COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université Bretagne Sud

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE SUD

ÉCOLE DOCTORALE Nº 644
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Électronique*

Par

Camille MONIÈRE

Implémentation Temps-Réel d'un Récepteur Quasi Cyclic Short Packet

Real-Time implementation of Quasi-Cyclic Short Packet Receiver

Thèse présentée et soutenue à Lorient, le 04/01/2022
Unité de recherches : Lab-STICC — UMR 6285 CNRS, Université de Bretagne Sud
IMS — UMR 5218 CNRS, Bordeaux INP
Thèse Nº : 650

Rapporteurs avant soutenance :

Matthieu GAUTIER	Maitre de Conférence HDR (Université Rennes 1)
Claire GOURSAUD	Maitre de Conférence HDR (INRIA, INSA Lyon)

Composition du Jury :

Président :	Bertrand GRANADO	Professeur des Universités (Sorbonne Université)
Examinateurs :	Christophe JEGO	Professeur des Universités (IMS, ENSEIRB-MATMECA/Bordeaux INP)
	Claire GOURSAUD	Maitre de Conférence HDR (INRIA, INSA Lyon)
	Matthieu GAUTIER	Maitre de Conférence HDR (Université Rennes 1)
Dir. de thèse :	Emmanuel BOUTILLON	Professeur des Universités (Lab-STICC, Université de Bretagne Sud)
Co-Enc. de thèse :	Bertrand LE GAL	Maitre de Conférence (IMS, ENSEIRB-MATMECA/Bordeaux INP)

Invité(s) :

Rémi CHAUVAT	Expert Télécommunications Spatiales (KINEIS)

Remerciements

Je tiens à remercier en premier lieu mon directeur de thèse, Emmanuel Boutillon, Professeur à l'Université de Bretagne Sud,ainsi que mon co-encadrant, Bertrand Le Gal, Maitre de Conférences à l'ENSEIRB-MATMECA. Chacun m'a soutenue dans mes initiatives et m'a apporté son expertise. Leur support et leurs conseils ont été d'une grande aide dans le contexte particulier de cette thèse.

Je remercie également Matthieu Gautier et Claire Goursaud, Maitres de Conférences HDR à l'Université de Rennes 1 et à l'IRISA de Lyon respectivement, pour avoir accepté d'être mes rapporteurs de thèse. Leur appréciation du manuscrit a été formatrice, et leurs observations et remarques ont notamment permis d'améliorer et de peaufiner ce manuscrit.

Je tiens à remercier aussi Bertrand Granado et Christophe Jego, Professeurs à l'Université de la Sorbonne et à l'ENSEIRB-MATMECA respectivement, mais aussi Rémi Chauvat, Expert en Télécommunications Spatiales à KINEIS pour avoir accepté de faire partie de mon jury de soutenance de thèse. Leurs questions m'ont offert d'autres perspectives quant à mes travaux.

Je me dois de remercier Cédric Marchand, docteur et ingénieur de recherche au Lab-STICC de Lorient pour l'aide qu'il m'a apporté, notamment autour des codes correcteurs d'erreurs.

De la même façon, je remercie Kassem Saied, docteur de l'Université de Bretagne Sud. Mes travaux et les siens étaient étroitement liés et notre collaboration s'est avérée fructueuse.

Je remercie aussi Hugues Almorin, doctorant à l'Université de Bordeaux. Ses résultats m'ont permis de mieux appréhender certains aspects de mes propres recherches.

Je me dois de citer Bertrand Orvoine, Leonardo M. Obesso et Joseph Jabour pour leurs contributions techniques dans certains de mes programmes, Léa Volpin, que j'ai encadré en tant qu'ingénieure, et aussi tant d'autres, que je ne pourrais citer par manque d'espace.

Enfin, je remercie Virginie Guillet, gestionnaire au Lab-STICC, et Fabienne Prévot, responsable RH à l'IMS de Bordeaux, qui ont garanti le bon déroulement de cette thèse.

Table of Contents

Acronyms **7**

Glossary **11**

List of Figures **15**

List of Tables **18**

1 Introduction **19**

1.1 Introduction . . . 19

1.2 Manuscript outline . . . 22

2 State of the Art **25**

2.1 Internet of Things and associated challenge . . . 25

2.2 Software-defined Radio platforms . . . 29

2.3 Implementation of digital communication systems . . . 31

2.4 Conclusion . . . 35

3 Quasi-Cyclic Short Packets Communication Chain **37**

3.1 Non Binary Error Correction Codes . . . 38

3.1.1 Galois Field of order q — GF(q) . . . 39

3.1.2 Non-Binary Low Density Parity Check . . . 40

3.2 Cyclic Code Shift Keying . . . 41

3.3 QCSP System Model . . . 43

3.3.1 Transmitter . . . 43

3.3.2 Channel model . . . 44

3.3.3 Detection . . . 45

3.3.3.1 Score function . . . 45

3.3.3.2 Search grid . . . 47

3.3.3.3 Performances . . . 49

3.3.4 Synchronization 53
3.4 Conclusion 54

4 Detection Algorithmic Optimizations 57
4.1 Mitigating possible gain instability 58
4.1.1 Normalization methods 59
4.1.2 Complexity estimations 60
4.1.3 Impact on detection performances 61
4.2 Time sliding windows 64
4.2.1 Principle 65
Iterative factor precision issue 67
Architectural variations 68
4.2.2 Detection performance and complexity comparisons 70
4.2.2.1 Impact of (p_ω, p_Δ) on detection performances 70
4.2.2.2 Computational complexity comparisons 71
4.2.3 Software implementation, benchmark, and analysis 73
4.3 Conclusion 74

5 Real-Time Implementation 77
5.1 Transmission 78
5.1.1 Principles 78
5.1.2 Transmitter software implementation 80
5.1.3 Transmitter hardware implementation 82
5.2 Detection 85
5.2.1 Critical task identification 86
5.2.2 Correlator inherent parallelism 87
5.2.3 Correlator in-depth complexity study 89
5.2.4 Correlator software implementation 92
5.2.4.1 Analysis of the results 94
5.2.5 Correlator hardware implementation 96
5.2.6 Quantized model of the score processor 98
5.2.6.1 Presentation of the quantized model 99
5.2.6.2 Quantization format effect on detection performances . . 102
5.2.6.3 Preliminary implementation results 102
5.3 Conclusion 104

6 Real-Time Full-Scale Experiments **105**
6.1 Experimental setup 105
6.1.1 GNSS-enabled mobile QCSP transmitter 108
6.1.2 Real-time QCSP receiver 111
6.1.3 Experimental environment and protocol 113
6.2 Experimental results analysis 113
6.3 Conclusion 116

7 Conclusion **117**
7.1 Synthesis 117
7.2 Perspectives 119

Appendices **123**
A Algebraic definitions 123

Bibliography **127**

Acronyms

$\mathcal{P}_{fa}$	probability of False Alarm.
$\mathcal{P}_{md}$	probability of Miss Detection.
ADC	Analog Digital Converter.
ALU	Arithmetic Logic Unit.
ASIC	Application-Specific Integrated Circuit.
BPSK	Binary Phase Shift Keying.
BRAM	Block RAM.
CAWGN	Complex Additive White Gaussian Noise.
CCSK	Cyclic Code Shift Keying.
CORDIC	COordinate Rotation DIgital Computer.
CPU	Central Processing Unit.
CSS	Chirp Spread Spectrum.
CU	Correlation Unit.
DAC	Digital Analog Converter.
DCS	Digital Communication System.
DSP	Digital Signal Processor.
ECC	Error Correction Code.
FCU	FFT Correlation Unit.
FF	Flip-Flop.
FFT	Fast Fourier Transform.
FPGA	Field-Programmable Gate Array.
GF	Galois Field.

GNSS Global Navigation Satellite System.

GPGPU General Purpose Graphical Processing Unit.

HLS High-Level Synthesis.

IFFT Inverse Fast Fourier Transform.

IoT Internet of Things.

ISA Instruction Set Architecture.

LDPC Low Density Parity Check.

LLR Log-Likelihood Ratio.

LPWAN Low Power Wide Area Network.

LSB Least Significant Bit.

LUT LookUp Table.

MC Monte-Carlo.

MSB Most Significant Bit.

MT Multithreading.

NB-ECC Non Binary Error Correction Codes.

NB-LDPC Non-Binary Low Density Parity Check.

OM Overmodulation.

OSI Open System Interconnection.

PCM Parity Check Matrix.

PN Pseudo-Noise.

QCSP Quasi-Cyclic Short Packet.

QPSK Quadrature Phase Shift Keying.

RF Radio-Frequency.

RFNoC RF Network-on-Chip.

ROC Receiver Operating Characteristic.

SBC	Single-Board Computer.
SdR	Software-defined Radio.
SIMD	Single Instruction Multiple Data.
SNR	Signal-to-Noise Ratio.
SoC	System on Chip.
SPU	Score Processing Unit.
TCU	TS Correlation Unit.
TS	Time Sliding.
UHD	USRP Hardware Driver.
USRP	Universal Software-define Radio Peripheral.
WSN	Wireless Sensor Network.

Glossary

Binary Phase Shift Keying Modulation scheme, where *"1"*-bits are mapped to a high level (e.g. $+1$) and *"0"*-bits are mapped to a the opposite level (e.g. -1).

Chirp Spread Spectrum Modulation scheme, where symbols are mapped to a signal with a particular frequency fingerprint, called a chirp. The key feature of the chirp, is that its frequency fingerprint can be circularly shifted. Each possible shift correspond to a specific symbol.

codeword Output of the channel encoder, that contains the transmitted message and some redundancy. The original message can be retrieved by using the right channel decoder, which can also correct transmission errors in the process.

Global Navigation Satellite System Regroups all navigation satellite systems. For instance: GPS is the American system, Galileo is the Europen one, GLONASS is Russian and BeiDou is Chinese.

ISM radio band Portion of the radio spectrum reserved internationally for industrial, scientific and medical (ISM) purposes. [...] Despite the intent of the original allocations, in recent years the fastest-growing use of these bands has been for short-range, low power wireless communications systems [1].

NMEA 0183 NMEA 0183 is a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined and is controlled by the National Marine Electronics Association (NMEA). [2].

Overmodulation Added frame-level information, used in the Quasi-Cyclic Short Packet chain to improve synchronization.

Quadrature Phase Shift Keying Modulation scheme, where the number k from 0 to 3 (2 bits) is mapped to the complex number $e^{j\frac{(2k+1)\pi}{4}}$.

Quasi-Cyclic Short Packet The aim of the Quasi-Cyclic Short Packet (QCSP) project is to contribute to the evolution of IoT networks by defining, implementing and testing a new coded modulation scheme dedicated to IoT networks. The “big bet” of the project is to work on the emergence of non-binary codes combined with a Cyclic Code Shift Keying (CCSK) modulation [3].

List of Figures

1.1 Classical communication chain. 20

1.2 Physical layer asynchronous communication chain. 21

1.3 OSI Model, with the layer of interest framed in red. 21

2.1 Different technologies represented in their respective field, depending on the data rates and power consumption involved against the achievable ranges [26]. 27

2.2 Comparisons between a long packet and a short packet, highlighting the major concern the preamble is for short packets. 29

2.3 Generic organization of Software-defined Radio (SdR) platforms. 30

2.4 Parallelism paradigms graphically summarized. 33

3.1 Complete QCSP system model. 38

3.2 Circular correlation results for a given $\boldsymbol{P}_0$ of $q = 64$ chips with four different rotations of itself. 42

3.3 Legacy Elementary Score Processing Unit. 45

3.4 Example of score values S_n^ω as a function of the chip offset error Δ and of the residual frequency offset error $\theta - \omega$ at an SNR of -7 dB. 46

3.5 Rotation interval division for $p_\omega = 2$ and $p_\omega = 4$, ϑ represented as corresponding arcs. 47

3.6 Toy example of a search grid (in red) represented on the received score associated to one frame in a noiseless asynchronous channel. The bluer the color is (resp. the yellower), the lower is the corresponding score (resp. the higher). 48

3.7 Complete detection system. 49

3.8 Results of 10^7 MC simulations for $q = 64$, $N = 60$, an SNR of -10 dB, and over a synchronous channel (i.e. $\Delta = 0$ and $\theta = 0$). 51

3.9 Overview of the results presented in [17], showing the impact of q and N on the achievable SNR. 52

4.1 Score Processing Unit (SPU) robust to an input scaling factor. 59
4.2 Effect of the norms $\mathcal{L}_\infty$, $\mathcal{L}_1$, $\mathcal{L}_d$ and $\mathcal{L}_2$ on detection performances with $N = 60$, $q = 64$ and an SNR of -10 dB. 62
4.3 FFT based correlation possible architecture for a given frequency offset value ω. 64
4.4 Comparison between frequency based and time based computation of correlations for $q = 8$ and $p_\Delta = 2$. Red and orange points correspond respectively to the output of two distinct SPUs. 66
4.5 Time Sliding (TS) correlation possible architectures for a given frequency offset value ω. 69
4.6 Resulting ROC curves for frame of $N = 60$ symbols, symbols of $q = 64$ chips, at SNR of -10 dB for several setup of the couple of parameters $(p_\omega,\ p_\Delta)$. 70
4.7 Impact of the couple (p_ω, p_Δ) on $\mathcal{P}_{md}$ for a $\mathcal{P}_{fa} = 10^{-6}$, represented on a color map. The colder the color is, the higher $\mathcal{P}_{md}$ is (thus, the lower the performances are). 71

5.1 Transmitter side of the QCSP chain. 78
5.2 QCSP uncommon functions represented for $q = 4$. 80
5.3 Detection tasks data flow graph, with associated path stress. 86
5.4 The two different correlators represented with their associated Δ-parallelism and ω-parallelism. 88
5.5 FCUs shared memory total usage. 89
5.6 FCUs memory can be distributed or shared in an FCU correlator. 90
5.7 $\mathbf{\Gamma}^\omega = \{e^{j\frac{k\omega}{q}}\}_{k\in[\![0,2qp_\omega]\!]-1}$ for $q = 64$ and $p_\omega = 4$, quantized on $\eta_\omega = 4$ and 6 bits, represented on the north-east quadrant. 99
5.8 Fixed-point model of the QCSP detector, broken down in smaller processing blocks, depending on η, the input quantization, $p = \log_2(q)$ and N. . . 100
5.9 ROC curves for $q = 64$, for various value of p_ω and an SNR of -9 dB. . . . 102

6.1 Complete QCSP communication chain represented with the implementation used, as well as the communication actual settings. 107
6.2 Implemented transmitter component diagram. 109
6.3 GNSS module embed into the Raspberry Pi chassis. 109
6.4 Structure of the payload sent over the radio channel. 110
6.5 Implemented receiver component diagram. 111

6.6 Languages and platforms involved in the receiver. 111
6.7 Maximum ranges reached for reliable QCSP transmissions in the first experiment (in blue) and in the second experiment (in magenta), along the tracks followed in each experiment. 114
6.8 Frames successfully detected, synchronized and decoded for various bad channel conditions. Red marking highlights the position of the frame. . . . 115

7.1 Prospected implementation planning 120

List of Tables

2.1 Overview of the features offered by three LPWAN technologies, SigFox, LoRa and NB-IoT. 28

3.1 Different representations of GF(8) elements for $\mathbb{P}_3 = \alpha^3 + \alpha + 1$. 39
3.2 Possible mapping of GF(8) on a generic $\boldsymbol{P}_0$, using the natural representation of GF symbols defined in Table 3.1. 41

4.1 Computational complexity of each norm in terms of square-roots, multiplications, additions, and comparisons 60
4.2 Computational complexity comparison depending on p_Δ value ($p_\omega = 1$) for each correlation method. 72
4.3 Benchmarking platforms technical characteristics 73
4.4 Algorithm benchmarking results depending on the platform used, the values of (p_Δ, p_ω) and the correlation method implemented, for $N = 60$ and $q = 64$. 75

5.1 Performance of QCSP transmitter software implementation for a payload of 120 bits, $q = 64$, and $N = 60$. 81
5.2 Targeted FPGA characteristics and price tag 83
5.3 Performance of QCSP hardware stack on Xilinx FPGA clocked at 100 MHz. 84
5.4 Complexity comparison depending on p_Δ and for $p_\omega \in [\![1, 8]\!]$. 92
5.5 Software optimized implementation throughput depending on the platform used, the values of (p_Δ, p_ω) and the correlation method used, for $N = 60$ and $q = 64$. 95
5.6 Hardware performance of the detector on a Xilinx Kintex 7 XC7410T clocked at 100 MHz, using floating-point arithmetic if not specified, for $q = 64$, $N = 60$ and several values of p_Δ and p_ω. 97
5.7 Hardware performance estimation of the quantized correlators on a Xilinx Kintex 7 XC7410T (speed grade -1) and XC7K325T (speed grade -2), with a working frequency of 100 MHz, for $q = 64$, and several values of p_ω. . . . 103

7.1 Examples of primitive polynomials $\mathbb{P}_p$ for different values of p and the corresponding order $q = 2^p$ 125
7.2 Binary and polynomial representations of GF(8) elements for $\mathbb{P}_3 = \alpha^3+\alpha+1$. 126

Chapter 1

Introduction

Foreword

This Ph.D. thesis is a collaborative work between the Université de Bretagne Sud (UBS, France) and the Bordeaux INP institute (France). It has been supervised by Prof. Emmanuel Boutillon and Dr. Bertrand Le Gal. The research leading to these results received funding from the French National Research Agency ANR-19-CE25-0013-01 part of the project entitled Quasi Cyclic Short Packet (QCSP) (website: https://qcsp.univ-ubs.fr/).

1.1 Introduction

The demand for interconnected objects, from the simplest *intelligent* thermostat to the most sophisticated autonomous vehicle, has grown exponentially throughout the last decade. The development of this Internet of Things (IoT) has been a strong incentive of innovation, as it allows dreaming of large unsupervised wireless sensor networks. These dense networks are supposed to be composed of low-cost devices, able to operate off-grid, autonomously, and reliably for several years. Thanks to this technology, it may be possible to improve renewable energy production [4], or elaborate a completely autonomous public transportation system [5].

However, with the increasing device density, issues arise. More and more objects need to communicate on the same radio band. The capacity of the channel is limited, thus the question of communication efficiency become increasingly pressing. A naive solution would be to supervise networks, with a human operator or an omniscient scheduler. The problem here is that the first solution would be inefficient in a massively connected network, and that the second one is simply impossible. In addition, since sensor nodes are expected to be used in large number, they must be resource-efficient and energy-efficient, to be in-line with the recent shortages. To tackle this issue, various technologies have emerged, like

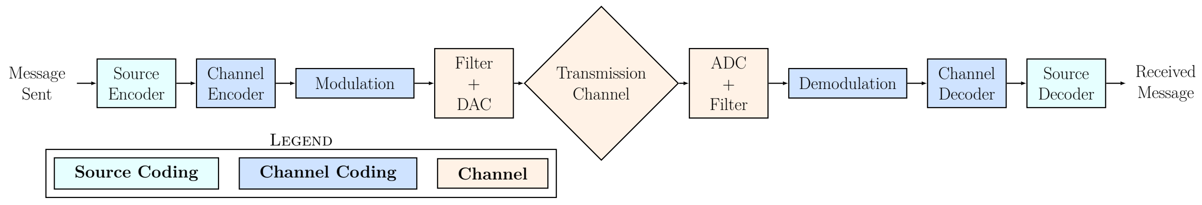


Figure 1.1 – Classical communication chain.

LoRa [6], or ZigBee [7]. However, some paradigms inherited from classical communication contexts and early technologies remains in use despite the drawbacks they come with.

A classical digital communication chain is represented in Fig. 1.1. A certain amount of data, called the message, is first source encoded. This process leverages the knowledge about the source data format to represent it efficiently. It is the case for audio data compression with Free Lossless Audio Compression (FLAC). Another example is the footprint lowering of often-used symbols, like it was done for the Morse code. In this code, the letter E, which was mostly used, was denoted by "." whereas the letter Q, rarely used, was denoted by "$- - .-$".

Then, this binary representation is channel encoded, which consists of adding a reliability to the transmission, often by adding redundancy. The most simple channel coding is the repetition code, where the binary input is just repeated. However, it is highly inefficient in most cases, since the coding rate (the ratio of clear data on the coded data) diminish way faster than the error correcting performance grows. Thus, other correcting codes have been invented, most notably the Low Density Parity Check (LDPC) codes [8], the turbo codes [9] or the polar codes [10].

The output of the channel encoder, called a codeword, is then modulated. This step serves to prepare the binary data to be transmitted through the channel, that is not in the digital world. For radio communications, it is the air. The modulation chosen also has an impact on the signal detection and synchronization capability, and on the sensibility of the signal to events that can appear in the channel, from the simplest additive noise, to more problematic erasures and multi-paths. The next step consists of filtering the digital signal, to prepare to the analog conversion. This filter, coupled with the corresponding optimum filter in reception, helps to maximize the Signal-to-Noise Ratio (SNR). However, a wrong filter choice can increase the sample interference, meaning that consecutive samples could cancel each others. That last step concludes the digital processing in transmission, the signal being then fed to a Digital Analog Converter (DAC). The resulting analog signal is transmitted through a physical channel, which for the IoT, is often the Radio-Frequency

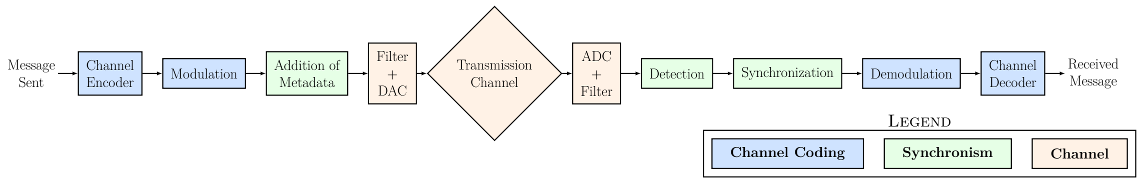


Figure 1.2 – Physical layer asynchronous communication chain.

(RF) channel. In reception, all those steps are done in reverse, using the corresponding filter, demodulator, channel decoder and source decoder.

However, there is an issue with the last presented chain. It assumes the synchronicity of the communication. In reality, the receiver does not know *when*, nor *in which condition* the signal will arrive. With the model depicted in Fig. 1.1, it cannot. This is why more accurate models exist, also taking two other steps into account. These steps are the detection of an incoming signal, and the synchronization of a detected signal. An updated model is depicted in Fig. 1.2.

These steps can be addressed in multiple ways, at different levels of the Open System Interconnection (OSI) model (see Fig. 1.3). The current thesis is part of the Quasi-Cyclic Short Packet (QCSP) project. In this project a new innovative waveform is introduced, and proposes a solution at the physical layer, the lowest level of the model.

Historically, the answer to the synchronization problem was to add a level of information shared between the transmitting and the receiving sides. Packets were preceded by a standardized "*preamble*", which eases the implementation of the detection and synchronization systems, like in 3GPP LTE [11] or Wi-Fi [12]. Indeed, knowing what should be looked up for simplifies the design of the receiving system. In classical contexts, the few

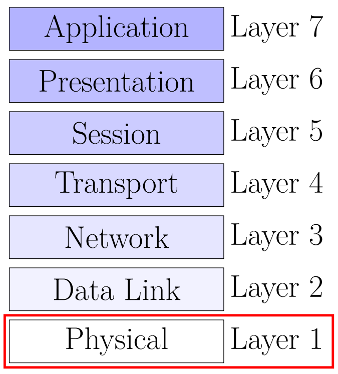


Figure 1.3 – OSI Model, with the layer of interest framed in red.

bytes of the preamble are negligible compared to the kilobytes or megabytes composing the payload. Unfortunately, in the IoT context, the packets sent are often small, from a few bytes to a few hundreds at most [13, 14]. The preamble footprint is not negligible anymore. Worse, Polyanskiy proved that the energy used during its transmission is simply lost from a communication point of view [15]. The bandwidth resource allocated to the preamble transmission is wasted for all other devices that use the same radio band.

The QCSP waveform has been designed specifically to address this issue. It allows detecting and synchronizing short packets at low SNR (lower than -20 dB), without using any preamble [16, 17]. This feature permits saving precious bandwidth resources. The waveform stems from the association of a Cyclic Code Shift Keying (CCSK) modulation and a Non Binary Error Correction Codes (NB-ECC), technologies already in use in the Chinese satellite system [18]. Theoretical results already demonstrated that a probability of Miss Detection ($\mathcal{P}_{md}$) inferior to 10^{-4} is achievable for a probability of False Alarm ($\mathcal{P}_{fa}$) below 10^{-6}. Incidentally, thanks to the lack of preamble, the complexity of the transmitter is reduced, which is logically accompanied by a drop in costs, and a higher energy efficiency. The drawback is a substantial increase in complexity at the receiving side. Although expected, it makes the implementation of a real-time receiver challenging.

The current work aims to study the feasibility of the real-time implementation of the QCSP communication chain. It thrives to propose evolution of the algorithm to improve the detection performances, while increasing the energy efficiency. Besides, several architectural optimizations beneficial to the algorithm-architecture adequacy are presented, for both software and hardware targets. The ultimate goal of the thesis is the implementation of an energy efficient cost constrained QCSP transmitter and of an associated real-time performant receiver. These devices will enable to conduct full-scale experiments that allows validating the waveform in IoT contexts, like those the Low Power Wide Area Networks (LPWANs).

It should be noted that this thesis has taken place in parallel of another, conducted by Dr. K. Saied [17]. His work was focused on the theoretical aspects of the QCSP waveform, yet we fruitfully collaborate with each other on many occasions. We built on each other strengths, the advances of one benefitting the other.

1.2 Manuscript outline

The current work is segmented in seven chapters, the first being this introduction.

The second chapter will precise the context of the thesis. It will consist in the definition of the main terms, and the detail of the main challenges. It will also propose a review of the existing optimization and implementation techniques, and how they may be relevant for the current work.

In the third chapter, the complete QCSP communication chain is detailed. The necessary concepts of CCSK modulation scheme and NB-ECC are explicated in the process, as well as their use in the overall algorithm. Both depend on the theory of Galois Fields (GFs), which is thus defined. An overview of the achievable detection performances of the waveform is also provided.

The fourth chapter is dedicated to the improvement made to the detection algorithm, for the purpose of efficiency. Indeed, the algorithm has been fortified, and is now immune to input scaling factors. This feature has been added after evaluating different normalization method from a complexity point of view, but also considering their impact on detection performances. Moreover, a new correlation method based on Time Sliding (TS) windows is presented. The method is demonstrated to be more efficient than the legacy method to compute the correlations crucial to the QCSP detection task. In addition, software simulations and benchmarks supporting this assumption are provided.

Throughout the fifth chapter, the QCSP transmission algorithm and the most critical task of the QCSP receiver, the detection, are analyzed under the prism of efficient implementation. In a first time, the processes involved in the transmission of a frame are detailed and context-aware optimizations are proposed for both CPU and FPGA circuits, enhancing throughput and energy efficiency. Benchmarks and achieved implementation results are presented, validating the relevance of the QCSP chain for use in LPWANs. In a second time, the most critical task of the receiver is defined, to better focus improvement efforts. After demonstrating the criticality of the correlation task inside the detector, the inherent parallelism levels of the algorithm are identified, and presented. As for the transmitting side, an updated complexity review of correlator possible implementations is provided. Indeed, we present how we took advantage of the properties of the QCSP algorithm and the features offered by manycore CPUs and FPGA circuits to reduce the complexity of the correlator. Besides, the superiority of the contributed TS approach is demonstrated, as well as the relevance of the QCSP waveform for IoT context. This is supported by the throughput achieved, and resource utilization of both the software and the hardware real-time implementations. In a third time, a quantized model of the receiver is introduced. It allows reducing yet again the resource footprint of the receiver, while doubling the throughput. The only downside is a small detection performance toll, that do not prevent the detector to meet the required $\mathcal{P}_{md} < 10^{-4}$ for a $\mathcal{P}_{fa} < 10^{-6}$.

The sixth and penultimate chapter presents the outcome of all the efforts related in the previous chapters: a complete real-time QCSP communication chain. The system is a complex heterogeneous system, yet flexible and modular. It has allowed conducting full scale experiments which permitted to put the waveform to test, and to measure its performances in real case scenarios. Besides, the results of two different experiments are provided.

Finally, the seventh and ultimate chapter concludes this thesis. It summarizes the contributions made throughout the current work, and offers perspectives for future research at short-term and long-term.

Chapter 2

State of the Art

Contents

2.1 Internet of Things and associated challenge 25
2.2 Software-defined Radio platforms 29
2.3 Implementation of digital communication systems 31
2.4 Conclusion 35

This chapter thrives to introduce the general context in which the current work takes place. To this end, the Internet of Things (IoT) context is presented, with its interests and the associated challenges. Notably, the relevance of short packets, as well as the non effectiveness of preambles for these packets is reviewed.

Then, the advances in Software-defined Radio (SdR) platforms are presented, as well as what they allow. First, we explain what an SdR platforms is, and give example of commonly used devices. Then, the advantages of these platforms are highlighted, and we explain how the current work can benefit from them.

Finally, the challenges related to the implementation of complex digital signal processing systems are highlighted.

2.1 Internet of Things and associated challenges

Wireless communication technically already existed in France in the 18th century, with technologies like the Chappe's telegraph [19]. Nowadays, the everlasting increase of popularity of the internet led to the appearance of several digital communication standards. The second generation mobile phone communications (2G) was initially released in the nineties, and nowadays, as of 2022, thirty years later, the fifth generation (5G) is currently in worldwide deployment [20]. The latter represents a considerable leap in complexity compared to the previous generation, the 4G, also called Long Term Evolution (LTE). In efficiency alone, 5G is expected to perform much better than 4G. The key difference between both generation, is the range of contexts embraced by the generation.

When 4G was mainly focused on human to machine or human to human communications, the 5G aims to enhance these applications, but also enables machine to machine communications. It has boosted the development of IoT applications, that began earlier thanks to technologies such as LoRa and SigFox. It has been a strong incentive for this research field lately [21]. As said in introduction, it is hoped that the development of IoT will bring solution to contemporary issues, like energy and resource shortages. It may help to slow the global warming, which is a threatening reality [22], especially after the COVID-19 outbreak, and the exceptional climatic events of 2022.

The networks involved in the IoT have considerably evolved throughout the years [23]. Nowadays, we can distinguish at least three main use cases.

Safety context — A set of systems that require safety and security cannot be unresponsive. The needs of such systems, like autonomous vehicles, or pilot-less drones, are supposed to be addressed by Ultra Reliable Low Latency Communications (URLLC) subset of the 5G standard [24].

Massively dense context — At the opposite side, massively dense networks of high activity sensor devices, that do not require enhanced reliability, are addressed by massive Machine Type Communications (mMTC) [25], another subset of the 5G standard.

Middle ground — In the between stands the older notion of LPWANs, networks aiming for long range communications, with strong constraints on energy efficiency. For those, latency and throughput are lesser issues, but remains likable features. This is for this context that standards like LoRa and SigFox were developed.

However, since cost and energy efficient implementations are required, the complexity of the algorithms involved in both transmitting and receiving sides can be a bottleneck. As represented in Fig. 2.1, LPWAN tries to achieve transmission ranges superior to cellular technologies, in conjunction with lower power consumption, thus logically for lower data rates.

Wireless Sensor Networks (WSNs) are typical examples of LPWANs. A WSN can represent a large distributed autonomous monitoring system, such as in work reported in [27] that presents ZigBee based network capabilities. The leafs of the network, that are low-end energy constrained sensor nodes, are numerous. They communicate information

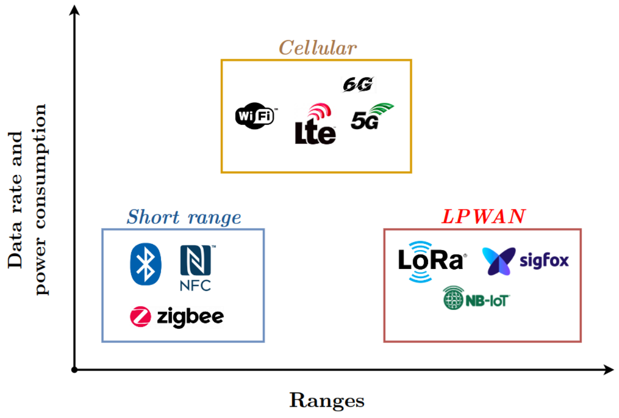


Figure 2.1 – Different technologies represented in their respective field, depending on the data rates and power consumption involved against the achievable ranges [26].

to intermediate relay nodes, that themselves transmit data of several sensor nodes to base stations that aggregate the data, to perform actions [28]. This results in a complex meshed network, with several massively interconnected devices. A network like this can be prohibitively expensive, in every sense. To allow such network topology in acceptable costs, the sensor nodes have to be low-cost, thus have to use communication protocols with reduced complexity. That is less true for the relays and the base stations. For them, the priority is real-time performances. However, lower costs and energy sobriety remain sought features. The LPWAN standard [29] define some requirements in terms of efficiency, range and throughput. The battery of LPWAN sensor nodes is assumed to last from a few days to a few years for one full charge. A transmission must be able to reach its recipient for ranges of 1 to 5 km in urban areas at data rates around 100 kb/s, and up to 10 km for lower throughputs. In rural area, this range requirement goes up to 40 km.

Throughout the years, several technologies have been proposed to offer low-complexity energy efficient solutions. For contextualization purpose, three of these technologies are summarized in Table 2.1. They offer different trade-offs, but all have in common a low transmitter complexity.

Another challenge of LPWANs is the coordination of the network [30, 31]. Indeed, there is a massive number of devices, each one may want to transmit data at any in time, and the amount of data cannot be known in advance. In this condition, using a supervised protocol is impossible. Only a random access protocol can sustain the inherent constraints associated to such non-slotted massive ALOHA system [32].

Methods exist on every layer of the OSI model [33], but the best way to reduce energy consumption is to address issues at the lowest level, the physical layer. At this level, the

Table 2.1 – Overview of the features offered by three LPWAN technologies, SigFox, LoRa and NB-IoT.

