation of the DFT and of the convolution has already been discussed extensively since the 1970s (see e.g. Winograd [1980], Nussbaumer [1982], Burrus and Parks [1985], Garg [1998], Blahut [2010]).

The focus is also more on hardware receivers than on software receivers, because I started my Ph.D. thesis implementing a hardware receiver. Nevertheless, we discuss also the applicability of some of the proposed algorithms for software receivers.

Mathematical tools used during this thesis

During the thesis, different mathematical tools have been used, because the correlation between two discrete signals can be expressed in various ways. The correlation is usually expressed in the time domain. But it can also be expressed as the product of two polynomials (through the use of the z transform for example), or using matrices, which involves circulant, skew-circulant, Toeplitz and Hankel matrices. And finally, the correlation is deeply related to the discrete Fourier transform.

Then, there are also different ways to manipulate the signals, as for example using a downsampling or a segmentation, or applying the Chinese remainder theorem.

Outline

This thesis is separated in two parts. The first part, which includes Chapters 1, 2 and 3, introduces GNSS signals and receivers with a closer look on the acquisition block, and presents a comparison of different known methods for the acquisition. The second part, which includes Chapters 4, 5 and 6, presents the proposed algorithms to reduce the complexity of the acquisition in different contexts.

Chapter 1 introduces some basic notions about GNSS, by describing the concepts, the current systems, the GNSS signals, and the different elements of a GNSS receiver. More importantly, this chapter introduces the model used for the discrete GNSS signals. Chapter 2 focuses on the acquisition, describing the operation performed, presenting different known methods, and briefly introducing some concepts that are part of the acquisition but not considered in this thesis. The content of these two chapters is not really new, but is presented in a rigorous way. In Chapter 3, and using results provided in Appendix C, we propose a new comparison framework to compare the implementations of the main GNSS signals acquisition architectures on FPGAs.

Using known algorithms, Chapter 4 presents some implementations to reduce the complexity of the FFT and of the correlation implemented on Altera FPGAs, using the Altera FFT. Chapter 5 discusses the problem of the acquisition in presence of sign transitions due to a secondary code (or data). We present two algorithms to reduce the complexity. These algorithms compute the output exactly and are thus not approximations. However, they are efficient only under certain conditions. Chapter 6 discusses the problem of the high sensitivity acquisition using the secondary code. We discuss implementations to use smaller FFTs (which will bring us back the Chapter 5), and then implementations to reduce the complexity (which will lead to approximations).

Chapter 7 concludes this work and provides some comments for future research.

Several appendices complete this thesis. Appendix A summarizes the definition of some operations (DFT, z transform, convolution, correlation) in different ways (time domain, matrix view, z transform view), and provides the relations between them. Appendix B gives some tips to reduce the complexity when we deal with FFT and correlation, which can be useful for GNSS receivers. Finally, Appendix C contains the details of the estimation of the FPGA resources for the implementations discussed in Chapter 3.

Overall, I tried to gather in this manuscript all the information I learnt and that someone would need to continue this research in the same direction.

Contributions

During this thesis, several articles have been published. The following articles are directly related to this thesis.

Chapter 3 and Appendix C
J. Leclère, C. Botteron, and P.-A. Farine. Resource and performance comparisons for different acquisition methods that can be applied to a VHDL-based GPS receiver in standalone and assisted cases. In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 745–751, May 2010.

J. Leclère, C. Botteron, and P.-A. Farine. Comparison framework of FPGA-based GNSS signals acquisition architectures. *IEEE Transactions on Aerospace and Electronic Systems*, 49(3):1497–1518, July 2013b.

Chapter 4 and Appendix B
J. Leclère, C. Botteron, and P.-A. Farine. Improving the performance of the FFT-based parallel code-phase search acquisition of GNSS signals by decomposition of the circular correlation. In *Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012)*, pages 1406–1416, September 2012.

Chapter 5
J. Leclère, C. Botteron, and P.-A. Farine. Modified parallel code-phase search for acquisition in presence of sign transition. In *International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–6, June 2013a.

J. Leclère, C. Botteron, and P.-A. Farine. Acquisition of modern GNSS signals using a modified parallel code-phase search architecture. *Signal Processing*, 95(0):177–191, February 2014.

The following articles are articles I participated on, but not directly related to this thesis.

K. Sheridan, T. Dyjas, C. Botteron, J. Leclère, F. Dominici et al. An assisted-GNSS solution for demanding road applications using the EGNOS data access system (EDAS). In *European Navigation Conference on Global Navigation Satellite Systems (ENC-GNSS)*, pages 1–10, October 2010.

K. Sheridan, D. Wells, C. Botteron, J. Leclère, F. Dominici et al. An assisted-GNSS solution for demanding road applications using the EGNOS data access system (EDAS). In *Toulouse Space Show'10*, pages 1–9, June 2010.

K. Sheridan, T. Dyjas, C. Botteron, J. Leclère, F. Dominici et al. Demands of the road - An assisted-GNSS solution uses the EGNOS data access service. *GPS World*, 22(3):28–35, March 2011.

Y. Tawk, A. Jovanovic, J. Leclère, C. Botteron and P.-A. Farine. A new FFT-Based algorithm for secondary code acquisition for Galileo signals. In *IEEE Vehicular Technology Conference (VTC Fall)*, pages 1–6, September 2011.

Y. Tawk, A. Jovanovic, P. Tomé, J. Leclère, C. Botteron et al. A new movement recognition technique for flight mode detection. *International Journal of Vehicular Technology*, pages 1–18, 2013.

C. Botteron, N. Dawes, J. Leclère, J. Skaloud, S.V. Weijs et al. Soil moisture & snow properties determination with GNSS in alpine environments: challenges, status, and perspectives. *Remote Sensing*, 5(7):3516–3543, July 2013.

S. Reboul, C. Botteron, M.A. Ribot Sanfelix, G. Stienne et al. Normalized GNSS interference pattern technique. In *URSI Commission F Microwave Signatures 2013 - Specialist Symposium on Microwave Remote Sensing of the Earth, Oceans, and Atmosphere*, page 1, September 2013.

GNSS signals and acquisition Part I

1 Some basics about GNSS

1.1 General principle of GNSS

The positioning in GNSS is based on trilateration, i.e. the measurement of distances from references whose positions are known. An example in two-dimension is given Fig. 1.1. If the receiver R knows that its distance from the reference S1 is d_1, it knows that its position is on the circle of radius d_1 and centered at S1. Moreover, if then the receiver knows that its distance from the reference S2 is d_2, it knows that its position may be on two points, namely the intersection of the two circles. Finally, the same information for the third reference S3 will indicate the true receiver position.

In GNSS, the references are the GNSS satellites, the distances are determined by measuring the travel time of the signals, and we have three dimensions instead of two. Therefore, the measure of the distance from a first satellite reduces the possibilities for the receiver position to a sphere; a second measure reduces the possibilities to a circle; a third measure reduces the possibilities to two points; and a fourth measure reduces the possibilities to one point, the receiver position. Consequently, the signals from at least four GNSS satellites are needed to obtain a position. Actually, a GNSS receiver needs at least four satellites, but for additional reasons. After the third measure, there are two possibilities for the position, but one of them is on the surface of the Earth, while the other is on an impossible place (e.g. far below the surface or in space (note that this is an impossible place for a terrestrial receiver, but not for a satellite)). Thus, with a clever algorithm, three satellites would be sufficient. However, the time of arrival of the signals is measured with the clock of the receiver, which has a limited accuracy (whereas GNSS satellites use atomic clocks). An error in the estimate of the travel time results in an error of the satellites position and in the estimates of the distances (in 1 µs the waves travel about 300 m), and thus in the estimate of the receiver position. To resolve this error, an additional measure is needed. Therefore, four satellites are needed to get a position. Nevertheless, a receiver usually uses more than four signals to improve the accuracy.

Regarding the GNSS satellites, they send their current time and their ephemeris (parameters that define their orbit), which allows the receiver to compute their position. Afterwards, to

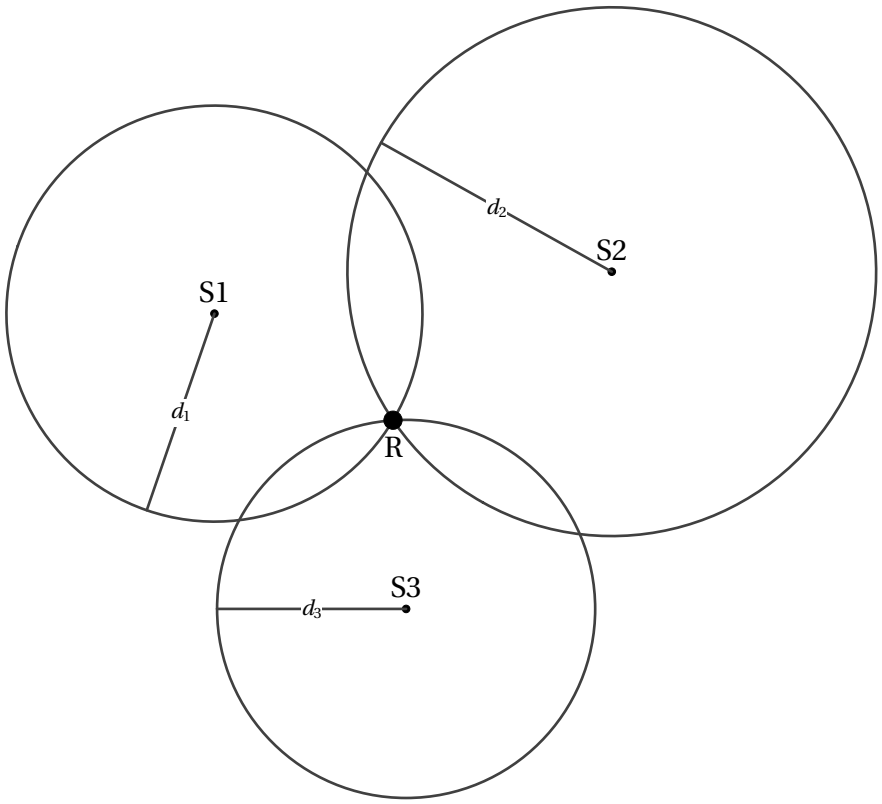


Figure 1.1: Illustration of the trilateration principle in a two-dimension case.

determine the distance from each GNSS satellite, the receiver determines the travel time (time between the reception and the emission of the signal) and multiplies it by the speed of propagation.

For more details, see for example http://www.navipedia.net/index.php/An_intuitive_approach_to_the_GNSS_positioning, or (Kaplan and Hegarty [2005] Chap. 2, El-Rabbany [2006]).

1.2 Overview of the terrestrial GNSSs

A GNSS is composed of three segments : the space segment, the control segment, and the user segment.

The space segment consists in the GNSS satellites themselves, which are in almost circular orbits around the Earth, at an altitude of about 20 000 km in a region called medium Earth orbit (MEO). The constellations rely on 24 to 30 satellites, distributed between several planes. The GNSS satellites send permanently signals in direction of the Earth.

The control segment consists in a network of stations on Earth, which includes a master control station, several stations that monitor the GNSS signals, and some stations that upload new data to the GNSS satellites (e.g. to adjust the orbit parameters of the satellites).

The user segment consists in all the equipments that receive and process the GNSS signals. This thesis, which discusses about the processing of GNSS signals, is thus in this category. For a thorough review of the application of GNSS, see (Jacobson [2007]) and (Gleason and Gebre-Egziabher [2009]).

In the next years, there will be four GNSSs fully operational. Two of them are currently fully operational (GPS and GLONASS), and two of them are in development (Galileo and BeiDou).

The GPS is an American system, started in the late 1970s by the Department of Defense for military applications. Later, in the 1990s, it was open to civil applications, but it's mainly when the selective availability was removed (making the positioning accuracy within 10 m instead of about 100 m (Adrados et al. [2002])) that the GPS has been democratized. Since the beginning, the system is evolving progressively, each new generation of satellites bringing new performances (e.g. using better clock and increasing the life span) or new signals. As of March 2014, the GPS constellation includes 31 satellites (http://www.gps.gov/systems/gps/space), whereas the baseline was 24 satellites.

GLONASS is a Russian system developed approximately at the same time as the GPS. However, due to a lack of maintenance, the system was no more fully operational, until a recent renovation. As of March 2014, GLONASS is fully operational with 24 satellites (http://www.glonass-center.ru/en/GLONASS). As GPS, GLONASS evolves, and the next generation of satellites will include new signals. Until now, GLONASS satellites used FDMA (frequency division multiple access) for the multiple access, i.e. each satellite uses a specific frequency. This is the only GNSS to use this technology, all the others using only CDMA (code division multiple access), where all the satellites use the same frequency but they have a specific code (GLONASS uses also such codes but only for ranging purpose). The new GLONASS signals will be based on CDMA only.

Galileo is a European system in development. As the modernized GPS, the Galileo satellites will transmit several signals for civil applications and governments, including wide band signals. According to the latest estimations, the system should be fully operational in 2018. BeiDou is a Chinese system in development. Most of the signals will be similar to those of Galileo or GPS. According to the latest estimations, the system should be fully operational in 2018. For a summary of the GNSS signals, see (Hegarty [2012]).

1.3 Signals transmitted by GNSS satellites

The GNSS satellites continuously send signals in direction of the Earth. Those signals have three essential components (Langley [1990]) :

- A carrier, which is a sinusoidal signal whose frequency is in the L band (band between 1 and 2 GHz).
- A spreading code, which is a known long binary sequence of $+1$ and -1 specific to each

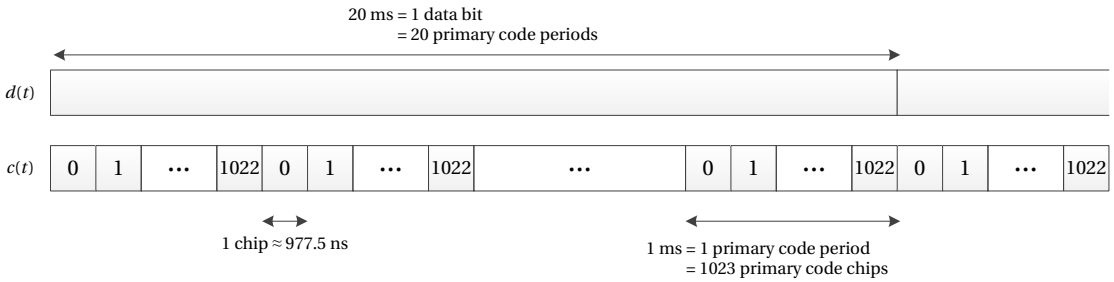


Figure 1.2: Illustration of the data ($d(t)$) and code ($c(t)$) synchronization for the GPS L1 C/A signal. The values inside boxes indicate the chip number.

satellite and transmitted at high rate. This sequence, also called pseudo-random noise (PRN) code, allows precise ranging and lets the satellites to broadcast signals at the same carrier frequency (see e.g. (Viterbi [1995]) for more details about CDMA). The values of a PRN code are usually called chips instead of bits, to emphasize that they do not carry information.

- Navigation data, which is a binary-coded message of value +1 or −1 transmitted at low rate to provide the information necessary for the navigation, such as time and orbital information (called ephemeris). The duration of one data bit is equal or is a multiple of the duration of one period of the PRN code (see Figs. 1.2 and 1.3).

The famous GPS L1 C/A signal has these three components, and the signal emitted by the satellite u is defined as

$$s_e^u(t) = \sqrt{2P_e}\, c^u(t)\, d^u(t)\, \cos(2\pi f_{L1} t + \varphi_e^u), \tag{1.1}$$

where t is the time, P_e is the signal power, $c(t)$ is the PRN code, $d(t)$ is the data, f_{L1} is the L1 carrier frequency, and φ_e is a phase. The synchronization between the code and the data is illustrated in Fig. 1.2. The modulation used for this signal is called binary phase shift keying, or BPSK (Ziemer and Tranter [2008] pp. 403–408).

Modern signals, on the other hand, have introduced some new components (Turunen [2007]) :

- A secondary code, which is a known binary sequence transmitted at low rate. This means that two codes (usually called primary and secondary codes) are combined to form a tiered spreading code (see Fig. 1.3). The secondary code helps, among others, for the synchronization with the data bits.
- A sub-carrier, which is a square wave multiplying each chip of the PRN code. This leads to a new modulation family (binary offset carrier, or BOC) and modifies the spectrum of the signals (Betz [2001]).
- A pilot channel, that includes only the spreading code and the carrier, and not the data. This provides a lot of advantages for the signal detection since it is fully deterministic. For

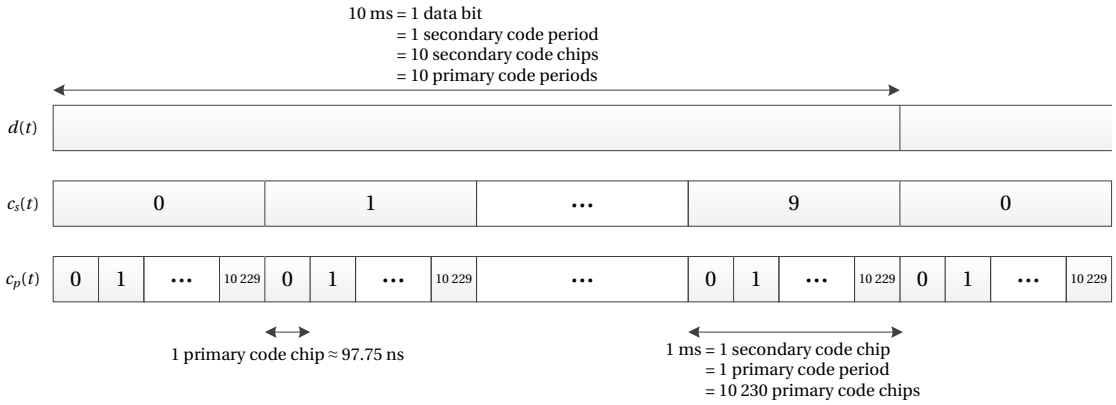


Figure 1.3: Illustration of the data ($d(t)$) and codes ($c_s(t)$ and $c_p(t)$) synchronization for the L5 signal. The values inside boxes indicate the chip number.

some signals (e.g. the GPS L5 and the Galileo E5a and E5b), the data and pilot channels are in quadrature, in this case, if each channel is BPSK modulated, the modulation is a quadrature phase shift keying (QPSK), and the data channel is usually denoted the I channel, and the pilot channel the Q channel. Otherwise, the two channels are transmitted in a different way, like time multiplexing (e.g. TMBOC for the GPS L1C signal) or code multiplexing (e.g. CBOC for the Galileo E1 signal). See (Ávila Rodríguez et al. [2006]) for more details.

So, the signal emitted by a GNSS satellite u when the data and pilot channels are in quadrature is defined as

$$s_e^u(t) = \sqrt{2P_e}\, c_i^u(t)\, d^u(t)\, \cos(2\pi f_L t + \varphi_e^u) + \sqrt{2P_e}\, c_q^u(t)\, \sin(2\pi f_L t + \varphi_e^u), \tag{1.2}$$

where P_e^u is the emitted power on each channel (assuming the same power on both channels), f_L is the carrier frequency in the L band, and $c_i(t)$ and $c_q(t)$ are the PRN codes of the data and pilot channels defined as

$$c_i^u(t) = c_{p,i}^u(t)\, c_{s,i}^u(t)\, sc_i^u(t), \tag{1.3}$$

and

$$c_q^u(t) = c_{p,q}^u(t)\, c_{s,q}^u(t)\, sc_q^u(t), \tag{1.4}$$

where $c_{p,i}$ and $c_{p,q}$ are the primary codes of the data and pilot channels, $c_{s,i}$ and $c_{s,q}$ are the secondary codes of the data and pilot channels, and sc_i and sc_q are the sub-carriers of the data and pilot channels. The synchronization between the codes and the data is illustrated in Fig. 1.3.

A summary of the properties of some GNSS signals is given Table 1.1. It can be seen that there are only two different chipping rates used for the primary code, 1.023 MHz and 10.23

Signal	GPS L1 C/A	Galileo E1		GPS L5		Galileo E5a		Galileo E5b	
		D	P	I	Q	I	Q	I	Q
Carrier frequency (MHz)	1575.42	1575.42		1176.45		1176.45		1207.14	
Modulation	BPSK	CBOC		QPSK		QPSK		QPSK	
Primary code chipping rate (Mchip/s)	1.023	1.023		10.23		10.23		10.23	
Primary code length (chip)	1023	4092		10 230		10 230		10 230	
Primary code length (ms)	1	4		1		1		1	
Secondary code chipping rate (chip/s)	-	-	250	1000		1000		1000	
Secondary code length (chip)	-	-	25	10	20	20	100	4	100
Secondary code length (ms)	-	-	100	10	20	20	100	4	100
Data rate (bit/s)	50	250	-	100	-	50	-	250	-

Table 1.1: Properties of some GPS and Galileo signals.

MHz, and that the primary code length is always a multiple of 1023. Regarding the secondary codes, those for the Q channel of the E5a and E5b signals are specific to each Galileo satellite (there are 100 codes defined), while the other secondary codes are unique (for example the secondary code for the I channel of the E5b signal is $-1\ -1\ -1\ +1$). For details on other signals, the reader can refer to (Hegarty [2012]) or (Ávila Rodríguez [2008]).

The spreading codes used in GNSS have not been randomly chosen, but have been carefully selected according to their auto-correlation (correlation of a signal with itself) and cross-correlation (correlation between two signals) properties. The codes are selected in order to have cross-correlation values as low as possible, and auto-correlation values as low as possible except for one case, when the code is aligned with itself. This is illustrated in Fig. 1.4 for the auto-correlation of some primary codes, in Fig. 1.5 for the cross-correlation of some primary codes, and in Fig. 1.6 of some secondary codes. It can be seen that the longer is the code, the better are the correlation properties.

Regarding the spreading codes composed of a primary and a secondary code, it can be tempting to think that the protection is the addition of the protection of each codes (if it was true, the minimum protection of the L5 spreading code would be about 27 + 14 = 41 dB for

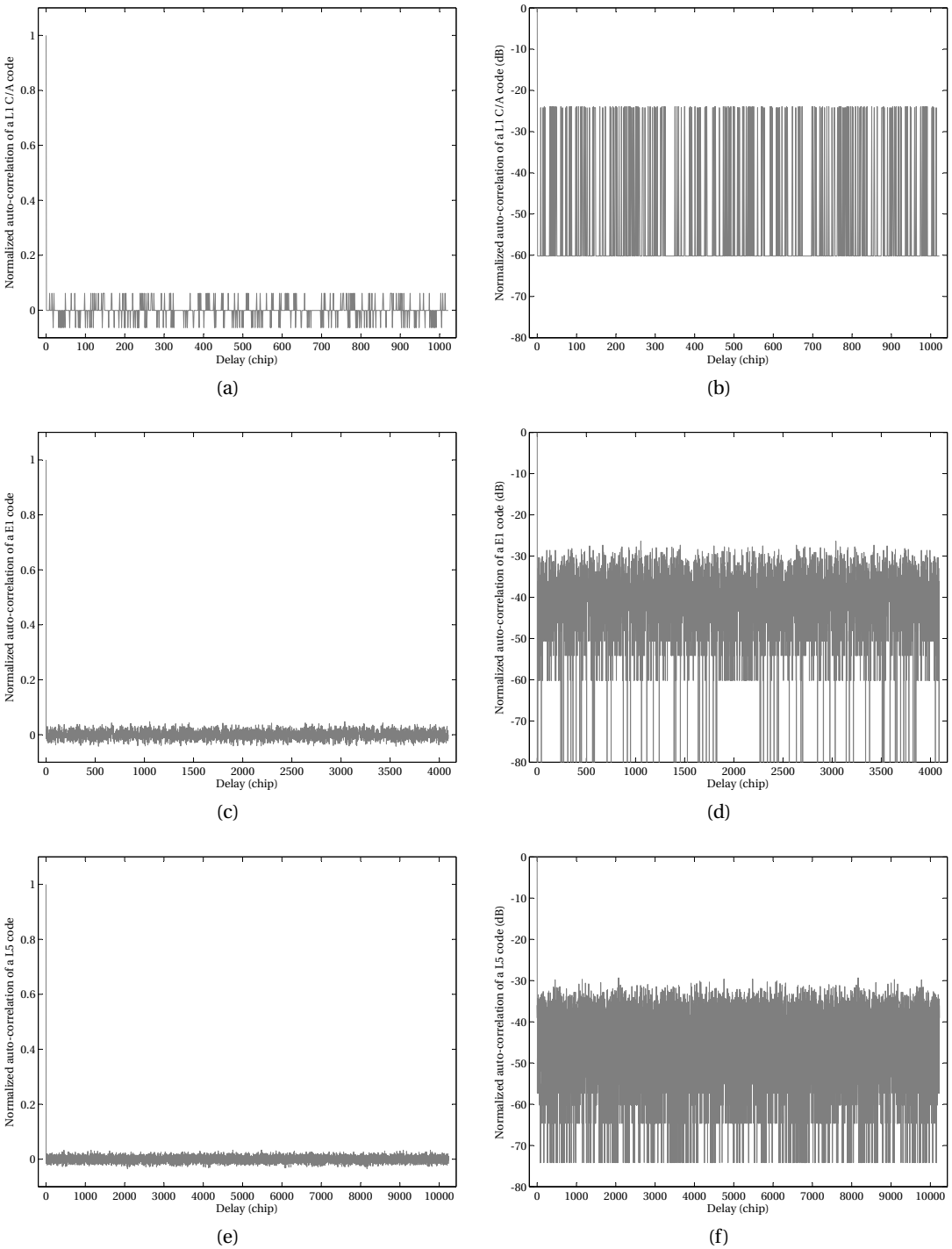


Figure 1.4: Auto-correlation of a L1 C/A code (1023 chips), an E1 code (4092 chips) and a L5 code (10 230 chips).

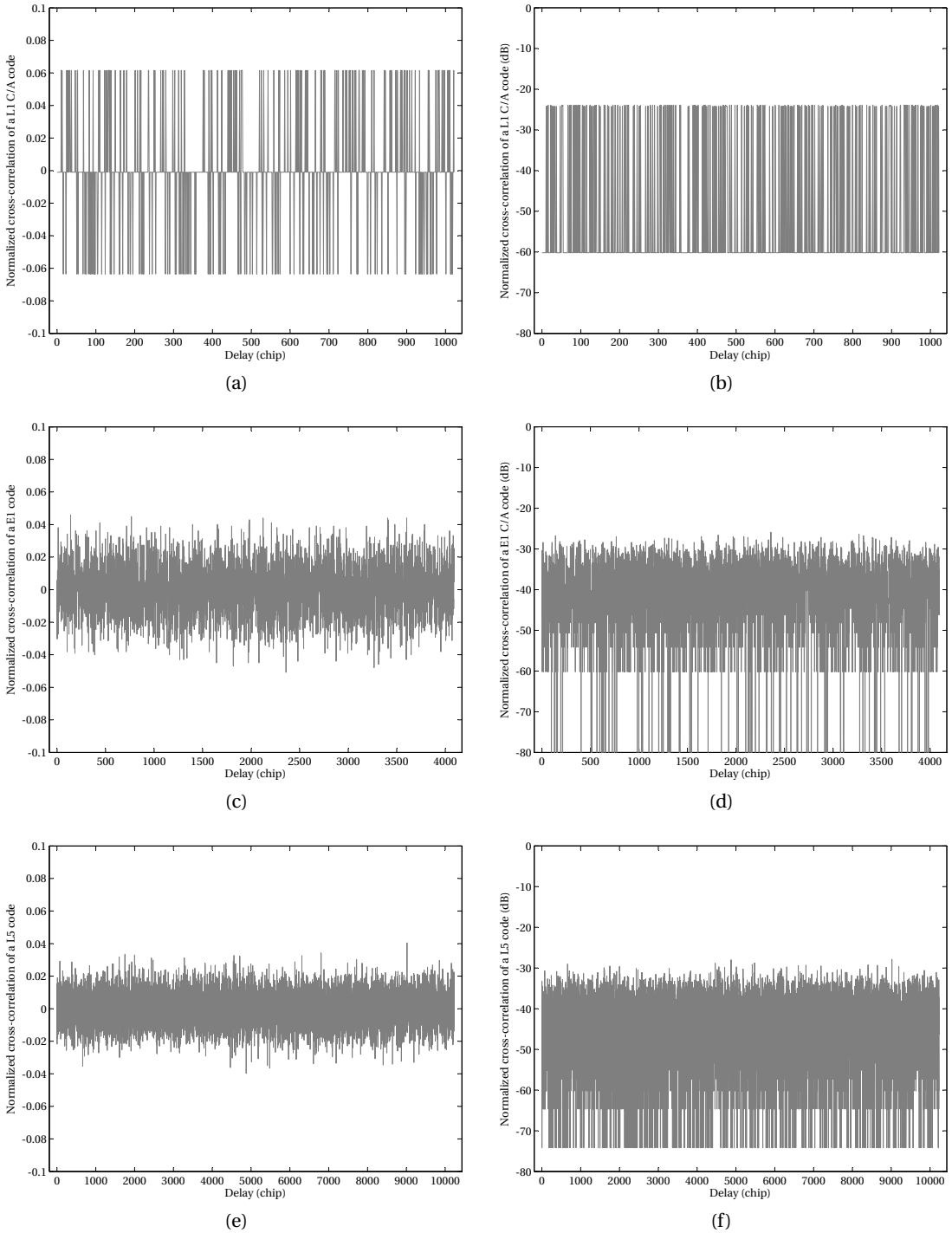


Figure 1.5: Cross-correlation of a L1 C/A code (1023 chips), an E1 code (4092 chips) and a L5 code (10 230 chips).

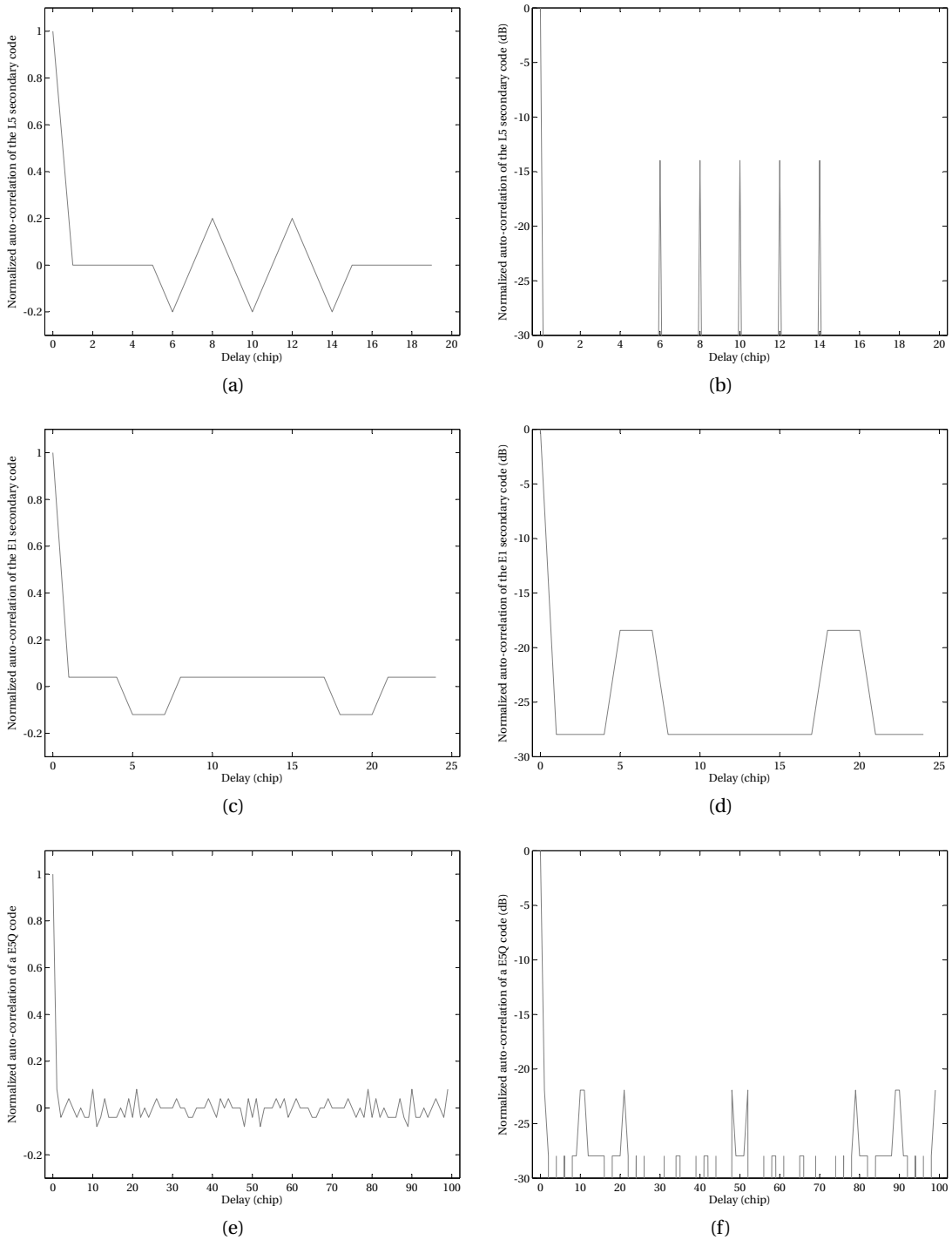


Figure 1.6: Auto-correlation of the L5 secondary code (20 chips), the E1 secondary code (25 chips) and an E5Q secondary code (100 chips).

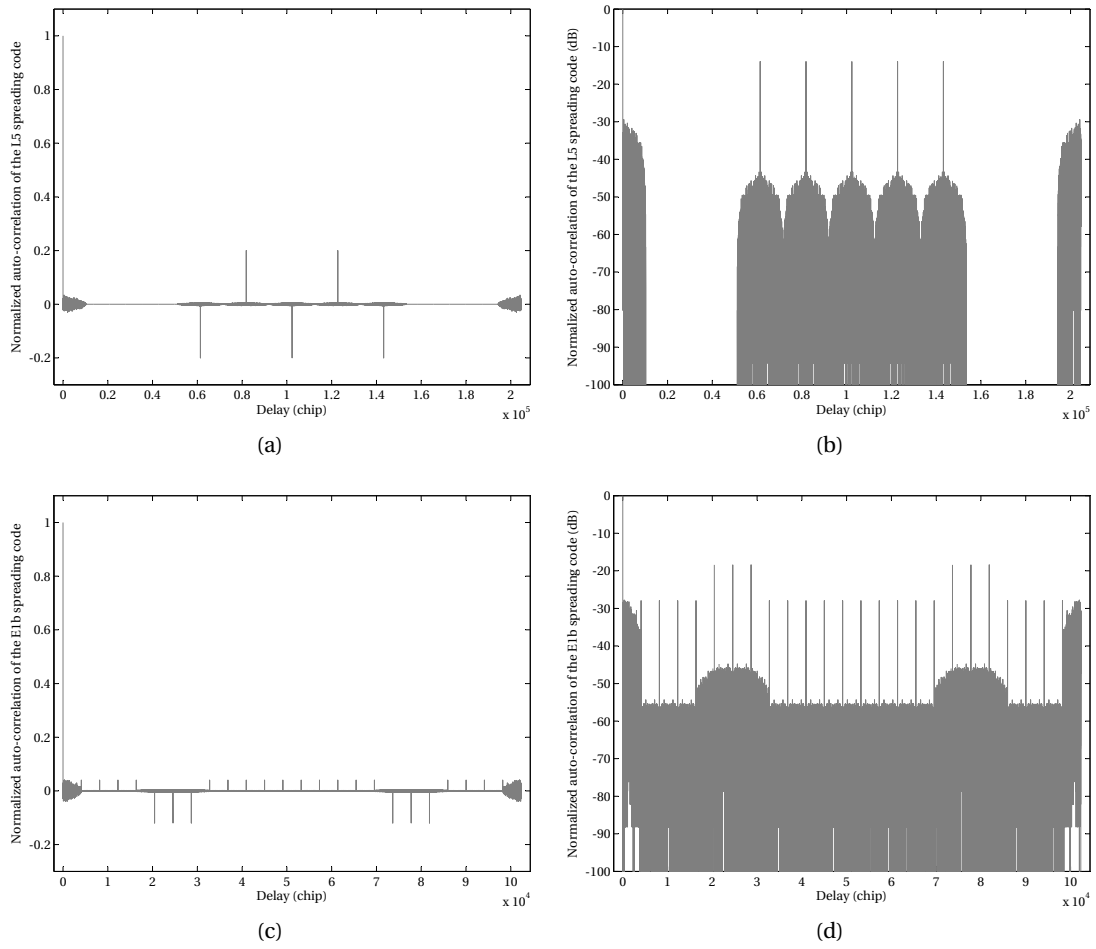


Figure 1.7: Auto-correlation of the L5 spreading code (204 600 chips) and the E1 secondary code (102 300 chips).

example). However, this is not the case as shown in Fig. 1.7. It can be seen that around the correct alignment, the correlation value is higher. Indeed, in this case the secondary code is nearly aligned since a shift of 1 chip of the L5 primary code is equal to a shift of $\frac{1}{10\,230}$ chip of the L5 secondary code, thus the correlation value is mainly due to the primary code. Then, when the secondary code is sufficiently shifted, the correlation value is much lower, except for some delays. These delays correspond to a correct alignment of the primary code, thus in these cases the correlation value is given by the secondary code, which explain the higher values.

Finding a set of codes with very good characteristics is not an easy task. Indeed, the number of possibilities is too large to make an exhaustive search. For example, for a length of 1023 chips, there are 2^{1022} code possible. With an imaginary supercomputer that could check the auto-correlation of 10^{15} codes in one second, it would still require about 10^{275} times the universe age to test all of them. And this does not take into account the research of a set of several codes

with good cross-correlations properties. Therefore, smart approaches have to be used, and today they are known ways to generate some code families having good properties using shift registers (Holmes [2007] Chap. 2). The most well-known codes are probably the Gold codes (Gold [1967]), which are used with the GPS L1 C/A signal. However, other characteristics can be included in the search of good codes, such as the correlation in presence of a transition, or the correlation in presence of a residual carrier (which affects a lot the short secondary code (Macabiau et al. [2003])). See (Soualle et al. [2005]) for more details.

1.4 Space travel

During the travel in space, the GNSS signals are affected by different elements, as well summarized in (MacGougan et al. [2001]). Here we describe two important effects, which are the reasons of the acquisition, and we mention some other effects.

1.4.1 Free space loss

Like every propagating signal, the GNSS signals are affected by a loss of power during the travel. This loss, usually called free space loss, is defined as

$$L_f = \left(\frac{4\pi d f}{c}\right)^2, \tag{1.5}$$

where d is the distance in m, f the carrier frequency in Hz, and c the speed of light in m/s (Ziemer and Tranter [2008], pp. 695–698). This loss can also be expressed in log scale, which gives

$$\begin{aligned} L_f(\text{dB}) &= 10\log_{10}\left(\frac{4\pi d f}{c}\right)^2 \\ &\approx 20\log_{10}(d) + 20\log_{10}(f) - 147.55. \end{aligned} \tag{1.6}$$

Therefore, for the signals at the L1 frequency, the loss is

$$\begin{aligned} L_{f_{L1}}(\text{dB}) &\approx 20\log_{10}(d) + 183.95 - 147.55 \\ &\approx 20\log_{10}(d) + 36.40, \end{aligned} \tag{1.7}$$

and for the signals at the L5 frequency, the loss is

$$\begin{aligned} L_{f_{L5}}(\text{dB}) &\approx 20\log_{10}(d) + 181.41 - 147.55 \\ &\approx 20\log_{10}(d) + 33.86. \end{aligned} \tag{1.8}$$

Signal	GPS L1 C/A	Galileo E1		GPS L5		Galileo E5a		Galileo E5b	
		D	P	I	Q	I	Q	I	Q
Received power (dBm)	−128.5	−130	−130	−127*	−127*	−128	−128	−128	−128

Table 1.2: Minimum received power on Earth (for a 3-dB gain linearly polarized antenna or a unity gain RHCP antenna). *For block III GPS satellites (−127.9 dBm for block IIF satellites.)

Using $d = 20\,200$ km (the approximate altitude of the GPS satellites), this gives

$$\begin{aligned} L_{f_{L1}}(\text{dB}) &\approx 146.11 + 36.40 \\ &\approx 182.50 \end{aligned} \tag{1.9}$$

and

$$\begin{aligned} L_{f_{L5}}(\text{dB}) &\approx 146.11 + 33.86 \\ &\approx 179.97 \end{aligned} \tag{1.10}$$

Of course, the distance traveled by the signals depends also on the position of the receiver on Earth. Thus, according to the position of the receiver, the free space loss is different. However, the pattern of the satellites antenna is designed to partially compensate this (Kaplan and Hegarty [2005] pp. 133–135, van Diggelen [2009] pp. 10–12). In the end, the minimum signal power received on Earth is given in Table 1.2 for different GNSS signals. But in presence of obstacles (e.g. glass, wood, concrete), the signals are strongly attenuated and the power can drop to −160 dBm (van Diggelen [2009] pp. 215–217). Furthermore, with a poor performance antenna (such as in mobile phones), this power can be reduced again by 10 dB to reach −170 dBm, which represents a factor of about 10 000 compared to the nominal value. It will be then the job of the signal processing in the receiver to compensate these losses. Of course, lower is the received signal power, higher is the processing required.

1.4.2 Doppler effect

Another important effect is the Doppler effect. The Doppler effect implies a time compression or expansion of the signals due to the relative motion between a transmitter and a receiver. Thus, a signal $s(t)$ will be seen as $s\big((1+\alpha)\,t\big)$, where $\alpha = \frac{v}{c}$, with v the relative velocity between the emitter and the receiver, and c the speed of light. Consequently, this means that the period T of a periodic signal is divided by $1+\alpha$, and that its frequency f is multiplied by $1+\alpha$, as illustrated in Fig. 1.8. The difference between the received frequency and the emitted frequency, equal to αf, is usually called the Doppler frequency or the Doppler shift.

According to (Tsui [2005] pp. 34–37), the GPS satellites are moving at a speed of about 3874 m/s, and the maximum relative speed between a GPS satellite and a static user on Earth is

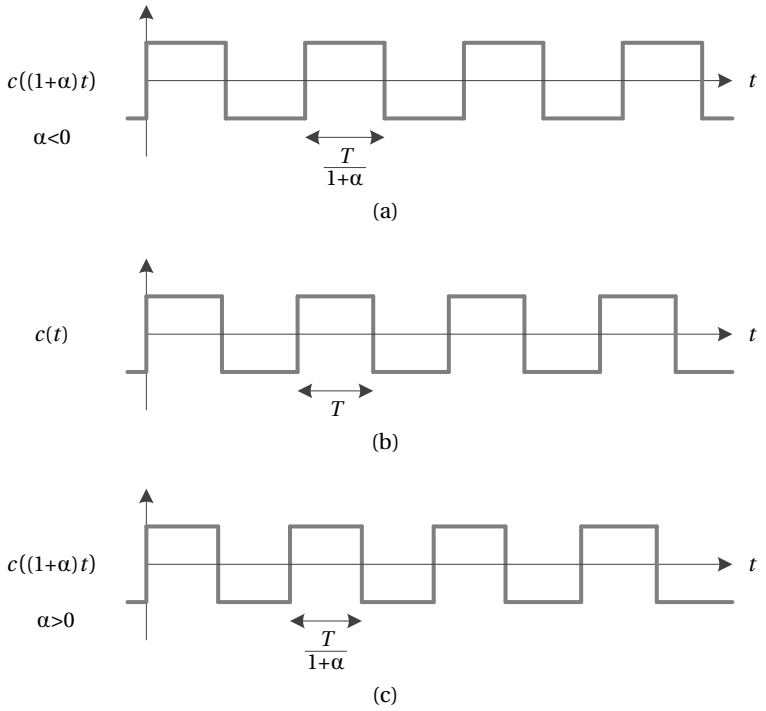


Figure 1.8: Illustration of the Doppler effect on a periodic signal when (a) the emitter and the receiver go away from each other (the relative speed between them is negative, thus $\alpha < 0$); (b) the relative speed between the emitter and the receiver is zero; (c) the emitter and the receiver are getting closer (the relative speed between them is positive, thus $\alpha > 0$).

928.7 m/s. If we follow the same procedure, the relative speed between a Galileo satellite and a static user on Earth is 17/20 the one with GPS (17/20 is the ratio between the altitude of GPS satellites and the altitude of Galileo satellites), i.e. 789.4 m/s. This leads to the Doppler shifts indicated in Table 1.3. It can be seen that the GNSS signals are not equivalent from the Doppler effect point of view. Regarding the code Doppler, even if it seems very low, it still plays an important role as will be shown in Chapter 2.

For a moving receiver, the relative speed between the satellites and the receiver is higher. Table 1.4 provides the maximum supplementary Doppler shift for an aircraft having a speed of 1000 km/h and a spacecraft in low Earth orbit (LEO) having a speed of 7.7 km/s. Note that the total Doppler shift in the case of the LEO is not the sum of the Doppler shifts of Tables 1.3 and 1.4, since Table 1.3 takes into account the geometry for a user on Earth. For example, the maximum carrier Doppler shift measured by a LEO receiver having a speed of 7.7 km/s in (Dion et al. [2008]) was 42 kHz for the L1 C/A signal.

1.4.3 Doppler rate of change

Since the GNSSs are dynamic systems, the relative velocity between the GNSS satellites and the user is changing over the time. According to (Tsui [2005] pp. 39–40), the maximum rate of

Signal	Carrier Doppler (kHz)	Code Doppler (Hz)
GPS L5	3.64	31.7
GPS L2	3.80	3.17
GPS L1	4.88	3.17
Galileo E5a	3.10	26.9
Galileo E5b	3.18	26.9
Galileo E1	4.15	2.69

Table 1.3: Maximum Doppler shift for GNSS signals for a static user.

Signal	Aircraft		Spacecraft	
	Carrier Doppler (kHz)	Code Doppler (Hz)	Carrier Doppler (kHz)	Code Doppler (Hz)
GPS L5	1.09	9.48	30.2	263
GPS L2	1.14	0.95	31.5	26.3
GPS L1	1.46	0.95	40.5	26.3
Galileo E5a	1.09	9.48	30.2	263
Galileo E5b	1.11	9.48	31.0	263
Galileo E1	1.46	0.95	40.5	26.3

Table 1.4: Maximum supplementary Doppler shift for GNSS signals for an aircraft (speed of 1000 km/h) and for a spacecraft (speed of 7.7 km/s).

change of the carrier Doppler shift is about 0.94 Hz/s for a static user on Earth considering the L1 frequency. In fact, the range of values of the rate of change depends on the latitude of the user (van Diggelen [2009] pp. 45–46).

For a moving receiver, the rate of change can be much higher. For example, when a car is on a roundabout, the relative speed between the receiver and the emitter changes rapidly and can imply a change of the carrier Doppler of dozens of Hz in few seconds. The rate of change can be quite high also in space. For example, considering the L1 C/A signal, the rate of change of the carrier Doppler shift can reach at least 62 Hz/s for a receiver on LEO (Dion et al. [2008]), 14 Hz/s for a receiver on a medium Earth orbit (MEO) and 5.5 Hz/s for a receiver on a geostationary Earth orbit (GEO) (Capuano et al. [2013]).

The code Doppler shift is also evolving over the time, although the change is not so significant (with the L1 C/A signal, a rate of change of 0.94 Hz/s for the carrier Doppler means a rate of change of 0.61 mHz/s for the code Doppler). The impact of the rate of change of Doppler is discussed in Chapter 2.

1.4.4 Other effects

There are also other effects, not so important for the acquisition, but important for the positioning. For example, there are perturbations coming from the atmosphere of the Earth, and more particularly from the ionosphere and the troposphere, that affect the speed of the signal (and consequently the distances that will be measured). However, some models have been developed to correct these effects, and the use of signals at different frequencies (L1 and L2 (or L5) for example) also allows us to correct them (Hoque and Jakowski [2012]).

1.5 Basic operation of a GNSS receiver

In this section, we describe the different elements of a GNSS receiver, and we provide the signal model at each stage. Here we consider two models : the first one takes into account the impact of the Doppler effect and of the reference frequency accuracy on both carrier and code. Whereas the second model takes into account only the major impacts on the carrier. A tilde is used to denote this simplified model.

The first model is typically required when we want to detect weak signals or when the speed of the receiver is high (like on an aircraft or a spacecraft), while the simplified model is enough for strong signals when the speed of the receiver is low (like on a boat or a car).

1.5.1 Antenna

The antenna is the element that converts the received electromagnetic wave into an electrical signal. The design of an antenna for GNSS can be quite complex and involves many parameters such as the bandwidth (it is not the same if we want to receive signals at a specific frequency (only L1 for example) or to receive signals at different frequencies (L1 and L5 for example)), the gain, the polarization, the phase center, etc. The reader can refer to (Wang [2012]) for an introduction on this topic, or to (Grewal et al. [2013] Chap. 5, Rama Rao et al. [2012], Chen et al. [2012]) for detailed information.

After the antenna, the received signal is the combination of signals coming from U different satellites plus a noise term (discussed later in Section 1.5.2), it is thus defined as

$$s_r(t) = \sum_{u=1}^{U} s_r^u(t) + \eta_r(t). \tag{1.11}$$

The signal coming from a satellite u when the data and pilot channels are in quadrature is

defined as

$$\begin{aligned} s_r^u(t) &= \sqrt{2P_r^u}\, c_i^u\big((1+\alpha^u)t-\tau^u\big)\, d^u\big((1+\alpha^u)t-\tau^u\big)\, \cos\Big(2\pi f_L\big((1+\alpha^u)t-\tau^u\big)+\varphi_e^u\Big) \\ &\quad + \sqrt{2P_r^u}\, c_q^u\big((1+\alpha^u)t-\tau^u\big)\, \sin\Big(2\pi f_L\big((1+\alpha^u)t-\tau^u\big)+\varphi_e^u\Big) \\ &= \sqrt{2P_r^u}\, c_i^u\big((1+\alpha^u)t-\tau^u\big)\, d^u\big((1+\alpha^u)t-\tau^u\big)\, \cos\big(2\pi(f_L+f_d^u)t+\varphi_r^u\big) \\ &\quad + \sqrt{2P_r^u}\, c_q^u\big((1+\alpha^u)t-\tau^u\big)\, \sin\big(2\pi(f_L+f_d^u)t+\varphi_r^u\big), \end{aligned} \tag{1.12}$$

where P_r^u is the received power on each channel, $\alpha^u = \frac{v^u}{c}$ with v^u is the relative velocity between the satellite u and the receiver, $f_d^u = \alpha^u f_L$ is the carrier Doppler frequency, τ^u is the delay due to the distance traveled by the signal, and $\varphi_r^u = \varphi_e^u - 2\pi f_L \tau^u$ the phase of the received carrier.

For the simplified model, the Doppler effect on the code is neglected, and the expression becomes

$$\begin{aligned} \tilde{s}_r^u(t) &= \sqrt{2P_r^u}\, c_i^u\big(t-\tau^u\big)\, d^u\big(t-\tau^u\big)\, \cos\big(2\pi(f_L+f_d^u)t+\varphi_r^u\big) \\ &\quad + \sqrt{2P_r^u}\, c_q^u\big(t-\tau^u\big)\, \sin\big(2\pi(f_L+f_d^u)t+\varphi_r^u\big). \end{aligned} \tag{1.13}$$

1.5.2 Front-end

After the antenna, the received signal goes through the front-end, which is depicted in Fig. 1.9, where f_{REF} is the reference frequency of the receiver, f_{LO} is the local oscillator frequency, $f_{IF} = f_L - f_{LO}$ is the intermediate frequency, and f_S is the sampling frequency. The signal is first amplified by a low-noise amplifier (LNA) and filtered around the frequency of interest. Then, the signal is mixed with a local sine wave of frequency f_{LO} and filtered, which brings the signal to baseband (i.e. the signal is now centered on the intermediate frequency f_I). After, the signal is amplified by an amplifier with an automatic gain control (AGC). And finally, the signal is sampled and quantized by the analog-to-digital converter (ADC).

Of course, Fig. 1.9 depicts the principle, however the design of a real front-end depends on many parameters such as the reception of one or several signals, power consumption, size, etc. (see e.g. (Chastellain [2010], Chastellain et al. [2011], La Valle et al. [2011], Ruegamer et al. [2012])).

Downconversion

The local oscillator frequency is defined as

$$f_{LO} = M_{LO}\, f_{REF}, \tag{1.14}$$

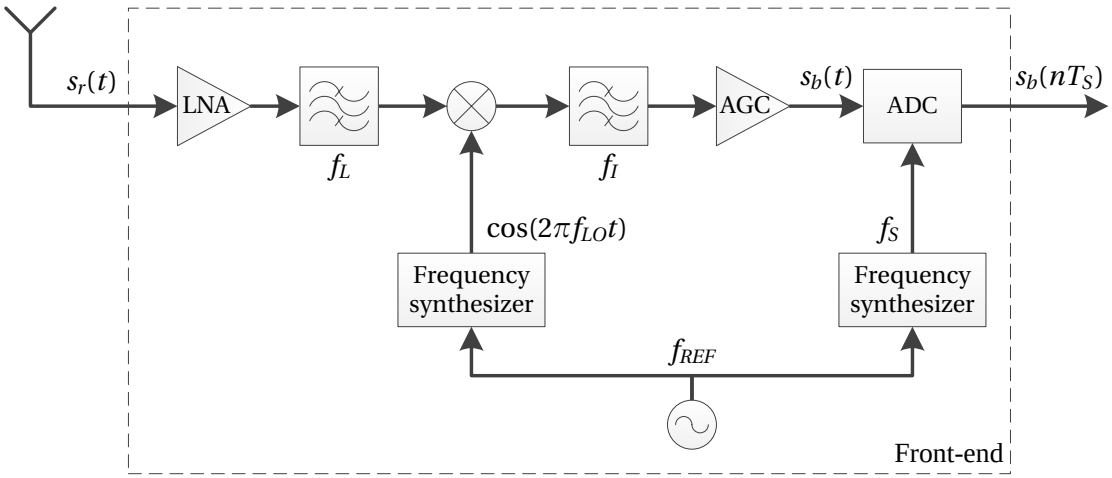


Figure 1.9: Illustration of a front-end.

where M_{LO} is a rational number. The intermediate frequency is defined as the difference between the frequency in the L band and the local frequency, we have thus

$$\begin{aligned} f_I &= f_L - f_{LO} \\ &= f_L - M_{LO}\, f_{REF}. \end{aligned} \tag{1.15}$$

However, the reference oscillator is not perfect, it has a certain accuracy (van Diggelen [2009] Chap. 3) and phase noise (Thombre et al. [2011]), and consequently its actual frequency is not exactly f_{REF}. To take into account the accuracy of the reference oscillator, we will denote f_{ref} the actual reference frequency, which is defined as

$$f_{ref} = f_{REF}(1+\beta), \tag{1.16}$$

where β represents the oscillator accuracy. The accuracy of an oscillator depends mainly on the type of oscillator used, and is usually defined in ppm. For example, if the oscillator accuracy is 1 ppm, $\beta = 10^{-6}$. Consequently, this impacts the local oscillator frequency and thus the intermediate frequency. The actual intermediate frequency, denoted f_i, is thus defined as

$$\begin{aligned} f_i &= f_L - M_{LO}\, f_{ref} \\ &= f_L - M_{LO}\, f_{REF}(1+\beta) \\ &= f_I - \Delta f_i, \end{aligned} \tag{1.17}$$

where $\Delta f_i = \beta\, M_{LO}\, f_{REF} = \beta\, f_{LO}$, and represents the difference between the theoretical and the actual intermediate frequency. Let's make an example to give an insight into the impact of the oscillator accuracy. Let's consider that $f_{REF} = 16.384$ MHz with an accuracy of ± 1 ppm, and $M_{LO} = 96$. We have then $f_I = 2.556$ MHz and $\Delta f_i = 1572.864$ Hz. So, the offset implied by the local oscillator may represent a significant amount, and this shows the importance to have a good oscillator. It can be noticed that Δf_i is proportional to the local oscillator frequency but

not to the reference frequency (because if we change the reference frequency, we also need to change the factor M_{LO}). This means that Δf_i will be lower considering the L5 frequency than the L1 frequency, but using a lower reference frequency will not change anything.

The baseband signal for a satellite u is thus defined as

$$\begin{aligned} s_b^u(t) = &\sqrt{2P_b^u}\, c_i^u\big((1+\alpha^u)t-\tau^u\big)\, d^u\big((1+\alpha^u)t-\tau^u\big)\, \cos\big(2\pi f_b^u t+\varphi_b^u\big) \\ &+\sqrt{2P_b^u}\, c_q^u\big((1+\alpha^u)t-\tau^u\big)\, \sin\big(2\pi f_b^u t+\varphi_b^u\big), \end{aligned} \tag{1.18}$$

where P_b^u is the power on each channel, $f_b^u = f_I - \Delta f_i + f_d^u$ is the baseband frequency, which contains two unknowns, one common to all the satellite signals (Δf_i), and one specific to each satellite signal (f_d^u), and φ_b^u is the phase of the carrier.

For the simplified model, the expression of the baseband signal for a satellite u is

$$\begin{aligned} \tilde{s}_b^u(t) = &\sqrt{2P_b^u}\, c_i^u\big(t-\tau^u\big)\, d^u\big(t-\tau^u\big)\, \cos\big(2\pi f_b^u t+\varphi_b^u\big) \\ &+\sqrt{2P_b^u}\, c_q^u\big(t-\tau^u\big)\, \sin\big(2\pi f_b^u t+\varphi_b^u\big). \end{aligned} \tag{1.19}$$

Sampling

The sampling frequency is defined as

$$f_S = M_S\, f_{REF}, \tag{1.20}$$

where M_S is a rational number. In the same way as for the intermediate frequency, the reference oscillator accuracy impacts the actual sampling frequency, which is defined as

$$\begin{aligned} f_s &= M_S\, f_{ref} \\ &= M_S\, f_{REF}(1+\beta) \\ &= f_S(1+\beta). \end{aligned} \tag{1.21}$$

Consequently, the actual sampling period is $T_s = \frac{T_S}{1+\beta}$, where $T_s = \frac{1}{f_s}$ and $T_S = \frac{1}{f_S}$. Therefore, after the ADC, the discrete baseband signal for a satellite u is

$$\begin{aligned} s_b^u(nT_s) = &\sqrt{2P_b^u}\, c_i^u\big((1+\alpha^u)nT_s-\tau^u\big)\, d^u\big((1+\alpha^u)nT_s-\tau^u\big)\, \cos\big(2\pi f_b^u nT_s+\varphi_b^u\big) \\ &+\sqrt{2P_b^u}\, c_q^u\big((1+\alpha^u)nT_s-\tau^u\big)\, \sin\big(2\pi f_b^u nT_s+\varphi_b^u\big), \end{aligned} \tag{1.22}$$

where n is the index of the samples. This signal can also be expressed using the theoretical sampling period,

$$\begin{aligned} s_b^u(nT_s) = &\sqrt{2P_b^u}\; c_i^u\left(\tfrac{1+\alpha^u}{1+\beta}\, nT_S - \tau^u\right)\, d^u\left(\tfrac{1+\alpha^u}{1+\beta}\, nT_S - \tau^u\right)\, \cos\left(2\pi \tfrac{f_b^u}{1+\beta}\, nT_S + \varphi_b^u\right) \\ &+ \sqrt{2P_b^u}\; c_q^u\left(\tfrac{1+\alpha^u}{1+\beta}\, nT_S - \tau^u\right)\, \sin\left(2\pi \tfrac{f_b^u}{1+\beta}\, nT_S + \varphi_b^u\right). \end{aligned} \tag{1.23}$$

Eqs. (1.22) and (1.23) are strictly equivalent, but they provide a different interpretation. With Eq. (1.22), we consider that the input signal is not modified, and that it is sampled with the actual sampling frequency. With Eq. (1.23), we consider that the input signal is modified, and that it is sampled with the theoretical sampling frequency. In this last case, the modification acts as a Doppler effect, but identical for all the satellite signals.

The first question is when Eq. (1.22) is more appropriate, and when Eq. (1.23) is more appropriate ? In short, the answer is with hardware receivers for Eq. (1.22), and with software receivers for Eq. (1.23). Indeed, for a hardware receiver, the clock signal used for the sampling is provided to the next stages of the receiver. So all the signals that will be generated by the receiver will be based on the actual sampling frequency, this is why it is better to consider the actual sampling frequency. However, for a software receiver, the sampling frequency is written in hard in a program. And this value corresponds to the theoretical sampling frequency of course, this is why in this case it is better to consider the theoretical sampling frequency.

The second question is what is the impact of this Doppler like effect ? In a perfect world, we should receive a signal sampled at the theoretical sampling frequency, and we should generate local signals sampled at the theoretical sampling frequency. In the reality, with a hardware receiver, we receive a signal sampled at the actual sampling frequency, and we generate local signals sampled at the actual sampling frequency, thus the effect is automatically compensated. But with a software receiver, we receive a signal sampled at the actual sampling frequency, and we generate local signals sampled at the theoretical sampling frequency, thus the effect is not compensated.

Let's take an example with $\beta = 10^{-6}$, where we receive a code with a chipping rate of 1.023 Mchip/s and where we want to generate a code with a chipping rate of 1.023 Mchip/s. For a hardware receiver, after the ADC, the code chipping rate is indeed 1.023 Mchip/s considering a sampling frequency f_s, when the local code chipping rate will be 1.023 Mchip/s considering a sampling frequency f_s. Whereas, for a software receiver, after the ADC, the code chipping rate is $\frac{1.023}{1+\beta} \approx 1.022\,998\,977$ Mchip/s considering a sampling frequency f_S, when the local code chipping rate will be 1.023 Mchip/s considering a sampling frequency f_S, which may cause a problem. Consequently, this effect needs to be taken into account during the design of software receivers.

Note that nevertheless, the offset due to the oscillator (Δf_i) can be estimated once the receiver has computed its position and tracks several signals, and thus the actual sampling frequency can be estimated too. However, if the receiver is switched off, and switched on later, the

oscillator offset has changed, so the uncertainty can be reduced but never cancelled when starting the receiver.

The carrier Doppler shift is also affected by the sampling, but in a lesser extent. For example, if $\beta = 10^{-6}$ and the baseband frequency f_b^u is 2.56 MHz, after the ADC, the baseband frequency will be shifted by be about 2.56 Hz, which is negligible compared to the range of the carrier Doppler shift.

For the simplified model, we can consider only the theoretical sampling frequency, thus the discrete baseband signal for a satellite u is

$$\begin{aligned}\tilde{s}_b^u(nT_s) = &\sqrt{2P_b^u}\, c_i^u\big(nT_S - \tau^u\big)\, d^u\big(nT_S - \tau^u\big)\, \cos\big(2\pi f_b^u nT_S + \varphi_b^u\big) \\ &+ \sqrt{2P_b^u}\, c_q^u\big(nT_S - \tau^u\big)\, \sin\big(2\pi f_b^u nT_S + \varphi_b^u\big).\end{aligned} \tag{1.24}$$

Complex sampling

In Fig. 1.9, the local oscillator generates one sine wave. It is also possible to generate two sine waves in quadrature, in order to obtain two sequences at the output of the front-end, as illustrated in Fig. 1.10. In this case, the discrete baseband signal $s_b(nT_s)$ can be modeled as a complex signal, where $s_i(nT_s)$ is the real part and $s_q(nT_s)$ is the imaginary part. For a satellite u, these two signals are defined as

$$\begin{aligned}s_i^u(nT_s) = &\sqrt{2P_b^u}\, c_i^u\big((1+\alpha^u)nT_s - \tau^u\big)\, d^u\big((1+\alpha^u)nT_s - \tau^u\big)\, \cos\big(2\pi f_b^u nT_s + \varphi_b^u\big) \\ &+ \sqrt{2P_b^u}\, c_q^u\big((1+\alpha^u)nT_s - \tau^u\big)\, \sin\big(2\pi f_b^u nT_s + \varphi_b^u\big)\end{aligned} \tag{1.25}$$

and

$$\begin{aligned}s_q^u(nT_s) = &\sqrt{2P_b^u}\, c_i^u\big((1+\alpha^u)nT_s - \tau^u\big)\, d^u\big((1+\alpha^u)nT_s - \tau^u\big)\, \sin\big(2\pi f_b^u nT_s + \varphi_b^u\big) \\ &- \sqrt{2P_b^u}\, c_q^u\big((1+\alpha^u)nT_s - \tau^u\big)\, \cos\big(2\pi f_b^u nT_s + \varphi_b^u\big).\end{aligned} \tag{1.26}$$

Therefore, the discrete baseband signal for a satellite u is

$$\begin{aligned}s_b^u(nT_s) &= s_i^u(nT_s) + j\, s_q^u(nT_s) \\ &= \sqrt{2P_b^u}\, \Big(c_i^u\big((1+\alpha^u)nT_s - \tau^u\big)\, d^u\big((1+\alpha^u)nT_s - \tau^u\big) \\ &\qquad - j\, c_q^u\big((1+\alpha^u)nT_s - \tau^u\big)\Big)\, e^{j\left(2\pi f_b^u nT_s + \varphi_b^u\right)},\end{aligned} \tag{1.27}$$

where $j = \sqrt{-1}$.