	SigFox	**LoRa**	**NB-IoT**
Modulation	BPSK	CSS	QPSK
Central frequency	868 MHz	868 MHz	LTE Bands
Bandwidth	0.1 kHz	125/250 kHz	200 kHz
Max data rate	0.1 kb/s	50 kb/s	200 kb/s
Max payload length	96 bits	1944 bits	12.8 kilobits
Urban range	10 km	5 km	1 km
Rural range	40 km	20 km	10 km
Interference resilience	very high	very high	low
Error correcting code	none	Hamming codes	Conventional codes

solution consists on developing methods to detect and synchronize packets reliably at low SNR. The low SNR is a consequence of the lesser quality of the equipments, the low transmission power required by the energy constraints, and of the noise produced by all the other devices trying to independently communicate. The classical way to allow detection and synchronization in an unsupervised network is, as mentioned in the introduction, the use of a pilot, also called a preamble. For instance, in 4G cellular networks, Zadoff-Chu sequences are used [34], due to their excellent correlation properties, and the particularity that two different sequences are near orthogonal. For short, it means that when looking for a specific sequence, one cannot erroneously detect another.

The issue is that in LPWANs, packets sent are, for the most part, short-sized [35], (a few hundreds to a thousand of bits typically). Thus, as represented in Fig. 2.2, while resources allocated to the preamble are negligible in conventional networks, it does not remain true for LPWANs. In current IoT systems, the preamble can represent up to 50% of the data sent [24], i.e. 50% of the bandwidth consumed. In a context where the amount of bandwidth resources is limited, since multiple devices try to access it, this is unacceptable. Moreover, Polyanskiy proved that the energy allocated to detection and synchronization are wasted from the point of view of the overall communication [15], and that preamble-

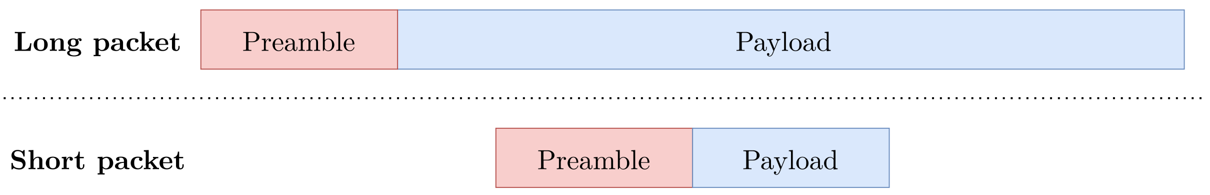


Figure 2.2 – Comparisons between a long packet and a short packet, highlighting the major concern the preamble is for short packets.

less approach are theoretically possible. Indeed, past the detection and synchronization task, the preamble does not convey any useful information.

Several methods to achieve preamble-less communications are described in the literature. All reported techniques use already present information, and try to use it for detection and synchronization purpose. A classical solution is the use of Error Correction Code (ECC) as a detection marker [36]. However, it has not achieved both low complexity and good detection performances at low SNR. Another common proposition is to superpose an additional frequency information over the payload [37, 38]. This is reminiscent of how “LoRa-like” physical layer works. Nevertheless, this kind of techniques is very sensible to frequency errors, that may arise due to local oscillator inaccuracies or Doppler effect. Some recent work introduced the use of complex modulation schemes [39, 40], but they have not been implemented yet.

A point all the method related in the literature have in common, is that they must have been prototyped, to validate their performances. A key enabler of the current advances in wireless technologies is the democratization of Software-defined Radio (SdR) platforms. These systems boost research on those topics, by greatly reducing development cycles.

2.2 Software-defined Radio platforms

The concept of Software-defined Radio (SdR) platforms has emerged in the late 20^{th} [41, 42]. A few years later, the apparition of various SdR devices and their associated platforms in the first decade of the 21^{st} century has been a great breakthrough. Indeed, devices like the high-end Universal Software-define Radio Peripheral (USRP) from Ettus Research [43], or low-cost HackRF [44], coupled with the GNU Radio software suite [45][46] have been key enablers for the digital communication progress. Importantly, they successfully combine high versatility and high performance, satisfying the need of an efficient prototyping and fast deployment platform [47–49].

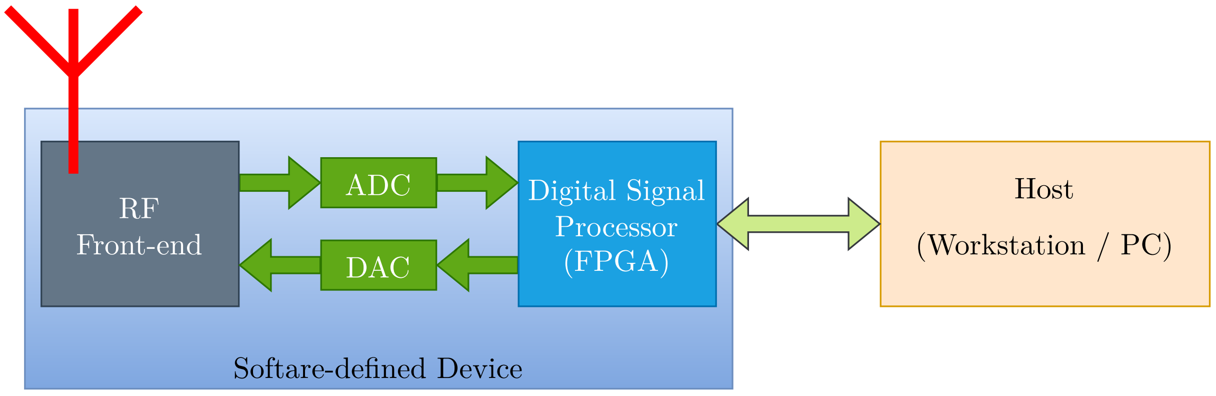


Figure 2.3 – Generic organization of Software-defined Radio (SdR) platforms.

An SdR system is often composed of an Radio-Frequency (RF) frontend associated to a reconfigurable digital processing system, as represented on Fig. 2.3. The reconfigurable system controls the RF frontend, and communicates with a "host" computer. The digital signal processing tasks can then be performed on the host, offering a very high flexibility. This is why SdR platforms are interesting for both research and industrial applications, as one unique physical device can be used to implement radically different protocols. For instance, it has enabled free and open source implementation of the LTE specification, through OpenLTE, and the very complete Open Air Interface (OAI) [50]. This framework allows simulating and even implementing state-of-the-art communication scheme on general purpose processors, sparing precious design time. In the recent years, even General Purpose Graphical Processing Unit (GPGPU) accelerated frameworks are created [51], enabling faster prototyping of complex communication schemes.

Speeding simulations and reducing prototyping times are not the only benefits of SdR advances. Recent platforms can be used to actually implement energy efficient solutions. Indeed, high-end FPGA circuits (like Xilinx Kintex and Zynq 7 in USRPs) can be intentionally oversized, to allow partially or even completely deferring digital processing to the device. Thanks to the seamless integration of GNU Radio with the hardware it targets, real-time SdR-powered LPWAN systems are born, like LoRa [52] and SigFox [53] transceivers.

However, while eased by SdR breakthroughs, the development and the efficient implementation of state-of-the-art digital communication systems remain challenging. That is especially true for LPWANs, since they have to achieve energy efficiency, reasonable data rates, and long range all together. Different ways exist to implement such communication systems, and they all require to work on the algorithm to take advantage of target-specific features.

2.3 Implementation of digital communication systems

While the implementation of digital communication systems has been fastened thanks to SdR platforms like USRPs used conjointly with GNU Radio, special care must be given to the development of the related algorithms. Indeed, real-time efficient radio systems can be implemented using SdR platforms [54]. However, the digital processing involved remains complex. Their implementation thus still requires intensive work in order to minimize algorithm complexity. Algorithms also have to be parallelized and pipelined. It enables to benefit from multicore CPU or FPGA circuit features.

In the late decades, the processing performance of multicore and manycore devices has constantly improved, thanks to clock frequency increases, and to the integration of more and more Instruction Set Architecture (ISA) extensions. Associated with easy-to-use programming models [55–57], it enabled the prototyping and implementation of real-time software communication systems. While an implementation on CPU may not reach the throughputs and energy efficiency achievable with ASICs or FPGA circuits, it provides undeniable advantages. The software implementation flexibility and scalability are much higher, and prototyping time is substantially lower than for its hardware counterparts. Nevertheless, achieving high performances is challenging and requires algorithm parallelization efforts.

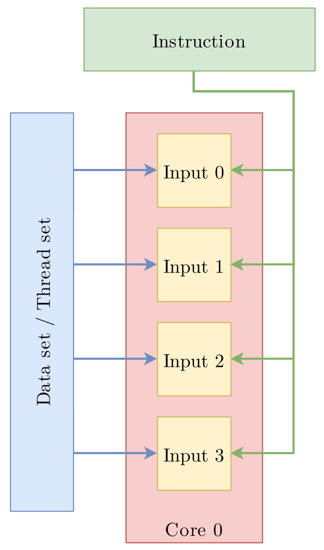
Nowadays, multicore and manycore systems can execute billions of instructions per second. They also offer parallelization opportunities that may help to achieve high throughputs and low latencies. Throughout the years, several programming languages and frameworks appears, to help in taking advantage of those opportunities. However, to actually benefit from them, it requires a substantial amount of developing time. Indeed, parallelization efforts are required to adapt the algorithm to the CPU or GPGPU architecture, and this is rarely addressed during the early stages of digital communication system design. Several types of parallelism exploitable in modern multicore and manycore architectures can help to increase data rates in baseband processing tasks. Based on Flynn's taxonomy [58, 59], we list four of them hereafter, the first three relevant for multicore CPUs and the fourth one more related to manycore GPGPUs. A graphical summary is depicted in Fig. 2.4 after the detailed explanations.

1. **Single Instruction Multiple Data (SIMD)**: The first type of parallelism is

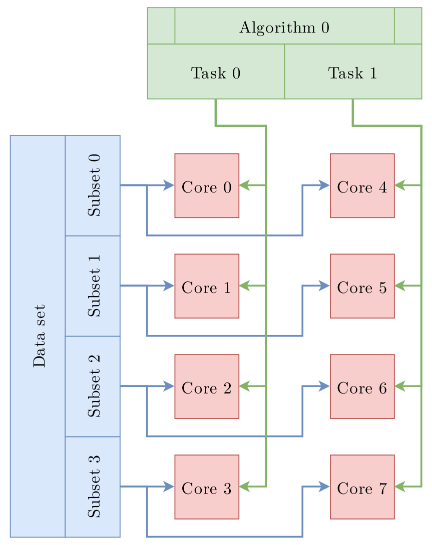
incidentally the one that is involved at the lowest level in processors. Current ARM and Intel architectures include vector instructions [60, 61], that are able to execute 4 to 16 floating-point operations in one clock cycle. It mechanically allows speedup by the same factors. The issue however, is that it requires the possibility for the processing to be described by unique operations on multiple inputs. Besides, to be effective, operands have to be aligned in memory, and the unique operation cannot be performed through independent tasks.

2. **Single Program Multiple Data (SPMD)**: The second approach takes advantage of the number of individual physical processor cores. Modern CPUs (Intel Core I7 or ARM Cortex A72) integrate several processor cores, typically 4 to 8 for ARM circuits, and from 10 to 40 for Intel ones. Computing speed can be increased by transforming the execution of an algorithm on a set of data to the execution of n sub-tasks on n subset of data. This can allow to substantially speed up the independent sub-tasks in the algorithm, but not by a factor n. Indeed, since neither the distribution of the task over several processor cores, nor the synchronization of all processor cores after the task completion are instantaneous. Thus, the performance is impacted in accordance to Amdahl's law [62]. To be beneficial, this parallelization paradigm requires startup and data gathering time to be negligible compared to the processing time.
3. **Multiple Programs Multiple Data (MPMD)**: Another way to leverage numerous processor cores is to segment the Digital Communication System (DCS), composed of several algorithms, into macro-tasks, as independent of each other as possible. Then, each one of them can be executed on its dedicated core. However, data still have to go through the complete processing pipeline. Consequently, the data rate is limited by the slowest task, incidentally called the bottleneck. A major advantage compared to the second approach though, is that tasks only need to be started and stopped once, minimizing startup and task synchronization times.
4. **Single Instruction Multiple Threads (SIMT)**: Massively parallel architectures like those involved in GPGPU excel in performing various calculations simultaneously on different data. The operations executed can slightly diverge, but it may hurt computation speed. Besides, they really shines for large data set, typically of around 64 kB [63], inducing extended latencies. As demonstrated in [64], while achieving honorable performances, they may not be well suited to implement a DCS.

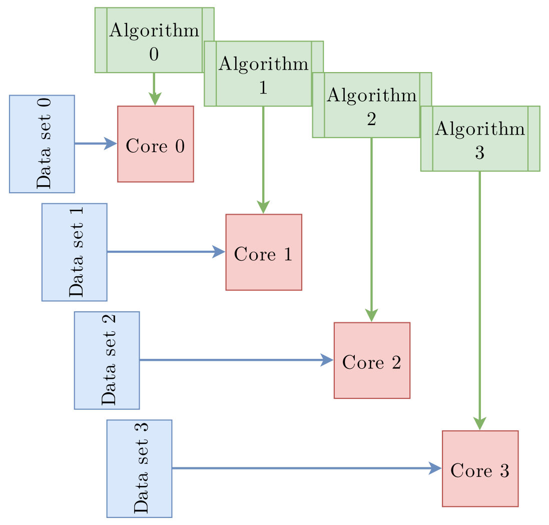
By making use of these parallelization techniques, it has been possible to implement



(a) Single Instruction Multiple Data (SIMD) or Single Instruction Multiple Threads (SIMT) (GPGPU only).



(b) Single Program Multiple Data (SPMD)



(c) Multiple Programs Multiple Data (MPMD)

Figure 2.4 – Parallelism paradigms graphically summarized.

performant 4G and 5G software chains [50]. Note that it requires the conjoint use of all the paradigms to ensure efficiency [65]. These implementations have even met industry-compliant constraints. Nevertheless, a glimpse to the efforts needed to develop efficient software implementations for various ECC decoders [66–68] allows to measure the non-negligible amount of engineering time and research work needed.

Despite the fact that such software platforms are already been developed and deployed, they require high-end devices to be up to the task [69]. The issue is that the relevant systems require CPUs like Intel Xeon processors, that can consume up to 500 Watts, and cost thousands of euros per unit. Intel themselves recommend the use of *(their own)* Stratix FPGA devices in order to improve digital processing capabilities of their platforms [70]. Indeed, the superiority of DCS hardware implementations over software implementations in terms of raw throughputs and latency but also of energy efficiency has been highlighted in work presented in [71]. The authors presented hardware ECC decoders achieving energy efficiency $\approx 30\times$ to $\approx 100\times$ superior to their software counterparts. Undoubtedly, the use of the FPGA circuits available in the SdR platforms boost DCS performances, as it is done in the computer vision domain [72]. However, the design of hardware architecture for reconfigurable targets is a long and difficult process. While implementations for numerous digital processing elements already exist in the literature [11, 73–75], the amount of work needed to actually use them can be substantial. Besides, they have often been optimized for specific use cases, which limits their adaptability. In the worst cases, the lack of flexibility can be an effective barrier to their deployment and to further evolutions.

To mitigate this issue, hardware design methodology has evolved. During the last two decades, High-Level Synthesis (HLS) industrial tools have matured, and are now widely adopted [76–79]. HLS have been developed for several decades now [80, 81], in order to allow the RTL design of digital processing systems from their behavioral descriptions [82]. The way HLS tools work is similar to how a compiler functions. The main difference is that the compiler has to translate the algorithmic description into machine understandable code, suited for a specific architecture. The HLS tool tries to design and generate the architecture that best suits the algorithmic description on the specified target. This allows achieving the high performances of RTL designs, without completely sacrificing flexibility and versatility. However, the procedure realized by the tool is extremely complex. For this procedure to be successful, the behavioral description have to be synthesizable[1], which requires that the designer has actual knowledge on digital circuit design. Thus, as for

1. i.e. understandable by the tool and implementable on the targeted circuits

RTL descriptions, algorithms have to be reformatted to suit hardware implementations. In addition, the description have to be tailored to the tool used, sometimes even to specific versions of a tool. For instance, in less than two years, several breaking changes have been introduced between the version 2019.1 [83] of the Xilinx HLS tool, and the version 2021.1 [84]. Consequently, the performances of the resulting implementation in throughput, latency and energy efficiency, depend on the algorithm itself, and the tool used, but also on the efforts put by the designer [85–89].

Nevertheless, HLS enables the development of hardware implementations in reduced design time, in comparison to RTL-based workflow. Even if achieved performances do not always match those of pure RTL descriptions, this time-saving feature, combined with higher flexibility, scalability and versatility, makes HLS a perfect tool for design exploration and other hardware-related research works. Especially since the democratization of Network-on-Chip systems [90] and their usage in SdR platforms, with frameworks like Ettus Research RF Network-on-Chip (RFNoC) [91].

These design procedures are complementary with multicore and manycore implementations. In an SdR platform, HLS coupled with development interface like RFNoC allow integrating all or part of the digital processing tasks as close to the RF frontend as possible [92].

2.4 Conclusion

As stressed out in the current chapter, the design and implementation of Digital Communication System for the IoT are anything but simple processes. The low complexity requirements, and the energy constraints, make achieving standard-compliant throughputs and latencies challenging. To make things worse, by the very nature of LPWANs, the available bandwidth is critically limited. It makes relying on preamble based approach unsustainable. Preamble-less strategies exist but can not trivially achieve acceptable detection and synchronization performances at the low SNR usual for the LPWAN context.

Nevertheless, the impressing development of SdR platforms is a game changer. Coupled with the increasing performances of multicore and manycore systems on one hand, and the maturity of HLS tools on another hand, it has enabled the real-time implementation of solutions to these problems. While design time required by algorithm parallelization and adaptation are still substantial, they are no longer prohibitive.

In the next chapter, the recently introduced Quasi-Cyclic Short Packet (QCSP) waveform [16, 17, 93] is presented. This new approach proposes a preamble-less detection and synchronization method, that has been demonstrated to work at low SNR. In the following chapter, the algorithm involved are analyzed and fortified. Moreover, the algorithm is optimized to increase its adequacy to intended targets. In a subsequent chapter, leveraging the parallelism types previously presented, the data rates are improved again. In addition, taking advantage of HLS, hardware specific optimizations and real-time implementations are also presented. A quantization work realized to maximize algorithm hardware adequacy is also related. Finally, a complete real-time QCSP communication system is disclosed, relying heavily on the previous efforts and on the features offered by SdR platforms.

Chapter 3

Quasi-Cyclic Short Packets Communication Chain

Contents

3.1 Non Binary Error Correction Codes 38

3.1.1 Galois Field of order q — GF(q) 39

3.1.2 Non-Binary Low Density Parity Check 40

3.2 Cyclic Code Shift Keying 41

3.3 QCSP System Model 43

3.3.1 Transmitter 43

3.3.2 Channel model 44

3.3.3 Detection 45

3.3.3.1 Score function 45

3.3.3.2 Search grid 47

3.3.3.3 Performances 49

3.3.4 Synchronization 53

3.4 Conclusion 54

This chapter aims to present the Quasi-Cyclic Short Packet (QCSP) communication chain in details, as depicted in Fig. 3.1. Indeed, understanding the algorithm and its whereabouts is needed in order to measure the requirements of its implementation. The core feature of the QCSP communication chain resides in the association of Non Binary Error Correction Codes (NB-ECC) and Cyclic Code Shift Keying (CCSK). Thus, this chapter begins by giving an overview of both concepts. For NB-ECC, it first summarizes the notion of Galois Fields (GFs), followed by Non-Binary Low Density Parity Check (NB-LDPC) codes, the family of NB-ECC used throughout the thesis. Then it introduces briefly how NB-LDPC codes works, followed by some available decoding methods. For CCSK, it explains how a Pseudo-Noise (PN) is generated and what is its properties, then summarizes the modulation and demodulation processes. It finishes by the presentation of the complete QCSP communication chain, from the transmitter side to the retrieve of the received message. Transmission and detection stages are detailed, and the synchronization and decoding stages are quickly summarized for completeness, since they have not been explored in this work.

3.1 Non Binary Error Correction Codes

On any medium, through any channel, exists the risk of an erroneous communication. Whether errors come from a faulty device or a hostile environment does not matter. They must be mitigated. That is the purpose of channel coding, which aims to add redundancy to a payload (i.e. the message to transmit) to make it resilient to a certain amount of errors.

There is a large variety of error correction codes. In this work, the focus has been given to NB-ECC, that exists over GF(q) for $q > 2$, such that NB-LDPC codes [94], non-binary turbo codes [95], or non-binary polar codes [96]. Non-binary codes have better

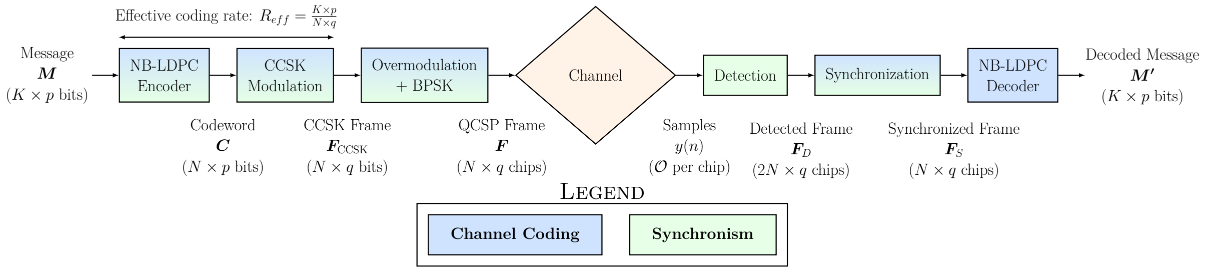

Effective coding rate: $R_{eff} = \frac{K \times p}{N \times q}$
Message $\boldsymbol{M}$ ($K \times p$ bits)
NB-LDPC Encoder
CCSK Modulation
Overmodulation + BPSK
Channel
Detection
Synchronization
NB-LDPC Decoder
Decoded Message $\boldsymbol{M'}$ ($K \times p$ bits)
Codeword $\boldsymbol{C}$ ($N \times p$ bits)
CCSK Frame $\boldsymbol{F}_{\text{CCSK}}$ ($N \times q$ bits)
QCSP Frame $\boldsymbol{F}$ ($N \times q$ chips)
Samples $y(n)$ ($\mathcal{O}$ per chip)
Detected Frame $\boldsymbol{F}_D$ ($2N \times q$ chips)
Synchronized Frame $\boldsymbol{F}_S$ ($N \times q$ chips)
Legend
Channel Coding
Synchronism

Figure 3.1 – Complete QCSP system model.

Table 3.1 – Different representations of GF(8) elements for $\mathbb{P}_3 = \alpha^3 + \alpha + 1$.

Power of α	Index	Polynomial	Binary	Natural
0	0	0	000	0
α^0	1	1	001	1
α^1	2	α	010	2
α^2	3	α^2	100	4
α^3	4	$\alpha + 1$	011	3
α^4	5	$\alpha^2 + \alpha$	110	6
α^5	6	$\alpha^2 + \alpha + 1$	111	7
α^6	7	$\alpha^2 + \quad 1$	101	5

error correction capabilities that their binary counterparts. These codes have been widely studied in the literature, due to their good performances for short payloads [97]. Moreover, they are highly compatible with high order modulations, since if the modulation order is equal to the one of the code, each symbol can be directly mapped to its modulated version, avoiding binary marginalization [98].

This section first introduces the notion of Galois field. It then briefly presents NB-LDPC codes with some decoding techniques.

3.1.1 Galois Field of order q — GF(q)

A Galois Field denoted GF, also called finite field, is a field which contains a finite number of elements. The number of element is called the order of the field. The Galois Field of order q is denoted GF(q).

Because it is a field, multiplication and addition are defined in GF(q) for any order $q = \kappa^p$, $\forall p \in \mathbb{N}$ and with κ a prime number. Both operations follow mainly the same rules they follow in traditional fields (like in $\mathbb{R}$, or in $\mathbb{C}$). Non-zero elements of GF(q) can be represented as power of a primitive element denoted α, such that the q elements GF(q) can be denoted $\{0, 1 = \alpha^0, \alpha^1, \ldots, \alpha^{q-2}\}$. In addition, when $\kappa = 2$, such that $q = 2^p$, elements of GF(q) have a binary representation. These results from the polynomial definitions of GF(q). An example of such representation is given in Table 3.1 for $q = 8$ and a primitive polynomial $\mathbb{P}_3 = \alpha^3 + \alpha + 1$. All necessary mathematic definitions and more in depth explanations can be found in appendix A.

3.1.2 Non-Binary Low Density Parity Check

NB-LDPC codes are the generalization of LDPC codes for high order GF. They have one interesting feature: for small codewords, they outperform traditional binary LDPC codes [99]. The gap in error-correction performances widens when the GF order grows. However, this is also accompanied by a substantial increase in complexity of the decoder. This has historically slowed down the adoption of non-binary codes. Nevertheless, increases in computing power in the recent years and research advances tend to reverse the trend [68]. For example, they are currently in use in the Chinese GPS system BeiDou [18], and in the CCSDS communication standard [100].

NB-LDPC are linear block codes [101], with the particularity that their Parity Check Matrix (PCM) is sparse, i.e. it contains more zeros than non-null coefficients. This matrix is used to produce a generator matrix $\mathbf{G}$ that allows to encode a message $\boldsymbol{M}$ of K symbols into a codeword $\boldsymbol{C}$ of N symbols, such that

$$\boldsymbol{C} = \boldsymbol{M}\mathbf{G} \tag{3.1}$$

PCM is often denoted $\mathbf{H}$, and its coefficients in GF(q). The number of rows M corresponds to the number of parity constraints in the code, while its number of columns N is the number of symbols of the code, defining also the codeword size. Indeed, a codeword is composed of N symbols, K of which are in fact the initial information symbols, the $M = N - K$ remaining being redundancy symbols. The parity constraints of $\mathbf{H}$ must be verified by $\boldsymbol{C}$ for it to be a codeword, i.e. $\boldsymbol{C}\mathbf{H}^T = \mathbf{0}_N$, where $\mathbf{H}^T$ is $\mathbf{H}$ transposed and $\mathbf{0}_N$ the null vector of size N. This also allows to define the syndrome $\boldsymbol{D}$ of a word,

$$\boldsymbol{D} = \boldsymbol{C}\mathbf{H}^T, \tag{3.2}$$

with $\boldsymbol{C}$ a potentially faulty codeword. When all elements of $\boldsymbol{D}$ are null, it means that $\boldsymbol{C}$ is a correct codeword. The number of non-bull elements in $\boldsymbol{D}$ can serve as an indication on how far from being corrected a codeword is.

A codeword having N symbols for a message of K symbols, it leads to the definition of a code rate $R_c = \frac{K}{N} = \frac{N-M}{N} \leq 1$.

To correct errors, the idea is to estimate the probability for the received word to be one codeword or another, thus calculating its likelihood ratio. This is done in the variable nodes (VNs). Then, VN transmit this information to check nodes (CNs). Those calculate

Table 3.2 – Possible mapping of GF(8) on a generic $\boldsymbol{P}_0$, using the natural representation of GF symbols defined in Table 3.1.

GF symbol		Shifted $\boldsymbol{P}_k$ sequence		
0	0	$\boldsymbol{P}_0$	—	$\{ p_0\ p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7 \}$
α^0	1	$\boldsymbol{P}_1$	—	$\{ p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7\ p_0 \}$
α^1	2	$\boldsymbol{P}_2$	—	$\{ p_2\ p_3\ p_4\ p_5\ p_6\ p_7\ p_0\ p_1 \}$
α^2	4	$\boldsymbol{P}_4$	—	$\{ p_4\ p_5\ p_6\ p_7\ p_0\ p_1\ p_2\ p_3 \}$
α^3	3	$\boldsymbol{P}_3$	—	$\{ p_3\ p_4\ p_5\ p_6\ p_7\ p_0\ p_1\ p_2 \}$
α^4	6	$\boldsymbol{P}_6$	—	$\{ p_6\ p_7\ p_0\ p_1\ p_2\ p_3\ p_4\ p_5 \}$
α^5	7	$\boldsymbol{P}_7$	—	$\{ p_7\ p_0\ p_1\ p_2\ p_3\ p_4\ p_5\ p_6 \}$
α^6	5	$\boldsymbol{P}_5$	—	$\{ p_5\ p_6\ p_7\ p_0\ p_1\ p_2\ p_3\ p_4 \}$

the parity check result probability, and propagate it to the VNs, for them to update their decision. Thus, after a sufficient number of iteration, every symbol should converge to their actual values. Multiple methods exist to implement this [68, 102], but they are not the topic of this thesis.

3.2 Cyclic Code Shift Keying

Modulating a signal allows to efficiently send it through the channel. It participates in the resilience of the communication to perturbations (noise, fading, and so on ...). This section aims to introduce to the Cyclic Code Shift Keying (CCSK) modulation technique, since it is a core concept of the overall QCSP system.

CCSK is a modulation scheme where bits are gathered in groups of p bits and mapped to unique circular rotations of a Pseudo-Noise (PN) sequence composed of $q = 2^p$ chips [103], denoted $\boldsymbol{P}_0 = \{p(k)\}_{k\in[\![0,q-1]\!]}$. A chip is the smallest element of $\boldsymbol{P}_0$. While a bit takes value in the binary set (0 or 1), a chip can be a bit, a binary real value (e.g. -1 and 1), or can even be a complex value, as it is the case for Zadoff-Chu sequences [34]. CCSK can thus be seen as a direct mapping from GF(q) (c.f. Table 3.1) to the set of all possible circular rotation of $\boldsymbol{P}_0$. The k^{th} circular rotation [1], denoted $\boldsymbol{P}_k$, is defined as

$$\forall k \in [\![0, q-1]\!], \quad \boldsymbol{P}_k = \{p(k), p(k+1), \ldots, p(q-1), p(0), p(1), \ldots, p(k-1)\}. \tag{3.3}$$

An example of mapping using natural representation of GF symbols is given in Table 3.2

1. Left-shift is assumed in this thesis. Right-shift can be used as well, without any impact on performances.

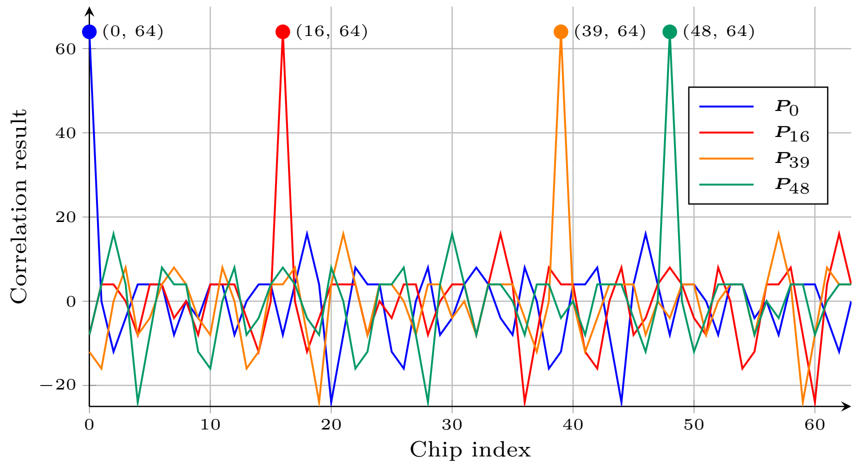


Figure 3.2 – Circular correlation results for a given $\boldsymbol{P}_0$ of $q = 64$ chips with four different rotations of itself.

for $q = 8$. The index representation could be used as well, but the natural representation is needed for the sake of synchronization performances (c.f. section 3.3.4). The modulation process is thus very intuitive, particularly in conjunction of a NB-ECC, since each symbol c of the codeword $\boldsymbol{C}$ can be directly mapped to the corresponding CCSK symbol $\boldsymbol{P}_c$.

The demodulation process is less straightforward. It consists on doing the circular correlation of the CCSK symbol with $\boldsymbol{P}_0$. The index of maximum of correlation is then the value of the initial symbol. The q-element circular correlation $\boldsymbol{L} = \{L(k)\}_{k \in [\![0,q-1]\!]}$ of $\boldsymbol{P}_0$ with a vector $\boldsymbol{y} = \{y(k)\}_{k \in [\![0,q-1]\!]}$ is obtained using

$$\forall k \in [\![0, q-1]\!], \quad L(k) = \sum_{i=0}^{q-1} y(i) \times p(k+i \mod q). \tag{3.4}$$

A representation of this relation is given in Fig. 3.2. Despite the PN sequence used, which has been generated only for the figure and not optimized at all, the maximum of correlation is clearly visible for each represented shift, at the index of the corresponding symbol.

Finally, the key feature of CCSK is also related to the demodulation process. In Fig. 3.2, the SNR is infinite, i.e. there is no noise at all. In a noisy environment, some CCSK symbols may be damaged, degrading the correlations presented in Fig. 3.2, which can lead to erroneous demodulation. However, the circular correlation $\boldsymbol{L}$ can be easily converted to a vector of Log-Likelihood Ratios (LLRs), which can be directly provided to an NB-LDPC decoder [99]. The latter then handle the transformation into a codeword, then into the decoded message.

3.3 QCSP System Model

The Quasi-Cyclic Short Packet (QCSP) project aims, as stated on the dedicated web page [3], *to contribute to the evolution of IoT networks by defining, implementing and testing a new coded modulation scheme dedicated to IoT networks. The "big bet" of the project is to work on the emergence of NB codes combined with a Cyclic Code Shift Keying (CCSK) modulation.* Thus, unsurprisingly, the whole communication system model, therefore referred to as the QCSP chain, heavily relies on CCSK and NB-LDPC.

This section has for objective to present the QCSP chain, from the transmitter to the decoded message. In particular, the channel model used is depicted, as well as how detection works. The steps involved in the synchronization algorithm are also detailed. It should be noted that this section mostly summarizes the extensive work done in [17] by K. Saied, a former Ph.D. student also involved in the QCSP project, but on theoretical aspects only. During the first two years of my Ph.D., K. Saied and I had a lot of fruitful exchanges.

3.3.1 Transmitter

The message $\boldsymbol{M}$ to be transmitted is composed of K_b information bits, with $K_b = K \times p$ and $(K, p) \in \mathbb{N}^2$. It is thus composed of K symbols in GF(q), with $q = 2^p$. This message is encoded using an NB-LDPC code of rate $R_c = \frac{K}{N}$, thus resulting in a codeword of N symbols in GF(q), or $N \times p$ information and redundancy bits. Each symbol of the codeword is CCSK modulated as described in Table 3.2 into the corresponding shift of $\boldsymbol{P}_0$, thus resulting in a CCSK frame $\boldsymbol{F}_{CCSK}$ of N CCSK symbols, or $N \times q$ chips.