In the same way, for the simplified model of the discrete baseband signal for a satellite u, we

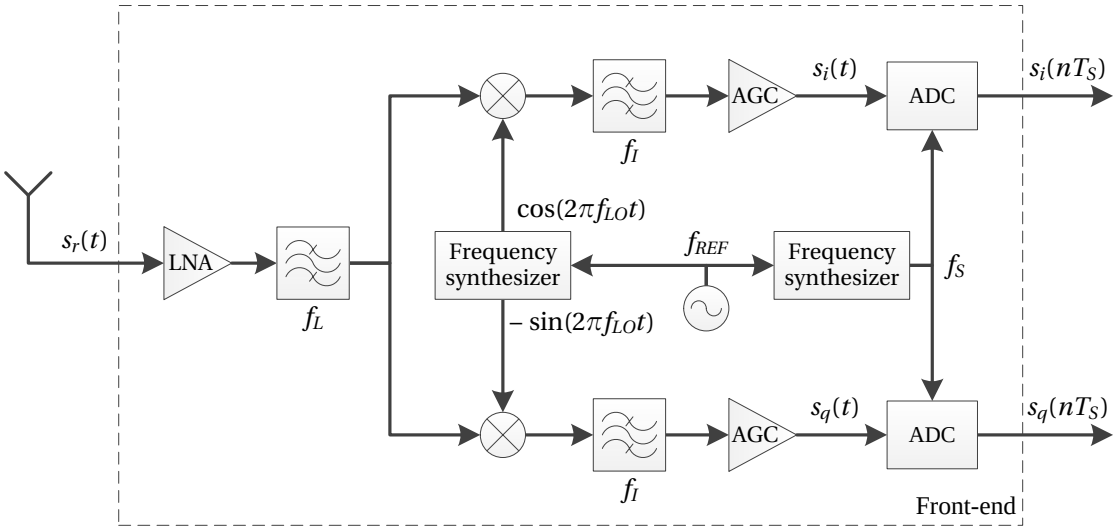


Figure 1.10: Illustration of a front-end using complex sampling.

have

$$\begin{aligned}\tilde{s}_i^u(nT_s) = &\sqrt{2P_b^u}\, c_i^u\big(nT_S-\tau^u\big)\, d^u\big(nT_S-\tau^u\big)\, \cos\big(2\pi f_b^u nT_S+\varphi_b^u\big)\\ &+\sqrt{2P_b^u}\, c_q^u\big(nT_S-\tau^u\big)\, \sin\big(2\pi f_b^u nT_S+\varphi_b^u\big)\end{aligned} \tag{1.28}$$

and

$$\begin{aligned}\tilde{s}_q^u(nT_s) = &\sqrt{2P_b^u}\, c_i^u\big(nT_S-\tau^u\big)\, d^u\big(nT_S-\tau^u\big)\, \sin\big(2\pi f_b^u nT_S+\varphi_b^u\big)\\ &-\sqrt{2P_b^u}\, c_q^u\big(nT_S-\tau^u\big)\, \cos\big(2\pi f_b^u nT_S+\varphi_b^u\big),\end{aligned} \tag{1.29}$$

and thus

$$\begin{aligned}\tilde{s}_b^u(nT_s) &= \tilde{s}_i^u(nT_s) + j\,\tilde{s}_q^u(nT_s)\\ &= \sqrt{2P_b^u}\,\Big(c_i^u\big(nT_S-\tau^u\big)\, d^u\big(nT_S-\tau^u\big) - j\, c_q^u\big(nT_S-\tau^u\big)\Big)\, e^{j\left(2\pi f_b^u nT_S+\varphi_b^u\right)}.\end{aligned} \tag{1.30}$$

The use of complex sampling presents some advantages and drawbacks for the front-end design (e.g. it resolves the image frequency problem, but it requires two paths instead of one. See e.g. (Chastellain et al. [2011]) for more details). It also allows us to have a lower intermediate frequency (it can be even 0 Hz) and a lower sampling frequency than for a real sampling (the minimum sampling frequency corresponds to the signal bandwidth, whereas it is twice the signal bandwidth for a real sampling).

Note that in Fig. 1.10, the local sine wave of the bottom branch is ahead compared to the sine wave of the top branch, i.e. there is a difference of phase of −90°. However, it is possible to have it late by removing the minus sign of the sine wave of the bottom branch, leading to a difference of phase of +90°. In this case, there are two differences with the model established by Eqs.

(1.27) and (1.30). The first is that the data and pilot channels are combined with $+j$ instead of $-j$, and the second is the insertion of a minus sign in the complex exponential, which implies a reversal of the spectrum. For example, if the carrier Doppler shift for a satellite is 1000 Hz, the baseband signal will be shifted by −1000 Hz, which implies additional corrections in the later processing.

Noise

As indicated previously, the received signal is the combination of several signals coming from different satellites, plus a noise term. The discrete baseband signal is thus

$$s_b(nT_s) = \sum_{u=1}^{U} s_b^u(nT_s) + \eta_b(nT_s), \tag{1.31}$$

where $\eta_b(nT_s)$ is a noise. This noise comes from the thermal noise induced by the antenna and the front-end themselves. The thermal noise is assumed to be an additive white Gaussian noise (AWGN) (Ziemer and Tranter [2008] pp. 341–342). The two-sided power spectral density value of an AWGN is constant and equal to $\frac{N_0}{2}$ (Ziemer and Tranter [2008] pp. 313–314), where N_0 is defined as

$$N_0 = k_B\, T_{EFF}, \tag{1.32}$$

with k_B the Boltzmann constant and T_{EFF} the effective temperature of the entire front-end, therefore N_0 is expressed in W/Hz (equivalent to joule). The effective temperature of the front-end depends on the noise figure of the front-end, on the ambient temperature and on the effective temperature of the antenna (van Diggelen [2009] pp. 133–137). This means that the front-end itself plays an important role in the noise level present at the output.

Before the ADC, the signal is filtered, and therefore the noise too. If we assume an ideal filter of two-sided bandwidth $2B$, the noise before the ADC is a bandlimited white noise (Ziemer and Tranter [2008] pp. 342–343). In this case, the noise power is

$$\sigma_b^2 = \frac{N_0}{2} 2B = N_0\, B. \tag{1.33}$$

Therefore, the noise power is proportional to the bandwidth of the filter front-end. This means that the noise power depends on the signal we want to receive (e.g. the bandwidth of the GPS L5 signal is ten times the bandwidth of the a GPS L1 C/A signal).

After the ADC, the spectrum is replicated at each multiple of the sampling frequency, as illustrated in Fig. 1.11a, which shows the power spectral density (PSD) of a bandlimited white noise when $B < f_s/2$. It can be seen that after the sampling the noise is still a bandlimited white noise. However, if we consider the special case $B = f_s/2$, illustrated Fig. 1.11b, after the sampling the noise is a white noise. This difference is important because in the later processing (acquisition and tracking), the samples will be accumulated. With a white noise,

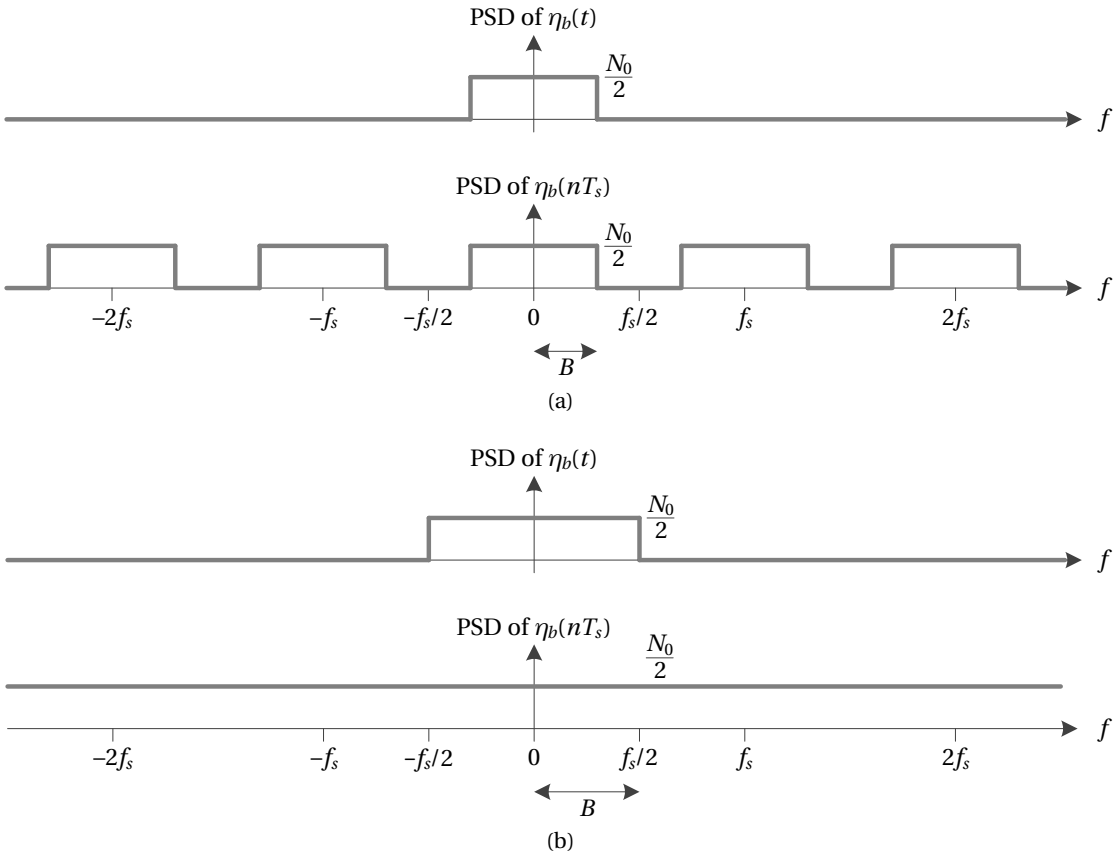


Figure 1.11: Illustration of the noise PSD before and after the sampling, (a) when $B < f_s/2$, (b) when $B = f_s/2$.

the samples are uncorrelated and thus the variance of the noise increases linearly with the number of samples accumulated, while with a bandlimited noise the samples are correlated and the variance increases faster. If we do not take this into account, we may think that keeping the same front-end bandwidth and increasing the sampling frequency will provide a better signal-to-noise ratio (SNR), which is not the case, as demonstrated in (van Diggelen [2009] pp. 146–154). Therefore, to consider $\eta_b(nT_s)$ as an AWGN, we should consider $B = \frac{f_s}{2}$ for a real sampling, or $B = f_s$ for a complex sampling. Of course, in the real world, it is not possible to build an ideal filter, consequently, a precise analysis can be performed only when the transfer function of the front-end filter is known.

Since the noise power depends on the front-end bandwidth (or on the sampling frequency), the SNR at the output of the front-end may be different for signals of same power but different bandwidth. However, in the later processing, for the same accumulation time, the SNR will be the same for signals of same power but different bandwidth (because whatever is the bandwidth of the input signal, the bandwidth after the accumulation is reduced to the same value). Therefore, the SNR at the output of the front-end is not providing a very meaningful information. In order to get rid of the front-end bandwidth, we usually use the carrier power-

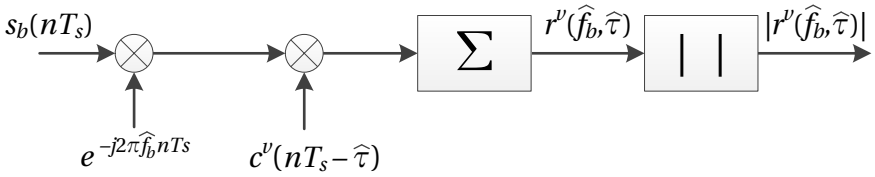


Figure 1.12: Acquisition principle.

to-noise density ratio (Joseph [2010], van Diggelen [2009] pp. 137–140), defined as

$$C/N_0 = \frac{P_r}{N_0}. \tag{1.34}$$

The C/N_0 is usually expressed in log scale, thus we have

$$\begin{aligned} C/N_0 &= 10\log_{10}\left(\frac{P_r}{N_0}\right) \\ &= 10\log_{10}(P_r) - 10\log_{10}(N_0). \end{aligned} \tag{1.35}$$

For example, considering a signal power of -160 dBW, an effective temperature of 300 K for the front-end, which means $N_0 = -203.8$ dBW/Hz, we have $C/N_0 = 43.8$ dBHz, which is a typical value for open sky view. Note that the C/N_0 does not depend on the front-end bandwidth, but it still depends on the noise figure of the front-end.

1.5.3 Acquisition

After the front-end, the first stage of a GNSS receiver is the acquisition. Its purpose is threefold : 1) Detect the satellites in view; 2) Obtain a rough estimation of the baseband frequency f_b^u; and 3) Obtain a rough estimation of the delay of the spreading code transmitted τ^u (in fact not τ^u itself, but τ^u modulo one code period).

The processing consists of four steps as shown in Fig. 1.12 : 1) Multiplication with a complex exponential of frequency $-\widehat{f}_b$; 2) Multiplication with the spreading code of the satellite searched with a delay $\widehat{\tau}$; 3) Accumulation of the signal samples in order to increase the SNR; and 4) Computation of the magnitude (or the power) of the signal.

When we are looking for a satellite v, if $\widehat{f}_b$ is close to f_b^v and if $\widehat{\tau}$ is close to τ^v, the value $\left|r^v(\widehat{f}_b,\widehat{\tau})\right|$ will be high. Else if $\widehat{f}_b$ is far from f_b^v or if $\widehat{\tau}$ is far from τ^v, the value $\left|r^v(\widehat{f}_b,\widehat{\tau})\right|$ is low. The notion of close and far will be formally defined in Chapter 2.

Consequently, this process is repeated for different values of $\widehat{f}_b$ and $\widehat{\tau}$ until a peak exceeds a predefined threshold for example, or until all the possibilities have been tested without success (which means that the satellite is not in view or that we have missed it). So, the acquisition is a two-dimensional search for each satellite. It is possible to compute $r^v(\widehat{f}_b,\widehat{\tau})$ for one or several couples $(\widehat{f}_b,\widehat{\tau})$ at a time, as described more in details in Chapter 2.

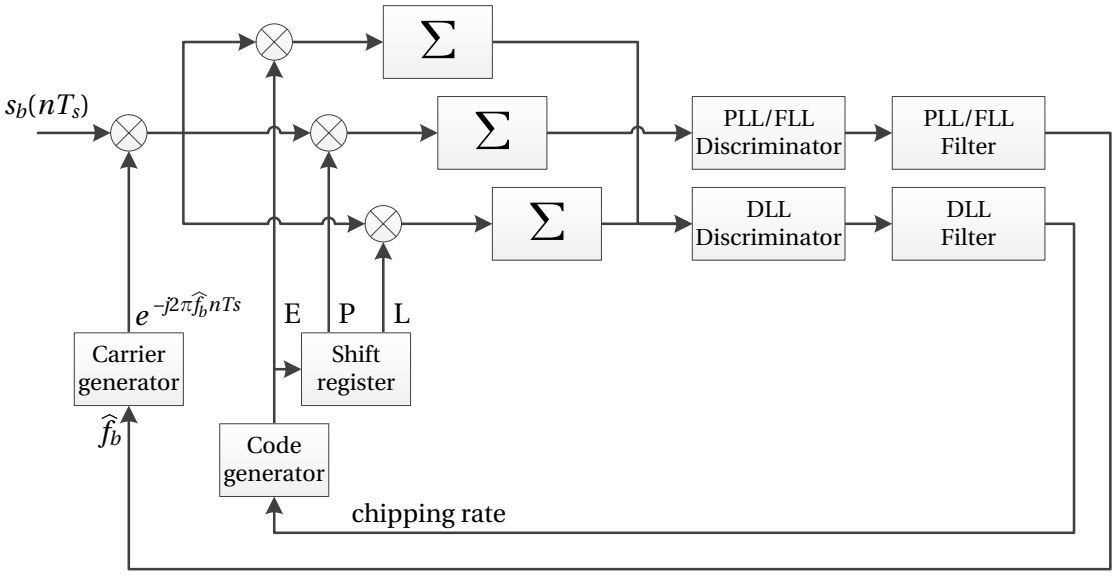


Figure 1.13: Tracking principle.

1.5.4 Tracking

Once a satellite has been detected during the acquisition, the estimates $\widehat{f}_b$ and $\widehat{\tau}$ have to be refined and followed, because they change over time since the satellites are constantly moving and the receiver may also move.

The operation performed in tracking is similar to the one in acquisition, i.e. the incoming signal is multiplied with a local carrier, then multiplied with a local code, and the result is then accumulated. To follow the received carrier frequency, a frequency-locked loop (FLL) can be used, but usually the receivers use a phase-locked loop (PLL) for better performance. To follow the received code, a delay-locked loop (DLL) is used. The DLL generates three versions of the local code (sometimes more, as with BOC modulations), called early, prompt and late, which are slightly shifted versions of each other (less than one chip), in order to keep the synchronization. This basic scheme is illustrated in Fig. 1.13. Of course, the design of the tracking loops depends on many parameters, such as the sensitivity and the accuracy expected, the oscillator noise, if we need to deal with multipath or not, etc. For more details on tracking, the reader can refer to (Kaplan and Hegarty [2005] Chap. 5, Curran [2010]).

1.5.5 Navigation

Once a signal is tracked, the navigation data bits can be extracted. This usually requires a significant amount of time. For example, for the GPS L1 C/A, the data essential for the navigation are transmitted during 18 s, and they are repeated each 30 s. Therefore, in the best case, 18 s are needed, while in the worst case 30 s are needed (assuming all the bits are correctly decoded).

Once the data have been extracted for at least four satellites, the position of the GNSS satellites can be computed. Then, using the pseudoranges (apparent distances between the receiver and the satellites), the position of the receiver can be determined. For more details on navigation, see e.g. (Misra and Enge [2011] Chap. 6, van Diggelen [2009] Chap. 3).

1.6 Summary

In this chapter, we have presented in a general way the GNSSs and the GNSS signals, and more in details the necessary elements to understand the following chapters. We have defined two models for the discrete baseband signal. The first one takes into account the impact of the Doppler effect and of the reference frequency accuracy on both carrier and code. Whereas the second model, which is a simplified version, takes into account only the major impacts on the carrier.

The simplified model can be largely sufficient for strong signals when the speed of the receiver is low (as for a boat or a car). However, if we want to detect weak signals or if the speed of the receiver is quite high, the simplified model is usually not sufficient. It is even worst with the signals having a chipping rate of 10.23 MHz (since the code Doppler is ten times higher than with the other GNSS signals).

Note that here we did not consider all the effects. For example, we did not show the impact of the quantization during the digitization. Usually, a loss is added to take this into account (van Diggelen [2009], p. 154), but it can also be the subject of deeper studies (Curran et al. [2009] , Curran et al. [2010]). We also mentioned the Doppler rate of change, however this does not appear in the equations defining the received signal, mainly to not overload the equations. But in case of high dynamics and high sensitivity, this should be taken into account.

2 Acquisition of GNSS signals

In this chapter, we present in details the operations performed during the acquisition. We also explain the principle, the advantages and drawbacks of different acquisition methods. And finally, we introduce briefly some notions that are important but out of the scope of this thesis.

2.1 The cross ambiguity function

2.1.1 Exact derivation

As indicated in Section 1.5.3, for the search of a satellite v, there are four steps as shown in Fig. 1.12 : 1) Multiplication with a complex exponential of frequency $-\widehat{f}_b$; 2) Multiplication with the spreading code of the satellite searched with a delay $\widehat{\tau}$; 3) Accumulation of the signal samples in order to increase the SNR; and 4) Computation of the magnitude (or the power) of the signal.

In the following developments, we will consider the simplified model with a complex input signal. If we search for the satellite v, after an accumulation over N_C samples, the signal can be expressed as

$$r^v(\widehat{f}_b,\widehat{\tau}) = \sum_{n=0}^{N_C-1} \tilde{s}_b(nT_s)\ c^v(nT_S-\widehat{\tau})\ e^{-j2\pi\widehat{f}_b nT_S}. \tag{2.1}$$

Using Eq. (1.31), we can then write

$$\begin{aligned} r^v(\widehat{f}_b,\widehat{\tau}) &= \sum_{n=0}^{N_C-1}\left(\sum_{u=1}^{U}\tilde{s}_b^u(nT_s)+\eta_b(nT_s)\right) c^v(nT_S-\widehat{\tau})\ e^{-j2\pi\widehat{f}_b nT_S} \\ &= \sum_{u=1}^{U}\left(\sum_{n=0}^{N_C-1}\tilde{s}_b^u(nT_s)c^v(nT_S-\widehat{\tau})e^{-j2\pi\widehat{f}_b nT_S}\right) \\ &\quad + \sum_{n=0}^{N_C-1}\eta_b(nT_s)\ c^v(nT_S-\widehat{\tau})\ e^{-j2\pi\widehat{f}_b nT_S}. \end{aligned} \tag{2.2}$$

$r^{\nu}(\widehat{f}_b, \widehat{\tau})$ is called the cross ambiguity function (CAF). We can also express Eq. (2.2) as

$$\begin{aligned} r^{\nu}(\widehat{f}_b, \widehat{\tau}) = & \underbrace{\sum_{n=0}^{N_C-1} \tilde{s}_b^{\nu}(nT_s)\; c^{\nu}(nT_S - \widehat{\tau})\; e^{-j2\pi \widehat{f}_b nT_S}}_{r^{\nu,\nu}(\widehat{f}_b, \widehat{\tau})} \\ & + \sum_{\substack{u=1 \\ u\neq\nu}}^{U} \underbrace{\left(\sum_{n=0}^{N_C-1} \tilde{s}_b^{u}(nT_s) c^{\nu}(nT_S - \widehat{\tau}) e^{-j2\pi \widehat{f}_b nT_S} \right)}_{r^{\nu,u}(\widehat{f}_b, \widehat{\tau})} \\ & + \underbrace{\sum_{n=0}^{N_C-1} \eta_b(nT_s)\; c^{\nu}(nT_S - \widehat{\tau})\; e^{-j2\pi \widehat{f}_b nT_S}}_{r^{\nu,\eta}(\widehat{f}_b, \widehat{\tau})} \\ = & \; r^{\nu,\nu}(\widehat{f}_b, \widehat{\tau}) + \sum_{\substack{u=1 \\ u\neq\nu}}^{U} r^{\nu,u}(\widehat{f}_b, \widehat{\tau}) + r^{\nu,\eta}(\widehat{f}_b, \widehat{\tau}), \end{aligned} \tag{2.3}$$

where $r^{\nu,\nu}$ corresponds to the CAF considering only the signal of interest, $r^{\nu,u}$ corresponds to the CAF considering only a signal coming from another satellite, thus this can be considered as an interference from our point of view, and $r^{\nu,\eta}$ corresponds to the CAF considering only the noise.

The number of samples used for the accumulation defines the coherent integration time T_C, which is equal to $T_C = N_C\, T_S$. The coherent integration time impacts the sensitivity (i.e. the minimum power of the input signal for which the signal can be detected) and some acquisition parameters, as shown later.

For the acquisition, we can either use the data channel or the pilot channel, or both channels. For the moment, we will consider the simplest case where we use only the pilot channel. In this case, the local code corresponds to the code of the pilot channel, and $r^{\nu,\nu}$ becomes

$$r^{\nu,\nu}(\widehat{f}_b, \widehat{\tau}) = \sum_{n=0}^{N_C-1} \tilde{s}_b^{\nu}(nT_s)\; c_q^{\nu}(nT_S - \widehat{\tau})\; e^{-j2\pi \widehat{f}_b nT_S}. \tag{2.4}$$

Using Eq. (1.30), we can then write

$$\begin{aligned} r^{\nu,\nu}(\widehat{f}_b, \widehat{\tau}) = & \sum_{n=0}^{N_C-1} \sqrt{2P_b^{\nu}} \left(c_i^{\nu}\big(nT_S - \tau^{\nu}\big)\, d^{\nu}\big(nT_S - \tau^{\nu}\big) - j\; c_q^{\nu}\big(nT_S - \tau^{\nu}\big) \right) e^{j\left(2\pi f_b^{\nu} nT_S + \varphi_b^{\nu}\right)} \\ & \quad c_q^{\nu}(nT_S - \widehat{\tau})\; e^{-j2\pi \widehat{f}_b nT_S} \\ = & \sqrt{2P_b^{\nu}} \sum_{n=0}^{N_C-1} c_i^{\nu}\big(nT_S - \tau^{\nu}\big)\; c_q^{\nu}(nT_S - \widehat{\tau})\; d^{\nu}\big(nT_S - \tau^{\nu}\big)\; e^{j\left(2\pi (f_b^{\nu} - \widehat{f}_b) nT_S + \varphi_b^{\nu}\right)} \\ & - j\; \sqrt{2P_b^{\nu}} \sum_{n=0}^{N_C-1} c_q^{\nu}\big(nT_S - \tau^{\nu}\big)\; c_q^{\nu}(nT_S - \widehat{\tau})\; e^{j\left(2\pi (f_b^{\nu} - \widehat{f}_b) nT_S + \varphi_b^{\nu}\right)} \\ = & \; r_{iq}^{\nu,\nu}(\widehat{f}_b, \widehat{\tau}) - j\; r_q^{\nu,\nu}(\widehat{f}_b, \widehat{\tau}), \end{aligned} \tag{2.5}$$

where $r_{iq}^{v,v}$ corresponds to an interference caused by the quadrature component of the same satellite, and $r_q^{v,v}$ corresponds to the signal of interest. Now, we will focus on the CAF of the signal of interest, and consider different cases to clearly show how looks $r_q^{v,v}$.

Correct estimation of code delay and carrier frequency

In the first case, we consider that the delay of the local code is equal to the delay of the received code, i.e. $\widehat{\tau} = \tau^v$, and that the frequency of the local carrier is equal to the frequency of the received carrier, i.e. $\widehat{f}_b = f_b^v$. In this case, we have $c_q^v\left(nT_S - \tau^v\right)\, c_q^v(nT_S - \widehat{\tau}) = 1$, and $r_q^{v,v}$ becomes

$$\begin{aligned} r_q^{v,v}(f_b^v, \tau^v) &= \sqrt{2P_b^v} \sum_{n=0}^{N_C-1} e^{j\varphi_b^v} \\ &= N_C \sqrt{2P_b^v}\, e^{j\varphi_b^v}. \end{aligned} \tag{2.6}$$

The magnitude of $r^{v,v}$ is then

$$\left| r_q^{v,v}(f_b^v, \tau^v) \right| = N_C \sqrt{2P_b^v}. \tag{2.7}$$

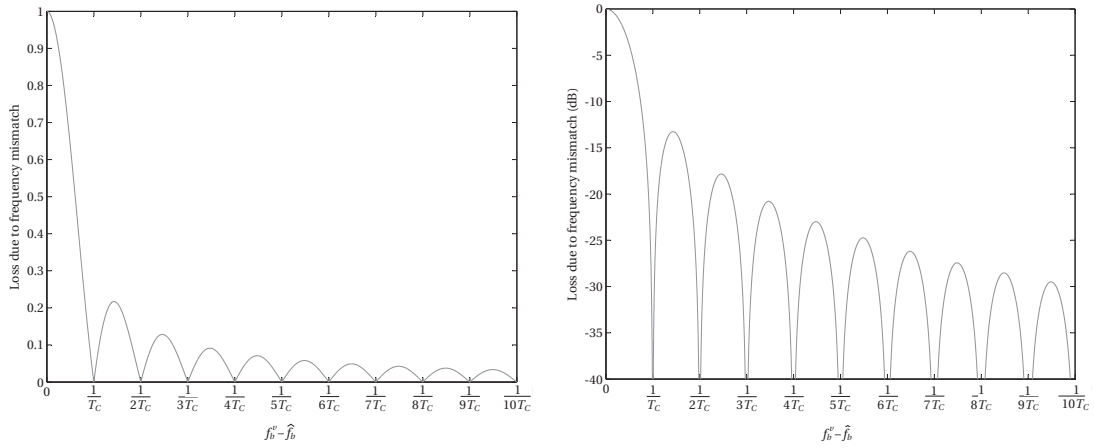
Correct estimation of code delay

In the second case, we consider that only the delay of the local code is equal to the delay of the received code, i.e. $\widehat{\tau} = \tau^v$. In this case, $r_q^{v,v}$ becomes

$$\begin{aligned} r_q^{v,v}(\widehat{f}_b, \tau^v) &= \sqrt{2P_b^v} \sum_{n=0}^{N_C-1} e^{j\left(2\pi(f_b^v - \widehat{f}_b)nT_S + \varphi_b^v\right)} \\ &= \sqrt{2P_b^v}\, e^{j\varphi_b^v} \frac{1 - e^{j2\pi(f_b^v - \widehat{f}_b)N_C T_S}}{1 - e^{j2\pi(f_b^v - \widehat{f}_b)T_S}} \\ &= \sqrt{2P_b^v}\, e^{j\varphi_b^v} \frac{e^{j\pi(f_b^v - \widehat{f}_b)N_C T_S}}{e^{j\pi(f_b^v - \widehat{f}_b)T_S}} \frac{e^{-j\pi(f_b^v - \widehat{f}_b)N_C T_S} - e^{j\pi(f_b^v - \widehat{f}_b)N_C T_S}}{e^{-j\pi(f_b^v - \widehat{f}_b)T_S} - e^{j\pi(f_b^v - \widehat{f}_b)T_S}} \\ &= \sqrt{2P_b^v}\, e^{j\varphi_b^v}\, e^{j\pi(f_b^v - \widehat{f}_b)(N_C-1)T_S} \frac{\sin\left(\pi(f_b^v - \widehat{f}_b)N_C T_S\right)}{\sin\left(\pi(f_b^v - \widehat{f}_b)T_S\right)}. \end{aligned} \tag{2.8}$$

The magnitude of $r^{v,v}$ is then

$$\begin{aligned} \left| r_q^{v,v}(\widehat{f}_b, \tau^v) \right| &= \sqrt{2P_b^v} \left| \frac{\sin\left(\pi(f_b^v - \widehat{f}_b)N_C T_S\right)}{\sin\left(\pi(f_b^v - \widehat{f}_b)T_S\right)} \right| \\ &\approx \sqrt{2P_b^v} \left| \frac{\sin\left(\pi(f_b^v - \widehat{f}_b)N_C T_S\right)}{\pi(f_b^v - \widehat{f}_b)T_S} \right| \\ &\approx N_C \sqrt{2P_b^v} \left| \operatorname{sinc}\left(\pi(f_b^v - \widehat{f}_b)T_C\right) \right|. \end{aligned} \tag{2.9}$$



(a) (b)

Figure 2.1: Illustration of the loss due to the frequency mismatch, (a) in linear scale, (b) in log scale.

On the second line, we can make this approximation because $f_b^v - \widehat{f}_b$ is several kHz at most (or dozens of kHz for high speed context), while T_S is less than a microsecond, the product is thus usually less than 10^{-2}, which allows the use of the small-angle approximation.

Eq. (2.9) means that if the estimation of the code delay is correct but the estimation of the carrier frequency is not correct, there is a loss proportional to the frequency mismatch and to the coherent integration time, as illustrated in Fig. 2.1.

Correct estimation of carrier frequency

In the third case, we consider that only the frequency of the local carrier is equal to the frequency of the received carrier, i.e. $\widehat{f}_b = f_b^v$. In this case, $r_q^{v,v}$ becomes

$$r_q^{v,v}(f_b^v, \widehat{\tau}) = \sqrt{2P_b^v}\, e^{j\varphi_b^v} \sum_{n=0}^{N_C-1} c_q^v\left(nT_s - \tau^v\right) c_q^v(nT_s - \widehat{\tau}). \tag{2.10}$$

The magnitude of $r^{v,v}$ is then

$$\left| r_q^{v,v}(f_b^v, \widehat{\tau}) \right| = \sqrt{2P_b^v} \left| \sum_{n=0}^{N_C-1} c_q^v\left(nT_s - \tau^v\right) c_q^v(nT_s - \widehat{\tau}) \right|. \tag{2.11}$$

This means that if the estimation of the carrier frequency is correct, the result is the correlation of the two codes. The correlation for integer delays is defined by the codes themselves. However, the shape of the correlation between integer delays is defined by the modulation used. This is illustrated in Fig. 2.2 for the BPSK and BOC(1,1) modulations when the codes are aligned within ±1 chip.

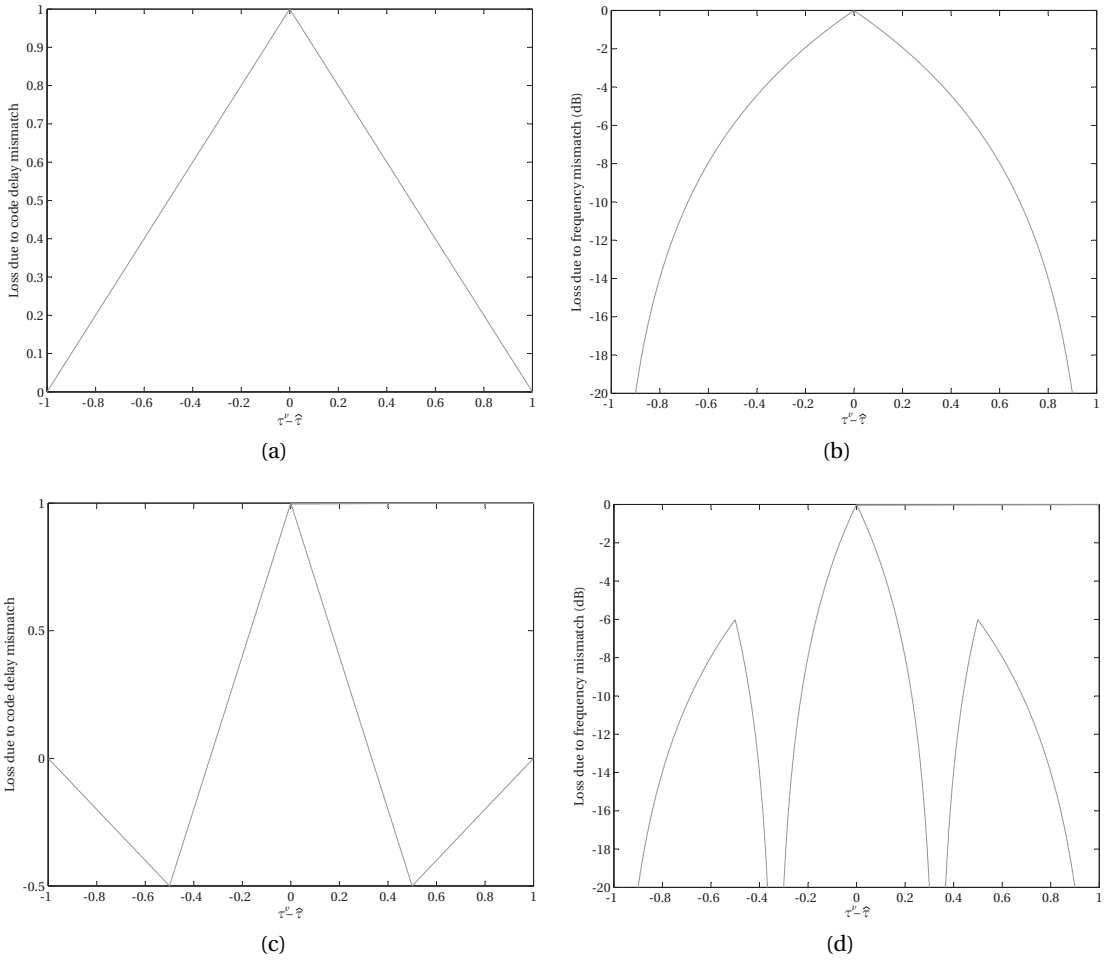


Figure 2.2: Illustration of the loss due to the code delay mismatch, (a) in linear scale for a BPSK modulation, (b) in log scale for a BPSK modulation, (c) in linear scale for a BOC(1,1) modulation, (d) in log scale for a BOC(1,1) modulation.

2.1.2 Approximation

In the case when neither the code delay nor the carrier frequency are correct, there is no better closed formed expression for $r^{\nu,\nu}$ than Eq. (2.4). Since the two effects are interacting, the total loss is not simply the product of the frequency loss and of the code loss. However, this is an often used approximation, and in this case we have

$$\left| r_q^{\nu,\nu}(\widehat{f}_b,\widehat{\tau}) \right| \approx \sqrt{2P_b^\nu} \left| \mathrm{sinc}\left(\pi(f_b^\nu - \widehat{f}_b)T_C\right) \right| \left| \sum_{n=0}^{N_C-1} c_q^\nu\left(nT_s - \tau^\nu\right) c_q^\nu(nT_s - \widehat{\tau}) \right|. \tag{2.12}$$

The question of the validity region of this approximation has been discussed in (Motella and Lo Presti [2010]), but not in a fully satisfactory way[1]. It can be checked that if we replace $\widehat{f}_b$

[1] In this paper, different components are mixed : the errors due to the double frequency term (which is not present anymore if we consider a complex sampling), the errors due to the cross-correlation with other codes, and

by f_b^ν in Eq. (2.12) we find Eq. (2.11), and if we replace $\widehat{\tau}$ by τ^ν in Eq. (2.12) we find Eq. (2.9). Therefore Eq. (2.12) is exact when at least one of the estimates is correct.

In (Motella and Lo Presti [2010], Foucras et al. [2014b]), it is shown that the approximation is very close to the exact value for any $\widehat{f_b}$ when $|\tau^\nu - \widehat{\tau}| < 1$ chip . Everywhere else, the approximation is not valid anymore, and it appears that the level is usually slightly higher than the code correlation value (Foucras et al. [2014b]). Some works on this topic include (Soualle et al. [2005], Wallner et al. [2007], Qaisar and Dempster [2007], Soualle [2009], Balaei and Akos [2010]).

2.1.3 Step of the search

As seen previously, to detect the signal, the local carrier frequency and code delay should be close to those of the received signal. Since those of the received signal are unknown, there is no other choice than computing $r^\nu(\widehat{f_b}, \widehat{\tau})$ for different couples $(\widehat{f_b}, \widehat{\tau})$ until a peak is detected.

The step between two values tested should be as small as possible to reduce the loss described previously and to have the best possible estimates. However, decreasing the step increases the number of possibilities to test, and thus increases the computational burden and the required processing time. Therefore, we must do a trade-off, which depends on the context (expected signal power, computational power available, required TTFF, etc.).

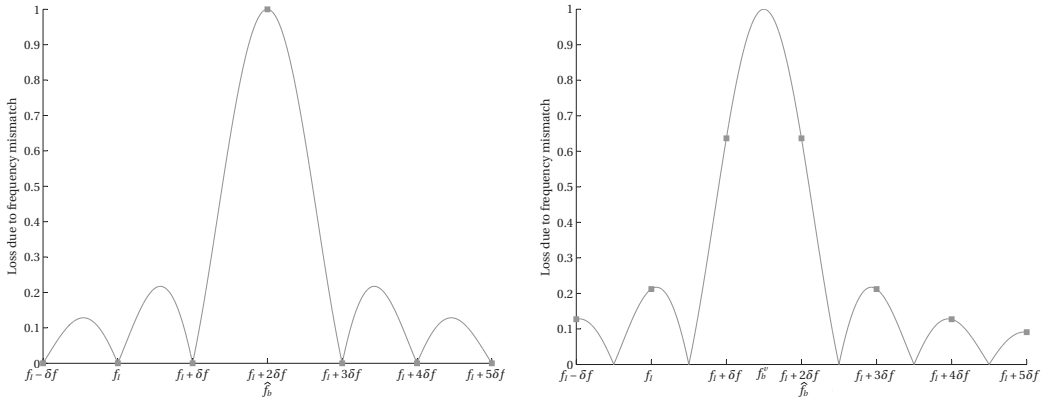
Frequency step

During the search, the parameter $\widehat{f_b}$ will take a finite number of values. This number depends on the range of the frequency search space, and on the step between two consecutive values.

Usually, the values taken by $\widehat{f_b}$ are $f_I + k\,\delta f$, where δf is the step between two frequencies tested, and k is an integer between $-\frac{N_{FB}-1}{2}$ and $\frac{N_{FB}-1}{2}$, with N_{FB} the number of frequency tested (called frequency bins). Thus, with this formula, there is a frequency bin centered on the intermediate frequency, and the same number of frequency bins above and below this frequency.

The best case that may happen during the search is when $\widehat{f_b} = f_b^\nu$, i.e. when the received frequency falls exactly on a frequency tested ($f_b^\nu = f_I + k\,\delta f$), as illustrated in Fig. 2.3a. The worst case that may happen is when the received frequency falls exactly on the middle of two frequencies tested ($f_b^\nu = f_I + \left(k + \frac{1}{2}\right)\delta f$), as illustrated in Fig. 2.3b. In this case, the frequency mismatch between f_b^ν and the closest frequency bin is $\frac{\delta f}{2}$. So, depending on the step used, the maximum loss will be different. In the literature (e.g. van Diggelen [2009], Kaplan and Hegarty [2005]), we can find mainly two rules of thumb for the choice of δf, $\frac{1}{2T_C}$ and $\frac{2}{3T_C}$, whose corresponding losses are given in Table 2.1. The second case leads to a higher loss, but

the errors due to the presence of the Doppler. This implies that at the end the real impact of the Doppler is not better known.



(a) (b)

Figure 2.3: Illustration of the loss due to the frequency step, (a) in the best case, (b) in the worst case. The frequency step δf is $\frac{1}{T_C}$ here.

Frequency step		$\frac{1}{2T_C}$	$\frac{2}{3T_C}$	$\frac{1}{T_C}$
Maximum frequency error		$\frac{1}{4T_C}$	$\frac{1}{3T_C}$	$\frac{1}{2T_C}$
Loss	Linear scale	0.9003	0.8270	0.6366
	Log scale (dB)	−0.9121	−1.6500	−3.9224

Table 2.1: Maximum loss due to the frequency mismatch for a coherent integration time T_C.

to fewer frequency bins.

Code step

Similarly as $\widehat{f}_b$, during the search, the parameter $\widehat{\tau}$ will take a finite number of values. The values taken by $\widehat{\tau}$ are $k\,\delta\tau$, where $\delta\tau$ is the step between two delays tested, and k is an integer between 0 and $N_{CB}-1$, with N_{CB} the number of code delay tested (called code bins).

The best case that may happen during the search is when $\widehat{\tau}=\tau^{\nu}$, i.e. when the code delay of the received signal falls exactly on a code delay tested ($\tau^{\nu}=k\delta\tau$), as illustrated in Figs. 2.4a and 2.4c. The worst case that may happen is when the code delay of the received signal falls exactly on the middle of two delays tested ($\tau^{\nu}=\left(k+\frac{1}{2}\right)\delta\tau$), as illustrated in Figs. 2.4b and 2.4d. In this case, the code delay mismatch between τ^{ν} and the closest code bin is $\frac{\delta\tau}{2}$. So again, depending on the step used, the maximum loss will be different. In the literature, the usual rule of thumb for the choice of $\delta\tau$ is half a chip for a BPSK modulation, and one sixth of a chip for a BOC(1,1) modulation (values chosen for Fig. 2.4 that give the same loss for both modulations), whose corresponding losses are given in Table 2.2 (van Diggelen [2009] pp. 155–158).

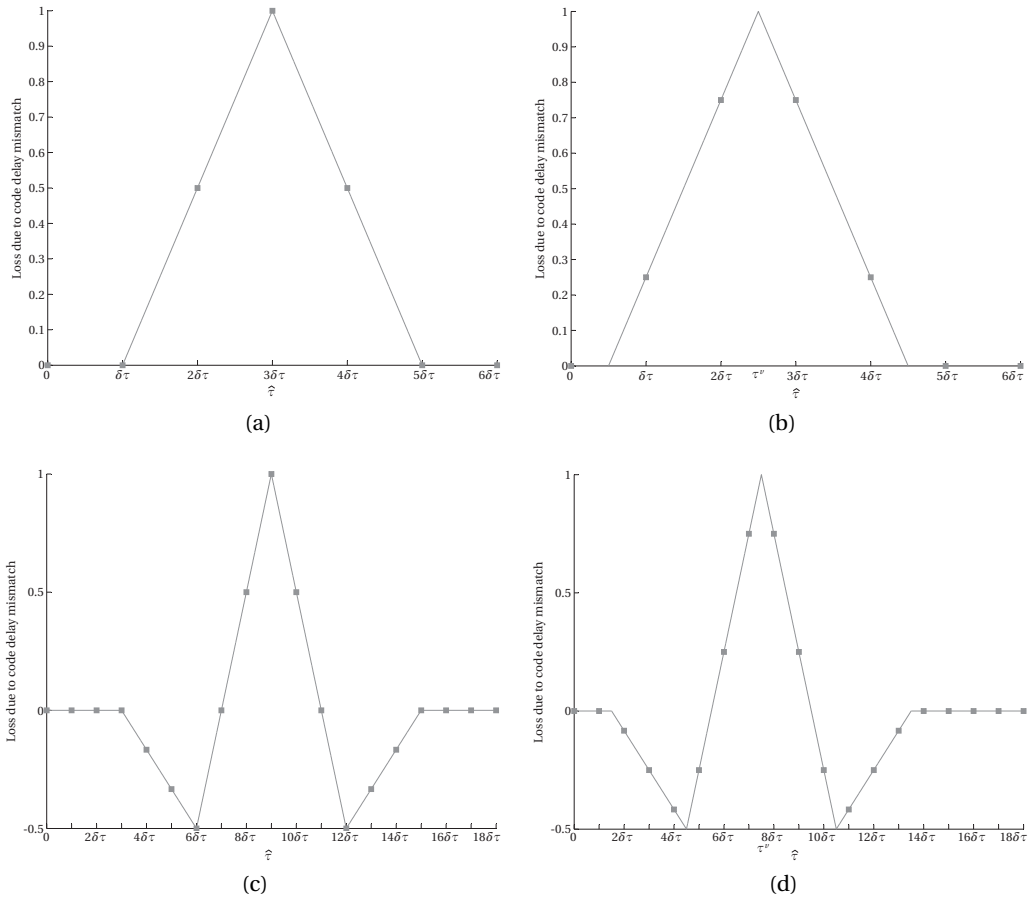


Figure 2.4: Illustration of the loss due to the code step, (a) in the best case for a BPSK modulation, (b) in the worst case for a BPSK modulation, (c) in the best case for a BOC(1,1) modulation, (d) in the worst case for a BOC(1,1) modulation. The code step $\delta\tau$ is half a chip for (a) and (b), and one sixth of a chip for (c) and (d).

2.1.4 Noise level

Regarding the noise part of the CAF, we have

$$\begin{aligned} r^{\upsilon,\eta}(\widehat{f}_b,\widehat{\tau}) &= \sum_{n=0}^{N_C-1} \eta_b(nT_s)\; c^{\upsilon}(nT_S-\widehat{\tau})\; e^{-j2\pi\widehat{f}_b nT_S} \\ &= \sum_{n=0}^{N_C-1} \left(\eta_i(nT_s)+j\;\eta_q(nT_s)\right)\; c^{\upsilon}(nT_S-\widehat{\tau})\; e^{-j2\pi\widehat{f}_b nT_S}, \end{aligned} \tag{2.13}$$

where $\eta_i(nT_s)$ and $\eta_q(nT_s)$ are AWGN of zero mean and variance σ^2.

Multiplying a noise by a spreading code does not change its mean or its variance since the code value is either $+1$ or -1.

Code step		$\frac{1}{4}$ chip	$\frac{1}{3}$ chip	$\frac{1}{2}$ chip	1 chip
Maximum code delay error		$\frac{1}{8}$ chip	$\frac{1}{6}$ chip	$\frac{1}{4}$ chip	$\frac{1}{2}$ chip
Maximum Loss	Linear scale	0.8750	0.8333	0.7500	0.5000
	Log scale (dB)	−1.1598	−1.5836	−2.4988	−6.0206
Average Loss	Linear scale	0.9375	0.9167	0.8750	0.7500
	Log scale (dB)	−0.5606	−0.7558	−1.1598	−2.4988

Table 2.2: Loss due to the code delay mismatch for a BPSK modulation.

Then, the noises are multiplied by a complex exponential, and we have

$$\begin{aligned}\eta_b(nT_s)\, e^{-j2\pi\widehat{f}_b nT_S} &= \big(\eta_i(nT_s) + j\,\eta_q(nT_s)\big)\big(\cos(2\pi\widehat{f}_b nT_S) - j\sin(2\pi\widehat{f}_b nT_S)\big)\\ &= \big(\eta_i(nT_s)\cos(2\pi\widehat{f}_b nT_S) + \eta_q(nT_s)\sin(2\pi\widehat{f}_b nT_S)\big)\\ &\quad + j\big(\eta_q(nT_s)\cos(2\pi\widehat{f}_b nT_S) - \eta_i(nT_s)\sin(2\pi\widehat{f}_b nT_S)\big)\end{aligned} \tag{2.14}$$

Multiplying a noise by a cosine wave divides its variance by 2. Since $\eta_i(nT_s)$ and $\eta_q(nT_s)$ are independent, $\eta_i(nT_s)\cos(2\pi\widehat{f}_b nT_S)$ and $\eta_q(nT_s)\sin(2\pi\widehat{f}_b nT_S)$ are also independent, and thus the variance of their sum is the sum of their variance, i.e. σ^2. Therefore, we can write

$$\begin{aligned}r^{\nu,\eta}(\widehat{f}_b,\widehat{\tau}) &= \sum_{n=0}^{N_C-1}\eta_{i,1}(nT_s) + j\,\eta_{q,1}(nT_s)\\ &= \eta_{i,2}(nT_s) + j\,\eta_{q,2}(nT_s),\end{aligned} \tag{2.15}$$

where $\eta_{i,1}(nT_s)$ and $\eta_{q,1}(nT_s)$ are noises of zero mean and variance σ^2, and $\eta_{i,2}(nT_s)$ and $\eta_{q,2}(nT_s)$ are Gaussian noises of zero mean and variance $N_C\sigma^2$.

If the result is normalized by N_C, the variance becomes $\frac{\sigma^2}{N_C}$. Considering $\sigma^2 = N_0 f_S$ (see Section 1.5.2), the variance is $\frac{N_0 f_S}{T_C f_S} = \frac{N_0}{T_C}$. Thus the longer is the coherent integration time, the lower is the noise variance.

2.2 Evaluation methods of the cross ambiguity function

In this section, we detail different methods to compute the CAF for a single or for multiple couples $(\widehat{f}_b,\widehat{\tau})$ simultaneously, and we present the advantages and drawbacks of each one.

2.2.1 Serial search

The simplest acquisition method is the serial search (or sequential search), where the CAF is computed for one couple $(\widehat{f}_b,\widehat{\tau})$ at a time (called a cell), and the computation is repeated for different $\widehat{f}_b$ and different $\widehat{\tau}$. This method is depicted in Fig. 2.5, which shows the two possibilities for the order of the operations.

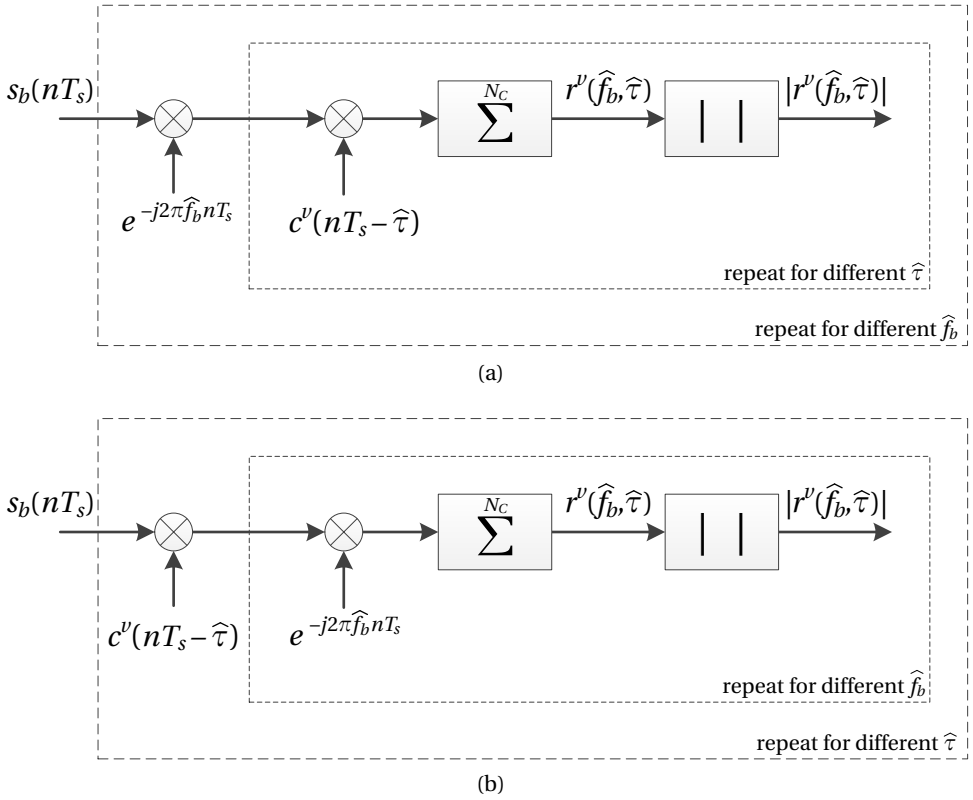


Figure 2.5: Principle of the serial search acquisition, (a) when we start by removing the carrier, (b) when we start by removing the code.

The advantage of this architecture is its simplicity, because it requires very few elements for a hardware implementation. However, this leads to an inefficient computation, and to a long acquisition time because the operations are repeated $N_{FB}\,N_{CB}$ times. This is especially true when the frequency search space is large, as for fast moving receivers, or with modern signals having long codes or modulations requiring a small code step.

2.2.2 Parallel code search

Direct approach

A first solution to reduce the acquisition time is to parallelize the search in the code search space. The second part of the processing in Fig. 2.5a (multiplication by a known sequence and accumulation for different delays) corresponds to a correlation. It is possible to implement this correlation as a circular correlation due to the repetition of the incoming code.

The circular correlation between two sequences can be computed efficiently by means of FFT (see Appendix A.3.3). Therefore, the acquisition can be done as depicted Fig. 2.6. This means that using FFTs, it is possible to obtain the CAF for all the code delays (or code bins) simultaneously for a specific $\widehat{f}_b$; hence the name of parallel code search (PCS).

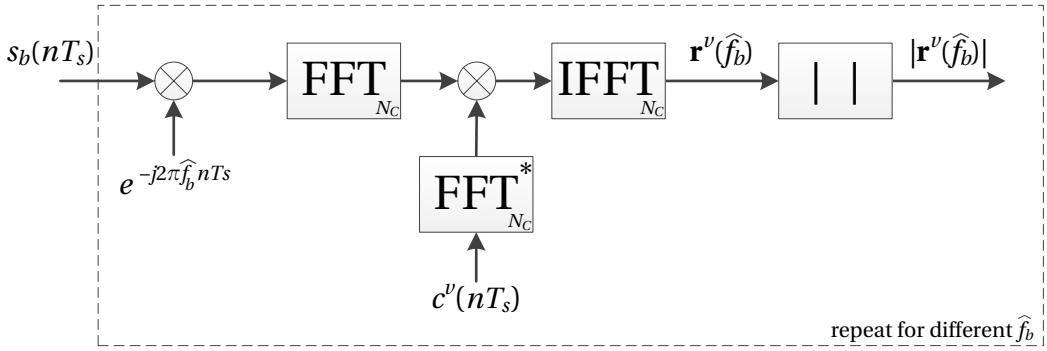


Figure 2.6: Principle of the parallel code search acquisition using a direct approach.

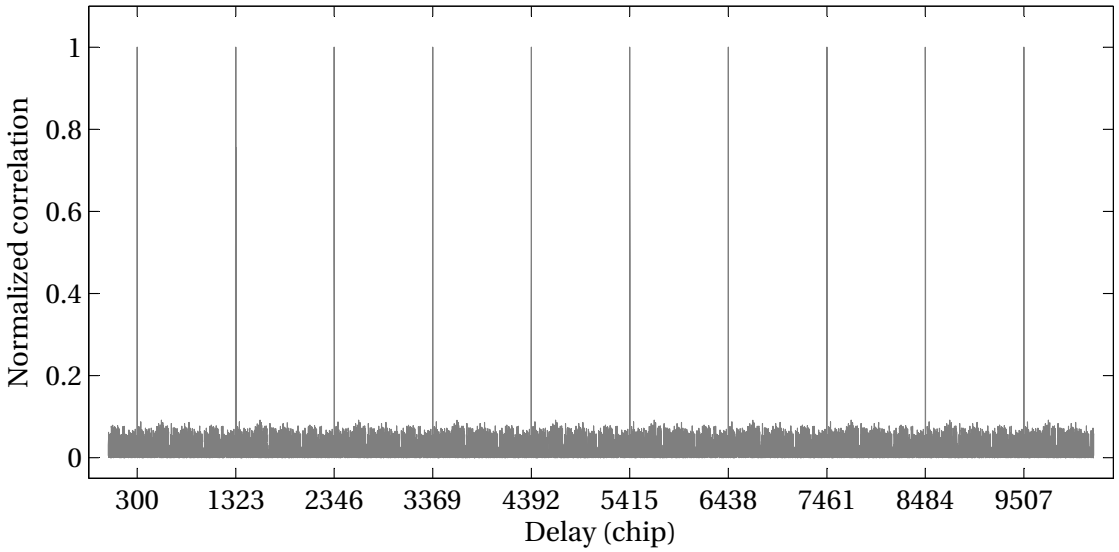


Figure 2.7: Output of the parallel code search acquisition using a direct approach, computing the circular correlation over 10 periods of the L1 C/A code.

However, if the integration is performed over several code periods, this approach is not the best one. First, the length of the FFTs is N_C and this can be quite large if the coherent integration time is long, which may lead to implementation difficulties. Second, since the received and the local codes are periodic, it is not necessary to make a circular correlation over several code periods because this will lead to several identical peaks, as shown in Fig. 2.7.

Smart approach

To circumvent the drawbacks described previously, it is possible to perform the FFTs and the IFFT over one code period, and then to add the results, as shown in Fig. 2.8, where N_{FFT}^{PCS} is the FFT length and corresponds to the number of samples in one code period, N_P is the number of code period during the coherent integration time, and we have $N_C = N_{FFT}^{PCS} N_P$.

The advantage of the parallel code search is its very high gain in processing time, since the

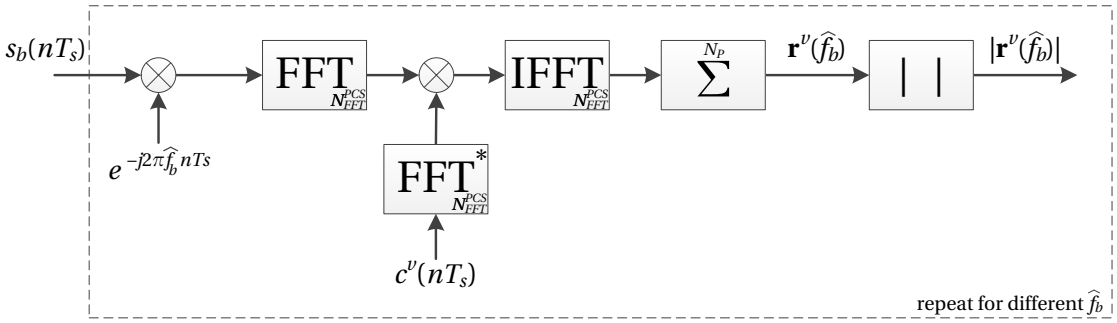


Figure 2.8: Principle of the parallel code search acquisition using a smart approach.

circular correlation computed with FFT is far more efficient than a direct implementation (Lyons [2010] Chap. 4, Smith [2002] Chap. 12). However, there are several drawbacks with this architecture.

The first drawback is its dependence on the sampling frequency. Indeed, the length of the FFT being the number of samples in one code period, it directly depends on the sampling frequency. And the choice for the length is sometimes limited.

The first FFT algorithm was proposed by (Cooley and Tukey [1965]) for sequences whose length is a composite number. The special case of sequences of length 2^n, with n a positive integer, is very popular thanks to its efficient implementation on binary computers. Still today, a lot of FFT implementations require a length that is a power of two, such as the FFTs provided by FPGA companies (Altera [2013], Xilinx [2013], Microsemi [2013]), the FFTs that we can find on www.opencores.org, or the FFTs provided by DSP companies (Texas Instruments [2013]). However, on computers, the flexibility is much higher than on embedded systems, and there are libraries that allow the computation of FFTs of sequences of any length, such as the FFTW library (Frigo and Johnson [2005]), even for sequences whose length is a prime number (Rader [1968], Bluestein [1970]). However, the performance of the FFT depends on the length, and typically it is better if the length is a composite number having small prime factors.

This is a problem because if the FFT length does not correspond to the length of a code period, it is not possible to extend the sequences by using some more samples or zero padding. Indeed, this may result in a reduction of the maximum peak, and thus in a reduction of the SNR. The number of points should be at least twice the number of samples in one code period to ensure that there will not be any loss (Leclère et al. [2010]). For example, for the GPS L1 C/A signal with a sampling frequency of 4 MHz, one code period result in 4000 samples. To avoid any losses, the sequences must be padded up to 8192 points, and not up to 4096).

The second drawback is its sensitivity to bit transition. Unlike the serial search, one period of the received code is sufficient to search all code delays (the serial search use two periods). Therefore, if there is a transition, due to data or a secondary code, it may result in losses (as shown in Chapter 5, Fig. 5.1). This problem can be resolved as the previous problem, by

doubling the FFT size and zero-padding the local code replica (case discussed in Chapter 5, see Fig. 5.2).

There exists modified versions of the parallel code search to reduce the complexity in exchange of losses in the SNR, such as those proposed in (Starzyk and Zhu [2001], Sajabi et al. [2006], Sagiraju et al. [2006], Qaisar et al. [2008]).

2.2.3 Parallel frequency search

Direct approach

A second solution to reduce the acquisition time is to parallelize the search in the frequency search space. To see this, it can be observed that the operation performed after the code removal in Fig. 2.5b is

$$r^{\nu}(\widehat{f}_b, \widehat{\tau}) = \sum_{n=0}^{N_C-1} \left(s_b^{\nu}(nT_s)\ c^{\nu}(nT_S - \widehat{\tau}) \right) e^{-j2\pi \widehat{f}_b nT_S}, \tag{2.16}$$

for different $\widehat{f}_b$. We can make a parallel with the discrete Fourier transform of a sequence x_n of N_C points, which is defined as

$$X_k = \sum_{n=0}^{N_C-1} x_n\ e^{-\frac{j2\pi kn}{N_C}}, \tag{2.17}$$

with $k = 0, 1, \ldots, N_C - 1$. So, Eq. (2.16) can be seen as a DFT of N_C points with $s_b^{\nu}(nT_s)\ c^{\nu}(nT_s - \widehat{\tau}) = x_n$ and $\widehat{f}_b nT_S = \frac{kn}{N_C}$, i.e.

$$\widehat{f}_b = \frac{k}{N_C T_s} = \frac{k}{T_C}. \tag{2.18}$$

This is interesting because a DFT can be computed efficiently with an FFT algorithm. The corresponding implementation is given in Fig. 2.9. In this case, we obtain the CAF for all the frequencies (or frequency bins) simultaneously for a specific $\widehat{\tau}$. This approach is presented for example in (Borre et al. [2007]). However, this approach has three drawbacks.

The first drawback of this method is that it can involve FFT of large size, depending on the sampling frequency and on the coherent integration time, and this may be a limitation for some implementations.

The second drawback is that the frequency search space is imposed by the FFT and is too large. Indeed, the frequencies searched by the FFT goes from 0 Hz up to $f_s \frac{N_C-1}{N_C}$. The span is thus about f_s, i.e. several MHz, while the useful frequency search space is several kHz, as illustrated in Fig. 2.10. So, most of the points computed are in fact not used, which is inefficient, and using pruning methods would not help because they are usually efficient only for small length

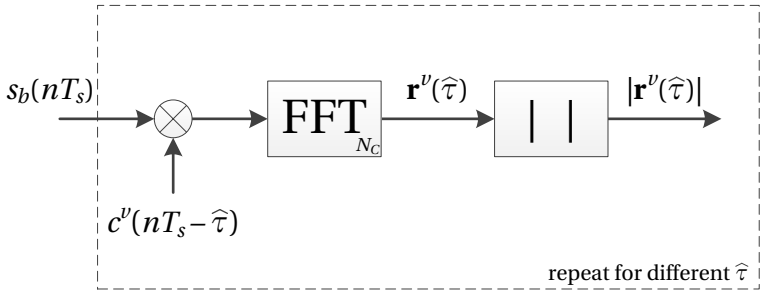


Figure 2.9: Principle of the parallel frequency search acquisition using a direct approach.

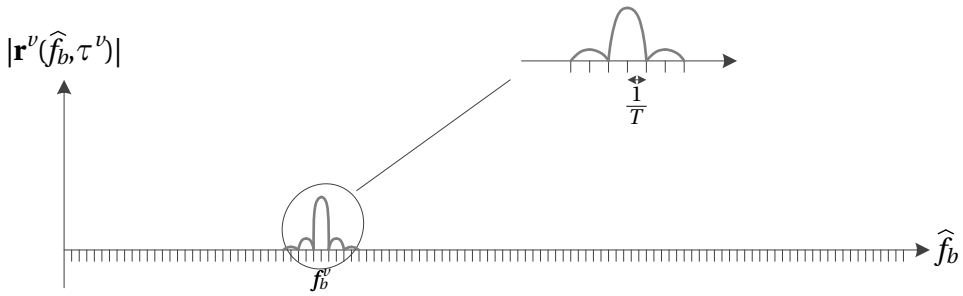


Figure 2.10: Output of the parallel frequency search acquisition using a direct approach.

(Sorensen and Burrus [1993], Huang et al. [2008], FFTW).

The third drawback is that the frequency step is $\frac{1}{T_C}$, which may imply a loss of about 3.92 dB (see Table 2.1). However, the frequency step may be reduced by zero-padding the sequence before performing the FFT. Indeed, adding $(P-1)N_C$ zeros reduces the frequency step to $\frac{1}{PT_C}$. Nevertheless, we have mentioned just before that the length of the FFT may be already large, so increasing it is not the best option.

Smart approach

To circumvent the drawbacks described previously, the idea consists of performing a short accumulation before the FFT (Mathis et al. [2003]), as illustrated in Fig. 2.11. In this way, the length of the FFT is reduced to scale to the frequency search space. In this case, the local carrier is generated only to remove the intermediate frequency. The comparison between the two approaches is illustrated in Fig. 2.12.