This modulation has a rate $R_m = \frac{p}{q}$. The combination of the two rates R_m and R_c results in the QCSP effective rate $R_{eff} = \frac{K \times p}{N \times q}$.

Then, Overmodulation (OM) is added, using a binary sequence $\boldsymbol{B} = \{B(i)\}_{i \in [\![0,N-1]\!]}$ of size N. It consists on adding a phase of 0 or π to each symbol depending on the value $B(i)$, $i \in [\![0, N-1]\!]$. Since the result would be modulated using Binary Phase Shift Keying (BPSK), which is a direct mapping $\{0, 1\} \Rightarrow \{-1, 1\}$, by representing chips and elements of $\boldsymbol{B}$ as 1 or -1, the QCSP frame $\boldsymbol{F}$ can be directly obtained using

$$\forall k \in [\![0, q-1]\!],\ \forall i \in [\![0, N-1]\!],\ \ F(i \times q + k) = F_{CCSK}(i \times q + k) \times B(i). \qquad (3.5)$$

Finally, the QCSP frame is shaped by a root-raised cosine filter, decreasing inter-symbol interferences. The root-raised cosine filter has a roll-off factor β such that $0 < \beta < 1$. In this study, β has been set to 0.35. The filtered frame is passed to a dedicated RF device which handle the analog tasks and the transmission to the channel.

3.3.2 Channel model

The targeted context of LPWAN imposes to consider low-end devices which try to transmit messages of few hundreds of bits sporadically and asynchronously. It means that the potential receiver cannot have any prior knowledge about time of arrival, carrier frequency offset, transmission phase nor any other possible channel parameters. The only parameters accessible are those specific to a QCSP communication (e.g. the PN and OM sequences, as well as the NB-ECC). Let T_c and $T = q \times T_c$ (in seconds) be the duration of a chip and a CCSK symbol respectively. The optimal half raised cosine filter is applied in reception. The frequency offset is assumed low enough to neglect chip interferences. Besides, the signal is over-sampled such that one chip corresponds to $\mathcal{O}$ samples, defining an oversampling factor $\mathcal{O}$ which is typically between 4 and 8. It results in a sampling frequency $F_s = \mathcal{O}/T_c$, used as the driving frequency of the Analog Digital Converter (ADC). The time of arrival t_a can thus be seen as a real value $x_a = t_a/T_c$, which can then be expressed as

$$x_a = n_a + \frac{r_a}{\mathcal{O}} + \epsilon_a, \tag{3.6}$$

where $n_a = \lfloor x_a \rfloor$, the integer part of x_a being the time in number of chips, r_a the closest clock cycle index within a chip ($r_a \in [\![0, \mathcal{O}-1]\!]$), and ϵ_a the residual error ($\epsilon_a \in [\![\frac{-1}{2\mathcal{O}}, \frac{1}{2\mathcal{O}}]\!]$). However, the latter is neglected in the sequel, $\mathcal{O}$ being considered high enough. It is worth noting that by testing in parallel all the values r_a can take, it is possible to find the one maximizing the reception performances, allowing to set r_a to 0. However, this results in an increased computational complexity. The earlier the decimation is done the more this burden is reduced, since fewer processes must be parallelized. In the current implementation, decimation is done after the detection stage.

Local oscillator mismatch and/or Doppler effects are also considered, leading to a random frequency offset F_0 added to the received frame. In T_c seconds, it results in a rotation $\frac{T_c F_0}{2\pi}$ radians between two consecutive chips, leading to the definition of the normalized frequency offset $f_0 = T_c F_0$. It generates a rotation $\theta = 2\pi f_0 q$ radians between two chips separated by a symbol duration. Finally, the phase offset ϕ is unknown too

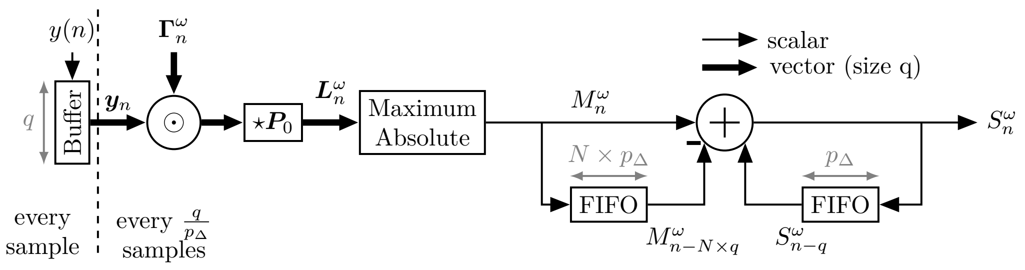


Figure 3.3 – Legacy Elementary Score Processing Unit.

($\phi \in [0, 2\pi]$). In summary, for a single frame received at index n_a with $r_a = 0$,

$$\forall n \in \mathbb{N},\ y(n) = \begin{cases} z(n) & \text{if } n \notin [\![n_a, n_a + Nq - 1]\!], \\ F(n - n_a)e^{j(n\theta/q+\phi)} + z(n) & \text{otherwise,} \end{cases} \tag{3.7}$$

with $z(n)$ being realizations of a random variable following $\mathcal{N}(0, \sigma)$, defining a Complex Additive White Gaussian Noise (CAWGN) with zero mean and standard deviation $\sigma = \sqrt{10^{\frac{-\text{SNR}}{10}}}$. Considering the absence of any prior knowledge, θ and ϕ are supposed to be uniformly distributed in their respective domain.

3.3.3 Detection

This section sums up the detection method presented in [17]. It first describes the score function used to probe the channel, followed by the presentation of the concept of time-frequency search grid. Then, the decision-making process is explained. Finally, it presents briefly the impact of q and N on detection performances.

In this section, the sampling frequency f_c is equal to the inverse of the chip period T_c, i.e. the optimal sampling time is assumed known. This is achieved in practice by testing in parallel all phase hypotheses resulting from oversampling and by keeping the one associated with the highest score (see section below).

3.3.3.1 Score function

CCSK based detection consists mainly in comparing the output of a score function to a threshold value U_0. The score is computed using the last $N \times q$ received chips (impersonating a potential received frame) at time n divided in N sub-vectors $\{y(n + (\aleph - N) \times q + 1 + i)\}_{i \in [\![0, q-1]\!]}$ of q chips (length of a CCSK symbol), for $\aleph$ in

$[\![0, N-1]\!]$. Thus, let us define $\boldsymbol{y}_n$ such that

$$\forall n \in \mathbb{N},\ \forall i \in [\![0, q-1]\!],\quad y_n(i) = y(n-q+i+1). \tag{3.8}$$

In other words, $\boldsymbol{y}_n$ represents the q chips of the $(N-1)^{th}$ symbol of a frame that begins at time $n - N \times q + 1$ and is completely arrived at time n. Consequently, the score is calculated from the N vectors $\boldsymbol{y}_{n-(N-\aleph)*q+1}$ for $\aleph$ in $[\![0, N-1]\!]$. Its value exceeding the threshold U_0 assesses the arrival of a new frame.

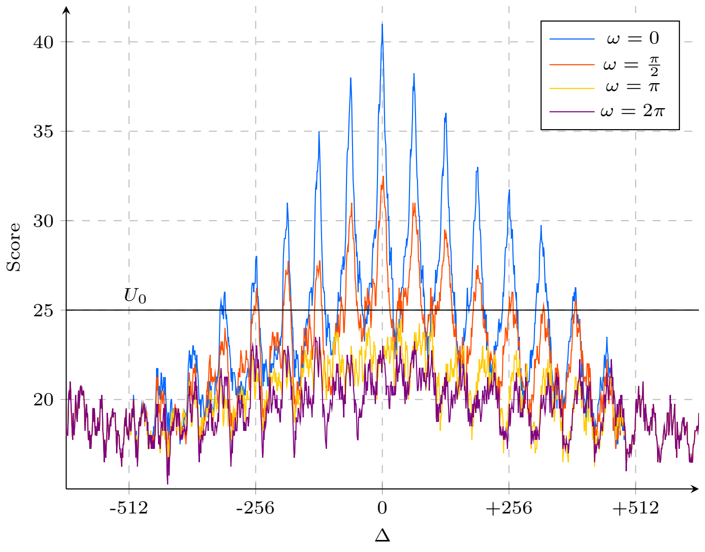


Figure 3.4 – Example of score values S_n^ω as a function of the chip offset error Δ and of the residual frequency offset error $\theta - \omega$ at an SNR of -7 dB.

The score function S_n^ω corresponds to a filter output that is maximized for a frame arrived at time n with a frequency offset $f_\omega = \frac{\omega}{2\pi q}$. The filter is locally coherent at the symbol level and non-coherent at the frame level due to potential residual frequency offset. At the symbol level, the first step is to mitigate the frequency offset by multiplying term by term (operator $\odot$) $\boldsymbol{y}_n$ with the vector $\boldsymbol{\Gamma}_n^\omega$, which is a pure complex sinusoidal of frequency $-f_\omega = \frac{-\omega}{2\pi}$, to obtain $\boldsymbol{y}_n^\omega$ i.e.

$$\boldsymbol{\Gamma}_n^\omega = \{e^{-j\frac{k\omega}{q}}\}_{k\in[\![n-q+1,\,n]\!]}, \tag{3.9}$$

$$\boldsymbol{y}_n^\omega = \boldsymbol{y}_n \odot \boldsymbol{\Gamma}_n^\omega \tag{3.10}$$

The residual frequency offset of $\boldsymbol{y}_n^\omega$ is supposed low enough to allow a coherent demodulation of the symbol, thus, denoting $\star$ the operation done in (3.4), the correlation vector

$\boldsymbol{L}_n^\omega$ is computed as

$$\boldsymbol{L}_n^\omega = (\boldsymbol{y}_n \odot \boldsymbol{\Gamma}_n^\omega) \star \boldsymbol{P}_0. \tag{3.11}$$

The infinite norm $\mathcal{L}_\infty$ of $\boldsymbol{L}_n^\omega$ is taken in order to perform the non-coherent integration over the whole frame. By denoting $M_n^\omega = \mathcal{L}(\boldsymbol{L}_n^\omega)_\infty = \max\{|L_n^\omega(i)|, i \in [\![0, q-1]\!]\}$ this norm, it gives

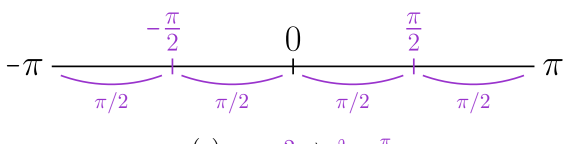
$$S_n^\omega = \sum_{i=0}^{N-1} M_{n-iq}^\omega. \tag{3.12}$$

The overall architecture that allows to compute simply the score S_n^ω every q chips is given in Fig. 3.3, and is called a Score Processing Unit (SPU). The value p_Δ should be considered set to 1 for now. This notation is explained in the next section.

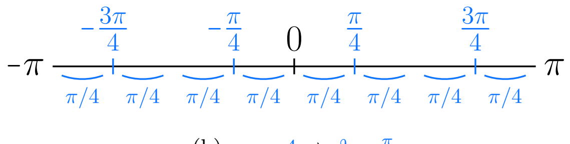
The resulting score for a frame of length $N = 10$, with a CCSK modulation of length $q = 64$, over a CAWGN channel with an SNR of -7 dB is shown in Fig. 3.4. Let's define Δ as the shift in chips between the exact chip of arrival and the score time of calculation. The value of Δ is equal to 0 at the exact time of arrival of the frame. In this figure, several frequency error situations are depicted, with $\theta - \omega$ taking the values $0, \frac{\pi}{2}, \pi$, and 2π (a full rotation of each CCSK symbol). In this example, the threshold value is set to $U_0 = 25$. One can note that when the residual frequency offset is high (greater than a rotation of π for each CCSK symbol, typically), the score magnitude is significantly reduced. Likewise, when the time offset error verifies $\Delta \mod q = \frac{q}{2}$, the detection of the frame is harder. The solution is to explore in parallel several hypotheses of time delay and frequency offset.

3.3.3.2 Search grid

Let us assume, without loss of generality, that the rotation θ (resulting from the frequency offset) is in the interval $[-\pi, \pi]$, i.e, giving at maximum a half clockwise or half counterclockwise rotation per CCSK symbol. Then it is possible to make p_ω hypotheses of rotation, effectively dividing the previous interval into p_ω sub-intervals. Each interval



(a) $p_\omega = 2 \Rightarrow \vartheta = \frac{\pi}{2}$



(b) $p_\omega = 4 \Rightarrow \vartheta = \frac{\pi}{4}$

Figure 3.5 – Rotation interval division for $p_\omega = 2$ and $p_\omega = 4$, ϑ represented as corresponding arcs.

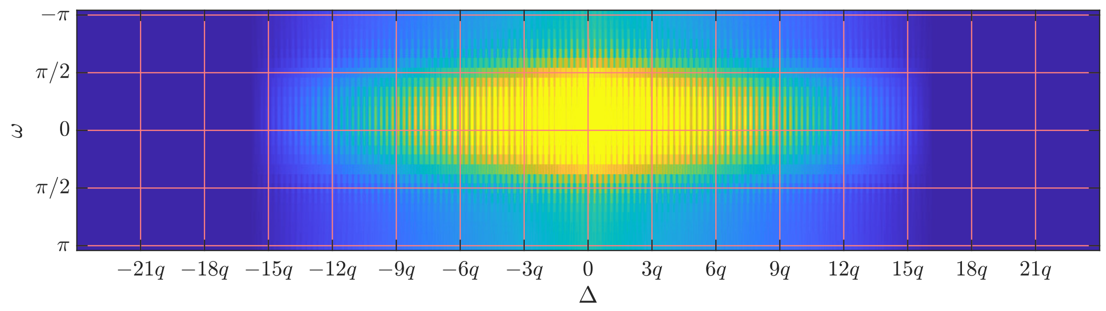


Figure 3.6 – Toy example of a search grid (in red) represented on the received score associated to one frame in a noiseless asynchronous channel. The bluer the color is (resp. the yellower), the lower is the corresponding score (resp. the higher).

is associated to a score filter $S_n^{\omega(r)}$ with

$$\forall p_\omega \in \mathbb{N}^*, \forall r \in [\![0, p_\omega - 1]\!],\ \ \omega(r) = \pi(-1 + \frac{2r+1}{p_\omega}). \tag{3.13}$$

So, the maximum distance between θ and the closest $\omega(r)$ value is bounded by $\vartheta = \frac{\pi}{p_\omega}$. For example, when $p_\omega = 4$, $\vartheta = \frac{\pi}{4}$, which corresponds to $1/8^{th}$ of residual rotation per CCSK symbol. A graphical representation is given in Fig. 3.5.

The score S_n^ω must be computed for each new received symbol (or every q samples). This ensures that $|\Delta|$ is bounded to $[\![0, \frac{q}{2}]\!]$. Computing the score less often results in skipping some chips or worse, some symbols. That would lead to the disappearance of some maxima in Fig 3.4, resulting in loss in performances that are unaffordable at lower SNR. Moreover, if the goal is to reduce complexity, it is better to reduce N or q directly [17]. In contrast, computing the score more often — up to q times more, i.e. for each new sample — increases detection performances. It leads to the introduction of the parameter p_Δ, power of two [2] indicating the number of score values computed every q samples for one rotation ω (i.e. $p_\Delta \in \{1,\ 2,\ 4,\ \dots\ ,\ 2^p = q\}$). This may also have an impact on the amount of memory required, since only $N \times p_\Delta$ values of M_n^ω would be needed. This results in a grid of scores, that divides the time-frequency space in bins. A toy example of such grid is given in Fig. 3.6, for bins of size $\frac{\pi}{2}$ along the frequency direction, and $3q$ (unusable in real case scenario) along the time direction.

The previously defined grid produces $p_\Delta \times p_\omega$ score values every q chips. Depending on the SNR, scores can exceed the threshold U_0 multiple times during frame arrival, for different frequencies. Thus, a potential new maximum is searched for the duration of a

2. A generalization to use any value is possible, but would complicate implementations. It has thus, without loss of generality, been left aside.

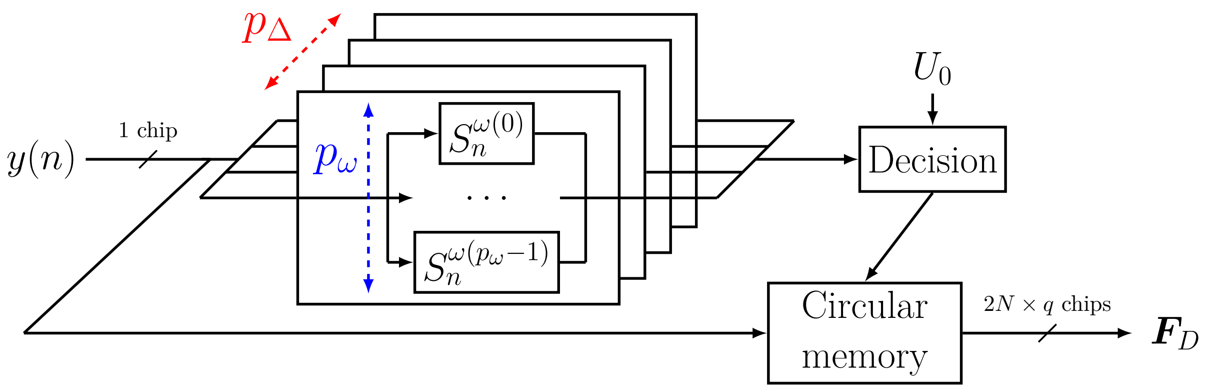


Figure 3.7 – Complete detection system.

frame. The index of this maximum of all maxima is then considered as the end of the frame, and used to produce a buffer $\boldsymbol{F}_D$ of $2N \times q$ chips. This buffer corresponds to the chips associated to the potential frame plus half of a frame before and after, to allow the synchronization stage to mitigate the inherent time inaccuracy of the detection stage. The theoretical system is depicted in Fig. 3.7. It represents p_Δ time hypothesis (i.e. the score is computed every $\frac{q}{p_\Delta}$ chips) where p_ω frequency hypothesis are computed in parallel, $\omega(r)$ defined as in (3.13). The $p_\Delta \times p_\omega$ score values are fed to a decision block, which, based on U_0 and its own internal state, control a circular memory. The decision block provides a read signal and (if needed) and address to the memory, in order to produce $\boldsymbol{F}_D$. It would require $p_\Delta \times p_\omega$ SPU, and a memory of at least $3N \times q$ chips (size of $\boldsymbol{F}_D$ plus one frame, in case the detection occurred for the first received chip).

3.3.3.3 Performances

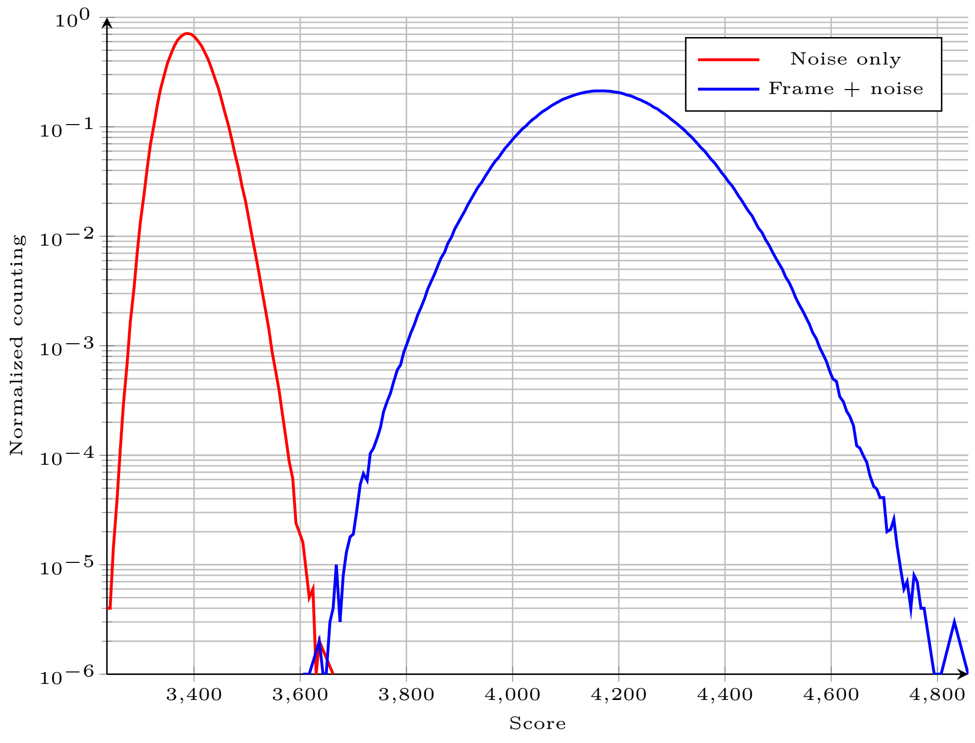
In order to measure detection performances two notions must be defined:

— The probability to consider a frame detected when there is none, called the probability of False Alarm ($\mathcal{P}_{fa}$),

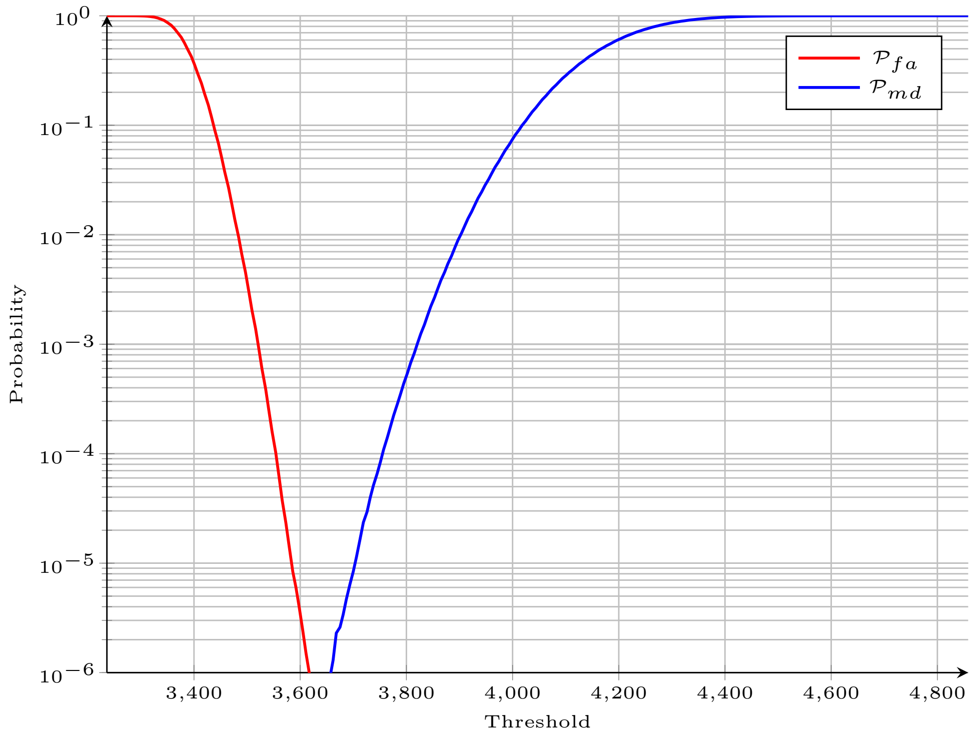
— the probability to miss an actual frame, called the probability of Miss Detection ($\mathcal{P}_{md}$).

A method to estimate these probabilities is to do Monte-Carlo (MC) simulations, by probing the score output of a detector in the absence of signal or in presence of a frame to estimate respectively $\mathcal{P}_{fa}$ and $\mathcal{P}_{md}$. An example of result for $q = 64$, $N = 60$, and for an SNR of -10 dB over a synchronous channel is depicted in Fig. 3.8a for the raw normalized counting and in Fig. 3.8b for the calculated probabilities. It is worth noting that a theoretical mathematical model has been verified in [93], and can be used to bypass MC simulation and obtain accurate results faster.

Unsurprisingly, increasing q and N led to an increase in detection performances. It would result in a greater gap between $\mathcal{P}_{fa}$ and $\mathcal{P}_{md}$ on Fig. 3.8b (due to the constant SNR of -10 dB). Most importantly, it translates to a lower achievable SNR when the probabilities are constrained. Arbitrary targets of $\mathcal{P}_{fa} < 10^{-6}$ and $\mathcal{P}_{md} < 10^{-4}$ have been decided for the QCSP project. An overview of the results has been extracted from [17] and is given in Fig. 3.9. They clearly prove the last assumption (increasing $q = 2^p$ improves detection performances). Moreover, as seen in Fig. 3.9a, the SNR that allow to reach the arbitrary target linearly decrease with the linear increase of p. Besides, they also demonstrate that the effect on performances of N, frame size in symbols, have the same behavior for all values of p. All curves have the same shape on Fig. 3.9b, but are shifted by 2.3 to 3 dB each. Thus, it is possible to study detection performances for low value of p, q, and N for an SNR, and then to estimate the results for higher values of p, q, and N for lower SNRs, simplifying the exploration.

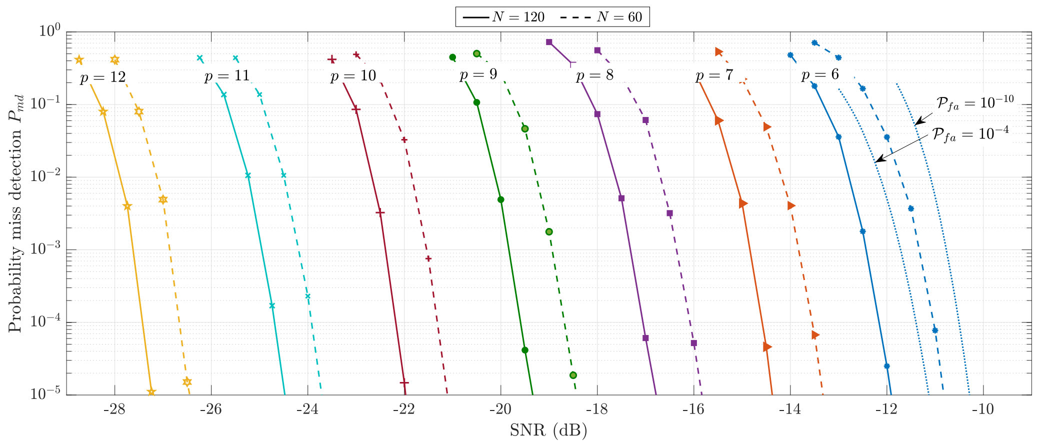


(a) Histograms (Normalized by 10^7)

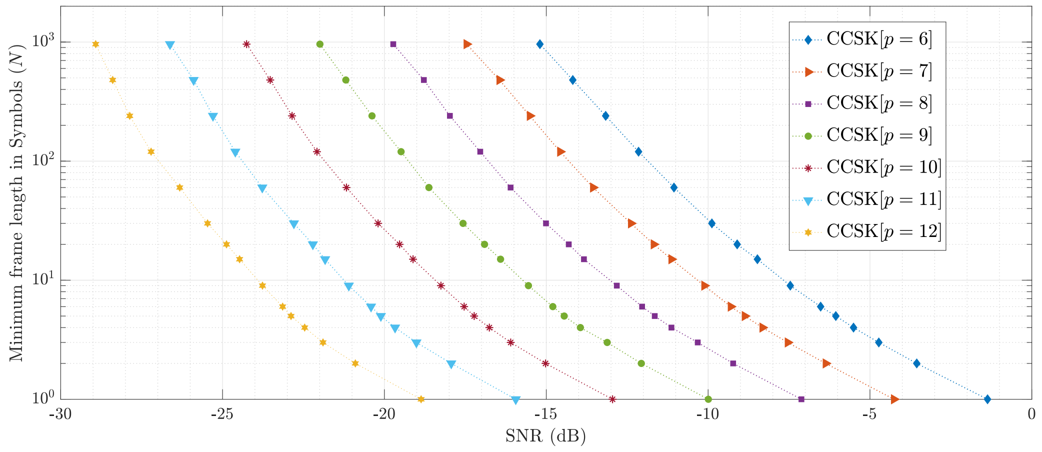


(b) $\mathcal{P}_{fa}$ and $\mathcal{P}_{md}$ functions of the threshold value.

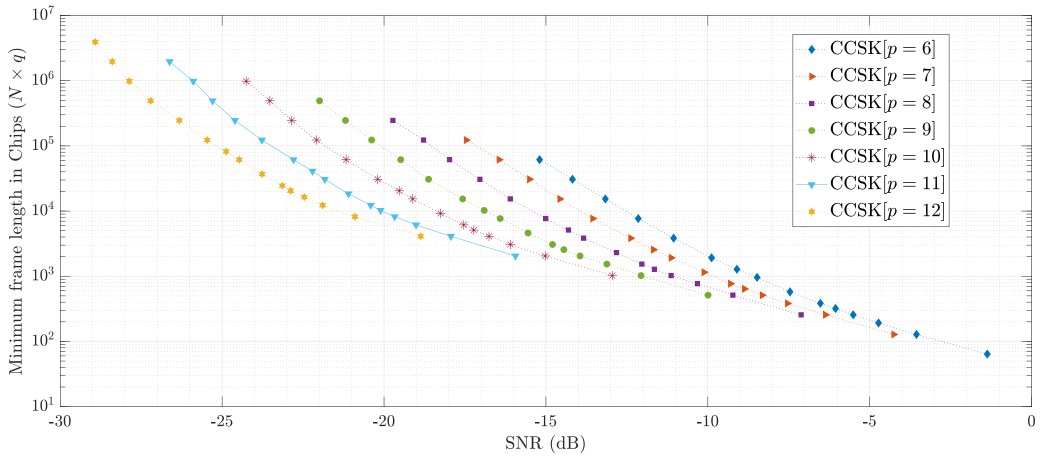
Figure 3.8 – Results of 10^7 MC simulations for $q = 64$, $N = 60$, an SNR of -10 dB, and over a synchronous channel (i.e. $\Delta = 0$ and $\theta = 0$).



(a) $\mathcal{P}_{md}$ functions of the SNR for p from 6 to 12, $\mathcal{P}_{fa} = 10^{-6}$ and $N = 60$ or 120.



(b) Minimum frame symbol length N required to ensure $\mathcal{P}_{fa} \leq 10^{-6}$ functions of the SNR, for p from 6 to 12.



(c) Minimum frame chip length $N \times q$ required to ensure $\mathcal{P}_{fa} \leq 10^{-6}$ functions of the SNR, for p from 6 to 12.

Figure 3.9 – Overview of the results presented in [17], showing the impact of q and N on the achievable SNR.

3.3.4 Synchronization

The detected frame buffer $\boldsymbol{F}_D$ produced by the detection stage cannot be used as-is, since an inaccuracy in the time-frequency-phase estimation drastically reduces CCSK demodulation and NB-LDPC decoding performances [93]. The objectives of the synchronization process are

1. to find the exact chip where the actual frame begin in $\boldsymbol{F}_D$,
2. to mitigate any remaining frequency offset,
3. to estimate as accurately as possible its initial phase ϕ.

The current section aims to present the synchronization process and to enumerate its steps. It does not dive into the details as it has not been subject to advanced research, partially due to the lack of time, but mainly because the constraint on synchronization is softer than the one on detection. Indeed, detection must be performed continuously. The synchronization process is invoked only when a frame is detected. Thus, in a context where latency is not a crucial point (like LPWANs [29]), it can be deferred to dedicated subtasks, with lower priority, using other computational resources (e.g. on another chip, or on dedicated low-priority process when working with a scheduler).

The first two steps are the same as running the detection process for $p_\Delta = q$ and $p_\omega \geq 16$. They allow to reduce the time inaccuracy to a handful of chips (± 8 at most) modulo the size of a symbol, and to limit the normalized frequency offset to $\pm 10^{-3}$ (less than an eighth of rotation per symbol). These conditions are required for the other steps to succeed.

The third step uses the Overmodulation (OM) introduced by (3.5) to find the symbol-level beginning of the frame. The N maximums of correlation with $\boldsymbol{P}_0$ for each possible symbol beginning are multiplied term-to-term with the OM sequence $\boldsymbol{B}_0$, and the maximum of the FFT of the product is computed. The highest value corresponds to the symbol-level beginning of the frame. Thanks to the previous steps, the true beginning of the frame is only a few chips away, at most [93].

In fourth, the FFT associated to the highest value from the last step is used to further correct a potential frequency offset. Indeed, since it estimates a frame-level rotation, the approximation is way more accurate than in previous steps, which was limited to the symbol level.

The two last steps are more complex. The second last leverage features of NB-LDPC. In a first version, it uses the syndrome as defined in (3.2), inspired by [36, 104]. The symbol hypothesis which results in the syndrome with the highest number of null-elements is assumed to be perfect. This has been further improved by using a *soft syndrome*, i.e. by considering the updated information in variable nodes instead of the hard decision check nodes (c.f. section 3.1). This step is the one that requires the use of natural CCSK mapping (c.f. 3.2). Indeed, the use of the index mapping would prevent an efficient use of such techniques, since a slightly shifted index mapped CCSK symbol is still mainly a codeword.

The last step but not the least aims to correct the phase offset. Two methods are in use. The first one simply consists in calculating the argument of the sum of the complex maximum of the time-frequency synchronized frame and to apply its opposite. It has thus been called the direct method (DM). While working, it is degraded at low SNRs and may not correctly estimate the phase in some cases. In opposite, the parametric method (PM) nearly reaches the best estimation possible, but at high cost. It first computes the full CCSK correlation of all symbols. Then, for each symbol, the first three maxima are weighted according to the number of parity check they allow fulfilling, the most highly weighted being kept. This results in N values. Then, an affine function with ϕ as origin and the residual frequency offset as steering factor is estimated. This function should maximize the probability of the observation. The values of ϕ and the residual frequency offset are then extracted and corrected.

The choice between both methods is another way to achieve different trade-offs.

In any case, after all six steps, a synchronized frame $\boldsymbol{F_S}$ of $N \times q$ chips is produce, and its frequency and phase offsets are assumed null.

The ultimate steps of the QCSP chain have been explained previously. The CCSK demodulation process (c.f. section 3.2) is applied to the synchronized frame $\boldsymbol{F_S}$, and the resulting LLR vector is fed to the NB-LDPC decoder (c.f. section 3.1).

3.4 Conclusion

The QCSP communication chain succeed to transmit, detect, synchronize, and retrieve short packets at low SNRs, approaching the Polyanskiy bound [15] (the Shannon limit for short packet) as close as 1.2 dB of distance. According to [93], it also saves up to 23% of bandwidth resources compared to classical preamble based approaches.

However, if it has been successfully tested on real data and in real life scenarios, the implementation is not efficient. It relied heavily on high level interpreted languages, and the legacy architecture is not adapted to real time processing of data. It cannot achieve the throughput required by standards and compares poorly to other solution on this point.

In the next chapters, several techniques are used, from algorithmic reorganizations, to target and language specific optimizations, in order to implement an efficient real-time capable QCSP system.

Chapter 4

Detection Algorithmic Optimizations

Contents

4.1 Mitigating possible gain instability 58
4.1.1 Normalization methods 59
4.1.2 Complexity estimations 60
4.1.3 Impact on detection performances 61
4.2 Time sliding windows 64
4.2.1 Principle 65
Iterative factor precision issue 67
Architectural variations 68
4.2.2 Detection performance and complexity comparisons 70
4.2.2.1 Impact of (p_ω, p_Δ) on detection performances 70
4.2.2.2 Computational complexity comparisons 71
4.2.3 Software implementation, benchmark, and analysis 73
4.3 Conclusion 74

The chapter is dedicated to the enhancement made on the detection algorithm. Indeed, while the detection method presented in chapter 3 has been verified on real data, they were not processed in real time. They were first gathered using RF devices over the radio channel, then fed to the chain offline.

In order to implement a real time receiver, the detection stage, called *the detector*, must solve two main issues:

1. the possible variations of the receiver gain,
2. the inherent complexity of the correlation process.

The current chapter is dedicated to the contributions made to address these points. It is worth noting these contributions have also been the object of a dedicated paper [16]. The first section introduces the normalization process, its impact on the detector's performances, and how it has been parallelized and then implemented. The second section deals with an innovative method to implement the correlations. The new method consistently reaches the best detection performances theoretically achievable for each value of q, with a lower complexity than the legacy method. However, the new method requires the process of every new chip (equivalent to $p_\Delta = q$ in Fig. 3.7, see 3.3.3.2), while the legacy method allowed a wider range of trade-offs.

4.1 Mitigating possible gain instability

In the context of LPWAN, a receiver may very well run continuously for several days. If variations of the RF receiver internal gain are not unexpected, they must not disrupt the detector.

First, it should be noted that (3.12), which is used to calculate S_n^ω, is equivalent to an averaging moving filter of length N. Thus, (3.12) can be simplified as

$$S_n^\omega = S_{n-p_\Delta}^\omega + M_n^\omega - M_{n-N\times p_\Delta}^\omega, \tag{4.1}$$

with M_n^ω still being the maximum of correlation of the last q received chips at time n. As defined in section 3.3.3.2, p_Δ represents the number of score values computed every q chips.

Let's denote $A \in \mathbb{R}^+$ the receiver constant gain effect on the maximum M_n^ω. Thus, (4.1) becomes

$$\begin{aligned} \mathfrak{S}_n^\omega &= \mathfrak{S}_{n-p_\Delta}^\omega + AM_n^\omega - AM_{n-N\times p_\Delta}^\omega, \\ &= \mathcal{A}S_n^\omega, \end{aligned} \tag{4.2}$$

with $\mathcal{A}$ the gain resulting from an unknown function of A and the maximums. Since the threshold U_0 was estimated against plain S_n^ω, detection now depends more on $\mathcal{A}$, that cannot be exactly known nor predicted, than on the actual score value. Worse, $\mathcal{A}$ can vary over time, thus being a function of n, making the comparison to a threshold all the more difficult.

A solution was needed to have a reliable threshold without requiring an over-expensive gain control and stabilization system.

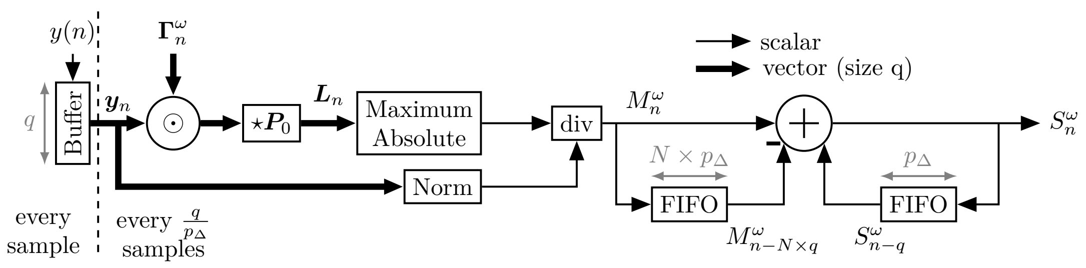


Figure 4.1 – Score Processing Unit (SPU) robust to an input scaling factor.

4.1.1 Normalization methods

Normalizing the input with a factor proportional to A suppresses the impact of any scaling factor in input. The normalization can be done directly per chip, per symbol or on any other scale. However, normalizing each chip independently is a nonsense, since it would literally flatten the input. On the opposite, the larger the scale is, the more sensitive to channel variations the system is, especially at low SNR. A large normalization window typically increases the impact of sporadic interferences. Since the CCSK correlation involved in the calculus of M_n^ω takes a symbol in input, it has been intuited that a norm extracted from the last q (symbol size) received chips would achieve correct results. This assumption has not been thoroughly tested in the current work, but some experiments conducted by L. Enrique Camacho Flores and L. Montaya Obesso confirms it. Thus, the mitigation consists in dividing M_n^ω by the norm of $\boldsymbol{y}_n$, the vector of the last q received samples at time n. This results in the SPU depicted in Fig. 4.1, that is nearly the same the prior version presented in Fig 3.3, but with a divider and a norm block added.

However, several norms exist, and one may be better than the other. Three classical norms have been investigated: the infinite norm $\mathcal{L}_\infty$, the 1-norm $\mathcal{L}_1$ and the 2-norm $\mathcal{L}_2$. Another norm, derived from $\mathcal{L}_1$ and denoted $\mathcal{L}_d$ is also investigated. They are defined as

$$\mathcal{L}_\infty : \quad \boldsymbol{y}_n \rightarrowtail \max(|y_n(i)|, \forall i \in [\![0, q-1]\!]), \tag{4.3}$$

$$\mathcal{L}_1 : \quad \boldsymbol{y}_n \rightarrowtail \sum_{i=0}^{q-1} |y_n(i)|, \tag{4.4}$$

$$\mathcal{L}_d : \quad \boldsymbol{y}_n \rightarrowtail \sum_{i=0}^{q-1} |\mathcal{R}(y_n(i))| + |\mathcal{I}(y_n(i))|, \tag{4.5}$$

$$\mathcal{L}_2 : \quad \boldsymbol{y}_n \rightarrowtail \sqrt{\sum_{i=0}^{q-1} |y_n(i)|^2}, \tag{4.6}$$

Table 4.1 – Computational complexity of each norm in terms of square-roots, multiplications, additions, and comparisons

(a) Complexity of a complete computation (consuming q chips)

	$\sqrt{}$	$\times$	$+$	$>$
$\mathcal{L}_\infty$	q	$2q$	q	$q(q-1)$
$(\mathcal{L}_\infty)^2$	0	$2q$	q	$q(q-1)$
$\mathcal{L}_1$	q	$2q$	$2q$	0
$\mathcal{L}_d$	0	0	$2q$	0
$\mathcal{L}_2$	1	$2q$	$2q$	0
$(\mathcal{L}_2)^2$	0	$2q$	$2q$	0

(b) Complexity of an iterative computation (consuming 1 chip).

	$\sqrt{}$	$\times$	$+$	$>$
$\mathcal{L}_\infty$	1	2	1	$q(q-1)$
$(\mathcal{L}_\infty)^2$	0	2	1	$q(q-1)$
$\mathcal{L}_1$	1	2	3	0
$\mathcal{L}_d$	0	0	4	0
$\mathcal{L}_2$	1	2	3	0
$(\mathcal{L}_2)^2$	0	2	3	0

with $\mathcal{R}$ (resp. $\mathcal{I}$) denoting the real part (resp. the imaginary part).

Applying the normalization to (4.1) gives

$$\bar{S}^{\omega}_{n,\gamma} = \bar{S}^{\omega}_{n-q,\gamma} + \bar{M}^{\omega}_{n,\gamma} - \bar{M}^{\omega}_{n-N\times q,\gamma}, \tag{4.7}$$

$$\bar{M}^{\omega}_{n,\gamma} = \frac{M^{\omega}_{n}}{\mathcal{L}_\gamma(\boldsymbol{y}_n)}, \quad \gamma \in \{\infty, 1, d, 2\}, \tag{4.8}$$

with $\bar{S}^{\omega}_{n}$ the normalized score and $\bar{M}^{\omega}_{n,\gamma}$ the normalized max.

4.1.2 Complexity estimations

First, let's remind that $\boldsymbol{y}_n$ is a complex vector, thus $|y_n(i)|$ is not an absolute value but a modulus, and

$$|y_n(i)| = \sqrt{\mathcal{R}(y_n(i))^2 + \mathcal{I}(y_n(i))^2}, \tag{4.9}$$

with $\mathcal{R}$ (resp. $\mathcal{I}$) denoting the real part (resp. the imaginary part). Recomputing the norm completely for each new score is thus not a trivial operation. The computational complexity of each norm in terms of square-roots, multiplications, additions, and comparisons (from the more complex to the most trivial operation) is reported in Table 4.1a. This table can be used to estimate the less complex norm, and it is, without a doubt, $\mathcal{L}_d$, followed by $\mathcal{L}_2$, since they require less square-roots.

Interestingly, all norms can be computed iteratively, by reusing a part of the last result and values associated to the last q inputs. The corresponding complexities are reported in Table 4.1b. Since $\mathcal{L}_1$ and $\mathcal{L}_d$ are moving filters, the transformation is trivial. For $\mathcal{L}_2$, the trick is to iterate on (and thus to save) the magnitude $|y(n)|^2$ instead of the modulus. Then the transformation is as trivial as for $\mathcal{L}_1$. Then, (4.4), (4.5), and (4.6) become

$$\mathcal{L}_1(\boldsymbol{y}_n) = \mathcal{L}_1(\boldsymbol{y}_{n-1}) + |y(n)| - |y(n-q)|, \tag{4.10}$$

$$\mathcal{L}_d(\boldsymbol{y}_n) = \mathcal{L}_d(\boldsymbol{y}_{n-1}) + |\mathcal{R}(y(n))| + |\mathcal{I}(y(n))| - |\mathcal{R}(y(n-q))| - |\mathcal{I}(y(n-q))|, \tag{4.11}$$

$$\mathcal{L}_2(\boldsymbol{y}_n) = \sqrt{\mathcal{L}_2(\boldsymbol{y}_{n-1})^2 + |y(n)|^2 - |y(n-q)|^2}. \tag{4.12}$$

For $\mathcal{L}_\infty$, there is a complication. If $\mathcal{L}_\infty(\boldsymbol{y}_{n-1})$ is $|y(n-q)|$, then a complete reprocessing of the norm is required. By the structure of the moving average filter, it happens at least every q chips, thus optimizing would not bring enough benefit. A simple optimization consists in saving the $|y(n)|$ values, reducing the need for square-roots, but to recompute all comparisons each time. Thus, (4.3) stay the same, but the complexity decreases.

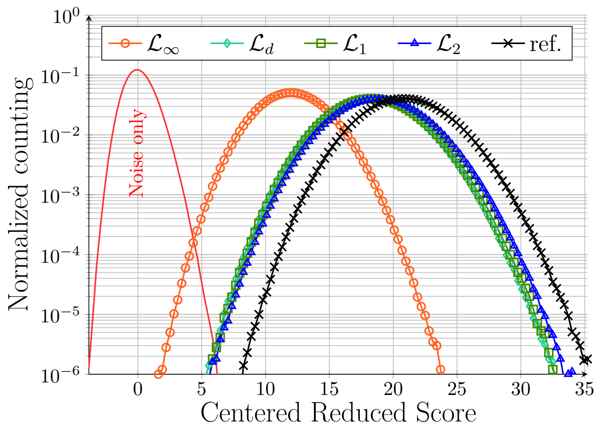
Finally, it is worth noting that the square value of $\mathcal{L}_2$ and $\mathcal{L}_\infty$ can be used as-is, by using $\frac{M_n^{\omega 2}}{\mathcal{L}_\gamma(\boldsymbol{y}_n)^2}$ in (4.7), with $\gamma \in \{\infty, 2\}$. This allows to spare two square-roots per score calculated.

At this point, the best candidate norm is $\mathcal{L}_d$, which has the lowest complexity and the simplest implementation, followed by $\mathcal{L}_2$ and $\mathcal{L}_\infty$. A study of the impact on detection performances has been carried out to settle on a decision.

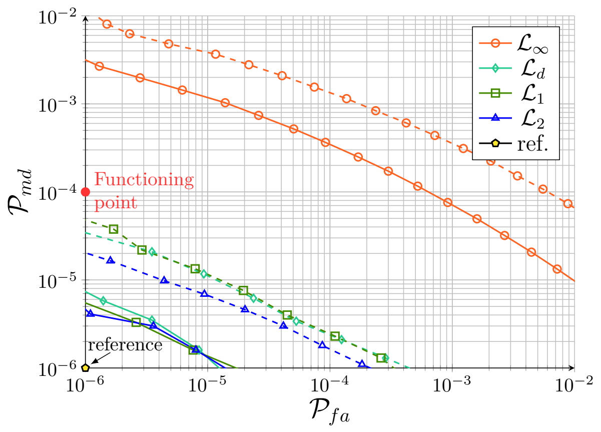
4.1.3 Impact on detection performances

Normalization is expected to reduce detection performances, since it is mostly influenced by the noise power at low SNR, and thus, reduce the signal dynamic (the difference between noise power and signal power). Besides, each normalization results in a different final score. Comparisons cannot be done directly.

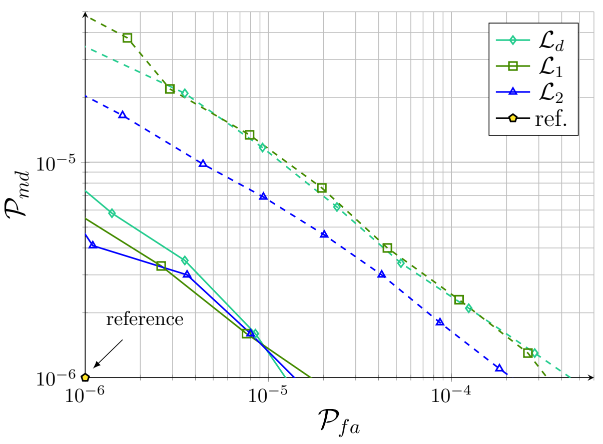
The solution resides in the properties of the Gaussian noise. Considering (3.7) and the fact that the noise $z(n)$ follow $\mathcal{N}(0, \sigma)$, but also according to [17], the score in absence of signal (affecting $\mathcal{P}_{fa}$) also follows a normal law. Let's denote $\mathcal{N}(\eta_0, \sigma_0)$ the law followed by the score without normalization. Since the normalization can be reduced to a division by a positive value, it is a continuous linear application, and so, does not modify the previously mentioned property. Consequently, let's denote $\mathcal{N}(\eta_\infty, \sigma_\infty)$ (resp. $\mathcal{N}(\eta_1, \sigma_1)$, $\mathcal{N}(\eta_d, \sigma_d)$ and $\mathcal{N}(\eta_2, \sigma_2)$) the law followed by the score resulting of the normalization by the norm $\mathcal{L}_\infty$ (resp. $\mathcal{L}_1$, $\mathcal{L}_d$ and $\mathcal{L}_2$). It is thus possible to center and reduce the scores by subtracting the corresponding mean η_γ and dividing by the standard deviation σ_γ, for $\gamma \in \{0, 1, d, 2, \infty\}$. The score in presence of signal is also centered by η_γ and reduced by



(a) Histograms of the centered reduced score $\hat{S}^{\omega}_{n,\gamma}$.



(b) Receiver Operating Characteristic (ROC) curves for frequency-synchronized channel (solid curves) and with a rotation error inferior to $\frac{\pi}{4}$ (dashed curves).



(c) Receiver Operating Characteristic (ROC) curves for frequency-synchronized channel (solid curves) and with a rotation error inferior to $\frac{\pi}{4}$ (dashed curves), zoomed.

Figure 4.2 – Effect of the norms $\mathcal{L}_\infty$, $\mathcal{L}_1$, $\mathcal{L}_d$ and $\mathcal{L}_2$ on detection performances with $N = 60$, $q = 64$ and an SNR of -10 dB.

σ_γ, giving a centered reduced score $\hat{S}^{\omega}_{n,\gamma}$

$$\hat{S}^{\omega}_{n,\gamma} = \frac{\bar{S}^{\omega}_{n,\gamma} - \eta_\gamma}{\sigma_\gamma}. \tag{4.13}$$

This method has been applied on scores gathered after 10^7 MC simulations, for each norm (namely $\mathcal{L}_2$, $\mathcal{L}_1$, $\mathcal{L}_d$, $\mathcal{L}_\infty$, and ref., which is without normalization), for $N = 60$, $q = 64$, at a perceived SNR in reception of -10 dB. Resulting scores $\hat{S}^{\omega}_{n,\gamma}$ are presented in Fig. 4.2a for each norm. Only one curve (the red unmarked one) is drawn to represent the result without a frame to lighten the figure, since it is nearly independent of the normalization method. The black crossed curve represents the reference score (without normalization). Being at -10 dB ensures that $\mathcal{P}_{fa} \ll 10^{-6}$, thus explaining why the curve doesn't even touch the red one. In contrast, the orange curve, which corresponds to $\mathcal{L}_\infty$, cross the noise curve embarrassingly early. The norms $\mathcal{L}_1$, $\mathcal{L}_d$ and $\mathcal{L}_2$ (represented resp. in green, in indigo and in blue) are closer to the reference, but also close to each other. The blue one seems a bit closer to the reference though.

In order to better analyze the results, they have been visualized using well-known Receiver Operating Characteristic (ROC) curves in Fig. 4.2b. They represent the $\mathcal{P}_{md}$ as a function of the $\mathcal{P}_{fa}$, so the more the curve is in the bottom-left corner, the better. The reference does not fit on the figure as it is too good, but is represented as a gold pentagon on the targeted corner. Another advantage of this representation is that it does not require the centering nor the reduction of the score, and consequently, are less prone to statistical analysis errors. To better demonstrate the behavior of each norm, results for simulations over a synchronous channel are plotted using solid curves, and results for simulations with a remaining symbol rotation $\varphi < \frac{\pi}{2}$ are plotted using dashed curves.

With this representation, the negative impact of $\mathcal{L}_\infty$ become obvious. It does not even meet the required $\mathcal{P}_{md} < 10^{-4}$ for $\mathcal{P}_{fa} = 10^{-6}$. Retrospectively, it should have been anticipated. Indeed, since the infinite norm results in a normalization factor resulting directly from one chip, it tends to smooth out the score.

More importantly, at $\mathcal{P}_{fa} = 10^{-6}$, the green curve ($\mathcal{L}_1$) and the indigo curve ($\mathcal{L}_d$) are notably higher than the blue one ($\mathcal{L}_2$), at respectively $\mathcal{P}_{md} \simeq 6.3 \times 10^{-4}$, $\mathcal{P}_{md} \simeq 7 \times 10^{-4}$, and $\mathcal{P}_{md} \simeq 4.7 \times 10^{-4}$. A zoom on the relevant $(\mathcal{P}_{fa}, \mathcal{P}_{md})$ intervals have been represented in Fig. 4.2c.

In summary, $\mathcal{L}_\infty$ has not the lowest complexity, due to the repetitive need of $q(q-1)$ comparisons, and has by far the worst impact on performances. $\mathcal{L}_1$ has both a reasonable complexity and introduces an acceptable degradation of performances, while $\mathcal{L}_d$ has the lowest complexity for a greater penalty on performances. Nevertheless, $\mathcal{L}_2$ has the second-lowest complexity and the lowest impact on performances. It is undoubtedly the best option of the three at low SNR.

From now on, the score is assumed to be normalized by the $\mathcal{L}_2$ norm, as depicted in Fig. 4.1, and the score S_n^ω is redefined as

$$S_n^\omega = S_{n-q}^\omega + \frac{M_n^\omega}{\mathcal{L}_2(\boldsymbol{y}_n)} - \frac{M_{n-N*q}^\omega}{\mathcal{L}_2(\boldsymbol{y}_{n-N*q})} \tag{4.14}$$

4.2 Time sliding windows

The most complex computation involved in the score function described in chapter 3 (and amended in the last section by normalization) is the circular correlation, as defined in (3.4), and represented by the block $\star \boldsymbol{P}_0$ in Fig. 4.1. Indeed, the naive implementation of this algorithm have a computational complexity in $O(q^2)$.

In [93, 99], this step leverages the circular property of the Fast Fourier Transform (FFT), which allows to compute the correlation vector $\boldsymbol{L}_n^\omega$ in the frequency domain in $\mathcal{O}(q \log_2(q))$ operations, using

$$\boldsymbol{L}_n^\omega = \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{y}_n \odot \boldsymbol{\Gamma}_n^\omega) \odot \mathcal{F}(\boldsymbol{P}_0)^*), \tag{4.15}$$

where operator $\odot$ denotes the element-wise product of two vectors, $\mathcal{F}(\boldsymbol{X})$ is the FFT of $\boldsymbol{X}$, $\mathcal{F}^{-1}(\boldsymbol{X})$ is the IFFT of $\boldsymbol{X}$, $\boldsymbol{X}^*$ represents the conjugate vector of $\boldsymbol{X}$, i.e., $\forall \bar{x} \in \boldsymbol{X}^*$, $\bar{x} = \mathcal{R}(x) - j\mathcal{I}(x)$, and $\boldsymbol{\Gamma}_n^\omega$ is the pure sinusoidal rotation vector introduced in (3.9).

While this method benefits from the extensive study of the FFT algorithm and its possible implementations in the literature, it is not well suited to a system that requires the computation of a data flow. Indeed, each new correlation vector $\boldsymbol{L}_n^\omega$ requires a fully buffered input vector $\boldsymbol{y}_n \odot \boldsymbol{\Gamma}_n^\omega$, and a complete re-computation of the first FFT and the IFFT (the result of $\mathcal{F}(\boldsymbol{P}_0)^*$ can be stored in memory), resulting in an SPU possible architecture depicted in Fig. 4.3.

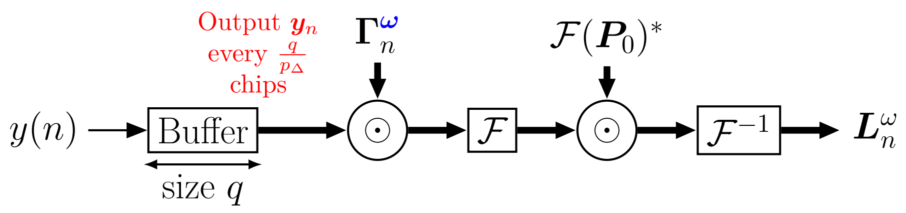

Output $\boldsymbol{y}_n$ every $\frac{q}{p_\Delta}$ chips
$\boldsymbol{\Gamma}_n^\omega$
$\mathcal{F}(\boldsymbol{P}_0)^*$
$y(n)$
Buffer
size q
$\odot$
$\mathcal{F}$
$\mathcal{F}^{-1}$
$\boldsymbol{L}_n^\omega$

Figure 4.3 – FFT based correlation possible architecture for a given frequency offset value ω.

In this section, the frequency domain computation of the correlation vector $\boldsymbol{L}_n^\omega$ is replaced by a "Time Sliding" (TS) computation. The name "Time Sliding" comes from the computation scheduling that uses the circular property of the CCSK modulation to reduce dramatically the computation burden.

4.2.1 Principle

First, it should be noted that to lighten notations, $y_n(i)$ as defined in (3.8) is used. Moreover, we have the following useful properties for all $(n,i) \in \mathbb{N}^2$:

$$y_n(-1) = y(n-q), \tag{4.16}$$

$$y_n(q-1) = y(n), \tag{4.17}$$

$$y_{n-1}(i) = y_n(i-1). \tag{4.18}$$

Let us express the k^{th} component $L_n^\omega(k)$ of $\boldsymbol{L}_n^\omega$ given in (4.15) in the time domain:

$$L_n^\omega(k) = \sum_{i=0}^{q-1} y_n(i) p_k(i) e^{-j\frac{i\omega}{q}} \tag{4.19}$$

Since M_n^ω is equal to $\mathcal{L}_\infty(\boldsymbol{L}_n^\omega)$, its value is not affected if $\boldsymbol{L}_n^\omega$ in (4.19) is replaced by $\bar{\boldsymbol{L}}_n^\omega$ such that:

$$\bar{L}_n^\omega(k) = \sum_{i=0}^{q-1} y_n(i) p_k(i) e^{-j(n-q+i)\frac{\omega}{q}}, \tag{4.20}$$

$\bar{L}_n^\omega(k)$ can be expressed as a function of $\bar{L}_{n-1}^\omega(k-1)$ and the values $y(n-q)$ and $y(n)$. According to (4.20), $\bar{L}_{n-1}^\omega(k-1)$ becomes

$$\bar{L}_{n-1}^\omega(k-1) = \sum_{i=0}^{q-1} y_{n-1}(i) p_{k-1}(i) e^{-j(n-1-q+i)\frac{\omega}{q}} \tag{4.21}$$

By the definition of the CCSK modulation in (3.3), for all k and i values,

$$p_{k-1}(i) = p_0(k-1+i) = p_k(i-1) \tag{4.22}$$

where additions are performed modulo q, thus (4.21) can be rewritten as

$$\bar{L}_{n-1}^\omega(k-1) = \sum_{i=0}^{q-1} y_n(i-1) p_k(i-1) e^{-j(n-q+i-1)\frac{\omega}{q}} \tag{4.23}$$

By changing the summation i by $i' = i-1$, (4.23) can be rewritten as

$$\bar{L}_{n-1}^\omega(k-1) = \sum_{i'=-1}^{q-2} y_n(i') p_k(i') e^{-j(n-(q-1)+i')\frac{\omega}{q}} \tag{4.24}$$

Thus, subtracting $y_n(-1)p_k(-1)e^{-j(n-q)\frac{\omega}{q}}$, adding $y_n(q-1)p_k(q-1)e^{-jn\frac{\omega}{q}}$ to $\bar{L}^{\omega}_{n-1}(k-1)$, and injecting (4.16) and (4.17) gives $\bar{L}^{\omega}_{n}(k)$. In summary, (4.19) becomes:

$$\bar{L}^{\omega}_{n}(k) = \bar{L}^{\omega}_{n-1}(k-1) + p_k(q-1)d^{\omega}_{n}, \tag{4.25}$$

where d^{ω}_{n} is defined as the iterative factor at time n for the rotation ω, and is given by

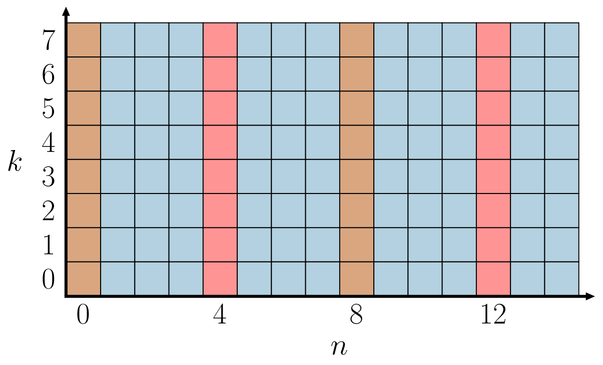
$$\text{distributed:} \quad d^{\omega}_{n} = y(n)e^{-jn\frac{\omega}{q}} - y(n-q)e^{-j\frac{\omega}{q}(n-q)}, \tag{4.26}$$

$$\text{factorized:} \quad d^{\omega}_{n} = (y(n) - y(n-q)e^{j\omega})e^{-jn\frac{\omega}{q}}. \tag{4.27}$$

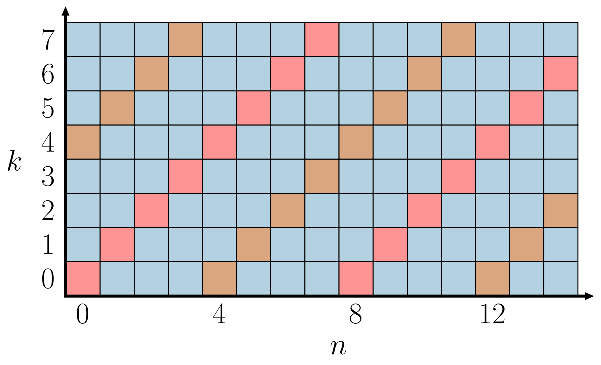
It modifies how SPUs (as depicted in Fig. 4.1) produce correlation values. This is depicted in Fig. 4.4, which compares the scheduling of the correlation for $p_\Delta = 2$ SPUs (FFT based on Fig. 4.4a, and TS based on Fig. 4.4b), with $q = 8$. In both cases, q values $L^{\omega}_{n}(k)$ are computed but with a fundamental difference:

FFT — A full vector $\boldsymbol{L}^{\omega}_{n}$ of length q is computed by each unique SPU. Every vertical bar of the same color in Fig. 4.4a thus represents the output of a unique SPU.

TS — Each SPU generates a diagonal of elements $L^{\omega}_{n+k}(k \mod n)$ every q chips. Every diagonal of the same color in Fig. 4.4b thus represents the output of a unique SPU. However, each SPU generates a unique value for each chip, resulting in p_Δ available values in total instead of a complete $\boldsymbol{L}^{\omega}_{n}$.



(a) FFT based



(b) TS based

Figure 4.4 – Comparison between frequency based and time based computation of correlations for $q = 8$ and $p_\Delta = 2$. Red and orange points correspond respectively to the output of two distinct SPUs.

Consequently, setting p_Δ to a value lower than q with TS based SPUs is similar to ignore some possible circular rotations $\boldsymbol{P}_k$ in (4.25) for all n. Unsurprisingly, this has a heavy negative impact on detection performances. If it could be interesting to explore different trade-offs, the loss is not affordable when targeting very low SNR as those depicted in Fig. 3.9.

Thus, in the current work, a complete $\boldsymbol{L}_n^\omega$ vector has been required to be available at time n, leading to the enforcement of $p_\Delta = q$ when the TS method is used.

Iterative factor precision issue

The factorized version of the recursive equation (4.27) can require fewer multiplications depending on $e^{j\omega}$. Indeed, when ω is a multiple of π or of $\frac{\pi}{2}$, $e^{j\omega}$ is resp. in $\{-1,\ 1\}$ or in $\{-1,\ -j,\ 1,\ j\}$. Besides, thanks to a trick inspired from [105], it is also the case when ω is a multiple of $\frac{\pi}{4}$. Indeed, one can write locally work with 2ω multiple of $\frac{\pi}{2}$, and multiply $y(n)$ by $e^{-j\omega}$ in input.

All those versions can be exactly implemented as long as the multiplications by $e^{j\omega}$ and $e^{-jn\frac{\omega}{q}}$ do not introduce rounding errors that would be accumulated in the integrator. This is not the case for the values $p_\omega \in \{1, 2, 4\}$ of interest (corresponding to $\omega = \pi, \frac{\pi}{2}, \frac{\pi}{4}$ resp.), at least when a 32-bits *single-precision* floating-point representation (as defined by the IEEE 754 standard [106]) of the numbers is used. For higher values of p_ω, *double-precision* has been required. Indeed, it has been observed during extensive simulations, that the correlation values drifted otherwise.

This can be explained mathematically. Let's suppose that $e^{j\omega}$ can be represented without error, but that $e^{-jn\frac{\omega}{q}}$ introduces an error ϵ_n, such that (4.27) becomes

$$\bar{d}_n^\omega = (y(n) - y(n-q)e^{j\omega})(e^{-jn\frac{\omega}{q}} + \epsilon_n). \tag{4.28}$$

Thus, expressing (4.28) q chips later gives

$$\begin{aligned} \bar{d}_{n+q}^\omega &= (y(n+q) - y(n)e^{j\omega})(e^{-j(n+q)\frac{\omega}{q}} + \epsilon_{n+q}), \\ &= (y(n+q) - y(n)e^{j\omega})(e^{-jn\frac{\omega}{q}}e^{-j\omega} + \epsilon_{n+q}). \end{aligned} \tag{4.29}$$

Finally, subtracting (4.28) to (4.29) does not completely remove $y(n)$ if $\epsilon_{n+q} \neq \epsilon_n$, which can alone explain the drift observed when $|\epsilon_{n+q} - \epsilon_n| \neq 0$. Worse, it can also be true if $y(n) - y(n)e^{j\omega}e^{-j\omega} \neq 0$, which can happen with floating-point arithmetic, or with fixed-point arithmetic if the quantization is unbalanced (e.g. a binary fixed-point representation with two complement and without saturation).

Considering this difficulty, it has been decided to handle rotation in input, using the distributed version of the iterative factor (4.26). It has the drawback to require p_ω complex multipliers, and p_ω FIFOs of q elements, to store the values $y(i)e^{-j\frac{i\omega}{q}}$ for all i in $[\![0,\ q-1]\!]$.

Architectural variations

Even with these considerations (the requirement of $p_\Delta = q$, and the need to use a distributed iterative factor computation), a degree of liberty remains. Indeed, to implement (4.25), it is possible to cycle the $L_n^\omega(k)$ values for all k in $[\![0,\ q-1]\!]$ (resulting in the architecture proposed in Fig. 4.5a), or to accumulate those values in place, circularly shifting the indexes instead (resulting in the architecture proposed in Fig. 4.5b).