However, the accumulation before the FFT has an impact. Let's consider that the local code is aligned with the received code. After the code and IF removal, if we perform accumulations

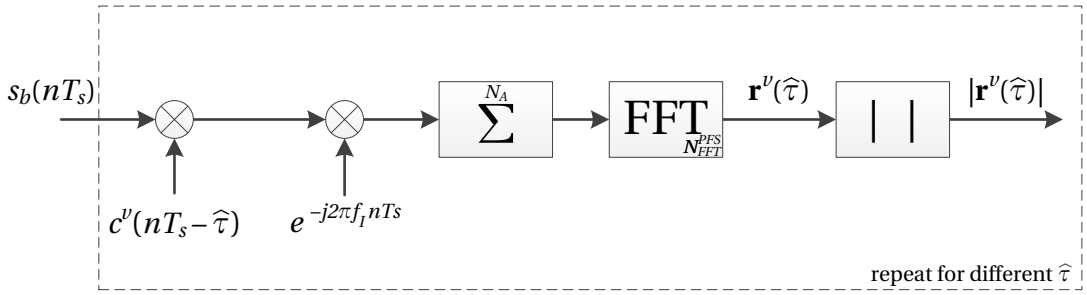


Figure 2.11: Principle of the parallel frequency search acquisition using a smart approach.

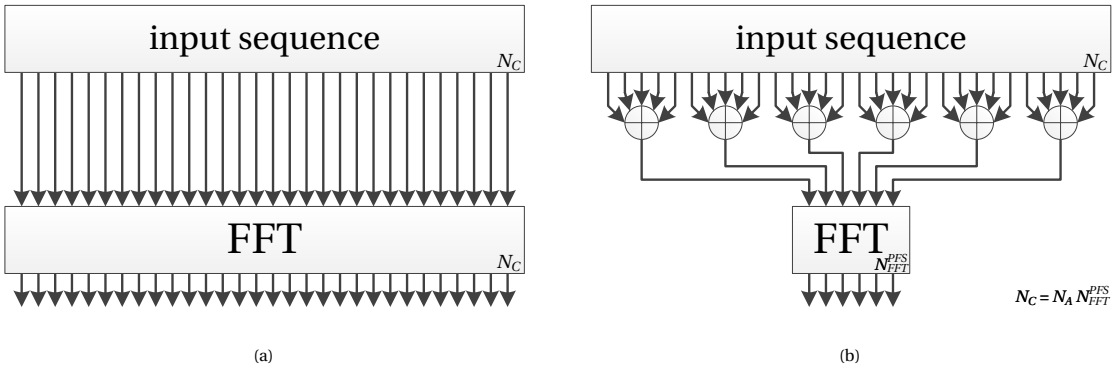


Figure 2.12: Comparison of the parallel frequency search using (a) the direct approach, (b) the smart approach, with $N_C = 30$, $N_A = 5$ and $N_{FFT}^{PFS} = 6$.

over N_A points, we obtain

$$
\begin{aligned}
r_0^v(m) &= \sqrt{2P_b^v} \sum_{n=mN_A}^{(m+1)N_A-1} e^{j(2\pi\Delta f_b^v nT_S+\varphi_b^v)} \\
&= \sqrt{2P_b^v}\, e^{j\varphi_b^v} \left(\sum_{n=0}^{(m+1)N_A-1} e^{j2\pi\Delta f_b^v nT_S} - \sum_{n=0}^{mN_A-1} e^{j2\pi\Delta f_b^v nT_S} \right) \\
&= \sqrt{2P_b^v}\, e^{j\varphi_b^v} \left(\frac{1-e^{j2\pi\Delta f_b^v (m+1)N_A T_S}}{1-e^{j2\pi\Delta f_b^v T_S}} - \frac{1-e^{j2\pi\Delta f_b^v mN_A T_S}}{1-e^{j2\pi\Delta f_b^v T_S}} \right) \\
&= \sqrt{2P_b^v}\, e^{j\varphi_b^v} \frac{1-e^{j2\pi\Delta f_b^v N_A T_S}}{1-e^{j2\pi\Delta f_b^v T_S}} e^{j2\pi\Delta f_b^v mN_A T_S} \\
&= \sqrt{2P_b^v}\, e^{j\varphi_b^v}\, e^{j\pi\Delta f_b^v (N_A-1)T_S} \frac{\sin(\pi\Delta f_b^v N_A T_S)}{\sin(\pi\Delta f_b^v T_S)} e^{j2\pi\Delta f_b^v mN_A T_S},
\end{aligned}
\tag{2.19}
$$

with $m = 0, 1, ..., N_{FFT}^{PFS}$, where $N_{FFT}^{PFS} = \frac{N_C}{N_A}$, and $\Delta f_b^v = f_b^v - f_I$. This means that the output signal still contains a complex exponential with the same frequency Δf_b^v but with a sampling period $N_A T_S$, and there is a loss proportional to N_A due to the sinc function. Now, to remove the

complex exponential, we have

$$\begin{aligned}
r^{\nu}(\widehat{\Delta f}_b) &= \sum_{m=0}^{N_{FFT}^{PFS}-1} r_0^{\nu}(m) e^{-j2\pi\widehat{\Delta f}_b m N_A T_S} \\
&= \sqrt{2P_b^{\nu}}\, e^{j\varphi_b^{\nu}}\, e^{j\pi\Delta f_b^{\nu}(N_A-1)T_S} \frac{\sin(\pi\Delta f_b^{\nu} N_A T_S)}{\sin(\pi\Delta f_b^{\nu} T_S)} \sum_{m=0}^{N_{FFT}^{PFS}-1} e^{j2\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)mN_A T_s} \\
&= \sqrt{2P_b^{\nu}}\, e^{j\varphi_b^{\nu}}\, e^{j\pi\Delta f_b^{\nu}(N_A-1)T_S} \frac{\sin(\pi\Delta f_b^{\nu} N_A T_S)}{\sin(\pi\Delta f_b^{\nu} T_S)} \frac{1-e^{j2\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)N_A N_{FFT}^{PFS} T_S}}{1-e^{j2\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)N_A T_S}} \\
&= \sqrt{2P_b^{\nu}}\, e^{j\varphi_b^{\nu}}\, e^{j\pi\Delta f_b^{\nu}(N_A-1)T_S} e^{j\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)N_A(N_{FFT}^{PFS}-1)T_S} \\
&\quad \frac{\sin(\pi\Delta f_b^{\nu} N_A T_S)}{\sin(\pi\Delta f_b^{\nu} T_S)} \frac{\sin(\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)N_A N_{FFT}^{PFS} T_S)}{\sin(\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)N_A T_S)}.
\end{aligned} \tag{2.20}$$

The magnitude of $r^{\nu}(\widehat{\Delta f}_b)$ is then

$$\begin{aligned}
\left|r^{\nu}(\widehat{\Delta f}_b)\right| &= \sqrt{2P_b^{\nu}} \left| \frac{\sin(\pi\Delta f_b^{\nu} N_A T_S)}{\sin(\pi\Delta f_b^{\nu} T_S)} \frac{\sin(\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)N_A N_{FFT}^{PFS} T_S)}{\sin(\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)N_A T_S)} \right| \\
&\approx \sqrt{2P_b^{\nu}} N_A\, N_{FFT}^{PFS} \left| \mathrm{sinc}(\pi\Delta f_b^{\nu} N_A T_S)\ \mathrm{sinc}(\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b)N_A N_{FFT}^{PFS} T_S) \right| \\
&\approx \sqrt{2P_b^{\nu}} N_C \left| \mathrm{sinc}(\pi\Delta f_b^{\nu} T_A)\ \mathrm{sinc}(\pi(\Delta f_b^{\nu}-\widehat{\Delta f}_b) T_C) \right|,
\end{aligned} \tag{2.21}$$

where $T_A = N_A T_S = \frac{T_C}{N_{FFT}^{PFS}}$ corresponds to the integration time of the accumulator before the FFT. If we compare Eq. (2.21) to Eq. (2.9), we see that there is an additional loss that depends on T_A and on the received frequency Δf_b^{ν}.

Eq. (2.20) corresponds to a DFT of N_{FFT}^{PFS} points with $\widehat{\Delta f}_b m N_A T_S = \frac{km}{N_{FFT}^{PFS}}$, i.e.

$$\widehat{\Delta f}_b = \frac{k}{N_{FFT}^{PFS} N_A T_S} = \frac{k}{N_C T_S} = \frac{k}{T_C}. \tag{2.22}$$

So the resolution is the same as with the direct approach. Therefore the loss due to the frequency step is the same as with the direct approach. To cope with this problem, as before zero-padding can be used.

With this architecture, the length of the FFT is thus divided by N_A and the frequencies tested are from 0 Hz to $\frac{f_s}{N_A}\frac{N_{FFT}^{PFS}-1}{N_{FFT}^{PFS}}$. This means that the choice of N_A depends on the frequency search space. On one hand, N_A should be as high as possible to reduce the length of the FFT. But, on the other hand, there is a loss proportional to N_A. There is thus a trade off to do. If the FFT search space is scaled exactly to the frequency search space, the loss can reach 3.92 dB for the maximum frequency. If the FFT search space is scaled to twice the frequency search space, the loss becomes 0.9 dB.

The main drawback of this architecture is the loss linked to the mismatch between the replica

code chipping rate and the received code chipping rate which is also affected by the Doppler effect (we called this code Doppler, see. Ziedan and Garrison [2004], Foucras et al. [2014a]). Indeed, several carrier frequencies are tested through the FFT whereas there is only one code chipping rate tested. For example, with the GPS L5 signal, if the carrier frequency is shifted by 2300 Hz due to the Doppler effect, the code chipping rate will be shifted by 20 chip/s. This means that the code replica and the received code will shift by 20 chips every second, or half a chip after 25 ms. Without compensation, this will imply a loss. So to reduce this effect, the frequency search space must be cut into several smaller spaces (Cheng et al. [1990]).

2.2.4 Other methods

A method that mixes the PCS and the PFS is called the double-block zero-padding (DBZP) (Ziedan and Garrison [2004], Ziedan [2005], Foucras et al. [2012]). This method search simultaneously several carrier frequencies as the PFS does, but the small accumulation performed before the FFT is now computed for several code delays using small FFTs. Since the correlation is not circular, it is needed to pad with zeros the portions of local code.

Another method that also mixes the PFS and the PCS has been presented in (Akopian [2005]), that uses FFT to search both the frequency and the code dimension at the same time, at the expense of an increase of the memory required to store intermediate results.

2.3 Sensitivity issues

The sensitivity of a receiver is a major feature. It depends on many parameters, such as the front-end noise figure, the integration time, the detection process (Paonni et al. [2009], van Diggelen [2009] Chap. 6). It has also been shown that the sensitivity depends on the acquisition search space, independently of the receiver architecture (Turunen [2010]). In this section, we discuss briefly some elements that contribute to the sensitivity, and provide interesting references for each element.

2.3.1 Integration time and integration methods

In the previous sections, we have considered a coherent integration time T_C. As shown before, in order to reduce the noise, and thus to detect signals with a power as low as possible, we need to increase the coherent integration time. However, T_C impacts the frequency mismatch loss (Eq. (2.9)), and thus the frequency step (Table 2.1). Doubling the coherent integration time means halving the frequency step. Thus the processing time is multiplied by a factor 4 (twice more data to process, and twice more frequency bins to test).

A solution to this problem is the use of non-coherent integration, which consists in accumulating the signal after the magnitude (or power) computation. This has the advantage to not impact the width of the sinc function in Eq. (2.9), thus the frequency step can stay the same.

The drawback is that the performance in terms of output SNR are lower than for the coherent integration (van Diggelen [2009] Chap. 6, Borio and Akos [2009], Jayaram and Murthy [2013]). This means that in terms of SNR, it's better to perform a coherent integration of 10 ms than 10 non-coherent accumulations of coherent integrations of 1 ms.

Another solution to this problem is the differential integration (Elders-Boll and Dettmar [2004], Ayaz [2005]), whose performances are close to the non-coherent integration (Esteves et al. [2012]).

2.3.2 Detection

Until now, we have discussed the computation of the CAF. The step after the computation is the detection of the satellite signal.

Quite often, the detection problem is addressed in a simple way using the probability of detection, the probability of false alarm and the SNR (van Diggelen [2009] Chap. 6). However, this is only an approximation, and the theory about detection is relatively complex. Indeed, a lot of parameters are involved in the detection performance, the SNR of course (Borio et al. [2008c]), but also the acquisition strategy (Borio et al. [2006], Borio et al. [2008a], Geiger et al. [2010]), the correlations between cells (Ta et al. [2012]), the detection method (threshold or ratio based, Geiger et al. [2012]), and the Doppler effect (Geiger and Vogel [2013]).

This shows the complexity of the topic, which may require a thesis for itself (O'Driscoll [2007], Borio [2008]). Therefore we do not discuss this topic here, and recommend the above references for the reader interested in this topic.

2.3.3 Time-to-first-fix

The TTFF is an important feature of GNSS receivers since it indicates the time required to obtain a position. The TTFF depends on several elements, such as the time to acquire the signals, the time to decode the data, the availability of any assistance, or the strategy used.

The acquisition time is not so easy to determine. Of course, during the design of a GNSS receiver, we know the number of code and frequency bins to test and we know the computational power, so it is easy to determine the time to explore the entire acquisition search space. The mean acquisition time depends on the time-frequency search space, but it also depends on the detection method, on the acquisition strategy (Holmes and Chen [1977], Park et al. [2002]), and on the Doppler (Lozow [1977]).

Therefore, the TTFF is also not easy to determine. However, some methodologies have been proposed (Anghileri et al. [2009], Anghileri et al. [2010], Paonni et al. [2010]).

2.4 Summary

In this chapter, we have presented the mathematical model behind the acquisition. We have also discussed the difference between the exact result and the approximation often considered.

Then, we have presented rigorously different acquisition methods, and discussed their advantages and drawbacks.

Finally, we discussed briefly some topics related to the acquisition, such as the integrations other than coherent, the detection process and the TTFF.

3 Comparison of GNSS signals acquisition architectures on FPGAs

In this chapter, the implementation on a FPGA of each acquisition method – serial search, parallel code search, and parallel frequency search – is presented and analyzed in details.

The parallel code search is much faster than the serial search, since it tests all the code delays in one time. However, for an implementation on a FPGA, the FFTs of the parallel code search will require much more resources than the simple accumulator of the serial search. Therefore, to compare the methods in a fair way, we should consider a given area (or amount of resources), and duplicate the basis structure of the methods to use all this area. Like this, the three implementations will require the same amount of resources, and then we can determine the performance by evaluating their processing time.

First, we introduce some notions about FPGAs and describe the elementary blocks that compose them. Then, we discuss briefly the implementation of GNSS receivers. After this, we describe the implementation of the different acquisition methods. Finally, we determine the parallelization and the processing time of each implementation, and we conclude by looking at the drawbacks of each implementation.

The initial work on this topic has been published in (Leclère et al. [2010]), and the final results have been published in (Leclère et al. [2013b]).

3.1 A few words on FPGAs

An FPGA is a programmable device containing mainly three types of resource (Maxfield [2008]) :

1. Logical blocks. These are small blocks containing a look-up table (to make logic function), a full adder, and one or several registers. These blocks are different for each manufacturer, and also sometimes between some FPGA families (low-cost, mid-range, high-end).

2. Memory blocks. These are memories of small size, typically between 0.5 and 128 Kibit[1].

3. DSP blocks. These blocks contain hardware multipliers, typically of 18 bits × 18 bits.

To evaluate the complexity of a system implemented on an FPGA, we must thus evaluate the resources in terms of logic, memory and DSP blocks. It is relatively easy to estimate the resource usage for the memory and DSP blocks because it is simple to determine the number of bits and the number of multiplications required in a system. However, for the logical blocks, the resource usage is more difficult to estimate for several reasons. First, these blocks contain logic and registers, and a block can use one or the other, or both, depending on the function implemented. Thus, if there are some functions (like an accumulator or a counter) whose resource usage is easy to evaluate, some other functions (like multiplexing or magnitude computation) are more difficult to evaluate and empirical formulas have to be used. Second, the compilation tools perform various optimizations that can affect the final implementation. And third, these blocks are different according to the manufacturer or even between different families, with different performances, which means that it is not possible to make a perfect and universal estimation.

Here, we consider the FPGAs from Altera, and more particularly the low-cost Cyclone III family (Altera [2012]) and the high-end Stratix III family (Altera [2011]). The logical blocks of the Cyclone III FPGAs are logical elements (LE) that contain one register, whereas the logical blocks of the Stratix III FPGAs are adaptive logic module (ALM) that contain two registers. The same estimation can be performed with FPGAs from other manufacturers, and approximate conversions can be applied between them although this is not undertaken here. The details of the resource estimation of the implementation are given in Appendix C.

3.2 A few words on GNSS receivers

The different satellite signals can be processed in parallel through several acquisition channels or sequentially using one bigger acquisition channel. Here, we consider a system with one acquisition channel, because it is more efficient in terms of resource sharing. However, the analysis proposed here can be adapted for several acquisition channels.

Also, the acquisition can be done in a streaming way by processing the signal at the sample rate, as shown in Fig. 3.1a, or the input signal can be stored in a buffer and then accessed at a higher rate, as shown in Fig. 3.1b. The second option requires of course an additional memory, but it allows a significant gain in processing time if the clock of the processing unit is much higher than the sampling frequency, as shown in Fig. 3.2. For example, if the sampling frequency is 5 MHz and the clock frequency of the processing unit is 200 MHz, once the signal is stored, the processing time will be divided by about 40. Four our comparison, we will consider the signal buffering.

[1] 1 Kibit = 1024 bits. See http://www.bipm.org/en/si/si_brochure/chapter3/prefixes.html

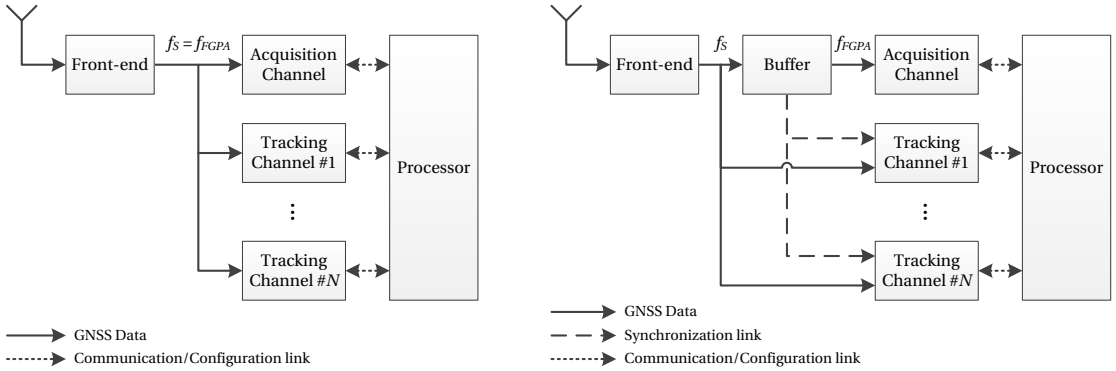


Figure 3.1: Overview of a GNSS receiver, (a) processing the samples at the sampling rate f_S, (b) using a buffer to process the samples at the FPGA rate (i.e. clock frequency of the FPGA, f_{FPGA}).

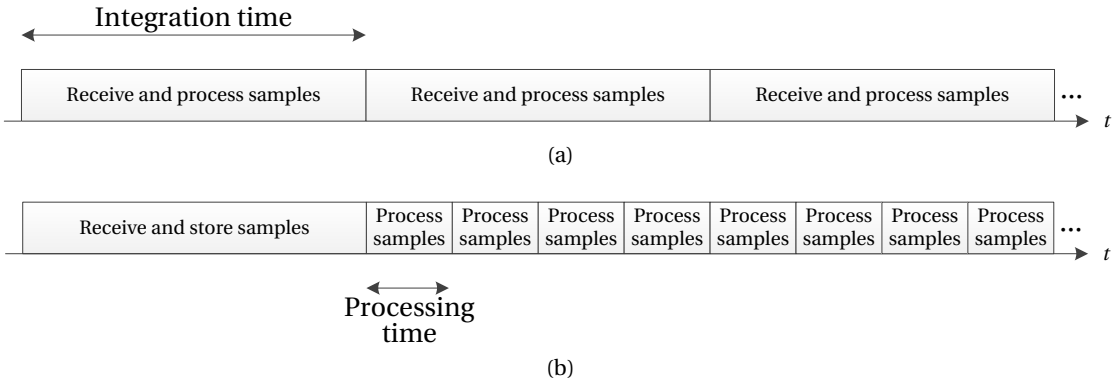


Figure 3.2: Illustration of the processing, (a) without buffer, (b) with a buffer when $f_{FPGA} = 4f_S$.

The signal buffering can also be used to share the hardware between the tracking channels and save resources, however it is more difficult to manage than for the acquisition (because in tracking, the accumulation must start at the first chip of the code, which corresponds to different instants for the different satellites).

3.3 Implementations

3.3.1 Serial search implementation

The direct implementation of the serial search method is given Fig. 3.3. It follows closely Fig. 2.5, but now we show how the local signals are usually generated, and we show the non-coherent accumulation. A numerically controlled oscillator (NCO) is a counter whose the input frequency is the sampling frequency, and whose the increment specifies the output frequency. The number of bits of the counter defines the resolution of the output frequency,

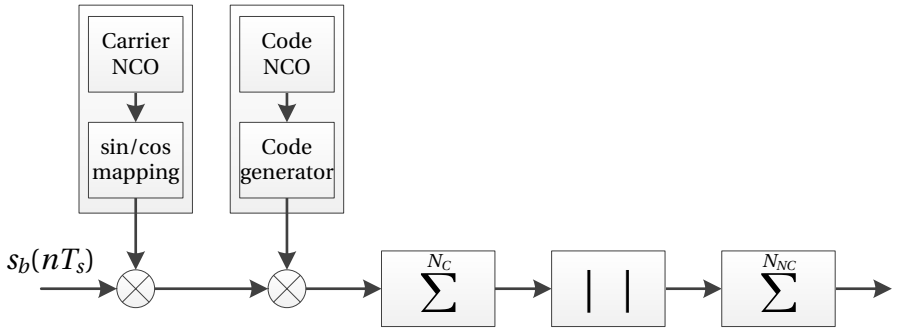


Figure 3.3: Implementation of the serial search architecture.

and is typically between 24 and 32 bits. The value of the NCO is then used to generate cosine and sine waves, or to generate the PRN code (Kaplan and Hegarty [2005] pp. 155–164).

This implementation tests the possible couples $(\widehat{f}_b, \widehat{\tau})$ one by one. In order to reduce the processing time, the blocks can be duplicated in order to have several branches (denoted as N_B in the following). There are two possibilities to do so, either testing several carrier frequencies simultaneously, or testing several code delays simultaneously. Generating different carrier frequencies requires as many NCOs as there are frequencies, while generating shifted versions of the code replica requires only one NCO and one register per delay. Since a NCO requires many registers, it is more efficient to test several code delays simultaneously.

The corresponding implementation considering N_B branches is shown in Fig. 3.4, which thus tests N_B code delays simultaneously. At the bottom of the figure, the data rate or the average data rate (when there is not a new sample at each clock cycle) is shown. The mixers and the coherent accumulators run at the frequency f_{FPGA}. The rate at the output of the coherent accumulators is then divided by N_C (the number of samples used for the coherent integration and equal to $f_S T_C$). Since the accumulation of the different accumulators starts and ends at different clock cycles (because an accumulation always starts at the first sample of the code), it is possible to use a multiplexer to share the following blocks, i.e. magnitude computation and non-coherent accumulation. To clarify this, a timing diagram is given Fig. 3.5, considering $N_C = 6$, $N_{NC} = 4$, and $N_B = 3$.

Note that there is nevertheless a limitation with the implementation of Fig. 3.4. After the accumulator, the data rate is divided by N_C, and after the multiplexer the data rate is multiplied by N_B. Since the data rate cannot be superior to the FPGA rate (f_{FPGA}), we must have $N_B \leq N_C$. Therefore, the number of branches is limited by the number of accumulations performed by the coherent accumulator. But since N_C corresponds to the number of samples in one or several code periods, its value is high enough to not have this problem. Otherwise, this limit may be easily circumvented by duplicating the blocks after the coherent accumulators, i.e. the multiplexer, the magnitude computation and the non-coherent accumulator.

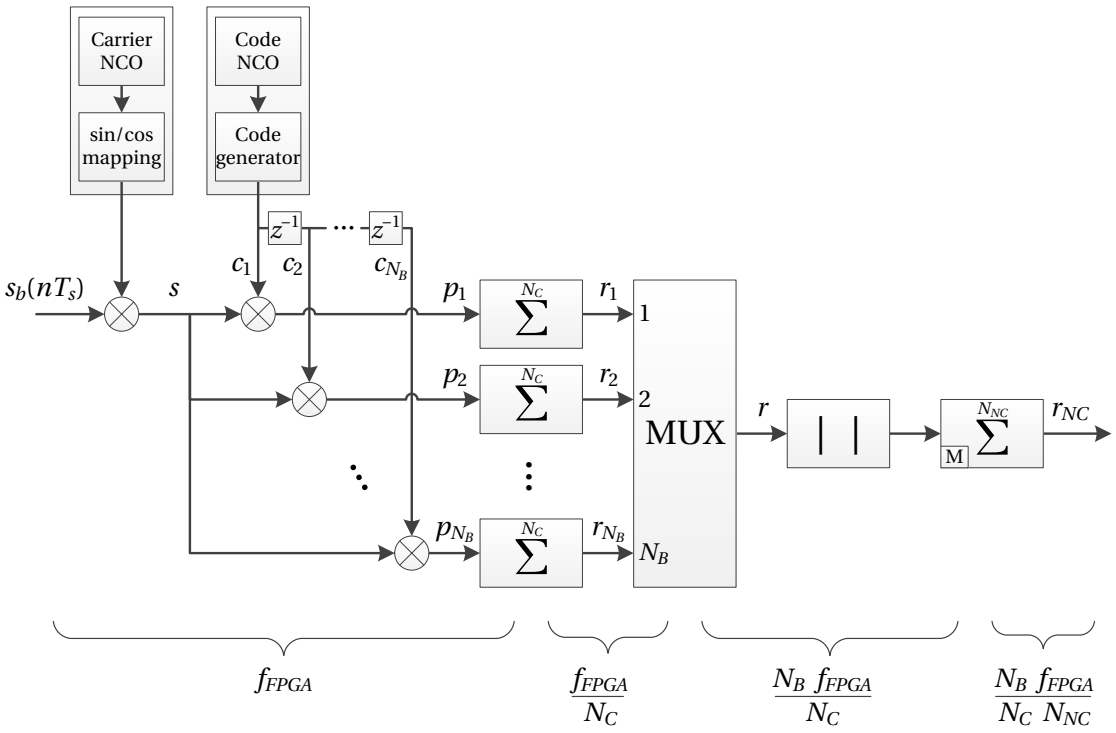


Figure 3.4: Implementation of the serial search architecture using duplication.

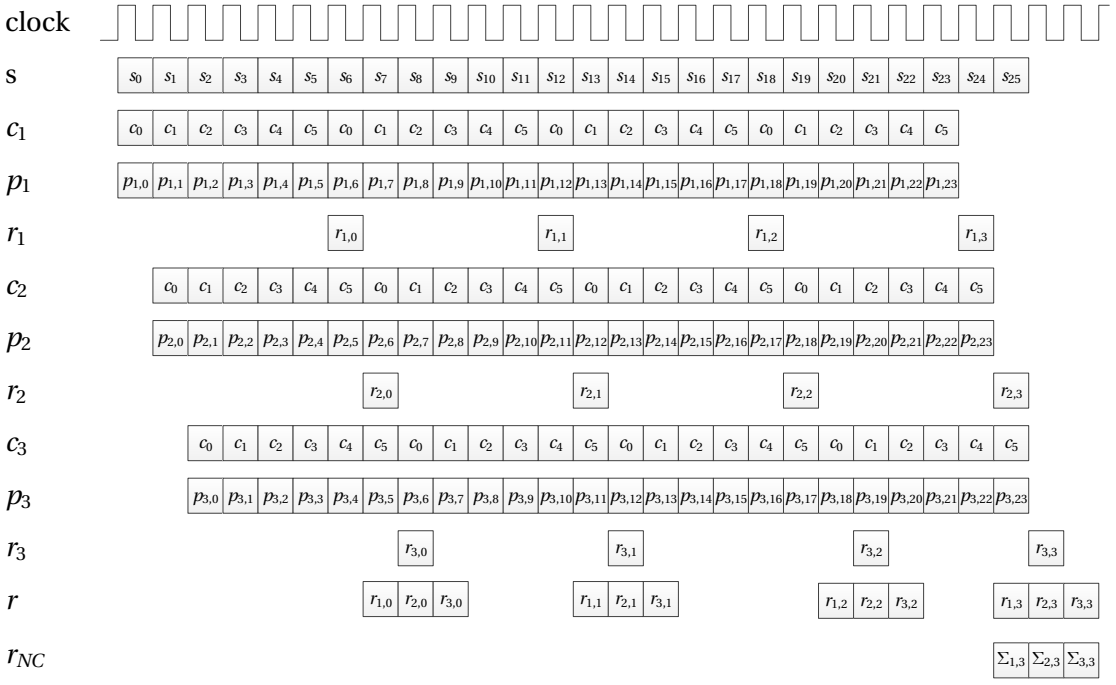


Figure 3.5: Timing diagram of Fig. 3.4 with $N_C = 6$, $N_{NC} = 4$, and $N_B = 3$. The notation $\sum_{b,p}$ corresponds to $\sum_{k=0}^{p} r_{b,k}$.

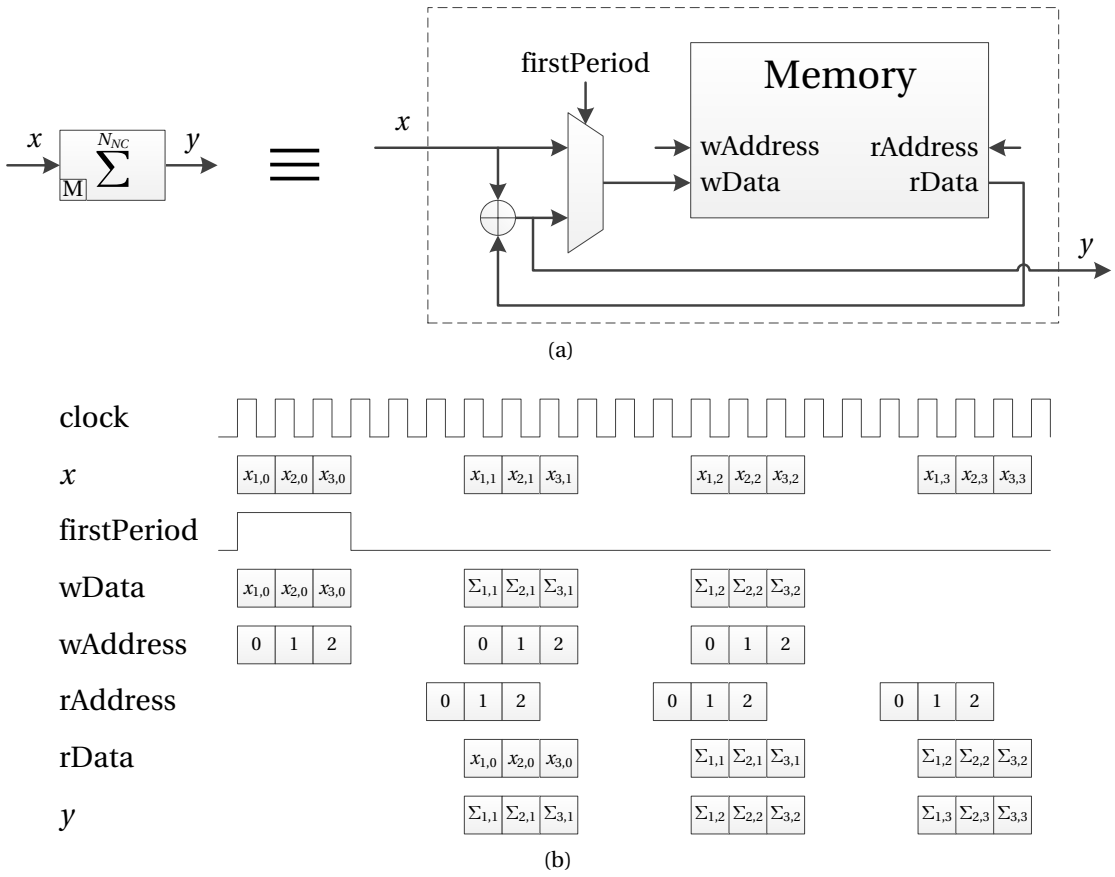


Figure 3.6: Memory-based accumulator, (a) schematic, (b) Timing diagram, with $N_{NC} = 4$. The notation $\sum_{b,p}$ corresponds to $\sum_{k=0}^{p} x_{b,k}$.

Now, let's have a closer look at the implementation of each element. Although the mixers perform a multiplication, they should be implemented with logical blocks instead of DSP blocks. Indeed, the input signal and the local carrier are quantized with few bits (between two and four usually), thus it would be a waste to use 18-bit multipliers for this. This is also true for the multiplication with the local code, since the code value is +1 or −1. The coherent accumulators are classical adders implemented with logical blocks. It is possible to optimize the implementation by fusing a code mixer and an accumulator into an accumulator that can add or subtract the input value according to the value of the code. This optimization is discussed in Section C.1.1 of Appendix C. The multiplexer is implemented with logical blocks. The magnitude computation can be performed using different approximations (Lyons [2010] pp. 679–683), the simplest being the Robertson approximation (Robertson [1971]). Finally, since the samples are in series, the non-coherent accumulator can be implemented using only one adder and a memory to save the accumulator value for each branch, in order to save logical blocks. The schematic of such accumulator is given Fig. 3.6a, and a timing diagram is given Fig. 3.6b for illustration. Each address is associated to a branch, and is accessed $N_{NC} - 1$ times to perform the accumulation. The memory has thus N_B addresses. To differentiate the

memory-based accumulator from the logic-based accumulator, the letter M is added in the bottom left corner of the block in figures.

In the serial search implementation, the duplicated elements are thus the code mixer and the coherent accumulator, and the multiplexer is proportional to the number of branches. The most-used resources are clearly the logical blocks, as the memory is used only with the non-coherent accumulator and to store the PRN code, and the DSP blocks are not used at all.

3.3.2 Parallel code search implementation

The direct implementation of the parallel code search method is given Fig. 3.7. It follows closely Fig. 2.8, but as previously, we show the generation of the local signals and the non-coherent accumulation. Since the samples at the output of the IFFT are in series, the accumulators for the coherent and non-coherent accumulations can both use a memory. The memories have thus N_{FFT}^{PCS} addresses.

Following the same idea used in the previous section, we can duplicate the elements to have several branches in order to test several carrier frequencies simultaneously. The corresponding implementation considering N_B branches is shown in Fig. 3.8.

Regarding the implementation of the elements, the carrier and code generators are identical to those seen previously, as well as the carrier mixers. The FFTs use logical, memory and DSP resources. The complex multipliers in the frequency domain use DSP blocks. And as said before, both the coherent and non-coherent accumulators are implemented with memory blocks. In this implementation, no multiplexing can be performed because the accumulation on the different branches starts at the same time. However, the addressing of the accumulators can be share, as well as the control signals of the FFTs.

This implementation has a better balance between the resources than the implementation of

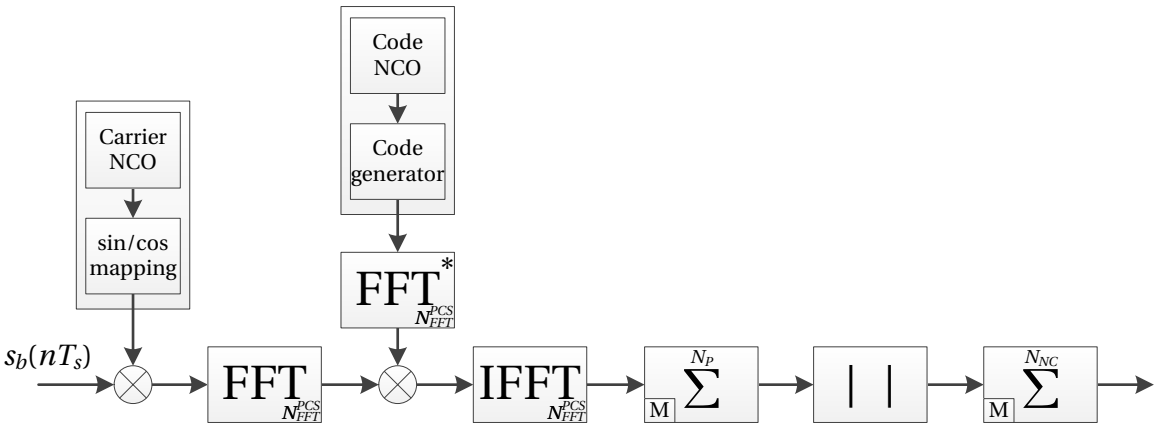


Figure 3.7: Implementation of the parallel code search architecture.

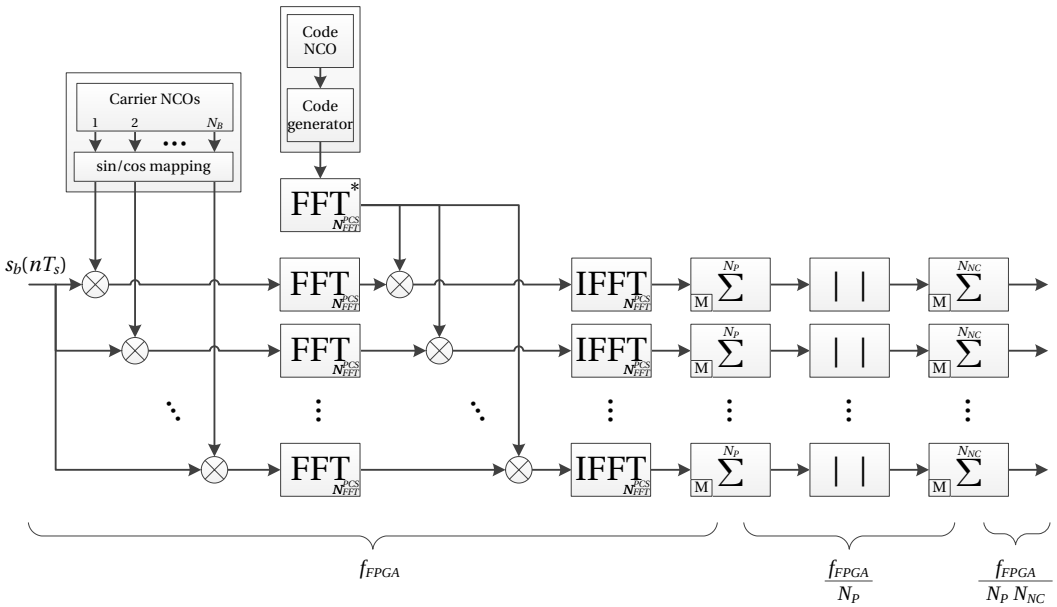


Figure 3.8: Implementation of the parallel code search architecture using duplication.

the serial search, because it uses logical (local signals generation, carrier mixers, magnitude, FFTs), memory (FFTs, accumulators), and DSP blocks (FFTs and FFT mixers).

With this implementation, there are several carriers (and thus carrier frequencies) generated while there is only one code (and thus one code frequency) generated. Therefore, there may be a code mismatch due to the Doppler effect. However, since the FFT uses a lot of resources, the number of branches is rather small, and thus the carrier frequencies may be close. For example, considering the GPS L1 C/A signal, if the carrier frequencies cover a Doppler range of ± 150 Hz, the maximum difference between the chipping rate that we should use and the one used is $\frac{150}{1540} \approx 0.097$ chip/s, which means a shift of one quarter of a chip after about 2.6 seconds, which gives a comfortable margin. Otherwise, the effect can be compensated by generating a code for each carrier frequency tested, at the expense of additional FFTs, or smarter by applying a correction during the coherent accumulation stage (shift of the IFFT outputs or multiplication by a carrier in the frequency domain, see (O'Driscoll [2007]) for more details).

An optimization of this architecture is possible if the frequency search space is wide enough. Instead of multiplying the input signal by different local carriers and performing several FFTs, only one local carrier and two FFTs can be used (one for the input signal and one for the code replica), and the multiplication by the different carriers is replaced by shifts of the FFT output (thanks to DFT shift theorem (Oppenheim and Schafer [2009] pp. 564–567)), as shown in Fig. 3.9. A shift of one sample is equivalent to a multiplication by a 1 kHz carrier if the FFT length is 1 ms. For example, considering five branches, it means that it would be possible to test simultaneously the following carrier frequencies, −2000, −1000, 0, 1000, 2000 Hz; and if no signal is found, to continue with the next frequencies, e.g. −1950, −950, 50, 1050, 2050 Hz,

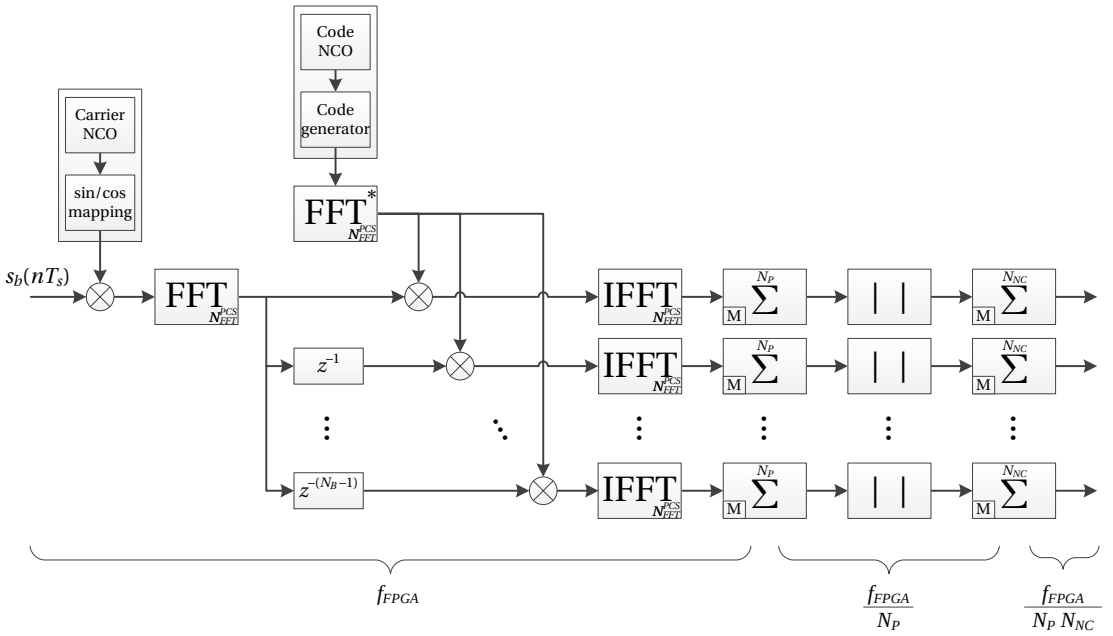


Figure 3.9: Implementation of the parallel code search architecture using shift in the frequency domain.

and so on and so forth. But in this case, the Doppler effect is more important, since with a Doppler range of ± 2 kHz, a shift of one quarter of a chip may happen after 192.5 ms only. This implementation is denoted PCS* in the following.

3.3.3 Parallel frequency search implementation

The direct implementation of the parallel frequency search method is given Fig. 3.10a. It follows closely Fig. 2.11, but as previously, we show the generation of the local signals and the non-coherent accumulation.

Following the same idea used in Section 3.3.1, we can duplicate the elements to have several branches in order to test several code delays simultaneously. The corresponding implementation considering N_B branches is shown in Fig. 3.10b.

Regarding the implementation of the elements, there is nothing new except the ping-pong buffer. This is a buffer, which has a writing order different from the reading order. This is because after the multiplexer there are first the first points of each branch, then the second points of each branch, etc., whereas the FFT should be first fed with all the points of the first branch, then with all the points of the second branch, etc. Also, since data can be written at addresses not yet read, it is necessary to use two buffers, one being read while the other is written to, which alternate their roles (which is often called a ping-pong buffer).

As mentioned in Chapter 2, according to the values selected for N_A and N_{FFT}^{PFS} , only a portion

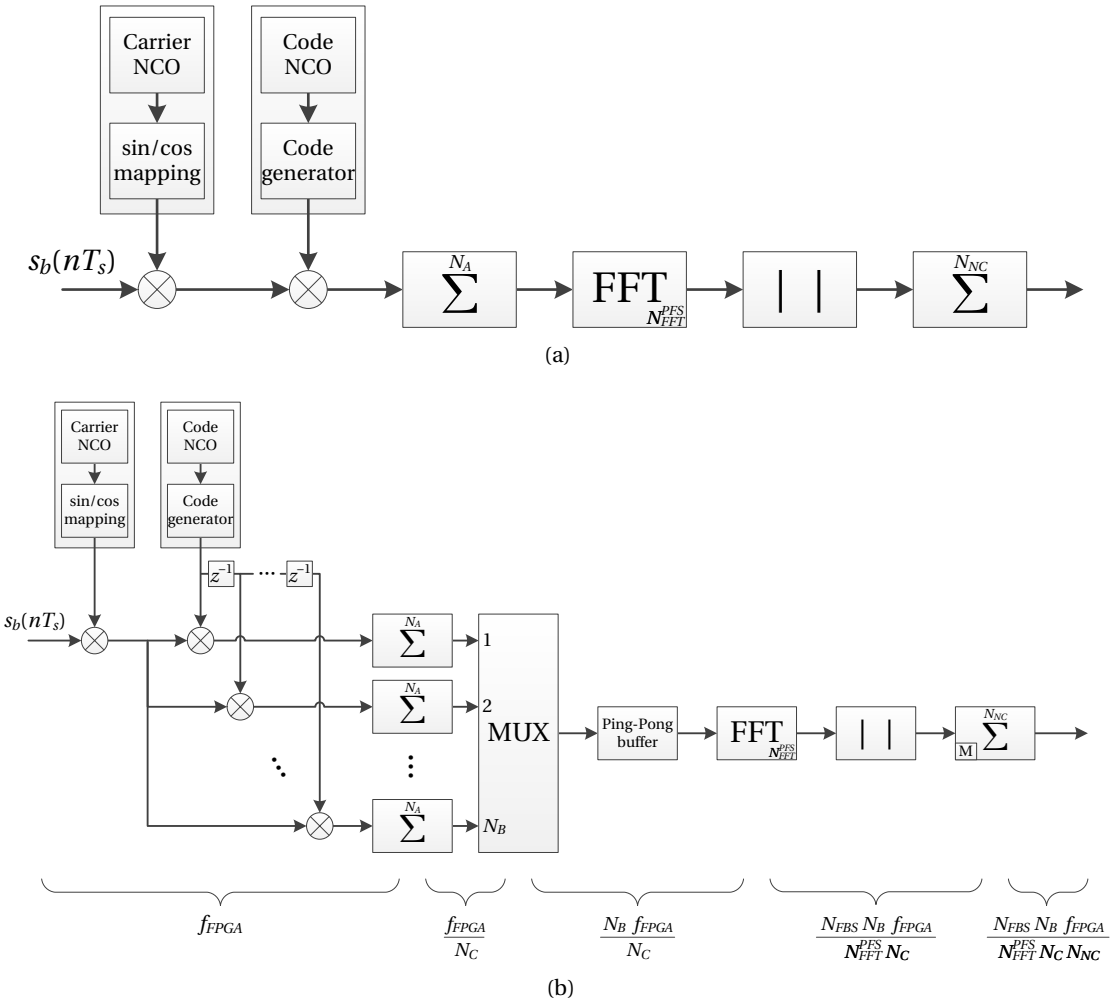


Figure 3.10: Implementation of the parallel frequency search architecture, (a) direct, (b) using duplication.

of the FFT bins may be necessary to cover the search space. The number of bins kept is then denoted N_{FBS}, which is equal to the number of frequency bins N_{FB} only if the entire frequency search space is covered by the FFT. The rate after the FFT is thus reduced by N_{FFT}^{PFS}/N_{FBS}. Finally there are the magnitude computation and the non-coherent accumulator based on memory blocks. The memory inside the non-coherent accumulator has $N_B N_{FBS}$ addresses in this case.

In this implementation, the resource usage of the logical elements is relatively similar to the serial search architecture because the accumulators are a little bit smaller and there is just one supplementary FFT, but the memory is far more used. However, there are two limitations with the implementation depicted in Fig. 3.10b. First, as for the serial search implementation, the number of branches is limited by the number of accumulations performed by the accumulator before the multiplexer. But here N_A is much lower than N_C. However, as before, this limit

may be easily circumvented by duplicating the multiplexer and the latter blocks. The second limitation is if zero-padding is used, because in this case the data rate after the FFT is higher than the data rate after the multiplexer. So, we should take care also to not obtain a rate higher than the FPGA rate. Otherwise, this limit can be circumvented in the same manner as before by duplicating the multiplexer and the latter blocks.

3.3.4 Parallelization

Now that the implementations of the different acquisition architectures have been described, we can determine the parallelization of each implementation, and the time needed to explore the entire acquisition search space.

With a coherent integration time of $T_C = N_C T_S$ and a number of non-coherent accumulations N_{NC}, the total integration time is $T_T = T_C N_{NC}$, and the number of samples to process is $N_C N_{NC}$. For a system running at the sampling frequency and using not any parallelism (i.e. the serial search without duplication), the time to test one code delay and one carrier frequency is thus $N_C N_{NC} T_S = T_T$, and the time to explore the entire search space is thus $T_T N_{CB} N_{FB}$, with N_{CB} the number of code bins and N_{FB} the number of frequency bins.

Until now, we did not discuss the question of the data transition. So here, we will consider two cases, either the data are ignored which implies a loss, or the alternate half-bits method is used, which requires to double the length of the input signal (see (van Diggelen [2009] and (Psiaki [2001]) respectively for more details). To differentiate these two cases, we use a parameter d, which is equal to 2 for the alternate half-bits method, and 1 otherwise.

Now, if we consider a system running at a higher frequency, the processing time will be divided by a factor $G_{FPGA} = \frac{f_{FPGA}}{f_S}$. Then, if we consider a parallelism, the processing time will be divided by a factor P that indicates the number of bins processed in parallel. Therefore, the time to explore the entire search space is

$$T_E = \frac{d T_T}{G_{FPGA}} \frac{N_{CB} N_{FB}}{P}. \tag{3.1}$$

Note that the time to load a new code or to modify the carrier and code frequencies on the channel, and the latency in the processing are not taken into account in this formula. Indeed, the loading time is very small, typically on the order of dozens of cycles. The latency is mainly due to the FFTs and corresponds to the size of the elements, i.e., a few thousands of clock cycles, whereas the input signal used is typically composed of hundreds of thousands of samples if high sensitivity is intended (since high sensitivity requires long integration times). Therefore, the latency represents only a low percentage of T_E. Note also that the time needed to store the signal into a memory before the processing (which is equal to $d T_T$) is also not taken into account in Eq. (3.1).

For the serial search (SS), the parallelization comes from the duplication of the elements, and

corresponds to the number of branches implemented. Therefore,

$$T_{E,SS} = \frac{dT_T}{G_{FPGA}} \frac{N_{CB} N_{FB}}{N_{B,SS}}. \tag{3.2}$$

For the parallel code search (PCS), the parallelization comes from the duplication of the elements and from the FFTs for the correlation. Therefore,

$$T_{E,PCS} = \frac{dT_T}{G_{FPGA}} \frac{N_{CB} N_{FB}}{N_{B,PCS} N_{FFT}^{PCS}}. \tag{3.3}$$

If the length of the FFT corresponds to the number of code bins, the equation simplifies to

$$T_{E,PCS} = \frac{dT_T}{G_{FPGA}} \frac{N_{FB}}{N_{B,PCS}}. \tag{3.4}$$

For the parallel frequency search (PFS), the parallelization comes from the duplication of the elements and from the FFT. Therefore,

$$T_{E,PFS} = \frac{dT_T}{G_{FPGA}} \frac{N_{CB} N_{FB}}{N_{B,PFS} N_{FBS}}. \tag{3.5}$$

If the FFT covers all the frequency bins, the equation simplifies to

$$T_{E,PFS} = \frac{dT_T}{G_{FPGA}} \frac{N_{CB}}{N_{B,PFS}}. \tag{3.6}$$

3.4 Application example

Now that the different parameters and implementations have been described, the performance of the three implementations is compared through an application example.

For this application, the implementation on a low-cost FPGA (the Altera Cyclone III EP3C120) and on a high-end FPGA (Altera Stratix III EP3SE260) is considered.

Then the signal considered is the GPS L1 C/A, with two cases. A stand-alone case where the receiver has no a priori information, and an assisted case where the receiver has a priori information on the Doppler frequency of the satellites, which reduces the frequency search space (Leclère et al. [2010]).

A sampling frequency of 4.096 MHz is selected, which is a good compromise between complexity and accuracy. The FPGA frequencies selected are multiples of the sampling frequency, and are realistic values obtained from real designs.

A sensitivity of −150 dBm is assumed, because this is the start of high sensitivity. The required coherent integration time and the number of non-coherent accumulation are then obtained

f_S	4.096 MHz	
FPGA	EP3C120	EP3SE260
f_{FPGA}	98.304 MHz	196.608 MHz
G_{FPGA}	24	48

Table 3.1: Summary of the FPGA parameters selected for the application.

Sensitivity	−150 dBm
T_C	10 ms
N_C	40 960
N_{NC}	40
T_T	400 ms

Table 3.2: Summary of the sensitivity parameters selected for the application.

Case	Assisted	Stand-alone
Frequency search space	1360 Hz	11 020 Hz
Frequency step (δ_f)	50 Hz	
N_{FB}	29	221
Code step (δ_C)	1 sample	
N_{CB}	4096 samples	

Table 3.3: Summary of the search space parameters selected for the application.

using the method from (van Diggelen [2009]), considering the alternate half-bits method for managing the data bit transitions (i.e. $d = 2$). These parameters are summarized in Tables 3.1, 3.2, and 3.3.

With such a long integration time, the maximum error allowed for the code chipping rate is about 0.156 chip/s (this ensures to have a shift smaller than half a sample). The PFS can thus search only ±240 Hz (0.156×1540) of the frequency search space simultaneously, i.e. $N_{FBS} = 11$.

3.4.1 Results

The details of the calculations are provided in Appendix C, and the results in terms of number of branches, parallelization, and time to explore the entire search space are given in Table 3.4. The number of branches gives the degree of duplication in the implementations depicted in Figs. 3.4, 3.8, 3.9, and 3.10b. The parallelization is the number of cells tested simultaneously,

		Low-cost FPGA EP3C120		High-end FPGA EP3SE260	
		Assisted case	Stand-alone case	Assisted case	Stand-alone case
Number of branches	$N_{B,SS}$	971		2911	
	$N_{B,PCS}$	2		8	
	N_{B,PCS^*}	-	4	-	11
	$N_{B,PFS}$	1095		3385	
Parallelization	P_{SS}	971		2911	
	P_{PCS}	8192		32 768	
	P_{PCS^*}	-	16 384	-	45 056
	P_{PFS}	12 045		37 235	
Time to explore the search space (ms)	$T_{E,SS}$	4078	31 075	680.1	5183
	$T_{E,PCS}$	483.3	3683	60.42	460.4
	T_{E,PCS^*}	-	1842	-	334.8
	$T_{E,PFS}$	328.7	2505	53.17	405.2

Table 3.4: Results and performance of the implementations.

and can be used to compare the implementations. The time to explore the entire search space is maybe more meaningful for GNSS users since it gives an idea of the processing time, and it can also be used to compare the implementations.

From Table 3.4, it can be seen that the SS implementation is the least efficient. Even with assistance, the result is worse than for the other implementations in the stand-alone case. The PFS implementation is slightly more efficient than the PCS implementation. But if the PCS implementation is optimized with shift in the frequency domain (PCS* implementation), then it becomes slightly better than PFS for the stand-alone case. In the assisted case, most of the frequencies that can be tested through the different branches fall outside of the frequency search space, and are thus useless.

3.4.2 Observations

Why the PFS and PCS are better than the SS ?

The SS and PFS implementations are identical until the multiplexer, except that the accumulators of the PFS are smaller since the integration length is smaller. Since the PFS has an FFT, the resource usage of the logical blocks is then almost equivalent for the two implementations. Since the PFS tests also several frequency bins simultaneously, it has a higher parallelism.

Regarding the PCS, it is well known that performing a convolution using an FFT is more efficient than with traditional filters, as soon as the filter length is more than 64 (Smith [2002] pp. 311–318). Therefore, it is not a surprise if the PCS is better than the SS.

Comparison of the PFS with the PCS

Each implementation has some drawbacks. Here we list them in order to determine weak point of the implementations, and when they are not suitable.

Regarding the PCS, the first drawback is that it uses a lot of DSP blocks with the FFTs. Although the high-end FPGA used was very rich in DSP blocks, it can be the element limiting the duplication (see Appendix C.2).

Second, the resolution for the data and the twiddle factors of the FFT in the PCS architecture needs to be higher than in the PFS, because the longer chain to compute the correlation (FFTs, multiplication, normalization to reduce the signal resolution and then IFFT), results in a propagation of the quantization errors.

Third, the PCS is more sensitive to the sampling frequency than the PFS. With the PFS implementation, to double the sampling frequency results in adding one bit in the coherent accumulators, i.e. $R+1$ bits to store instead of R. Thus we can interpolate roughly by saying that keeping the same hardware resources, the number of branches would be divided by $\frac{R+1}{R}$. Whereas with the PCS architecture, to double the sampling frequency doubles the size of the FFT and of the accumulators. Thus, the number of branches would be divided by 2. This means that using a higher sampling frequency than 4.096 MHz would be better for the PFS, but using a sampling frequency lower would be better for the PCS. However, this may be avoided if a resampling block is included in the acquisition channel, but this block would still require additional resources.

Regarding the PFS, the first drawback is a small loss of sensitivity. As discussed in Section 2.2.3, due to the small accumulation performed before the FFT, there is a loss proportional to the input carrier frequency. This loss depends on the accumulation time, but can easily reach more than 1 dB. In our application, we did not take into account this loss, this means that the actual sensitivity for the PFS would be slightly lower than −150 dBm.

But the main drawback of the PFS is that it cannot handle the Doppler effect on the code. If the code chipping rate was not altered, the entire frequency search space would be covered by the FFT, regardless of the total integration time used, and the PFS would be clearly better than the PCS. But since the code chipping rate is altered, the space searched by the FFT must be reduced. In our application, where a relatively long total integration time was considered, the PFS searches only 11 frequency bins simultaneously while the frequency space contains 221 bins in the stand-alone case. With a GNSS signal having a chipping rate of 10.23 Mchip/s instead of 1.023 Mchip/s, the effect of this drawback will be worst, and the number of bins searched simultaneously would be divided by the same factor.

Influence of the FPGA

From Table 3.4, it can be seen that despite the large differences in the absolute results for the two FPGAs, the ranking of the implementations is the same.

Generally, inside an FPGA family, the ratio between the different types of resources is relatively similar, i.e. using a bigger FPGA will provide an equivalent increase of the logical, memory and DSP blocks. Consequently for different FPGAs of the same family, we do not expect the ranking to change significantly.

Between different families, the ratios between logical and memory, as well as logical and DSP blocks, are different. For the same amount of logical blocks, a high-end FPGA will have more memory and DSP blocks than a low-cost FPGA. High-end FPGAs are consequently more suited for FFT-based implementations. However, this should not impact the ranking since the SS implementation is far inferior to the others in terms of performance.

High-end FPGAs also allows the use of a higher clock frequency, which improves the performance of all the implementations in the same manner.

3.5 Summary

In this chapter, we have presented a framework to compare the implementations of the main GNSS signals acquisition architectures on FPGAs. The implementations have been optimized towards achieving maximum parallelization for a single acquisition channel and fixed resources.

Considering the GPS L1 C/A signal and long integration times, it has been shown that the two FFT-based implementations are far more efficient than a simple duplication of mixers and accumulators. Then, these two implementations have provided similar results, with a slight advantage to the PFS over the PCS. However, we have shown the parameters that influences each implementations, and depending on those, the PCS can sometimes be more efficient than the PFS.

Moreover, the implementation of the PCS discussed here is straightforward, in the sense that no additional techniques are used. However, it is possible to use techniques to reduce the complexity, as those proposed in (Sajabi et al. [2006], Sagiraju et al. [2006], Qaisar et al. [2008]) for example. Also, here we did not use the fact that the local code is real, whereas we can use this fact to reduce the complexity of its FFT (see Appendix B).

After this comparison, we decided to direct our research towards the PCS rather than the PFS, for the following reasons :

1. The PFS cannot handle the code Doppler, which is important with long integration times, high Doppler, and high chipping rate. Long integrations are needed for high

sensitivity receivers. A high Doppler is usual in space applications, where GNSS is more and more used. And a high chipping rate is available with modern signals that allow a higher positioning accuracy. Therefore, the PFS does not seem adapted to answer these challenges.

2. It is relatively easy to have an estimate of the Doppler on the carrier frequency when we start a receiver. For a terrestrial user, the last position obtained by the receiver can be used and the almanac can be used to determine a rough position of the satellites. Therefore, the frequency search space can be reduced to dozens or hundreds of Hz instead of thousands of Hz. For a space user, the user position can be obtained using an orbital filter, and thus in the same way the frequency search space can be reduced to hundreds of Hz instead of dozens of thousands of Hz. Whereas having an estimate of the code delay is much more difficult to obtain.

3. Additional properties or methods can be used to reduce the complexity of the PCS, as stated before. Especially, using the fact the the local code is real allows a reduction of the resources without any degradation.

Advanced acquisition architectures Part II

The question of the reduction of the complexity for the acquisition of GNSS signals has been discussed for a long time, and a large variety of solutions have been proposed.

For example, some people tried to exploit the fact that the local code is binary for the design of fast correlation algorithms (Guo et al. [1991b], Guo et al. [1991a]). Theoretically, this means that it is possible to not have any multiplications. However, the number of additions is much higher as compared to an FFT algorithm, making the proposed method not efficient in practice.

Regarding the parallel code search, different methods have been proposed to reduce the complexity. For example, (Starzyk and Zhu [2001]) proposed to perform an average over few samples in order to get one point per chip, and thus a smaller FFT (however the codes length contain high prime factors). (Sajabi et al. [2006]) proposed to use only half of the samples to compute the IFFT in the correlation, and (Sagiraju et al. [2006]) proposed to sum samples to reduce the IFFT length. (Qaisar et al. [2008]) proposed to simply filter and downsample the signal before performing the correlation. (Hassanieh et al. [2012]) and (Rao and Ratnam [2013]) proposed algorithms based on the sparse FFT. Of course, the price to pay is usually a reduction of the SNR.

Then, the advent of new GNSS signals has brought new constraints (longer codes, tiered codes, higher chipping rate, new modulations), which has thus required new algorithms. For some specific context, like the acquisition in presence of transition (due to a secondary code or data), specific solutions have been proposed (detailed in Chapter 5), but most of the time at the expense of a reduction of the SNR (and thus a reduction of the probability of detection).

Therefore, when we have carried out this research, the idea was to exploit the characteristics of the GNSS signals to compute the circular correlation accurately and efficiently, and not to obtain approximations that impact the detection performance. In this part of the thesis, we thus concentrate only on the circular correlation, represented in Fig. 3.11, where x_n corresponds to the signal after the multiplication with the local carrier, h_n is the local code, and y_n is their circular correlation. Moreover, we do not discuss side problems, such as the impacts of the oscillator effects such as phase noise or the user dynamics (because the coherent integration times considered are short enough (van Diggelen [2014])), or interferences.

In Chapter 4, we propose some ideas to reduce the processing time or the resource of the FFT and of the circular correlation on Altera FPGAs. These algorithms can be used with any GNSS

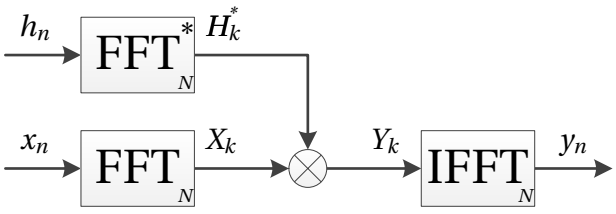


Figure 3.11: Operation considered for the next chapters : The circular correlation computed using FFTs.

signal, and even with any signal, since we simply use the properties of the Altera FFT, and not the characteristics of the signal.

In Chapter 5, we focus on the acquisition of GNSS signals in presence of sign transition where the coherent integration time corresponds to a period of the primary code. We start from a known solution to the problem, and improve it to reduce the complexity.

In Chapter 6, we focus on the acquisition of GNSS signals where the coherent integration time corresponds to a period of the tiered code. We discuss first implementations to use small FFTs (because large FFTs are sometimes not possible to implement), and then implementations to reduce the complexity using the specificities of the secondary code.

4 Efficient FFT and correlation implementations on Altera FPGAs

In this chapter, we discuss the implementation of the FFT on Altera FPGAs, and the implementation of the circular correlation using FFTs (the discussion can also be applied to the circular convolution). Using known algorithms, it is shown that is is possible to obtain more efficient implementations than the traditional ones, by reducing the amount of resources, especially the memory.

Some of the work presented here – the implementation of the correlation with a separation by downsampling – has been published in (Leclère et al. [2012]).

4.1 Description of the Altera FFT

The implementation of an FFT algorithm on an FPGA is not an easy task. Hopefully, FPGA companies provide an FFT as Intellectual Property (IP). In this thesis, we will discuss of the FFT provided by Altera, because we use Altera FPGAs in our research projects. However the characteristics of the FFT provided by Xilinx are probably similar.

The Altera FFT is highly configurable, for example we can select :

- The transform length, which must be a power of two. Currently, the minimum length is 8 points and the maximum length is 262 144 points (this is for the version 13.0 of November 2013, the maximum length may grow in the next years).
- The input/output (I/O) data flow (more details about this are provided below).
- The number of bits to quantify the input and output data, and the twiddle factors.
- The order of the input and of the output (natural order or bit-reversed order, see Section B.1 for more information on this).
- Some options for the FFT engine.

- Some options for the implementation of the complex multipliers (we can use 4 multipliers and 2 adders or 3 multipliers and 5 adders, and we can implement them using DSP blocks or logic cells).
- The repartition of the memory between the different memory types.

There are four I/O data flows available : variable streaming, streaming, buffered burst, burst. For the variable streaming and the streaming I/O data flows, the input and output data flow can be continuous, without any break between consecutive transforms. The corresponding timing diagram of an N-point FFT (shown in Fig. 4.1a) is given Fig. 4.1b. Between the last input sample and the first output sample of the FFT, there is a latency, denoted L_N, which depends on the transform length. Therefore, in this case, the Pth FFT result is fully available after $N + L_N + PN = (P+1)N + L_N$ clock cycles. For the burst I/O data flow, it is possible to load a new input only when the output is completely unloaded, as shown in Fig. 4.1c. This means that the throughput is reduced compared to the variable streaming and streaming implementations. The buffered burst data flow is between the two previous cases. The flow

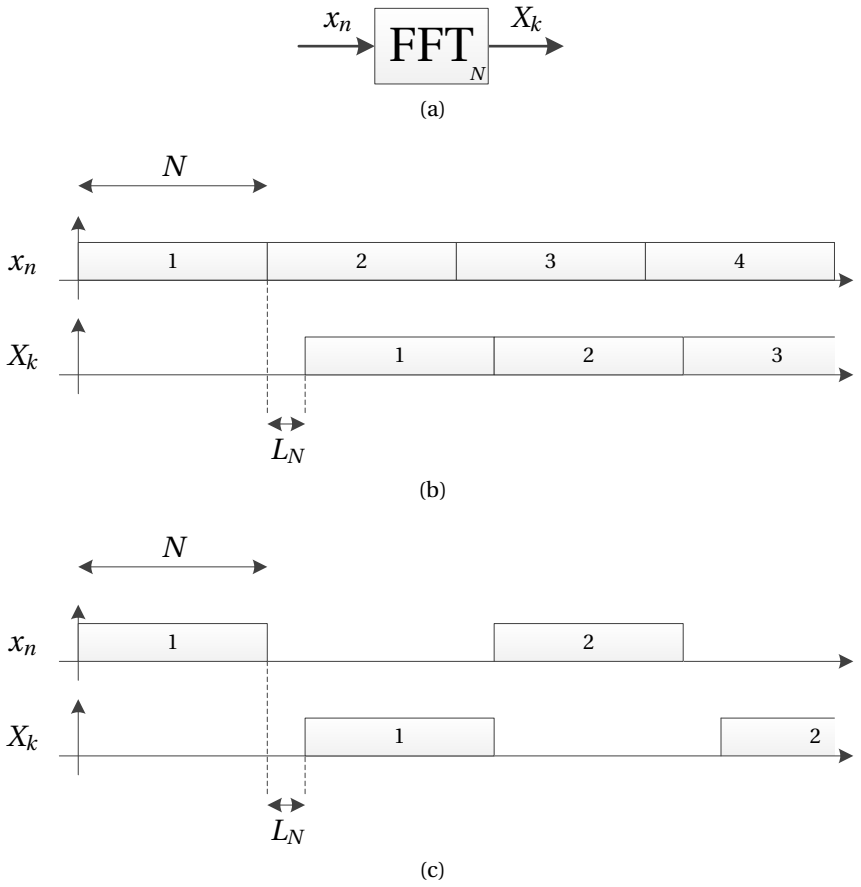


Figure 4.1: (a) N-point Altera FFT, (b) timing diagram of an N-point Altera FFT with the streaming I/O data flow, (c) timing diagram of an N-point Altera FFT with the burst I/O data flow. The number in the boxes identifies the sequences.

I/O data flow	Inverse of the throughput* (cycle)	Logic usage (ALUT)	Memory usage (M9K+)	Multipliers usage (DSP element)
Streaming	4096	7326	76	24
Buffered Burst	4608	7607	60	24
Burst	10 861	7151	28	24

Table 4.1: Resources estimated with the Altera MegaWizard Plug-in Manager for an FFT of 4096 points implemented on a Stratix III FPGA, considering 18 bits for the data and twiddle precision, and 2 FFT engines with quad output. *The inverse of the throughput is defined as the minimum number of cycles between the start of two consecutive periods. +One M9K is a memory of 9 Kibit = 9216 bits.

cannot be continuous, but it is not required to wait for the complete unload of the output samples before loading new input samples.

Of course, the higher is the throughput, the higher are the required resources. For example, an estimate of the resources is provided Table 4.1 for three I/O data flows. Playing with the engine options may lower the resources (logic, memory and DSP) for the buffered burst and burst I/O data flows, in exchange of a reduced throughput.

Due to the large number of possibilities for the FFT implementation, for the evaluation of the resources in the following sections, we will consider a Stratix III FPGA, a transform length of 2048, the streaming I/O data flow, a data and twiddle precision of 18 bits, complex multipliers implemented in DSP blocks using four real multipliers, and no logic function implemented in memory. However, the discussion of this chapter can be applied to other FPGA families and with other FFT parameters.

Note that the Altera FFT can be used to compute the FFT or the IFFT. More details about the Altera FFT can be found in (Altera [2013]).

4.2 How to compute an FFT on Altera FPGAs more efficiently than the Altera FFT

4.2.1 Computing an FFT of N points using two FFTs of $N/2$ points

The DFT of a sequence x_n of N points is defined as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi kn}{N}}, \tag{4.1}$$

with $k = 0, 1, \ldots, N-1$.

Input separated by parity, output separated by section

If we separate the input sequence in even and odd samples, we have

$$\begin{aligned} X_k &= \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi k(2n)}{N}} + \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi k(2n+1)}{N}} \\ &= \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi kn}{N/2}} + e^{-\frac{j2\pi k}{N}} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi kn}{N/2}}. \end{aligned} \tag{4.2}$$

Of course, this requires that N is divisible by 2, which will be assumed throughout this chapter. The first half of the DFT is

$$X_k = \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi kn}{N/2}} + e^{-\frac{j2\pi k}{N}} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi kn}{N/2}}, \tag{4.3}$$

with $k = 0, 1, \ldots, N/2-1$ (the equation is the same, only the range of k has changed). The two sums correspond to the DFT of the sequences x_{2n} and x_{2n+1}, respectively. The second half of the DFT is

$$\begin{aligned} X_{k+N/2} &= \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi(k+N/2)n}{N/2}} + e^{-\frac{j2\pi(k+N/2)}{N}} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi(k+N/2)n}{N/2}} \\ &= \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi kn}{N/2}} - e^{-\frac{j2\pi k}{N}} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi kn}{N/2}}, \end{aligned} \tag{4.4}$$

with $k = 0, 1, \ldots, N/2-1$. The two sums also correspond to the DFT of the sequences x_{2n} and x_{2n+1}, respectively. Therefore, an FFT of N points can be computed using two FFTs of $N/2$ points as shown in Fig. 4.2a.

Input separated by section, output separated by parity

If we separate the input sequence in two sections, we have

$$\begin{aligned} X_k &= \sum_{n=0}^{N/2-1} x_n e^{-\frac{j2\pi kn}{N}} + \sum_{n=N/2}^{N-1} x_n e^{-\frac{j2\pi kn}{N}} \\ &= \sum_{n=0}^{N/2-1} x_n e^{-\frac{j2\pi kn}{N}} + \sum_{n=0}^{N/2-1} x_{n+N/2} e^{-\frac{j2\pi k(n+N/2)}{N}} \\ &= \sum_{n=0}^{N/2-1} x_n e^{-\frac{j2\pi kn}{N}} + e^{-j\pi k} \sum_{n=0}^{N/2-1} x_{n+N/2} e^{-\frac{j2\pi kn}{N}} \\ &= \sum_{n=0}^{N/2-1} \left(x_n + e^{-j\pi k} x_{n+N/2} \right) e^{-\frac{j2\pi kn}{N}}. \end{aligned} \tag{4.5}$$

The even samples of the DFT correspond to

$$\begin{aligned} X_{2k} &= \sum_{n=0}^{N/2-1} \left(x_n + e^{-j\pi(2k)} x_{n+N/2}\right) e^{-\frac{j2\pi(2k)n}{N}} \\ &= \sum_{n=0}^{N/2-1} \left(x_n + x_{n+N/2}\right) e^{-\frac{j2\pi kn}{N/2}}, \end{aligned} \tag{4.6}$$

with $k = 0, 1, \ldots, N/2-1$. This corresponds to the DFT of the sequence $x_n + x_{n+N/2}$. The odd samples of the DFT correspond to

$$\begin{aligned} X_{2k+1} &= \sum_{n=0}^{N/2-1} \left(x_n + e^{-j\pi(2k+1)} x_{n+N/2}\right) e^{-\frac{j2\pi(2k+1)n}{N}} \\ &= \sum_{n=0}^{N/2-1} \left(x_n - x_{n+N/2}\right) e^{-\frac{j2\pi n}{N}} e^{-\frac{j2\pi kn}{N/2}}, \end{aligned} \tag{4.7}$$

with $k = 0, 1, \ldots, N/2-1$. This corresponds to the DFT of the sequence $\left(x_n - x_{n+N/2}\right)e^{-\frac{j2\pi n}{N}}$. Therefore, an FFT of N points can also be computed using two FFTs of $N/2$ points as shown in Fig. 4.2c.

4.2.2 Computing an IFFT of N points using two IFFTs of $N/2$ points

The IDFT of a sequence X_k of N points is defined as

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{j2\pi kn}{N}}, \tag{4.8}$$

with $n = 0, 1, \ldots, N-1$.

Input separated by section, output separated by parity

If we separate the input sequence in two sections, we have

$$\begin{aligned} x_n &= \frac{1}{N} \sum_{k=0}^{N/2-1} X_k e^{\frac{j2\pi kn}{N}} + \frac{1}{N} \sum_{k=N/2}^{N-1} X_k e^{\frac{j2\pi kn}{N}} \\ &= \frac{1}{N} \sum_{k=0}^{N/2-1} X_k e^{\frac{j2\pi kn}{N}} + \frac{1}{N} \sum_{k=0}^{N/2-1} X_{k+N/2} e^{\frac{j2\pi(k+N/2)n}{N}} \\ &= \frac{1}{N} \sum_{k=0}^{N/2-1} X_k e^{\frac{j2\pi kn}{N}} + e^{j\pi n} \frac{1}{N} \sum_{k=0}^{N/2-1} X_{k+N/2} e^{\frac{j2\pi kn}{N}} \\ &= \frac{1}{N} \sum_{k=0}^{N/2-1} \left(X_k + e^{j\pi n} X_{k+N/2}\right) e^{\frac{j2\pi kn}{N}}. \end{aligned} \tag{4.9}$$

The even samples of the IDFT correspond to

$$\begin{aligned} x_{2n} &= \frac{1}{N}\sum_{k=0}^{N/2-1}\left(X_k + e^{j\pi(2n)}X_{k+N/2}\right)e^{\frac{j2\pi k(2n)}{N}} \\ &= \frac{1}{N}\sum_{k=0}^{N/2-1}\left(X_k + X_{k+N/2}\right)e^{\frac{j2\pi kn}{N/2}}, \end{aligned} \tag{4.10}$$

with $n = 0, 1, \ldots, N/2-1$. This corresponds to the IDFT of the sequence $X_k + X_{k+N/2}$. The odd samples of the IDFT correspond to

$$\begin{aligned} x_{2n+1} &= \frac{1}{N}\sum_{k=0}^{N/2-1}\left(X_k + e^{j\pi(2n+1)}X_{k+N/2}\right)e^{\frac{j2\pi k(2n+1)}{N}} \\ &= \frac{1}{N}\sum_{k=0}^{N/2-1}\left(X_k - X_{k+N/2}\right)e^{\frac{j2\pi k}{N}}e^{\frac{j2\pi kn}{N/2}}, \end{aligned} \tag{4.11}$$

with $n = 0, 1, \ldots, N/2-1$. This corresponds to the IDFT of the sequence $(X_k - X_{k+N/2})\,e^{\frac{j2\pi k}{N}}$. Therefore, an IFFT of N points can be computed using two IFFTs of $N/2$ points as shown in Fig. 4.2b.

Input separated by parity, output separated by section

If we separate the input sequence in even and odd samples, we have

$$\begin{aligned} x_n &= \frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k}e^{\frac{j2\pi(2k)n}{N}} + \frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k+1}e^{\frac{j2\pi(2k+1)n}{N}} \\ &= \frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k}e^{\frac{j2\pi kn}{N/2}} + e^{\frac{j2\pi n}{N}}\frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k+1}e^{\frac{j2\pi kn}{N/2}}. \end{aligned} \tag{4.12}$$

The first half of the IDFT is

$$x_n = \frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k}e^{\frac{j2\pi kn}{N/2}} + e^{\frac{j2\pi n}{N}}\frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k+1}e^{\frac{j2\pi kn}{N/2}}, \tag{4.13}$$

with $n = 0, 1, \ldots, N/2-1$. The two sums correspond to the IDFT of the sequences X_{2k} and X_{2k+1}, respectively. The second half of the IDFT is

$$\begin{aligned} x_{n+N/2} &= \frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k}e^{\frac{j2\pi k(n+N/2)}{N/2}} + e^{\frac{j2\pi(n+N/2)}{N}}\frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k+1}e^{\frac{j2\pi k(n+N/2)}{N/2}} \\ &= \frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k}e^{\frac{j2\pi kn}{N/2}} - e^{\frac{j2\pi n}{N}}\frac{1}{N}\sum_{k=0}^{N/2-1}X_{2k+1}e^{\frac{j2\pi kn}{N/2}}, \end{aligned} \tag{4.14}$$

with $n = 0, 1, \ldots, N/2-1$. The two sums also correspond to the IDFT of the sequences X_{2k} and X_{2k+1}, respectively. Therefore, an IFFT of N points can also be computed using two IFFTs of $N/2$ points as shown in Fig. 4.2d.

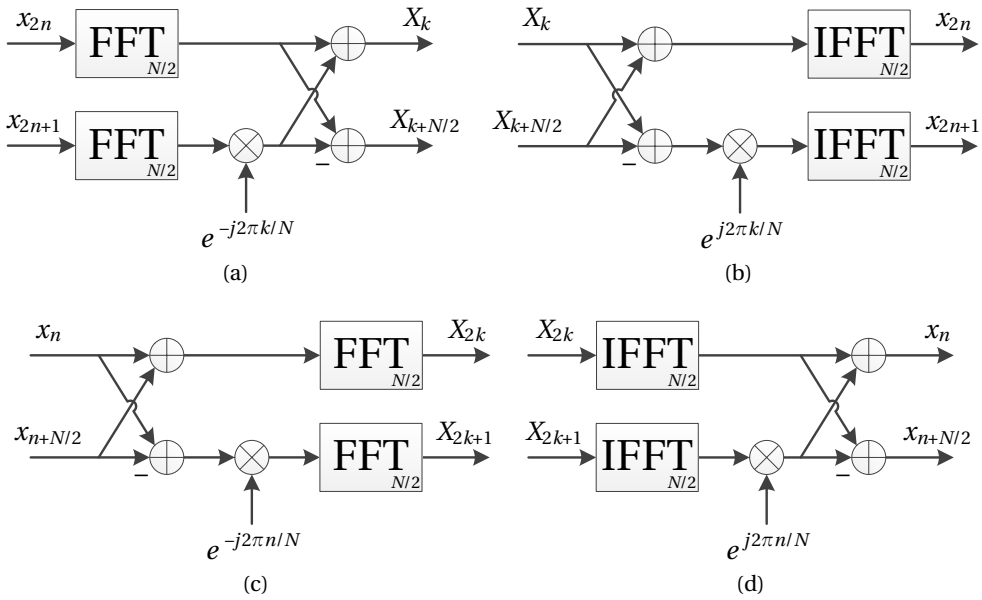


Figure 4.2: Computation of an N-point FFT using two $N/2$-point FFTs, (a) where the input is separated by parity and the output is separated by section, (c) where the input is separated by section and the output is separated by parity. Computation of an N-point IFFT using two $N/2$-point IFFTs, (b) where the input is separated by section and the output is separated by parity, (d) where the input is separated by parity and the output is separated by section.

4.2.3 Theoretical complexity

Let's consider that an N-point FFT requires $\frac{N}{2}\log_2 N$ complex multiplications and $N\log_2 N$ complex additions (Lyons [2010] pp. 135–159). The implementations in Fig. 4.2 then require $2\left(\frac{N}{4}\log_2\frac{N}{2}\right)+\frac{N}{2} = \frac{N}{2}\log_2 N$ complex multiplications, and $2\left(\frac{N}{2}\log_2\frac{N}{2}\right)+2\frac{N}{2} = N\log_2 N$ complex additions.

Thus, the number of operations is identical in both cases, which is normal since the separation presented corresponds precisely to the first step of the radix-2 FFT algorithm. Therefore, at first glance, the implementations of Fig. 4.2 seem useless, but actually not, as shown in the next sections.

4.2.4 Application to reduce the processing time

For the following, we will consider the implementation of Fig. 4.2a, but the same results would be obtained with any implementation of Fig. 4.2. The corresponding timing diagram using the Altera FFT is depicted Fig. 4.3, where $L_{N/2}$ denotes the latency of the FFT when the transform length is $\frac{N}{2}$. It can be seen that the Pth FFT result is fully available after $\frac{N}{2}+L_{N/2}+P\frac{N}{2} = (P+1)\frac{N}{2}+L_{N/2}$ clock cycles. Therefore, compared to the direct implementation of an N-point FFT, the processing time is approximately halved (see Fig. 4.1b).

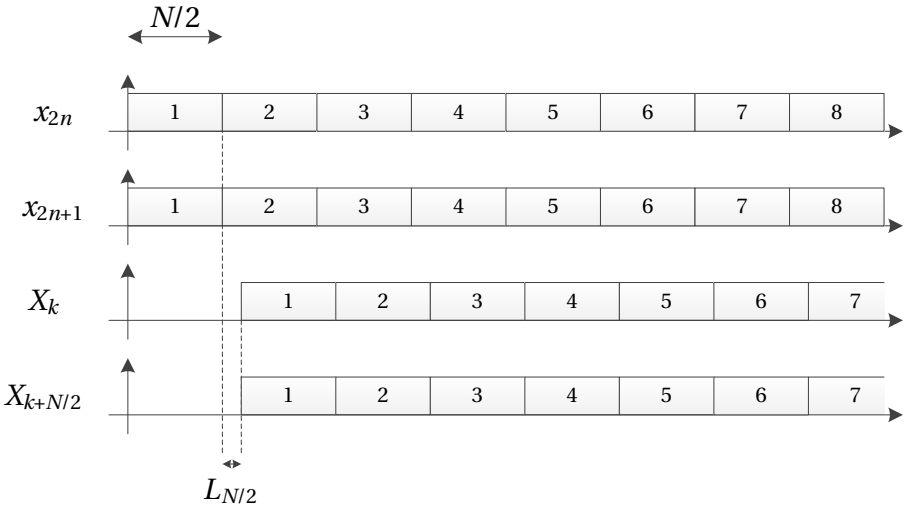


Figure 4.3: Timing diagram of the implementation of Fig. 4.2a using Altera FFTs. The number in the boxes identifies the sequences.

Implementation	Function	Logic usage (ALUT)	Memory usage (M9K)	Multipliers usage (DSP element)
Fig. 4.2a	2 1024-point FFTs	2 × 5248	2 × 19	2 × 12
	1 NCO	180	2	4
	1 Multiplier	0	0	4
	2 Adders	2 × 36	2 × 0	2 × 0
	Total	10 748	40	32
Fig. 4.1a	1 2048-point FFT	6906	38	24
	Total	6906	38	24
Ratio		1.56	1.05	1.33

Table 4.2: Comparison of the resources for Fig. 4.2a and Fig. 4.1a using the Altera FFT with $N = 2048$.

For the evaluation of the resources, we consider $N = 2048$. The complex exponential in Fig. 4.2a can be generated using the NCO IP provided by Altera. Therefore, the resources for the FFT and the NCO are estimated with the Altera MegaWizard Plug-In Manager (the parameters for the NCO are keep to the default ones), and the models defined in Appendix C are used for the other elements (multiplier and adder). The summary of the resources is given Table. 4.2. It can be seen that the resources are higher for the implementation of Fig. 4.2a (two $N/2$-point FFTs) than Fig. 4.1a (one N-point FFT). However, we have seen just before that the processing time for Fig. 4.2a was divided by a factor two. Since the resources are increased by a factor less than two, the implementation of Fig. 4.2a is more efficient than the implementation of Fig. 4.1a.

4.2.5 Application to reduce the resources

In the previous section, the proposed implementation was more efficient, but the resources were increased. In this section, we adapt it to use only one FFT, at the expense of an additional memory. The implementation is given Fig. 4.4, and the corresponding timing diagram is given Fig. 4.5. The idea is to first compute the FFT of the even samples of the sequence, and to store the result in a memory. Then, we compute the FFT of the odd samples of the sequence. When the result is available, we read the memory and we compute the first half and the second half of the FFT of the initial sequence. The first half is outputted while the second half is stored in the memory. Once the first half is fully outputted, the memory is read and the second half is outputted. In this way the FFT result is provided in the exact same order as with Fig. 4.1a. In this case, the Pth FFT result is fully available after $\frac{N}{2} + L_{N/2} + \frac{N}{2} + P\frac{N}{2} = (P+1)N + L_{N/2}$ clock cycles, which is slightly lower than for Fig. 4.1a because of the lower latency.

The corresponding resources are given Table 4.3. For the memory, we need to store twice (because the signal is complex) 1024×18 bits, which requires 4 M9K memories, and we consider few logic for the addressing. It can be seen that the resources are reduced, by 20 % for the logic, 34 % for the memory, and 17 % for the DSP elements, which is not negligible.

If we extrapolate to larger transform lengths, the results regarding the logic and the memory would be about the same, however for the DSP elements the results would be not as good because the number of DSP elements does not increase when we increase the transform length above 2048.

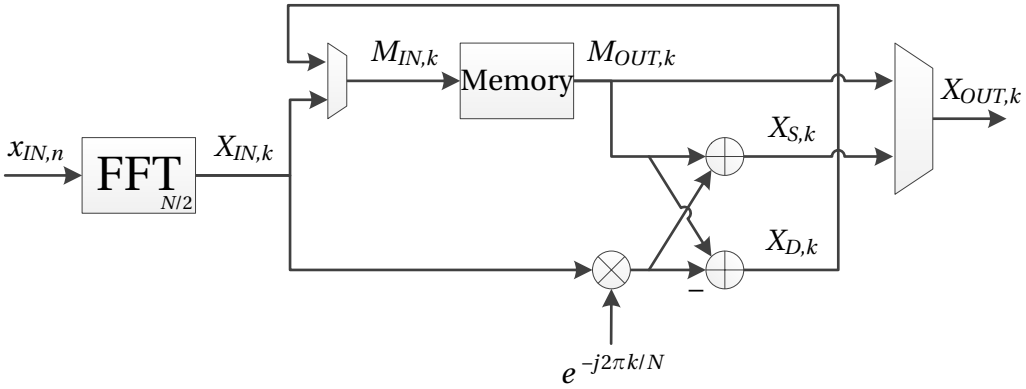


Figure 4.4: Computation of an N-point FFT using one $N/2$-point FFTs and a memory.

4.3 Efficient implementation of the correlation on Altera FPGAs

In this section, we propose alternative architectures to compute a circular correlation that use smaller FFTs than the traditional architecture, in the same way as in the previous section. We show different methods to obtain slightly different architectures, and then we perform an evaluation with the most interesting one.

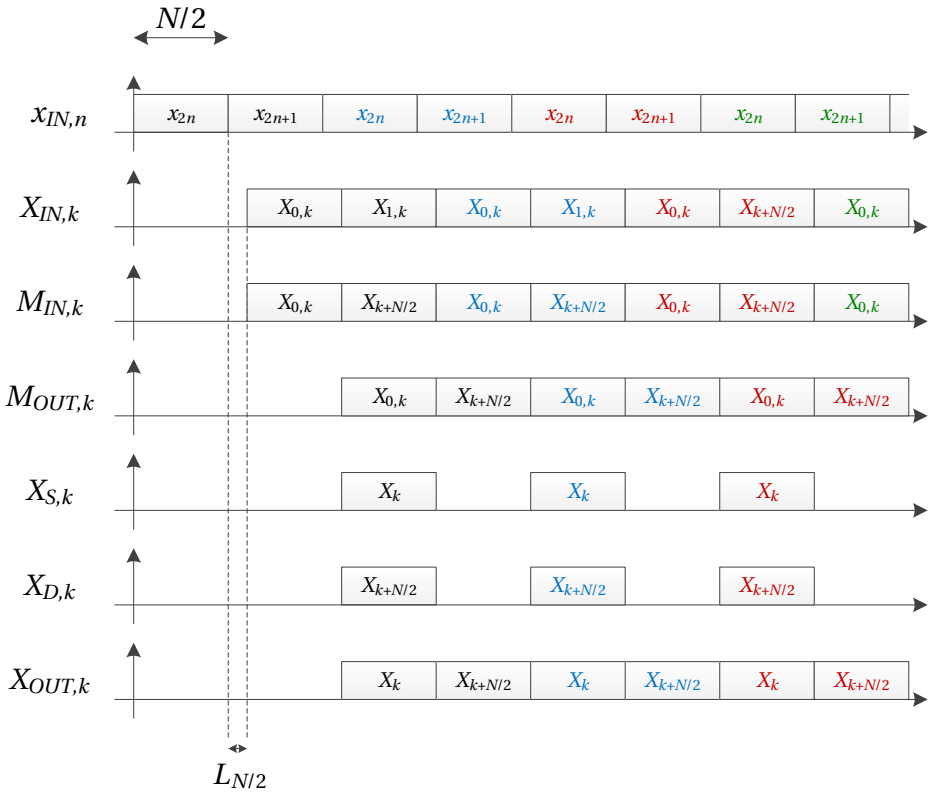


Figure 4.5: Timing diagram corresponding to Fig. 4.4 using Altera FFTs. The colors inside the boxes identify the sequences.

Implementation	Function	Logic usage (ALUT)	Memory usage (M9K)	Multipliers usage (DSP element)
Fig. 4.4	1 1024-point FFT	5248	19	12
	1 NCO	180	2	4
	1 Multiplier	0	0	4
	2 Adders	2 × 36	2 × 0	2 × 0
	1 Memory	22	4	0
	Total	5522	25	20
Fig. 4.1a	2048-point FFT	6906	38	24
	Total	6906	38	24
Ratio		0.80	0.66	0.83

Table 4.3: Resources for Fig. 4.4 and comparison with the direct use of an N-point FFT, with $N = 2048$.

4.3.1 Traditional correlation implementation with FFTs

The circular correlation y_n of two sequences h_n and x_n of N points is defined as

$$y_n = \sum_{k=0}^{N-1} h_k^* x_{(n+k) \bmod N}, \tag{4.15}$$

with $n = 0, 1, \ldots, N-1$, and mod denotes the modulo operation, i.e. $(n + mN) \bmod N = n$ with $m \in \mathbb{Z}$. The circular correlation can also be expressed as

$$Y_k = H_k^* X_k, \tag{4.16}$$

where Y_k, H_k and X_k are the DFTs of y_n, h_n and x_n, respectively (see Appendix A.3.3). So, by computing the IDFT of $H_k^* X_k$ we obtain y_n. Therefore, the circular correlation can be computed efficiently as shown in Fig. 4.6.

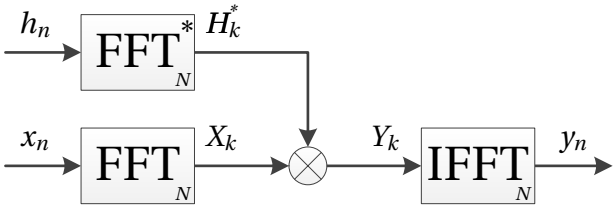


Figure 4.6: Computation of the circular correlation of two sequences of N points using FFTs.

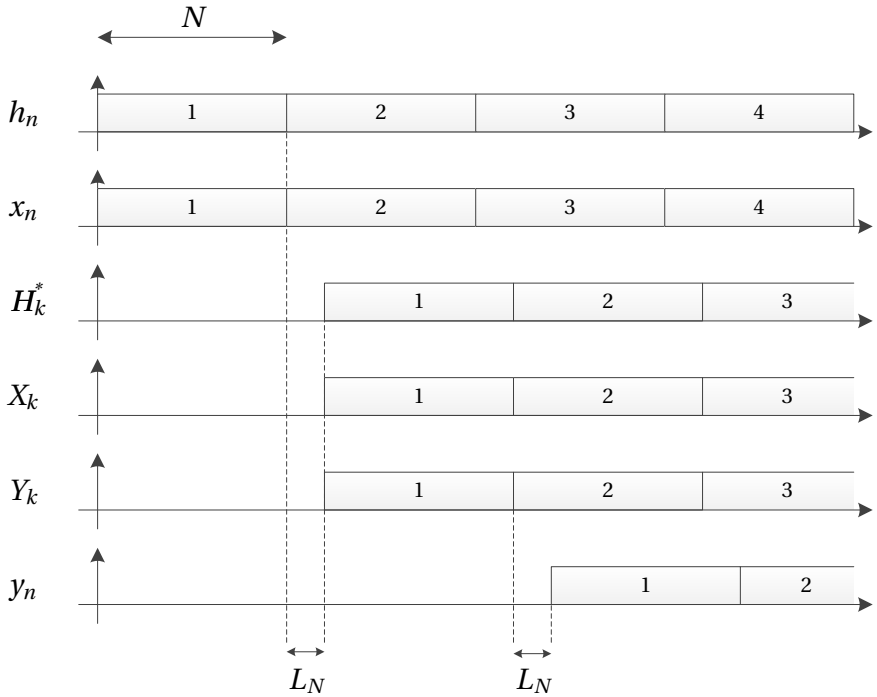


Figure 4.7: Timing diagram of Fig. 4.6 using Altera FFTs. The number in the boxes identifies the sequences.

The timing diagram corresponding to Fig. 4.6 using the Altera FFT is depicted Fig. 4.7. It can be seen that the Pth correlation result is fully available after $N+L_N+N+L_N+PN = (P+2)N+2L_N$ clock cycles.

4.3.2 FFT separation

It is possible to compute the circular correlation using the separations of the FFT and IFFT of Fig. 4.2, to either obtain the implementation of Fig. 4.8a, where the inputs and the output are separated in even and odd samples, or the implementation of Fig. 4.8b, where the inputs and the output are separated in sections.

4.3.3 Separation using the Chinese remainder theorem

The circular correlation can also be expressed as

$$Y(z) = H^*(1/z^*)X(z) \mod (z^{-N}-1), \tag{4.17}$$

where $Y(z)$, $H(z)$ and $X(z)$ are the z transforms of y_n, h_n and x_n, respectively, and N is the length of the sequences (see Appendix A.3.3). Noting that $z^{-N}-1 = (z^{-N/2}-1)(z^{-N/2}+1)$, we can define

$$Y_0(z) = Y(z) \mod (z^{-N/2}-1) \tag{4.18}$$

$$Y_1(z) = Y(z) \mod (z^{-N/2}+1) \tag{4.19}$$

and using the Chinese remainder theorem (CRT) (see Nussbaumer [1982], Ding et al. [1996]), we have

$$\begin{aligned} 2Y(z) &= \left(\frac{Y_0(z)}{z^{-N/2}-1} - \frac{Y_1(z)}{z^{-N/2}+1}\right)(z^{-N}-1) \\ &= (z^{-N/2}+1)Y_0(z) - (z^{-N/2}-1)Y_1(z) \\ &= \big(Y_0(z)+Y_1(z)\big) + z^{-N/2}\big(Y_0(z)-Y_1(z)\big). \end{aligned} \tag{4.20}$$

Detailing $Y_0(z)$ and $Y_1(z)$, we have

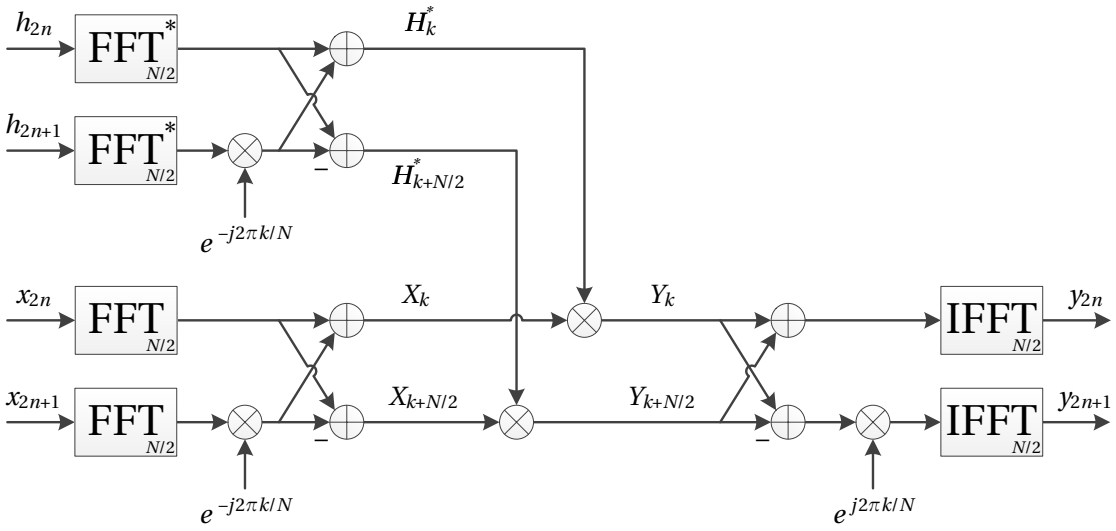
$$Y_0(z) = H^*(1/z^*)X(z) \mod (z^{-N/2}-1) = H_0^*(1/z^*)X_0(z) \mod (z^{N/2}-1) \tag{4.21}$$

$$Y_1(z) = H^*(1/z^*)X(z) \mod (z^{-N/2}+1) = H_1^*(1/z^*)X_1(z) \mod (z^{N/2}+1), \tag{4.22}$$

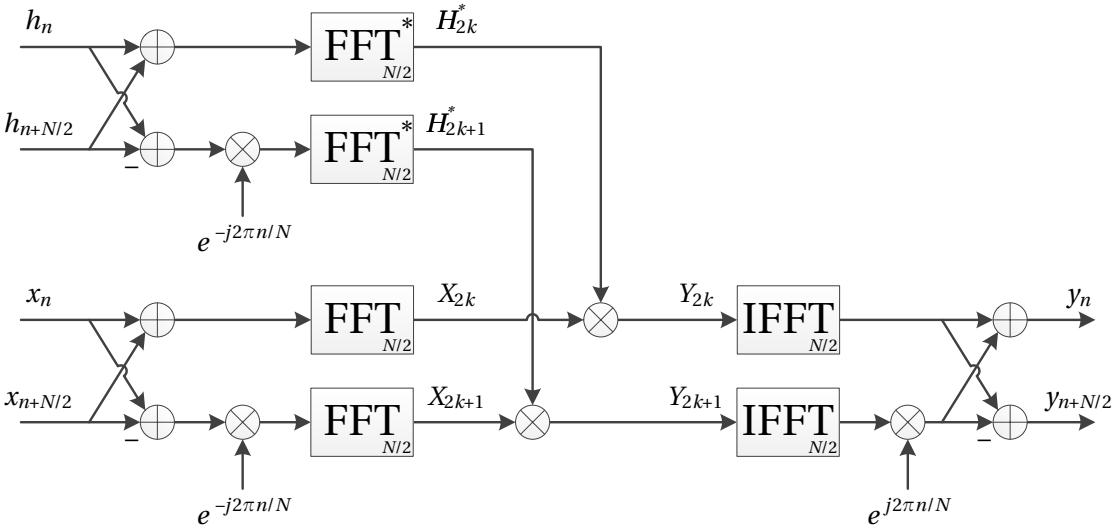
where

$$H_0(1/z^*) = H(1/z^*) \mod (z^{-N/2}-1), \quad H_1(1/z^*) = H(1/z^*) \mod (z^{-N/2}+1), \tag{4.23}$$

$$X_0(z) = X(z) \mod (z^{-N/2}-1), \quad X_1(z) = X(z) \mod (z^{-N/2}+1). \tag{4.24}$$



(a)



(b)

Figure 4.8: Computation of a circular correlation of N points using $N/2$-point FFTs, (a) where the inputs and the output are separated by parity, (b) where the inputs and the output are separated by section.

Eq. (4.21) corresponds to a circular correlation, and Eq. (4.22) corresponds to a skew-circular correlation (see Appendix A.3.4). In Eq. (4.23), $H_0(z)$ and $H_1(z)$ are the z transforms of $h_{0,n}$ and $h_{1,n}$, where $h_{0,n}$ is the sum of the first $N/2$ samples and of the last $N/2$ samples of h_n, and $h_{1,n}$ is the difference of the first $N/2$ samples and of the last $N/2$ samples of h_n. Idem for Eq. (4.24).

The corresponding implementation is given Fig. 4.9. It can be seen that in fact, this architecture is identical to the one of Fig. 4.8b.

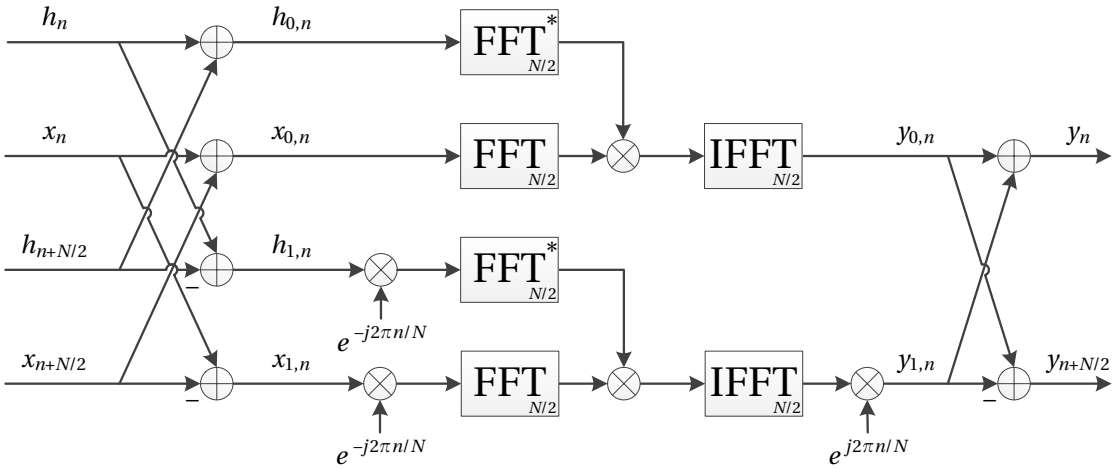


Figure 4.9: Computation of a circular correlation of N points using $N/2$-point FFTs (algorithm based on the Chinese remainder theorem).

4.3.4 Separation by downsampling

Matrix view

Using the matrix notation, the circular correlation y_n of two sequences h_n and x_n of length N can be expressed as

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-4} \\ y_{N-3} \\ y_{N-2} \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & \cdots & h_{N-4} & h_{N-3} & h_{N-2} & h_{N-1} \\ h_{N-1} & h_0 & h_1 & h_2 & \cdots & h_{N-5} & h_{N-4} & h_{N-3} & h_{N-2} \\ h_{N-2} & h_{N-1} & h_0 & h_1 & \cdots & h_{N-6} & h_{N-5} & h_{N-4} & h_{N-3} \\ h_{N-3} & h_{N-2} & h_{N-1} & h_0 & \cdots & h_{N-7} & h_{N-6} & h_{N-5} & h_{N-4} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ h_4 & h_5 & h_6 & h_7 & \cdots & h_0 & h_1 & h_2 & h_3 \\ h_3 & h_4 & h_5 & h_6 & \cdots & h_{N-1} & h_0 & h_1 & h_2 \\ h_2 & h_3 & h_4 & h_5 & \cdots & h_{N-2} & h_{N-1} & h_0 & h_1 \\ h_1 & h_2 & h_3 & h_4 & \cdots & h_{N-3} & h_{N-2} & h_{N-1} & h_0 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-4} \\ x_{N-3} \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \tag{4.25}$$

$$\mathbf{y} = \mathbf{H}^* \, \mathbf{x}.$$

If we separate y_n in even and odd samples, we have

$$\begin{bmatrix} y_0 \\ y_2 \\ \vdots \\ y_{N-4} \\ y_{N-2} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & \cdots & h_{N-4} & h_{N-3} & h_{N-2} & h_{N-1} \\ h_{N-2} & h_{N-1} & h_0 & h_1 & \cdots & h_{N-6} & h_{N-5} & h_{N-4} & h_{N-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ h_4 & h_5 & h_6 & h_7 & \cdots & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & h_4 & h_5 & \cdots & h_{N-2} & h_{N-1} & h_0 & h_1 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-4} \\ x_{N-3} \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \tag{4.26}$$

$$= \begin{bmatrix} h_0 & h_2 & \cdots & h_{N-4} & h_{N-2} \\ h_{N-2} & h_0 & \cdots & h_{N-6} & h_{N-4} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_4 & h_6 & \cdots & h_0 & h_2 \\ h_2 & h_4 & \cdots & h_{N-2} & h_0 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_2 \\ \vdots \\ x_{N-4} \\ x_{N-2} \end{bmatrix} + \begin{bmatrix} h_1 & h_3 & \cdots & h_{N-3} & h_{N-1} \\ h_{N-1} & h_1 & \cdots & h_{N-5} & h_{N-3} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_5 & h_7 & \cdots & h_1 & h_3 \\ h_3 & h_5 & \cdots & h_{N-1} & h_1 \end{bmatrix}^* \begin{bmatrix} x_1 \\ x_3 \\ \vdots \\ x_{N-3} \\ x_{N-1} \end{bmatrix}$$

$$\mathbf{y}_0 = \mathbf{H}_0^* \, \mathbf{x}_0 + \mathbf{H}_1^* \, \mathbf{x}_1,$$

and

$$\begin{bmatrix} y_1 \\ y_3 \\ \vdots \\ y_{N-3} \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} h_{N-1} & h_0 & h_1 & h_2 & \cdots & h_{N-5} & h_{N-4} & h_{N-3} & h_{N-2} \\ h_{N-3} & h_{N-2} & h_{N-1} & h_0 & \cdots & h_{N-7} & h_{N-6} & h_{N-5} & h_{N-4} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ h_3 & h_4 & h_5 & h_6 & \cdots & h_{N-1} & h_0 & h_1 & h_2 \\ h_1 & h_2 & h_3 & h_4 & \cdots & h_{N-3} & h_{N-2} & h_{N-1} & h_0 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-4} \\ x_{N-3} \\ x_{N-2} \\ x_{N-1} \end{bmatrix}$$

$$= \begin{bmatrix} h_{N-1} & h_1 & \cdots & h_{N-5} & h_{N-3} \\ h_{N-3} & h_{N-1} & \cdots & h_{N-7} & h_{N-5} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_3 & h_5 & \cdots & h_{N-1} & h_1 \\ h_1 & h_3 & \cdots & h_{N-3} & h_{N-1} \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_2 \\ \vdots \\ x_{N-4} \\ x_{N-2} \end{bmatrix} + \begin{bmatrix} h_0 & h_2 & \cdots & h_{N-4} & h_{N-2} \\ h_{N-2} & h_0 & \cdots & h_{N-6} & h_{N-4} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_4 & h_6 & \cdots & h_0 & h_2 \\ h_2 & h_4 & \cdots & h_{N-2} & h_0 \end{bmatrix}^* \begin{bmatrix} x_1 \\ x_3 \\ \vdots \\ x_{N-3} \\ x_{N-1} \end{bmatrix} \tag{4.27}$$

$$= \begin{bmatrix} h_1 & h_3 & \cdots & h_{N-3} & h_{N-1} \\ h_{N-1} & h_1 & \cdots & h_{N-5} & h_{N-3} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_5 & h_7 & \cdots & h_1 & h_3 \\ h_3 & h_5 & \cdots & h_{N-1} & h_1 \end{bmatrix}^* \begin{bmatrix} x_2 \\ x_4 \\ \vdots \\ x_{N-2} \\ x_0 \end{bmatrix} + \begin{bmatrix} h_0 & h_2 & \cdots & h_{N-4} & h_{N-2} \\ h_{N-2} & h_0 & \cdots & h_{N-6} & h_{N-4} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_4 & h_6 & \cdots & h_0 & h_2 \\ h_2 & h_4 & \cdots & h_{N-2} & h_0 \end{bmatrix}^* \begin{bmatrix} x_1 \\ x_3 \\ \vdots \\ x_{N-3} \\ x_{N-1} \end{bmatrix}$$

$$\mathbf{y}_1 = \mathbf{H}_1^* \, \mathbf{P} \, \mathbf{x}_0 + \mathbf{H}_0^* \, \mathbf{x}_1,$$

where $\mathbf{H}_0$ and $\mathbf{H}_1$ are circulant matrices corresponding to the even and odd samples of h_n, $\mathbf{x}_0$ and $\mathbf{x}_1$ are vectors corresponding to the even and odd samples of x_n, and $\mathbf{P}$ is the following

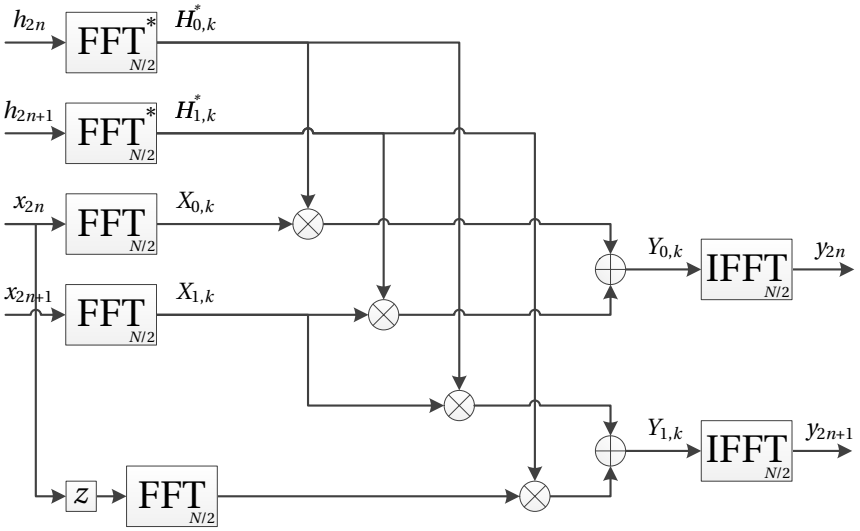


Figure 4.10: Computation of a circular correlation of N points using $N/2$-point FFTs (algorithm obtained using matrices).

permutation matrix,

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}. \tag{4.28}$$

The permutation matrix implies a circular shift of one sample of the signal. Since the matrices $\mathbf{H}_0$ and $\mathbf{H}_1$ are circulant, we can use the FFT to implement the matrix-vector products, which gives Fig. 4.10. However, since a circular shift of one sample in the time domain of a sequence of N samples corresponds to a multiplication by $e^{\frac{j2\pi k}{N}}$ in the frequency domain (Oppenheim and Schafer [2009], pp. 564-567), one FFT can be removed and then we obtain Fig. 4.11.

Z transform view

The circular correlation can also be expressed as

$$Y(z) = H^*(1/z^*)X(z) \mod (z^{-N}-1), \tag{4.29}$$

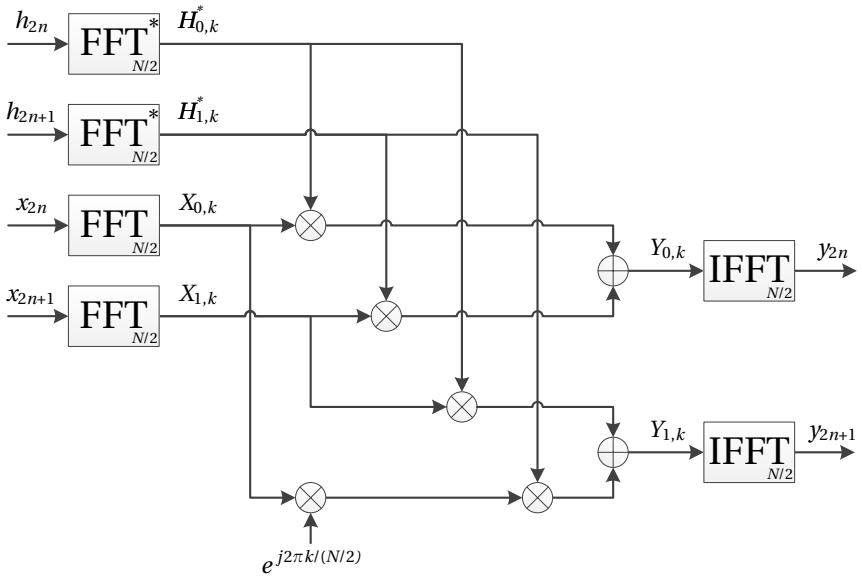


Figure 4.11: Computation of a circular correlation of N points using $N/2$-point FFTs (algorithm obtained using matrices replacing the time domain shift by a frequency domain multiplication).

with $Y(z)$, $H(z)$ and $X(z)$, the z transforms of y_n, h_n and x_n, respectively, and N is the length of the sequences. By separating the even and odd samples of the sequence x_n, we can write

$$X(z) = \sum_{n=0}^{N/2-1} x_{2n} z^{-2n} + \sum_{n=0}^{N/2-1} x_{2n+1} z^{-(2n+1)}. \tag{4.30}$$

Defining

$$X_0(z) = \sum_{n=0}^{N/2-1} x_{2n} z^{-n}, \qquad X_1(z) = \sum_{n=0}^{N/2-1} x_{2n+1} z^{-n}, \tag{4.31}$$

which are the z transforms of the sequences from the even and odd samples of x_n, we can write

$$X(z) = X_0\left(z^2\right) + z^{-1} X_1\left(z^2\right). \tag{4.32}$$

This is the polyphase representation (Vaidyanathan [1993] pp. 120–122). Note that $X_0\left(z^2\right)$ and $X_1\left(z^2\right)$ contain only even powers of z. In the same way, we have

$$Y(z) = Y_0\left(z^2\right) + z^{-1} Y_1\left(z^2\right), \tag{4.33}$$

with

$$Y_0(z) = \sum_{n=0}^{N/2-1} y_{2n} z^{-n}, \qquad Y_1(z) = \sum_{n=0}^{N/2-1} y_{2n+1} z^{-n}, \tag{4.34}$$

and

$$\begin{aligned} H^*(1/z^*) &= \sum_{n=0}^{N-1} h_n^* z^n \\ &= \sum_{n=0}^{N/2-1} h_{2n}^* z^{2n} + \sum_{n=0}^{N/2-1} h_{2n+1}^* z^{2n+1} \\ &= H_0^*\left((1/z^*)^2\right) + z\, H_1^*\left((1/z^*)^2\right), \end{aligned} \tag{4.35}$$

with

$$H_0(z) = \sum_{n=0}^{N/2-1} h_{2n} z^{-n}, \qquad H_1(z) = \sum_{n=0}^{N/2-1} h_{2n+1} z^{-n}. \tag{4.36}$$

Applying this to Eq. (4.29), we have

$$\begin{aligned} Y_0\left(z^2\right) + z^{-1} Y_1\left(z^2\right) &= \left(H_0^*\left((1/z^*)^2\right) + z\, H_1^*\left((1/z^*)^2\right)\right)\left(X_0\left(z^2\right) + z^{-1} X_1\left(z^2\right)\right) \mod \left(z^{-N}-1\right) \\ &= \left(H_0^*\left((1/z^*)^2\right) X_0\left(z^2\right) + H_1^*\left((1/z^*)^2\right) X_1\left(z^2\right)\right) \mod \left(z^{-N}-1\right) \\ &\quad + z^{-1}\left(H_0^*\left((1/z^*)^2\right) X_1\left(z^2\right) + z^2 H_1^*\left((1/z^*)^2\right) X_0\left(z^2\right)\right) \mod \left(z^{-N}-1\right) \end{aligned} \tag{4.37}$$

Inside both parenthesis, there are only even powers of z. The second parenthesis being multiplied by z^{-1}, this term contains only odd powers of z. If N is even, after the modulo operation, the parity of the powers of z are unchanged. Consequently, we have

$$Y_0\left(z^2\right) = \left(H_0^*\left((1/z^*)^2\right) X_0\left(z^2\right) + H_1^*\left((1/z^*)^2\right) X_1\left(z^2\right)\right) \mod \left(z^{-N}-1\right) \tag{4.38}$$

and

$$Y_1\left(z^2\right) = \left(H_0^*\left((1/z^*)^2\right) X_1\left(z^2\right) + z^2 H_1^*\left((1/z^*)^2\right) X_0\left(z^2\right)\right) \mod \left(z^{-N}-1\right). \tag{4.39}$$

Evaluating the previous equations for $z = e^{\frac{j2\pi k}{N}}$ with $k = 0, 1, \ldots, N-1$, we obtain

$$Y_{0,k} = H_{0,k}^* X_{0,k} + H_{1,k}^* X_{1,k}, \tag{4.40}$$

and

$$Y_{1,k} = H_{0,k}^* X_{1,k} + e^{\frac{j2\pi k}{N/2}} H_{1,k}^* X_{0,k}, \tag{4.41}$$

where $Y_{0,k}$ and $Y_{1,k}$ are the DFTs of y_{2n} and y_{2n+1}, $H_{0,k}$ and $H_{1,k}$ are the DFTs of h_{2n} and h_{2n+1}, and $X_{0,k}$ and $X_{1,k}$ are the DFTs of x_{2n} and x_{2n+1}. We obtain the same result as using the matrix notation, and the corresponding implementation is in Fig. 4.11.

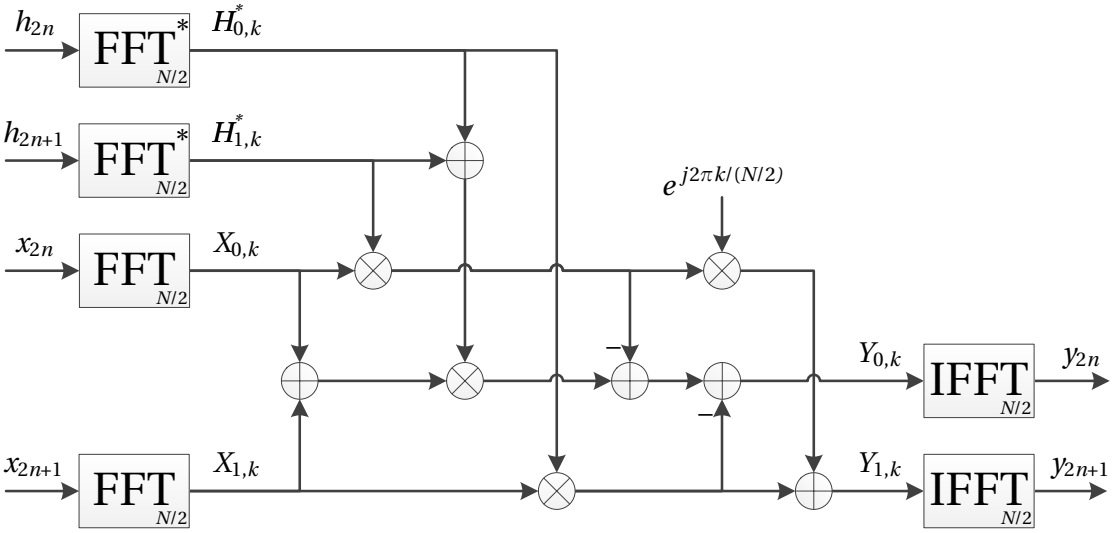


Figure 4.12: Computation of a circular correlation of N points using $N/2$-point FFTs and the minimum number of multipliers, where the inputs and the output are separated by parity.

Reduction of multipliers

The developments presented previously can be adapted to separate the sequences into 3, 4 or more sub-sequences. If each sequence is split in S sub-sequences, the number of multipliers is S^2+S-1 (S^2 for the products between the FFTs, and $S-1$ for the products with the exponentials) and the number of adders is $S(S-1)$. However, it is possible to reduce the number of multipliers. For example, noting that

$$\left(H^*_{0,k}+H^*_{1,k}\right)\left(X_{0,k}+X_{1,k}\right)=H^*_{0,k}X_{0,k}+H^*_{1,k}X_{1,k}+H^*_{0,k}X_{1,k}+H^*_{1,k}X_{0,k}, \tag{4.42}$$

Eq. (4.40) becomes

$$Y_{0,k}=\left(H^*_{0,k}+H^*_{1,k}\right)\left(X_{0,k}+X_{1,k}\right)-H^*_{0,k}X_{1,k}-H^*_{1,k}X_{0,k}. \tag{4.43}$$

Therefore, using Eqs. (4.43) and (4.41), the circular correlation can be computed using 4 multipliers and 5 adders as shown in Fig. 4.12, compared to 5 multipliers and 2 adders using Eqs. (4.40) and (4.41). These developments are based on the same principle as the fast FIR (finite impulse response) algorithms (FFA) (Mou and Duhamel [1991], Parker and Parhi [1997], Parhi [1999] Chap. 9), except that they are adapted to the circular correlation implemented with FFTs.

However, the FFAs do not always provide the minimum number of multipliers, but only a sub-optimal reduction. The minimum number of multipliers that can be obtained is $3S-2$.

Algorithm	Number of sub-sequences (S)	Number of complex multipliers	Number of complex adders
No reduction of the multipliers	2	5	2
	3	11	6
	4	19	12
Sub-optimal reduction of the multipliers	2	4	5
	3	8	13
	4	13	25
Optimal reduction of the multipliers	2	4	5
	3	7	25
	4	10	78

Table 4.4: Number of operations for the different algorithms.

Indeed, if we express the relation between the FFTs using matrices, we have

$$\begin{bmatrix} Y_{0,k} \\ Y_{1,k} \end{bmatrix} = \begin{bmatrix} H_{0,k} & H_{1,k} \\ e^{-\frac{j2\pi k}{N/2}} H_{1,k} & H_{0,k} \end{bmatrix}^* \begin{bmatrix} X_{0,k} \\ X_{1,k} \end{bmatrix}. \tag{4.44}$$

If we split the sequences in three sub-sequences, we have

$$\begin{bmatrix} Y_{0,k} \\ Y_{1,k} \\ Y_{2,k} \end{bmatrix} = \begin{bmatrix} H_{0,k} & H_{1,k} & H_{2,k} \\ e^{-\frac{j2\pi k}{N/2}} H_{2,k} & H_{0,k} & H_{1,k} \\ e^{-\frac{j2\pi k}{N/2}} H_{1,k} & e^{-\frac{j2\pi k}{N/2}} H_{2,k} & H_{0,k} \end{bmatrix}^* \begin{bmatrix} X_{0,k} \\ X_{1,k} \\ X_{2,k} \end{bmatrix}. \tag{4.45}$$

It can be seen that the matrix in Eq. (4.45) is a Toeplitz matrix (see Section A.2.3), and it is known that the minimum number of multiplications required to compute the product between a Toeplitz matrix of size $S \times S$ and a vector of length S is $2S-1$ (Lafon [1974]). Since there are also $S-1$ multipliers needed for the multiplication with the complex exponentials, the total minimum number of multipliers is $3S-2$. However, when the number of multipliers is minimum, the number of adders increases very fast, as shown in Table 4.4 for the some small values of S (Leclère et al. [2012]).

Note that when splitting the signals in two (i.e. $S = 2$), it could be possible to have only two multipliers for the product between the FFTs, but this requires that the length of the sequences be the product of two coprime numbers (Garg [1998] pp. 313–316). Therefore, this cannot be applied when the length of the sequences is a power of two.

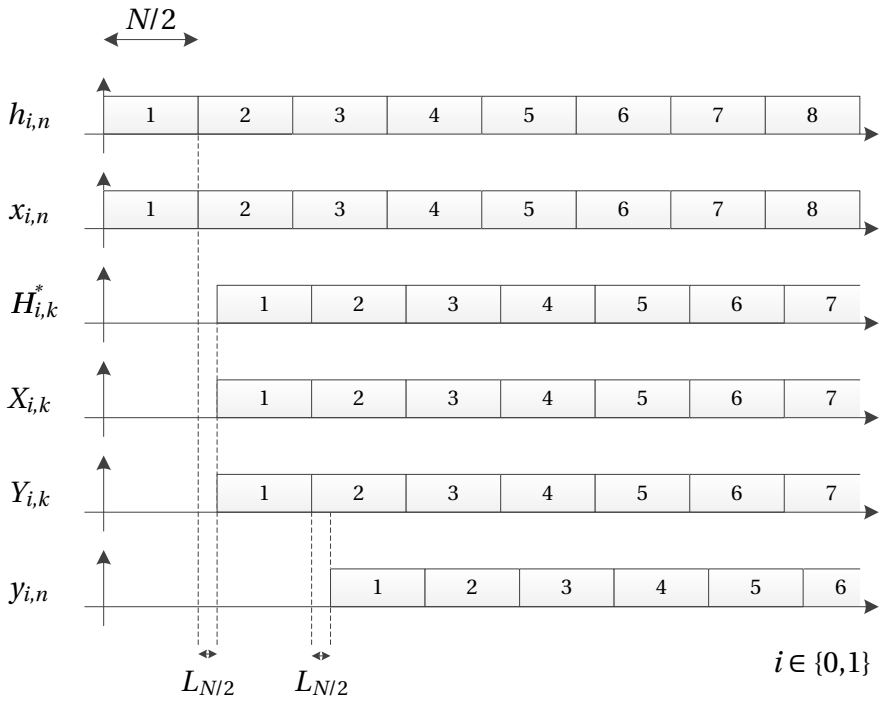


Figure 4.13: Timing diagram corresponding to Fig. 4.12 using Altera FFTs. The number in the boxes identifies the sequences.

4.3.5 Application to reduce the processing time

Implementations of Figs. 4.8 and 4.9 require 5 multipliers and 6 adders, and the implementation of Fig. 4.12 requires 4 multipliers and 5 adders. Since this last implementation uses less DSP resources, we consider it for the evaluation of the resources in this section.

The timing diagram corresponding to Fig. 4.12 using the Altera FFT is depicted Fig. 4.13. It can be seen that the Pth correlation result is fully available after $\frac{N}{2} + L_{N/2} + \frac{N}{2} + L_{N/2} + P\frac{N}{2} = (P+2)\frac{N}{2} + 2L_{N/2}$ clock cycles. Therefore, compared to the traditional implementation of the circular correlation (Fig. 4.6), the processing time is approximately halved (see Fig. 4.7).

For the evaluation of the resources, we consider $N = 2048$. As previously, the resources for the FFT and the NCO are estimated with the Altera MegaWizard Plug-In Manager (the parameters for the NCO are keep to the default ones), and the models defined in Appendix C are used for the other elements (multiplier and adder). The summary of the resources is given Table 4.5. It can be seen that the resources are higher for the implementation of Fig. 4.12 than Fig. 4.6. However, we have seen just before that the processing time for Fig. 4.12 was divided by a factor two. Since the resources are increased by a factor less than two, the implementation of Fig. 4.12 is more efficient than the implementation of Fig. 4.6.

Implementation	Function	Logic usage (ALUT)	Memory usage (M9K)	Multipliers usage (DSP element)
Fig. 4.12	6 1024-point FFTs	6 × 5248	6 × 19	6 × 12
	NCO	180	2	4
	4 Multipliers	0	0	4 × 4
	5 Adders	5 × 36	0	0
	Total	31 848	116	92
Fig. 4.6	3 2048-point FFTs	3 × 6906	3 × 38	3 × 24
	1 Multiplier	0	0	4
	Total	20 718	114	76
Ratio		1.54	1.02	1.21

Table 4.5: Comparison of the resources for Fig. 4.12 and Fig. 4.6 using the Altera FFT with $N = 2048$.

4.3.6 Application to reduce the resources

In the previous section, the proposed implementation was more efficient, but the resources were increased. In this section, we adapt it to use only three FFTs instead of six, at the expense of an additional memory. In this case, the implementations based on the CRT (Fig. 4.9) is more interesting because it requires less memory. Indeed, each IFFT output is obtained using only two FFTs results, whereas in Fig. 4.11 the four FFTs results are needed to compute the each IFFT output. The implementation is given Fig. 4.14, and the corresponding timing diagram is given Fig. 4.15.

Here is a summary of how works Fig. 4.14 :

1. Compute the FFTs of $h_{0,n}$ and $x_{0,n}$.
2. Compute the product of the FFTs.
3. Compute the IFFT to obtain $y_{0,n}$.
4. Store $y_{0,n}$ in a memory.
5. Repeat the first three steps for $h_{1,n}$ and $x_{1,n}$ (which are before multiplied by the complex exponential) to obtain $y_{1,n}$ (which involved also a product with a the complex exponential).
6. When $y_{1,n}$ is available, read $y_{0,n}$ from the memory and compute their sum and difference.

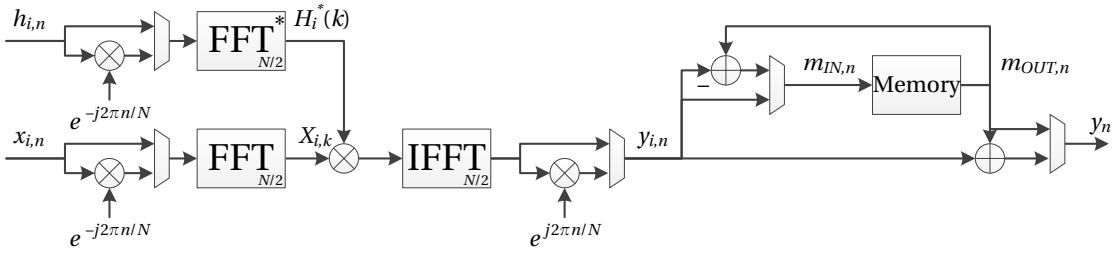


Figure 4.14: Computation of the correlation using three $N/2$-point FFTs and a memory.

Implementation	Function	Logic usage (ALUT)	Memory usage (M9K)	Multipliers usage (DSP element)
Fig. 4.14	3 1024-point FFTs	3 × 5248	3 × 19	3 × 12
	1 NCO	180	2	4
	4 Multipliers	0	0	4 × 4
	4 Adders	4 × 36	0	0
	1 Memory	22	4	0
	Total	16 090	63	56
Fig. 4.6	3 2048-point FFTs	3 × 6906	3 × 38	3 × 24
	1 Multiplier	0	0	4
	Total	20 718	114	76
Ratio		0.78	0.55	0.74

Table 4.6: Comparison of the resources for Fig. 4.14 and Fig. 4.6 using the Altera FFT with $N = 2048$.