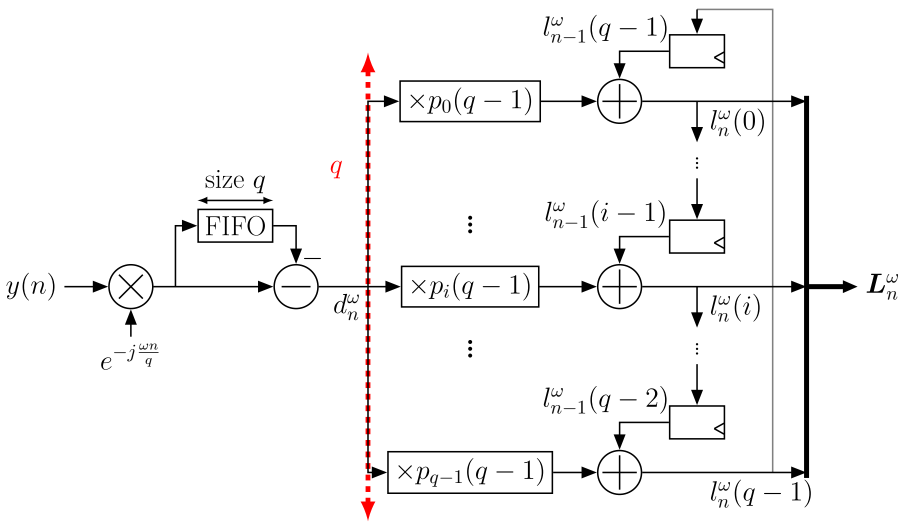
Both approaches produce the exact same result, with the $L_n^\omega(k)$ values being not ordered when using the index shifting method. However, since the step directly following the correlation is the extraction of $\mathcal{L}_\infty(\boldsymbol{L}_n^\omega)$, the absolute maximum of $\boldsymbol{L}_n^\omega$ (see Fig. 4.1), the reordering of data is not needed. Indeed, the index of the maximum is not used in the current detection algorithm.

For a software implementation, the two approaches should be equivalent. Both are embarrassingly parallel and can be vectorized using Single Instruction Multiple Data (SIMD) instructions [60], like the AVX and AVX-2 instruction set extensions of Intel [61] for their x86 64 bits CPUs. For hardware implementations targeting FPGA or ASIC circuits however, there are two main differences.

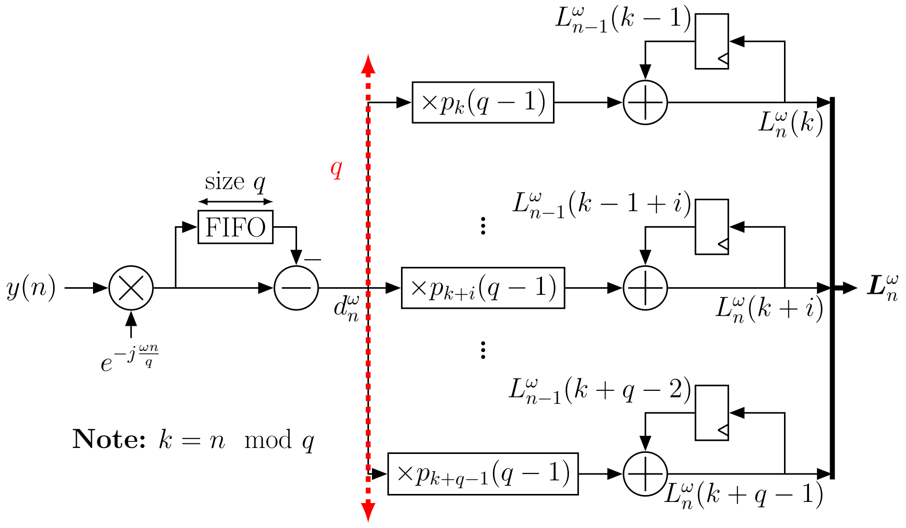
On one hand, the data shifting architecture allows more refinement, especially when $\boldsymbol{P}_0$ is a fixed parameter. In that case, the multipliers (represented by the $\times p_i(q-1)$ nodes on Fig. 4.5a) can be replaced by a single register, or a sign inverter followed by a register, depending on the value of $p_i(q-1)$. If the index shifting method also allows a similar optimization, it still requires more logic, as it should use either multiplexers or conditional sign inverters, since $p_{k+i}(q-1)$ values varies over time.

On the other hand, and especially when q value grows on FPGA circuits, the data shifting approach would be harder and harder to route, as all $L_n^\omega(i)$ values depend on each other, and must be linked. The index shifting architecture accumulating values "in-place", it is less affected by this issue.

Architectures impact on performances, the different levels of parallelism they offer, and how they can be alleviated are the topic of the next chapter. For now, the focus is put on the algorithm, whether it uses FFT or TS based correlations. The main algorithm settings impacting frame detection are the number of score values computed every q chips (p_Δ) and the number of frequency hypotheses tested over the interval $[-\pi, \pi]$ (p_ω). Besides, the



(a) Data shifting: $L^{\omega}_{n}(k)$ values are propagated, indexes are static.



(b) Index shifting: $L^{\omega}_{n}(k)$ values are accumulated in place, indexes are cycling.

Figure 4.5 – Time Sliding (TS) correlation possible architectures for a given frequency offset value ω.

settings of the communication N, the number of symbols in a frame, and q, the number of chip per symbol are known for their direct effect on performances. These parameters directly impact performances and complexity. The following section is dedicated to the performance and complexity comparison.

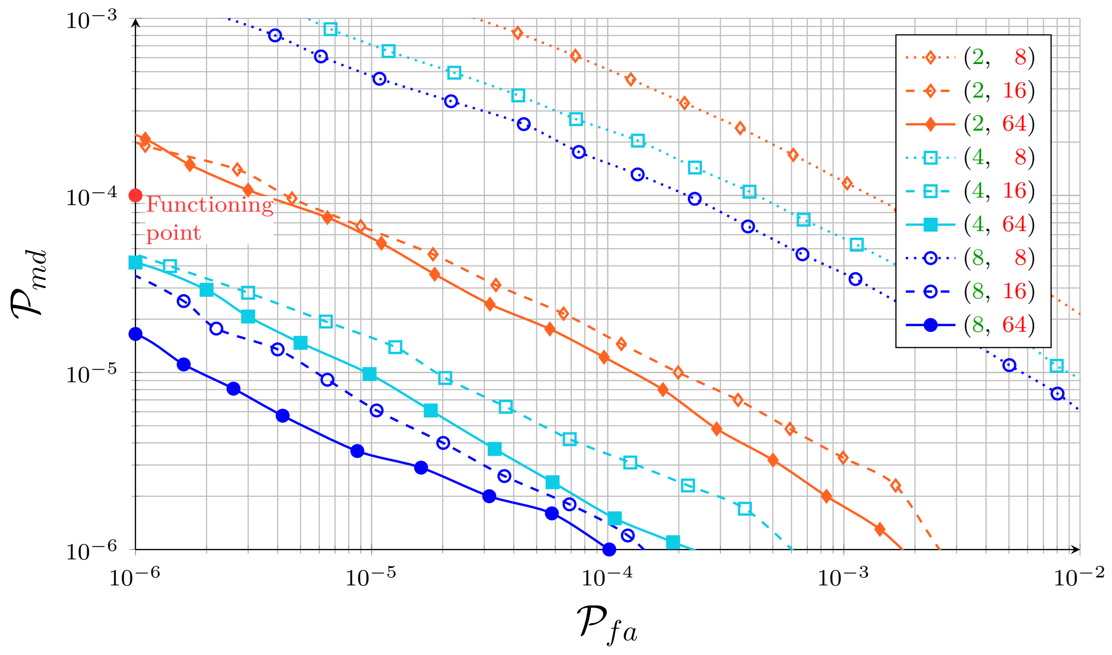


Figure 4.6 – Resulting ROC curves for frame of $N = 60$ symbols, symbols of $q = 64$ chips, at SNR of -10 dB for several setup of the couple of parameters (p_ω, p_Δ).

4.2.2 Detection performance and complexity comparisons

4.2.2.1 Impact of (p_ω, p_Δ) on detection performances

In this section, we compare the performance in terms of Receiver Operating Characteristic (ROC) as in section 4.1. As previously mentioned, ROC curve gives the evolution of the probability of Miss Detection ($\mathcal{P}_{md}$) function of the probability of False Alarm ($\mathcal{P}_{fa}$). To compare with work reported in [17], the frame size is first set to $N = 60$ with a CCSK length of $q = 64$.

The impact of values of the couple (p_ω, p_Δ) on detection performances are presented in Fig. 4.6. Different configurations were evaluated with $p_\Delta \in \{8, 16, 64\}$, which correspond to a score value produced every 8, 4 and 1 chips respectively, and $p_\omega \in \{2, 4, 8\}$ which correspond to a maximum remaining rotation error $\vartheta = \frac{\pi}{2}, \frac{\pi}{4}$ and $\frac{\pi}{8}$ respectively.

The first observation, is that a linear increase of the total computational complexity (represented by the product $p_\Delta p_\omega$) does not result in a linear performance improvement. If doubling p_ω from 2 to 4 significantly improves performances for every possible p_Δ, it is not the case for doubling p_ω from 4 to 8. This can be easily observed on Fig. 4.7, which represents $\mathcal{P}_{md}$ function of (p_ω, p_Δ) for $\mathcal{P}_{fa} = 10^{-6}$, associating a color colder when $\mathcal{P}_{md}$ increases, warmer otherwise. Going from (p_ω, p_Δ) = (2, 8) in the bottom left corner to

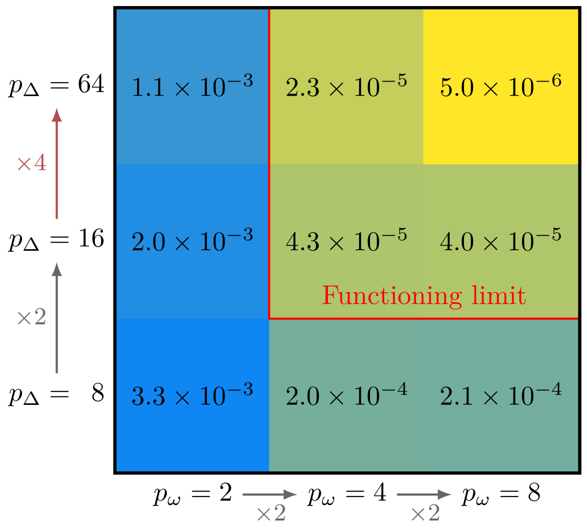


Figure 4.7 – Impact of the couple $(p_\omega,\ p_\Delta)$ on $\mathcal{P}_{md}$ for a $\mathcal{P}_{fa} = 10^{-6}$, represented on a color map. The colder the color is, the higher $\mathcal{P}_{md}$ is (thus, the lower the performances are).

$(p_\omega,\ p_\Delta) = (4,\ 8)$ at the right lowers $\mathcal{P}_{md}$ of more than a decade, while repeating the same operation to go to $(p_\omega,\ p_\Delta) = (8,\ 8)$ in the bottom right corner does not change $\mathcal{P}_{md}$ significantly.

Moreover, a given $p_\Delta p_\omega$ product lead to different performances for each different $(p_\omega,\ p_\Delta)$ couple. In other words, a constant total parallelism level (thus, a constant computational complexity) does not ensure the same performances. For example, achieving $p_\Delta p_\omega = 128$ with $(p_\omega = 2, p_\Delta = 64)$ results in a decade and a half of performances loss compared to $(p_\omega = 8, p_\Delta = 16)$ on Fig. 4.6. This is also observable on Fig. 4.7. The optimal path to go from the bottom left-hand corner (highest $\mathcal{P}_{md}$) to the top right-hand corner (lowest $\mathcal{P}_{md}$) consists on increasing p_ω once $(2 \rightarrow 4)$, then increasing p_Δ twice $(8 \rightarrow 16 \rightarrow 64)$ and finally increasing p_ω $(4 \rightarrow 8)$.

Thus, trade-offs are achievable between targeted performances and complexity. For instance, a $\mathcal{P}_{md} < 10^{-4}$ with $\mathcal{P}_{fa} < 10^{-6}$ can be reach for $(p_\Delta = 16, p_\omega = 4)$ resulting in a computational complexity $p_\Delta p_\omega = 64$, which is also the lowest complexity that allows to meet this requirement.

4.2.2.2 Computational complexity comparisons

This section aims to estimate and compare the computational complexity of the legacy correlation method, referred to as the "FFT method", and the computational complexity of the new proposed TS method. Estimation results are provided in Table 4.2, reported in terms of single-precision (32-bit) floating-point additions and multiplications. The memory footprint has also been estimated. In any case, it is supposed that complexities are proportional to p_ω, which is therefore set to one in the table, to lighten the results. It should be noted that *algorithms* themselves are compared, not their implementations and the context-aware optimizations that may be relevant. These topics are addressed in chapter 5.

It seems obvious that for $p_\Delta = q$, the time sliding method should be preferred. Moreover, one can broadly estimate the values of p_Δ for which the complexity of the FFT method remains inferior to the TS method one. Lets denote $\mathcal{C}_{\mathrm{FFT},p_\Delta}$ the complexity in terms of elementary operations of the FFT method depending on p_Δ, and $\mathcal{C}_{\mathrm{TS}}$ the one of the TS method. Elementary operations are the ones used to implement additions and multiplications, e.g. logical NANDs on the silicon substrate. It is assumed that additions are less complex than multiplications. So, we can write the following:

$$p_\Delta q(\log_2(q) + 2) < \mathcal{C}_{\mathrm{FFT},p_\Delta}, \tag{4.30}$$

$$q^2 < \mathcal{C}_{\mathrm{TS}}. \tag{4.31}$$

Lets denotes these lower-bounds $\mathcal{C}^-_{\mathrm{FFT},p_\Delta}$ and $\mathcal{C}^-_{\mathrm{TS}}$. Thus, it is possible to deduce a value p^-_Δ for p_Δ such that $\mathcal{C}^-_{\mathrm{FFT},p_\Delta} < \mathcal{C}^-_{\mathrm{TS}}$. It gives

$$p^-_\Delta q(\log_2(q) + 2) < q^2 \quad \Leftrightarrow \quad p^-_\Delta < \frac{q}{\log_2(q) + 2}, \tag{4.32}$$

which for $q = 64$, gives $p^-_\Delta < 8$. Specifying an upper-bound is not properly possible, since it would require to estimate the complexity of control and memory handling operations, a task too time-consuming compared to its benefits. To better compare each variation, each method have been implemented in software.

Table 4.2 – Computational complexity comparison depending on p_Δ value ($p_\omega = 1$) for each correlation method.

	p_Δ	Add	Multiply	Memory
FFT	$[\![1,\ q-1]\!]$	$2p_\Delta q\log_2(q)$	$p_\Delta q(\log_2(q)+2)$	$(\frac{p_\Delta-1}{p_\Delta} + p_\Delta)q$
	q	$2q^2\log_2(q)$	$q^2(\log_2(q)+2)$	q^2+q+1
TS	q	q^2+q	q^2	$2q$

Table 4.3 – Benchmarking platforms technical characteristics

	Server	**Workstation**	**SBC**
ISA	x86_64	x86_64	ARMv8
CPU family	Intel Xeon	Intel Core	ARM Cortex
CPU model	Gold 6148	i7-7700	A72
CPU sockets	2	1	1
CPU cores	20×2	4	4
Threads/core	2	2	1
BogoMIPS	4800	7200	108
L2 cache/core	1 MiB	250 kiB	250 kib
L3 cache/socket	27.5 MiB	8 MiB	$\emptyset$
Base clock	2.4 GHz	3.6 GHz	1.5 GHz
Max clock	3.7 GHz	4.2 GHz	1.5 GHz
Average clock	3.0 GHz	4.0 GHz	1.5 GHz
RAM	256 GiB	16 GiB	4 GiB

4.2.3 Software implementation, benchmark, and analysis

This first software implementation follows the "*What you see is what you get (WYSIWYG)*" principle, meaning that the code is written to reflect as much as possible what is depicted in Fig. 4.3 and Fig. 4.5. It is written in C/C++14. While TS SPU code is written from scratch, FFT have been implemented using the well-known library FFTW [107]. This may impact the results, as FFTW offers already-optimized FFT implementation. However, it allows focusing on the algorithm itself, rather than on optimizing FFTs.

Three benchmark platforms have been used:

1. a high-end server,
2. a mid-range workstation,
3. a Raspberry Pi 4, a low-cost Single-Board Computer (SBC).

Characteristics of each platform are reported in Table 4.3. All platforms run a GNU/Linux distribution (Namely Ubuntu 22.04 for the server, Debian 11 for the other two) with the same Linux kernel version (v5.15). The server uses GCC v11.2 as compiler, while the other two use GCC v10.2. Since all optimizations are disabled through the use of the *-O0* compiler flag, this should not affect the results. To measure the throughput of each variation, the C++11 "Chrono" API has been used, working on enough samples to ensure a stable CPU clock (typically, more than few millions, to keep the process running for few seconds at least).

The settings used in the detection performance comparisons are used again ($q = 64$, $N = 60$). The data are generated using an NB-LDPC of code rate $\frac{1}{3}$, resulting in an effective code rate $R_{eff} = 1/32$ (see chapter 3). Thus, payload throughput in kilo bits per second (kb/s) are computed from the measured throughput in kilo chips per second (kC/s) by applying R_{eff}. The worst latency corresponds to a detection assessed after two frames ($2 \times N \times q$ chip), while the best latency correspond to a detection that is assessed after only one frame ($N \times q$ chips). Benchmarking results are reported in Table 4.3, for the $p_\Delta \in \{8, 16, 64\}$ and $p_\omega \in \{4, 8\}$. It should be noted that memory usage has not been successfully measured, since the correlation itself has a far lower footprint that the overall benchmark program.

First, it should be noted that results are heavily impacted by the platform, as one could expect. Since all optimizations are disabled, performance variations can be reduced to CPU clock frequency differences and cache sizes. For instance, the SBC's clock is 2.7 times slower than the workstation's average clock, and the latter benefits from an L3 cache. Unsurprisingly, results are roughly 3.5 times better on the workstation.

Second, independently of the platform or the method, doubling p_Δ halves the throughput (thus doubles latencies). The same goes for p_ω. These observations confirm the assumptions made during complexity estimations in the previous section.

Finally, throughput and latencies of the TS method (which requires $p_\Delta = q = 64$) are close to the ones of the FFT method with p_Δ set to 8. When using FFT with p_Δ set to $q = 64$, the throughput is divided by 6 compared to the TS method, dropping from 20.3 kC/s to 3.3 kC/s.

In other words, the TS method allows to achieve the best performances (detection and throughput) for the lowest complexity.

4.3 Conclusion

In this chapter, a solution has been provided to the legacy detection algorithm sensibility to scaling factors in input. This solution takes the form of a normalization at symbol level. Multiple candidate norms have been compared in terms of computational complexity and impact on detection performances. It led to the choice of the $\mathcal{L}_2$, defined in (4.6).

Moreover, a new method to compute the correlations, key piece of the detection algorithm, has been produced. This *Time Sliding* method can reach the highest detection performances possible for a lower complexity than the legacy FFT method. This has been

Table 4.4 – Algorithm benchmarking results depending on the platform used, the values of (p_Δ, p_ω) and the correlation method implemented, for $N = 60$ and $q = 64$.

(a) Server benchmarking results

	p_Δ	p_ω	**Throughput** *kC/s*	**Throughput** *kb/s*	**Worst latency** *ms*	**Best latency** *ms*
FFT	8	4	135	4.2	60.7	28.4
		8	77	2.4	106.4	49.9
	16	4	68	2.1	120.5	56.5
		8	39	1.2	210.1	98.5
	64	4	17	0.5	481.9	225.9
		8	10	0.3	819.2	384.0
TS	64	4	122	3.8	67.1	31.5
		8	69	2.1	118.7	55.7

(b) Workstation benchmarking results

	p_Δ	p_ω	**Throughput** *kC/s*	**Throughput** *kb/s*	**Worst latency** *ms*	**Best latency** *ms*
FFT	8	4	160	5.0	48.0	24.0
		8	91	2.8	84.4	42.2
	16	4	81	2.5	94.8	47.4
		8	46	1.4	167.0	83.5
	64	4	20	0.6	384.0	192.0
		8	11	0.3	698.2	349.1
TS	64	4	130	4.0	59.1	29.5
		8	76	2.4	101.1	50.5

(c) SBC benchmarking results

	p_Δ	p_ω	**Throughput** *kC/s*	**Throughput** *kb/s*	**Worst latency** *ms*	**Best latency** *ms*
FFT	8	4	45.9	1.43	167	84
		8	26.2	0.82	293	147
	16	4	23.1	0.72	332	166
		8	13.0	0.41	591	295
	64	4	5.8	0.18	1324	662
		8	3.3	0.10	2327	1164
TS	64	4	35.4	1.11	217	108
		8	20.3	0.63	378	189

verified by benchmarking software implementations of each method for various parameters. By the very structure of each algorithm, the complexity relations remain the same for any value of N or q. Thus, to limit the amount of tables represented, they are not reported here.

It is worth noting that the work related in this chapter have resulted in a peer-reviewed contribution [16].

Nevertheless, the achieved throughput do not meet the requirements of LPWANs [29], which demand at least 6.25 kb/s of payload throughput. However, each algorithm offers different parallelism levels, and may also benefit from platform and target specific optimizations.

In order to meet the requirements of the LPWAN standard, the next chapter is dedicated to the implementation of the QCSP chain, from transmission to reception, studying target-specific and context-aware optimizations.

Chapter 5

Real-Time Implementation

Contents

5.1 Transmission 78
5.1.1 Principles 78
5.1.2 Transmitter software implementation 80
5.1.3 Transmitter hardware implementation 82
5.2 Detection 85
5.2.1 Critical task identification 86
5.2.2 Correlator inherent parallelism 87
5.2.3 Correlator in-depth complexity study 89
5.2.4 Correlator software implementation 92
5.2.4.1 Analysis of the results 94
5.2.5 Correlator hardware implementation 96
5.2.6 Quantized model of the score processor 98
5.2.6.1 Presentation of the quantized model 99
5.2.6.2 Quantization format effect on detection performances . . 102
5.2.6.3 Preliminary implementation results 102
5.3 Conclusion 104

An algorithm achieving the best theoretical performances possible is interesting to study to get the performance bound of the communication system. However, it has no practical interest if it is too complex to be implemented in a way that satisfies the constraints of the targeted application context.

As written in the previous chapters, the QCSP communication chain allows reducing the resources used for transmitting data in the context of IoT, and more precisely, of LPWANs. These networks are mid-range, e.g. the standard requires from 10 to 15 km of range in the rural space. The transmitting devices are expected to be low-cost and battery powered, thus, the transmission has to be as energy efficient as possible. At the receiving side, the constraints are less severe. However, the complexity of the receivers have to remain restrained, to ensure scalability and flexibility.

Previous work has proved the relevance of the QCSP chain for this context in terms of communication quality and robustness against channel conditions [17]. In the previous chapter, the detection algorithm has been enhanced, making it resilient to input scaling factors and improving its algorithm-architecture adequacy. Particularly, the computational complexity has been reduced, which should ease software and hardware implementations.

This chapter aims to demonstrate the rational of a QCSP chain real-time usage in the context of LPWANs. To this end, the transmission process is addressed, covering both software and hardware implementations. Then, the detection stage is studied. First, its inherent parallelism levels are detailed, followed by the presentation of possible parallelization strategies for software and hardware implementations. Finally, the quantization study performed on the algorithm is described. Note that the half raise cosine filter is not considered in the chapter, since implementing a finite-impulse response filter is not a challenge.

5.1 Transmission

5.1.1 Principles

The transmission side of the QCSP system is not complex compared to the receiver side. However, as transmitting devices are likely to be low-end sensor nodes, that often are battery powered, a special care must be given to implementation properties. As recalled in Fig. 5.1, a QCSP transmitter requires an NB-LDPC encoder, a CCSK mapper and a way to apply the Overmodulation (OM) before BPSK modulation. These functions map the $K \times p$ bits of the payload message $\boldsymbol{M}$ to the $N \times q$ chips of the QCSP frame $\boldsymbol{F}$.

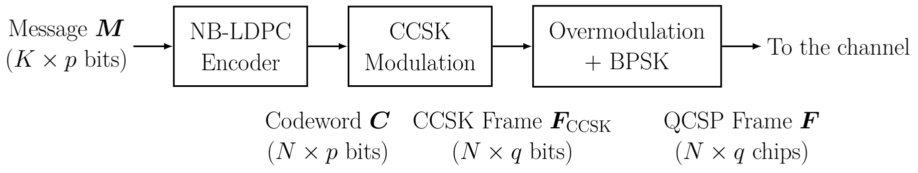


Figure 5.1 – Transmitter side of the QCSP chain.

Let us evacuate the uncertainties on the two uncommon functions, namely the CCSK mapping and OM.

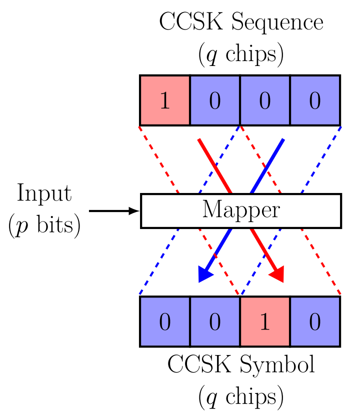
First, let us address CCSK mapping. We consider that the PN sequence $\boldsymbol{P}_0$ is represented as bits in the transmitter, i.e. for all $i \in [\![0, q-1]\!]$, $P_0(i)$ is in $\{0, 1\}$. CCSK mapping then consists on associating a GF(q) of natural value k to the k^{th} circular rotation $\boldsymbol{P}_k$ of the PN sequence, as defined in (3.3). Thus, it can be reduced to a memory mapping. In software (thus, on CPU), this operation can be accelerated using the specialized SIMD instructions if such instructions are supported, or any other feature available. When targeting hardware, other kinds of optimizations can be applied, whether they are FPGAs circuits or ASICs. Their specificity is that the mapping can use LookUp Tables (LUTs) to directly associate the GF(q) symbol binary value to the right CCSK symbol. For instance, for $q = 64$ (i.e. $p = \log_2(q) = 6$), on a Xilinx Artix 7 FPGA which has 6-bits input LUTs, 64 LUTs are enough to implement the mapping. A schematic view of the CCSK mapping is given in Fig. 5.2a for $q = 4$, with red square representing ones and blue square representing zeros.

The OM is not more complex. In (3.5), $\boldsymbol{B}$ was assumed to be composed of $+1$ and -1, to aggregate the BPSK mapping to it. However, as for $\boldsymbol{P}_0$, the OM sequence $\boldsymbol{B}$ can be represented as bits, such that for all $i \in [\![0, N-1]\!]$, $B(i)$ is in $\{0, 1\}$. Thus, the multiplication by $B(i)$ become a "*not exclusive or*" (denoted $\overline{\text{XOR}}$) logical operation, since BPSK can be applied after. Thus, (3.5) become

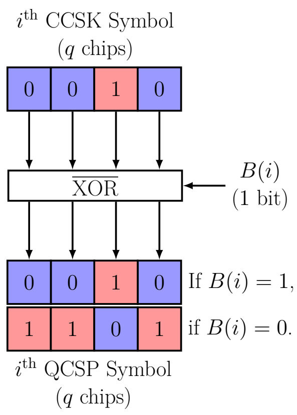
$$\forall k \in [\![0, q-1]\!],\ \forall i \in [\![0, N-1]\!],\ \ F(i \times q + k) = F_{CCSK}(i \times q + k)\, \overline{\text{XOR}}\, B(i). \tag{5.1}$$

A graphic representation of the process is given in Fig. 5.2b, using the same convention and parameters than for CCSK. It should be noted that the optimizations considered for CCSK mapping can be used for OM too. Indeed, $\overline{\text{XOR}}$ SIMD instructions are common on CPUs, and logical evaluation of such $\overline{\text{XOR}}$ is straightforward with LUTs on FPGA circuits.

Thus, the most complex part to implement in the transmitter is the NB-LDPC encoder. For flexibility reasons, the NB-LDPC encoder used in the QCSP chain has not been parallelized. Indeed, without prior information on the NB-LDPC code structure, the encoding algorithm does not have inherent parallelism levels. The encoding must be performed as described by (3.1), by multiplying the message vector $\boldsymbol{M}$ of K symbol by the generating matrix $\mathbf{G}$ of K lines and $M = N - K$ columns. Nevertheless, since N is small



(a) CCSK mapping process as a direct mapping.



(b) OM as a simple logical operation.

Figure 5.2 – QCSP uncommon functions represented for $q = 4$.

(from a few tens to a few hundreds symbols), this operation can be serialized efficiently with a short latency. For instance, for $N = 60$ and a code rate of $\frac{1}{3}$ (thus $K = \frac{N}{3} = 20$), it results in 800 ($K \times M$) multiply accumulate operations over GF(q). It can be implemented using pre-computed LUTs on both CPUs and FPGA circuits.

To illustrate the previous assumptions, and demonstrate the low computational complexity of the QCSP transmitter, it has been implemented on various CPUs and FPGAs circuits. The resulting implementations and their performances are presented in the next sections.

5.1.2 Transmitter software implementation

First, the transmitter has been implemented on software, to benefit from shorter development time. They allow to try different algorithm settings. Such software implementations are already used in other prototyping systems [108]. Moreover, thanks to recent advances in CPU efficiency, a real-time software system is feasible [72]. We decided to target both the low-cost Raspberry Pi 4 ARM SBC and the mid-range Intel x86_64 workstation, previously introduced in Table 4.3 in chapter 4. This choice allows demonstrating the low computational complexity of the transmitter, and its real-time capabilities. Besides, the software flexibility eases the use of Software-defined Radio (SdR) modules.

The transmitter stack has been split in functions, which have in turn been implemented in C/C++14. To improve the performances of the software implementation, the source codes are finely tuned to benefit from GCC compiler optimization routines. The options passed to the compiler during the compilation (e.g. *-O3 -mcpu=native*) enable all

Table 5.1 – Performance of QCSP transmitter software implementation for a payload of 120 bits, $q = 64$, and $N = 60$.

Target		**Clock**	**Throughput**		**Latency**	**Power**	**Energy**
System	*CPU*	*GHz*	*Mc/s*	*Mb/s*	*μs*	*mW*	*nJ/b*
SBC	Cortex A72	1.5	70	2.19	53.0	95	42
Workstation	Core i7	4.0	700	21.9	5.3	102×10^3	4500

optimizations available for the CPU targeted (e.g. specific SIMD instructions). Moreover, the source codes have been written to be portable, i.e. target independent (ARM or Intel) and operating-system independent (Linux, MAC OS, Windows). For instance, the CCSK mapping is implemented using the rotate_copy function defined in the C++14 standard, which result in the following code:

```
for (int i = 0; i < N; i++) {
  int *        symbol_i = frame_ccsk + i * q;
  const int * Pk        = P0 + codeword[i];
  const int * Pe        = P0 + q;

  // do: [P0, ..., Pk, ..., Pe - 1] → [Pk, ..., Pe - 1, P0, ..., Pk - 1 ]
  std::rotate_copy(P0, Pk, Pe, symbol);
}
```

In this code, P0 is a pointer to an address in memory which stores $\boldsymbol{P}_0$, and frame_ccsk is a pointer to the beginning of $\boldsymbol{F}_{CCSK}$. N and q are constexpr constant, known at compile time, allowing the compiler to optimize loops and dependent function calls more aggressively. codeword is an array of integers that represents $\boldsymbol{C}$. While being a direct description of CCSK mapping, the constexpr feature coupled with the fact that rotate_copy is provided by the already optimized standard C++ library, result in an efficient and portable software implementation.

This implementation has been benchmarked on each platform, by simulating the transmission of 10^4 frames, as fast as possible. The resulting throughputs, latencies, and amounts of energy consumed per payload bit are given in Table 5.1 when $q = 64$, $N = 60$ and with a code rate $R_c = \frac{1}{3}$ (thus a payload of $\log_2(q) \times K = 120$ bits). Energies are measured with a Hall-effect measurement device put on the power cord for the SBC, and using the Running Average Power Limit (RAPL) API of Intel for the workstation.

On one hand, the results of the workstation, which is a prototyping platform, are really high for the context targeted. It achieves a chip rate of 700 Mc/s, which translates to 21.9 Mb/s, comparable to the throughput required by the Wi-Fi. Its energy consumption of 4.5 μJ/b is not compliant with IoT constraints. On another hand, on the ARM SBC closer to actual devices, achieved throughputs are 10 times lower. Nonetheless, the full software stack bit rate of 2.19 Mb/s on the SBC is 20 times greater than the one required by LPWANs (which is a little lower than 100 kb/s). It is also 40 times higher than the fastest standard LoRa (using SF7). The energy consumed per payload bit transmitted on the SBC has been measured at 42 nJ/bit. In comparison, the workstation consumes a hundred time more.

The difference between the results of the SBC and the workstation can be explained by their architectural differences:

1. the workstation's clock frequency is 2.7 times higher,
2. the SIMD widths is 2 times higher on the Intel CPU compared to the ARM ones.

Assuming results can be transposed linearly depending on N [1], and that q does not impact throughput [2], the current ARM platform should be able to meet LPWAN requirements for any value of q, and any value of N up to a thousand of symbols. In other words, it demonstrates the viability of the QCSP waveform for IoT devices.

The transmitter has also been implemented on hardware targets. Indeed, current Software-defined Radio (SdR) modules, such as the Ettus Research Universal Software-define Radio Peripheral (USRP) B205 mini or the HackRF, include FPGA circuits. So, it is possible to integrate the transmitter stack in hardware to reduce the stress put on the CPU.

5.1.3 Transmitter hardware implementation

Implementing the transmitting stack on FPGA can help to reduce the power consumption, thus improving efficiency, thanks to dedicated hardware architecture. It is well known that RTL description of digital architectures is time-consuming and provides low design exploration capabilities. Consequently, we decided to take advantage of High-Level Synthesis (HLS) design flows. HLS flow allows the designer to describe the behavior of the system in C/C++. However, all the features of the C/C++ language are not available,

1. It can be safely assumed, since no optimization has been performed at the frame level. It will just fulfill pipelines more efficiently.

2. Since values are represented by 32-bit integers, the performances should not be impacted while q remains inferior to 2^{32}.

and mastering the workflow requires an increase in competencies. It still reduces substantially the required design times. Thanks to fast design exploration features provided by the Xilinx Vivado HLS tool, different hardware architectures have been designed for several low-end Xilinx 7 Series FPGAs, as presented in Table 5.2.

Different kinds of architectures have been developed, and two of them are presented. The first architecture has been designed with timings as a priority, maximizing the throughput and minimizing the latency, no matter the cost in resources or energy. The second architecture targeted a different trade-off, aiming to reduce resource utilization without sacrificing timing performances.

They were obtained using the same C/C++ behavioral description, by inserting different #pragma directives supported by Vivado HLS. Both architectures are composed of a task pipeline to improve the processing throughput. It is achieved using a *dataflow* directive on the complete transmitter. In the first architecture, *pipeline* directives are applied to the tasks internally, to improve their own throughput. In the second one, only the loops inside the functions are pipelined. Function level pipelining is disabled, to save resources. The *pipeline* directive instructs the HLS tool to divide the loop or the function into steps separated by registers, effectively breaking the critical path, and thus, improving throughput. The *dataflow* directive takes advantage of task-level parallelism to start the execution of a task from the already produced data of a previous dependent task.