7. Output their sum, which corresponds to y_n, and store in the memory their difference, which corresponds to $y_{n+N/2}$.

8. Read the memory to output $y_{n+N/2}$.

In this way the correlation result is provided in the exact same order as with Fig. 4.6. In this case, the Pth correlation result is fully available after $\frac{N}{2}+L_{N/2}+\frac{N}{2}++\frac{N}{2}+PN=(P+\frac{3}{2})N+2L_{N/2}$ clock cycles, which is about $N/2$ cycles less than for Fig. 4.6 because of the lower latency.

The corresponding resources are given Table 4.6, still with $N = 2048$. For the memory, we need to store twice (because the signal is complex) 1024 × 18 bits, which requires 4 M9K memories, and we consider few logic for the addressing. It can be seen that the resources are reduced, by about 22 % for the logic, 45 % for the memory, and 26 % for the DSP elements, which is not negligible.

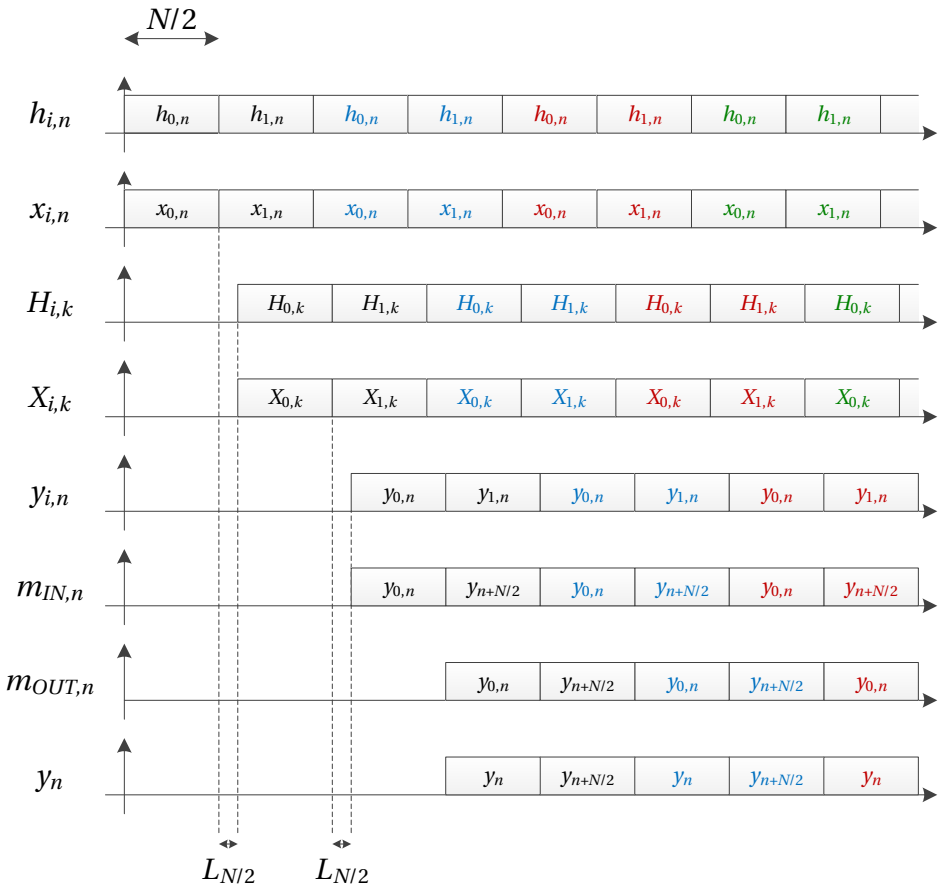


Figure 4.15: Timing diagram corresponding to Fig. 4.14 using Altera FFTs. The colors inside the boxes identify the sequences.

4.4 Summary

In this chapter, we have shown different ways to compute an FFT in Altera FPGAs, with lower resources and the same processing time than the direct implementation of one Altera FFT. Then, it was shown also that it is possible to reduce the resources for an FFT-based circular correlation compared to the direct implementation that uses three FFTs.

It has been shown that all the resources can be reduced, i.e the logic, the memory and the DSP blocks, but it is mainly the memory that is reduced (33 % for the FFT and 45 % for the correlation with sequences of 2048 samples). If we extrapolate to other transform lengths, the results regarding the logic and the memory would be about the same, however for the DSP elements the results would be not as good because the number of DSP elements does not increase when we increase the transform length above 2048.

The algorithms presented do not make any assumptions about the input or output signals, therefore they can be applied not only for GNSS but for any other systems computing FFTs, convolutions, or correlations. Besides, in addition to the implementations proposed in this

chapter, it is possible to use the method for computing the FFT of two real sequences using the complex Altera FFT (see Appendix B), which is useful in GNSS since the local code is real. Some other examples are also given in (Leclère et al. [2012]).

5 Acquisition of GNSS signals in presence of transition

In this chapter, we discuss the problem of the parallel code search acquisition when one primary code period is used, and when the received signal contains bit transitions, that can be due to a secondary code or data modulation.

After discussing the proposition of two algorithms, we consider their application to different GNSS signals, such as GPS L5 and Galileo E5a, E5b and E1 OS.

This work has been published in (Leclère et al. [2013a]) and (Leclère et al. [2014]), except for the second proposed algorithm that has been discovered later.

5.1 Introduction

As indicated in Chapter 1, the recently introduced GPS and Galileo signals bring new features compared to the initial civilian GPS L1 C/A signal, such as a higher power, longer codes for a better cross-correlation between satellites signals, pilot channels that do not carry data to facilitate long integrations and improve the sensitivity threshold, and secondary codes that are usually short PRN codes to make the data synchronization easier.

The presence of a secondary code brings advantages and additional performance, but also makes the acquisition more difficult (Shivaramaiah et al. [2008], Borio [2011]). Exploiting the secondary code adds a third dimension to the acquisition search, besides the Doppler frequency and primary code dimensions (or it increases the code dimension, depending on the point of view), and implies the use of long coherent integration times, which impacts also the search in the Doppler frequency dimension.

To acquire very weak signals, such as in indoor or urban environments (Ayaz et al. [2010]), the coherent integration time should be as long as possible (Pany et al. [2009]). Therefore, with modern signals this means that the secondary code must be synchronized too. However, it is still possible to perform the acquisition using only the primary code, with the possibility to synchronize with the secondary code afterwards if the sensitivity is not the priority. This

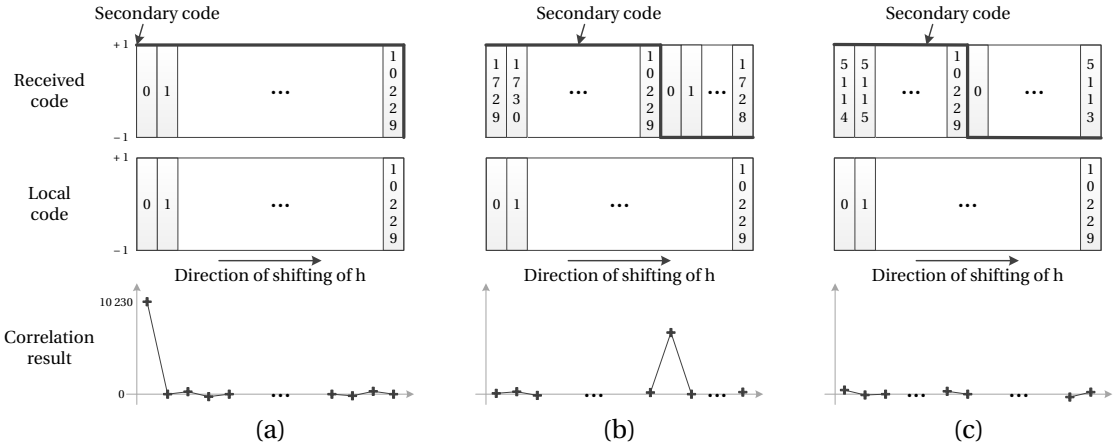


Figure 5.1: Illustration of the problem due to the secondary code transition. The values inside boxes indicate the chip number. (a) By chance, the incoming primary code starts with the first chip, the correlation at the correct alignment is maximum. (b) The incoming primary code does not start at the first chip (usual case), the correlation at the correct alignment is reduced. (c) In the worst case, the incoming primary code starts at the middle of a period, the correlation at the correct alignment is 0.

chapter focuses on this case. This case may happen if we want to process a modern signal to get a better accuracy (the L5, E5a and E5b signals have a chipping rate of 10.23 MHz), or to get a fast position (the E1 OS and E5b signals have a data rate of 250 bit/s). This case happens also if we want to perform a coherent integration over a secondary code period, but the limited resources prevent the computation of so large FFTs (as will be shown in Chapter 6, which also discuss the case using large FFTs).

Even if the secondary code is not exploited, the potential transitions between consecutive periods of the primary code prevent the direct use of the parallel code search acquisition. Indeed, this can result in very high losses leading to the non-detection of the signal (Borio et al. [2008b]), as illustrated in Fig. 5.1. When the local replica of the primary code is aligned with the incoming primary code, the magnitude of the correlation peak is maximum only if there is no transition (Fig. 5.1 (a)). Else, in case of transition, the correlation peak is reduced (Fig. 5.1 (b)), or even vanishes if the incoming primary code starts at the middle of the period (Fig. 5.1 (c)). In fact, the problem is worse than a simple non-detection, because the correlation peak may be detected at an incorrect frequency (Lo Presti et al. [2009]), which means that the receiver will start tracking the signal incorrectly and waste time before performing again an acquisition.

There were different propositions to overcome this problem. For example, people proposed to perform a kind of average in the Doppler search space (Lo Presti et al. [2009]); to have two steps to find first the code delay, and then the Doppler frequency (Sun and Lo Presti [2012]); or to generate two local codes, one without a transition and one with a transition, requiring to compute five FFTs instead of the usual three (Jeon et al. [2012]).

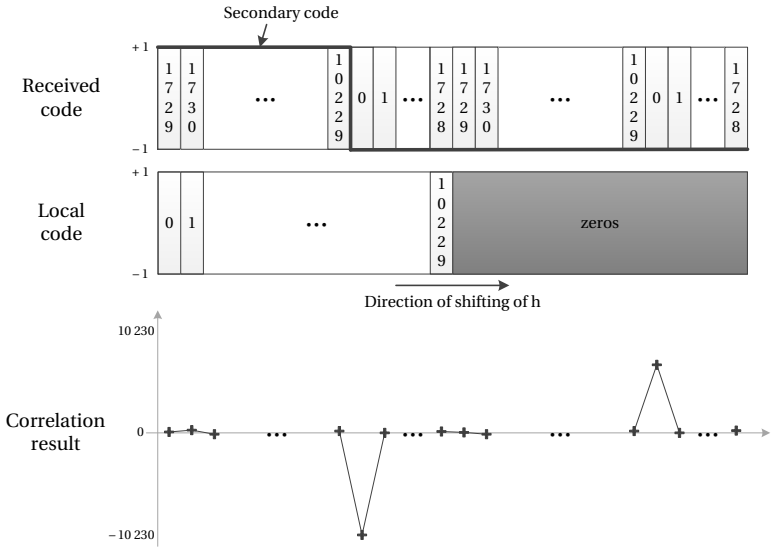


Figure 5.2: Straightforward solution to the secondary code transition problem. The values inside boxes indicate the chip number, and the gray area contains only zeros. The magnitude of the first peak is always maximum, whereas the second peak can be reduced due to transition.

A straightforward solution exists and consists in using two consecutive periods of the incoming primary code and one period of the primary code padded with zeros for the local code to perform the correlation (Yang et al. [2004], Borio et al. [2008b]). In this way, there is always one period of the incoming code free of transition, and thus a maximum correlation peak, as illustrated in Fig. 5.2 (the sign of the peak is not important, only its magnitude is). It can be seen that there is a second peak, similar to the one of Fig. 5.1 (b). Indeed, since there are two periods of the incoming primary code, the local code is correctly aligned twice. However, the magnitude of the first peak is always maximum, whereas the second peak can be reduced or vanish due to the transition. Since the first peak always occurs in the first half of the correlation, the second half of the correlation is discarded.

However, this solution increases the complexity and is not so efficient since half of the points calculated are unused. To tackle this problem, we propose two new algorithms that reduce the complexity by transforming the initial correlation into two sub-correlations. These algorithms are not approximations, but other ways to compute the samples of interest. Therefore, there is no degradation of the sensitivity. The concept has some similarities with classical divide-and-conquer approaches such as the overlap-and-add or overlap-and-save methods (Proakis and Manolakis [2006]), although different.

The straightforward and proposed algorithms are compared first for the acquisition of the GPS L5, Galileo E5a and E5b signals, which are equivalent for this problem since their primary codes have the same length and the modulation is the same. Then, the algorithms are compared for the acquisition of the Galileo E1 OS signal, considering first the BOC(1,1) modulation and then the BPSK modulation. The comparison of the algorithms is done for a software

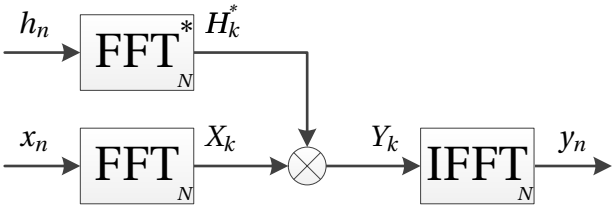


Figure 5.3: Implementation of the straightforward algorithm using FFTs. The second half of h_n contain only zeros, and the second half of y_n is not used.

implementation using Matlab, and for a hardware implementation on an FPGA.

5.2 Proposed Algorithms

The idea of the proposed algorithms is to transform the initial correlation into two sub-correlations of smaller size. The proposed algorithm exploits two facts for this : 1) Half of the points of one of the signals are zero; and 2) Half of the points of the correlation output are discarded. A third fact will be exploited for the reduction of the complexity, the usage of additional zero-padding when it is needed to reach a specific sequence length.

Let's first define the operation we want to perform. Using the matrix notation, the circular correlation y_n of two sequences h_n and x_n of N points can be expressed as

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{N}{2}-1} \\ y_{\frac{N}{2}} \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{2}-1} & h_{\frac{N}{2}} & \cdots & h_{N-2} & h_{N-1} \\ h_{N-1} & h_0 & \cdots & h_{\frac{N}{2}-2} & h_{\frac{N}{2}-1} & \cdots & h_{N-3} & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{\frac{N}{2}+1} & h_{\frac{N}{2}+2} & \cdots & h_0 & h_1 & \cdots & h_{\frac{N}{2}-1} & h_{\frac{N}{2}} \\ h_{\frac{N}{2}} & h_{\frac{N}{2}+1} & \cdots & h_{N-1} & h_0 & \cdots & h_{\frac{N}{2}-2} & h_{\frac{N}{2}-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h_2 & h_3 & \cdots & h_{\frac{N}{2}+1} & h_{\frac{N}{2}+2} & \cdots & h_0 & h_1 \\ h_1 & h_2 & \cdots & h_{\frac{N}{2}} & h_{\frac{N}{2}+1} & \cdots & h_{N-1} & h_0 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{2}-1} \\ x_{\frac{N}{2}} \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \tag{5.1}$$

$$\mathbf{y} = \mathbf{H}^* \, \mathbf{x},$$

where h_n corresponds to the local code, and x_n to the incoming signal after the carrier removal. The corresponding implementation using FFTs is given Fig. 5.3.

Using the two conditions described previously, namely that the second half of h_n contains

only zeros and that the second half of y_n is not used, what we want to compute is

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{N}{2}-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{2}-1} & 0 & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{2}-2} & h_{\frac{N}{2}-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0 & h_1 & \cdots & h_{\frac{N}{2}-1} \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{2}-1} \\ x_{\frac{N}{2}} \\ \vdots \\ x_{N-2} \end{bmatrix} \tag{5.2}$$

$$\mathbf{y}_T = \mathbf{H}_T^* \, \mathbf{x}_T.$$

It can be noted that the sample x_{N-1} is not required to compute the first half of y_n, since it is multiplied by the samples from $h_{\frac{N}{2}}$ to h_{N-1}, which are zero here. That is why we remove it in Eq. (5.2). At this stage, $\mathbf{y}_T$ (T stands for truncated) is a vector of $N/2$ points, $\mathbf{H}_T$ is a matrix of $N/2 \times N-1$, and $\mathbf{x}_T$ is a vector of $N-1$ points. We start from this equation to describe the two proposed algorithms.

5.2.1 Algorithm 1

Step 1 : Separation of the matrix

The first step consists in separating the matrix $\mathbf{H}_T$ in two matrices, $\mathbf{H}_{T0}$ and $\mathbf{H}_{T1}$, where $\mathbf{H}_{T0}$ is $\mathbf{H}_T$ with $h_n = 0$ for $\frac{N}{4} \leq n \leq \frac{N}{2} - 1$, and $\mathbf{H}_{T1}$ is $\mathbf{H}_T$ with $h_n = 0$ for $0 \leq n \leq \frac{N}{4} - 1$, such that $\mathbf{H}_T$ is the sum of $\mathbf{H}_{T0}$ and $\mathbf{H}_{T1}$. Of course, this requires that N is divisible by 4, which will be assumed throughout this chapter. We thus obtain

$$\begin{aligned} \mathbf{y}_T &= \mathbf{H}_T^* \, \mathbf{x}_T \\ &= (\mathbf{H}_{T0}^* + \mathbf{H}_{T1}^*) \, \mathbf{x}_T \\ \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{N}{2}-1} \end{bmatrix} &= \left(\begin{bmatrix} h_0 & h_1 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & h_0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0 & h_1 & \cdots & 0 \end{bmatrix}^* \right. \\ &\quad \left. + \begin{bmatrix} 0 & 0 & \cdots & h_{\frac{N}{2}-1} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & h_{\frac{N}{2}-2} & h_{\frac{N}{2}-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & h_{\frac{N}{2}-1} \end{bmatrix}^* \right) \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{2}-1} \\ x_{\frac{N}{2}} \\ \vdots \\ x_{N-2} \end{bmatrix}. \end{aligned} \tag{5.3}$$

Step 2 : Removing of columns of zeros

The last $\frac{N}{4}$ columns of $\mathbf{H}_{T0}$ and the first $\frac{N}{4}$ columns of $\mathbf{H}_{T1}$ contain only zeros. Thus, we can remove these columns to obtain $\mathbf{H}_{TT0}$ and $\mathbf{H}_{TT1}$ and then

$$\begin{aligned}\mathbf{y}_T &= (\mathbf{H}^*_{T0} + \mathbf{H}^*_{T1})\,\mathbf{x}_T \\ &= \mathbf{H}^*_{TT0}\,\mathbf{x_0} + \mathbf{H}^*_{TT1}\,\mathbf{x_1}\end{aligned}$$

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{N}{2}-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & 0 & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & h_{\frac{N}{4}-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & h_{\frac{N}{4}-1} \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{4}-1} \\ x_{\frac{N}{4}} \\ \vdots \\ x_{\frac{3N}{4}-2} \end{bmatrix} + \begin{bmatrix} h_{\frac{N}{4}} & h_{\frac{N}{4}+1} & \cdots & h_{\frac{N}{2}-1} & 0 & \cdots & 0 \\ 0 & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-2} & h_{\frac{N}{2}-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & h_{\frac{N}{2}-1} \end{bmatrix}^* \begin{bmatrix} x_{\frac{N}{4}} \\ x_{\frac{N}{4}+1} \\ \vdots \\ x_{\frac{N}{2}-1} \\ x_{\frac{N}{2}} \\ \vdots \\ x_{N-2} \end{bmatrix}. \tag{5.4}$$

At this stage, $\mathbf{H}_{TT0}$ and $\mathbf{H}_{TT1}$ are matrices of $\frac{N}{2} \times \frac{3N}{4} - 1$, and $\mathbf{x_0}$ and $\mathbf{x_1}$ are vectors of $\frac{3N}{4} - 1$ points.

Step 3 : Making the matrices circulant

The matrices $\mathbf{H}_{TT0}$ and $\mathbf{H}_{TT1}$ have a circular pattern. It is thus possible to include them into circulant matrices by adding $\frac{N}{4} - 1$ rows, as shown in Eqs. (5.5) and (5.6).

$$\mathbf{H}_0 = \begin{bmatrix} \mathbf{H}_{TT0} \\ \mathbf{H}_{D0} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & 0 & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & h_{\frac{N}{4}-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & h_{\frac{N}{4}-1} \\ h_{\frac{N}{4}-1} & 0 & \ldots & 0 & 0 & \ldots & h_{\frac{N}{4}-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & 0 & 0 & \cdots & h_0 \end{bmatrix} \tag{5.5}$$

$$\mathbf{H_1} = \begin{bmatrix} \mathbf{H}_{TT1} \\ \mathbf{H}_{D1} \end{bmatrix} = \begin{bmatrix} h_{\frac{N}{4}} & h_{\frac{N}{4}+1} & \cdots & h_{\frac{N}{2}-1} & 0 & \cdots & 0 \\ 0 & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-2} & h_{\frac{N}{2}-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & h_{\frac{N}{2}-1} \\ h_{\frac{N}{2}-1} & 0 & \dots & 0 & 0 & \dots & h_{\frac{N}{2}-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ h_{\frac{N}{4}+1} & h_{\frac{N}{4}+2} & \cdots & 0 & 0 & \cdots & h_{\frac{N}{4}} \end{bmatrix} \tag{5.6}$$

Eq. (5.4) can then be modified to obtain

$$\begin{bmatrix} \mathbf{y}_T \\ \mathbf{y}_D \end{bmatrix} = \begin{bmatrix} \mathbf{H}^*_{TT0} \\ \mathbf{H}^*_{D0} \end{bmatrix} \mathbf{x}_0 + \begin{bmatrix} \mathbf{H}^*_{TT1} \\ \mathbf{H}^*_{D1} \end{bmatrix} \mathbf{x}_1$$

$$\mathbf{y}_M = \mathbf{H}^*_0\, \mathbf{x}_0 + \mathbf{H}^*_1\, \mathbf{x}_1$$

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{N}{2}-1} \\ y_{D,0} \\ \vdots \\ y_{D,\frac{N}{4}-2} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & 0 & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & h_{\frac{N}{4}-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & h_{\frac{N}{4}-1} \\ h_{\frac{N}{4}-1} & 0 & \dots & 0 & 0 & \dots & h_{\frac{N}{4}-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & 0 & 0 & \cdots & h_0 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{4}-1} \\ x_{\frac{N}{4}} \\ \vdots \\ x_{\frac{3N}{4}-2} \end{bmatrix} + \begin{bmatrix} h_{\frac{N}{4}} & h_{\frac{N}{4}+1} & \cdots & h_{\frac{N}{2}-1} & 0 & \cdots & 0 \\ 0 & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-2} & h_{\frac{N}{2}-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & h_{\frac{N}{2}-1} \\ h_{\frac{N}{2}-1} & 0 & \dots & 0 & 0 & \dots & h_{\frac{N}{2}-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ h_{\frac{N}{4}+1} & h_{\frac{N}{4}+2} & \cdots & 0 & 0 & \cdots & h_{\frac{N}{4}} \end{bmatrix}^* \begin{bmatrix} x_{\frac{N}{4}} \\ x_{\frac{N}{4}+1} \\ \vdots \\ x_{\frac{N}{2}-1} \\ x_{\frac{N}{2}} \\ \vdots \\ x_{N-2} \end{bmatrix}. \tag{5.7}$$

At this stage, $\mathbf{y}_M$ (M stands for modified) is a vector of $\frac{3N}{4}-1$ points, $\mathbf{H}_0$ and $\mathbf{H}_1$ are circulant matrices of $\frac{3N}{4}-1 \times \frac{3N}{4}-1$, and $\mathbf{x}_0$ and $\mathbf{x}_1$ are vectors of $\frac{3N}{4}-1$ points. The vector $\mathbf{y}_M$ is composed of the initial vector $\mathbf{y}_T$ of $\frac{N}{2}$ points, and of the vector $\mathbf{y}_D$ (D stands for discarded) of $\frac{N}{4}-1$ points. Therefore, the first $\frac{N}{2}$ points of the result are identical to those of the initial correlation, while the other points are different, which is not important since they are discarded. This means that this algorithm discards only about one third of the points calculated, instead of discarding about half. However, the matrices contains about two third of zeros instead of about half.

The corresponding implementation using FFTs is given in Fig. 5.4, with

$$\begin{aligned}
\mathbf{h}_0 &= \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & \overbrace{0 \ \cdots \ 0}^{N/2-1} \end{bmatrix} \\
\mathbf{h}_1 &= \begin{bmatrix} h_{\frac{N}{4}} & h_{\frac{N}{4}+1} & \cdots & h_{\frac{N}{2}-1} & 0 \ \cdots \ 0 \end{bmatrix} \\
\mathbf{x}_0 &= \begin{bmatrix} x_0 & x_1 & \cdots & x_{\frac{3N}{4}-2} \end{bmatrix} \\
\mathbf{x}_1 &= \begin{bmatrix} x_{\frac{N}{4}} & x_{\frac{N}{4}+1} & \cdots & x_{N-2} \end{bmatrix}.
\end{aligned} \tag{5.8}$$

If needed, the sequences can be zero-padded to reach a specific length, without impacting the desired samples.

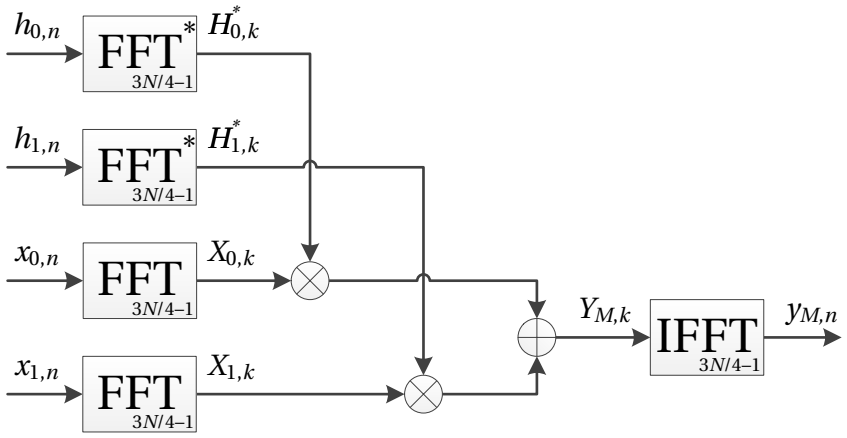


Figure 5.4: Implementation of the first proposed algorithm using FFTs. The last two thirds of $h_{0,n}$ and $h_{1,n}$ contain only zeros, and the last third of $y_{M,n}$ is not used.

5.2.2 Algorithm 2

Step 1 : Separation of the output

The first step consists in separating the output vector $\mathbf{y}_T$ in two, to obtain

$$\mathbf{y}_{T0} = \mathbf{H}^*_{T0}\,\mathbf{x}_T$$

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{N}{4}-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-1} & \cdots & 0 & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & \cdots & h_{\frac{N}{2}-2} & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0 & \cdots & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-1} & \cdots & 0 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{4}-1} \\ \vdots \\ x_{\frac{N}{2}-1} \\ \vdots \\ x_{\frac{3N}{4}-2} \\ \vdots \\ x_{N-2} \end{bmatrix}, \tag{5.9}$$

and

$$\mathbf{y}_{T1} = \mathbf{H}^*_{T1}\,\mathbf{x}_T$$

$$\begin{bmatrix} y_{\frac{N}{4}} \\ y_{\frac{N}{4}+1} \\ \vdots \\ y_{\frac{N}{2}-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & h_0 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-1} & \cdots & 0 \\ 0 & 0 & 0 & \cdots & h_{\frac{N}{4}-2} & \cdots & h_{\frac{N}{2}-2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & h_0 & \cdots & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-1} \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{4}} \\ \vdots \\ x_{\frac{N}{2}-1} \\ \vdots \\ x_{\frac{3N}{4}-1} \\ \vdots \\ x_{N-2} \end{bmatrix}. \tag{5.10}$$

At this stage, $\mathbf{y}_{T0}$ and $\mathbf{y}_{T1}$ are vectors of $\frac{N}{4}$ points, $\mathbf{H}_{T0}$ and $\mathbf{H}_{T1}$ are matrices of $\frac{N}{4} \times N-1$, and $\mathbf{x}_T$ is still a vector $N-1$ points. Be careful to not confuse the matrix $\mathbf{H}_{T0}$ in Eq. (5.9) with the matrix $\mathbf{H}_{T0}$ used in Eq. (5.3) for the first algorithm, they correspond to different matrices (idem for $\mathbf{H}_{T1}$).

Step 2 : Removing of columns of zeros

The last $\frac{N}{4}$ columns of $\mathbf{H}_{T0}$ and the first $\frac{N}{4}$ columns of $\mathbf{H}_{T1}$ contain only zeros. Thus, we can remove these columns to obtain

$$\mathbf{y}_{T0} = \mathbf{H}^*_{TT}\,\mathbf{x}_0$$

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{N}{4}-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-1} & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & \cdots & h_{\frac{N}{2}-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0 & \cdots & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-1} \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{4}-1} \\ \vdots \\ x_{\frac{N}{2}-1} \\ \vdots \\ x_{\frac{3N}{4}-2} \end{bmatrix}, \tag{5.11}$$

and

$$\mathbf{y}_{T1} = \mathbf{H}^*_{TT}\,\mathbf{x}_1$$

$$\begin{bmatrix} y_{\frac{N}{4}} \\ y_{\frac{N}{4}+1} \\ \vdots \\ y_{\frac{N}{2}-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-1} & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & \cdots & h_{\frac{N}{2}-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0 & \cdots & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-1} \end{bmatrix}^* \begin{bmatrix} x_{\frac{N}{4}} \\ x_{\frac{N}{4}+1} \\ \vdots \\ x_{\frac{N}{2}-1} \\ \vdots \\ x_{\frac{3N}{4}-1} \\ \vdots \\ x_{N-2} \end{bmatrix}. \tag{5.12}$$

At this stage, $\mathbf{H}_{TT}$ is a matrix of $\frac{N}{4} \times \frac{3N}{4} - 1$, and $\mathbf{x}_0$ and $\mathbf{x}_1$ are vectors of $\frac{3N}{4} - 1$ points. Note that the vectors $\mathbf{x}_0$ and $\mathbf{x}_1$ are the same as for the first algorithm (see Eq. (5.4)).

Step 3 : Making the matrix circulant

The matrix $\mathbf{H}_{TT}$ has a circular pattern. It is thus possible to include it into a circulant matrix by adding $\frac{N}{2}-1$ rows, as shown in Eq. (5.13).

$$\mathbf{H}_C = \begin{bmatrix} \mathbf{H}_{TT} \\ \mathbf{H}_D \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-1} & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & \cdots & h_{\frac{N}{2}-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0 & \cdots & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & h_{\frac{N}{4}} & \cdots & 0 & \cdots & h_0 \end{bmatrix} \tag{5.13}$$

Eqs. (5.11) and (5.12) can then be modified to obtain

$$\begin{bmatrix} \mathbf{y}_{T0} \\ \mathbf{y}_{D0} \end{bmatrix} = \begin{bmatrix} \mathbf{H}^*_{TT} \\ \mathbf{H}^*_D \end{bmatrix} \mathbf{x}_0$$

$$\mathbf{y}_{M0} = \mathbf{H}^*_C \, \mathbf{x}_0$$

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{N}{4}-1} \\ y_{D0,0} \\ \vdots \\ y_{D0,\frac{N}{4}-2} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-1} & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & \cdots & h_{\frac{N}{2}-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0 & \cdots & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-1} \\ 0 & 0 & \cdots & 0 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & h_{\frac{N}{4}} & \cdots & 0 & \cdots & h_0 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{4}-1} \\ \vdots \\ x_{\frac{N}{2}-1} \\ \vdots \\ x_{\frac{3N}{4}-2} \end{bmatrix} \tag{5.14}$$

and

$$\begin{bmatrix} \mathbf{y}_{T1} \\ \mathbf{y}_{D1} \end{bmatrix} = \begin{bmatrix} \mathbf{H}^*_{TT} \\ \mathbf{H}^*_D \end{bmatrix} \mathbf{x}_1$$

$$\mathbf{y}_{M1} = \mathbf{H}^*_C \, \mathbf{x}_1$$

$$\begin{bmatrix} y_{\frac{N}{4}} \\ y_{\frac{N}{4}+1} \\ \vdots \\ y_{\frac{N}{2}-1} \\ y_{D1,0} \\ \vdots \\ y_{D1,\frac{N}{4}-2} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-1} & \cdots & 0 \\ 0 & h_0 & \cdots & h_{\frac{N}{4}-2} & \cdots & h_{\frac{N}{2}-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0 & \cdots & h_{\frac{N}{4}} & \cdots & h_{\frac{N}{2}-1} \\ 0 & 0 & \cdots & 0 & \cdots & h_{\frac{N}{4}-1} & \cdots & h_{\frac{N}{2}-2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & h_{\frac{N}{4}} & \cdots & 0 & \cdots & h_0 \end{bmatrix}^* \begin{bmatrix} x_{\frac{N}{4}} \\ \vdots \\ x_{\frac{3N}{4}-1} \\ \vdots \\ x_{N-2} \end{bmatrix} \tag{5.15}$$

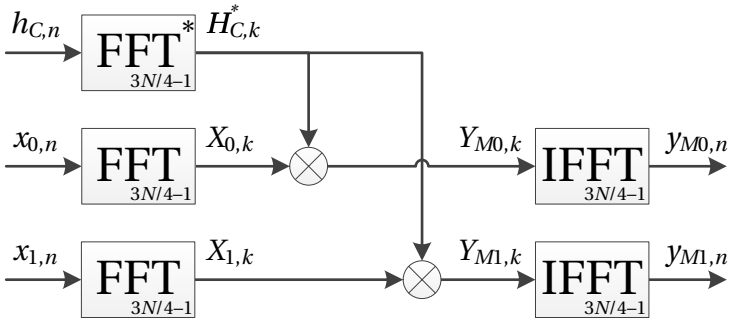


Figure 5.5: Implementation of the second proposed algorithm using FFTs. The last third of $h_{C,n}$ contains only zeros, and the last two thirds of $y_{M0,n}$ and $y_{M1,n}$ are not used.

At this stage, $\mathbf{y}_{M0}$ and $\mathbf{y}_{M1}$ are vectors of $\frac{3N}{4}-1$ points, $\mathbf{H}_C$ is a circulant matrix of $\frac{3N}{4}-1 \times \frac{3N}{4}-1$, and $\mathbf{x}_0$ and $\mathbf{x}_1$ are vectors of $\frac{3N}{4}-1$ points. The vectors $\mathbf{y}_{M0}$ and $\mathbf{y}_{M1}$ are composed of one half the initial vector $\mathbf{y}_T$, and of the vectors $\mathbf{y}_{D0}$ and $\mathbf{y}_{D1}$ of $\frac{N}{2}-1$ points, respectively. Therefore, the first $\frac{N}{2}$ points of the result are identical to those of the initial correlation, while the other points are different, which is not important since they are discarded. This means that this algorithm discards about two thirds of the points calculated, instead of discarding about half. However, the matrices now contains only about one third of zeros, compared to about half.

The corresponding implementation using FFTs is given in Fig. 5.5, with

$$\begin{aligned}
\mathbf{h}_C &= \begin{bmatrix} h_0 & h_1 & \cdots & h_{\frac{N}{2}-1} & \overbrace{0 \ \ldots \ 0}^{N/4-1} \end{bmatrix} \\
\mathbf{x}_0 &= \begin{bmatrix} x_0 & x_1 & \cdots & x_{\frac{3N}{4}-2} \end{bmatrix} \\
\mathbf{x}_1 &= \begin{bmatrix} x_{\frac{N}{4}} & x_{\frac{N}{4}+1} & \cdots & x_{N-2} \end{bmatrix}.
\end{aligned} \tag{5.16}$$

As for the first algorithm, if needed, the sequences can be zero-padded to reach a specific length, without impacting the desired samples.

5.2.3 Algorithms complexity

The proposed algorithms perform five FFTs or IFFTs of $\frac{3N}{4}-1$ points, whereas the straightforward algorithm performs three FFTs or IFFT of N points (or $N-1$).

Considering that an FFT of N points requires about $N\log(N)$ multiplications, the straightforward algorithm requires approximately $3N\log(N)+N$ multiplications, while the proposed algorithms require $5\frac{3N}{4}\log\left(\frac{3N}{4}\right)+2\frac{3N}{4}$ multiplications. The number of operations is thus greater for the proposed algorithms, by about 21 to 23 %.

This means that while the initial aim of finding a new algorithm was to decrease the complexity of the straightforward algorithm, the proposed algorithms in fact require more operations.

l	5	7	9	11	13	15	17	19
$L=2^l$	32	128	512	2048	8192	32 768	131 072	524 288
N	44	172	684	2732	10 924	43 692	174 764	699 052

Table 5.1: Correlation length (N) and radix-2 FFT length (L) of the proposed algorithms for l odd.

l	6	8	10	12	14	16	18	20
$L=2^l$	64	256	1024	4096	16 384	65 536	262 144	1 048 576
N	84	340	1364	5460	21 844	87 380	349 524	1 398 100

Table 5.2: Correlation length (N) and radix-2 FFT length (L) of the proposed algorithms for l even.

However, when there are some constraints on the FFT length (e.g. to be a power of two), the sequences are zero-padded to meet the requirements, and the proposed algorithms can be advantageous, as shown in the next sections.

5.2.4 Use with the radix-2 FFT

The proposed algorithms performs FFTs on $\frac{3N}{4}-1$ points. If the use of radix-2 FFTs is desired (by radix-2 FFT we mean that the FFT length must be a power of two), there is the following constraint,

$$\frac{3N}{4}-1=2^l \Leftrightarrow N=\frac{4}{3}(2^l+1), \tag{5.17}$$

where l is a positive integer. This equation has integer solutions only if l is odd, and the result for a range of suitable values is provided in Table 5.1. We can use zero-padding to obtain sequences of length $\frac{3N}{4}+1$, which gives the following constraint,

$$\frac{3N}{4}+1=2^l \Leftrightarrow N=\frac{4}{3}(2^l-1), \tag{5.18}$$

where l is a positive integer. This equation has integer solutions only if l is even, and the result for a range of suitable values is provided in Table 5.2.

To make the link with the GNSS signals, the FFT length for the straightforward and the proposed algorithms in function of the sampling frequency is provided in Table 5.3, considering a code of 1 ms. It can be seen that there are two possibilities. Either the FFT lengths of the algorithms are identical, or the FFT length of the proposed algorithms is half the FFT length of the straightforward algorithm. For a code of 4 ms, the length of the sequence would be four times longer, therefore Table 5.3 can be read by multiplying the sampling frequency by four.

Sampling frequency range (MHz)	span (MHz)	FFT length with the straightforward algorithm	FFT length with the proposed algorithms
1.023 – 1.024	0.001	2048	2048
1.025 – 1.366	0.341	4096	2048
1.367 – 2.048	0.681	4096	4096
2.049 – 2.730	0.681	8192	4096
2.731 – 4.096	1.365	8192	8192
4.097 – 5.462	1.365	16 384	8192
5.463 – 8.192	2.729	16 384	16 384
8.193 – 10.922	2.729	32 768	16 384
10.923 – 16.384	5.461	32 768	32 768
16.385 – 21.846	5.461	65 536	32 768
21.847 – 32.768	10.921	65 536	65 536
32.769 – 43.690	10.921	131 072	65 536

Table 5.3: Radix-2 FFT length for the straightforward and the proposed algorithms in function of the sampling frequency considering a code of 1 ms (for a code of 4 ms, multiply the actual sampling frequency by 4 to find the corresponding FFT length).

When the FFT lengths are identical, it is clear that the proposed algorithms are less efficient, since they compute more FFTs than the straightforward algorithm. For example, if we use a sampling frequency of 24 MHz, we have $N = 48\,000$, and $\frac{3N}{4} - 1 = 35\,999$. Therefore, both the straightforward and the proposed algorithms will use FFTs of 65 536 points.

When the FFT length of the proposed algorithms is half the FFT length of the straightforward algorithm, the proposed algorithms seem more efficient. Indeed, the theoretical number of multiplications is $5\frac{N}{4}\log(\frac{N}{2}) + 2\frac{N}{2}$ for the proposed algorithms, against $3\frac{N}{2}\log(N) + N$ for the straightforward algorithm, which means a reduction of about 20 %. For example, if we use a sampling frequency of 21 MHz, we have $N = 42\,000$, and $\frac{3N}{4} - 1 = 31\,499$. Therefore, the traditional algorithm still uses FFTs of 65 536 points, while the proposed algorithms now use FFTs of 32 768 points.

It can be also noted that the sampling frequency span is the same when the FFT lengths of the algorithms are equal and when the FFT length of the proposed algorithms is half the FFT length of the straightforward algorithm. Therefore, there is a 50 % chance that the proposed algorithms are efficient.

5.2.5 Algorithm to obtain *P* sub-correlations

The proposed algorithms can be generalized to more than two sub-correlations. For example, to obtain P sub-correlations with the algorithm 1, h_n and x_n must be decomposed into P components, respectively $h_{i,n}$ and $x_{i,n}$ with $i = \{0, 1, ..., P-1\}$, of $\frac{P+1}{P}\frac{N}{2}$ points, where

- $h_{i,n}$ contains $\frac{1}{P}\frac{N}{2}$ points of h_n and $\frac{N}{2}$ points of zeros,
- $x_{i,n}$ contains $\frac{P+1}{P}\frac{N}{2}$ points of x_n,
- $h_{i,n}$ and $x_{i,n}$ start at the samples $\frac{i}{P}\frac{N}{2}$ of h_n and x_n, respectively.

The output is then computed as

$$y_{M,n} = \text{IFFT}\left(\sum_{i=0}^{P-1}\left(\text{FFT}^*(h_{i,n})\,\text{FFT}(x_{i,n})\right)\right), \tag{5.19}$$

with

$$\begin{aligned}\mathbf{h}_i &= \left[\begin{array}{cccc} h_{\frac{i}{P}\frac{N}{2}} & h_{\frac{i}{P}\frac{N}{2}+1} & \cdots & h_{\frac{i+1}{P}\frac{N}{2}-1} \end{array}\ \overbrace{\begin{array}{ccc} 0 & \ldots & 0 \end{array}}^{N/2}\right] \\ \mathbf{x}_i &= \left[\begin{array}{cccc} x_{\frac{i}{P}\frac{N}{2}} & x_{\frac{i}{P}\frac{N}{2}+1} & \cdots & x_{\frac{i+1+P}{P}\frac{N}{2}-1} \end{array}\right].\end{aligned} \tag{5.20}$$

There are thus $2P$ FFTs and one IFFT of $\frac{P+1}{P}\frac{N}{2}-1$ points. Note that as P increases, the number of output samples get closer to $N/2$, which is the number of samples of interest. However, the efficiency of the algorithm decreases because the number of FFTs increases linearly with P while the FFT length reduces only as $\frac{P+1}{P}$, consequently the number of sub-correlations should be as low as possible, i.e. 2.

5.3 Application for the acquisition of the L5, E5a and E5b signals

In this section, the straightforward and the proposed algorithms are compared for the acquisition of the GPS L5, Galileo E5a and E5b signals.

5.3.1 FFT lengths

The L5, E5a and E5b signals are BPSK signals and have a code chipping rate of 10.23 MHz. The minimum sampling frequency to get the main lobe (which contains 90 % of the signal power) is twice the chipping rate, i.e. 20.46 MHz. To have the usual code step of $\frac{1}{2}$ chip (see Section 2.1.3), the sampling frequency must also be twice the chipping rate. Therefore, we will consider a minimum sampling frequency of 20.46 MHz for these signals. This means that $N = 2 \times 20\,460 = 40\,920$, since the primary code length of these three signals is 1 ms.

Algorithm	Minimum FFT length	Minimum FFT length with small prime factors	Minimum power of two FFT length
Straightforward	$L = 40\,920$ $f_S = 20.46$	$L = 41\,472$ $f_S \in [20.46 - 20.736]$	$L = 65\,536$ $f_S \in [20.46 - 32.768]$
Proposed	$L = 30\,690$ $f_S = 20.46$	$L = 31\,104$ $f_S \in [20.46 - 20.736]$	$L = 32\,768$ $f_S \in [20.46 - 21.846]$

Table 5.4: FFT length (L) and sampling frequency range (f_S, in MHz) for the acquisition of the GPS L5, Galileo E5a and E5b signals.

Since the minimum length requires already relatively large FFTs, we will concentrate on values close to this minimum. Three cases are considered for the comparison : 1) The use of the smallest FFT length; 2) The use of the smallest FFT length that has 2 and 3 as prime factor only, which should provide better performance than the first case (see p. 46 the discussion about FFTs and the length of sequences); and 3) The use of the smallest FFT length that is a power of two, to check the conclusion obtained in Section 5.2.4.

For the first case, the FFT length is 40 920 for the straightforward algorithm, and 30 690 ($\frac{3}{4} \times 40\,920$) for the proposed algorithms. Note that using an FFT length of $N-1 = 40\,919$ for the straightforward algorithm is not interesting since it has higher prime factors than 40 920, and using $\frac{3N}{4} - 1 = 30\,689$ for the proposed algorithms is also not interesting because this is a prime number. Since the FFT lengths have relatively high prime factors ($40\,920 = 2^3 \times 3 \times 5 \times 11 \times 31$), the performance should be lower than for the other cases.

For the second case, the smallest number higher than 40 920 that has 2 and 3 as prime factors only is 41 472 ($= 2^9 \times 3^4$). Thus, the FFT length is 41 472 for the straightforward algorithm, and 31 104 ($= 2^7 \times 3^5$) for the proposed algorithms.

For the third case, the smallest power of two higher than 40 920 is 65 536 ($= 2^{16}$), and the smallest power of two higher than 30 690 is 32 768 ($= 2^{15}$). Thus, the FFT length is 65 536 for the straightforward algorithm, and 32 768 for the proposed algorithms.

The FFT lengths and the corresponding range for the sampling frequency are summarized in Table 5.4. For example, an FFT length of 41 472 for the straightforward algorithm can be obtained using a sampling frequency of 20.46 MHz and padding the sequence with 552 zeros, or using a sampling frequency of 20.736 MHz, or with any sampling frequency between these two values.

5.3.2 Software implementation

In this section, the straightforward algorithm and the proposed algorithm 1 are compared on five different personal computers using Matlab. The FFT function of Matlab is based on the FFTW library, which has no restriction on the FFT length (Frigo and Johnson [2005]). The

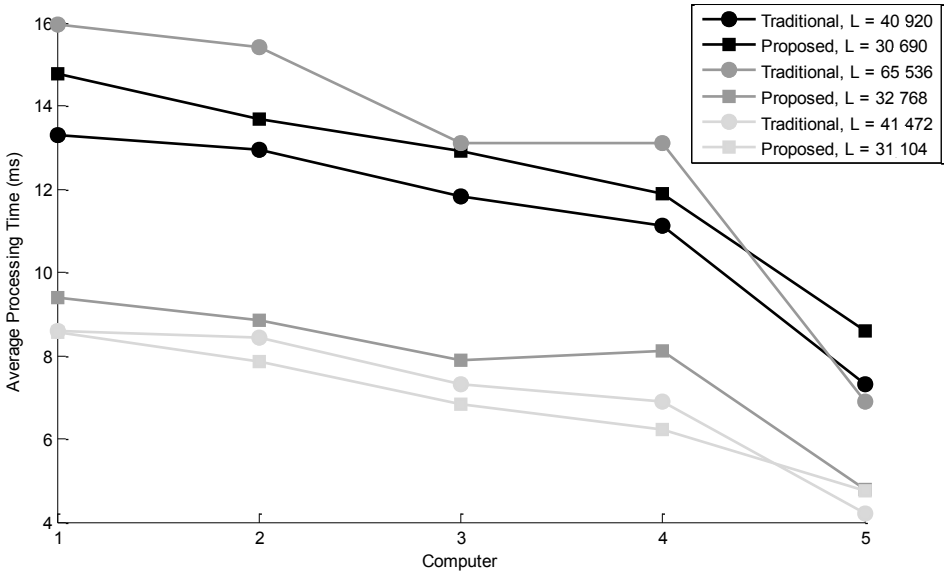


Figure 5.6: Average processing time of the algorithms on different computers for the GPS L5, Galileo E5a and E5b signals (L is the FFT length). The CPU of the computers are respectively : Core 2 Duo E4600 @ 2.40 GHz, QuadCore Xeon E5430 @ 2.66 GHz, Core 2 Duo E6700 @ 2.66 GHz, QuadCore Xeon E5430 2.66 GHz, Mobile Dual Core i7-2620M @ 3.2 GHz.

average processing time over 1000 runs is shown in Fig. 5.6.

Focusing on the ranking of the algorithms, it can be seen that there are mainly two groups. The first, with the longest processing time, includes the traditional algorithm for a power of two length, and both algorithms for the minimum length case. This is coherent with the expectations. Indeed, for $L = 65\,536$, the FFT length is far higher than for the other cases, which explains a longer processing time; and for the minimum FFT length case, since the length contains high prime factors, the FFT algorithm is less efficient than for the other cases. It can be noted that the straightforward algorithm is better than the proposed one for this case, as expected (see Section 5.2.3). The other group includes first the proposed algorithm for a power of two length, and then both algorithms for length that have small prime factors. The last two have equivalent performance, the proposed algorithm being slightly better on most of the computers, although not directly expected according to Section 5.2.3.

Regarding the case where the length is a power of two, the processing time is reduced by about 39 % in average using the proposed algorithm. This result is thus better than foreseen by the theoretical complexity (reduction of about 20 %). This is probably due to the fact that the performance of the FFTW algorithm in terms of FLOPS is slightly higher for a 32 768-point FFT than for a 65 536-point FFT (Frigo and Johnson [2005]), and due to to some internal specificities of the implemented FFT algorithm.

Implementation	Function	Logic usage (ALUT)	Memory usage (M9K)	Multipliers usage (DSP element)
Proposed algorithm 2 (Fig. 5.5)	5 32 768-point FFTs	5 × 7194	5 × 608	5 × 24
	2 Multipliers	0	0	2 × 4
	Total	35 970	3040	128
Straightforward algorithm (Fig. 5.3)	3 65 536-point FFTs	3 × 7627	3 × 1216	3 × 24
	1 Multiplier	0	0	4
	Total	22 881	3648	76
Ratio		1.57	0.83	1.68

Table 5.5: Comparison of the resources for the proposed and straightforward algorithms using the Altera FFT for the L5, E5a and E5b signals.

5.3.3 Hardware implementation

In this section, the straightforward and the proposed algorithms are compared for an implementation on an Altera FPGA. As in Chapter 4, the FFT used is the one provided by Altera (Altera [2013]), which requires a number of points that is a power of two. Thus, only the third case is considered for the hardware implementation.

Due to the large number of possibilities for the FFT implementation, for the evaluation of the resources, we consider a Stratix III FPGA, the streaming I/O data flow, a data and twiddle precision of 18 bits, complex multipliers implemented in DSP blocks using four real multipliers, and no logic function implemented in memory. As for Chapter 4, the resources for the FFT is estimated with the Altera MegaWizard Plug-In Manager, and the models defined in Appendix C are used for the other elements (multiplier and adder).

Application to reduce the processing time

The summary of the resources is given Table 5.5, considering the proposed algorithm 2 (which has an adder less than algorithm 1, but this is a negligible difference). It can be seen that the resources are higher for the proposed algorithm than for the straightforward algorithm. However, as discussed in Chapter 4, the processing time with the proposed algorithm is divided by a factor of two since the FFT length is divided by a factor two. Since the resources are increased by a factor less than two (the memory is even reduced), the proposed algorithm is thus more efficient than the straightforward algorithm.

Application to reduce the resources

If an increase of the resources is not wanted, the proposed algorithm can be adapted to use only three FFTs instead of five. In this case, the two proposed algorithms are not equivalent.

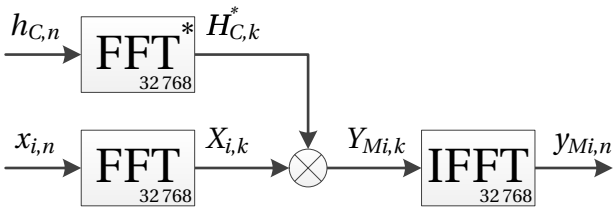


Figure 5.7: Implementation of the proposed algorithm 2 using three FFTs.

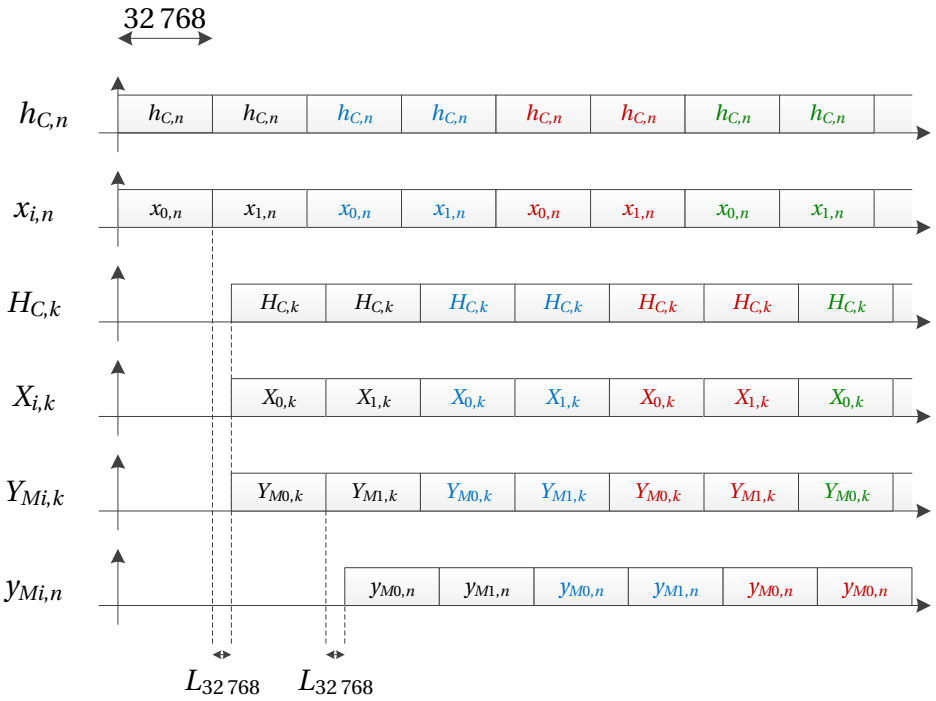


Figure 5.8: Timing diagram corresponding to Fig. 5.7. The colors inside the boxes identify the sequences.

Indeed, for algorithm 1, the output is the sum of two results. Consequently, a memory is needed to store an intermediate result. Whereas for algorithm 2, there are two outputs that can be computed one after the other, which does not require a memory for any intermediate result. However, this requires to generate twice the local code, but this is simple and do not require an additional memory.

The corresponding implementation of algorithm 2 is given Fig. 5.7, and the corresponding timing diagram using the Altera FFT is depicted Fig. 5.8. It can be seen that the Pth correlation result is fully available after $(P+1)65\,536+2L_{32\,768}$ clock cycles, which is about 65 536 cycles less than for the straightforward implementation.

The corresponding resources are given in Table 5.6. It can be seen that the logic resources are slightly reduced, the DSP resources are identical, and that the memory is reduced by 50 %, which is a significant reduction.

Implementation	Function	Logic usage (ALUT)	Memory usage (M9K)	Multipliers usage (DSP element)
Proposed algorithm 2 (Fig. 5.7)	3 32 768-point FFTs	3 × 7194	3 × 608	3 × 24
	1 Multiplier	0	0	1 × 4
	Total	21 582	1824	76
Straightforward algorithm (Fig. 5.3)	3 65 536-point FFTs	3 × 7627	3 × 1216	3 × 24
	1 Multiplier	0	0	4
	Total	22 881	3648	76
Ratio		0.94	0.5	1

Table 5.6: Comparison of the resources for the proposed and straightforward algorithms using the Altera FFT for the L5, E5a and E5b signals, when the proposed algorithm uses less FFTs.

5.3.4 Case with pre-averaging

To use smaller FFTs, it is possible to perform a sum before the FFT in order to have one sample per chip (Starzyk and Zhu [2001], Hegarty et al. [2003]). Applying this technique to the L5, E5a and E5b signals results in 10 230 points per code period, i.e. the equivalent sampling frequency is 10.23 MHz. Therefore the results will be similar to those obtained previously. Indeed, the prime factors of the length are similar, and for the radix-2 FFT case, according to Table 5.3, the FFT length for the proposed algorithms would be half the one for the straightforward algorithm.

5.4 Application to the acquisition of the E1 OS signal processed as BOC(1,1)

In this section, a similar application study as in Section 5.3 is done, but applied to the E1 OS signal processed as BOC(1,1). We do not consider that CBOC modulation, because it would require a higher sampling frequency and higher complexity, however the interested reader can easily verified if the proposed algorithms are interesting or not in this case.

5.4.1 FFT lengths

The E1 OS BOC(1,1) signal has a code chipping rate of 1.023 MHz. The minimum sampling frequency to get the two main lobes is four times the chipping rate, i.e. 4.092 MHz. To have the usual code step of $\frac{1}{6}$ chip (see Section 2.1.3), the sampling frequency must be six times the chipping rate, i.e. 6.138 MHz. Therefore, we will consider a minimum sampling frequency of 6.138 MHz for this signal. This means that $N = 2 \times 24\,552 = 49\,104$, since the primary code length of this signal is 4 ms.

Algorithm	Minimum FFT length	Minimum FFT length with small prime factors	Minimum power of two FFT length
Straightforward	$L = 49\,104$ $f_S = 6.138$	$L = 49\,152$ $f_S \in [6.138 - 6.144]$	$L = 65\,536$ $f_S \in [6.138 - 8.192]$
Proposed	$L = 36\,828$ $f_S = 6.138$	$L = 36\,864$ $f_S \in [6.138 - 6.144]$	$L = 32\,768$* $f_S \in [6.138 - 6.144]$

Table 5.7: FFT length (L) and sampling frequency range (f_S, in MHz) for the acquisition of the E1 OS BOC(1,1) signal. *use of three sub-correlations.

For the first case (minimum FFT length), the FFT length is 49 104 for the straightforward algorithm, and 36 828 ($\frac{3}{4} \times 49\,104$) for the proposed algorithms. Note that using an FFT length of $N - 1 = 49\,103$ for the straightforward algorithm is not interesting since this is a prime number, and using $\frac{3N}{4} - 1 = 36\,827$ for the proposed algorithms is also not interesting since its has higher prime factors than 36 828. Since the FFT lengths have relatively high prime factors ($49\,104 = 2^4 \times 3^2 \times 11 \times 31$), the performance should be lower than for the other cases.

For the second case (minimum FFT length with small prime factors), the smallest number higher than 49 104 that has 2 and 3 as prime factors only is 49 152 ($= 2^{14} \times 3$). Thus, the FFT length is 49 152 for the straightforward algorithm, and 36 864 ($= 2^{12} \times 3^2$) for the proposed algorithms.

For the third case (minimum power of two FFT length), the smallest power of two higher than 49 104 is 65 536, and the smallest power of two higher than 36 828 is also 65 536. Thus, the FFT length for the straightforward and the proposed algorithms is 65 536. We know that in this case the proposed algorithms are less efficient, so we will consider three sub-correlations, which gives FFT length of 32 768 points. Since here we have three sub-correlations, it is sure that the proposed algorithms are less efficient for E1 OS than for L5, E5a and E5b.

The FFT lengths and the corresponding range for the sampling frequency are summarized in Table 5.7.

5.4.2 Software implementation

The proposed and traditional algorithms are compared on five different personal computers using Matlab as in Section 5.3.2. The average processing time over 1000 runs is shown in Fig. 5.9.

The results are more heterogeneous than in Fig. 5.6. It can be seen that however, the least three performing algorithms are the same as previously. The proposed algorithm for a power of two length is less efficient than previously, which is expected since there are more FFTs performed. Finally, the best two options are for an FFT length that has small prime factors, with an advantage for the straightforward algorithm this time (as was expected according to

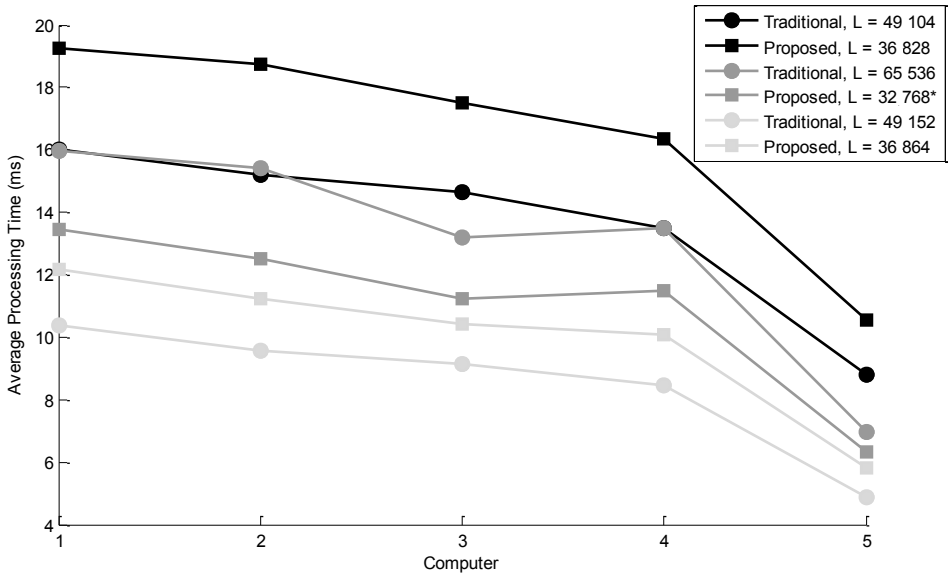


Figure 5.9: Average processing time of the algorithms on different computers for the Galileo E1 OS BOC(1,1) signal. *use of three sub-correlations.

Section 5.2.3).

In summary, it can be seen that the most efficient way to perform the correlation is to use lengths with small prime factors and limited zero-padding. Here, the small prime factors are 2 and 3, but 5 and 7 can also be considered. With this wide variety of lengths available, it is rarely possible to reduce significantly the zero-padding with the proposed algorithms, and even in the best cases the gain in performance would be marginal.

Regarding the case where the length is a power of two, the processing time is reduced by about 15 % in average using the proposed algorithm. Thus, as in Section 5.3.2, this result is better than foreseen by the theoretical complexity (increase of about 11 %).

5.4.3 Hardware implementation

The implementation of the straightforward algorithm is identical to the one for the L5, E5a and E5b signals, and the implementation of the proposed algorithm 2 with three sub-correlations is provided in Fig. 5.10. The corresponding resources are provided in Table 5.8, where it can be seen that the proposed algorithm requires far more resources than the straightforward one, more than a factor 2 (except for the memory). Implementing only three FFTs as done in Section 5.3.3 would reduce the resources, but the processing time would be increased by a factor 1.5. Consequently, the proposed algorithm is not more efficient than the traditional algorithm in this case.

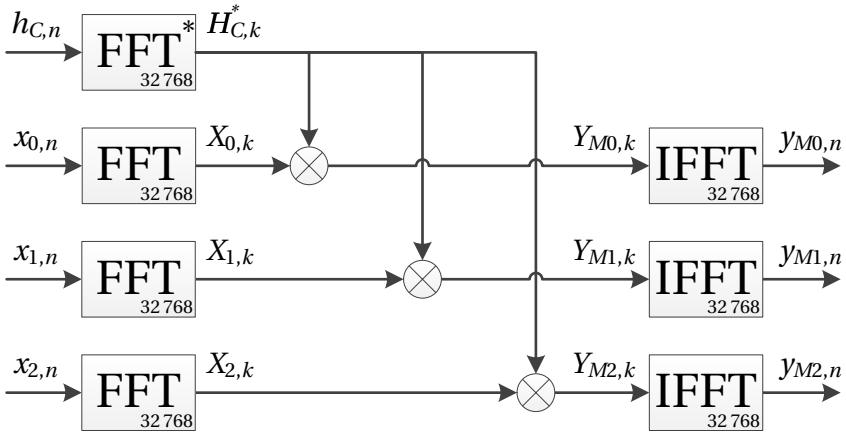


Figure 5.10: Implementation of the proposed algorithm for the E1 OS BOC(1,1) signal.

Implementation	Function	Logic usage (ALUT)	Memory usage (M9K)	Multipliers usage (DSP element)
Proposed algorithm 2 (Fig. 5.10)	7 32 768-point FFTs	7 × 7194	7 × 608	7 × 24
	3 Multipliers	0	0	3 × 4
	Total	50 358	4256	180
Straightforward algorithm (Fig. 5.3)	3 65 536-point FFTs	3 × 7627	3 × 1216	3 × 24
	1 Multiplier	0	0	4
	Total	22 881	3648	76
Ratio		2.20	1.17	2.37

Table 5.8: Comparison of the resources for the proposed and straightforward algorithm using the Altera FFT for the E1 OS BOC(1,1) signal, when the proposed algorithm uses three sub-correlations.

5.4.4 Modifying the correlation length to improve the performance

The minimum correlation length was fixed according to the code length and the code step. While the code length is kept fixed, however, the code step can be modified using a different sampling frequency (which will impact the code alignment loss).

Previously, the maximum code step considered was $\frac{1}{6}$ since the minimum sampling frequency was 6.138 MHz, which gave a minimum correlation length of $N = 49\,104$. By increasing the code step in order to obtain a correlation length of 43 692 instead of 49 104, the proposed algorithms could use FFTs of 32 768 points with two sub-correlations. In this case, the results would be the same as for the L5, E5a and E5b signals. Such length is reached for a maximum sampling frequency of 5.4615 MHz, which means a code step of about $\frac{1}{5.34}$ chip, resulting in a maximum and average loss of 2.87 dB and 1.31 dB (van Diggelen [2009] pp. 155–158), respectively. This is very close to the loss using a code step of $\frac{1}{6}$ chip, namely 2.50 dB at

maximum and 1.16 dB in average.