Implementation results are reported in Table 5.3 for the transmitter settings $q = 64$, $N = 60$, and $K = 20$. Targets are clocked at 100 MHz. The table includes the transmitter chip rate in Mc/s, the corresponding payload bit rate in Mb/s, and the latency in μs. It is accompanied by the resource utilization, the package (static and dynamic) power consumption in mW, the energy consumed per payload bit in nJ/b.

Results have been gathered after place and route stage on the circuit. The reported power usage was gathered from Xilinx Vivado, assuming a room temperature of 25°C.

The slowest implementation (Arch. 2 on the Nexys A7) consumes 401 LUTs, 374 FFs

Table 5.2 – Targeted FPGA characteristics and price tag

Board	**FPGA**		**Price (€)**	
Name	**Family**	**Serial**	**Board**	**Chip**
Nexys A7	Artix 7	XC7A100T-1	375	160
CMOD A7	Artix 7	XC7A35T-1	110	50
Arty S7	Spartan 7	XC7S50-1	165	65

Table 5.3 – Performance of QCSP hardware stack on Xilinx FPGA clocked at 100 MHz.

Target	**Arch.**	**Throughput**		**Latency**	**Usage**			**Power**	**Energy**
Name	*id*	*Mc/s*	*Mb/s*	*µs*	*FF*	*LUT*	*BRAM*	*mW*	*nJ/b*
Nexys A7	1	310	9.7	12	870	958	11	255	25.5
	2	86	2.7	88	374	401	12	129	94.6
Arty S7	1	310	9.7	12	870	958	11	112	11.2
	2	96	3.0	80	373	409	12	89	59.3
CMOD A7	1	310	9.7	12	870	958	11	112	11.2
	2	86	2.7	88	370	390	12	90	66.0

and 12 BRAMs. It achieves a bit rate of 2.7 Mb/s, for a latency of 88 μs. Surprisingly, it is on par with the ARM-based software implementation. Worse, it consumes more energy per payload bit, due to the higher latency on one hand, and due to the Artix 7 100T non-negligible static power consumption on another hand. These results are in fact expected, since the current designs consumes less than 12% of the available BRAM, and less than 2% of the available LUTs and FFs. Since the static consumption depends on the circuit size, it appears that the chosen targets are over-sized. It is worth noting that those targets have been selected also for prototyping purposes, since we had development boards which embedded such chips available. Other FPGA product lines exist, like the Low-Power Artix 7, or the ICE family from Lattice Semiconductor. Those have lower static consumption, which would change the results.

Nevertheless, all high throughput architectures are around 4.4 times faster than the ARM-based implementation due to the lower latency. They even challenge the workstation. Indeed, the latter is 2.3 times faster (21.9 Mb/s) than the fastest FPGA implementations. However, these implementations consumes from 400 to up to 900 times less power. They are 200 times more power efficient. This demonstrates that the transmitting hardware stack complies with IoT requirements. Considering the low resource utilization, it leaves a lot of possibilities to improve the design. For instance, one could target smaller circuits, more in line with the needs.

The performances of the hardware transmitter confirm the relevance of the QCSP chain in the context of IoT. We succeeded to implement both software and hardware real-time transmitters, that achieves high throughput (from 20 to 100 times faster than LoRa), with low resource utilization and low energy consumption. The first step to implement the complete real-time QCSP communication chain is thus completed. Let us now focus on the main topic of this chapter, the detector.

5.2 Detection

The most demanding task of the receiver is the detector. It must process all received samples, as fast as possible. For instance, to achieve LPWAN compatible throughput (from 6 kb/s to 100 kb/s [29]), with $q = 64$, it must process the input with a throughput from 192 kc/s, up to 3.2 Mc/s. Moreover, the computations are non-trivial, as described in chapter 3. The detector implementation has to take advantage of multicore CPU or FPGA circuit features. This section presents all the work performed on the QCSP detector implementation, to unlock its full potential. Indeed, this required the fine-tuning of the algorithmic descriptions, to adapt it to each selected target.

The increasing processing performance of multicore and manycore devices, associated with easy-to-use programming models [56, 57], made the implementation of prototypes or real communication systems possible. A software-based implementation may not achieve the throughput and energy efficiency of an ASIC or FPGA implementation. However, it offers a higher flexibility, a higher scalability, and a lower prototyping time than its hardware counterparts. Nevertheless, achieving high performances is challenging and requires algorithm parallelization efforts.

In chapter 4, a new method to implement the correlations required in the QCSP detection algorithm has been presented. This new Time Sliding (TS) method has the advantage in terms of efficiency, and can reach the highest detection performances accessible to a QCSP detector for a lower computational complexity than the legacy FFT method. Two TS correlator architectures have been presented, depicted in Fig. 4.5. In the current chapter, only the "*index shifting*" architecture will be considered. Several parallelism levels have been identified and reported. However, the parallelization levels and associated strategies are influenced by the correlation method used. They are also affected by the waveform settings. Those settings are the frame size N, the size of $\boldsymbol{P}_0$ (q), the number of score computed every q chips p_Δ, and the number of frequency hypotheses p_ω. They are also influenced by the chosen implementation target.

The current section first focuses on identifying the most critical task of the detection system. Then, the inherent parallelism levels of this task are detailed. Subsequently, strategies used to exploit the parallelism in software implementations are presented and benchmarked. The same is performed for hardware implementations. Finally, the first

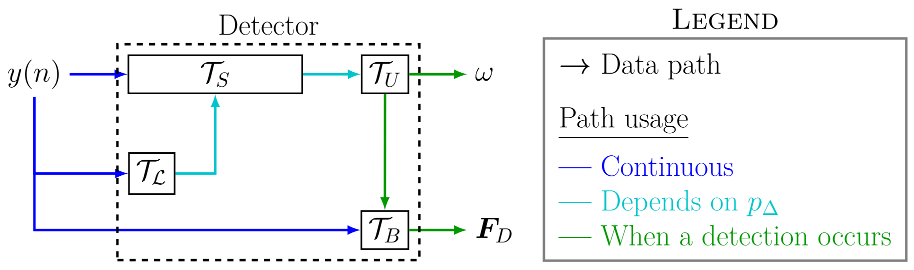


Figure 5.3 – Detection tasks data flow graph, with associated path stress.

drafted quantized model of the QCSP detector is presented and put to test.

5.2.1 Critical task identification

First, let us remember all the tasks involved in the detection process of a QCSP frame, as shown in Fig 5.3:

— The norm calculation ($\mathcal{T}_\mathcal{L}$), which consumes $\boldsymbol{y}_n$ to produce a normalization factor (c.f. section 4.1),

— the score processing ($\mathcal{T}_S$), consuming the same $\boldsymbol{y}_n$ but also the normalization factor to produce a score S_n^ω, associated to a rotation hypothesis ω,

— the state evaluation and updating ($\mathcal{T}_U$), that compares the resulting score S_n^ω to the threshold U_0, but also keeps tracks of the score evolution, in order to select the best score hypothesis,

— the buffering and output on-demand of data ($\mathcal{T}_B$), controlled by the previous tasks, that memorizes the last $\boldsymbol{y}_n$ to output the detected frame buffer $\boldsymbol{F}_D$ when a frame arrival is assumed.

The last three tasks are detailed in section 3.3. All tasks depend on each other, at least indirectly. $\mathcal{T}_S$ requires the normalization factor of $\mathcal{T}_\mathcal{L}$ before accumulating the score. $\mathcal{T}_U$ needs the score output of $\mathcal{T}_S$. $\mathcal{T}_B$ must buffer the new data before it can output $\boldsymbol{F}_D$, as commended by $\mathcal{T}_U$. The different colors on Fig. 5.3 represent the usage rate of the data path. The blue paths are continuously supplied with data. The tasks they link must consume each new chip (or at least buffer it) to be real time. The green paths are activated only when a detection occurs, so they are less critical. In between, the turquoise paths have a usage rate constrained by p_Δ, i.e. they are activated every $\frac{q}{p_\Delta}$ chips (so at least every q chips, but up to each new chip).

In order to optimize the whole detector, one must focus on the $\mathcal{T}_S$, which is the most critical task. Indeed, $\mathcal{T}_U$ consists only on a comparison with the threshold, and a simple state comparison. As for $\mathcal{T}_\mathcal{L}$, a quick look on the complexities previously reported in Table 4.1.2 compared to those in Table 4.2 for just the correlation task of $\mathcal{T}_S$ is enough to measure how much more complex $\mathcal{T}_S$ is.

The most complex subtask of $\mathcal{T}_S$ is the correlation, since the score accumulation can be reduced to a few additions while the maximum and the division are basic operations. It has thus been decided to restrain the optimization study to the correlation subtask of $\mathcal{T}_S$.

5.2.2 Correlator inherent parallelism

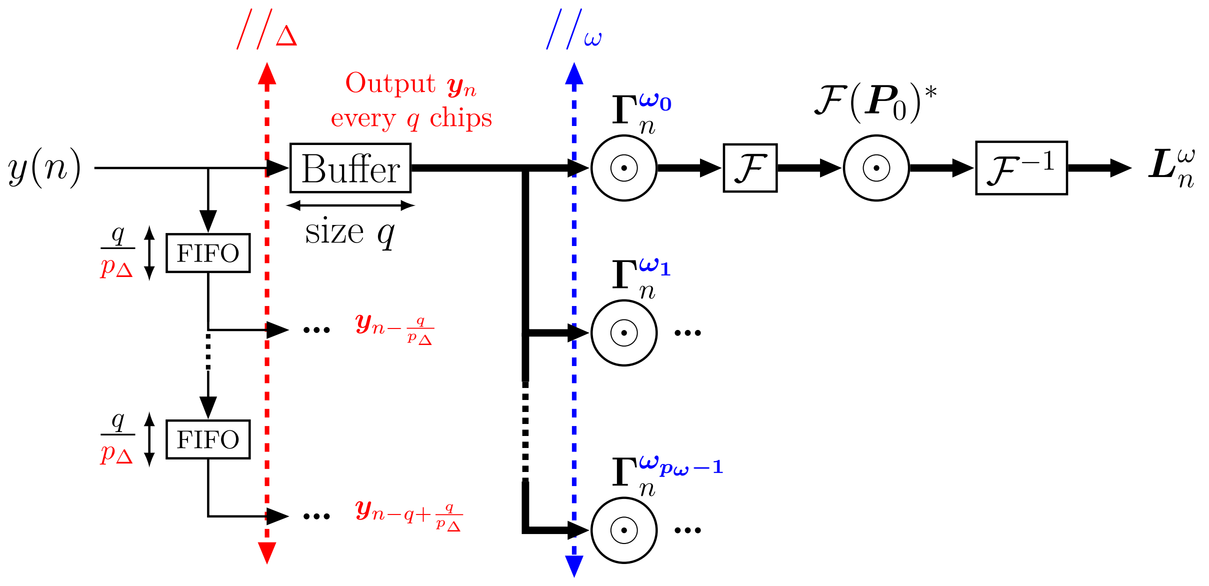
In accordance with the last statement, the optimization and parallelization efforts have been focused on the correlation subtask. Two different methods to compute the correlations exist. In chapter 4, the TS method has been proven to be more suited to the task than the legacy FFT method without optimization. However, taking the context into consideration, and making use of target specific optimization may alter the results.

Let us introduce two naming conventions. First, the block or function that compute a correlation vector $\boldsymbol{L}_n^\omega$ using the TS method is called a TS Correlation Unit (TCU), while the one using the FFT method is called an FFT Correlation Unit (FCU). Second, the system that allows to compute the $p_\Delta \times p_\omega$ correlation vectors $\boldsymbol{L}_{n-m\frac{q}{p_\Delta}}^{\omega_i}$ for each $i \in [\![0, p_\omega - 1]\!]$, and $m \in [\![0, p_\Delta - 1]\!]$, is called the correlator.

The two possible correlator implementations are depicted in Fig. 5.4. The first one in (a) uses FCUs, whereas the second one in (b) use TCUs.
These figures bring to light two coarse grain parallelism levels:

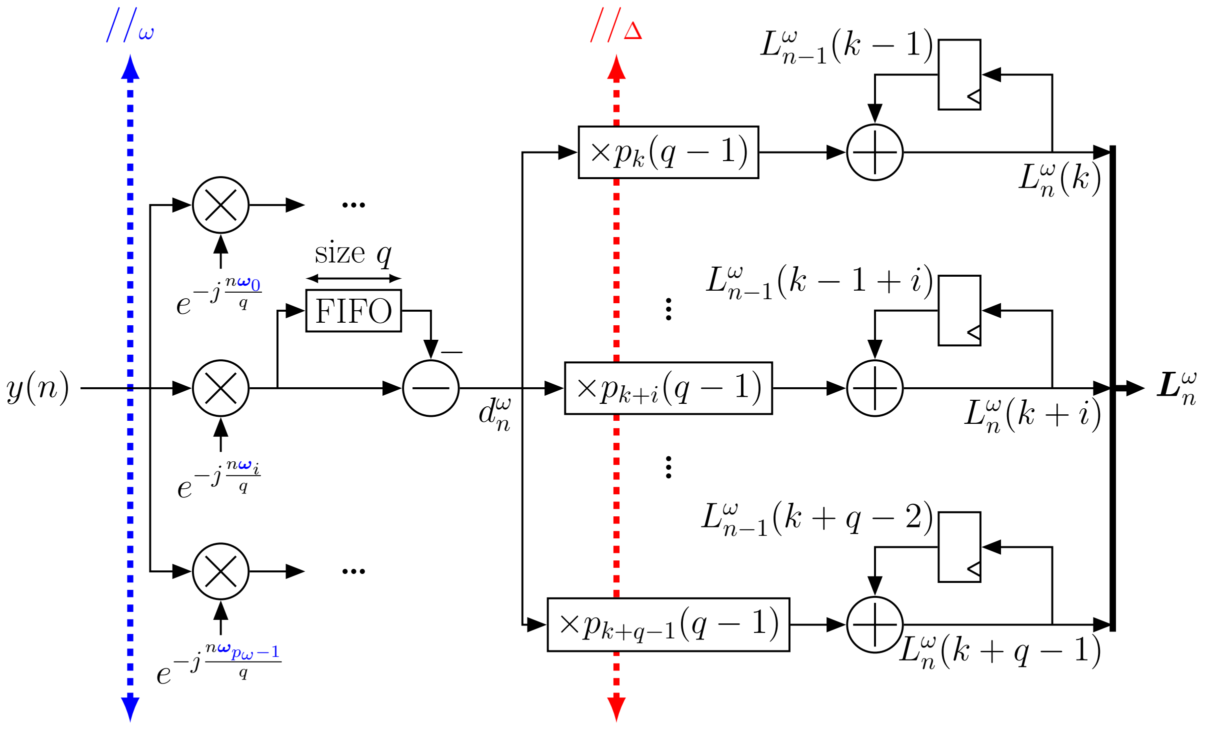
1. The Δ-parallelism ($//_\Delta$), related to p_Δ,
2. the ω-parallelism ($//_\omega$), related to p_ω.

The parameter p_Δ takes value in $[\![1, q]\!]$, and the parameter p_ω is limited to $[\![1, 8]\!]$. The value of 8 ensures compliant detection performances while restraining the computational complexity. The pair (p_ω, p_Δ) impacts on detection performances and on system complexity as reported in chapter 4. Computations performed in any of the $p_\Delta \times p_\omega$ CU are data independent, allowing fully concurrent evaluation. In addition, correlations along $//_\omega$ are time-independent. However, for $//_\Delta$, if for FCUs, each correlation is time-independent of the others, as discussed previously it is not the case for TCUs. The two coarse grain



$//_\Delta$
$//_\omega$
Output $\boldsymbol{y}_n$ every q chips
$y(n)$
Buffer
size q
FIFO
$\frac{q}{p_\Delta}$
$\boldsymbol{y}_{n-\frac{q}{p_\Delta}}$
$\boldsymbol{y}_{n-q+\frac{q}{p_\Delta}}$
$\boldsymbol{\Gamma}_n^{\boldsymbol{\omega_0}}$
$\boldsymbol{\Gamma}_n^{\boldsymbol{\omega_1}}$
$\boldsymbol{\Gamma}_n^{\boldsymbol{\omega_{p_\omega-1}}}$
$\mathcal{F}$
$\mathcal{F}(\boldsymbol{P}_0)^*$
$\mathcal{F}^{-1}$
$\boldsymbol{L}_n^\omega$

(a) Correlator using FFT Correlation Units ($p_\Delta \in [\![1, q]\!]$, $p_\omega \in [\![1, 8]\!]$).



(b) Correlator using TS Correlation Units ($p_\Delta = q$, $p_\omega \in [\![1, 8]\!]$).

Figure 5.4 – The two different correlators represented with their associated Δ-parallelism and ω-parallelism.

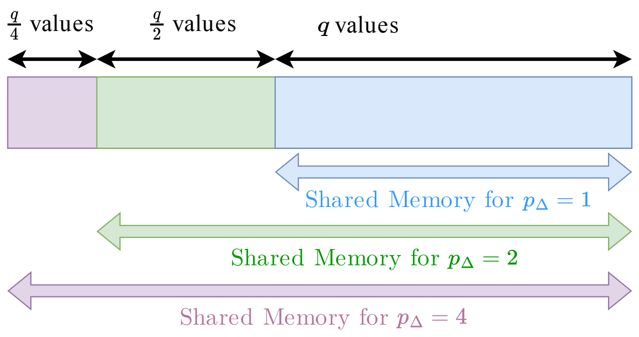

$\frac{q}{4}$ values
$\frac{q}{2}$ values
q values
Shared Memory for $p_\Delta = 1$
Shared Memory for $p_\Delta = 2$
Shared Memory for $p_\Delta = 4$

Figure 5.5 – FCUs shared memory total usage.

parallelism levels enable a wide range of setups. Indeed, the overall correlation task can be seen as a single serialized task or can be split in up to $p_\Delta \times p_\omega$ parallel sub-tasks.

5.2.3 Correlator in-depth complexity study

In this section, optimized versions of each correlator are presented. Indeed, the computational complexity previously reported in Table 4.2 were estimated without specific optimizations. The complexities reported in Table 5.4 correspond to architectures that take advantage of CPU and FPGA circuits features.

As depicted on Fig. 5.4a, each of the p_Δ FCUs requires a vector $\boldsymbol{y}_{n-i\frac{q}{p_\Delta}}$ of q chips. However, the use of p_Δ buffers of q chips is memory inefficient because most values are reused by all CUs. In fact, since the first FCU already stores the q last samples, the second just need $\frac{q}{2}$ samples more, the third and the fourth need $\frac{q}{4}$ in addition, and so on and so forth, as shown on Fig. 5.5. The two memory setups are depicted in Fig. 5.6. Besides, in opposition to what was stated in the previous chapter, the memory does not need to be duplicated for each p_ω hypotheses. This effectively reduces the memory complexity by a factor p_ω. Sharing memory over p_ω and p_Δ allows sparing resources, which translates to a more efficient cache utilization on CPU, and a reduction of hardware complexity on FPGA circuits.

However, to work with shared memory, there is two possibilities. The i^{th} FCU can either receive the values associated to $\boldsymbol{y}_n$ delayed by $\frac{q}{p_\Delta}$, e.g. by using delay buffers on hardware circuits as represented on Fig. 5.6. It can also be assigned the range $[n - i\frac{q}{p_\Delta} - q + 1,\ n - i\frac{q}{p_\Delta}]$ in a shared circular memory, and read it every q chips[3]. This memory

3. Reading it more often is equivalent to serializing ; It would result in recomputing a score already computed (see Fig. 4.4a)

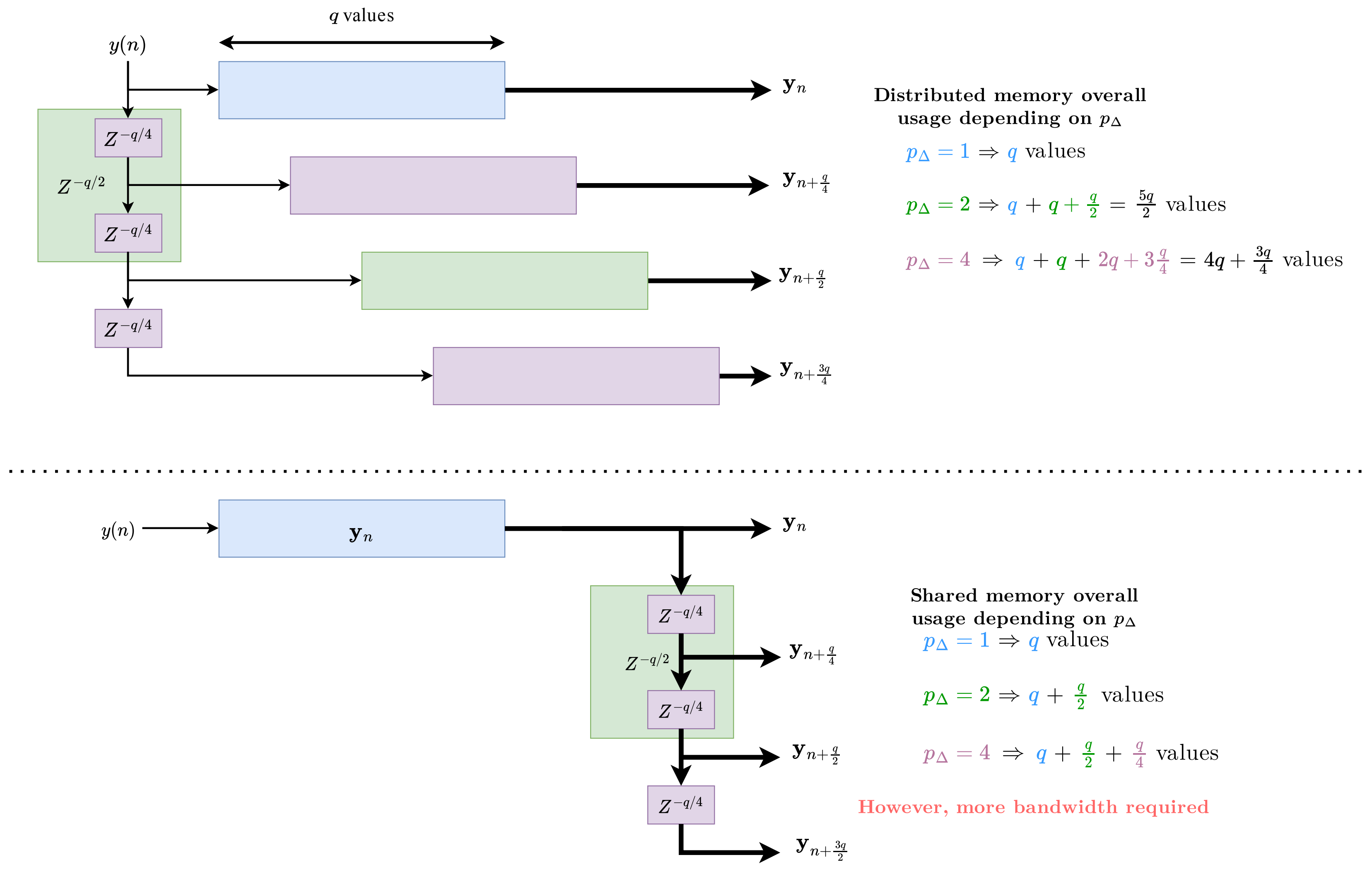

q values
$y(n)$
$Z^{-q/4}$
$Z^{-q/2}$
$Z^{-q/4}$
$Z^{-q/4}$
$\mathbf{y}_n$
$\mathbf{y}_{n+\frac{q}{4}}$
$\mathbf{y}_{n+\frac{q}{2}}$
$\mathbf{y}_{n+\frac{3q}{4}}$
Distributed memory overall usage depending on p_Δ
$p_\Delta = 1 \Rightarrow q$ values
$p_\Delta = 2 \Rightarrow q + q + \frac{q}{2} = \frac{5q}{2}$ values
$p_\Delta = 4 \Rightarrow q + q + 2q + 3\frac{q}{4} = 4q + \frac{3q}{4}$ values
$y(n)$
$\mathbf{y}_n$
$\mathbf{y}_n$
$Z^{-q/4}$
$Z^{-q/2}$
$Z^{-q/4}$
$Z^{-q/4}$
$\mathbf{y}_{n+\frac{q}{4}}$
$\mathbf{y}_{n+\frac{q}{2}}$
$\mathbf{y}_{n+\frac{3q}{2}}$
Shared memory overall usage depending on p_Δ
$p_\Delta = 1 \Rightarrow q$ values
$p_\Delta = 2 \Rightarrow q + \frac{q}{2}$ values
$p_\Delta = 4 \Rightarrow q + \frac{q}{2} + \frac{q}{4}$ values
However, more bandwidth required

Figure 5.6 – FCUs memory can be distributed or shared in an FCU correlator.

range corresponds to $\boldsymbol{y}_{n-i\frac{q}{p_\Delta}}$. This results in an updated memory complexity, reported in Table 5.4, below the shared memory version complexities. From a computational complexity point of view, the FFT algorithm is already optimized. It is assumed that an optimal operator is available, with a complexity of $\frac{q\log_2(q)}{2}$ additions, and of $\frac{q\log_2(q)+2}{2}$ multiplications.

In a TCU correlator, the last multiplication can be replaced by a conditional negation thanks to the binary nature of $\boldsymbol{P}_0$. Since $\forall(k,i) \in [\![0, q-1]\!]^2$, $P_k(i) \in \{1, -1\}$, the multiplication is for all n

$$d_n^\omega \times P_k(i) = \begin{cases} d_n^\omega & \text{if } P_k(i) = 1, \\ -d_n^\omega & \text{if } P_k(i) = -1. \end{cases} \tag{5.2}$$

It reduces by a factor q the number of complex multiplications. This method is reported as the optimized TS in Table 5.4. This is useful for hardware implementations, where complex multiplications consume more resources than conditional negation. The latter can be reduced to the use of multiplexers. This is less relevant on software, since CPUs often integrate specialized floating-point multiplication in their Arithmetic Logic Unit (ALU) and where conditionals result in general in branching and thus loss in throughput, due to pipeline interruptions.

It is worth noting that for integers (thus, in case of fixed-point arithmetic), the operation can be branch-less, by the use of a binary operation. Indeed, for any integer x and denoting $b = P_k(i) < 0$,

$$x \times P_k(i) = x \oplus bb\ldots b + b,$$

with $bb\ldots b$ being b broadcasted to the length of x in bits. This may not benefit to FPGA design, as good synthesis tools may replace the multiplication by LUTs anyway, however it can greatly improve performances on small microcontrolers devoid of dedicated multipliers[4].

To put these algorithm improvements to test, each correlator variation has been implemented on both software and hardware targets, trying to achieve real-time performances.

4. For instance, a 16-bit microcontroller would struggle to implement a multiplication with 32-bit integers, and would perform better with two consecutive binary XOR and one bit addition.

Table 5.4 – Complexity comparison depending on p_Δ and for $p_\omega \in [\![1, 8]\!]$.

Method	$\boldsymbol{p_\Delta}$	**Add**	**Multiply**	**Memory**
FFT (*distributed*)	$[\![1,\ q-1]\!]$	$2p_\omega p_\Delta q \log_2(q)$	$p_\omega p_\Delta q(\log_2(q)+2)$	$(\frac{p_\Delta - 1}{p_\Delta} + p_\Delta)q$
	q	$2p_\omega q^2 \log_2(q)$	$p_\omega q^2(\log_2(q)+2)$	q^2+q+1
FFT (*shared*)	$[\![1,\ q-1]\!]$	*unchanged*	*unchanged*	$\sum_{i=0}^{\log_2(p_\Delta)} 2^{-i} q$
	q	*unchanged*	*unchanged*	$2q-1$
TS (*original*)	q	$p_\omega q(1+q)$	$p_\omega q^2$	$p_\omega 2q$
TS (*optimized*)	*unchanged*	*unchanged*	$p_\omega q$	*unchanged*

5.2.4 Correlator software implementation

The complexity of the receiver is much higher than the transmitter one. Thus, its parallelization and implementation have received a particular care, and required even more efforts than for the transmitter.

The benchmarking platforms are the exact same ones used detailed in Table 4.3. As a reminder, these platforms are a Raspberry Pi 4 ARM low-cost SBC, an Intel workstation and a high-end Intel multicore server.

The correlation algorithms presented before have been described in C++14 language. As for the transmitter, the source code is precisely written, to leverage every available CPU features while remaining portable. It allows to use the same software description on every platform while still taking advantage of the specific features of each CPU, like the dedicated SIMD instructions AVX on Intel platforms, and NEON on ARM. Moreover, the well-known OpenMP API have been used to unlock Multithreading (MT) capabilities, enabling to take advantage of task parallelism. For benchmarking purpose, the code has been compiled with the GNU compiler v10, enabling optimization and unsafe arithmetic. This is done by specifying the following compiler flags: *-Ofast* on all platforms, *-mcpu=native* for the ARM CPU, *-march=native -mtune=native* on Intel CPUs.

The software detector has been implemented for the two detailed CUs variations, for several values of p_Δ and p_ω. FCU software correlator has only been implemented with shared memory, as it allows a better usage of parallelism on CPU. Since several cores can read the memory in parallel, and since the cache is actually a shared memory, the adequacy is maximized. For comparisons, a mono-thread FCU correlator has been implemented, as well as a MT one. Concerning TCUs, as discussed in the previous section, they don't benefit from replacing the multiplication by a conditional negation. Indeed, all platforms have dedicated efficient floating-point multipliers. Even the trick of using the $\overline{\text{XOR}}$ has been benchmarked, and it was always slower. In these implementations, the data are manipulated using single-precision arithmetic to avoid precision issues involved by fixed-point arithmetic.

The parallelization strategy described in section 5.2.2 has been applied to the C++ source codes. For FCUs, the optimized library FFTW3 [107] is still used, its performances being excellent. For the MT version, $//_\Delta$ has been leveraged, using the OpenMP API. We tried to use OpenMP along $//_\omega$ as well, unfortunately, despite our best efforts, it always resulted in a slower implementation. It appears that computations involved in one FCU are too light in regard to the thread synchronization overhead. Parallelization would demand to increase latency, thus filling pipelines, and use the distributed memory version. However, it would require *a lot* of CPU cores. For instance, for $q = 64$, $N = 60$, $p_\Delta = 8$ and $p_\omega = 8$ (which does not even meet detection performance requirement, c.f. Fig. 4.7), it would require 64 cores to have one thread per core and fill CPU pipelines.

At the opposite, the TCU correlator has been implemented only thanks to handmade C/C++ codes. The sources are written using floating point values as well, but first, as close to the algorithm a possible. For instance, it uses `std::complex<float>` standard type and associated functions. In a second time, for performance purpose, the code has also been written in a way that takes advantage of the compiler auto-vectorization feature [109]. It has been achieved by decomposing complex arithmetic into operations on real values, is reported as TS SIMD in Table 5.5. An MT version has not been tried, first because one TCU is less complex than one FCU, thus the observation on thread synchronization overhead is still relevant. Second, as explained in section 5.2.2, $//_\Delta$ is not time-independent in TCUs, making using MT a recipe to disaster ; threads would constantly synchronize with each other, wasting time.

Measured input throughput (in Mc/s) and corresponding payload throughput (in kb/s) for different setups are reported in Table 5.5. Throughputs were measured using the C++14 Chrono API. Latency is left aside this time, as it remains proportional to the throughput. It would not bring information and would clutter the results. In Table 5.5, the MT FCU correlator uses p_Δ threads whereas all the other implementations are single threaded. As it can be seen, the throughput performance depends on the correlation method applied and the selected parameters. Note that for the SBC, input throughputs are reported in kC/s, values being consistently lower than 1 Mc/s.

5.2.4.1 Analysis of the results

Results gathered in Table 5.5 lead to several observations. First and foremost, it is important to note that the TS SIMD correlator, while not exceeding the throughput of MT FFT for $p_\Delta \leq 8$ on ARM, nor for $p_\Delta \leq 16$ on Intel's platforms, is substantially faster when $p_\Delta = 64$. It is twice faster on the server, 3 times faster on the workstation, and more than 5 times faster on the SBC. Considering the single threaded nature of the implementation, it leaves a lot of room to use other correlator in parallel. For instance, this could be leveraged to address oversampling, actually running independent detector in parallel for each sampling hypothesis. Even the not optimized TS beats the MT FFT when $p_\Delta = 64$. The single threaded FFT is completely outperformed for any values, on any platforms. It clearly demonstrates the superiority of the TS approach when targeting the best performances.

Some other results are intriguing. MT FFT throughputs on the multicore server are not always higher than those on the workstation. It can be explained easily though. When all cores are under load, the CPU frequency of the workstation remains set at 4.0 GHz for all cores, while the one of the server drops to 2.7 GHz instead of the peak 3.0 GHz frequency reached when only one core is used. Besides, the use on multiple cores does not result in a linear increase in performances, due partly to thread synchronization. Thus, for lower value of $p_\omega \times p_\Delta$, the server cannot keep up with the workstation. For instance, for $p_\omega = 4$ and $p_\Delta < 64$, the workstation is faster than the server, but for $p_\omega = 8$ and $p_\Delta > 8$, it is the opposite. However, the server uses p_Δ cores, while the workstation only uses 4, to prevent the use of simultaneous MT (SMT). SMT is called hyperthreading on Intel, and may hurt efficiency. It enables 2 threads to run on the same core, but when all threads are fully used, the operating system is more likely to interrupt them. Moreover, while always allowing better raw performances, the gain is lesser than the drawback in terms of consumption [63]. In the same manner, MT FFT is never p_Δ times faster than plain FFT. In fact, it is at most 5.7 times faster, when $p_\Delta = 16$ and $p_\omega = 8$ on the server, and at least 1.9 times, also on the server but for $p_\Delta = 8$ and $p_\omega = 4$. On average, it is 3 times faster than plain FFT on the workstation and on the SBC. So while being performant, the MT FFT approach is not efficient.

Similarly, the TS throughput are practically the same on the server and on the workstation, due to the frequency issue being counterbalanced by the server having access

Table 5.5 – Software optimized implementation throughput depending on the platform used, the values of (p_Δ, p_ω) and the correlation method used, for $N = 60$ and $q = 64$.