Therefore, it is possible to use a slightly larger code step for the Galileo E1 OS signal processed as BOC(1,1), in order to be able to use the proposed algorithms with two sub-correlations and smaller FFTs, as for the L5, E5a and E5b signals, at the expense of a very small loss in terms of SNR.

5.5 Application to the acquisition of the E1 OS signal processed as BPSK

If the E1 OS signal is processed as a BPSK signal, the minimum sampling frequency to get a main lobe is twice the chipping rate, i.e. 2.046 MHz. To have the usual code step of $\frac{1}{2}$ chip, the sampling frequency must also be twice the chipping rate. Therefore, we will consider a minimum sampling frequency of 2.046 MHz for this signal. This means that $N = 2 \times 8184 = 16\,368$, since the primary code length of this signal is 4 ms.

Considering the same three cases as before, for the first case, the FFT length is 16 368 for the straightforward algorithm, and 12 276 ($\frac{3}{4} \times 16\,368$) for the proposed algorithms.

For the second case, the smallest number higher than 16 368 that has 2 and 3 as prime factors only is 16 384 ($= 2^{14}$), which in fact has just 2 as prime factor. Thus, the FFT length is thus 16 384 for the straightforward algorithm, and 12 288 ($= 2^{12} \times 3$) for the proposed algorithms. Since the FFT length for the straightforward algorithm is a power of two, it is expected that the proposed algorithms will be less efficient for this case.

For the third case, the smallest power of two higher than 16 368 is 16 384, and the smallest power of two higher than 12 288 is also 16 384. Thus, the FFT length for the straightforward and the proposed algorithms is 16 384. We know that in this case the proposed algorithms are less efficient. Even considering three sub-correlations, the FFT length would still be 16 384 for the proposed algorithms. In fact, to halve the FFT length, we should use 1023 sub-correlations, which is clearly not efficient. The proposed algorithm is thus less efficient for this case also.

Consequently, the proposed algorithm is not efficient for the acquisition of the E1 OS signal processed as BPSK considering the minimum sampling frequency, 2.046 MHz. However, such a low sampling frequency has an impact on the positioning accuracy, it is thus common to use a higher frequency. In this case, the choice for the best algorithm with the radix-2 FFT can be found using Table 5.3.

5.6 Summary

In this chapter, we discussed the problem of decreasing the complexity of the acquisition of the modernized GPS and Galileo signals that have a secondary code that is not exploited. The

straightforward solution of doubling the size of the correlation and discarding half of the points calculated is not satisfying, which led us to look for a more efficient algorithm. We then proposed two algorithms that transform the initial correlation into two smaller sub-correlations, without loss of sensitivity since the samples of interest are computed exactly. From these two algorithms, it was shown that one is preferable when the local code is computed offline, and that the other is preferable for hardware implementations.

The proposed algorithms are more efficient than the straightforward algorithm when the latter requires significant zero-padding. The zero-padding size depends on the sampling frequency and the type of FFT used, and it can be significant only when the radix-2 FFT is used. Therefore the proposed algorithms are interesting for hardware and DSP-based receivers, but not for a computer-based receivers (which can efficiently implement FFT of any lengths). Considering the radix-2 FFT, the proposed algorithms are more efficient for half of the possible sampling frequencies. To rapidly know if the proposed algorithms are more efficient for a specific case, it is enough to look at Fig. 5.11, which is a graphical version of Table 5.3.

If the sampling frequency is in a range where the proposed algorithms are more efficient, it has been shown that the theoretical number of operations is reduced by about 20 %, and that the memory required for an FPGA implementation is divided by two.

For the GPS L5, Galileo E5a and E5b signals, if the sampling frequency is between 20.46 MHz and 21.846 MHz, the proposed algorithms are more efficient.

For the Galileo E1 OS signal processed as BOC(1,1), the minimum sampling frequency considered, 6.138 MHz, is in a range where the proposed algorithms are not more efficient. However, if the sampling frequency is decreased to 5.4615 MHz at the expense of an average loss 0.15 dB (due to a larger code step in the acquisition), this sampling frequency is in a range where the proposed algorithms are more efficient.

For the Galileo E1 OS signal processed as BPSK, and the GPS L1 C/A signal (where transition can be due to the data), the minimum sampling frequency considered, 2.046 MHz, is in a range where the proposed algorithms are not more efficient. But from 2.049 MHz to 2.730 MHz, the proposed algorithms are more efficient. Thus, the conclusion really depends on the context.

Using similar analysis as done in this chapter, it is easy to determine if the proposed algorithms are suitable for other and/or future GNSS signals. For example, the E6 CS signal is a good candidate since the minimum correlation length is half the one for the L5, E5a and E5B signals. The proposed algorithm may be also interesting for the E6 PRS signal, the GPS M signal or the future GLONASS CDMA and BeiDou signals.

The problem discussed in this chapter does not necessarily restrict to GNSS and to the parallel code search. Any system that performs a circular correlation (or a convolution) between two signals where one of them has half of zeros and where half of the output is discarded can

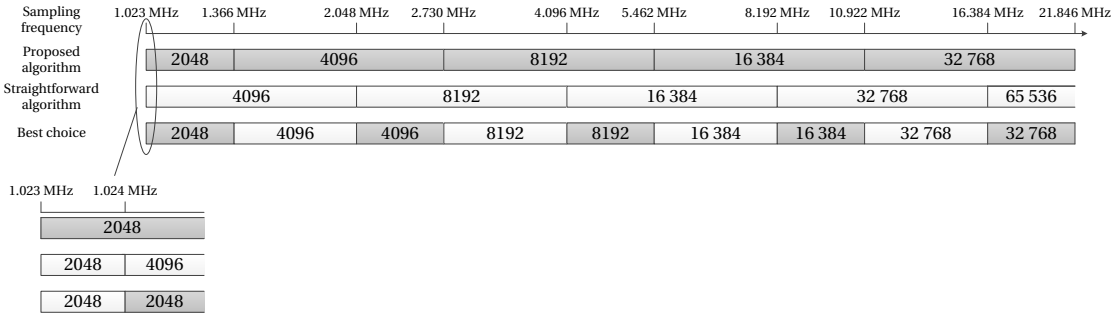


Figure 5.11: Radix-2 FFT length for the proposed and the straightforward algorithms in function of the sampling frequency considering a code of 1 ms (for a code of 4 ms, multiply the actual sampling frequency by 4). Note that the axis is logarithmic.

use the proposed algorithm. In GNSS, this problem is also present in the DBZP acquisition method (Foucras et al. [2012]), as well as for the acquisition of signals using the secondary code, as will shown be in Chapter 6.

6 Acquisition of modern GNSS signals for high sensitivity

In this chapter, we discuss the problem of the parallel code search acquisition when the complete tiered code is used, i.e. the code used to compute the correlation using FFTs contains a primary and a secondary code. This case is typically for high sensitivity receivers, since in this case there is no limit for the coherent integration time due to transitions (but there is still the limits due to the oscillator or the receiver dynamics (van Diggelen [2014])). However, a direct implementation requires a significant amount of resources, because large FFTs are involved. For example, the L5 signal has a primary code of 10 230 chips and a secondary code of 20 chips, therefore considering two samples per chip, the tiered code is composed of 409 200 samples. In this chapter, we discuss two directions to answer this problem.

First, we look for implementations that use smaller FFTs (same order of magnitude as the length of the primary code), in order to be able to implement the parallel code search in FPGAs since the direct implementation is not possible. The aim is not to reduce the complexity (i.e. the number of operations), but we compute the complexity to compare the different implementations. We also show the relation with the problem of Chapter 5. Since the matrix notation is used, to show more easily the results, we consider very small sequences for the demonstration (4 samples for the primary code and 3 chips for the secondary code), but we provide the complexity for the general case, and we consider 20 460 samples and 20 chips (as for the GPS L5 signal) for the numerical application. For the evaluation of the complexity, we consider that an FFT of N points requires $\frac{N}{2}\log_2(N)$ multiplications and $N\log_2(N)$ additions. Of course, this is true only for the radix-2 FFT, however this approximation will allow us to have an idea of the complexity.

Second, we look for implementations that reduce the complexity. The idea we exploit is that performing some combinations of the tiered code may lead to some sub-sequences that will contain only zeros. For this, we use the Chinese remainder theorem already seen in Chapter 4. We apply this to the GPS L5 signal, and in addition to the L5 secondary code, we also study the other binary codes of 20 bits length to verify the potential of the proposed method for other codes, and finally we briefly discuss the application to other GNSS signals.

6.1 Expression of the tiered code

As indicated in Chapter 1, a tiered code is composed of a primary code repeated several times where each primary code period is multiplied by a chip of the secondary code. In the following, we will denote h_n the tiered code, p_n the primary code and s_n the secondary code (the corresponding notations in Chapter 1 for the pilot channel were $c_q(nT_s)$, $c_{p,q}(nT_s)$ and $c_{s,q}(nT_s)$, respectively).

Considering that the primary code p_n contains N_P samples, and that the secondary code s_n contains N_S chips, we can write

$$\mathbf{p} = \begin{bmatrix} p_0 & p_1 & \dots & p_{N_P-1} \end{bmatrix}, \tag{6.1}$$

and

$$\mathbf{s} = \begin{bmatrix} s_0 & s_1 & \dots & s_{N_S-1} \end{bmatrix}. \tag{6.2}$$

Then, the tiered can be defined as

$$\begin{aligned} \mathbf{h} &= \mathbf{s} \otimes \mathbf{p} \\ &= \begin{bmatrix} s_0\mathbf{p} & s_1\mathbf{p} & \dots & s_{N_s-1}\mathbf{p} \end{bmatrix} \\ &= \begin{bmatrix} s_0 p_0 & s_0 p_1 & \dots & s_0 p_{N_p-1} & s_1 p_0 & s_1 p_1 & \dots & s_1 p_{N_p-1} & \dots & s_{N_s-1} p_{N_p-1} \end{bmatrix} \\ &= \begin{bmatrix} h_0 & h_1 & \dots & h_{N-1} \end{bmatrix}, \end{aligned} \tag{6.3}$$

where $\otimes$ denotes the Kronecker product, and $N = N_P N_S$.

6.2 Direct implementation of the circular correlation

The implementation of the circular correlation of the incoming signal x_n and the local tiered code h_n can be performed as usual using 3 FFTs, as shown in Fig. 6.1.

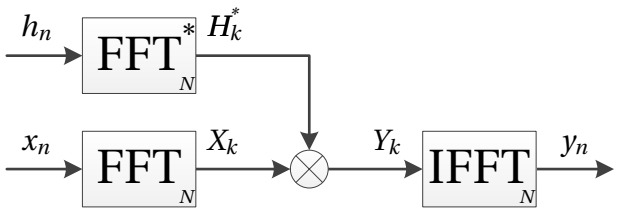


Figure 6.1: Computation of a circular correlation of N points using FFTs.

Considering for example a number of samples in one primary code period of $N_P = 4$, and a number of chips in the secondary codes of $N_S = 3$, the tiered code is composed of $N = N_P N_S = 12$ samples, and the circular correlation between the incoming signal x_n and the local tiered

code can be expressed as

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \end{bmatrix} = \begin{bmatrix} s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 \\ s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 \\ s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 \\ s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 \\ s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 \\ s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 \\ s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 \\ s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 \\ s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 \\ s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 \\ s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 \\ s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix} \tag{6.4}$$

$$\mathbf{y} = \mathbf{H}\,\mathbf{x},$$

or as

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \end{bmatrix} = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_0 \\ x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_0 & x_1 \\ x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_0 & x_1 & x_2 \\ x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_0 & x_1 & x_2 & x_3 \\ x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_0 & x_1 & x_2 & x_3 & x_4 \\ x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\ x_7 & x_8 & x_9 & x_{10} & x_{11} & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_8 & x_9 & x_{10} & x_{11} & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ x_9 & x_{10} & x_{11} & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ x_{10} & x_{11} & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \\ x_{11} & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} \end{bmatrix} \begin{bmatrix} s_0p_0 \\ s_0p_1 \\ s_0p_2 \\ s_0p_3 \\ s_1p_0 \\ s_1p_1 \\ s_1p_2 \\ s_1p_3 \\ s_2p_0 \\ s_2p_1 \\ s_2p_2 \\ s_2p_3 \end{bmatrix} \tag{6.5}$$

$$\mathbf{y} = \mathbf{X}\,\mathbf{h},$$

where $\mathbf{H}$ is a right circulant matrix, and $\mathbf{X}$ is a left circulant matrix. Implementing this operation with FFTs requires 3 FFTs of N points and 1 product of N points, as shown in Fig. 6.1. Therefore the number of multiplications is

$$\begin{aligned} N_{mul} &= 3\frac{N}{2}\log_2(N) + N \\ &= 3\frac{N_P N_S}{2}\log_2(N_P N_S) + N_P N_S \\ &= N_P N_S\left(\frac{3}{2}\log_2(N_P) + \frac{3}{2}\log_2(N_S) + 1\right), \end{aligned} \tag{6.6}$$

and the number of additions is

$$\begin{aligned} N_{add} &= 3N\log_2(N) \\ &= 3N_P N_S\log_2(N_P N_S) \\ &= N_P N_S\left(3\log_2(N_P) + 3\log_2(N_S)\right) \end{aligned} \tag{6.7}$$

Considering $N_P = 20\,460$ and $N_S = 20$, this gives

$$\begin{aligned} N_{mul} &\approx 409\,200\,(28.96) \approx 11\,851\,934 \\ N_{add} &\approx 409\,200\,(55.93) \approx 22\,885\,467. \end{aligned} \tag{6.8}$$

6.3 Implementations to use smaller FFTs

In this section, we present different implementations to compute the circular correlation defined by Eqs. (6.4) and (6.5) using smaller FFTs. This is done by separating the sequences either by downsampling or by segmentation, and either by a factor N_P or N_S.

6.3.1 Separation by downsampling

In this section, we look at the results when we separate the input and output samples by performing a downsampling. First, we perform a downsampling by a factor N_S, and then by a factor N_P.

Downsampling by a factor N_S using the equation with the matrix H

By separating the samples by performing a downsampling by a factor N_S, each sequence is separated into N_S sub-sequences of N_P samples, and Eq. (6.4) can be expressed as

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_3 \\ y_6 \\ y_9 \end{bmatrix} &= \begin{bmatrix} s_0p_0 & s_0p_3 & s_1p_2 & s_2p_1 \\ s_2p_1 & s_0p_0 & s_0p_3 & s_1p_2 \\ s_1p_2 & s_2p_1 & s_0p_0 & s_0p_3 \\ s_0p_3 & s_1p_2 & s_2p_1 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_3 \\ x_6 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_1 & s_1p_0 & s_1p_3 & s_2p_2 \\ s_2p_2 & s_0p_1 & s_1p_0 & s_1p_3 \\ s_1p_3 & s_2p_2 & s_0p_1 & s_1p_0 \\ s_1p_0 & s_1p_3 & s_2p_2 & s_0p_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_7 \\ x_{10} \end{bmatrix} + \begin{bmatrix} s_0p_2 & s_1p_1 & s_2p_0 & s_2p_3 \\ s_2p_3 & s_0p_2 & s_1p_1 & s_2p_0 \\ s_2p_0 & s_2p_3 & s_0p_2 & s_1p_1 \\ s_1p_1 & s_2p_0 & s_2p_3 & s_0p_2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_5 \\ x_8 \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_1 \\ y_4 \\ y_7 \\ y_{10} \end{bmatrix} &= \begin{bmatrix} s_2p_3 & s_0p_2 & s_1p_1 & s_2p_0 \\ s_2p_0 & s_2p_3 & s_0p_2 & s_1p_1 \\ s_1p_1 & s_2p_0 & s_2p_3 & s_0p_2 \\ s_0p_2 & s_1p_1 & s_2p_0 & s_2p_3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_3 \\ x_6 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_0p_3 & s_1p_2 & s_2p_1 \\ s_2p_1 & s_0p_0 & s_0p_3 & s_1p_2 \\ s_1p_2 & s_2p_1 & s_0p_0 & s_0p_3 \\ s_0p_3 & s_1p_2 & s_2p_1 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_7 \\ x_{10} \end{bmatrix} + \begin{bmatrix} s_0p_1 & s_1p_0 & s_1p_3 & s_2p_2 \\ s_2p_2 & s_0p_1 & s_1p_0 & s_1p_3 \\ s_1p_3 & s_2p_2 & s_0p_1 & s_1p_0 \\ s_1p_0 & s_1p_3 & s_2p_2 & s_0p_1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_5 \\ x_8 \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_2 \\ y_5 \\ y_8 \\ y_{11} \end{bmatrix} &= \begin{bmatrix} s_2p_2 & s_0p_1 & s_1p_0 & s_1p_3 \\ s_1p_3 & s_2p_2 & s_0p_1 & s_1p_0 \\ s_1p_0 & s_1p_3 & s_2p_2 & s_0p_1 \\ s_0p_1 & s_1p_0 & s_1p_3 & s_2p_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_3 \\ x_6 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_2p_3 & s_0p_2 & s_1p_1 & s_2p_0 \\ s_2p_0 & s_2p_3 & s_0p_2 & s_1p_1 \\ s_1p_1 & s_2p_0 & s_2p_3 & s_0p_2 \\ s_0p_2 & s_1p_1 & s_2p_0 & s_2p_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_7 \\ x_{10} \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_0p_3 & s_1p_2 & s_2p_1 \\ s_2p_1 & s_0p_0 & s_0p_3 & s_1p_2 \\ s_1p_2 & s_2p_1 & s_0p_0 & s_0p_3 \\ s_0p_3 & s_1p_2 & s_2p_1 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_5 \\ x_8 \\ x_{11} \end{bmatrix},
\end{aligned} \tag{6.9}$$

or in a more concise way

$$\begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 \\ \mathbf{P}\,\mathbf{H}_2 & \mathbf{H}_0 & \mathbf{H}_1 \\ \mathbf{P}\,\mathbf{H}_1 & \mathbf{P}\,\mathbf{H}_2 & \mathbf{H}_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}, \tag{6.10}$$

where $\mathbf{P}$ is a permutation matrix, and the matrices $\mathbf{H}_i$ are circulant. Implementing these operations using FFTs requires

- N_S FFTs of N_P points, for the sequences $h_{i,n}$,
- N_S FFTs of N_P points, for the sequences $x_{i,n}$,
- N_S IFFTs of N_P points, for the sequences $y_{i,n}$,

- $N_S - 1$ products of N_P points, for the permutation performed by a multiplication with a complex exponential,
- N_S^2 products of N_P points, for the matrix-vector products,
- $N_S(N_S - 1)$ additions of N_P points, for the additions of the results of the matrix-vector products.

Therefore, the number of multiplications is

$$\begin{aligned} N_{mul} &= 3N_S\left(\frac{N_P}{2}\log_2(N_P)\right) + \left(N_S^2 + N_S - 1\right)N_P \\ &= N_P N_S\left(\frac{3}{2}\log_2(N_P) + N_S + 1 - \frac{1}{N_S}\right), \end{aligned} \tag{6.11}$$

and the number of additions is

$$\begin{aligned} N_{add} &= 3N_S\left(N_P\log_2(N_P)\right) + N_S(N_S-1)N_P \\ &= N_P N_S\left(3\log_2(N_P) + N_S - 1\right). \end{aligned} \tag{6.12}$$

Considering $N_P = 20\,460$ and $N_S = 20$, this gives

$$\begin{aligned} N_{mul} &\approx 409\,200\,(42.48) - 20\,460 \approx 17\,362\,674 \\ N_{add} &\approx 409\,200\,(61.96) \approx 25\,354\,669, \end{aligned} \tag{6.13}$$

which means an increase of 46.5 % for the multiplications and 10.8 % for the additions, compared to the direct implementation. The large increase for the multiplications is due to the N_S^2 matrix-vector products.

Here, each matrix $\mathbf{H}_i$ contains all the secondary code chips, thus we cannot exploit the repetitions in the tiered code. This means that the same result would be obtained with any signal.

Downsampling by a factor N_S using the equation with the matrix X

Doing the same separation as previously using Eq. (6.5) leads to a similar implementation with the same complexity, therefore we do not give the details.

Downsampling by a factor N_P using the equation with the matrix H

By separating the samples by performing a downsampling by a factor N_P, each sequence is separated into N_P sub-sequences of N_S samples, and Eq. (6.4) can be expressed as

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_4 \\ y_8 \end{bmatrix} &= \begin{bmatrix} s_0p_0 & s_1p_0 & s_2p_0 \\ s_2p_0 & s_0p_0 & s_1p_0 \\ s_1p_0 & s_2p_0 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_4 \\ x_8 \end{bmatrix} + \begin{bmatrix} s_0p_1 & s_1p_1 & s_2p_1 \\ s_2p_1 & s_0p_1 & s_1p_1 \\ s_1p_1 & s_2p_1 & s_0p_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_5 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_2 & s_1p_2 & s_2p_2 \\ s_2p_2 & s_0p_2 & s_1p_2 \\ s_1p_2 & s_2p_2 & s_0p_2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_6 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_3 & s_1p_3 & s_2p_3 \\ s_2p_3 & s_0p_3 & s_1p_3 \\ s_1p_3 & s_2p_3 & s_0p_3 \end{bmatrix} \begin{bmatrix} x_3 \\ x_7 \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_1 \\ y_5 \\ y_9 \end{bmatrix} &= \begin{bmatrix} s_2p_3 & s_0p_3 & s_1p_3 \\ s_1p_3 & s_2p_3 & s_0p_3 \\ s_0p_3 & s_1p_3 & s_2p_3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_4 \\ x_8 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_1p_0 & s_2p_0 \\ s_2p_0 & s_0p_0 & s_1p_0 \\ s_1p_0 & s_2p_0 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_5 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_1 & s_1p_1 & s_2p_1 \\ s_2p_1 & s_0p_1 & s_1p_1 \\ s_1p_1 & s_2p_1 & s_0p_1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_6 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_2 & s_1p_2 & s_2p_2 \\ s_2p_2 & s_0p_2 & s_1p_2 \\ s_1p_2 & s_2p_2 & s_0p_2 \end{bmatrix} \begin{bmatrix} x_3 \\ x_7 \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_2 \\ y_6 \\ y_{10} \end{bmatrix} &= \begin{bmatrix} s_2p_2 & s_0p_2 & s_1p_2 \\ s_1p_2 & s_2p_2 & s_0p_2 \\ s_0p_2 & s_1p_2 & s_2p_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_4 \\ x_8 \end{bmatrix} + \begin{bmatrix} s_2p_3 & s_0p_3 & s_1p_3 \\ s_1p_3 & s_2p_3 & s_0p_3 \\ s_0p_3 & s_1p_3 & s_2p_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_5 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_1p_0 & s_2p_0 \\ s_2p_0 & s_0p_0 & s_1p_0 \\ s_1p_0 & s_2p_0 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_6 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_1 & s_1p_1 & s_2p_1 \\ s_2p_1 & s_0p_1 & s_1p_1 \\ s_1p_1 & s_2p_1 & s_0p_1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_7 \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_3 \\ y_7 \\ y_{11} \end{bmatrix} &= \begin{bmatrix} s_2p_1 & s_0p_1 & s_1p_1 \\ s_1p_1 & s_2p_1 & s_0p_1 \\ s_0p_1 & s_1p_1 & s_2p_1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_4 \\ x_8 \end{bmatrix} + \begin{bmatrix} s_2p_2 & s_0p_2 & s_1p_2 \\ s_1p_2 & s_2p_2 & s_0p_2 \\ s_0p_2 & s_1p_2 & s_2p_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_5 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_2p_3 & s_0p_3 & s_1p_3 \\ s_1p_3 & s_2p_3 & s_0p_3 \\ s_0p_3 & s_1p_3 & s_2p_3 \end{bmatrix} \begin{bmatrix} x_2 \\ x_6 \\ x_9 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_1p_0 & s_2p_0 \\ s_2p_0 & s_0p_0 & s_1p_0 \\ s_1p_0 & s_2p_0 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_7 \\ x_{11} \end{bmatrix},
\end{aligned} \tag{6.14}$$

or in a more concise way

$$\begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} = \begin{bmatrix} p_0\mathbf{S} & p_1\mathbf{S} & p_2\mathbf{S} & p_3\mathbf{S} \\ p_3\mathbf{P}\,\mathbf{S} & p_0\mathbf{S} & p_1\mathbf{S} & p_2\mathbf{S} \\ p_2\mathbf{P}\,\mathbf{S} & p_3\mathbf{P}\,\mathbf{S} & p_0\mathbf{S} & p_1\mathbf{S} \\ p_1\mathbf{P}\,\mathbf{S} & p_2\mathbf{P}\,\mathbf{S} & p_3\mathbf{P}\,\mathbf{S} & p_0\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}, \tag{6.15}$$

where **P** is a permutation matrix, and **S** a circulant matrix. Implementing these operations using FFTs requires

- 1 FFT of N_S points, for the sequence s_n,
- N_P FFTs of N_S points, for the sequences $x_{i,n}$,
- N_P IFFTs of N_S points, for the sequences $y_{i,n}$,
- 1 product of N_S points, for the permutation performed by a multiplication with a complex exponential,
- $2N_P - 1$ products of N_S points, for the matrix-vector products,
- $N_P(N_P - 1)$ additions of N_S points, for the combinations of the results of the matrix-vector products.

Note that here we do not consider the multiplication by the samples p_i, as it can be seen as an addition or a subtraction. Therefore, the number of multiplications is

$$\begin{aligned}
N_{mul} &= (2N_P + 1)\left(\frac{N_S}{2}\log_2(N_S)\right) + (2N_P - 1 + 1)\,N_S \\
&= N_P N_S\left(\log_2(N_S) + 2 + \frac{\log_2(N_S)}{2N_P}\right),
\end{aligned} \tag{6.16}$$

and the number of additions is

$$\begin{aligned} N_{add} &= (2N_P+1)\left(N_S\log_2(N_S)\right) + N_P(N_P-1)N_S \\ &= N_P N_S\left(2\log_2(N_S) + N_P - 1 + \frac{\log_2(N_S)}{N_P}\right). \end{aligned} \tag{6.17}$$

Considering $N_P = 20\,460$ and $N_S = 20$, this gives

$$\begin{aligned} N_{mul} &\approx 409\,200\,(6.32) + 43.22 \approx 2\,586\,976 \\ N_{add} &\approx 409\,200\,(20\,467.64) + 86.44 \approx 8\,375\,359\,952, \end{aligned} \tag{6.18}$$

which means an reduction of 78.2 % for the multiplications but an increase of 36 496.8 % for the additions, compared to the direct implementation. The large increase for the additions is due to the $N_P(N_P-1)$ additions between the matrix-vector products, since N_P is a high value.

Here, we are able to exploit the repetitions in the tiered code since we have only one matrix, which means that the number of FFTs is reduced by about one third. However, there is a huge increase of the number additions, due to the additions between intermediate results, which make this implementation inefficient.

Downsampling by a factor N_P using the equation with the matrix X

Doing the same separation as previously using Eq. (6.5) leads a similar implementation with the same complexity, therefore we do not give the details.

6.3.2 Separation by segmentation

In this section, we look at the results when we separate the input and output samples by sections. First, we perform a segmentation by a factor N_S, and then by a factor N_P.

Segmentation by a factor N_S using the equation with the matrix H

By segmenting the sequences by a factor N_S, each sequence is separated into N_S sub-sequences of N_P samples, and Eq. (6.4) can be expressed as

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} &= \begin{bmatrix} s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 \\ s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 \\ s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 \\ s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 \\ s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 \\ s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 \\ s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 \end{bmatrix} \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 \\ s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 \\ s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 \\ s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 \end{bmatrix} \begin{bmatrix} x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} &= \begin{bmatrix} s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 \\ s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 \\ s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 \\ s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 \\ s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 \\ s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 \\ s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 \\ s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 \\ s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 \\ s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 \end{bmatrix} \begin{bmatrix} x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_8 \\ y_9 \\ y_{10} \\ y_{11} \end{bmatrix} &= \begin{bmatrix} s_1p_0 & s_1p_1 & s_1p_2 & s_1p_3 \\ s_0p_3 & s_1p_0 & s_1p_1 & s_1p_2 \\ s_0p_2 & s_0p_3 & s_1p_0 & s_1p_1 \\ s_0p_1 & s_0p_2 & s_0p_3 & s_1p_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} s_2p_0 & s_2p_1 & s_2p_2 & s_2p_3 \\ s_1p_3 & s_2p_0 & s_2p_1 & s_2p_2 \\ s_1p_2 & s_1p_3 & s_2p_0 & s_2p_1 \\ s_1p_1 & s_1p_2 & s_1p_3 & s_2p_0 \end{bmatrix} \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_0p_1 & s_0p_2 & s_0p_3 \\ s_2p_3 & s_0p_0 & s_0p_1 & s_0p_2 \\ s_2p_2 & s_2p_3 & s_0p_0 & s_0p_1 \\ s_2p_1 & s_2p_2 & s_2p_3 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix},
\end{aligned} \tag{6.19}$$

or in a more concise way

$$\begin{bmatrix}\mathbf{y}_0\\ \mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix} = \begin{bmatrix}\mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2\\ \mathbf{H}_2 & \mathbf{H}_0 & \mathbf{H}_1\\ \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_0\end{bmatrix}\begin{bmatrix}\mathbf{x}_0\\ \mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}. \tag{6.20}$$

Each matrix $\mathbf{H}_i$ contains only two secondary code chips, one is present on and above the diagonal while the other is present below the diagonal. This means that a matrix $\mathbf{H}_i$ is either circulant or skew-circulant (see Appendix A.2). There are thus only two possible matrices. For example, if we consider that $s_0 = 1$, $s_1 = 1$, and $s_2 = -1$, Eq. (6.20) becomes

$$\begin{aligned}\begin{bmatrix}\mathbf{y}_0\\ \mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix} &= \begin{bmatrix}\mathbf{P}_S & \mathbf{P}_C & -\mathbf{P}_S\\ -\mathbf{P}_S & \mathbf{P}_S & \mathbf{P}_C\\ \mathbf{P}_C & -\mathbf{P}_S & \mathbf{P}_S\end{bmatrix}\begin{bmatrix}\mathbf{x}_0\\ \mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{P}_C & \mathbf{P}_S\\ \mathbf{P}_C & \mathbf{P}_S\\ \mathbf{P}_C & \mathbf{P}_S\end{bmatrix}\begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_0-\mathbf{x}_2\\ \mathbf{x}_2 & \mathbf{x}_1-\mathbf{x}_0\\ \mathbf{x}_0 & \mathbf{x}_2-\mathbf{x}_1\end{bmatrix},\end{aligned} \tag{6.21}$$

with

$$\mathbf{P}_C = \begin{bmatrix}p_0 & p_1 & p_2 & p_3\\ p_3 & p_0 & p_1 & p_2\\ p_2 & p_3 & p_0 & p_1\\ p_1 & p_2 & p_3 & p_0\end{bmatrix}, \quad \text{and} \quad \mathbf{P}_S = \begin{bmatrix}p_0 & p_1 & p_2 & p_3\\ -p_3 & p_0 & p_1 & p_2\\ -p_2 & -p_3 & p_0 & p_1\\ -p_1 & -p_2 & -p_3 & p_0\end{bmatrix}. \tag{6.22}$$

Implementing these operations using FFTs requires

- 2 FFTs of N_P points, for the sequence p_n,
- $2N_S$ FFTs of N_P points, for the combinations of the sequences $x_{i,n}$,
- $2N_S$ IFFTs of N_P points, for the sequences $y_{i,n}$,
- $2N_S$ products of N_P points, for the matrix-vector products,
- $2N_S + 1$ products of N_P points, for the skew-circular correlations,
- $N_S(N_S - 2)$ additions of N_P points, for the combinations of the sequences $x_{i,n}$,
- N_S additions of N_P points, for the additions to obtain the sequences $y_{i,n}$,

Therefore, the number of multiplications is

$$\begin{aligned} N_{mul} &= (4N_S+2)\left(\frac{N_P}{2}\log_2(N_P)\right) + (4N_S+1)\,N_P \\ &= N_P N_S\left(2\log_2(N_P) + 4 + \frac{\log_2(N_P)+1}{N_S}\right), \end{aligned} \tag{6.23}$$

and the number of additions is

$$\begin{aligned} N_{add} &= (4N_S+2)\left(N_P\log_2(N_P)\right) + N_S(N_S-1)N_P \\ &= N_P N_S\left(4\log_2(N_P) + N_S - 1 + \frac{2\log_2(N_P)}{N_S}\right). \end{aligned} \tag{6.24}$$

Considering $N_P = 20\,460$ and $N_S = 20$, this gives

$$\begin{aligned} N_{mul} &\approx 409\,200\,(32.64) + 313\,457.81 \approx 13\,670\,170 \\ N_{add} &\approx 409\,200\,(76.28) + 585\,995.62 \approx 31\,800\,620, \end{aligned} \tag{6.25}$$

which means an increase of 15.3 % for the multiplications and 39.0 % for the additions, compared to the direct implementation. Compared to the downsampling by a factor N_S, the number of FFTs is higher ($2+4N_S$ instead of $3N_S$), but the number of multiplications for the matrix-vector products is reduced a lot ($2N_S$ instead of N_S^2). This explains a lower increase for the multiplications, and a higher increase for the additions.

Here, it is possible to exploit the repetitions, but the fact to have at the same time circulant and skew-circulant matrices implies to double the number of FFTs.

Segmentation by a factor N_S using the equation with the matrix X

By segmenting the sequences by a factor N_S, each sequence is separated into N_S sub-sequences of N_P samples, and Eq. (6.5) can be expressed as

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} &= \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \end{bmatrix} \begin{bmatrix} s_0 p_0 \\ s_0 p_1 \\ s_0 p_2 \\ s_0 p_3 \end{bmatrix} + \begin{bmatrix} x_4 & x_5 & x_6 & x_7 \\ x_5 & x_6 & x_7 & x_8 \\ x_6 & x_7 & x_8 & x_9 \\ x_7 & x_8 & x_9 & x_{10} \end{bmatrix} \begin{bmatrix} s_1 p_0 \\ s_1 p_1 \\ s_1 p_2 \\ s_1 p_3 \end{bmatrix} + \begin{bmatrix} x_8 & x_9 & x_{10} & x_{11} \\ x_9 & x_{10} & x_{11} & x_0 \\ x_{10} & x_{11} & x_0 & x_1 \\ x_{11} & x_0 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} s_2 p_0 \\ s_2 p_1 \\ s_2 p_2 \\ s_2 p_3 \end{bmatrix} \\
\begin{bmatrix} y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} &= \begin{bmatrix} x_4 & x_5 & x_6 & x_7 \\ x_5 & x_6 & x_7 & x_8 \\ x_6 & x_7 & x_8 & x_9 \\ x_7 & x_8 & x_9 & x_{10} \end{bmatrix} \begin{bmatrix} s_0 p_0 \\ s_0 p_1 \\ s_0 p_2 \\ s_0 p_3 \end{bmatrix} + \begin{bmatrix} x_8 & x_9 & x_{10} & x_{11} \\ x_9 & x_{10} & x_{11} & x_0 \\ x_{10} & x_{11} & x_0 & x_1 \\ x_{11} & x_0 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} s_1 p_0 \\ s_1 p_1 \\ s_1 p_2 \\ s_1 p_3 \end{bmatrix} + \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \end{bmatrix} \begin{bmatrix} s_2 p_0 \\ s_2 p_1 \\ s_2 p_2 \\ s_2 p_3 \end{bmatrix} \\
\begin{bmatrix} y_8 \\ y_9 \\ y_{10} \\ y_{11} \end{bmatrix} &= \begin{bmatrix} x_8 & x_9 & x_{10} & x_{11} \\ x_9 & x_{10} & x_{11} & x_0 \\ x_{10} & x_{11} & x_0 & x_1 \\ x_{11} & x_0 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} s_0 p_0 \\ s_0 p_1 \\ s_0 p_2 \\ s_0 p_3 \end{bmatrix} + \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \end{bmatrix} \begin{bmatrix} s_1 p_0 \\ s_1 p_1 \\ s_1 p_2 \\ s_1 p_3 \end{bmatrix} + \begin{bmatrix} x_4 & x_5 & x_6 & x_7 \\ x_5 & x_6 & x_7 & x_8 \\ x_6 & x_7 & x_8 & x_9 \\ x_7 & x_8 & x_9 & x_{10} \end{bmatrix} \begin{bmatrix} s_2 p_0 \\ s_2 p_1 \\ s_2 p_2 \\ s_2 p_3 \end{bmatrix},
\end{aligned} \tag{6.26}$$

or in a more concise way

$$\begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} s_0\mathbf{X}_0 & s_1\mathbf{X}_1 & s_2\mathbf{X}_2 \\ s_0\mathbf{X}_1 & s_1\mathbf{X}_2 & s_2\mathbf{X}_0 \\ s_0\mathbf{X}_2 & s_1\mathbf{X}_0 & s_2\mathbf{X}_1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{p} \\ \mathbf{p} \end{bmatrix}, \tag{6.27}$$

where the matrices $\mathbf{X}_i$ are Hankel (see Appendix A.2.4). Hankel matrices can be embedded into circulant matrices of double size.

Thus, implementing these operations using FFTs requires

- 1 FFT of $2N_P$ points, for the sequence p_n,
- N_S FFTs of $2N_P$ points, for the sequences $x_{i,n}$,
- N_S IFFTs of $2N_P$ points, for the sequences $y_{i,n}$,
- N_S products of $2N_P$ points, for the matrix-vector products,
- $N_S(N_S-1)$ additions of N_P points, for the combinations of the results of the matrix-vector products.

Therefore, the number of multiplications is

$$\begin{aligned} N_{mul} &= (2N_S+1)\left(\frac{2N_P}{2}\log_2(2N_P)\right) + N_S N_P \\ &= N_P N_S\left(2\log_2(N_P) + 3 + \frac{\log_2(2N_P)}{N_S}\right), \end{aligned} \tag{6.28}$$

and the number of additions is

$$\begin{aligned} N_{add} &= (2N_S+1)\left(2N_P\log_2(2N_P)\right) + N_S(N_S-1)N_P \\ &= N_P N_S\left(4\log_2(N_P) + N_S + 3 + \frac{2\log_2(2N_P)}{N_S}\right). \end{aligned} \tag{6.29}$$

Considering $N_P = 20\,460$ and $N_S = 20$, this gives

$$\begin{aligned} N_{mul} &\approx 409\,200\,(31.64) + 313\,457.81 \approx 13\,260\,970 \\ N_{add} &\approx 409\,200\,(80.28) + 626\,915.62 \approx 33\,478\,340, \end{aligned} \tag{6.30}$$

which means an increase of 11.9 % for the multiplications and 46.3 % for the additions, compared to the traditional implementation.

Compared to the segmentation by a factor N_S using the matrix $\mathbf{H}$, the number of FFTs is divided by two but the FFT length is doubled (which requires slightly more operations), but there is not the additional product required by the skew-circular correlations. This explains the lower increase for the multiplications, and the higher increase for the additions.

Here, it is possible to exploit the repetitions, but the fact to have Hankel matrices implies to double the length of the FFTs.

Segmentation by a factor N_P using the equation with the matrix H

By segmenting the sequences by a factor N_P, each sequence is separated into N_P sub-sequences of N_S samples, and Eq. (6.4) can be expressed as

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} &= \begin{bmatrix} s_0p_0 & s_0p_1 & s_0p_2 \\ s_2p_3 & s_0p_0 & s_0p_1 \\ s_2p_2 & s_2p_3 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} s_0p_3 & s_1p_0 & s_1p_1 \\ s_0p_2 & s_0p_3 & s_1p_0 \\ s_0p_1 & s_0p_2 & s_0p_3 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} s_1p_2 & s_1p_3 & s_2p_0 \\ s_1p_1 & s_1p_2 & s_1p_3 \\ s_1p_0 & s_1p_1 & s_1p_2 \end{bmatrix} \begin{bmatrix} x_6 \\ x_7 \\ x_8 \end{bmatrix} + \begin{bmatrix} s_2p_1 & s_2p_2 & s_2p_3 \\ s_2p_0 & s_2p_1 & s_2p_2 \\ s_1p_3 & s_2p_0 & s_2p_1 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_3 \\ y_4 \\ y_5 \end{bmatrix} &= \begin{bmatrix} s_2p_1 & s_2p_2 & s_2p_3 \\ s_2p_0 & s_2p_1 & s_2p_2 \\ s_1p_3 & s_2p_0 & s_2p_1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_0p_1 & s_0p_2 \\ s_2p_3 & s_0p_0 & s_0p_1 \\ s_2p_2 & s_2p_3 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} s_0p_3 & s_1p_0 & s_1p_1 \\ s_0p_2 & s_0p_3 & s_1p_0 \\ s_0p_1 & s_0p_2 & s_0p_3 \end{bmatrix} \begin{bmatrix} x_6 \\ x_7 \\ x_8 \end{bmatrix} + \begin{bmatrix} s_1p_2 & s_1p_3 & s_2p_0 \\ s_1p_1 & s_1p_2 & s_1p_3 \\ s_1p_0 & s_1p_1 & s_1p_2 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_6 \\ y_7 \\ y_8 \end{bmatrix} &= \begin{bmatrix} s_1p_2 & s_1p_3 & s_2p_0 \\ s_1p_1 & s_1p_2 & s_1p_3 \\ s_1p_0 & s_1p_1 & s_1p_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} s_2p_1 & s_2p_2 & s_2p_3 \\ s_2p_0 & s_2p_1 & s_2p_2 \\ s_1p_3 & s_2p_0 & s_2p_1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_0p_1 & s_0p_2 \\ s_2p_3 & s_0p_0 & s_0p_1 \\ s_2p_2 & s_2p_3 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_6 \\ x_7 \\ x_8 \end{bmatrix} + \begin{bmatrix} s_0p_3 & s_1p_0 & s_1p_1 \\ s_0p_2 & s_0p_3 & s_1p_0 \\ s_0p_1 & s_0p_2 & s_0p_3 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \\ x_{11} \end{bmatrix} \\
\begin{bmatrix} y_9 \\ y_{10} \\ y_{11} \end{bmatrix} &= \begin{bmatrix} s_0p_3 & s_1p_0 & s_1p_1 \\ s_0p_2 & s_0p_3 & s_1p_0 \\ s_0p_1 & s_0p_2 & s_0p_3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} s_1p_2 & s_1p_3 & s_2p_0 \\ s_1p_1 & s_1p_2 & s_1p_3 \\ s_1p_0 & s_1p_1 & s_1p_2 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} s_2p_1 & s_2p_2 & s_2p_3 \\ s_2p_0 & s_2p_1 & s_2p_2 \\ s_1p_3 & s_2p_0 & s_2p_1 \end{bmatrix} \begin{bmatrix} x_6 \\ x_7 \\ x_8 \end{bmatrix} + \begin{bmatrix} s_0p_0 & s_0p_1 & s_0p_2 \\ s_2p_3 & s_0p_0 & s_0p_1 \\ s_2p_2 & s_2p_3 & s_0p_0 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \\ x_{11} \end{bmatrix},
\end{aligned} \tag{6.31}$$

or in a more concise way

$$\begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 \\ \mathbf{H}_3 & \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 \\ \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{H}_0 & \mathbf{H}_1 \\ \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{H}_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}, \tag{6.32}$$

where the matrices $\mathbf{H}_i$ are Toeplitz (see Appendix A.2.3). Toeplitz matrices can be changed to circulant matrices by doubling their size. Thus, implementing these operations using FFTs requires

- N_P FFTs of $2N_S$ points, for the sequence $h_{i,n}$,
- N_P FFTs of $2N_S$ points, for the sequences $x_{i,n}$,
- N_P IFFTs of $2N_S$ points, for the sequences $y_{i,n}$,
- N_P products of $2N_S$ points, for the matrix-vector products,
- $N_P(N_P-1)$ additions of N_S points, for the combinations of the results of the matrix-vector products.

Therefore, the number of multiplications is

$$\begin{aligned}
N_{mul} &= 3N_P\left(N_S\log_2(2N_S)\right) + N_PN_S \\
&= N_PN_S\left(3\log_2(2N_S)+1\right),
\end{aligned} \tag{6.33}$$

and the number of additions is

$$\begin{aligned} N_{add} &= 3N_P\left(2N_S\log_2(2N_S)\right) + N_P(N_P-1)N_S \\ &= N_P N_S\left(6\log_2(2N_S) + N_P - 1\right). \end{aligned} \tag{6.34}$$

Considering $N_P = 20\,460$ and $N_S = 20$, this gives

$$\begin{aligned} N_{mul} &\approx 409\,200\,(16.97) \approx 6\,942\,399 \\ N_{add} &\approx 409\,200\,(20\,490.93) \approx 8\,384\,889\,198, \end{aligned} \tag{6.35}$$

which means a reduction of 41.4 % for the multiplications but an increase of 36 538.5 % for the additions, compared to the direct implementation.

Compared to the downsampling by a factor N_P, the number of FFTs is higher and the FFT length is doubled, which explains a higher number of the multiplications. The large increase for the additions is still due to the $N_P(N_P-1)$ additions of the matrix-vector products to compute.

Here, each matrix $\mathbf{H}_i$ contains all the secondary code chips, thus we cannot exploit the repetitions in the tiered code. This means that the same result would be obtained with any signal.

Segmentation by a factor N_P using the equation with the matrix X

Doing the same separation as previously using Eq. (6.5) leads a similar implementation with the same complexity, therefore we do not give the details.

6.3.3 Summary

Table 6.1 provides a summary of the different implementations according to the separation (downsampling or segmentation), the factor (N_S or N_P) and the equation used (Eq. 6.4 with **H** as matrix or Eq. 6.5 with **X** as matrix). It can be seen that using a factor N_P is not efficient because of the prohibitive number of additions, even if we can exploit the repetition of the primary code.

The most efficient implementations (downsampling and segmentation by a factor N_S) have similar performance. However, for a hardware implementation the implementation obtained by segmentation is more interesting. Indeed, for each section of the output, there is only N_S products to perform between the FFTs and the storage of intermediate value is relatively small (the result of one product), while with the implementation obtain by downsampling, there are N_S^2 product and the storage is much higher, as already shown for the case $N_S = 2$ in Section 4.3.6.

Factor		N_S		N_P	
Matrix used		**H**	**X**	**H**	**X**
Separation by downsampling	Matrices obtained	Circulant	Circulant	Circulant	Circulant
	Exploitation of the repetition	No	No	Yes	Yes
	Complexity	+46.5 % +10.8 %	+46.5 % +10.8 %	−78.2 % +36 496.8 %	−78.2 % +36 496.8 %
Separation by segmentation	Matrices obtained	Circulant and skew-circulant	Hankel	Toeplitz	Toeplitz
	Exploitation of the repetition	Yes	Yes	No	No
	Complexity	+15.3 % +39.0 %	+11.9 % +46.3 %	−41.4 % 36 538.5 %	−41.4 % 36 538.5 %

Table 6.1: Summary of the implementations according to the separation and separation factor. The complexity is given in comparison to the direct implementation (Fig. 6.1), for the number of multiplications and additions.

6.3.4 Relation to Chapter 5

In the separation by segmentation with a factor N_S and the matrix **X**, the small matrices obtained were Hankel. As indicated, such matrices can be embedded into circulant matrices. Below, we detail this operation. In Eq. (6.27), we have

$$\mathbf{X}_0 = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \end{bmatrix}, \tag{6.36}$$

an $N_P \times N_P$ matrix. This Hankel matrix can be embedded into the $2N_P - 1 \times 2N_P - 1$ circulant matrix

$$\mathbf{X}_{C0} = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_0 \\ x_2 & x_3 & x_4 & x_5 & x_6 & x_0 & x_1 \\ x_3 & x_4 & x_5 & x_6 & x_0 & x_1 & x_2 \\ x_4 & x_5 & x_6 & x_0 & x_1 & x_2 & x_3 \\ x_5 & x_6 & x_0 & x_1 & x_2 & x_3 & x_4 \\ x_6 & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}. \tag{6.37}$$

Thus, the product between the matrix $\mathbf{X}_0$ and the vector $\mathbf{p}$ can be obtained as

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_0 \\ x_2 & x_3 & x_4 & x_5 & x_6 & x_0 & x_1 \\ x_3 & x_4 & x_5 & x_6 & x_0 & x_1 & x_2 \\ x_4 & x_5 & x_6 & x_0 & x_1 & x_2 & x_3 \\ x_5 & x_6 & x_0 & x_1 & x_2 & x_3 & x_4 \\ x_6 & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \\ x_4 & x_5 & x_6 & x_0 \\ x_5 & x_6 & x_0 & x_1 \\ x_6 & x_0 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{6.38}$$

$$\mathbf{X}_{C0}\,\mathbf{p}_Z = \begin{bmatrix} \mathbf{X}_0 \\ \mathbf{X}_D \end{bmatrix} \mathbf{p},$$

where $\mathbf{p}_Z$ is $\mathbf{p}$ padded with $N_P - 1$ zeros, by keeping only the first N_P samples of the product. This matrix-vector product can be reformulated to have the samples of x_n in the vector, i.e.

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_0 \\ x_2 & x_3 & x_4 & x_5 & x_6 & x_0 & x_1 \\ x_3 & x_4 & x_5 & x_6 & x_0 & x_1 & x_2 \\ x_4 & x_5 & x_6 & x_0 & x_1 & x_2 & x_3 \\ x_5 & x_6 & x_0 & x_1 & x_2 & x_3 & x_4 \\ x_6 & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 & 0 & 0 & 0 \\ 0 & p_0 & p_1 & p_2 & p_3 & 0 & 0 \\ 0 & 0 & p_0 & p_1 & p_2 & p_3 & 0 \\ 0 & 0 & 0 & p_0 & p_1 & p_2 & p_3 \\ p_3 & 0 & 0 & 0 & p_0 & p_1 & p_2 \\ p_2 & p_3 & 0 & 0 & 0 & p_0 & p_1 \\ p_1 & p_2 & p_3 & 0 & 0 & 0 & p_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}. \tag{6.39}$$

Such a matrix-vector product has already been encountered in Chapter 5 (see Eq. (5.2)), where the aim was to perform a circular correlation over one primary code avoiding any loss due to a transition caused by the secondary code. Therefore, the algorithms proposed in Chapter 5 to reduce the complexity can be applied for the circular correlation over a full period of the tiered code, using a segmentation by a factor N_S when $\mathbf{X}$ is the matrix.

6.4 Implementations to reduce the complexity

In this section, we present alternative implementations of the correlation where the aim is to use the specificities of the secondary code to reduce the complexity. The idea is that since the secondary code is binary, adding or subtracting chips will lead to 0, +2 or −2. And maybe it is possible to obtain some sub-sequences with only zeros, which could reduce the complexity.

6.4.1 Computation of the correlation using the Chinese remainder theorem

Computation using two sub-correlations

As already presented in Section 4.3.3, the circular correlation can be computed as depicted in Fig. 6.2 using the CRT. In this case, the inputs of the circular and skew-circular correlations

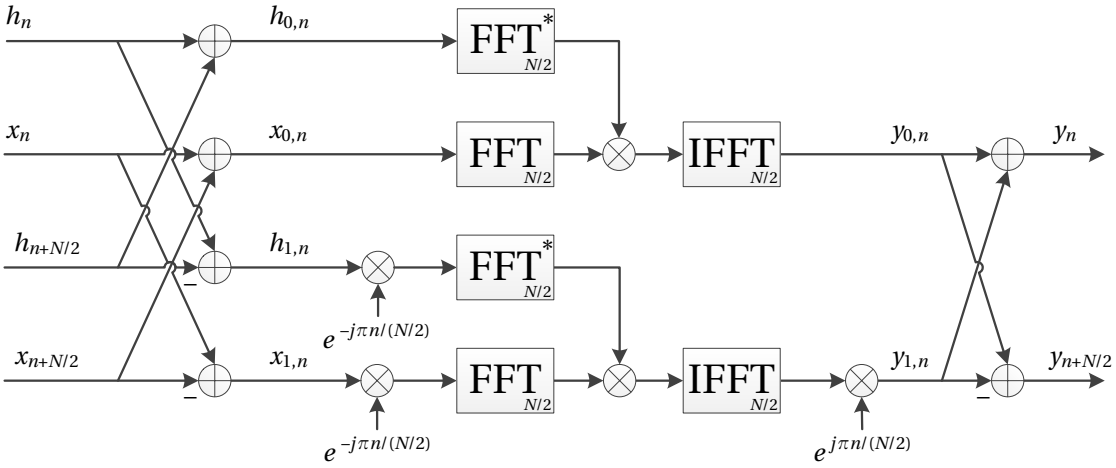


Figure 6.2: Computation of a circular correlation of N points using $N/2$-point FFTs (algorithm based on the Chinese remainder theorem).

are the sum and the difference of the first and second half of the input sequences, i.e. $h_{0,n} = h_n + h_{n+N/2}$ and $h_{1,n} = h_n - h_{n+N/2}$. If h_n is a tiered code, from Eq. (6.3) we have

$$\begin{aligned}\mathbf{h}_0 &= \left[(s_0 + s_{N_S/2})\mathbf{p} \quad (s_1 + s_{N_S/2+1})\mathbf{p} \quad \ldots \quad (s_{N_S/2-1} + s_{N_S-1})\mathbf{p}\right] \\ &= \left[s_{0,0}\mathbf{p} \quad s_{0,1}\mathbf{p} \quad \ldots \quad s_{0,N_S/2-1}\mathbf{p}\right] \\ &= \mathbf{s}_0 \otimes \mathbf{p},\end{aligned} \tag{6.40}$$

and

$$\begin{aligned}\mathbf{h}_1 &= \left[(s_0 - s_{N_S/2})\mathbf{p} \quad (s_1 - s_{N_S/2+1})\mathbf{p} \quad \ldots \quad (s_{N_S/2-1} - s_{N_S-1})\mathbf{p}\right] \\ &= \left[s_{1,0}\mathbf{p} \quad s_{1,1}\mathbf{p} \quad \ldots \quad s_{1,N_S/2-1}\mathbf{p}\right] \\ &= \mathbf{s}_1 \otimes \mathbf{p}.\end{aligned} \tag{6.41}$$

Since the chips of the secondary code s_n can take as value +1 or −1, the chips of the sub-sequences $s_{0,n}$ and $s_{1,n}$ can take as value +2, 0 or −2. Therefore, about half of the samples of $h_{0,n}$ and $h_{1,n}$ may be zeros (since there are two possibilities to obtain 0, but only one to obtain +2 or −2).

Computation using one sub-correlation and two sub-sub-correlations

Since the top part in Fig. 6.2 with the 3 FFTs corresponds also to a circular correlation, we can apply the CRT once again to obtain Fig. 6.3, where $h_{00,n} = h_{0,n} + h_{0,n+N/4}$ and $h_{01,n} =$

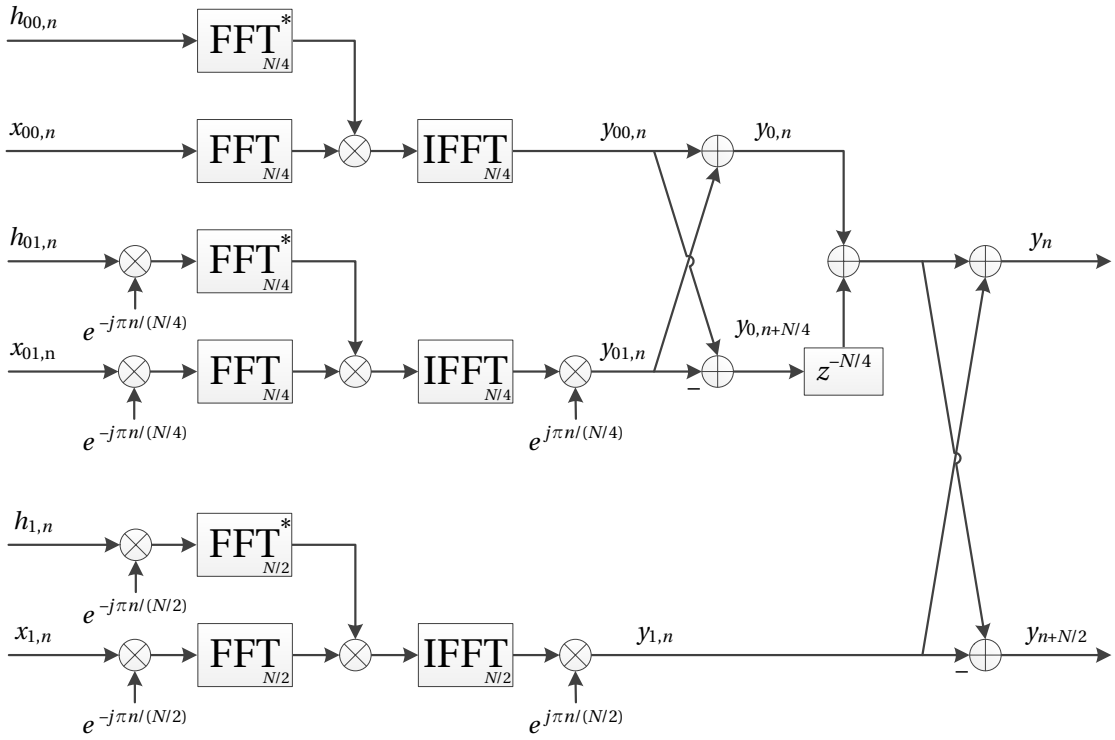


Figure 6.3: Computation of a circular correlation of N points using $N/2$-point and $N/4$-point FFTs.

$h_{0,n} - h_{0,n+N/4}$. If h_n is a tiered code, Eq. (6.40) we have

$$\begin{aligned}\mathbf{h}_{00} &= \left[(s_{0,0} + s_{0,N_S/4})\mathbf{p} \quad (s_{0,1} + s_{0,N_S/4+1})\mathbf{p} \quad \ldots \quad (s_{0,N_S/4-1} + s_{0,N_S/2-1})\mathbf{p}\right] \\ &= \left[s_{00,0}\mathbf{p} \quad s_{00,1}\mathbf{p} \quad \ldots \quad s_{00,N_S/4-1}\mathbf{p}\right] \\ &= \mathbf{s}_{00} \otimes \mathbf{p},\end{aligned} \tag{6.42}$$

and

$$\begin{aligned}\mathbf{h}_{01} &= \left[(s_{0,0} - s_{0,N_S/4})\mathbf{p} \quad (s_{0,1} - s_{0,N_S/4+1})\mathbf{p} \quad \ldots \quad (s_{0,N_S/4-1} - s_{0,N_S/2-1})\mathbf{p}\right] \\ &= \left[s_{01,0}\mathbf{p} \quad s_{01,1}\mathbf{p} \quad \ldots \quad s_{01,N_S/4-1}\mathbf{p}\right] \\ &= \mathbf{s}_{01} \otimes \mathbf{p}.\end{aligned} \tag{6.43}$$

Since the chips of the sub-sequences $s_{0,n}$ and $s_{1,n}$ can take as value +2, 0 or −2, the chips of the sub-sequences $s_{00,n}$ and $s_{01,n}$ can take as value +4, +2, 0, −2 or −4.

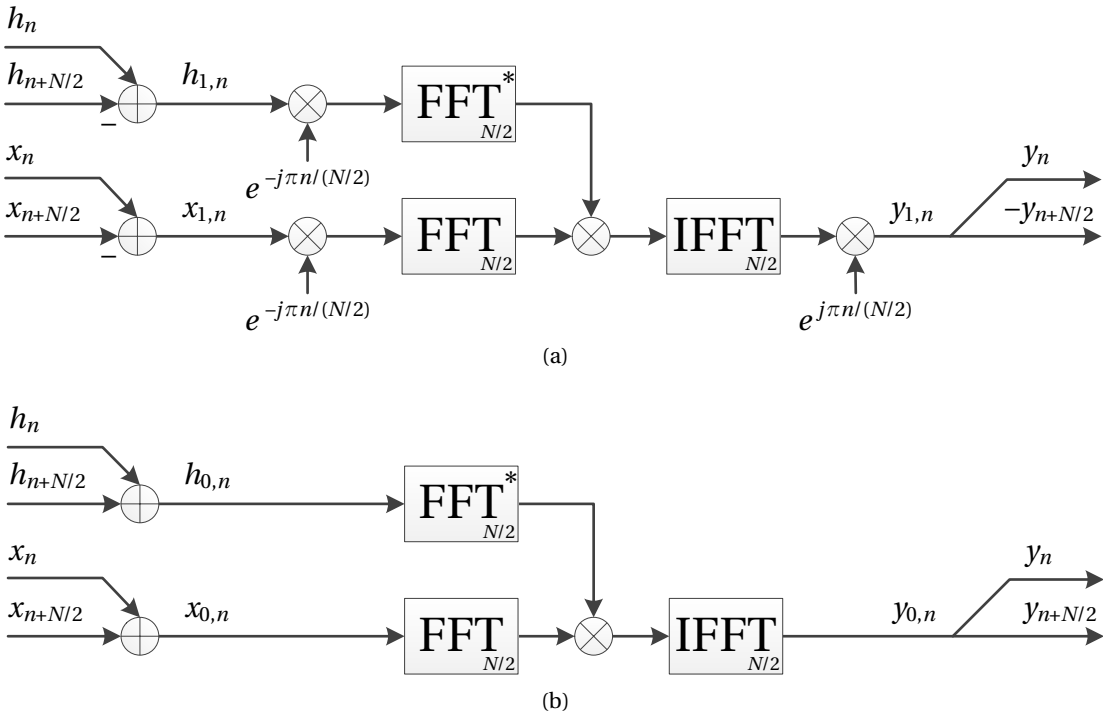


Figure 6.4: Implementation of Fig. 6.2, (a) when $h_{0,n}$ is null, (b) when $h_{1,n}$ is null.

Application to reduce the complexity of the correlation with tiered codes

If we can find a sub-code $s_{0,n}$ that contains only zeros (as shown later in Section 6.4.2), the corresponding tiered code $h_{0,n}$ would also contain only zeros according to Eq. 6.3. Therefore, in this case it would not be necessary to compute the FFTs of the circular correlation in Fig. 6.2, and the implementation would reduce to Fig. 6.4a. In the same way, if the sub-code $s_{1,n}$ contains only zeros, $h_{1,n}$ will also contain only zeros, and Fig. 6.2 becomes 6.4b. In both cases, the number of operations is approximately reduced by 50 %. It can be seen that in this case the two halves of the output y_n are either identical or opposite, which is normal because having $s_{0,n}$ or $s_{1,n}$ null means that the two halves of s_n are opposite or identical, respectively.

In the same way, if the sub-codes $s_{00,n}$ or $s_{01,n}$ contains only zeros, $h_{00,n}$ or $h_{01,n}$ will also contain only zeros, and Fig. 6.3 becomes 6.5a or 6.5b. In both cases, the number of operations is approximately divided by 25 %.

If the sub-codes $s_{0,n}$ or $s_{00,n}$ are constant, $h_{0,n}$ or $h_{00,n}$ will contain the primary code repeated several times. The FFT of a sequence repeated $P-1$ times is the same as the FFT of this sequence with $P-1$ zeros added between two samples. Therefore, it is possible to compute an N_P-point FFT instead of an FFT of $N/2 = N_P N_S/2$ points if $s_{0,n}$ is constant, or instead of an FFT of $N/4 = N_P N_S/4$ points if $s_{00,n}$ is constant. In this case, the reduction of the complexity is respectively of about 14 % and 5 %, with $N_S = 20$.

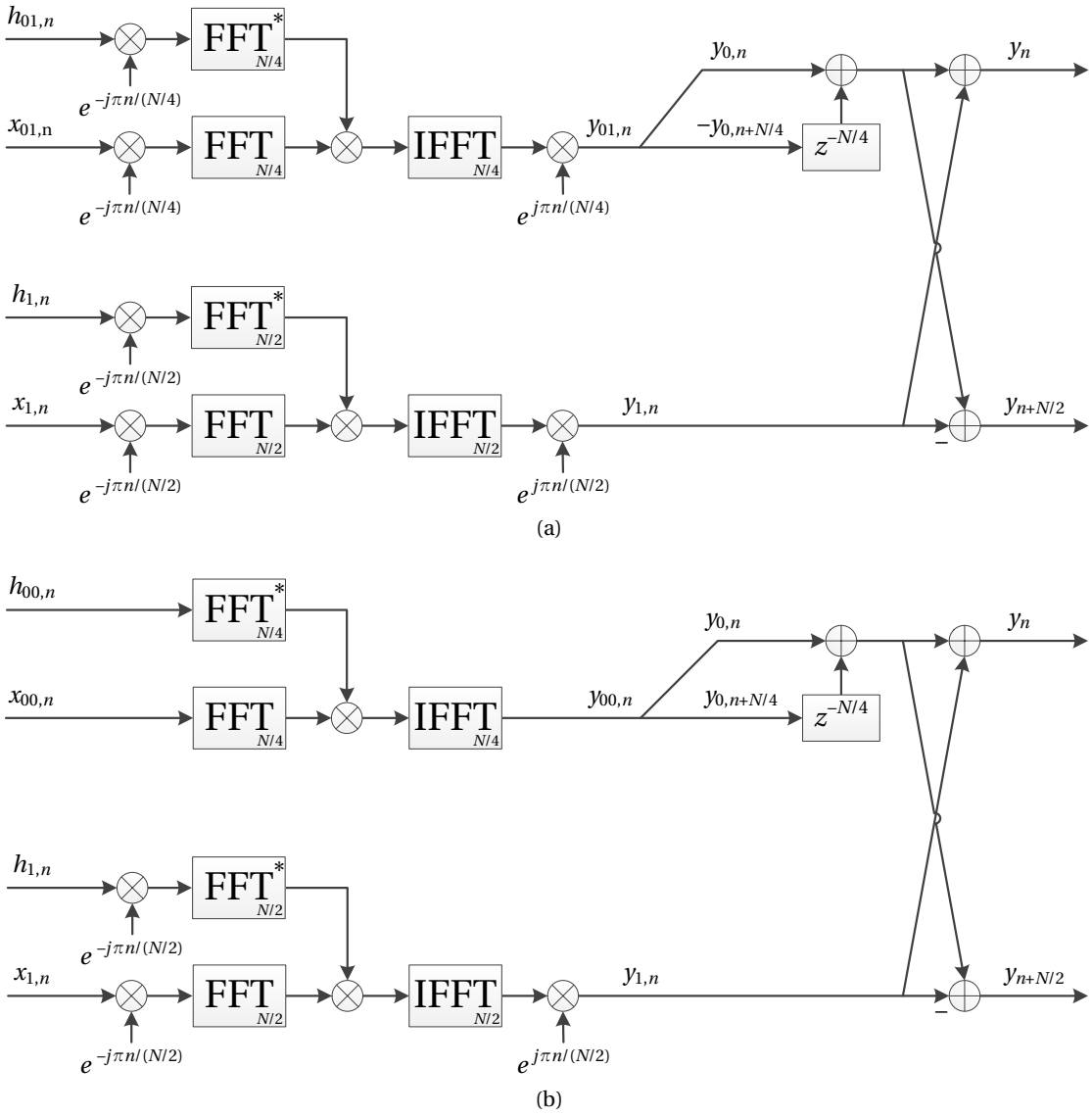


Figure 6.5: Implementation of Fig. 6.3, (a) when $h_{00,n}$ is null, (b) when $h_{01,n}$ is null.