(a) Server benchmarking results

p_Δ	p_ω	FFT		MT FFT		TS		TS SIMD	
		Mc/s	*kb/s*	*Mc/s*	*kb/s*	*Mc/s*	*kb/s*	*Mc/s*	*kb/s*
8	4	2.22	69.4	4.30	134.4	N/A	N/A	N/A	N/A
	8	1.33	41.6	4.15	129.7	N/A	N/A	N/A	N/A
16	4	1.81	56.6	3.67	114.7	N/A	N/A	N/A	N/A
	8	0.69	21.6	3.21	100.3	N/A	N/A	N/A	N/A
64	4	0.30	9.4	1.60	50.0	2.21	69.1	3.26	101.9
	8	0.18	5.6	1.03	32.2	1.37	42.8	1.93	60.3

(b) Workstation benchmarking results

p_Δ	p_ω	FFT		MT FFT		TS		TS SIMD	
		Mc/s	*kb/s*	*Mc/s*	*kb/s*	*Mc/s*	*kb/s*	*Mc/s*	*kb/s*
8	4	2.48	77.5	6.49	202.8	N/A	N/A	N/A	N/A
	8	1.45	45.3	4.22	131.9	N/A	N/A	N/A	N/A
16	4	1.28	40.0	4.09	127.8	N/A	N/A	N/A	N/A
	8	0.74	23.1	2.50	78.1	N/A	N/A	N/A	N/A
64	4	0.32	10.0	1.18	36.9	2.23	69.7	3.79	118.4
	8	0.19	5.9	0.70	21.9	1.33	41.6	2.18	68.1

(c) SBC benchmarking results

p_Δ	p_ω	FFT		MT FFT		TS		TS SIMD	
		kC/s	*kb/s*	*kC/s*	*kb/s*	*kC/s*	*kb/s*	*kC/s*	*kb/s*
8	4	328	10.25	702	21.9	N/A	N/A	N/A	N/A
	8	190	5.94	489	15.3	N/A	N/A	N/A	N/A
16	4	169	5.28	460	14.4	N/A	N/A	N/A	N/A
	8	95	2.97	264	8.3	N/A	N/A	N/A	N/A
64	4	43	1.34	127	4.0	364	11.4	669	20.9
	8	24	0.75	74	2.3	192	6.0	410	12.8

to AVX-512 SIMD instructions, which sizes are twice as large than AVX-2 ones on the workstation. Nonetheless, the beneficial effect of the code adaptation to vectorization is obvious, since TS SIMD is 1.7 times faster on the workstation, 1.4 times faster on the server, and around 2 times faster on the SBC.

Based on these conclusions, the most well suited method to implement an efficient QCSP correlator have to be using TCUs. Besides, TS SIMD implementations on all platforms reach the throughput required by the LPWAN standard, even on the Raspberry Pi 4, a low-cost small form factor Single-Board Computer. It demonstrates the viability of the approach for low-power software receiver designs.

However, as stated before, a software implementation does not provide the best solution in terms of throughput and energy consumption. To unlock the full potential of the proposed detector that is massively parallel, FPGA circuits have thus been targeted. They can exploit the inherent parallelism of the detection algorithm, and may result in the best possible implementation.

5.2.5 Correlator hardware implementation

In contrary to the last sections, where software implementations were studied to leverage their flexibility, the current section is focused on hardware implementation that should provide better efficiency.

Like in the transmitter case, HLS workflow has been used to reduce the hardware development time, and keep relative flexibility, compared to traditional RTL design approach. Thus, the C++ detector source codes were rewritten in C language using the HLS compliant C language subset [79] supported by the Xilinx Vivado HLS 2019 [83] tool. To test various architectural variations, for different values of the settings q, N, p_ω, or even p_Δ, descriptions for both detection approaches, have been described to be both efficient and configurable. It has been achieved by using #define and #pragma keywords, as used for the transmitter, to control the HLS design tool. These C models are synthesized for the Xilinx Kintex 7 FPGA XC7K410TFFG900, an FPGA found in the Ettus Research USRP X310. Our goal is to embed our design directly onto the SdR device. The feature that decided this choice is RF Network-on-Chip (RFNoC)[91]. This framework should allow embedding custom IPs into the USRP easily. Achieved chip rate and corresponding payload bit rate are reported in Table 5.6. This values are available after the synthesis of the hardware description, even when the circuit cannot be actually implemented on the target, e.g. when too many resources are required. To ease the readability of the results,

Table 5.6 – Hardware performance of the detector on a Xilinx Kintex 7 XC7410T clocked at 100 MHz, using floating-point arithmetic if not specified, for $q = 64$, $N = 60$ and several values of p_Δ and p_ω.

Method	$\boldsymbol{p_\Delta}$	$\boldsymbol{p_\omega}$	**Chip Rate**	**Bit Rate**	**Usage**			
			Mc/s	*kb/s*	*FF*	*LUT*	*BRAM*	*DSP*
FFT	8	4	2.1	65	insufficient resources			
FFT (*16bits fixed)*	8	4	2.6	81	25726	43206	950	380
		8	2.6	81	51452	94468	1900	760
	16	4	2.6	81	51452	94468	1900	760
		8	2.6	81	102904	188936	3800	1520
TS	64	4	9.1	284	144064	80836	79	383
		8	9.1	284	insufficient resources			

resource utilization is reported in Table 5.6 only for the methods that successfully pass the place and route stage.

To consolidate the observations made for software implementations, both TS and FFT correlators were implemented. Two variants of FFT have been implemented. They were designed with the idea of high throughput at all costs, instantiating $p_\Delta \times p_\omega$ processing sub-chains in parallel. They both have task pipelining and parallel processing enabled. The difference is that the first variant use floating-point arithmetic, which should be the worst case scenario in terms of resource utilization. To improve hardware efficiency and to have a a better estimation a resource utilization, the second variant has been designed with fixed-point arithmetic, with conservative quantization. It should be more efficient than the floating-point version, while not being subject to any performance loss. The FFT models used are derived from work presented in [87]. To estimate the overall detector footprint, results for one FFT have been duplicated. While not giving the exact results, this hypothesis also neglects signal propagations and supposes free synchronization in terms of logic glue, which is in fact clearly in the advantage of the design.

The TS correlator has also been described using pipeline and parallel task execution approaches, as the FFTs. In contrary to the latter, a complete correlator have been implemented this time. Besides, using the simplification mentioned in section 5.2.3, the computational complexity at $//_\Delta$ is significantly lower compared to the FFT ones. This

TS correlator has been implemented using floating-point arithmetic, to ensure undamaged detection performances.

The floating-point FFT detector reaches 2.1 Mc/s before place and route stage for a circuit clocked at 100 MHz. It is on par in terms of throughput than its software single-core counterpart on Intel's platforms clocked at 4 GHz. The failure to meet the resource constraints was expected, since floating-point arithmetic is known to cause issues on hardware implementations. The fixed-point variant performed slightly better, reaching a throughput of 2.9 Mc/s, this time without exceeding available FPGA resources. Concerning the TS correlator, it achieved a throughput 3 times faster than the best hardware FFT implementation, despite the use of floating-point arithmetic. More importantly, its throughput is 1.8 times higher than the best software correlator, the MT FFT, which was set up with $p_\Delta = 8$, while the hardware time-sliding detector has a native $p_\Delta = 64$. It thus results in much higher detection performances, even higher energy efficiency.

These results demonstrate the efficiency of the contributed TS method, at least for $q = 64$. For other q values, based on complexities reported in Table 5.4, it is safe to assume that the TS correlator is advantageous when aiming for the best energy-performances trade-offs.

Since the TS correlator with $p_\omega = 8$ failed to meet the resource constraints, we decided to further improve it. It is due the inadequacy of floating-point arithmetic to hardware implementations. To address the problem, a fixed-point description of the system is need. That is why a quantized model was investigated. Theoretically, this approach should increase both the throughput and the energy efficiency, while reducing resource utilization.

5.2.6 Quantized model of the score processor

This section is dedicated to the presentation of the quantized model of the score processor. Based on previous section results, the FFT method has been discarded, to focus our efforts on the TS one. Moreover, we don't limit the study to the correlator but extend it to the complete score processor. It associates a coarse rotation hypothesis $\hat{\omega}$ and the corresponding score $S_n^{\hat{\omega}}$ to each input $y(n)$. The p_Δ value is set to $q = 2^p$, since TCUs cannot use another value, as discussed in section 4.2. Complex number quantization is denoted as $2 \diamond Q$, with Q the quantization of both the real and the imaginary part, while the quantization of real number is simply denoted Q. For instance, the quantization of a complex number c on 8 bits would be denoted $Q_c = 2 \diamond 8$, while the quantization of a, the real part of c, would simply be denoted $Q_a = 8$.

5.2.6.1 Presentation of the quantized model

Quantizing a model enable the use of fixed-point arithmetic. This arithmetic transformation will improve hardware efficiency. The complete fixed-point model estimated for the score processor presented in this section defines a well-sized quantization scheme. This approximated quantization scheme will provide a trade-off between implementation complexity, detection performances, and throughput. It is an already competitive implementation, that may also benefit from future optimizations. The resulting bit-accurate model is presented in Fig. 5.8.

Let us define η the quantization of $y(n)$, i.e. $Q_y = 2\diamond\eta$ bits. The whole model has been extrapolated from this symbolic value, to ensure a flexible and scalable model. Classical fixed-point arithmetic rules have been applied. As a reminder, let us take u and v two real number quantized on Q_u and Q_v bits respectively. Thus,

1. the result of their addition is quantized on $Q_{u+v} = \max(Q_u, Q_v) + 1$ bits,
2. the result of their multiplication is quantized on $Q_{u\times v} = Q_u + Q_v$ bits.

Consequently, since complex number multiplications essentially are addition of multiplied real numbers, if u and v are complex numbers, $2\diamond Q_{u\times v} = 2\diamond(Q_u+Q_v+1)$. Finally, to reduce the footprint of the model, truncation and saturation have been applied. A truncation consists on removing of superfluous Least Significant Bits (LSBs), while saturation allows removing unnecessary Most Significant Bits (MSBs). Let us walk through the model, block per block.

The first block allows computing d_n^ω as defined in (4.28), and is depicted in Fig. 5.8a.

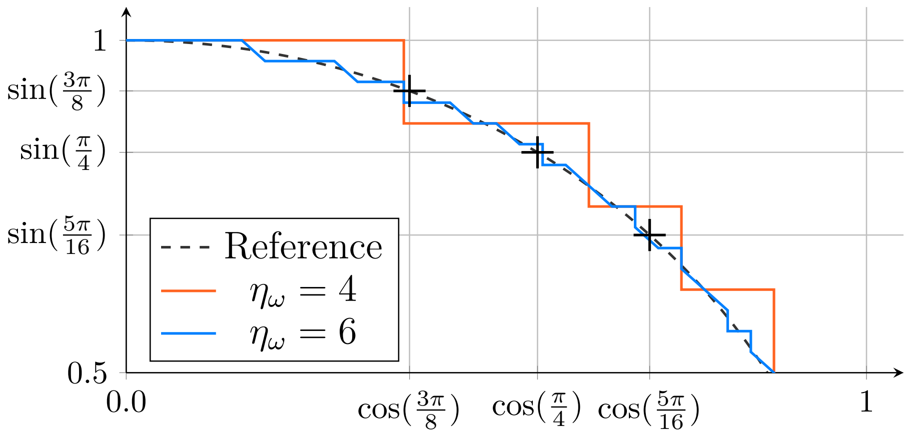
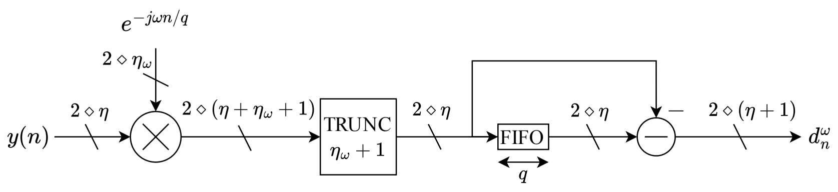
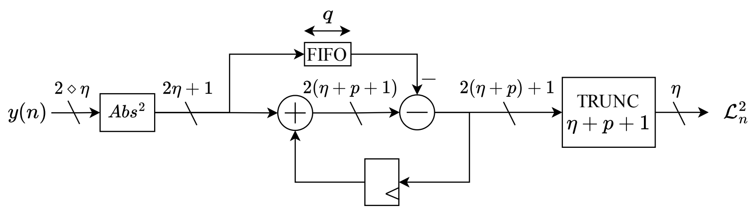


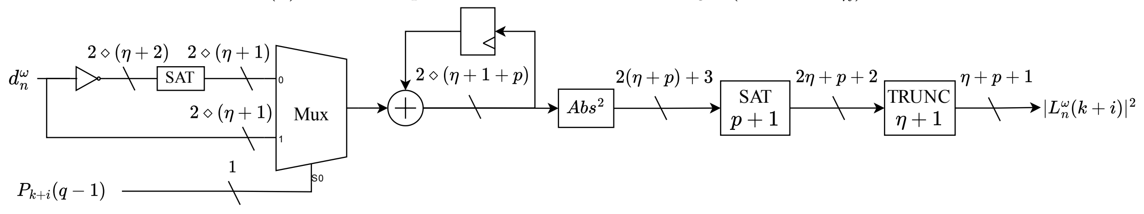
Figure 5.7 – $\mathbf{\Gamma}^\omega = \{e^{j\frac{k\omega}{q}}\}_{k\in[\![0,2qp_\omega]\!]-1}$ for $q = 64$ and $p_\omega = 4$, quantized on $\eta_\omega = 4$ and 6 bits, represented on the north-east quadrant.



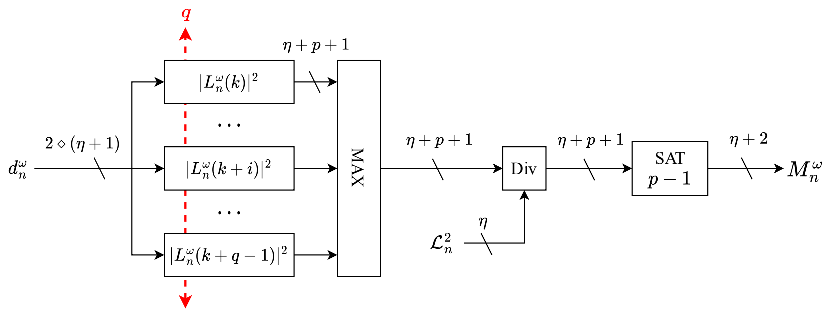
(a) Rotation correction and computation of d_n^ω, with $\eta_\omega = 6$ in practice.



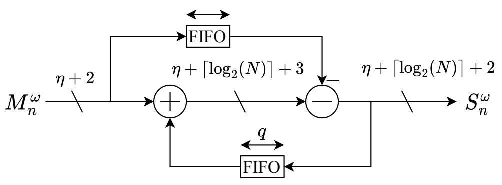
(b) Iterative square $\mathcal{L}_2$-norm estimation of $\boldsymbol{y}_n$ (denoted $\mathcal{L}_n^2$).



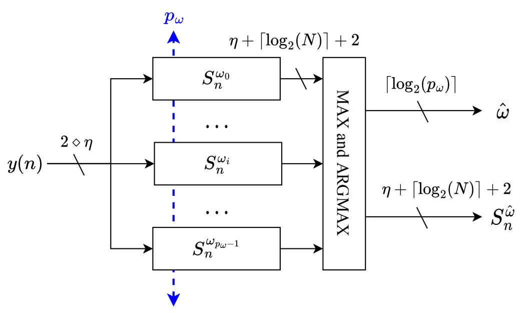
(c) Square modulus of $L_n^\omega(k+i)$.



(d) Normalized maximum of correlation M_n^ω.


FIFO
M_n^ω
$\eta+2$
$\eta+\lceil\log_2(N)\rceil+3$
$-$
$\eta+\lceil\log_2(N)\rceil+2$
S_n^ω
q
FIFO

(e) Accumulation of the score S_n^ω.



(f) Score processor for p_ω rotation hypotheses, producing a rotation hypothesis $\hat{\omega}$ and the corresponding score $S_n^{\hat{\omega}}$.

Figure 5.8 – Fixed-point model of the QCSP detector, broken down in smaller processing blocks, depending on η, the input quantization, $p = \log_2(q)$ and N.

The point of interest here is the multiplication by the rotation correcting factors $e^{j\frac{\omega n}{q}}$, as defined in (3.9). These values are periodic ; a period of the rotation is enough to represent the signal over time. They are represented quantized by η_ω bits, but in practice, η_ω is set to 6 bits, as it proved sufficient to produce a reliable rotation. For instance in Fig. 5.7, it is clear that with $\eta_\omega = 4$ (orange plot), the circle is damaged, whereas with $\eta_\omega = 6$ (blue plot), circular shape is accurate. Since the modulus of $e^{j\frac{\omega n}{q}}$ is one (i.e. $|e^{j\frac{\omega n}{q}}| = 1$), quantizing $y(n) \times e^{j\frac{\omega n}{q}}$ on η bits does not result in change in the signal dynamic. The $\eta_\omega + 1$ LSBs added by the multiplication are thus truncated.

The second block depicted in Fig. 5.8b is the norm estimator. The transitive value of $2(\eta + p) + 1$ allows preventing overflow throughout the accumulation of $q = 2^p$ values of η bits. Then, the truncation of $\eta + p + 1$ bits reduces the footprint while preserving the overall norm. Indeed, at low SNR, the norm is mainly influenced by the noise, so using more than η bits to quantized it is pointless.

The third block is a TCU, depicted in Fig. 5.8c. It uses the *index-shifting* architecture depicted in Fig. 4.5b, for the reasons discussed in section 4.2. It also alleviates (5.2) to replace the multiplication by $P_{k+i}(q-1)$ by a multiplexer. The same reasoning as for the norm applied to justify the $2 \diamond (\eta + p + 1)$ quantization format for the accumulated correlation. The following saturation has been selected to reduce the dynamic range of the maximums without damaging the detection performance. Indeed, the value of the maximum is not of interest in the remaining system. The only requirement is that the saturation value must be high enough that it would not mask a detection. In practice, since the norm is quantized on η bits, and since the score is the accumulation of N normalized value, setting the saturation value to $2^\eta \times U_0$ is more than safe.

The remaining blocks are self-explanatory. The chosen saturation value in the correlator depicted in Fig. 5.8d is safe, since it is performed after normalization, and $\eta + 2$ bits are enough to distinguish signal and noise. The score accumulator represented in Fig. 5.8e follows the same reasoning as the norm accumulator, and the complete score processor is the combination of all previously presented blocks. The rotation hypothesis $\hat{\omega}$ is quantized on $\lceil \log_2(p_\omega) \rceil$ bits, as it is the address to a ROM which contains the actual rotation values.

Now that the quantization format are explicated, the performances achieved by the quantized model are presented.

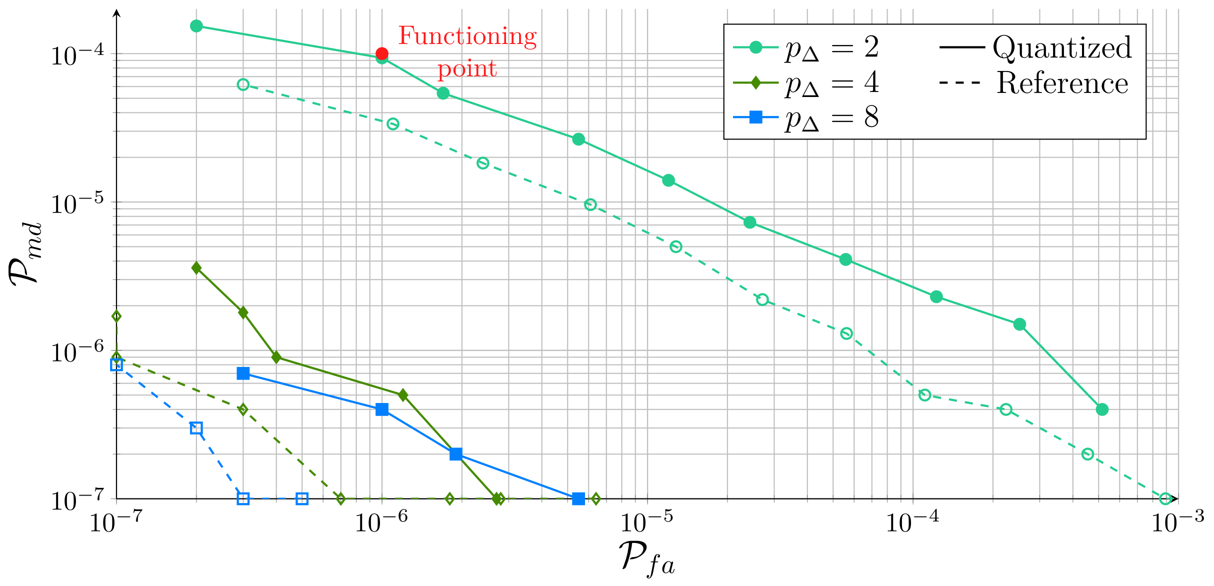


Figure 5.9 – ROC curves for $q = 64$, for various value of p_ω and an SNR of -9 dB.

5.2.6.2 Quantization format effect on detection performances

Data quantization allows more efficient hardware implementations. The drawback is a possible loss in detection performances. To evaluate the impact of our own quantization choices, the model depicted in Fig. 5.8 has been simulated in software. It has been simulated with the same results as used in precedent simulations, e.g. $q = 64$ and $N = 60$. For the simulation, η has been set to 16 bits. While being a large value for a low SNR application, it is the bit wideness natively supported be X310 USRP, and we wanted to spare ourselves the extra work required to adapt the USRP FPGA image.

ROC curves obtained by simulation are plotted in Fig. 5.9. As expected, quantization has impacts performances negatively. Indeed, for $p_\Delta = 4$ (aquamarine curve), the $\mathcal{P}_{md}$ of the reference is at 3×10^{-5}, while the one of quantized version is near $\times 10^{-4}$. However, it still satisfies the detection performance requirements. This is encouraging, since it means that a further studied quantization can help reduce resource utilization even more, while achieving acceptable detection performances.

5.2.6.3 Preliminary implementation results

At the time of writing, the quantized model is being implemented onto FPGA targets, using HLS description in C/C++ for Vivado HLS 2019.1. To implement fixed-point arithmetic with bit-accurate precision, the “arbitrary precision types” provided by Xilinx

[83] has been used. While the implementation is not completed, preliminary results for $q = 64$ and $\eta = 16$ are available for the TS correlator. Indeed, a correlator IP has been completed and verified on chip. The utilization reports as well as the achieved throughput of the IP are reported in Table 5.7, for two different targets:

1. a Kintex 7 XC7410T (speed grade -1) which is the FPGA embedded in the USRP X310,
2. a Kintex 7 XC7325T (speed grade -2) which can be found on Digilent Genesys 2 boards.

The latter has been used to validate the correlator IP, since no standalone XC7410T was accessible. Indeed, the one embed on the X310 cannot be used as standalone circuit. To better compare with values in Table 5.6, the results of one IP have been multiplied by p_ω to approximate the overall system utilization. It assumes that multiple independent correlators can be implemented in parallel to address the $//_\omega$.

Table 5.7 – Hardware performance estimation of the quantized correlators on a Xilinx Kintex 7 XC7410T (speed grade -1) and XC7K325T (speed grade -2), with a working frequency of 100 MHz, for $q = 64$, and several values of p_ω.

Target	p_Δ	p_ω	**Chip Rate**	**Bit Rate**	**Usage**			
			Mc/s	*kb/s*	*FF*	*LUT*	*BRAM*	*DSP*
XC7410T-1	64	4	25	781	32100	45348	12	256
					6.3%	17.8%	1.4%	16.6%
		8	25	781	64200	90696	24	512
					12.6%	35.7%	2.9%	33.3%
XC7325T-2	64	4	25	781	25980	46980	12	256
					6.4%	22.9%	2.7%	30.5%
		8	25	781	51960	93960	24	512
					12.7%	45.8%	5.4%	61.0%

First, it should be noted that the (coarsely) quantized TS correlator, which achieve the same detection performances as the floating-point one, consumes around the same amount of FFs and LUTs as the quantized FFT for $p_\Delta = 8$, for any p_ω. However, it requires even less DSPs and BRAMs than the floating-point TS correlator, while having throughputs $2.8\times$ higher. It is thus the best solution for $q = 64$.

The $64 \times p_\omega$ DSPs used in the overall reported architecture correspond to the "Abs^2" functions. Concerning BRAMs, $2 \times p_\omega$ correspond to the IP input and output buffers, and p_ω BRAMs correspond to the correlation registers. This indicates that a main axis of enhancement could be to reduce the complexity of the "Abs^2" function. An idea would be to use a CORDIC [110] to implement the absolute value. Another obvious way to minimize the hardware complexity would be to reduce the value of η, but it cannot be performed blindly as it may impact detection performances. Since quantization format manipulation is sensible, a thorough study to explore different trade-offs is needed.

5.3 Conclusion

In this chapter, we studied the viability of a QCSP real-time communication chain implementation. Both transmitter and detector have been successfully characterized and implemented on both CPU and FPGA targets.

The current transmitter implementations, whether it is on software or hardware targets, achieved throughputs superior to thoses demanded in LPWAN standard. Both software and hardware implementations reach information throughputs superior to $2Mb/s$. These bit rates are 20× higher than the upper bound of the LPWAN requirements, for costs comparable to established solutions. At the receiver side, the efforts have been focused on detection, especially on the most intensive task of the detector, the correlator. It has been implemented in real-time on both CPU and FPGA circuits. It has been optimized, taking target specific features in account. Software implementations reach up to 100 kb/s of information throughput, and hardware implementations reach throughputs 3× higher, around 300 kb/s. Moreover, it has been demonstrated that a QCSP detector can already be implemented for low costs thanks to the TS architecture, since a low-end Raspberry Pi 4 can suffice to meet LPWAN requirements. In addition, hardware implementations demonstrated the advantages of the contributed TS architecture compared to the legacy FFT one. Finally, to further improve the hardware implementation efficiency, a quantized model is proposed. It enabled to achieve even higher throughputs, reaching up to 780 kb/s on FPGA circuits.

While simulations and synthetic benchmarks allow apprehending achievable performances, nothing can replace full-scale experimentation. The next chapter (the last before the conclusion) is dedicated to the presentation of the full-scale experiments conducted using the work presented in this thesis.

Chapter 6

Real-Time Full-Scale Experiments

Contents

6.1 Experimental setup 105
6.1.1 GNSS-enabled mobile QCSP transmitter 108
6.1.2 Real-time QCSP receiver 111
6.1.3 Experimental environment and protocol 113
6.2 Experimental results analysis 113
6.3 Conclusion 116

The implementation results presented in chapter 5 demonstrate that the implementation of a real-time QCSP chain is feasible. These implementations offer new experimental perspectives.

In this chapter, the first real-time QCSP communication chain ever fully implemented is presented, as well as the experiments it has enabled. In a first time, the experimental setup is detailed. It includes the description of the transmitter implementation and the content of the messages it sends, as well as the description of the complete real-time receiver. It is followed by the presentation of the experimental environment, as well as the experimental protocol. In a second time, the results of two different experiments conducted are reported and analyzed.

6.1 Experimental setup

This section aims to define the experimental setup, from the transmitting side to the receiving side, as well as the various settings in use.

The complete system is represented on Fig. 6.1, with the actual settings used, as well as the implementation used. This figure highlights the high heterogeneity of the communication system, that was required to meet real-time performances while still allowing an easy exploitation of experimental results. It required adapting and interconnecting several components, from different projects, implemented in different languages and/or frameworks. Thus, the full demonstrator has required a non-negligible work. This thankless engineering work, necessary to validate performances, is often time-consuming. We wanted to evaluate the performances of the QCSP chain in the IoT context. We also wanted to explore experimentally the worst performances of the waveform. The payload to be sent must still contain information valuable for further research.

Consequently, let us present the values chosen for the QCSP specific settings and explain the rational of their choice. The first value of interest is q (and p since $q = 2^p$) which is both the size of GF(q) and the size of the PN sequence $\boldsymbol{P}_0$. It is set to $64 = 2^6$, as it is the smallest value planned to be supported. This choice is motivated by the desire of defining a performance floor. Indeed, the more q grows, the more the performances improve, as explained in chapter 3. Incidentally, the lower q is, the lesser the implementation constraints are. Thus, it also limits the risks of unforeseen implementation issues. For the same reasons, the number of GF(64) information symbols K is set to 20, which is the lowest supported value as well. Thus, the transmitted message $\boldsymbol{M}$ is composed of $K \times p = 120$ bits. The NB-LDPC code selected has a rate $R_c = \frac{1}{3}$, and has decoding performance above theoretical detection performances. It ensures that any detected and synchronized frame is decoded, thus focusing on detection and synchronization performances.

The size of the codeword in symbols N is equal to $\frac{K}{R_c}$, the codeword $\boldsymbol{C}$ is thus composed of $N = 60$ symbols of 6 bits, or 360 bits in total. The CCSK mapping (see section 3.2) transform each 6-bit symbol into a circular rotation of $\boldsymbol{P}_0$, resulting in a CCSK frame $\boldsymbol{F}_{CCSK}$ of $N \times q = 3840$ chips, which result in a modulation rate $R_m = \frac{6}{64}$. After application of the Overmodulation (OM) and of the BPSK, the frame $\boldsymbol{F}$ of $N \times q = 3840$ chips is shaped by the root-raised cosine filter of roll-off factor $\beta = 0.35$. The result is provided to a Digital Analog Converter (DAC) tuned to achieve a chip rate of 125 kc/s. Considering the effective rate $R_{eff} = R_c \times R_m = \frac{1}{32}$, the chip rate corresponds to a bit rate of $\frac{125}{32} = 3.9$ kb/s. This value is below the Low Power Wide Area Network (LPWAN) requirements, but issues have been encountered with the B205 mini USRP (the SdR device used in transmission, see section 6.1.1) when higher values were used. The packet is broadcasted over the ISM radio band centered on $f_c = 433.92$ MHz. It is worth noting that since the payload is composed of 120 bits, it takes $\frac{120}{3.9} \simeq 31$ ms to send one frame.

In reception, an oversampling factor $\mathcal{O} = 4$ is used, i.e. 4 samples per chip. It results in a sampling frequency $f_s = \mathcal{O} \times 125 = 500$ kHz. An oversampling of 2 and a frequency

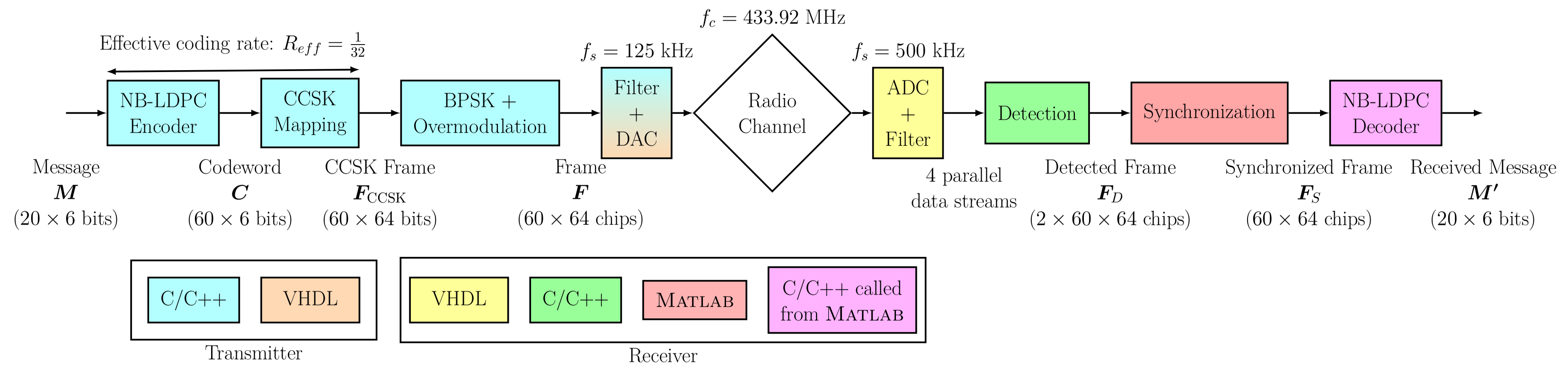

Effective coding rate: $R_{eff} = \frac{1}{32}$
NB-LDPC Encoder
CCSK Mapping
BPSK + Overmodulation
$f_s = 125$ kHz
Filter + DAC
$f_c = 433.92$ MHz
Radio Channel
$f_s = 500$ kHz
ADC + Filter
Detection
Synchronization
NB-LDPC Decoder
Message $\boldsymbol{M}$ (20×6 bits)
Codeword $\boldsymbol{C}$ (60×6 bits)
CCSK Frame $\boldsymbol{F}_{\text{CCSK}}$ (60×64 bits)
Frame $\boldsymbol{F}$ (60×64 chips)
4 parallel data streams
Detected Frame $\boldsymbol{F}_D$ ($2 \times 60 \times 64$ chips)
Synchronized Frame $\boldsymbol{F}_S$ (60×64 chips)
Received Message $\boldsymbol{M'}$ (20×6 bits)
C/C++
VHDL
Transmitter
VHDL
C/C++
MATLAB
C/C++ called from MATLAB
Receiver

Figure 6.1 – Complete QCSP communication chain represented with the implementation used, as well as the communication actual settings.

of 250 kHz would have been enough to satisfy the Shannon criterion, so 4 ensures good performances. Samples are filtered by another root-raised cosine filter of roll-off factor 0.35, then 4 parallel detectors are run on 4 parallel data streams cadenced at 125 kc/s. Since they process each new sample, it results in $p_\Delta = q = 64$. They all use 8 hypotheses of rotation over $[-2\pi,\ 2\pi]$, which is equivalent to $p_\omega = 4$. The detailed explanation has been given in section 3.3.3.2. In summary, frequency errors induced by local oscillator inaccuracies or Doppler effect result in rotation of the arriving frame. The rotation is considered at symbol level for representation reasons. A rotation span of $[-2\pi,\ 2\pi]$ means that each incoming symbol is expected to make at most one full rotation clockwise or counter-clockwise. Since p_ω has been defined over $[-\pi,\ \pi]$, which is half as long as $[-2\pi,\ 2\pi]$, 8 hypotheses over the latter result in 4 over $[-\pi,\ \pi]$. The symbol rotation error is thus bound by $\vartheta = \frac{\pi}{4}$, which corresponds to a normalized residual frequency of $\frac{\vartheta}{2\pi q} = \frac{1}{512}$. Taking into account the chip rate of 125 kc/s, it results in a maximum residual frequency error of $\frac{125\times10^3}{512} = 244$ Hz.