6.4.2 Results with the GPS L5 signal

Sub-codes

For the GPS L5 pilot signal, we have $N_S = 20$, and N_P depends on the sampling frequency. The L5 secondary code s_n and its sub-codes are given Table 6.2. It can be seen that none of these sequences is null or constant. The sequences $s_{0,n}$, $s_{1,n}$ and $s_{00,n}$ are invariant by translation of s_n, but not $s_{01,n}$. However, even for the other delays of s_n, $s_{01,n}$ is not null or constant.

The idea is then to use a different code for the local secondary code, in order to get null sub-codes. Of course, this will have an impact on the SNR and on the protection against

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
s_n	1	1	1	1	1	−1	1	1	−1	−1	1	−1	1	−1	1	1	−1	−1	−1	1
$s_{0,n}$	2	0	2	0	2	0	0	0	−2	0										
$s_{1,n}$	0	2	0	2	0	−2	2	2	0	−2										
$s_{00,n}$	2	0	2	−2	2															
$s_{01,n}$	2	0	2	2	2															

Table 6.2: Secondary code of the GPS L5 pilot signal, and its sub-codes.

noise and cross-interference with other GNSS signals. In order to find a code having a sub-code null and the smallest impact on the SNR, we performed an exhaustive search. For this, we have generated all the possible secondary codes s_n, where each chip can take a value among $\{-2,-1.5,-1,-0.5,0,+0.5,+1,+1.5,+2\}$, and for each of them we have computed the protection and the loss of SNR. The protection is determined easily since it is the ratio between the first maximum and the second maximum of the correlation. For the SNR loss, it is slightly more complicated and we detail below its computation.

Determination of the loss

Let's consider that after the accumulation over one primary code period, the amplitude of the signal is a, and the noise has a variance σ^2. Thus, the SNR is

$$SNR_P = 10\log_{10}\left(\frac{a^2}{\sigma^2}\right). \tag{6.44}$$

Then, during the correlation with the secondary code, we compute N_S products and we accumulate the results. The variance of the noise after the correlation with the secondary code is then

$$Var(\eta_T) = \sigma^2 \sum_{n=0}^{N_S-1} s_n^2, \qquad \text{with } s_n \in \{-2,-1.5,-1,-0.5,0,+0.5,+1,+1.5,+2\}. \tag{6.45}$$

The SNR after the correlation with the secondary code is then

$$\begin{aligned} SNR_T &= 10\log_{10}\left(\frac{a^2 corr_{max}^2}{\sigma^2 \sum_{n=0}^{N_S-1} s_n^2}\right) \\ &= SNR_P + 10\log_{10}\left(\frac{corr_{max}^2}{\sum_{n=0}^{N_S-1} s_n^2}\right), \end{aligned} \tag{6.46}$$

where $corr_{max}$ is the maximum of the correlation of the received and local secondary code. If the local code is identical to the received code, $corr_{max} = N_S$. For the L5 signal, the secondary code is binary, thus $s_n^2 = 1$ and $\sum_{n=0}^{N_S-1} s_n^2 = N_S$. Therefore, if the local code is the L5 secondary

code, the SNR after the correlation with the secondary code is maximum and is equal to

$$\begin{aligned} SNR_{T,max} &= SNR_P + 10\log_{10}\left(\frac{N_S^2}{N_S}\right) \\ &= SNR_P + 10\log_{10}(N_S). \end{aligned} \tag{6.47}$$

Consequently, using a modified code implies a loss defined as

$$\begin{aligned} loss &= SNR_{T,max} - SNR_T \\ &= 10\log_{10}(N_S) - 10\log_{10}\left(\frac{corr_{max}^2}{\sum_{n=0}^{N_S-1} s_n^2}\right). \end{aligned} \tag{6.48}$$

Reduction of the complexity by 1/2

As indicated previously, if $s_{0,n}$ or $s_{1,n}$ is null, the complexity is reduced by about 50 %. We performed an exhaustive search considering all the codes giving $s_{0,n} = 0$ or $s_{1,n} = 0$, i.e. codes where $s_n = s_{n+N_S/2}$ or $s_n = -s_{n+N_S/2}$. Therefore, the number of possible codes for each case is $9^{10} = 3\,486\,784\,401$.

Table 6.3 shows the minimum loss that we can obtain to have $s_{0,n}$ null when receiving the L5 secondary code, according to the first and the second maximum of the correlation. The best case that we can obtain is a loss of about 2.22 dB. This can be obtained with several codes, one of them is given Table 6.4. It can be noticed that such code uses only three levels (+1, 0, and −1). The auto and cross-correlation of the L5 secondary code with this local code is illustrated Fig. 6.6. It can be seen that the maximum of the cross-correlation is 12 instead of 20 for the auto-correlation of the L5 code, which is expected since the local code contains 8 zeros. Therefore this is equivalent to have an integration time of 12 ms instead of 20 ms (hence a loss of $10\log_{10}\left(\frac{20}{12}\right) \approx 2.22$ dB). However, it can be noticed that the protection is increased to $20\log_{10}\left(\frac{12}{2}\right) \approx 15.56$ dB, instead of $20\log_{10}\left(\frac{20}{5}\right) \approx 13.98$ dB for the autocorrelation of the L5 secondary code.

The same search has been performed to have $s_{1,n}$ null, but the results are less good (in the best case the minimum loss we can obtain is about 3.98 dB).

Reduction of the complexity by 1/4

In this case, we performed an exhaustive search considering all the codes giving $s_{00,n} = 0$ or $s_{01,n} = 0$. Since the constraints are less strong than before, we could have expected better results. However, the minimum loss that we can obtain is still about 2.22 dB.

1st max \ 2nd max	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	10.00	6.99	5.23	3.98	3.01	2.22	2.64	2.73	2.68		2.79		2.35	2.39	2.61	2.55								
2		6.99	5.23	3.98	3.01	2.89	2.64	2.73	2.68	2.55	2.39	2.22	2.35	2.39	2.39	2.55	2.47	2.38	2.28	2.30				
3			5.23	3.98	3.80	3.47	3.10	2.73	2.68	2.55	2.39	2.40	2.35	2.39	2.39	2.35	2.29	2.22	2.28	2.30	2.31	2.39	2.45	
4				4.95	4.47	3.98	3.51	3.08	2.68	2.55	2.60	2.57	2.49	2.39	2.39	2.35	2.29	2.30	2.28	2.30	2.31	2.29	2.26	2.22
5					5.05	4.44	3.89	3.40	2.96	2.79	2.79	2.73	2.63	2.52	2.39	2.35	2.38	2.38	2.35	2.30	2.31	2.29	2.26	2.26
6						4.85	4.24	3.70	3.47	3.22	2.97	2.89	2.77	2.64	2.50	2.45	2.47	2.45	2.42	2.37	2.31	2.29	2.31	2.40
7							4.56	4.24	3.93	3.62	3.32	3.04	2.91	2.76	2.71	2.64	2.55	2.53	2.49	2.43	2.36	2.34	2.45	2.61
8								4.73	4.34	3.98	3.64	3.33	3.04	2.99	2.91	2.82	2.71	2.60	2.55	2.49	2.48	2.50	2.68	2.89
9									4.72	4.31	3.94	3.60	3.40	3.21	3.11	2.99	2.87	2.75	2.62	2.61	2.64	2.74	2.98	3.08
10										4.62	4.22	3.98	3.74	3.51	3.29	3.16	3.03	2.89	2.81	2.79	2.90	3.06	3.18	3.26
11											4.61	4.33	4.06	3.80	3.55	3.32	3.17	3.09	3.06	3.06	3.24	3.36	3.37	3.43
12												4.65	4.35	4.07	3.80	3.55	3.38	3.35	3.35	3.42	3.64	3.57	3.56	3.60
13													4.62	4.32	4.04	3.84	3.72	3.65	3.72	3.98	3.85	3.76	3.73	3.83
14														4.56	4.33	4.18	4.08	4.03	4.34	4.19	4.05	3.94	3.97	4.04
15															4.67	4.55	4.48	4.72	4.55	4.39	4.24	4.19	4.20	4.24
16																4.95	5.17	4.98	4.80	4.62	4.49	4.42	4.41	4.44
17																	5.48	5.27	5.07	4.88	4.73	4.64	4.61	
18																		5.54	5.32	5.12	4.95	4.85		
19																			5.56	5.35	5.17			
20																				5.56				

Table 6.3: Minimum loss in dB when receiving the L5 secondary code and using a local code s_n having its sub-sequence $s_{0,n}$ null.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
s_n	1	1	0	−1	0	−1	0	−1	0	1	−1	−1	0	1	0	1	0	1	0	−1
$s_{0,n}$	0	0	0	0	0	0	0	0	0	0										
$s_{1,n}$	2	2	0	−2	0	−2	0	−2	0	2										

Table 6.4: Example of code where the sub-code $s_{0,n}$ is null, and giving a loss of 2.22 dB.

Reduction of the complexity by 1/3

As indicated previously, having $s_{0,n}$ or $s_{1,n}$ equal to zeros means that the first half and the second half of s_n are identical or opposite. In order to allow new possibilities, we now consider a local code that is the sum of two codes. One code that gives $s_{0,n}$ or $s_{1,n}$ null as before, and another code where its half are not identical or opposite, but whose the correlation with can be computed in a efficient way. This code and its sub-codes are given Table 6.5. It can be seen the values of $s_{0,n}$ alternate between 0 and 1, and the values of $s_{1,n}$ also alternate between 0 and 1 or −1.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
s_n	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
$s_{0,n}$	0	1	0	1	0	1	0	1	0	1										
$s_{1,n}$	0	−1	0	1	0	−1	0	1	0	−1										

Table 6.5: Special code.

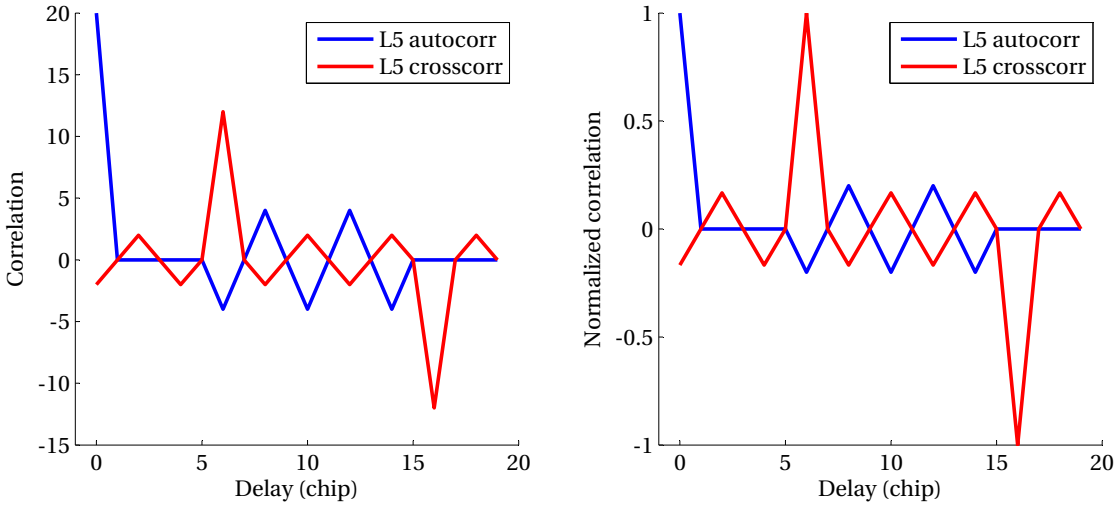


Figure 6.6: Auto and cross-correlation of the L5 secondary code with the local secondary code of Table 6.4.

If we write the circular correlation with the special sub-code $s_{0,n}$ in matrix form, and taking for example $N_P = 3$ and $s_{0,n} = [0\ 1\ 0\ 1\ 0\ 1]$ (this is simply to be able to display the matrix), we have

$$\begin{bmatrix} y_{0,0} \\ y_{0,1} \\ y_{0,2} \\ y_{0,3} \\ y_{0,4} \\ y_{0,5} \\ y_{0,6} \\ y_{0,7} \\ y_{0,8} \\ y_{0,9} \\ y_{0,10} \\ y_{0,11} \\ y_{0,12} \\ y_{0,13} \\ y_{0,14} \\ y_{0,15} \\ y_{0,16} \\ y_{0,17} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 \\ p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 \\ p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 \\ p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 \\ 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 \\ 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 \\ 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 \\ p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 \\ p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 \\ p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 \\ 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 \\ 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 \\ 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 \\ p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 \\ p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 \\ p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 \\ 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 \\ 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 & 0 & 0 & p_0 & p_1 & p_2 & 0 \end{bmatrix} \begin{bmatrix} x_{0,0} \\ x_{0,1} \\ x_{0,2} \\ x_{0,3} \\ x_{0,4} \\ x_{0,5} \\ x_{0,6} \\ x_{0,7} \\ x_{0,8} \\ x_{0,9} \\ x_{0,10} \\ x_{0,11} \\ x_{0,12} \\ x_{0,13} \\ x_{0,14} \\ x_{0,15} \\ x_{0,16} \\ x_{0,17} \end{bmatrix} \tag{6.49}$$

$$\begin{bmatrix} \mathbf{y}_{0,0} \\ \mathbf{y}_{0,1} \\ \mathbf{y}_{0,2} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_C & \mathbf{P}_C & \mathbf{P}_C \\ \mathbf{P}_C & \mathbf{P}_C & \mathbf{P}_C \\ \mathbf{P}_C & \mathbf{P}_C & \mathbf{P}_C \end{bmatrix} \begin{bmatrix} \mathbf{x}_{0,0} \\ \mathbf{x}_{0,1} \\ \mathbf{x}_{0,2} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{P}_C \\ \mathbf{P}_C \\ \mathbf{P}_C \end{bmatrix} \begin{bmatrix} \mathbf{x}_{0,0} + \mathbf{x}_{0,1} + \mathbf{x}_{0,2} \\ \mathbf{x}_{0,0} + \mathbf{x}_{0,1} + \mathbf{x}_{0,2} \\ \mathbf{x}_{0,0} + \mathbf{x}_{0,1} + \mathbf{x}_{0,2} \end{bmatrix},$$

where $\mathbf{P}_C$ is a 6×6 circulant matrix. It can be seen that each section of the output is the same, and that only one matrix-vector product is needed. Therefore, instead of performing a circular correlation of $N = N_P N_S$ points, we can sum sections of the incoming signal and compute a circular correlation of $2N_P$ points.

If we write the skew-circular correlation with the special sub-code $s_{1,n}$ in matrix form, and

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
s_n	1.5	0.5	1.5	−0.5	1.5	0.5	−1.5	1	1.5	0.5	−1.5	−0.5	−1.5	0.5	−1.5	−0.5	1.5	−1	−1.5	−0.5
$s_{0,n}$	0	0	0	0	0	0	0	0	0	0										
$s_{1,n}$	3	1	3	−1	3	1	−3	2	3	1										

Table 6.6: Example of code where the sub-code $s_{0,n}$ is null, and giving a loss of 1.69 dB using the special code.

taking as example $N_P = 3$ and $s_{1,n} = [0\ -1\ 0\ 1\ 0\ -1]$, we have

$$\begin{aligned}\begin{bmatrix}\mathbf{y}_{1,0}\\ \mathbf{y}_{1,1}\\ \mathbf{y}_{1,2}\end{bmatrix} &= \begin{bmatrix}\mathbf{P}_S & -\mathbf{P}_S & \mathbf{P}_S\\ -\mathbf{P}_S & \mathbf{P}_S & -\mathbf{P}_S\\ \mathbf{P}_S & -\mathbf{P}_S & \mathbf{P}_S\end{bmatrix}\begin{bmatrix}\mathbf{x}_{1,0}\\ \mathbf{x}_{1,1}\\ \mathbf{x}_{1,2}\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{P}_S\\ -\mathbf{P}_S\\ \mathbf{P}_S\end{bmatrix}\begin{bmatrix}\mathbf{x}_{1,0}-\mathbf{x}_{1,1}+\mathbf{x}_{1,2}\\ \mathbf{x}_{1,0}-\mathbf{x}_{1,1}+\mathbf{x}_{1,2}\\ \mathbf{x}_{1,0}-\mathbf{x}_{1,1}+\mathbf{x}_{1,2}\end{bmatrix},\end{aligned} \tag{6.50}$$

where $\mathbf{P}_S$ is the following 6×6 skew-circulant matrix,

$$\mathbf{P}_S = \begin{bmatrix}0 & 0 & 0 & -p_0 & -p_1 & -p_2\\ p_2 & 0 & 0 & 0 & -p_0 & -p_1\\ p_1 & p_2 & 0 & 0 & 0 & -p_0\\ p_0 & p_1 & p_2 & 0 & 0 & 0\\ 0 & p_0 & p_1 & p_2 & 0 & 0\\ 0 & 0 & p_0 & p_1 & p_2 & 0\end{bmatrix}. \tag{6.51}$$

It can be seen that each section of the output is the same (or the sign is changed), and that only one matrix-vector product is needed. Therefore, instead of performing a skew-circular correlation of $N = N_P N_S$ points, we can sum and subtract sections of the incoming signal and compute a skew-circular correlation of $2N_P$ points.

Therefore, the circular correlation using this code can be computed as in Fig. 6.2 except that the FFT length is $2N_P$ instead of $N/2$, and the input for the incoming signal requires additional additions and subtractions. With this additional computations, the complexity is reduced by about 33 %.

Then, in the same way as before, we performed an exhaustive search considering all the codes giving $s_{00,n} = 0$ or $s_{01,n} = 0$. Now the minimum loss that can be obtained is about 1.69 dB, with a protection of 12.88 dB. This can be obtained with several codes, one of them is given Table 6.6. If we want no decrease of the protection, it is also possible to get a loss of 1.75 dB for a protection of 14.40 dB.

2nd max	4	8	12	16	20
Number of codes	4565	16 042	4924	632	109

Table 6.7: Magnitude of the second maximum of the autocorrelation of unique codes of 20 bits and repartition (the first maximum being 20 when the code is aligned with itself).

Number of peaks	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Number of codes	45	82	0	184	768	648	0	648	852	752	0	314	176	96

Table 6.8: Number of peaks at ±4 in the autocorrelation of the 4564 codes whose the magnitude of the second maximum is 4, and repartition.

6.4.3 Applicability to other binary codes of 20 bits

In this section, we perform the same study as before for the other binary codes of 20 bits, in order to evaluate the potential of the proposed method for other codes.

Binary codes of 20 bits

There are $2^{20} = 1\,048\,576$ binary codes of 20 bits. Among these codes, some are equivalent from the correlation point of view, like a code where each bit is the opposite of each bit of another code, or like two codes where one is a shifted version of the other one. Once these redundant codes have been removed, there are 26 272 binary codes that are unique for our discussion. These 26 272 codes have different autocorrelation properties. Table 6.7 gives the number of codes according to the value of the second maximum in the autocorrelation (the first maximum being 20 when the code is aligned with itself).

For the 4565 codes where the magnitude of the second maximum is 4, the autocorrelation can have as value only 20, 4, 0 and -4. For these 4565 codes, the repartition of the codes according to the number of peaks of ±4 in their auto-correlation is given in Table 6.8. Interestingly, the GPS L5 secondary code is one of the 82 codes having 5 peaks of magnitude 4.

Among these 4565 codes, none of them gives a sub-code $s_{00,n}$ null or constant. Consequently, as before, we need to modify the local secondary code to reduce the complexity. For the following we concentrate on the 45 codes having 4 peaks and the 82 codes having 5 peaks.

Reduction of the complexity

As before, we performed an exhaustive search considering all the codes giving $s_{0,n} = 0$ or $s_{1,n} = 0$, to have a reduction of the complexity of 50 %. The minimum loss that can be obtained depends on the code. For the 82 codes having 5 peaks, the minimum loss can be 1.69 dB, 1.87 dB , 2.33 dB, or 3.01 dB. Therefore, the L5 secondary code is among the best codes for this

application. For the other reductions of complexity (25 % and 33 %), the L5 secondary code is also among the best codes.

6.4.4 Applicability to the Galileo E5 and E1 signals

The secondary code of the Galileo E1 signal has an odd number of chips, which prevents the use of the proposed algorithms. The secondary codes for the Galileo E5a and E5b are 100-ms long, therefore the Doppler effect prevent to use the proposed algorithms, since Eq. (6.3) assume that the samples for each primary code period are identical.

6.5 Summary

In this chapter, we have discussed the problem of the parallel code search acquisition considering the secondary code, and we have explored two directions to solve it.

First, we have presented different ways to use smaller FFTs, by either downsampling or segmenting the sequences. In some cases, it is possible to use the repetition of the primary code, in some cases not. But it is not because we can use the repetition that the complexity is reduced (as shown with a downsampling by a factor N_P). The less complex implementation requires about 11 to 15 % more multiplications and 39 to 45 % more additions compared to the direct implementation of the circular correlation. However, it has been shown that the elementary operation obtained is the same as the one in Chapter 5, therefore the algorithms proposed in Chapter 5 can be applied here as well. Note that it was not possible to make a more accurate comparison for an FPGA implementation of the proposed methods as compared to the direct correlation since it is currently not possible to implement the direct correlation because of the large FFTs required.

Second, we have proposed to change the initial circular correlation into smaller circular and skew-circular correlations by combining some sections of the secondary code in order to be able to reduce the complexity. Unfortunately, it was not possible to reduce the complexity for the L5 secondary code and for any binary code of 20 bits without making an approximation (namely a modification of the local secondary code). To find the best approximation, i.e. the one leading to the lowest SNR loss, we performed an exhaustive search, which was possible thanks to the small size of the problem. The main results for the L5 signal is that it is possible to reduce the complexity by about 50 % in exchange of a SNR loss of 2.2 dB with a protection (ratio between the first and second maximum of the cross-correlation between the received and local secondary code) of 15.56 dB (compared to the 13.98 dB of the L5 secondary code autocorrelation), or it is possible to reduce the complexity by about 33 % in exchange of a SNR loss of 1.69 dB with a protection of 12.04 dB, or of a SNR loss of 1.75 dB with a protection of 14.40 dB.
To have a more accurate evaluation of the performances of the proposed algorithms, the acquisition time should be evaluated. However, as indicated in Section 2.3, this depends on

many parameters, and thus was not undertaken here to concentrate on the computational aspect.

7 Conclusions

7.1 Thesis achievements

In this thesis, we have addressed the problem of the acquisition of GNSS signals using FFT-based parallel architectures, and more specifically the complexity of the acquisition and the ways to reduce it.

In Chapter 3, with the comparison of the implementation on a FPGA of different acquisition methods, we were able to point out the advantages and drawbacks of each implementation. Although the parallel frequency search and the parallel code search may provide similar performance, the parallel code search appeared to be more suitable to process the future GNSS signals and for the future GNSS applications. Indeed, due to its inability to compensate the code Doppler, the parallel frequency search becomes less efficient if we want high sensitivity that requires long integration times, or if we use codes with high chipping rate, or if the frequency search space is very large (which is a bit bothersome for a method whose aim is to search all the frequencies in parallel). Moreover, in case of assistance (self-assistance or external assistance), it is much easier to obtain a rough estimate of the carrier Doppler than an estimate of the code delay, whether we are on Earth, in the air or in space.

In Chapter 4, we have shown different ways to compute an FFT and an FFT-based circular correlation in Altera FPGAs using the Altera FFT, where the resource usage may be reduced, especially for the memory (up to 45 % of reduction), for the same processing time. Although one may think that the FFT proposed by Altera should be optimized for its devices, this work has shown that there is still room for improvement. The algorithms presented are not only useful for a GNSS receiver that needs to perform FFTs or correlations, such as in the parallel code search, but can also be applied to any system that performs a correlation or convolution, since we do not make any assumptions about the input or output signals. Therefore, we hope that this will help either to improve the Altera FFT itself, or the other users of the Altera FFT by reducing the resource usage of their systems.

In Chapter 5, we have proposed two algorithms to reduce the complexity of the parallel code

search acquisition in presence of sign transitions, that may remind the overlap-and-add method usually used for the linear convolution. It has been shown that for an implementation on a FPGA, the algorithms may reduce the resource usage, especially the memory (typically 50 %), for the same processing time. The advantage of these algorithms compared to the other solutions proposed in the literature is that the correlation is computed exactly, i.e. no approximation is done and thus the SNR is not affected. The drawback is that these algorithms do not reduce the complexity all the time, but only under certain conditions. The first condition is to have an FFT that requires a length that is a power of two (which is the case of the FFT provided by FPGA and DSP companies, but not the case on computers where softwares can compute FFT of any lengths). The second condition is linked to the length of the sequences, i.e. to the sampling frequency. However, it is easy to determine if the algorithms are efficient or not with a simple look on a table or on a figure (Table 5.3 p. 116, Fig. 5.11 p. 128). It has been shown that the usual sampling frequencies used for the GPS L5 and Galileo E5a and E5b signals (between 20 to 21 MHz) are in a range where the proposed algorithms reduce the complexity. The algorithms are thus well suited for these signals.

In Chapter 6, we have discussed the problem of the parallel code search acquisition using the secondary code. First we have discussed different implementations that use small FFTs (same order of magnitude as the length of the primary code), because the use of very large FFTs ($> 2^{18}$ points) may be not possible. Such implementations require slightly more operations than the direct implementation of the correlation, even if they are able to exploit the repetition of the primary code. However, it has been shown that one of the implementation was related to the problem discussed in Chapter 5. Therefore, the algorithms proposed in Chapter 5 can be applied here as well, to potentially reduce the complexity. Second, we have discussed implementations to specifically reduce the complexity. For this, we have proposed to build some sub-codes from the secondary code in the hope to obtain some sub-codes that contain only zeros, which would allow us to remove some FFTs. Unfortunately, it was not possible to obtain such sub-codes for the L5 secondary code, or for any binary codes of 20 bits. Therefore, we looked for an approximation, where the local secondary code is different than the received secondary code, for which we have evaluated the impact on the SNR. It has been shown amongst other results that it is possible to reduce the complexity by about 50 % in exchange of a loss of 2.22 dB in the SNR, or to reduce the complexity by about 33 % in exchange of a loss of 1.69 dB.

In Appendix A, we have summarized a lot of definitions and relations, about the DFT, the convolution and the correlation. We have detailed the computation of special matrix-vector products with FFTs (circulant, skew-circulant, Toeplitz, and Hankel matrices), and expressed the convolution and the correlation in different ways (in time domain, using matrix, and polynomial (z transform)). To the best knowledge of the author, such a summary cannot be found in one publication, but the information are spread over the literature. Therefore, we hope to have filled this small gap. Appendix B gives some useful tips regarding the implementation of the FFT and of the correlation applicable in GNSS. Finally, Appendix C provides models for the FPGA implementation of the acquisition architectures, which can help other designers.

In this thesis, we have tried to show the problems and operations in different ways, including the time domain, using matrices and using polynomials (with the z transform). First, because viewing a problem from different points of view helps to better understand the problem. Second, because the different domains provide different mathematical tools and insights.

7.2 Recommendations for future research

The research performed and presented in this thesis is of course not exhaustive, and future improvements are possible.

Use of *a priori* information

Regarding the problem of the acquisition in presence of sign transitions (Chapter 5), the proposed algorithms used two facts, half of the local code values are zeros (due to zero-padding), and half of the output is not needed. However, we know that the input signal contains two code periods with two possible transitions. This third *a priori* information could be used to obtain a better algorithm. This, however, would lead to an approximation algorithm, since the input signal actually also includes a residual carrier and a noise, but the impact on the SNR should not be significant.

Regarding the problem of the acquisition over a secondary code period (Chapter 6), we were not able to find a more efficient algorithm that is not an approximation. However, we deeply think that it may be possible to find one, because in this problem we know a lot of *a priori* information. For the local code, first we know that the primary code is repeated several times; second the secondary code is short, which may allow the use of its specificity; third the codes are binary. With all these information, we really believe that it should be possible to find an algorithm less complex than the classic circular correlation by FFTs that does not use any information about the signals.

Number theory

The fast algorithms developed to compute the DFT or the convolution of sequences are often related to the number theory. Moreover, having a sequence with a length that is a power of two, or a length that is a prime number, or a length that can be expressed as the product of two coprime numbers leads to different properties and algorithms. Therefore it could be interesting to have a deeper look on number theory, for the selection of future secondary codes that would allow efficient computation of the tiered code correlation.

Part III Appendices and bibliography

A Transforms, special matrix-vector products, convolutions and correlations

The goal of this appendix is to provide a summary of some definitions, expressions and relations that have been used during this Ph.D thesis. These information can be found in many books, however, usually a book uses one representation whereas the operations can be represented in several ways. Also, some definitions are not unique (like the definition of a circulant matrix based on a vector), therefore, we clarify the operations corresponding to the different definitions.

A.1 Transforms

A.1.1 z transform

The z transform of a sequence x_n of N points is defined as

$$\begin{aligned} X(z) &= \sum_{n=0}^{N-1} x_n z^{-n} \\ &= x_0 + x_1 z^{-1} + x_2 z^{-2} + x_3 z^{-3} + \cdots + x_{N-1} z^{-(N-1)}, \end{aligned} \tag{A.1}$$

with $z \in \mathbb{C}^*$. $X(z)$ is thus a polynomial of degree $N-1$. See (Proakis and Manolakis [2007], Chapter 3) or (Oppenheim and Schafer [2009], Chapter 3) for more details about the z transform.

The z transform is mainly used for digital filters design, but it is also used to express the convolution of two sequences in a simple way (see next sections). In this last case, the idea is simply to represent a sequence with a polynomial. To do so, it is then also possible to consider the following expression,

$$\begin{aligned} X(z) &= \sum_{n=0}^{N-1} x_n z^{n} \\ &= x_0 + x_1 z + x_2 z^2 + x_3 z^3 + \cdots + x_{N-1} z^{N-1}. \end{aligned} \tag{A.2}$$

Both expressions can be used with convolutions, but the second one is usually preferred because of its simplicity of writing (no minus sign). However, in this thesis, the z transform is used.

A.1.2 Discrete Fourier transform

Time domain view

The discrete Fourier transform (DFT) of a sequence x_n of N points is defined as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi kn}{N}}, \tag{A.3}$$

with $k = 0, 1, \ldots, N-1$. Note that the DFT of a sequence is periodic, i.e. $X_{k+mN} = X_k$ with $m \in \mathbb{Z}$. The DFT can be computed efficiently using fast Fourier transform (FFT) algorithms, which have a complexity in $O(N \log N)$ instead of $O\left(N^2\right)$. See (Brigham [1988], Smith [2007], Chu [2008], Burrus [2012]) for more details about the DFT and the FFT algorithms.

Matrix view

Using matrix notation, the DFT can be expressed as

$$\begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix} = \begin{bmatrix} e^{-\frac{j2\pi\cdot 0\cdot 0}{N}} & e^{-\frac{j2\pi\cdot 0\cdot 1}{N}} & \cdots & e^{-\frac{j2\pi\cdot 0\cdot (N-1)}{N}} \\ e^{-\frac{j2\pi\cdot 1\cdot 0}{N}} & e^{-\frac{j2\pi\cdot 1\cdot 1}{N}} & \cdots & e^{-\frac{j2\pi\cdot 1\cdot (N-1)}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-\frac{j2\pi\cdot (N-1)\cdot 0}{N}} & e^{-\frac{j2\pi\cdot (N-1)\cdot 1}{N}} & \cdots & e^{-\frac{j2\pi\cdot (N-1)\cdot (N-1)}{N}} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} \tag{A.4}$$

$$\overline{\mathbf{X}} = \mathbf{F}\,\mathbf{x},$$

where $\mathbf{F}$ is called the DFT matrix (sometimes there is a factor $\frac{1}{\sqrt{N}}$ included for normalization purpose). The bar over $\mathbf{X}$ in Eq. (A.4) is simply to differentiate it from a matrix. It can be noted that the DFT matrix is symmetric, i.e. $\mathbf{F}^{\mathrm{T}} = \mathbf{F}$.

z transform view

The DFT can also be obtained from the z transform. Indeed, if we evaluate the z transform for $z = e^{\frac{j2\pi k}{N}}$ with $k = 0, 1, \ldots, N-1$, we obtain

$$\begin{aligned} X\left(z = e^{\frac{j2\pi k}{N}}\right) &= \sum_{n=0}^{N-1} x_n \left(e^{\frac{j2\pi k}{N}}\right)^{-n} \\ &= \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi kn}{N}} \\ &= X_k \end{aligned} \tag{A.5}$$

A.1.3 Inverse discrete Fourier transform

Time domain view

The inverse discrete Fourier transform (IDFT) of a sequence X_k of N points is defined as

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{j2\pi nk}{N}}, \tag{A.6}$$

with $n = 0, 1, \ldots, N-1$. Note that the IDFT of a sequence is periodic, i.e. $x_{n+mN} = x_n$ with $m \in \mathbb{Z}$.

Matrix view

Using matrix notation, the IDFT can be expressed as

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} e^{\frac{j2\pi\cdot 0\cdot 0}{N}} & e^{\frac{j2\pi\cdot 0\cdot 1}{N}} & \cdots & e^{\frac{j2\pi\cdot 0\cdot (N-1)}{N}} \\ e^{\frac{j2\pi\cdot 1\cdot 0}{N}} & e^{\frac{j2\pi\cdot 1\cdot 1}{N}} & \cdots & e^{\frac{j2\pi\cdot 1\cdot (N-1)}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{\frac{j2\pi\cdot (N-1)\cdot 0}{N}} & e^{\frac{j2\pi\cdot (N-1)\cdot 1}{N}} & \cdots & e^{\frac{j2\pi\cdot (N-1)\cdot (N-1)}{N}} \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix} \tag{A.7}$$

$$\mathbf{x} = \mathbf{F}^{-1}\,\overline{\mathbf{X}}.$$

It can be noted that $\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$, therefore the IDFT (IFFT) can be computed using the DFT (FFT).

z transform view

In the same way as the DFT, the IDFT can be obtained from the z transform by evaluating it for $z = e^{-\frac{j2\pi k}{N}}$ with $k = 0, 1, \ldots, N-1$, not taking into account the factor $\frac{1}{N}$.

A.2 Special matrices

A.2.1 Circulant matrix

A circulant matrix is a matrix where each row is obtained by shifting the previous row by one element (or by shifting the previous column by one element). Considering a vector $\mathbf{h} = [h_0\ h_1\ \cdots\ h_{N-1}]$, there are three main ways to define a circulant matrix. First, the vector $\mathbf{h}$ can correspond to the first column or to the first row. Second, the shift between two consecutive rows can be on the right or on the left direction. Therefore, we differentiate these cases, and indicate the result of the diagonalization for each one. See (Davis [1979]) for more details about the circulant matrices.

In the following, the matrices are denoted with a subscript that contains up to three elements between curly brackets. The first element indicates the type of matrix (C for circulant, S for skew-circulant, etc.); the second element indicates if the vector used to "generate" the matrix corresponds to its first column (symbol $|$) or its first row (symbol $-$); and the third element indicates the direction of the shift, $\searrow$ is used for a right shift between consecutive rows (or equivalently a down shift between consecutive columns), and $\swarrow$ is used for a left shift between consecutive rows (or equivalently a up shift between consecutive columns).

Right circulant matrix defined by its first column

Considering that the vector $\mathbf{h}$ corresponds to the first column, and that the shift is on the right direction, we obtain the following matrix,

$$\mathbf{H}_{\{C|\searrow\}} = \begin{bmatrix} h_0 & h_{N-1} & \cdots & h_2 & h_1 \\ h_1 & h_0 & \cdots & h_3 & h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-2} & h_{N-3} & \cdots & h_0 & h_{N-1} \\ h_{N-1} & h_{N-2} & \cdots & h_1 & h_0 \end{bmatrix}. \tag{A.8}$$

Such a matrix is used in the circular convolution of two sequences (see Section A.3.2). It can be diagonalized by the DFT matrix (see Eq. (A.4)), i.e. we can write

$$\mathbf{H}_{\{C|\searrow\}} = \mathbf{F}^{-1}\,\mathrm{diag}\,(\mathbf{F}\,\mathbf{h})\,\mathbf{F}, \tag{A.9}$$

or

$$\mathbf{H}_{\{C|\searrow\}} = N\,\mathbf{F}\,\mathrm{diag}\,(\mathbf{F}^{-1}\mathbf{h})\,\mathbf{F}^{-1}. \tag{A.10}$$

Right circulant matrix defined by its first row

Considering that the vector $\mathbf{h}$ corresponds to the first row, and that the shift is on the right direction, we obtain the following matrix,

$$\mathbf{H}_{\{C-\searrow\}} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{N-2} & h_{N-1} \\ h_{N-1} & h_0 & \cdots & h_{N-3} & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_2 & h_3 & \cdots & h_0 & h_1 \\ h_1 & h_2 & \cdots & h_{N-1} & h_0 \end{bmatrix}. \tag{A.11}$$

Such a matrix is used in the circular correlation of two sequences (see Section A.3.3). It can be

diagonalized by the DFT matrix, i.e. we can write

$$\mathbf{H}_{\{C-\searrow\}} = N\,\mathbf{F}^{-1}\,\mathrm{diag}\,(\mathbf{F}^{-1}\mathbf{h})\,\mathbf{F}, \tag{A.12}$$

or

$$\mathbf{H}_{\{C-\searrow\}} = \mathbf{F}\,\mathrm{diag}\,(\mathbf{F}\,\mathbf{h})\,\mathbf{F}^{-1}. \tag{A.13}$$

Left circulant matrix

Considering that the vector $\mathbf{h}$ corresponds to the first column or to the first row, and that the shift is on the left direction, we obtain the following matrix,

$$\mathbf{H}_{\{C\swarrow\}} = \begin{bmatrix} h_0 & h_1 & \cdots & h_{N-2} & h_{N-1} \\ h_1 & h_2 & \cdots & h_{N-1} & h_0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-2} & h_{N-1} & \cdots & h_{N-4} & h_{N-3} \\ h_{N-1} & h_0 & \cdots & h_{N-3} & h_{N-2} \end{bmatrix}. \tag{A.14}$$

Such a matrix is used in the circular correlation of two sequences (see Section A.3.3). It can be diagonalized by the DFT matrix, i.e. we can write

$$\mathbf{H}_{\{C\swarrow\}} = N\,\mathbf{F}^{-1}\,\mathrm{diag}\,(\mathbf{F}\,\mathbf{h})\,\mathbf{F}^{-1}, \tag{A.15}$$

or

$$\mathbf{H}_{\{C\swarrow\}} = \mathbf{F}\,\mathrm{diag}\,(\mathbf{F}^{-1}\mathbf{h})\,\mathbf{F}. \tag{A.16}$$

A.2.2 Skew-circulant matrix

A skew-circulant matrix is a matrix where each row is obtained by shifting the previous row by one element and the sign of the elements above the diagonal is changed.

Right circulant matrix defined by its first column

Considering that the vector $\mathbf{h}$ corresponds to the first column, and that the shift is on the right direction, we obtain the following matrix,

$$\mathbf{H}_{\{S|\searrow\}} = \begin{bmatrix} h_0 & -h_{N-1} & \cdots & -h_2 & -h_1 \\ h_1 & h_0 & \cdots & -h_3 & -h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-2} & h_{N-3} & \cdots & h_0 & -h_{N-1} \\ h_{N-1} & h_{N-2} & \cdots & h_1 & h_0 \end{bmatrix} \tag{A.17}$$

Such a matrix is used in the skew-circular convolution of two sequences (see Section A.3.4). It can be expressed using a diagonal matrix and a circulant matrix as

$$\mathbf{H}_{\{S|\searrow\}} = \Omega^{-1}\, \mathbf{H}_{\Omega\{C|\searrow\}}\, \Omega, \tag{A.18}$$

where $\Omega = \text{diag}\left(e^{\frac{-j\pi\cdot 0}{N}}, e^{\frac{-j\pi\cdot 1}{N}}, \cdots, e^{\frac{-j\pi\cdot(N-1)}{N}}\right)$, and the first column of $\mathbf{H}_{\Omega\{C|\searrow\}}$ is $\Omega\,\mathbf{h}$ (Vaidyanathan et al. [2010] pp. 771-775). Therefore, a skew-circulant matrix can be diagonalized by the DFT matrix. Using Eq. (A.9), we can write

$$\mathbf{H}_{\{S|\searrow\}} = \Omega^{-1}\, \mathbf{F}^{-1}\, \text{diag}\,(\mathbf{F}\,\Omega\,\mathbf{h})\, \mathbf{F}\,\Omega, \tag{A.19}$$

and using Eq. (A.10), we can write

$$\mathbf{H}_{\{S|\searrow\}} = N\,\Omega^{-1}\, \mathbf{F}\text{diag}\,(\mathbf{F}^{-1}\Omega\,\mathbf{h})\, \mathbf{F}^{-1}\Omega. \tag{A.20}$$

A.2.3 Toeplitz matrix

A Toeplitz matrix is a matrix where each descending diagonal is constant, it is then defined by $2N-1$ elements.

$$\mathbf{H}_T = \begin{bmatrix} h_0 & h_{-1} & \cdots & h_{-(N-2)} & h_{-(N-1)} \\ h_1 & h_0 & \cdots & h_{-(N-3)} & h_{-(N-2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-2} & h_{N-3} & \cdots & h_0 & h_{-1} \\ h_{N-1} & h_{N-2} & \cdots & h_1 & h_0 \end{bmatrix} \tag{A.21}$$

Therefore, a right circulant matrix is a special case of Toeplitz matrix.

Change to a circular matrix

A Toeplitz matrix can be embedded in a circulant matrix by approximately doubling its size.

Here is an example with $N = 3$. The following $N \times N$ Toeplitz matrix,

$$H_T = \begin{bmatrix} h_0 & h_{-1} & h_{-2} \\ h_1 & h_0 & h_{-1} \\ h_2 & h_1 & h_0 \end{bmatrix}, \tag{A.22}$$

can be embedded into the following $2N-1 \times 2N-1$ right circulant matrix,

$$\begin{bmatrix} h_0 & h_{-1} & h_{-2} & h_2 & h_1 \\ h_1 & h_0 & h_{-1} & h_{-2} & h_2 \\ h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\ h_{-2} & h_2 & h_1 & h_0 & h_{-1} \\ h_{-1} & h_{-2} & h_2 & h_1 & h_0 \end{bmatrix}, \tag{A.23}$$

or into the following $2N \times 2N$ right circulant matrix,

$$\begin{bmatrix} h_0 & h_{-1} & h_{-2} & \cdot & h_2 & h_1 \\ h_1 & h_0 & h_{-1} & h_{-2} & \cdot & h_2 \\ h_2 & h_1 & h_0 & h_{-1} & h_{-2} & \cdot \\ \cdot & h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\ h_{-2} & \cdot & h_2 & h_1 & h_0 & h_{-1} \\ h_{-1} & h_{-2} & \cdot & h_2 & h_1 & h_0 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_T & \mathbf{B} \\ \mathbf{B} & \mathbf{H}_T \end{bmatrix}, \tag{A.24}$$

where the $\cdot$ can be replaced by any values.

Expression with a circulant and skew-circulant matrix

A Toeplitz matrix can be expressed as the sum of a circulant matrix and a skew-circulant matrix of same size, i.e. $\mathbf{H}_T = \mathbf{H}_C + \mathbf{H}_S$ (Ng [2003]).

Here is an example with $N = 3$,

$$\begin{aligned} H_T = \begin{bmatrix} h_0 & h_{-1} & h_{-2} \\ h_1 & h_0 & h_{-1} \\ h_2 & h_1 & h_0 \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} h_0 & h_2 + h_{-1} & h_1 + h_{-2} \\ h_1 + h_{-2} & h_0 & h_2 + h_{-1} \\ h_2 + h_{-1} & h_1 + h_{-2} & h_0 \end{bmatrix} \\ &\quad + \frac{1}{2} \begin{bmatrix} h_0 & -(h_2 - h_{-1}) & -(h_1 - h_{-2}) \\ h_1 - h_{-2} & h_0 & -(h_2 - h_{-1}) \\ h_2 - h_{-1} & h_1 - h_{-2} & h_0 \end{bmatrix}. \end{aligned} \tag{A.25}$$

A.2.4 Hankel matrix

A Hankel matrix is a matrix where each ascending diagonal is constant, it is then defined by $2N-1$ elements.

$$\mathbf{H}_H = \begin{bmatrix} h_0 & h_1 & \cdots & h_{N-2} & h_{N-1} \\ h_1 & h_2 & \cdots & h_{N-1} & h_{-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-2} & h_{N-1} & \cdots & h_{-(N-3)} & h_{-(N-2)} \\ h_{N-1} & h_{-1} & \cdots & h_{-(N-2)} & h_{-(N-1)} \end{bmatrix} \tag{A.26}$$

Therefore, a left circulant matrix is a special case of Hankel matrix. In the same way as a Toeplitz matrix, a Hankel matrix can be embedded into a circulant matrix, or can be expressed as the sum of a circulant and skew-circulant matrix.

A.3 Convolutions and correlations

A.3.1 Linear convolution

Time domain view

The linear convolution y_n of two sequences h_n and x_n of N points is defined as

$$\begin{aligned} y_n &= \sum_{k=0}^{N-1} h_k x_{n-k} \\ &= \sum_{k=0}^{N-1} x_k h_{n-k}, \end{aligned} \tag{A.27}$$

with $n = 0, 1, \ldots, 2N-2$. Note that the linear convolution is a commutative operation. See (Winograd [1980], Nussbaumer [1982], Garg [1998], Blahut [2010]) for more details about the convolution.

Matrix view

Using matrix notation, the linear convolution can be expressed as

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \\ y_N \\ \vdots \\ y_{2N-3} \\ y_{2N-2} \end{bmatrix} &= \begin{bmatrix} x_0 & 0 & \cdots & 0 & 0 \\ x_1 & x_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N-2} & x_{N-3} & \cdots & x_0 & 0 \\ x_{N-1} & x_{N-2} & \cdots & x_1 & x_0 \\ 0 & x_{N-1} & \cdots & x_2 & x_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & x_{N-1} & x_{N-2} \\ 0 & 0 & \cdots & 0 & x_{N-1} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-2} \\ h_{N-1} \end{bmatrix} \\
&= \begin{bmatrix} h_0 & 0 & \cdots & 0 & 0 \\ h_1 & h_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-2} & h_{N-3} & \cdots & h_0 & 0 \\ h_{N-1} & h_{N-2} & \cdots & h_1 & h_0 \\ 0 & h_{N-1} & \cdots & h_2 & h_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & h_{N-1} & h_{N-2} \\ 0 & 0 & \cdots & 0 & h_{N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \\
\mathbf{y} &= \mathbf{X}_T\, \mathbf{h} \\
&= \mathbf{H}_T\, \mathbf{x}.
\end{aligned} \tag{A.28}$$

z transform view

Using the z transform, the linear convolution is defined as

$$Y(z) = H(z)X(z), \tag{A.29}$$

where $Y(z)$, $H(z)$ and $X(z)$ are the z transforms of y_n, h_n and x_n, respectively, i.e. polynomials of degree $2N-2$, $N-1$ and $N-1$ respectively. This means that the linear convolution of two sequences can be treated as the product of two polynomials.

A.3.2 Circular convolution

Time domain view

The circular convolution y_n of two sequences h_n and x_n of N points is defined as

$$\begin{aligned} y_n &= \sum_{k=0}^{N-1} h_k x_{(n-k) \bmod N} \\ &= \sum_{k=0}^{N-1} x_k h_{(n-k) \bmod N}, \end{aligned} \tag{A.30}$$

with $n = 0, 1, \ldots, N-1$, and mod denotes the modulo operation, i.e. $(n + mN) \bmod N = n$ with $m \in \mathbb{Z}$. Note that the circular convolution is a commutative operation.

Matrix view

Using matrix notation, the circular convolution can be expressed as

$$\begin{aligned} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} &= \begin{bmatrix} x_0 & x_{N-1} & \cdots & x_2 & x_1 \\ x_1 & x_0 & \cdots & x_3 & x_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N-2} & x_{N-3} & \cdots & x_0 & x_{N-1} \\ x_{N-1} & x_{N-2} & \cdots & x_1 & x_0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-2} \\ h_{N-1} \end{bmatrix} \\ &= \begin{bmatrix} h_0 & h_{N-1} & \cdots & h_2 & h_1 \\ h_1 & h_0 & \cdots & h_3 & h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-2} & h_{N-3} & \cdots & h_0 & h_{N-1} \\ h_{N-1} & h_{N-2} & \cdots & h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \\ \mathbf{y} &= \mathbf{X}_{\{C|\searrow\}}\, \mathbf{h} \\ &= \mathbf{H}_{\{C|\searrow\}}\, \mathbf{x}. \end{aligned} \tag{A.31}$$

The matrices $\mathbf{X}_{\{C|\searrow\}}$ and $\mathbf{H}_{\{C|\searrow\}}$ are right circulant matrices with $\mathbf{x}$ and $\mathbf{h}$ as first column, respectively. Therefore, using the diagonalization given by Eq. (A.9), the circular convolution can be computed as

$$\begin{aligned} \mathbf{y} &= \mathbf{H}_{\{C|\searrow\}}\mathbf{x} \\ &= \mathbf{F}^{-1} \operatorname{diag}(\mathbf{F}\,\mathbf{h})\, \mathbf{F}\,\mathbf{x} \\ &= \mathbf{F}^{-1}\big((\mathbf{F}\,\mathbf{h}) \circ (\mathbf{F}\,\mathbf{x})\big), \end{aligned} \tag{A.32}$$

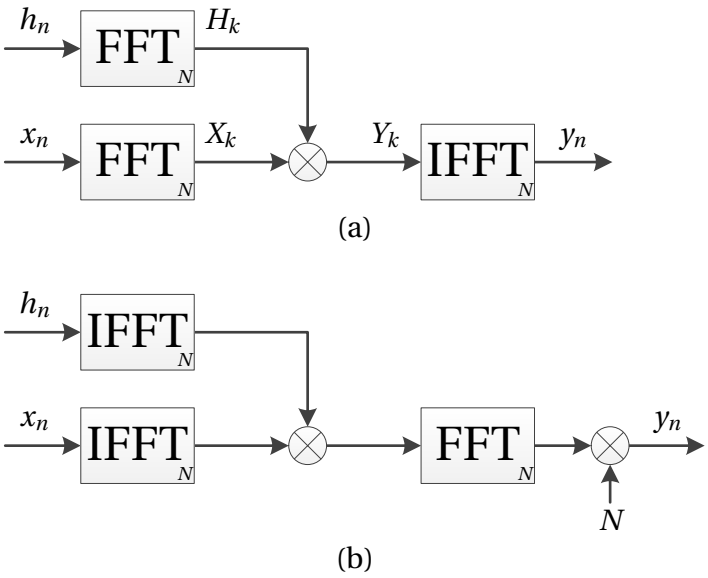


Figure A.1: Computation of the circular convolution of two sequences of N points with FFTs, (a) using Eq. (A.32), (b) using Eq. (A.33).

or using the diagonalization given by Eq. (A.10), as

$$\begin{aligned}\mathbf{y} &= \mathbf{H}_{\{C|\diagdown\}}\mathbf{x} \\ &= N\,\mathbf{F}\,\mathrm{diag}\,(\mathbf{F}^{-1}\mathbf{h})\,\mathbf{F}^{-1}\mathbf{x} \\ &= N\,\mathbf{F}\left((\mathbf{F}^{-1}\mathbf{h})\circ(\mathbf{F}^{-1}\mathbf{x})\right),\end{aligned} \tag{A.33}$$

where $\circ$ denotes the Hadamard product (element by element product). Therefore, the circular convolution can be computed efficiently using FFTs, as shown in Fig. A.1.

z transform view

Using the z transform, the circular convolution is defined as

$$Y(z) = H(z)X(z) \mod (z^{-N}-1), \tag{A.34}$$

where $Y(z)$, $H(z)$ and $X(z)$ are the z transforms of y_n, h_n and x_n, respectively, i.e. polynomials of degree $N-1$ (Nussbaumer [1982], pp. 22-23). Eq. (A.34) can also be written as

$$Y(z) = H(z)X(z) - Q(z)(z^{-N}-1), \tag{A.35}$$

where $Q(z)$ is a polynomial of degree $N-2$. If we evaluate this equation for $z = e^{\frac{j2\pi k}{N}}$ with $k = 0,1,\ldots,N-1$, we have $z^{-N}-1 = 0$ and thus

$$Y_k = H_k X_k, \tag{A.36}$$

where Y_k, H_k and X_k are the DFTs of y_n, h_n and x_n, respectively. So, by computing the IDFT of $H_k X_k$ we obtain y_n, as already shown in Fig. A.1.

A.3.3 Circular correlation

Time domain view

The circular correlation y_n of two sequences h_n and x_n of N points can be defined as

$$y_n = \sum_{k=0}^{N-1} h_k^* x_{(n+k) \bmod N}, \tag{A.37}$$

with $n = 0, 1, \ldots, N-1$, and mod denotes the modulo operation, i.e. $(n + mN) \bmod N = n$ with $m \in \mathbb{Z}$.

Note that the correlation is not commutative. Indeed, in the following equation,

$$w_n = \sum_{k=0}^{N-1} x_k^* h_{(n+k) \bmod N}, \tag{A.38}$$

with $n = 0, 1, \ldots, N-1$, w_n is the conjugate of the sequence y_n flipped, i.e. $w_n = y^*_{-n \bmod N}$. So, what follows applies also to if we commute h_n and x_n, but then y_n will be flipped and conjugated.

Matrix view

Using matrix notation, the circular correlation can be expressed as

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} &= \begin{bmatrix} x_0 & x_1 & \cdots & x_{N-2} & x_{N-1} \\ x_1 & x_2 & \cdots & x_{N-1} & x_0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N-2} & x_{N-1} & \cdots & x_{N-4} & x_{N-3} \\ x_{N-1} & x_0 & \cdots & x_{N-3} & x_{N-2} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-2} \\ h_{N-1} \end{bmatrix}^* \\
&= \begin{bmatrix} h_0 & h_1 & \cdots & h_{N-2} & h_{N-1} \\ h_{N-1} & h_0 & \cdots & h_{N-3} & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_2 & h_3 & \cdots & h_0 & h_1 \\ h_1 & h_2 & \cdots & h_{N-1} & h_0 \end{bmatrix}^* \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \\
\mathbf{y} &= \mathbf{X}_{\{C\swarrow\}}\, \mathbf{h}^* \\
&= \mathbf{H}^*_{\{C-\searrow\}}\, \mathbf{x}.
\end{aligned} \tag{A.39}$$

The matrix $\mathbf{X}_{\{C\swarrow\}}$ is a left circulant matrix with $\mathbf{x}$ as first row and first column, and the matrix

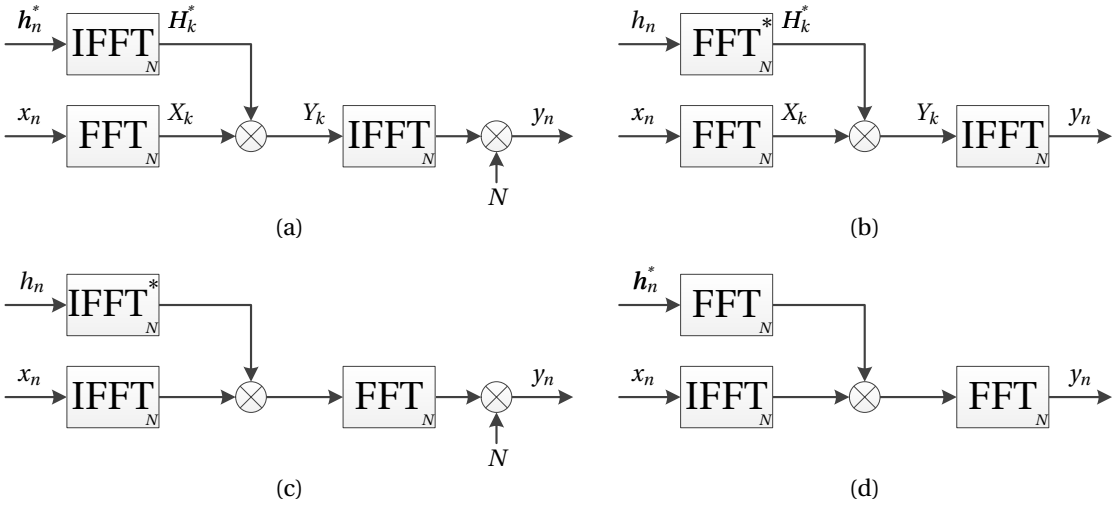


Figure A.2: Computation of the circular correlation of two sequences of N points with FFTs, (a) and (b) using Eq. (A.40), (b) and (d) using Eq. (A.41).

$\mathbf{H}_{\{C-\searrow\}}$ is a right circulant matrix with $\mathbf{h}$ as first row.

Therefore, using the diagonalization given by Eq. (A.15), this circular correlation can be computed as

$$\begin{aligned}\mathbf{y} &= \mathbf{X}_{\{C\swarrow\}}\,\mathbf{h}^* \\ &= N\,\mathbf{F}^{-1}\,\mathrm{diag}\,(\mathbf{F}\,\mathbf{x})\,\mathbf{F}^{-1}\mathbf{h}^* \\ &= N\,\mathbf{F}^{-1}\big((\mathbf{F}^{-1}\mathbf{h}^*)\circ(\mathbf{F}\,\mathbf{x})\big) \\ &= \mathbf{F}^{-1}\big((\mathbf{F}\,\mathbf{h})^*\circ(\mathbf{F}\,\mathbf{x})\big),\end{aligned} \tag{A.40}$$

or using the diagonalization given by Eq. (A.16), as

$$\begin{aligned}\mathbf{y} &= \mathbf{X}_{\{C\swarrow\}}\,\mathbf{h}^* \\ &= \mathbf{F}\;\mathrm{diag}\,(\mathbf{F}^{-1}\mathbf{x})\,\mathbf{F}\,\mathbf{h}^* \\ &= \mathbf{F}\,\big((\mathbf{F}\,\mathbf{h}^*)\circ(\mathbf{F}^{-1}\mathbf{x})\big) \\ &= N\,\mathbf{F}\,\big((\mathbf{F}^{-1}\mathbf{h})^*\circ(\mathbf{F}^{-1}\mathbf{x})\big).\end{aligned} \tag{A.41}$$

Therefore, the circular correlation can be computed efficiently using FFTs, as shown in Fig. A.2.

z transform view

Using the z transform, the circular correlation is defined as

$$Y(z) = H^*(1/z^*)X(z) \mod (z^{-N}-1), \tag{A.42}$$

where $Y(z)$, $H(z)$ and $X(z)$ are the z transforms of y_n, h_n and x_n, respectively. Eq. (A.42) can also be written as

$$Y(z) = H^*(1/z^*)X(z) - Q(z)(z^{-N} - 1), \tag{A.43}$$

where $Q(z)$ is a polynomial of degree $N-2$. If we evaluate this equation for $z = e^{\frac{j2\pi k}{N}}$ with $k = 0, 1, \ldots, N-1$, we have $z^{-N} - 1 = 0$ and thus

$$Y_k = H_k^* X_k, \tag{A.44}$$

where Y_k, H_k and X_k are the DFTs of y_n, h_n and x_n, respectively. So, by computing the IDFT of $H_k^* X_k$ we obtain y_n, as already shown in Fig. A.2.

A.3.4 Skew-circular convolution

Time domain view

The skew-circular convolution y_n of two sequences h_n and x_n of N points is defined as

$$\begin{aligned} y_n &= \sum_{k=0}^{N-1} \operatorname{sgn}(n-k) h_k x_{(n-k) \bmod N} \\ &= \sum_{k=0}^{N-1} \operatorname{sgn}(n-k) x_k h_{(n-k) \bmod N}, \end{aligned} \tag{A.45}$$

with $n = 0, 1, \ldots, N-1$, and sgn the function defined as

$$\operatorname{sgn}(m) = \begin{cases} 1, & \text{for } m \geqslant 0 \\ -1, & \text{for } m < 0 \end{cases}. \tag{A.46}$$

Note that the skew-circular convolution is a commutative operation.

Matrix view

Using matrix notation, the skew-circular convolution can be expressed as

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} &= \begin{bmatrix} x_0 & -x_{N-1} & \cdots & -x_2 & -x_1 \\ x_1 & x_0 & \cdots & -x_3 & -x_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N-2} & x_{N-3} & \cdots & x_0 & -x_{N-1} \\ x_{N-1} & x_{N-2} & \cdots & x_1 & x_0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-2} \\ h_{N-1} \end{bmatrix} \\
&= \begin{bmatrix} h_0 & -h_{N-1} & \cdots & -h_2 & -h_1 \\ h_1 & h_0 & \cdots & -h_3 & -h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-2} & h_{N-3} & \cdots & h_0 & -h_{N-1} \\ h_{N-1} & h_{N-2} & \cdots & h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \\
\mathbf{y} &= \mathbf{X}_{\{S|\searrow\}}\,\mathbf{h} \\
&= \mathbf{H}_{\{S|\searrow\}}\,\mathbf{x}.
\end{aligned} \tag{A.47}$$

The matrices $\mathbf{X}_{\{S|\searrow\}}$ and $\mathbf{H}_{\{S|\searrow\}}$ are right skew-circulant matrices with $\mathbf{x}$ and $\mathbf{h}$ as first column, respectively. Therefore, using the diagonalization given by Eq. (A.19), the skew-circular convolution can be computed as

$$\begin{aligned}
\mathbf{y} &= \mathbf{H}_{\{S|\searrow\}}\mathbf{x} \\
&= \Omega^{-1}\,\mathbf{F}^{-1}\,\mathrm{diag}\,(\mathbf{F}\,\Omega\,\mathbf{h})\,\mathbf{F}\,\Omega\,\mathbf{x} \\
&= \omega^* \circ \mathbf{F}^{-1}\Big(\big(\mathbf{F}(\omega \circ \mathbf{h})\big) \circ \big(\mathbf{F}(\omega \circ \mathbf{x})\big)\Big),
\end{aligned} \tag{A.48}$$

or using the diagonalization given by Eq. (A.20), as

$$\begin{aligned}
\mathbf{y} &= \mathbf{H}_{\{S|\searrow\}}\mathbf{x} \\
&= N\,\Omega^{-1}\,\mathbf{F}\,\mathrm{diag}\,(\mathbf{F}^{-1}\Omega\,\mathbf{h})\,\mathbf{F}^{-1}\Omega\,\mathbf{x} \\
&= N\,\omega^* \circ \mathbf{F}\Big(\big(\mathbf{F}^{-1}(\omega \circ \mathbf{h})\big) \circ \big(\mathbf{F}^{-1}(\omega \circ \mathbf{x})\big)\Big)
\end{aligned} \tag{A.49}$$

with $\omega = \begin{bmatrix} e^{\frac{-j\pi\cdot 0}{N}} & e^{\frac{-j\pi\cdot 1}{N}} & \cdots & e^{\frac{-j\pi\cdot(N-1)}{N}} \end{bmatrix}$. Therefore, the skew-circular convolution can be computed efficiently using an FFT, as shown in Fig. A.3.

z transform view

Using the z transform, the skew-circular convolution is defined as

$$Y(z) = H(z)X(z) \mod (z^{-N}+1), \tag{A.50}$$

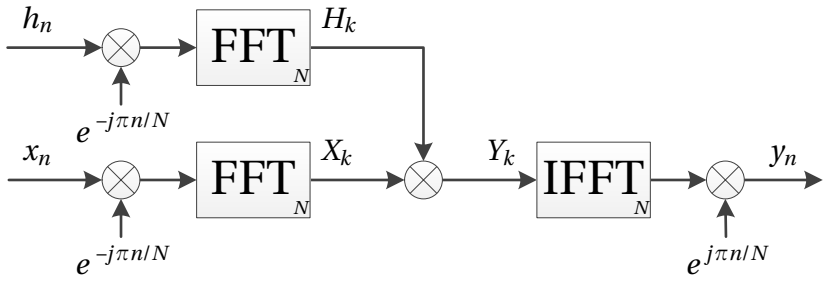


Figure A.3: Computation of the skew-circular convolution of two sequences of N points using FFTs.

where $Y(z)$, $H(z)$ and $X(z)$ are the z transforms of y_n, h_n and x_n, respectively, i.e. polynomials of degree $N-1$.

B Useful tips for the use of FFTs in GNSS

In this appendix, we have collected some useful tips when the FFT is used to compute the correlation of sequences, which can therefore be applied to the parallel code search acquisition of GNSS signals.

B.1 Data order with the radix-2 FFT

The radix-2 FFT algorithm naturally scrambles the order of the samples of the input sequence or of the output sequence. More specifically, we talk about bit-reversed indexing (Lyons [2010], pp. 135-159). This is illustrated in Table B.1 considering an 8-point FFT. This means that to have the natural index order for both the input and the output, an additional stage is needed. This stage of course implies a longer time for software FFTs, and more memory and latency for hardware FFTs (Altera [2013]).

However, if we compute a circular convolution or correlation using FFTs, it is not needed to reorder the sample for each FFT and IFFT. Indeed, for the product of the FFT results, there is no need to have the samples in the natural order, thus it is sufficient to select cleverly the index order for the input and output samples of the FFTs, as shown in Fig. B.1.

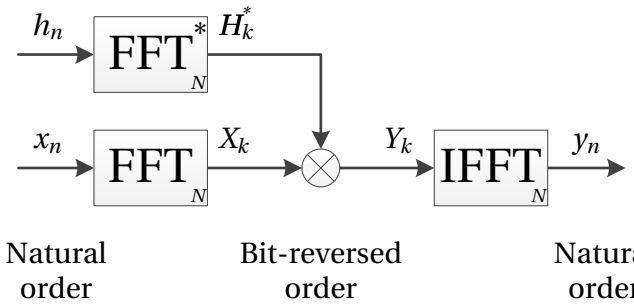


Figure B.1: Circular correlation of two sequences computed by FFT using smart choice for the index order of the FFT input and output samples.

Natural order of index n	Bit-reversed order of index n
0 (000)	0 (000)
1 (001)	4 (100)
2 (010)	2 (010)
3 (011)	6 (110)
4 (100)	1 (001)
5 (101)	5 (101)
6 (110)	3 (011)
7 (111)	7 (111)

Table B.1: Natural and bit-reversed index order for an 8-point FFT. Between parenthesis is the value in binary.

For the Altera FFT, it is possible to select the order of the samples only with the variable streaming I/O data flow. Table B.2 gives the estimates of the resources of a 4096-point FFT for the different index orders. Therefore, implementing a correlation using only the natural index order requires $3 \times 452\,904 = 1\,358\,712$ bits, whereas using the order depicted in Fig. B.1 requires $2 \times 182\,568 + 1 \times 271\,626 = 636\,762$, i.e. a reduction of about 53 %. Therefore, if the variable streaming I/O data flow is used, it is definitely worthy to consider different index order for the convolution or correlation of sequences implemented by FFTs.

B.2 FFT of real sequences

The FFT is an algorithm considering complex sequences as input. However, it may happen that we have to compute the FFT of a real sequence, as in GNSS with the parallel code search where we compute the FFT of the local code. Using directly the FFT with the imaginary part of the input equal to zero is thus a waste. However, with some manipulations, it is possible to use one FFT of N points to compute the FFT of two real sequences of N points, or to compute the FFT of a real sequence of $2N$ points (Sorensen et al. [1987], Lyons [2010] pp. 687-699).

B.2.1 Computing the N-point FFT of two real sequences using one N-point FFT

Let's consider two real sequences, $h_{0,n}$ and $h_{1,n}$ of length N, and their corresponding FFT, $H_{0,k}$ and $H_{1,k}$. Now, let's consider the complex sequence h_n defined as $h_n = h_{0,n} + j h_{1,n}$, and its FFT H_k. Then, $H_{0,k}$ and $H_{1,k}$ can be obtained from H_k as

$$\begin{aligned} H_{0,k} &= \frac{H^*_{N-k} + H_k}{2} \\ &= \frac{\mathrm{Re}(H_{N-k}) + \mathrm{Re}(H_k)}{2} + j\frac{\mathrm{Im}(H_k) - \mathrm{Im}(H_{N-k})}{2}, \end{aligned} \tag{B.1}$$

Function	Logic usage (ALUT)	Memory usage (bits)	Multipliers usage (DSP element)
4096-point FFT with input and output in natural order	8534	452 904	40
4096-point FFT with input in natural order and output in bit-reversed order	8534	182 568	40
4096-point IFFT with input in bit-reversed order and output in natural order	8534	271 626	40

Table B.2: Resources estimated with the MegaWizard Plug-in Manager for an FFT of 4096 points implemented on a Stratix III FPGA, considering the variable streaming I/O data flow, and 18 bits for the data and twiddle precision.

and

$$\begin{aligned} H_{1,k} &= j\frac{H^*_{N-k} - H_k}{2} \\ &= \frac{\mathrm{Im}(H_{N-k}) + \mathrm{Im}(H_k)}{2} + j\frac{\mathrm{Re}(H_{N-k}) - \mathrm{Re}(H_k)}{2}, \end{aligned} \tag{B.2}$$

where Re and Im denotes the real and imaginary part, respectively. For the case $k = 0$, remember that $H_N = H_0$ (see Section A.1.2).

For a hardware implementation, since we have to add and subtract the sequence H_k and its reverse H_{N-k}, we need to store the samples at the output of the FFT. The implementation is given Fig. B.2, and its corresponding timing diagram in Fig. B.3, where consecutive FFTs are computed. It can be seen that the writing of the samples of the second period starts while the reading of the reversed samples of the first period is not yet finished, which prevents to use only one memory. This implies to use two memories of N complex words with a write access and a double read access, and these memories will be written and read alternatively. The combination block implements Eqs. (B.1) and (B.2), it is then composed of four real adders. Note that this implementation requires an extra latency of N cycles compared to the use of two FFTs. Note also that if the FFTs are not performed directly one after the other, it is possible to use only one memory.

Let's see an example with the Altera FFT. Considering the case $N = 4096$ and the streaming I/O data flow, the Altera FFT uses 6906 ALUTs, 38 M9Ks, and 24 DSP elements (considering the same parameters as in Chapter 4, see p. 79). Therefore implementing two FFTs would requires 13 812 ALUTs, 76 M9Ks, and 48 DSP elements. Whereas using the implementation of Fig. B.2, there would be one FFT and two memories. Each memory requires $2 \times 18 \times 4096 = 36\,864$ bits, which can be stored with 4 M9Ks. Therefore, the implementation of Fig. B.2 requires 56 M9Ks,

which means 29 % less than using two FFTs. But the number of DSP elements is divided by two, and the logic usage is also almost divided by two, since the addressing of the memory requires few logic but not that much.

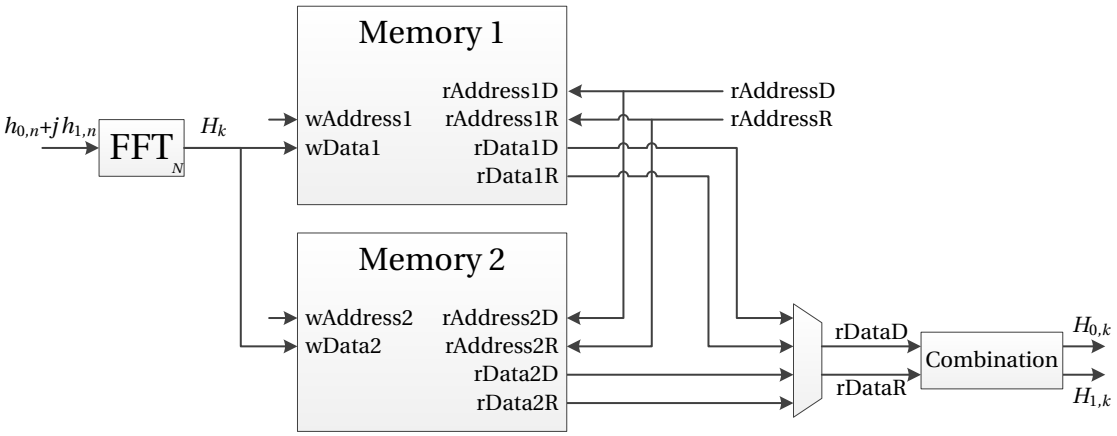


Figure B.2: Hardware implementation to compute simultaneously the FFTs of two real sequences.

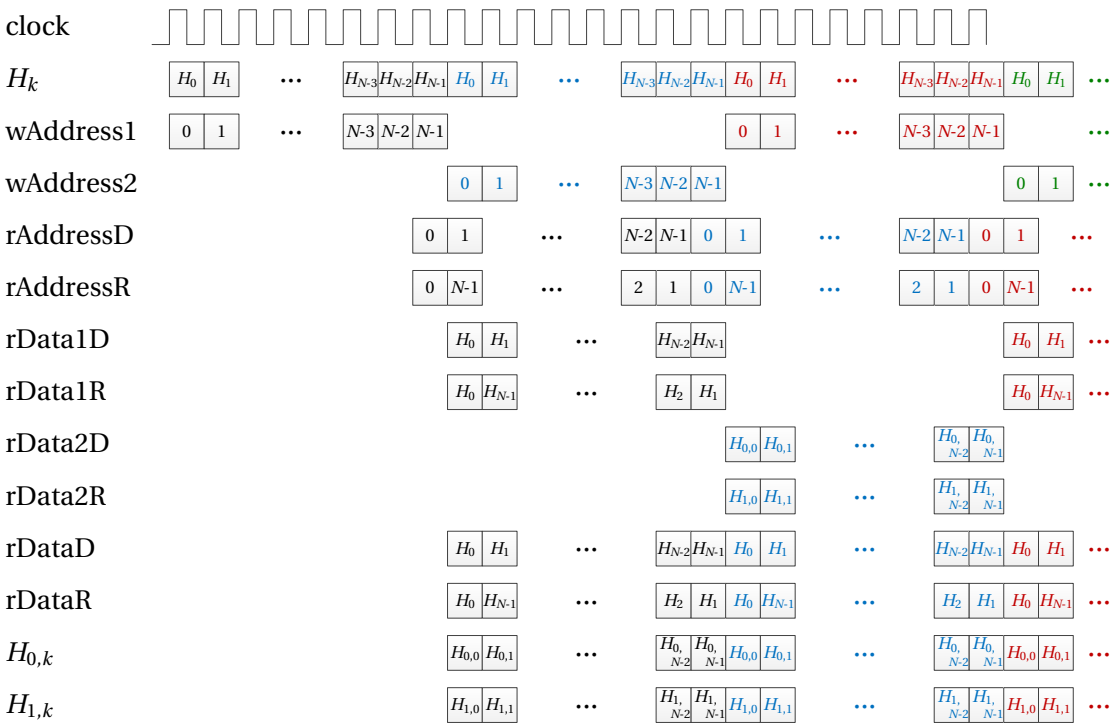


Figure B.3: Timing diagram of Fig. B.2.

B.2.2 Computing the 2*N*-point FFT of a real sequence using one *N*-point FFT

Let's consider a real sequence, h_n of length $2N$, and its corresponding FFT, H_k. Now, let's consider the complex sequence $h_{C,n}$ defined as $h_n = h_{2n} + jh_{2n+1}$, and its FFT $H_{C,k}$.

Then, in a similar way as previous, H_k can be obtained from $H_{C,k}$ using combinations of its real and imaginary part. However, an additional step is needed, which involves the multiplication by a cosine and a sine (see Lyons [2010]). Therefore, the complexity will be a little bit higher than previously, because of the generation of the cosine and sine wave, but there will still be a significant reduction in terms of logic and memory.

B.3 Conjugate of the FFT of a real sequence

In Section A.1.3, it has been shown that $\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$, with $\mathbf{F}$ the DFT matrix. Using this relation, we can write

$$\text{FFT}^*(h_n) = N\,\text{IFFT}(h_n^*). \tag{B.3}$$

In GNSS, with the parallel code search, the circular correlation implies to compute the conjugate of the FFT of the local code. Since the local code is real, using (B.3), we can avoid to perform the conjugate operation, as shown in Fig. B.4 (this was already shown in Fig. A.2d without assuming h_n real). This is not a significant reduction of the complexity, but it's still better than nothing, and the additional multiplication by the factor N is not compulsory since it is just a question of normalization (moreover, for Altera FPGAs, this factor is not considered in the IFFT computation, so the multiplication by N is not required).

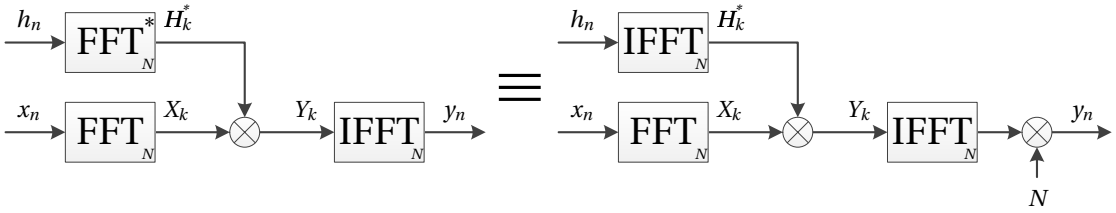


Figure B.4: Circular correlation of two sequences computed by FFT when h_n is real.

C Estimation of the resources for an implementation on FPGAs

In this appendix, we provide the details of the resource usage estimates of different functions, and then evaluate the resources for the acquisition implementation presented in Chapter 3. The estimates are based on FPGAs from Altera, and are given in terms of logic, memory and DSP elements. Among the Altera FPGAs, we find two different basis blocks for the logic, LEs (logical elements) used in the Cyclone and old Stratix series, and ALMs (adaptive look-up tables) used in recent Cyclone, Arria, and Stratix series. An LE consists of one register and a 4-input LUT, whereas an ALM consists of two registers and two 4-input ALUTs (adaptive LUTs). The conversion ratio stated by Altera is 1 ALM = 2.5 LEs, but it is not an exact formula that works all the time. According to our experience the ratio tends to be rather 1 ALM = 2 LEs most of the time, which is the ratio of the number of register. In order to obtain the most accurate results possible, the estimate of the different functions is done for both basis blocks.

The formulas provided here are empirical and obtained by analysis and verifications of compilation. Implementation inside a complete system would affect the real resources usage as well as the different optimizations performed during compilation (e.g. maximizing the clock frequency or minimizing the area). Note that the most important estimates are those of duplicated functions (or those linked to duplication), namely the code mixer, the coherent accumulator, the multiplexer, the FFT, and the coherent and noncoherent memory-based accumulators, which are not the most difficult functions to estimate.

For the following, we use $L_.$ to denote the resources in terms of logical blocks, $M_.$ for the resources in terms of memory blocks, $D_.$ for the resources in terms of DSP blocks, and $R_.$ for the resolution of a signal in bit.

C.1 Estimation of the resources

C.1.1 Resource estimates of blocks

Carrier Generator

The carrier generator is composed of an NCO and a mapping to generate sine and cosine waveforms. Taking a 32-bit counter, the NCO thus needs 64 bits (32 bits for the counter increment and 32 bits for the counter value). The mapping is a very simple combinatorial function, requires nothing or just a few elements, and is neglected here. The resources can thus be estimated as

$$\begin{aligned} L_{CaGe} &= 64 \text{ LEs} \\ &= 32 \text{ ALMs.} \end{aligned} \tag{C.1}$$

Carrier mixer

The carrier mixer is composed of four mixers, one adder, and one subtractor (this is for a complex input signal; if the input signal is real, only two mixers are required). The resolution after the adder and the subtracter is denoted R_0, and the resolution at the output of the mixers is then $R_0 - 1$. This directly provides the number of LEs, but the number of ALMs is the same as the number of LEs for the mixers. The resources can thus be estimated as

$$\begin{aligned} L_{CaMi} &= 2R_0 + 4(R_0 - 1) = 6R_0 - 4 \text{ LEs} \\ &= \ R_0 + 4(R_0 - 1) = 5R_0 - 4 \text{ ALMs.} \end{aligned} \tag{C.2}$$

Code generator

Like for the carrier generator, the code generator is composed of an NCO, plus a memory where the code is stored. This requires more logic elements to access the memory. If several shifted versions have to be generated (denoted as N_B in the formula since it corresponds to the number of branches in the implementations), this will require one register per delay. The resources can thus be estimated as

$$\begin{aligned} L_{CoGe} &= N_B + 64 + \log_2(N_{chip}) \text{ LEs} \\ &= \frac{N_B}{2} + 32 + \log_4(N_{chip}) \text{ ALMs.} \end{aligned} \tag{C.3}$$

Regarding the memory we need as many bits as there are chips in the code (insofar as the code has only two levels):

$$M_{CoGe} = N_{chip} \text{ bits.} \tag{C.4}$$

Code mixer and logic-based coherent accumulator

These blocks are used only in the SS and PFS implementations. The code mixer consists in inverting the I and Q signals according to the value of the code, and it is followed by a complex accumulator. In order to optimize the implementation, both blocks can be combined to build an accumulator that adds or subtracts the input value according to the value of the code. This optimization works well with ALM-based FPGAs since it does not require more resources than the accumulator alone. However, for LE-based FPGAs, it uses slightly more resources than the two blocks apart, so in this case, it is better to keep them separate. The accumulators also need a signal to start/restart the integration. The resources can thus be estimated as

$$\begin{aligned} L_{CoMi} &= 2R_0 \text{ LEs} \\ L_{CoAcc} &= 2R_C + 1 \text{ LEs} \\ L_{CoMiAcc} &= R_C + 0.5 \text{ ALMs.} \end{aligned} \tag{C.5}$$

Multiplexer

This block is used only in the SS and PFS implementations. The multiplexer is fully combinatorial, consequently, its resources have been evaluated empirically. The resources with N_B inputs of R_C bits can be estimated as

$$\begin{aligned} L_{MUX} &= \frac{N_B R_C}{0.75} \text{ LEs} \\ &= \frac{N_B R_C}{1.5} \text{ ALMs.} \end{aligned} \tag{C.6}$$

Magnitude computation

There are many possible algorithms for the computation of the magnitude. Here, we consider the Robertson approximation (Robertson [1971], Lyons [2010]). The estimate is obtained empirically, and it is a piecewise function that depends on the resolution of the input R.

$$\begin{aligned} L_{Mag} &= 3R + f_1(R) \text{ LEs} \\ &= 1.5R + f_2(R) \text{ ALMs.} \end{aligned} \tag{C.7}$$

with

$$f_1(R) = \begin{cases} 33, & \text{for } 10 \leq R \leq 16 \\ 67, & \text{for } 17 \leq R \leq 32 \\ 99, & \text{for } 33 \leq R \leq 49 \end{cases}, \tag{C.8}$$

and

$$f_2(R) = \begin{cases} 22, & \text{for } 12 \leq R \leq 20 \\ 42, & \text{for } 21 \leq R \leq 40 \end{cases}. \tag{C.9}$$

Complex multiplier

This function is used only in the PCS implementation. It consists of four multiplications and addition/subtraction. This is done by a DSP block. If the resolution of the input is less than or equal to the basis DSP elements (18 bits for Altera FPGAs), it requires four blocks, otherwise, it requires sixteen blocks.

$$D_{CMul} = \begin{cases} 4, & \text{for } R \leq 18 \\ 16, & \text{for } 18 > R \end{cases} \text{ DSP elements.} \tag{C.10}$$

Ping-pong buffer

This function is used only in the PFS implementation. It is composed of two memories. The number of addresses of each buffer corresponds to the number of branches multiplied by the number of signal points in the FFT, and four address buses are needed to write and read both buffers.

$$\begin{aligned} L_{PPB} &= 4\log_2(N_B N_{FFT,S}) \text{ LEs} \\ &= 2\log_2(N_B N_{FFT,S}) \text{ ALMs.} \end{aligned} \tag{C.11}$$

The number of bits needed corresponds to the number of addresses multiplied by four times the resolution of the input signal (I and Q path, in two memories to avoid the overwriting of data not yet read).

$$M_{PPB} = 4N_B N_{FFT,S} R_C \text{ bits.} \tag{C.12}$$

Memory-based coherent accumulator

This function is used only in the PCS implementation. It consists, for each signal path (I and Q), of a memory, an adder, and a 2-input multiplexer. We also count the read and write address buses needed to access the memory.

$$\begin{aligned} L_{CoAcc} &= 4R_C + 2\log_2(N_{FFT}) \text{ LEs} \\ &= 2R_C + \log_2(N_{FFT}) \text{ ALMs.} \end{aligned} \tag{C.13}$$

The number of bits corresponds to the number of points in the FFT multiplied by twice the resolution of the input signal (I and Q path).

$$M_{CoAcc} = 2N_{FFT}R_C \text{ bits.} \tag{C.14}$$

Memory-based non-coherent accumulator

This is the same block as the coherent accumulator except that there is now only one input signal instead of two.

$$\begin{aligned} L_{NoCoAcc} &= 2R_{NC} + 2\log_2(N_@) \text{ LEs} \\ &= R_{NC} + \log_2(N_@) \text{ ALMs,} \end{aligned} \tag{C.15}$$

where $N_@$ is the number of addresses and is defined as

$$N_@ = \begin{cases} N_B, & \text{for the SS implementation} \\ N_B N_{FT}, & \text{for the PFS implementation} \\ N_{FFT}, & \text{for the PCS implementation} \end{cases}. \tag{C.16}$$

The number of bits corresponds to the number of addresses multiplied by the resolution of the input signal.

$$M_{NoCoAcc} = N_@ R_{NC} \text{ bits.} \tag{C.17}$$

FFT

The resource usage of the FFT depends on a lot of parameters, and it is estimated with the Altera Mega Wizard Plug-In Manager.

C.1.2 Resource estimates of the implementations

Serial search

The functions in the SS implementation and their sizes in terms of logical blocks are summarized in Table C.1 for N_B branches. The total number of logical blocks of the SS implementation is obtained by summing all the elements of Table C.1 and is

$$\begin{aligned} L_{SS} &= N_B\left(2R_0 + \frac{10}{3}R_C + 2\right) \\ &\quad + 2\log_2(N_B) + 6R_0 + 3R_C + f_1(R_C) + 2R_{NC} + \log_2(N_{chip}) + 124 \text{ LEs} \\ &= N_B\left(\frac{5}{3}R_C + 1\right) \\ &\quad + \log_2(N_B) + 5R_0 + 1.5R_C + f_2(R_C) + R_{NC} + \log_4(N_{chip}) + 60 \text{ ALMs.} \end{aligned} \tag{C.18}$$

Function	Number of LEs	Number of ALMs
Carrier generator	64	32
Carrier mixer	$6R_0 - 4$	$5R_0 - 4$
Code generator	$N_B + 64 + \log_2(N_{chip})$	$\frac{N_B}{2} + 32 + \log_4(N_{chip})$
Code mixers	$2N_B R_0$	$N_B(R_C + 0.5)$
Coherent accumulators	$N_B(2R_C + 1)$	
Multiplexer	$\frac{N_B R_C}{0.75}$	$\frac{N_B R_C}{1.5}$
Magnitude computation	$3R_C + f_1(R_C)$	$1.5R_C + f_2(R_C)$
Non-coherent accumulator	$2R_{NC} + 2\log_2(N_B)$	$R_{NC} + \log_2(N_B)$

Table C.1: Logical resource estimates of SS implementation.

The memory blocks are used only by the non-coherent accumulator, and the total number of bits is

$$M_{SS} = N_B R_{NC} \text{ bits.} \tag{C.19}$$

Parallel frequency search

The functions in the PFS implementation and their sizes in terms of logical blocks are summarized in Table C.2 for N_B branches. The total number of logical blocks of the PFS implementation is obtained by summing all the elements of Table C.2 and is

$$\begin{aligned} L_{PFS} &= N_B\left(2R_0 + \frac{10}{3}R_C + 2\right) \\ &\quad + 6\log_2(N_B) + L_{FFT} + 4\log_2(N_{FFT,S}) + 2\log_2(N_{FT}) \\ &\quad + 6R_0 + 3R_{FFT} + f_1(R_{FFT}) + 2R_{NC} + \log_2(N_{chip}) + 124 \text{ LEs} \\ &= N_B\left(\frac{5}{3}R_C + 1\right) \\ &\quad + 3\log_2(N_B) + L_{FFT} + 2\log_2(N_{FFT,S}) + \log_2(N_{FT}) \\ &\quad + 5R_0 + 1.5R_{FFT} + f_2(R_{FFT}) + R_{NC} + \log_4(N_{chip}) + 60 \text{ ALMs.} \end{aligned} \tag{C.20}$$

The functions in the PFS implementation and their sizes in terms of memory blocks are summarized in Table C.3 for N_B branches. The total number of bits of the PFS implementation is obtained by summing all the elements of Table C.3 and is

$$M_{PFS} = N_B(4N_{FFT,S}R_C + N_{FT}R_{NC}) + M_{FFT} + N_{chip} \text{ bits.} \tag{C.21}$$

Function	Number of LEs	Number of ALMs
Carrier generator	64	32
Carrier mixer	$6R_0 - 4$	$5R_0 - 4$
Code generator	$N_B + 64 + \log_2(N_{chip})$	$\frac{N_B}{2} + 32 + \log_4(N_{chip})$
Code mixers	$2N_B R_0$	$N_B(R_C + 0.5)$
Coherent accumulators	$N_B(2R_C + 1)$	
Multiplexer	$\frac{N_B R_C}{0.75}$	$\frac{N_B R_C}{1.5}$
Ping-pong buffer	$4\log_2(N_B N_{FFT,S})$	$2\log_2(N_B N_{FFT,S})$
FFT	L_{FFT}	L_{FFT}
Magnitude computation	$3R_{FFT} + f_1(R_{FFT})$	$1.5R_{FFT} + f_2(R_{FFT})$
Non-coherent accumulator	$2R_{NC} + 2\log_2(N_B N_{FT})$	$R_{NC} + \log_2(N_B N_{FT})$

Table C.2: Logical resource estimates of PFS implementation.

Function	Number of bits
Code generator	N_{chip}
Ping-pong buffer	$4N_B N_{FFT,S} R_C$
FFT	M_{FFT}
Non-coherent accumulator	$N_B N_{FT} R_{NB}$

Table C.3: Memory resource estimates of PFS implementation.

Parallel code search

The functions in the PCS implementation and their sizes in terms of logical blocks are summarized in Table C.4 for N_B branches. The total number of logical blocks of the PCS implementation is obtained by summing all the elements of Table C.4 and is

$$\begin{aligned} L_{PCS} &= N_B\left(L_{FFT} + L_{IFFT} + 7R_C + 2R_{NC} + f_1(R_C) + 6R_0 + 60\right) \\ &\quad + L_{FFT} + 4\log_2(N_{FFT}) + \log_2(N_{chip}) + 65 \text{ LEs} \\ &= N_B\left(L_{FFT} + L_{IFFT} + 3.5R_C + R_{NC} + f_2(R_C) + 5R_0 + 28\right) \\ &\quad + L_{FFT} + 2\log_2(N_{FFT}) + \log_4(N_{chip}) + 32.5 \text{ ALMs}. \end{aligned} \tag{C.22}$$

The functions in the PCS implementation and their sizes in terms of memory blocks are summarized in Table C.5 for N_B branches. The total number of bits of the PCS implementation is obtained by summing all the elements of Table C.5 and is

$$M_{PCS} = N_B\big(M_{FFT} + M_{IFFT} + N_{FFT}(2R_C + R_{NC})\big) + M_{FFT} + N_{chip} \text{ bits}. \tag{C.23}$$

Function	Number of LEs	Number of ALMs
Carrier generator	$64N_B$	$32N_B$
Carrier mixer	$N_B(6R_0-4)$	$N_B(5R_0-4)$
Code generator	$65+\log_2(N_{chip})$	$32.5+\log_4(N_{chip})$
FFTs	$(N_B+1)L_{FFT}$	$(N_B+1)L_{FFT}$
Complex multipliers	0	0
IFFTs	$N_B L_{IFFT}$	$N_B L_{IFFT}$
Coherent accumulators	$4N_B R_C+2\log_2(N_{FFT})$	$2N_B R_C+\log_2(N_{FFT})$
Magnitude computations	$N_B\left(3R_C+f_1(R_C)\right)$	$N_B\left(1.5R_C+f_2(R_C)\right)$
Non-coherent accumulator	$N_B 2R_{NC}+2\log_2(N_{FFT})$	$N_B R_{NC}+\log_2(N_{FFT})$

Table C.4: Logical resource estimates of PCS implementation.

Function	Number of bits
Code generator	N_{chip}
FFTs	$(N_B+1)M_{FFT}$
IFFTs	$N_B M_{IFFT}$
Coherent accumulator	$2N_B N_{FFT} R_C$
Non-coherent accumulator	$N_B N_{FFT} R_{NC}$

Table C.5: Memory resource estimates of PCS implementation.

The functions in the PCS implementation and their sizes in terms of DSP blocks are summarized in Table C.6 for N_B branches. The total number of DSP elements of the PCS implementation is obtained by summing all the elements of Table C.6 and is

$$D_{PCS}=N_B\left(D_{FFT}+D_{IFFT}+D_{CMul}\right)+D_{FFT}\text{ DSP elements.} \tag{C.24}$$

C.2 Application example details

In this section, we provide the details of the application example used in Section 3.4. First, the target FPGA is presented, then for each implementation, all the values (number of accumulations, resolution of signals, and size of FFTs) are specified, and the formulas of the previous section are used to determine the number of branches that can be implemented. The results are given here and summarized in Table 3.4.

Function	Number of bits
FFTs	$(N_B+1)D_{FFT}$
Complex multipliers	$N_B D_{CMul}$
IFFTs	$N_B D_{IFFT}$

Table C.6: DSP resource estimates of PCS implementation.

C.2.1 Application with a low-cost FPGA series : Altera Cyclone III

The target device considered is the EP3C120 with the following resources,

- 119 088 LEs,
- 432 blocks of 9216 bits (9216 bits = 1 M9K),
- 288 18-bit multipliers.

Note that it is not possible to entirely fill an FPGA due to routing constraints. We then consider the use of 85 % of the logical blocks inside the FPGAs (Altera [2007]).

Therefore, taking into account that only 85 % of the FPGA logical blocks can be used, and considering the other functions in the FPGA such as the tracking channels, the management, and the processor (evaluated to about 20 000 LEs according to our experience), this gives about 80 000 LEs available for the acquisition channel. Note that the assumptions made here impact the absolute results (i.e. the performance), but not the comparison between the different implementations (i.e. the ranking).

Serial search

The parameters of the SS implementation are summarized in Table C.7. Using Eq. (C.18) we obtain

$$82N_B + 2\log_2(N_B) + 348 \leq 80\,000, \tag{C.25}$$

from which we deduce that the maximum number of branches implementable is $N_B = 971$.

Parallel frequency search

The parameters of the PFS implementation are summarized in Table C.8. For an FFT of this size, the implementation that uses the natural and bit-reversed order requires fewer logic and memory resources (but more DSP resources) than the implementation that uses only the natural order. Consequently, the evaluation is made for the first implementation (since the DSP resources are not critical with the PFS implementation).

Parameter	Value
R_0	5 bits
N_C	40 960
R_C	21 bits
N_{NC}	40
R_{NC}	27 bits

Table C.7: Parameters of the SS implementation.

Parameter	Value
R_0	5 bits
N_A	2560
$T_A = \frac{N_A}{f_S}$	625 µs
R_C	17 bits
$N_{FFT,S}$	16
N_{FFT}	32
R_{FFT}	18 bits
N_{NC}	40
R_{NC}	24 bits
L_{FFT}	4359 LEs
M_{FFT}	6 M9Ks = 55 296 bits

Table C.8: Parameters of the PFS implementation with a Cyclone III FPGA.

Using Eq. (C.20) we obtain

$$\frac{206}{3}N_B + 6\log_2(N_B) + 4716 \leq 80\,000, \tag{C.26}$$

from which we deduce that the maximum number of branches implementable due to the logic is $N_B = 1095$. Using Eq. (C.21) we obtain

$$1352N_B + 7 \times 9216 \leq 432 \times 9216, \tag{C.27}$$

from which we deduce that the maximum number of branches implementable due to the memory resources is $N_B = 2897$. Consequently, the limitation comes from the logical blocks, and the maximum number of branches is $N_B = 1095$.

Parameter		Value
R_0		5 bits
N_{FFT}		4096
R_{FFT}		18 bits
N_P		10
R_C		22 bits
N_{NC}		40
R_{NC}		28 bits
FFT with input and output in natural order	L_{FFT}, L_{IFFT}	7756 LEs
	M_{FFT}, M_{IFFT}	76 M9Ks = 700 416 bits
	D_{FFT}, D_{IFFT}	24 DSP elements
FFT with input in natural order and output in bit-reversed order	L_{FFT}	9962 LEs
	M_{FFT}	37 M9Ks = 340 992 bits
	D_{FFT}	40 DSP elements
IFFT with input in bit-reversed order and output in natural order	L_{IFFT}	10 149 LEs
	M_{IFFT}	48 M9Ks = 442 368 bits
	D_{IFFT}	40 DSP elements

Table C.9: Parameters of the PCS implementation with a Cyclone III FPGA.

Parallel code search

The parameters of the PCS implementation are summarized in Table C.9. We evaluate the number of branches for both FFT index ordering. Let's consider first the FFTs with only the natural order. Using Eq. (C.22) we obtain

$$15\,879 N_B + 7879 \leq 80\,000, \tag{C.28}$$

from which we deduce that the maximum number of branches implementable due to the logic is $N_B = 4$. Using Eq. (C.23) we obtain

$$1\,695\,744 N_B + 77 \times 9216 \leq 432 \times 9216, \tag{C.29}$$

from which we deduce that the maximum number of branches implementable due to the memory resources is $N_B = 1$. Using Eq. (C.24) we obtain

$$52 N_B + 24 \leq 288, \tag{C.30}$$

from which we deduce that the maximum number of branches implementable due to the DSP resources is $N_B = 5$. Consequently, the limitation comes from the memory resources.

Now, let us consider the FFTs with the natural and bit-reversed order. Using Eq. (C.22) we obtain

$$20\,478 N_B + 10\,085 \leq 80\,000, \tag{C.31}$$

from which we deduce that the maximum number of branches implementable due to the logic is $N_B = 3$. Using Eq. (C.23) we obtain

$$1\,078\,272 N_B + 38 \times 9216 \leq 432 \times 9216, \tag{C.32}$$

from which we deduce that the maximum number of branches implementable due to the memory resources is $N_B = 3$. Using Eq. (C.24) we obtain

$$84 N_B + 40 \leq 288, \tag{C.33}$$

from which we deduce that the maximum number of branches implementable due to the DSP resources is $N_B = 2$. In this case, the limitation comes from the DSP resources. Thus, finally, the FFT using different orders is preferable, and the maximum number of branches is $N_B = 2$, the limitation coming from the DSP blocks.

C.2.2 Application with a high-end FPGA series : Altera Stratix III

The target device considered is the EP3SE260 with the following resources,

- 135 200 ALMs,
- 864 blocks of 9216 bits (9216 bits = 1 M9K),
- 48 blocks of 147 456 bits (= 1 M144K = 16 M9K),
- 768 18-bit multipliers.

The remark made before regarding the space in the FPGA remains valid here, and we consider that 105 000 ALMs are available for the acquisition channel. The number of accumulations and the resolution of signals are identical to those already indicated and are not repeated here.

Serial search

Using Eq. (C.18) we obtain

$$36 N_B + \log_2(N_B) + 191 \leq 105\,000, \tag{C.34}$$

Parameter	Value
L_{FFT}	1790 ALMs
M_{FFT}	2 M9Ks = 18 432 bits

Table C.10: Parameters of the PFS implementation with a Stratix III FPGA.

FFT with input and output in natural order	L_{FFT}, L_{IFFT}	3806 ALMs
	M_{FFT}, M_{IFFT}	76 M9Ks = 700 416 bits
	D_{FFT}, D_{IFFT}	24 DSP elements
FFT with input in natural order and output in bit-reversed order	L_{FFT}	5083 ALMs
	M_{FFT}	31 M9Ks = 285 696 bits
	D_{FFT}	40 DSP elements
IFFT with input in bit-reversed order and output in natural order	L_{IFFT}	5146 ALMs
	M_{IFFT}	42 M9Ks = 387 072 bits
	D_{IFFT}	40 DSP elements

Table C.11: Parameters of the PCS implementation with a Stratix III FPGA.

from which we deduce that the maximum number of branches implementable is $N_B = 2911$.

Parallel frequency search

The parameters of the FFT are summarized in Table C.10. Using Eq. (C.20) we obtain

$$\frac{88}{3}N_B + 3\log_2(N_B) + 1970 \leq 105\,000, \tag{C.35}$$

from which we deduce that the maximum number of branches implementable due to the logic is $N_B = 3511$. Using Eq. (C.21) we obtain

$$1352N_B + 3 \times 9216 \leq 1632 \times 9216, \tag{C.36}$$

from which we deduce that the maximum number of branches implementable due to the memory resources is $N_B = 11\,104$. Consequently, the limitation comes from the logical blocks. Due to the limitation in the multiplexing described in Section 3.3, it is necessary to use three multiplexer chains. Taking this into account, the maximum number of branches is $N_B = 3385$.

Parallel code search

The parameters of the FFT are summarized in Table C.11. Let's consider first the FFTs with only the natural order. Using Eq. (C.22) we obtain

$$7812N_B + 3868 \leq 105\,000, \tag{C.37}$$

from which we deduce that the maximum number of branches implementable due to the logic is $N_B = 12$. Using Eq. (C.23) we obtain

$$1\,695\,744N_B + 77 \times 9216 \leq 1632 \times 9216, \tag{C.38}$$

from which we deduce that the maximum number of branches implementable due to the memory resources is $N_B = 8$. Using Eq. (C.24) we obtain

$$52N_B + 24 \leq 768, \tag{C.39}$$

from which we deduce that the maximum number of branches implementable due to the DSP resources is $N_B = 14$. Consequently, the limitation comes from the memory resources.

Now, let us consider the FFTs with the natural and bit-reversed order. Using Eq. (C.22) we obtain

$$10\,429N_B + 5145 \leq 105\,000, \tag{C.40}$$

from which we deduce that the maximum number of branches implementable due to the logic is $N_B = 9$. Using Eq. (C.23) we obtain

$$967\,680N_B + 32 \times 9216 \leq 432 \times 9216, \tag{C.41}$$

from which we deduce that the maximum number of branches implementable due to the memory resources is $N_B = 15$. Using Eq. (C.24) we obtain

$$84N_B + 40 \leq 768, \tag{C.42}$$

from which we deduce that the maximum number of branches implementable due to the DSP resources is $N_B = 8$. In this case, the limitation comes from the DSP resources. Finally, both FFT types gives the same maximum number of branches, $N_B = 8$, in one case the limitation comes from the memory and on the other case it comes from the DSP blocks.

The summary of the number of branches can be found in Table 3.4 on page 68.

Bibliography

C. Adrados, I. Girard, J.-P. Gendner, and G. Janeau. Global positioning system (GPS) location accuracy improvement due to selective availability removal. *Comptes Rendus Biologies*, 325 (2):165 – 170, February 2002.

D. Akopian. Fast FFT based GPS satellite acquisition methods. *IEE Proceedings Radar, Sonar and Navigation*, 152(4):277–286, August 2005.

Altera. *Designing and using FPGAs for double-precision floating-point math*, August 2007. White paper.

Altera. *Stratix III device handbook*, March 2011.

Altera. *Cyclone III device handbook*, August 2012.

Altera. *FFT MegaCore function user guide*, November 2013.

M. Anghileri, M. Paonni, S. Wallner, J.A. Ávila Rodríguez, and B. Eissfeller. Estimating the time-to-first-fix for GNSS signals. Theory and simulation results. In *Fourth European Workshop on GNSS Signals and Signal Processing*, December 2009.

M. Anghileri, M. Paonni, S. Wallner, J.A. Ávila Rodríguez, and B. Eissfeller. Ready to navigate ! a methodology for the estimation of the time-to-first-fix. *Inside GNSS*, 5(2):47–56, March/April 2010.

A.S. Ayaz. Analysis of differential acquisition methods by using Monte-Carlo simulations. In *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS)*, pages 1922–1930, September 2005.

A.S. Ayaz, T. Pany, and B. Eissfeller. Assessment of GNSS signal acquisition sensitivities for indoor and urban scenarios. In *IEEE 21st International Symposium on Personal, Indoor and Mobile Radio Communications Workshops (PIMRC Workshops)*, pages 223–227, September 2010.

A.T. Balaei and D.M. Akos. Cross correlation impacts and observations in GNSS receivers. *NAVIGATION, Journal of The Institute of Navigation*, 58(4):323–333, Winter 2010.

J.W. Betz. Binary offset carrier modulation for radionavigation. *NAVIGATION, Journal of The Institute of Navigation*, 48(4):227–246, Winter 2001.

R.E. Blahut. *Fast algorithms for signal processing*. Cambridge University Press, 2nd edition, 2010.

L. Bluestein. A linear filtering approach to the computation of discrete Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 18(4):451–455, December 1970.

D. Borio. *A statistical theory for GNSS signal acquisition*. PhD thesis, Politecnico di Torino, 2008.

D. Borio. M-sequence and secondary code constraints for GNSS signal acquisition. *IEEE Transactions on Aerospace and Electronic Systems*, 47(2):928–945, April 2011.

D. Borio and D. Akos. Noncoherent integrations for GNSS detection : Analysis and comparisons. *IEEE Transactions on Aerospace and Electronic Systems*, 45(1):360–375, January 2009.

D. Borio, L. Camoriano, and L. Lo Presti. Impact of the acquisition searching strategy on the detection and false alarm probabilities in a CDMA receiver. In *IEEE/ION Position, Location, And Navigation Symposium*, pages 1100–1107, April 2006.

D. Borio, L. Camoriano, and L. Lo Presti. Impact of GPS acquisition strategy on decision probabilities. *IEEE Transactions on Aerospace and Electronic Systems*, 44(3):996–1011, July 2008a.

D. Borio, M. Fantino, and L. Lo Presti. The impact of the Galileo signal in space in the acquisition system. In E. Del Re and M. Ruggieri, editors, *Satellite Communications and Navigation Systems*, Signals and Communication Technology, pages 151–167. Springer US, 2008b.

D. Borio, C. Gernot, F. Macchi, and G. Lachapelle. The output SNR and its role in quantifying GNSS signal acquisition performance. In *European Navigation Conference (ENC-GNSS)*, April 2008c.

K. Borre, D.M. Akos, N. Bertelsen, P. Rinder, and S.H Jensen. *A software-defined GPS and Galileo receiver. Single-frequency approach*. Applied and Numerical Harmonic Analysis. Birkhäuser Boston, 2007.

E.O. Brigham. *The fast Fourier transform and its applications*. Prentice Hall, 1988.

C.S. Burrus. *Fast Fourier transforms*. 2012.

C.S. Burrus and T.W. Parks. *DFT/FFT and convolution algorithms and implementation*. Wiley, 1985.

V. Capuano, C. Botteron, and P.-A. Farine. GNSS performances for MEO, GEO and HEO. In *64th International Astronautical Congress*, October 2013.

F. Chastellain. *Dual-frequency RF front-ends for GNSS receivers*. PhD thesis, École polytechnique fédérale de Lausanne, 2010.

F. Chastellain, C. Botteron, and P. Farine. Looking inside modern receivers. *IEEE Microwave Magazine*, 12(2):87–98, 2011.

X. Chen, C.G. Parini, B. Collins, Y. Yao, and M.U. Rehman. *Antennas for global navigation satellite systems*. Wiley, 2012.

U. Cheng, W.J. Hurd, and J.I. Statman. Spread-spectrum code acquisition in the presence of Doppler shift and data modulation. *IEEE Transactions on Communications*, 38(2):241–250, 1990.

E. Chu. *Discrete and continuous Fourier transforms: Analysis, applications and fast algorithms*. Chapman and Hall/CRC, 2008.

J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, April 1965.

J. Curran, D. Borio, and C.C. Murphy. Front-end filtering and quantisation effects on GNSS signal processing. In *1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE 2009)*, pages 227–231, May 2009.

J.T. Curran. *Weak signal digital GNSS tracking algorithms*. PhD thesis, National University of Ireland, Cork, 2010.

J.T. Curran, D. Borio, G. Lachapelle, and C.C. Murphy. Reducing front-end bandwidth may improve digital GNSS receiver performance. *IEEE Transactions on Signal Processing*, 58(4): 2399–2404, 2010.

P.J. Davis. *Circulant matrices*. Wiley, 1979.

C. Ding, D. Pei, and A. Salomaa. *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*. World Scientific Publishing Company, 1996.

A. Dion, V. Calmettes, and E. Boutillon. Reconfigurable GPS-Galileo receiver for satellite based applications. In *Proceedings of the 2008 National Technical Meeting of The Institute of Navigation*, pages 277–287, January 2008.

A. El-Rabbany. *Introduction to GPS: The global positioning system*. GNSS Technology and Applications Series. Artech House, 2nd edition, 2006.

H. Elders-Boll and U. Dettmar. Efficient differentially coherent code/Doppler acquisition of weak GPS signals. In *IEEE Eighth International Symposium on Spread Spectrum Techniques and Applications*, pages 731–735, September 2004.

P. Esteves, M. Sahmoudi, N. Ziedan, and M.-L. Boucheret. A new adaptive scheme for high-sensitivity GNSS acquisition in presence of large Doppler shifts. In *Proceedings of the 25th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2012)*, pages 28–40, September 2012.

FFTW. Pruned FFTs. http://www.fftw.org/pruned.html. Accessed the 25th February 2014.

M. Foucras, O. Julien, C. Macabiau, and B. Ekambi. A novel computationally efficient Galileo E1 OS acquisition method for GNSS software receiver. In *Proceedings of the 25th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2012)*, pages 365–383, September 2012.

M. Foucras, O. Julien, C. Macabiau, and B. Ekambi. Detailed analysis of the impact of the code doppler on the acquisition performance of new GNSS signals. In *Proceedings of the 2014 International Technical Meeting of The Institute of Navigation (ION ITM)*, January 2014a.

M. Foucras, J. Leclère, O. Julien, C. Botteron, C. Macabiau, P.-A. Farine, and B. Ekambi. Approximations in the correlation and the cross ambiguity function. 2014b. In preparation.

M. Frigo and S.G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

H.K. Garg. *Digital signal processing algorithms: Number theory, convolution, fast Fourier transforms, and applications*. Computer Science & Engineering. CRC Press, 1998.

B.C. Geiger and C. Vogel. Influence of Doppler bin width on GPS acquisition probabilities. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2570–2584, 2013.

B.C. Geiger, M. Soudan, and C. Vogel. On the detection probability of parallel code phase search algorithms in GPS receivers. In *IEEE 21st International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pages 865–870, 2010.

B.C. Geiger, C. Vogel, and M. Soudan. Comparison between ratio detection and threshold comparison for GNSS acquisition. *IEEE Transactions on Aerospace and Electronic Systems*, 48(2):1772–1779, 2012.

S. Gleason and D. Gebre-Egziabher. *GNSS applications and methods*. GNSS Technology and Applications Series. Artech House, 2009.

Robert Gold. Optimal binary sequences for spread spectrum multiplexing (corresp.). *IEEE Transactions on Information Theory*, 13(4):619–621, 1967.

M.S. Grewal, A.P. Andrews, and C.G. Bartone. *Global navigation satellite systems, inertial navigation, and integration*. Wiley, 3rd edition, 2013.

X.Y. Guo, G. Maral, and A. Marguinaud. Un algorithme rapide pour l'acquisition du synchronisme d'une séquence PN dans un système de transmission par étalement de spectre à séquence directe. In *13ème Colloque GRETSI (Groupe d'Etudes du Traitement du Signal et des Images)*, pages 425–428, September 1991a.

X.Y. Guo, G. Maral, A. Marguinaud, and R. Sauvagnac. Méthode de calcul rapide de convolution entre deux séquences dont l'une est binaire. *Annales des Télécommunications*, 46(3-4):181–190, March/April 1991b.

H. Hassanieh, F. Adib, D. Katabi, and P. Indyk. Faster GPS via the sparse Fourier transform. In *Proceedings of the 18th annual international conference on Mobile computing and networking (Mobicom'12)*, pages 353–364, August 2012.

C Hegarty, M. Tran, and A.J. van Dierendonck. Acquisition algorithms for the GPS L5 signal. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS/GNSS 2003)*, pages 165–177, September 2003.

C.J. Hegarty. GNSS signals - An overview. In *IEEE International Frequency Control Symposium (FCS)*, May 2012.

J.K. Holmes. *Spread spectrum systems for GNSS and wireless communications*. GNSS Technology and Applications Series. Artech House, 2007.

J.K. Holmes and C.C. Chen. Acquisition time performance of PN spread-spectrum systems. *IEEE Transactions on Communications*, 25(8):778–784, August 1977.

M.M. Hoque and N. Jakowski. *Ionospheric Propagation Effects on GNSS Signals and New Correction Approaches*. InTech, 2012.

W.-C. Huang, C.-P. Li, and H.-J. Li. A computationally efficient DFT scheme for applications with a subset of nonzero inputs. *IEEE Signal Processing Letters*, 15:206–208, 2008.

L. Jacobson. *GNSS markets and applications*. GNSS Technology and Applications Series. Artech House, 2007.

C. Jayaram and C.R. Murthy. Noncoherent integration for signal detection : Analysis under model uncertainties. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2413–2430, October 2013.

S. Jeon, H. So, G. Kim, C. Kee, and K. Kwon. Bit transition cancellation signal acquisition method for modernized GPS and Galileo signal. In *Proceedings of the 24th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pages 1028–1039, September 2012.

A. Joseph. What is the difference between SNR and C/N_0 ? *Inside GNSS*, 5(6):20–25, November/December 2010.

E.D. Kaplan and C.J. Hegarty. *Understanding GPS: Principles and applications*. GNSS Technology and Applications Series. Artech House, 2nd edition, 2005.

R.L. La Valle, J.G. Garcia, P.A. Roncagliolo, and C.H. Muravchik. A practical RF front-end for high performance GNSS receivers. In *International Conference on Localization and GNSS (ICL-GNSS)*, pages 104–109, 2011.

J.-C. Lafon. Base tensorielle des matrices de Hankel (ou de Toeplitz) applications. *Numerische Mathematik*, 23(4):349–361, 1974.

R.B. Langley. Why is the GPS signal so complex ? *GPS World*, 1(3):56–59, May/June 1990.

J. Leclère, C. Botteron, and P.-A. Farine. Resource and performance comparisons for different acquisition methods that can be applied to a VHDL-based GPS receiver in standalone and assisted cases. In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 745–751, May 2010.

J. Leclère, C. Botteron, and P.-A. Farine. Improving the performance of the FFT-based parallel code-phase search acquisition of GNSS signals by decomposition of the circular correlation. In *Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012)*, pages 1406–1416, September 2012.

J. Leclère, C. Botteron, and P.-A. Farine. Modified parallel code-phase search for acquisition in presence of sign transition. In *International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–6, June 2013a.

J. Leclère, C. Botteron, and P.-A. Farine. Comparison framework of FPGA-based GNSS signals acquisition architectures. *IEEE Transactions on Aerospace and Electronic Systems*, 49(3): 1497–1518, July 2013b.

J. Leclère, C. Botteron, and P.-A. Farine. Acquisition of modern GNSS signals using a modified parallel code-phase search architecture. *Signal Processing*, 95(0):177–191, February 2014.

L. Lo Presti, X. Zhu, M. Fantino, and P. Mulassano. GNSS signal acquisition in the presence of sign transition. *IEEE Journal of Selected Topics in Signal Processing*, 3(4):557–570, August 2009.

J.B. Lozow. Analysis of direct P(Y)-code acquisition. *NAVIGATION, Journal of The Institute of Navigation*, 44(1):89–98, Spring 1977.

R.G. Lyons. *Understanding digital signal processing*. Prentice Hall, 3rd edition, 2010.

C. Macabiau, L. Ries, F. Bastide, and J.-L. Issler. GPS L5 receiver implementation issues. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, pages 153–164, September 2003.

G. MacGougan, G. Lachapelle, R. Nayak, and A. Wang. Overview of GNSS signal degradation phenomena. In *Proceedings of the International Symposium on Kinematic Systems in Geodesy (KIS 2001)*, pages 87–100, June 2001.

H. Mathis, P. Flammant, and A. Thiel. An analytic way to optimize the detector of a post-correlation FFT acquisition algorithm. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, pages 689–699, September 2003.

C. Maxfield. *FPGAs: Instant access*. Newnes, 2008.

Microsemi. *LogiCORE IP fast Fourier transform*, September 2013.

P. Misra and P. Enge. *Global positioning system: Signals, measurements, and performance*. Ganga-Jamuna Press, 2nd edition, 2011.

B. Motella and L. Lo Presti. The math of ambiguity: What is the acquisition ambiguity function and how is it expressed mathematically ? *Inside GNSS*, 5(4):20–28, June 2010.

Z.-J. Mou and P. Duhamel. Short-length FIR filters and their use in fast nonrecursive filtering. *IEEE Transactions on Signal Processing*, 39(6):1322–1332, June 1991.

M.K. Ng. Circulant and skew-circulant splitting methods for toeplitz systems. *Journal of Computational and Applied Mathematics*, 159(1):101–108, October 2003.

H.J. Nussbaumer. *Fast Fourier transform and convolution algorithms*. Springer Series in Information Sciences. Springer Berlin Heidelberg, 2nd edition, 1982.

C. O'Driscoll. *Performance analysis of the parallel acquisition of weak GPS signals*. PhD thesis, National University of Ireland, Cork, 2007.

A.V. Oppenheim and R.W. Schafer. *Discrete-time signal processing*. Prentice Hall, 3rd edition, 2009.

T. Pany, B. Riedl, J. Winkel, T. Wörz, R. Schweikert, H. Niedermeier, S. Lagrasta, G. López-Risueño, and D. Jiménez-Baños. Coherent integration time: The longer, the better. *Inside GNSS*, 4(6):52–61, November/December 2009.

M. Paonni, M. Anghileri, S. Wallner, J.A. Ávila Rodríguez, and B. Eissfeller. Methodologies for the determination of the minimum required carrier to noise ratio to receive GNSS signals. In *Fourth European Workshop on GNSS Signals and Signal Processing*, December 2009.

M. Paonni, M. Anghileri, S. Wallner, J.A. Ávila Rodríguez, and B. Eissfeller. Performance assessment of GNSS signals in terms of time to first fix for cold, warm and hot start. In *Proceedings of the International Technical Meeting of the Institute of Navigation (ION ITM)*, pages 1051–1066, January 2010.

K.K. Parhi. *VLSI digital signal processing systems: Design and implementation*. Wiley, 1999.

S.H. Park, I.H. Choi, S.J. Lee, and Y.B. Kim. A novel GPS initial synchronization scheme using decomposed differential matched filter. In *Proceedings of the 2002 National Technical Meeting of The Institute of Navigation*, pages 246–253, January 2002.

D.A. Parker and K.K. Parhi. Low-area/power parallel FIR digital filter implementations. *Journal of VLSI signal processing systems for signal, image and video technology*, 17(1):75–92, 1997.

J.G. Proakis and D.G. Manolakis. *Digital signal processing: Principles, algorithms, and applications*. Prentice Hall, 4th edition, 2006.

J.G. Proakis and D.K. Manolakis. *Digital signal processing*. Prentice Hall, 4th edition, 2007.

M. L. Psiaki. Block acquisition of weak GPS signals in a software receiver. In *Proceedings of the 14th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 2001)*, pages 2838–2850, September 2001.

S.U. Qaisar and A.G. Dempster. An analysis of L1-C/A cross correlation & acquisition effort in weak signal environments. In *International Global Navigation Satellite Systems (IGNSS) Symposium*, December 2007.

S.U. Qaisar, N.C. Shivaramaiah, A.G. Dempster, and C. Rizos. Filtering IF samples to reduce the computational load of frequency domain acquisition in GNSS receivers. In *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS)*, pages 236 – 243, September 2008.

C.M. Rader. Discrete Fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, June 1968.

B. Rama Rao, W. Kunysz, R. Fante, and K. McDonald. *GPS/GNSS antennas*. GNSS Technology and Applications Series. Artech House, 2012.

M.V.G. Rao and D.V. Ratnam. Faster GPS/IRNSS acquisition via sub sampled fast Fourier transform (ssFFT) and thresholding. In *2013 Annual IEEE India Conference (INDICON)*, pages 1–4, December 2013.

G. H. Robertson. A fast amplitude approximation for quadrature pairs. *Bell System Technical Journal*, 50(8):2849–2852, 1971.

A. Ruegamer, F. Foerster, M. Stahl, and G. Rohmer. A flexible and portable multiband GNSS front-end system. In *Proceedings of the 25th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2012)*, pages 2378–2389, September 2012.

P.K. Sagiraju, S. Agaian, and D. Akopian. Reduced complexity acquisition of gps signals for software embedded applications. *IEE Proceedings - Radar, Sonar and Navigation*, 153(1): 69–78, February 2006.

C. Sajabi, C.-I.H. Chen, D.M. Lin, and J.B.Y Tsui. FPGA frequency domain based GPS coarse acquisition processor using FFT. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference (IMTC)*, pages 2353–2358, April 2006.

N.C. Shivaramaiah, A.G. Dempster, and C. Rizos. Exploiting the secondary codes to improve signal acquisition performance in Galileo receivers. In *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008)*, pages 1497–1506, September 2008.

J.O. Smith. *Mathematics of the discrete Fourier transform (DFT)*. W3K Publishing, 2nd edition, 2007.

S.W. Smith. *Digital signal processing: a practical guide for engineers and scientists*. Newnes, 2002.

H.V. Sorensen and C.S. Burrus. Efficient computation of the DFT with only a subset of input or output points. *IEEE Transactions on Signal Processing*, 41(3):1184–1200, March 1993.

H.V. Sorensen, D.L. Jones, M. Heideman, and C.S. Burrus. Real-valued fast fourier transform algorithms. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(6):849–863, June 1987.

F. Soualle. Correlation and randomness properties of the spreading coding families for the current and future GNSSs. In *Fourth European Workshop on GNSS signal and signal processing (GNSS Signals 2009)*, December 2009.

F. Soualle, M. Soellner, S. Wallner, J.Á. Ávila Rodríguez, G.W. Hein, B. Barnes, T. Pratt, L. Ries, J. Winkel, C. Lemenager, and P. Erhard. Spreading code selection criteria for the future GNSS Galileo. In *Proceedings of the European Navigation Conference GNSS*, July 2005.

J.A. Starzyk and Z. Zhu. Averaging correlation for C/A code acquisition and tracking in frequency domain. In *Proceedings of the 44th IEEE Midwest Symposium on Circuits and Systems (MWSCAS)*, volume 2, pages 905–908, August 2001.

K. Sun and L. Lo Presti. Bit sign transition cancellation method for gnss signal acquisition. *Journal of Navigation*, 65:73–97, January 2012.

T.H. Ta, N.C. Shivaramaiah, A.G. Dempster, and L. Lo Presti. Significance of cell-correlation phenomenon in GNSS matched filter acquisition engines. *IEEE Transactions on Aerospace and Electronic Systems*, 48(2):1264–1286, April 2012.

Texas Instruments. *FFT implementation on the TMS320VC5505, TMS320C5505, and TMS320C5515 DSPs*, January 2013.

S. Thombre, J. Raasakka, H. Hurskainen, J. Nurmi, M. Valkama, and S. Lohan. Local oscillator phase noise effects on phase angle component of GNSS code correlation. In *International Conference on Localization and GNSS (ICL-GNSS)*, pages 110–115, 2011.

J.B.-Y. Tsui. *Fundamentals of global positioning system receivers : a software approach*. Series in Microwave & Optical Engineering. Wiley, 2nd edition, 2005.

S. Turunen. Network assistance : What will new GNSS signals bring to it ? *Inside GNSS*, 2(3): 35–41, Spring 2007.

S. Turunen. *Weak signal acquisition in satellite positioning*. PhD thesis, Tampere University of Technology, 2010.

P.P. Vaidyanathan. *Multirate systems and filter banks*. Prentice Hall, 1993.

P.P Vaidyanathan, S.-M. Phoong, and Y.-P. Lin. *Signal processing and optimization for transceiver systems*. Cambridge University Press, 2010.

F. van Diggelen. *A-GPS: Assisted GPS, GNSS, and SBAS*. GNSS Technology and Applications Series. Artech House, 2009.

F. van Diggelen. Why GPS will continue to dominate consumer GNSS. *Inside GNSS*, 9(2):30–41, March/April 2014.

J.Á. Ávila Rodríguez. *On generalized signal waveforms for satellite navigation*. PhD thesis, University FAF Munich, 2008.

J.Á. Ávila Rodríguez, G.W. Hein, S. Wallner, A.R. Pratt, J.I.R. Owen, J.-L. Issler, J.W. Betz, C.J. Hegarty, S. Lenahan, J.J. Rushanan, A.L. Kraay, and T.A. Stansell. MBOC : The new optimized spreading modulation recommended for Galileo E1 OS and GPS L1C. In *ESA Navitec 2006*, December 2006.

A.J. Viterbi. *CDMA: Principles of spread spectrum communication*. Prentice Hall, 1995.

S. Wallner, J.Á. Ávila Rodríguez, G.W. Hein, and J.J. Rushanan. Galileo E1 OS and GPS L1C pseudo random noise codes - requirements, generation, optimization and comparison -. In *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007)*, pages 1549–1563, September 2007.

J.J.H. Wang. Antennas for global navigation satellite system (GNSS). *Proceedings of the IEEE*, 100(7):2349–2355, July 2012.

S. Winograd. *Arithmetic complexity of computations*. Society for Industrial and Applied Mathematics, 1980.

Xilinx. *CoreFFT handbook*, October 2013.

C. Yang, C Hegarty, and M. Tran. Acquisition of the GPS L5 signal using coherent combining of I5 and Q5. In *Proceedings of the 17th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2004)*, pages 2184–2195, September 2004.

N.I. Ziedan. Acquisition and fine acquisition of weak GPS L2C and L5 signals under high dynamic conditions for limited-resource applications. In *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS)*, pages 1577–1588, September 2005.

N.I. Ziedan and J.L. Garrison. Unaided acquisition of weak GPS signals using circular correlation or double-block zero padding. In *Position Location and Navigation Symposium (PLANS)*, pages 461–470, April 2004.

R.E. Ziemer and W.H. Tranter. *Principles of communications*. Wiley, 6th edition, 2008.

Jérôme Leclère – Ph.D. in Electronics and Signal Processing

Nationality : French
E-mail : jerome.leclere@epfl.ch
Phone : +41 21 695 42 15

Address : EPFL STI IMT ESPLAB
Rue de la Maladière 71b
2002 Neuchâtel, Switzerland

Education

2009 – 2014 **Ph.D. in Global Navigation Satellite System (GNSS)**
Title : *Resource-efficient parallel acquisition architectures for modernized GNSS signals*
École Polytechnique Fédérale de Lausanne (EPFL), Electronics and Signal Processing Laboratory (ESPLAB), Neuchâtel, Switzerland

2005 - 2008 **Master degree in Electronics and Signal Processing**
ENSEEIHT, Toulouse, France

2003 - 2005 **DUT in Electronics (equivalent to an HND)**
IUT de Cachan, Université Paris-Sud XI, France

Professional experiences

2009 – 2014 **Ph.D. Research**
- Development of an assisted GPS L1 C/A receiver on an Altera FPGA.
- Research of fast correlation algorithms for GNSS signals acquisition.

2008 **Internship,** ESPLAB, University of Neuchâtel, Switzerland. 5 months.
- Development of a Matlab-based image processing software for the extraction of neurons and electrodes from fluorescent images.

2007 **Internship,** Vocally, Paris, France. 2 months.
- Study of the performance of different digital signal processors that could be used for a voice processing software.

Skills

Signal processing	DSP, GNSS, CDMA systems
Digital systems	FPGAs (Altera), microprocessors (Nios II), microcontroller (PIC)
Software	Matlab, Quartus II, ModelSim, MPLAB, Office suite, LaTeX
Programming	VHDL, C, C++ (Qt)
Languages	French (mother tongue), English (spoken and written), Japanese (1 year course)

External activities

Sports Tchoukball, swimming, hiking.
Readings History of mathematics, ethics, society issues.

Journal publications

As main author :

J. Leclère, C. Botteron and P.-A. Farine. *Acquisition of modern GNSS signals using a modified parallel code-phase search architecture.* Signal Processing, vol. 95, pp. 177–191, Feb. 2014.

J. Leclère, C. Botteron and P.-A. Farine. *Comparison framework of FPGA-based GNSS signals acquisition architectures.* IEEE Transactions on Aerospace and Electronic Systems, vol. 49, no. 3, pp. 1497–1518, July 2013.

As co-author :

C. Botteron, N. Dawes, J. Leclère, J. Skaloud, S.V. Weijs and P.-A. Farine. *Soil moisture & snow properties determination with GNSS in alpine environments : challenges, status, and perspectives*. Remote Sensing, vol. 5, no. 7, pp. 3516–3543, July 2013.

Y. Tawk, A. Jovanovic, P. Tomé, J. Leclère, C. Botteron, P.-A. Farine, R. Riem-Vis, and B. Spaeth. *A new movement recognition technique for flight mode detection*, International Journal of Vehicular Technology, vol. 2013, pp. 1–18, 2013.

K. Sheridan, T. Dyjas, C. Botteron, J. Leclère, F. Dominici, G. Marucco. *Demands of the road : An assisted-GNSS solution uses the EGNOS data access service*, GPS World, vol. 22, no. 3, pp. 28–35, March 2011.

Conference publications

As main author :

J. Leclère, C. Botteron and P.-A. Farine. *Modified parallel code-phase search for acquisition in presence of sign transition*. ICL-GNSS, Torino, Italy, June 2013.

J. Leclère, C. Botteron and P.-A. Farine. *Improving the performance of the FFT-based parallel code-phase search acquisition of GNSS signals by decomposition of the circular correlation*. ION GNSS, Nashville, USA, Sept. 2012.

J. Leclère, C. Botteron and P.-A. Farine. *Resource and performance comparisons for different acquisition methods that can be applied to a VHDL-based GPS receiver in standalone and assisted cases*. IEEE/ION PLANS, Palm Springs, USA, May 2010.

As co-author :

S. Reboul, C. Botteron, R. Sanfelix, M. Angel, G. Stienne, J. Leclère, J.-B. Choquel, M. Benjelloun and P.-A. Farine. *Normalized GNSS interference pattern technique*. URSI Commission F Microwave Signatures, Espoo, Finland, Oct. 2013.

Y. Tawk, A. Jovanovic, J. Leclère, C. Botteron and P.-A. Farine. *A new FFT-based algorithm for secondary code acquisition for Galileo signals*. IEEE VTC Fall, San Francisco, USA, Sept. 2011.

K. Sheridan, T. Dyjas, C. Botteron, J. Leclère, F. Dominici, and G. Marucco. *An assisted-GNSS solution for demanding road applications using the EGNOS data access system (EDAS)*. ENC-GNSS, Braunschweig, Germany, Oct. 2010.

K. Sheridan, D. Wells, C. Botteron, J. Leclère, F. Dominici, and A. Defina. *An assisted-GNSS solution for demanding road applications using the EGNOS data access system (EDAS)*. Toulouse Space Show'10, Toulouse, France, 2010.