When a detection occurs, only the detector associated with the highest score transmit the detected frame $\boldsymbol{F_D}$ of $2N \times q = 7680$ chips to the synchronizer. This is similar to a decimation by a factor 4 of the signal. A synchronized frame $\boldsymbol{F_S}$ of 3840 chips is passed to the NB-LDPC decoder [99], which correct eventual errors. When the resulting codeword syndrome is composed exclusively of null-element, it means that the message is correct. It is thus printed to the operator and saved. Otherwise, the erroneous codeword is shown and saved. The detailed implementation of the detector, the synchronizer and the decoder are given in section 6.1.2.

Now that all settings have been defined and explicated, let us present the heterogeneous transmitter implementation and the physical devices used in details. The content of the messages sent is also given. It will explain how the payload is generated and why we decided to add those data.

6.1.1 GNSS-enabled mobile QCSP transmitter

Transmitter system description

This section is dedicated to the description of the real-time mobile transmitter system used during the experiments, depicted in Fig. 6.1. The transmitter is required to be mobile, to allow moving it throughout distances. From a system point of view, it is composed of the QCSP transmitter, an SdR device, a Global Navigation Satellite System (GNSS) module, and a battery.

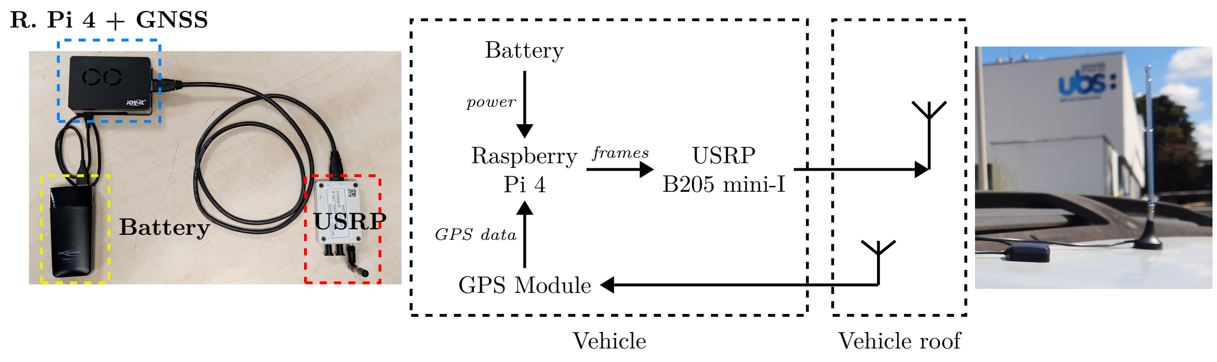


Figure 6.2 – Implemented transmitter component diagram.

The QCSP transmitter is fully implemented in software, including the root-raise cosine filter. We used the version presented in chapter 5, as well as the Raspberry Pi 4 SBC used in benchmarking. This choice is motivated by the higher flexibility of this implementation, where bug fixes and enhancement can be added on-the-fly. Moreover, it allows to access all the sensors already present on the SBC. This platform is also the most energy efficient of the one we have tested. A determinant advantage, since it enables hours long experiments. It must be noted that the NB-LDPC encoder comes from work presented into [68].

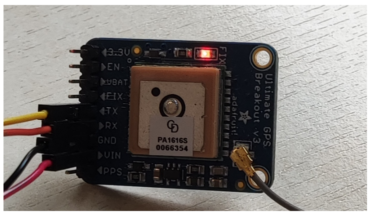


Figure 6.3 – GNSS module embed into the Raspberry Pi chassis.

The SdR device is an USRP B205mini-I. The selling point of the device is that it both communicates with its "host" (the Raspberry Pi) and is powered through USB. Thus, it is easily transportable. The drawback is that the FPGA image can not be tweaked easily, since it required an old software, Xilinx ISE 14.7, and is entirely implemented in pure RTL language. That also explains why the filter has been implemented in software. The transmitting device is hooked up to a GNSS module, depicted in Fig. 6.3. This cheap module allows to fetch the precise location of the mobile transmitter during experiments. In order to track the device in real time from the QCSP receiver, the NMEA 0183 frame are parsed to add the position to the payload sent. The power source of the device is a 4000 mAh battery, that has been measured to last at least 12 hours in function. This helps to embody an off-grid device.

The whole package can be transported in a vehicle, or in a backpack. When transported inside a vehicle, it can be linked to external antennas, to ensure good reception of the GNSS signal and good transmission properties for the QCSP signal.

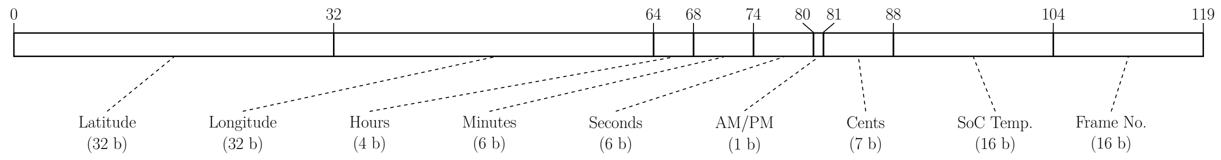


Figure 6.4 – Structure of the payload sent over the radio channel.

It results in a highly flexible, easy-to-transport and to use mobile transmitter, which can be used in various environment.

Payload specification

Data sent in the packet should reflect the reality of the targeted context. More importantly, they have to help in analyzing the results, in order to highlight possible improvement axis. Thus, the packet sent by the transmitter can not be composed of random data.

To fulfill the two precedent objectives, the payload represented in Fig. 6.4 has been elaborated. This payload contains the latitude and longitude of the mobile transmitter, as fetched by the GNSS module. To avoid precision issues, these values are represented using 32-bit floating-point values. The next 24 bits are used to store the time of sent, as reported by the Raspberry Pi clock. We use this reference instead of the GNSS time to be able to easily find the associated logs on the SBC, if needed. The classical way of measuring time, the Unix time in epochs, the number of seconds since the 1st of January 1970, would require 64 bits. To be able to use only 24 bits while sending the cents of seconds too, a 12-hours representation has been used. It allows to store the hours on 5 bits using $\lceil \log_2(12) \rceil = 4$ and a bit to store if we are the morning or the afternoon. Minutes and seconds are stored on $\lceil \log_2(60) \rceil = 6$ bits, and cents on $\lceil \log_2(99) \rceil = 7$ bits. To increase the likelihood of the frame with some already in-use in IoT, we added the SoC temperature, as reported by ARM on-board sensors. Incidentally, it can also be used to monitor the proper functioning of the device from the receiver. The payload is ended by a counter, that allows to keep track of the frames. A 16-bit integer is enough, since it enable unique frame IDs during up to 18 hours of experimentation while sending 1 frame per second. It allows to immediately notice missing frame at the receiver side.

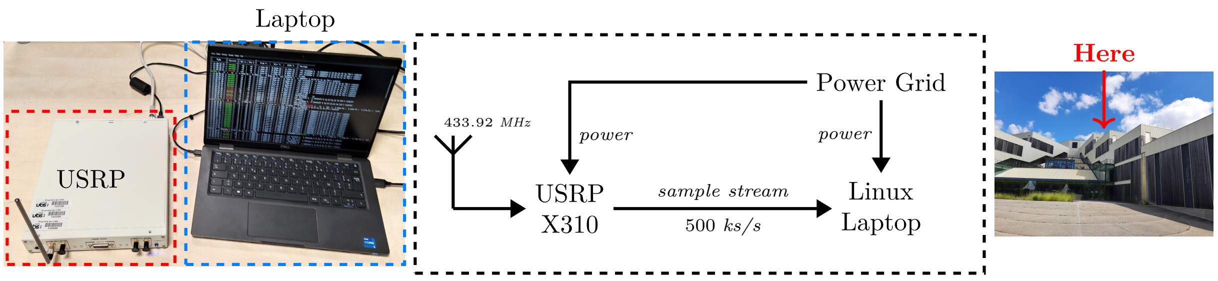


Figure 6.5 – Implemented receiver component diagram.

6.1.2 Real-time QCSP receiver

This section aims to explain the choices and trade-offs made to implement the complete real-time receiver, depicted in Fig. 6.5. Due to its higher complexity compared to the transmitter, it is not mobile and powered using the grid. It is composed of two parts. The first one is the “host”, a Linux laptop that performed all software related tasks. The second one is the “device”, an USRP X310, which is assigned all the RF related tasks, as well as those implemented on hardware, thanks to RFNoC [91].

As already stressed out, this is a highly heterogeneous system. Like represented in Fig. 6.6, the QCSP receiver is implemented using four different languages, on two different platforms used simultaneously, and involves the use of at least two multithreaded processes on the software side. Besides, it uses contributions from other people, who had different research objectives. It thus needed adaptations. Let us quickly walk through each function.

The first step is the sampling of the channel. This is performed by the USRP. Alleviating RFNoC, the root raise cosine filter has been implemented directly on the device. It uses an already tried and tested FIR filter block, provided by Ettus Research. It allows releasing the stress put on the CPU off the host.

The filtered data are streamed to the host using USRP Hardware Driver (UHD) [111], handled directly in the software implementation using the UHD C++ API. The complete real-time detector is implemented using the software TS SIMD correlators described in

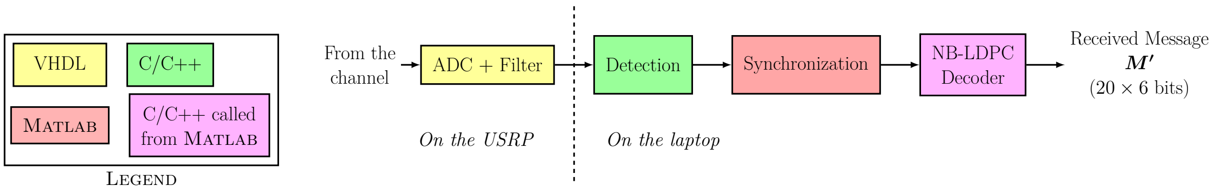


Figure 6.6 – Languages and platforms involved in the receiver.

chapter 5. The software implementation has been chosen due to its higher flexibility and scalability. As for the transmitter, it permitted us to experiment more easily different setup. It also eases the transport of the receiver to reach the experimentation site. Leveraging the single threaded nature of TS correlators, the 4 detectors required to address oversampling can be reliably executed in parallel. We used OpenMP to enable MT. Since the host is a multicore system, it means that each detector have a dedicated core. Moreover, the kernel used in the host has been patched with PREEMPT-RT to enable real-time capabilities [112]. Thus, thread priorities have been tweaked, ensuring the detector will be run in priority.

When a detection occurs, the detector transfers the detected frame $\boldsymbol{F_D}$ to the synchronizer using a software FiFo, allowing both process to communicate and to be synchronized with each other. The synchronizer is the adaptation of the MATLAB/Octave code developed by K. Saied in is work [17]. It has been heavily optimized, in order to reduce the original code execution time. We leveraged MATLAB/Octave underlying Basic Linear Algebra Subroutines (BLAS) [113], which perform better on large matrix calculations than on incremental computations. After analysis of the algorithm and of the code, it has been reformatted to let such matrix calculations appears. The original proof-of-concept description was able to process one frame every 15 seconds. The results of our efforts is able to process one frame every 0.7 seconds. In other words, since $\boldsymbol{F_D}$ contains $2 \times N \times q = 7680$ chips, we improved the input throughput from 512 c/s, to 10100 c/s, which is 21× higher.

The NB-LDPC decoder is based on a software implementation of C. Marchand [114]. To reduce inter-process communications, it has been implemented as a MATLAB executable "*MEX*" file by J. Jabour, and as an Octave executable "*octfile*" by myself, to be called within the same process as the synchronizer.

Finally, a terminal-based Human Machine Interface has been added, reporting in real-time the decoding results. It allows the operator to monitor the time of arrival of the frames on the host. It also informs about symbol and chip errors, as well as measured frequency inaccuracies.

In the end, the development work performed results in a GNSS-enabled mobile transmitter and a real-time receiver, able to carry out full-scale experiments for a wide range of settings. The main experimental protocol elaborated to validate the waveform and the implemented system in real conditions is presented in the next section.

6.1.3 Experimental environment and protocol

To study the viability of the QCSP waveform, it is necessary to try it out in uncontrolled environments. It has been decided to conduct an experiment in real conditions, with a mobile transmitter broadcasting messages and a static receiver listening. We choose to experiment in the urban area around the laboratory, to find out if indirect transmissions can be successful. Indeed, simulations has been performed on the CAWGN channel, while the real radio channel in urban area looks more like a Rician channel, where multipath interferences are possible.

It has been decided to take advantage of the high mobility of the transmitter system. It has been equipped with external antennas, and is disposed in a vehicle. This way it can be transported, allowing to test different locations and also to add Doppler effect. The real-time receiver is placed in-height, typically on the edge of a building roof. It allows to ensure the predominance of the direct path. The transmitter is then set to send a frame every 2 seconds, to be sure that the synchronizer can keep up. The frame transmitted are locally saved by the transmitter, to allow to track it post-experiment. For redundancy reasons, the path taken by the mobile is also recorded by an independent GNSS device. A telephonic communication is maintained between the operator monitoring the transmitter in the vehicle and the one monitoring the receiver. This way, the experiment can be adapted in real-time, to avoid losing time. After the experiment, the number of frame missed is estimated, and the data associated to erroneously synchronized frames are inspected. In the next section, the results associated to two of the conducted experiments are presented.

6.2 Experimental results analysis

The two different experiments presented in this section have respected the experimental protocol, and used the previously described transmitter and receiver.

They are distinguished by one setting: the quality of the 433.92 MHz ISM radio band antennas. Indeed, the antennas used during the second experiment was marketed as adapted for the targeted frequency, but revealed themselves to be flawed. Incidentally, it allows to define the worst case scenario.

A first interesting data is the maximum range reached in each experiment. Indeed, as represented on the map realized thanks to [115] in Fig. 6.7, we successfully detected, synchronized and decoded frames sent up to 500 m away from the receiver in the second experiment. It is remarkable, since the antennas used were not adapted. Another impressing point is that we have successfully sent and received frames within a 1 km radius in an urban area. Second, some frames have been received and successfully decoded even in the absence of a direct path, which consolidates the hypothesis of robustness of the QCSP waveform.

Finally, let us look at the absolute value of the channel associated to some successfully decoded frames, presented in Fig. 6.8. The red mark on the figures highlights the location of the frame in the channel. When the SNR is high like in Fig. 6.8a, i.e. when conditions are optimum, the frame can be seen easily, even by a naked eye. The strength of the QCSP system is its ability to transmit frames even when the frame is drowned in the noise, like in Fig. 6.8b. Even more impressing, it seems to be robust to interferences, as frames in Fig. 6.8c and Fig. 6.8d were successfully detected, synchronized, and decoded. These kinds of interferences are expected on the ISM radio band, since the frequency band is already used by other IoT protocols like LoRa and SigFox. The algorithm has not been developed to be robust to interferences, but thanks to its resilience to low SNRs and also thanks to the new normalization method, sudden variation in SNR does not block frame detection. These observations would not have been possible without the real-time communication chain developed in the current work.

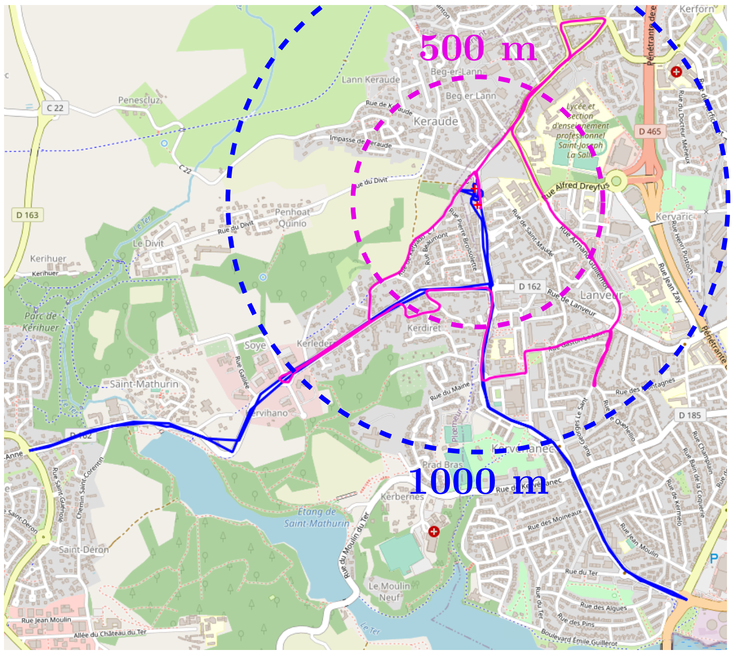
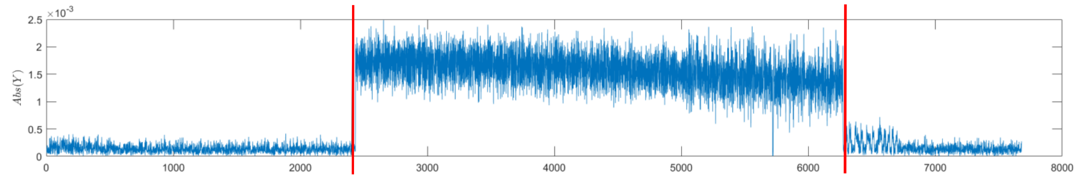
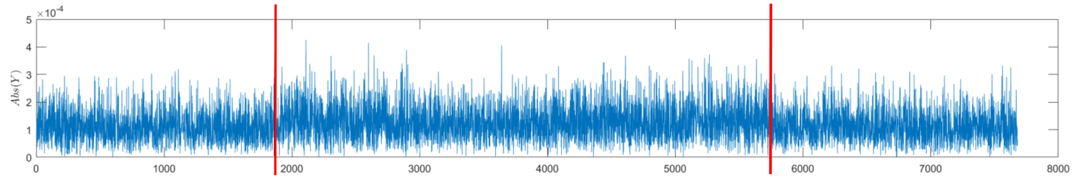


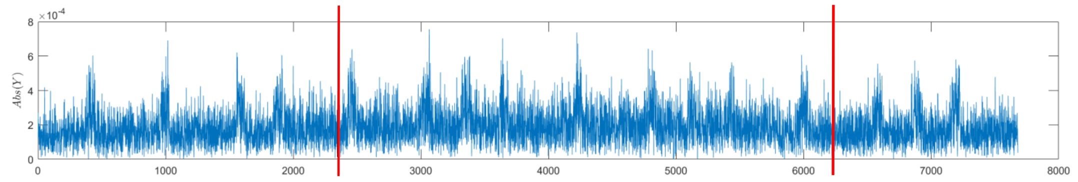
Figure 6.7 – Maximum ranges reached for reliable QCSP transmissions in the first experiment (in blue) and in the second experiment (in magenta), along the tracks followed in each experiment.



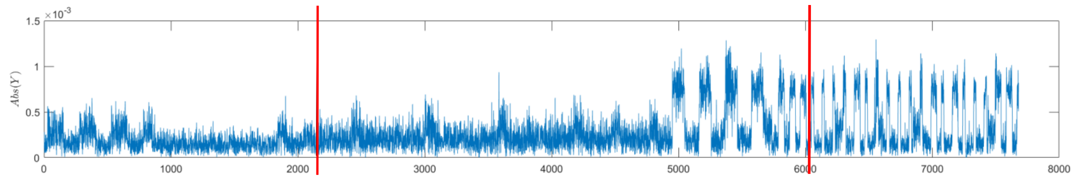
(a) High SNR.



(b) low SNR.



(c) Weak interferences.



(d) Strong interferences.

Figure 6.8 – Frames successfully detected, synchronized and decoded for various bad channel conditions. Red marking highlights the position of the frame.

6.3 Conclusion

In the current chapter, both a complete standalone real-time mobile transmitter and a fully functional real-time receiver have been described. These systems were challenging to design and to implement. However, they will be decisive in the future. They allow conducting full scale experiments, which have already served to validate the QCSP waveform. These experiments also proved the viability and flexibility of the implementations presented in previous chapters.

Results in terms of range are very encouraging, and suggests that the current implementations could already be used in the IoT context, for transmitting tracking data for instance.

Chapter 7

Conclusion

7.1 Synthesis

The ever-growing number of interconnected devices, coupled with the need of energetic efficiency induced by recent shortages, require innovations in the field of digital communications. This thesis presented a solution to improve the efficiency of transmitting devices in the context of Low Power Wide Area Networks (LPWANs). In addition, it also demonstrated the relevance and efficiency of the Quasi-Cyclic Short Packet (QCSP) communication chain proposed implementation. To this end, several computational and architectural optimizations have been proposed, supported by benchmarks and real-time implementation results on both hardware and software targets.

The involved issues and the context are stated in chapter 2. The limitation induced by context-specific constraints, the critical issue of short packets and limited amount of bandwidth resources are also explicated. An overview of existing methods trying to answer the same question as the current work, like LoRa [4] and SigFox [53], is given. In particular, the challenge that their implementation represent is stressed out. It is followed by the introduction to Software-defined Radio (SdR) platforms, and the breakthrough they represent for the design and implementation of Digital Communication Systems (DCSs). This chapter also includes an extensive presentation of the different parallelism levels that can be alleviated in digital signal processing algorithms, and how to leverage them on multicore and manycore systems. The chapter is concluded by a report on the state of High-Level Synthesis (HLS) and associated tools, that highlight how this workflow, combined with the previous points, allow the design of efficient and flexible implementations.

In the third chapter, the QCSP communication chain is thoroughly presented. To allow a complete comprehension of the challenges involved in its implementation, the necessary notions are defined and explicated. This includes the Galois Fields (GFs) theory, as well as the functioning of Non-Binary Low Density Parity Check (NB-LDPC) codes, which are a core concept of the QCSP system. The second most important concept of the system, the Cyclic Code Shift Keying (CCSK) is also presented. Then, the communication chain is explained in details, from the transmitting side to the receiving side. In the process, the exact transmitter algorithm is summarized, as well as the channel model in use. Then, the detection algorithm is introduced. It mainly consists on the comparison of the output of the score function to a threshold U_0. When the score exceeds the threshold, a frame is supposed detected. In order to be resilient to frequency inaccuracies and Doppler effect, and also to reduce the overall complexity, the time-frequency space is divided in a search grid of $p_\Delta \times p_\omega$ bins. The impact on detection performances of the values of p_Δ and p_ω is thus explicated. For completeness, the synchronization process is briefly summarized.

The fourth chapter introduces a method that makes the detection algorithm robust against input scaling factors, which may appear due to gain instability. The 2-norm is selected after a comparative study between several candidate normalization method, due to its good efficiency/complexity rate. Then, the new Time Sliding (TS) windows correlation method is proposed. This method permit computing CCSK correlations in the time domain for a computational complexity lower than the legacy FFT based method for $p_\Delta = q$. The new contributed approach has been demonstrated to have a superior efficiency when aiming for the best achievable detection performances. A software implementation has been used to benchmark both approach, reinforcing the conclusions.

The fifth chapter is the heart of the thesis. It begins by proving the viability of QCSP waveform for the LPWANs by proposing optimizations to the transmitting side, made for the purpose of energy efficiency and throughput. Moreover, both software and hardware implementations are proposed, both exceeding the throughput required in an LPWAN by at least an order of magnitude. Then, the QCSP detection is addressed. To concentrate the efforts, the most critical task of the detection, the correlator, is specified. Then, parallelism levels inherent to the correlator are identified, for each architectural variations. Based on the identification results, the complexity of each correlator approach is reduced, taking advantage of the specific structure of $\boldsymbol{P}_0$, and of the features offer by multicore CPUs and FPGA circuits. Both software and hardware real-time efficient implementations are then realized, leveraging the previous studies. Both implementations met the requirements of LPWAN, and even competed with longtime established solution like LoRa. More impressing the TS hardware real-time implementation proved to be very efficient, while using a disadvantageous quantization scheme. To further improve its performances, a quantized model of the score processor is introduced, and explicated. It has been proved to have a negligible impact on detection performances, while doubling the throughput compared to the original hardware implementation. It even reached 200× the highest required throughput in LPWANs, utilizing less than half of the FPGA circuit that was targeted. Considering the results, a lot of room is available to improve the model.

The penultimate chapter proposed a complete QCSP demonstrator, taking advantage of all the efforts made during the current work, but also leveraging the results and products developed in other related research works. This system required a titanic amount of often thankless engineering work, due to the constraints. Indeed, the demonstrator is, at the same time, a convenient research object, a real-time highly heterogeneous communication chain, and a flexible test bench enabling full-scale experiments. The demonstrator has been already used to validate the behavior of the waveform under real world conditions, and established the encouraging lowest range of 1 km in an urban area, when transmitting over the ISM radio band (433.92 MHz) with adapted antennas.

7.2 Perspectives

The current work is considered a success, as it has fulfilled its original goal. However, many research opportunities have been encountered during the studies. A non-exhaustive list of perspectives is thus provided hereafter.

A complete hardware description of the QCSP detector implemented following the quantized model is at last stages of development. When finished and validated, it will be embedded onto the X310 USRP FPGA circuit. It should enable even more accurate real-time experiments, and allow achieving very high throughput. The next efforts will be directed on the efficient implementation of the synchronization process, following the same process successfully used for the detector. Due to algorithm similarities, the synchronizer will undoubtedly benefit from the improvement made to the real-time detector. A complete overview of the implementation planning is presented in Fig. 7.1. Step 1 and 2 are completed. Step 3 and 4, as said, are in course to be validated. The remaining steps will be worked on in the near future. In the same vein, an updated and stabilized version of the simulation and demonstration software framework can be finalized and released under a free and open source license, to support future research on the topic.

It is worth noting that the systems implemented along this work are considered for use in high seas buoys. These buoys have to transmit information about marine occupation in harsh environments, especially due to the sea swell. Thus, sending small packets can lower the risk of losing data, thanks to the lower transmission latency.

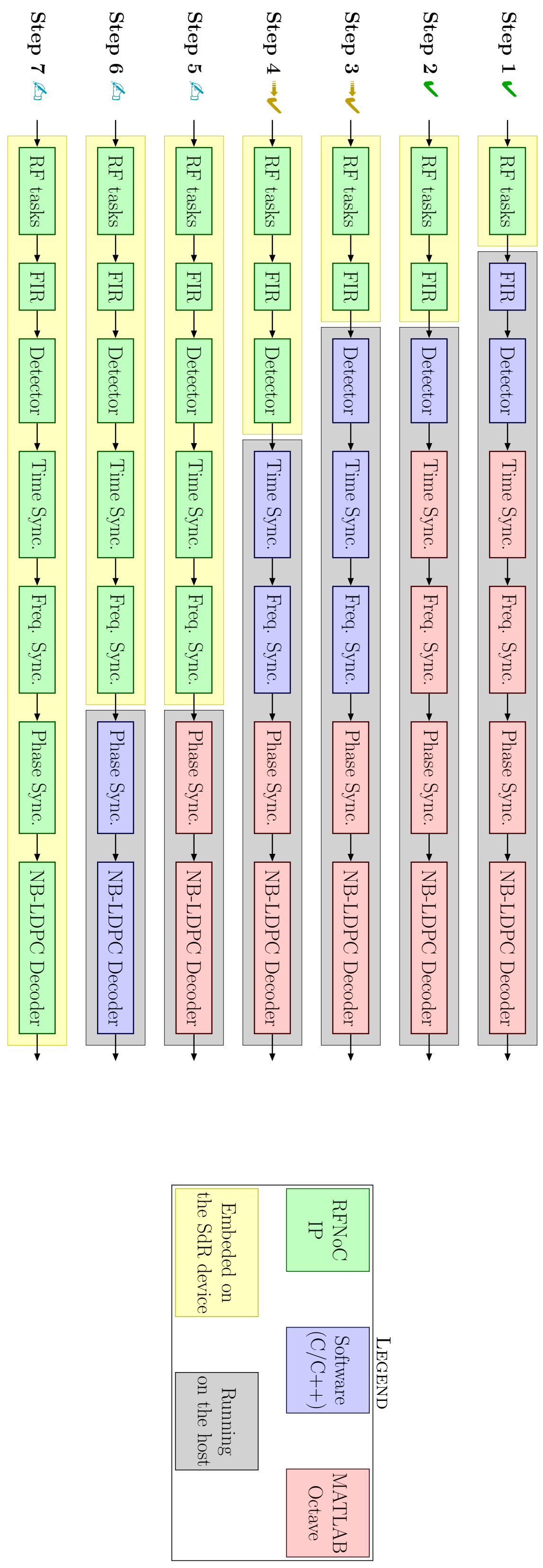


Figure 7.1 – Prospected implementation planning

In a more distant future, the issues related to multi-user systems should be addressed. Some works are already performed on the subject by L. Montaya Obesso. The implementation of a multi-users capable QCSP receiving system will be challenging. But if the current work has demonstrated one thing, is that such implementation is possible.

Besides, the waveform is also considered for spatial use cases. Indeed, since the communication chain is resilient to Doppler effect by design, and is energy efficient, it may be well suited for use in satellite systems. Nevertheless, the spatial context will require thorough implementation studies, as space-grade devices are even more constrained than LPWAN ones. It may be possible to draw inspiration from work realized in this topic for the LoRa waveform [6].

In this prospect, the energy efficiency have to be enhanced, as well as resource utilization. This can be achieved by mutualizing operators, by redesigning the receiving system no longer as a purely dataflow system, but as a modular system, composed of hardware accelerators accessible to a bus attached CPU. This would be particularly well suited to the system-on-programmable-chips used in the spatial context.

For the long foreseeable future, it could be interesting to study the QCSP waveform in a mesh network context. This network topology allows simplifying transmitters, since it reduces direct range requirements. Recent works in this direction are reported in the literature [116].

Appendices

A Algebraic definitions

Algebra is a mathematic broad area that allows to describe and define interactions inside and between sets of numbers (like $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ or $\mathbb{C}$) or even more abstracted sets (e.g. the set of second-degree polynomial functions or the set of bijective applications in $\mathbb{R}$). It uses symbols instead of numbers, while operations are defined as applications on one or more symbols that returns one or more symbols. An Algebraic structure consists on a non-empty set and a collection of operations with a finite number of operands, which verify a number of properties. Each structure have its own set of properties, but some structures can be defined as superset of others. It is especially the case for a Galois field that is obviously a field, the latter being itself composed of groups.

This appendix aims to detail some mathematic definitions use inside this thesis.
First, the notions of associativity, commutativity and distributivity are defined. Then, the algebraic structures *group*, *field*, and *Galois field* are defined.

Associativity

An operation "$\Box$" is called *associative* on a set $\mathcal{A}$ if it verifies

$$\forall (a,b,c) \in \mathcal{A}^3,\ a \Box (b \Box c) = (a \Box b) \Box c. \tag{7.1}$$

Associativity denotes that for multiple operation "$\Box$", the execution order does not affect the result.

Commutativity

An operation "$\Box$" is called *commutative* on a set $\mathcal{A}$ if it verifies

$$\forall (a,b) \in \mathcal{A}^2,\ a \Box b = b \Box a. \tag{7.2}$$

Commutativity denotes that for in a unique operation “$\Box$”, operands order does not affect the result.

Distributivity

An operation “$\diamond$” is called *distributive* over another operation “$\Box$” on a set $\mathcal{A}$ if it verifies

$$\forall (a,b,c) \in \mathcal{A}^3,\ a \diamond (b \Box c) = a \Box b \diamond a \Box c. \tag{7.3}$$

Distributivity denotes a priority of execution of one operation over another.

Group

Let $\mathcal{A}$ be a set of elements, and “$\Box$” a binary operation on $\mathcal{A}$ which takes two elements of $\mathcal{A}$ and results in one element of $\mathcal{A}$ (i.e. $\forall (a,b) \in \mathcal{A}^2,\ \exists!\ c \in \mathcal{A}\ /\ a \Box b = c$). Then, the couple formed by the set $\mathcal{A}$ and the operation “$\Box$” is a group if and only if:

1. the operation “$\Box$” is associative,
2. an identity element I exists in $\mathcal{A}$, i.e. $\forall a \in \mathcal{A},\ I \Box a = a$,
3. any element of $\mathcal{A}$ has an inverse in $\mathcal{A}$, i.e. $\forall a \in \mathcal{A},\ \exists\ a' \in \mathcal{A}\ /\ a' \Box a = I$.

From these requirements follow the two properties that I is unique, and that for all a in $\mathcal{A}$, its respective inverse a' is also unique. A group can optionally be commutative if “$\Box$” is commutative. It this then called an Abelian group or a commutative group.

Field

Let $\mathcal{A}$ be a set of elements defined with a binary operation ”$+$” called the addition, and another distinct binary operation ”$\times$” called the multiplication. The tuple formed by $\mathcal{A}$ and the two operations is a field if:

1. $(\mathcal{A}, +)$ is an Abelian group (The identity element for the addition is called the zero element and is denoted $0_{\mathcal{A}}$),
2. $(\mathcal{A}\backslash\{0_{\mathcal{A}}\}, \times)$ is an Abelian group. The identity element for the multiplication is called the unit element and is denoted $1_{\mathcal{A}}$,
3. the multiplication is distributive over the addition.

Table 7.1 – Examples of primitive polynomials $\mathbb{P}_p$ for different values of p and the corresponding order $q = 2^p$

Polynomial degree p	Galois field order q	Primitive polynomial $\mathbb{P}_p$
1	2	$x+1$
2	4	x^2+x+1
3	8	x^3+x+1
4	16	x^4+x+1
6	64	x^6+x+1
8	256	$x^8+x^4+x^3+x^2+1$

Galois Field

Before defining what a Galois Field (GF) is, the remembering of some algebra notions is needed. The exact definition of each structure is given in appendix A.

A Galois Field denoted GF, also called finite field, is a field which contains a finite number of elements, the number of elements being called the order of the field. A Galois Field of order q is denoted GF(q). The q elements are denoted $\{0, \alpha^0 = 1, \alpha^1 = \alpha, \alpha^2, \dots, \alpha^{q-2}\}$, where α is called the field primary symbol. All non-zero elements are power of the primary symbol. The only orders considered in this thesis are the power of 2, i.e. when it exists p in $\mathbb{N}$, such that $q = 2^p$. All elements of GF$(q = 2^p)$ can be represented in a polynomial format[117]. In this format, each non-zero element can be seen as an element of GF$_2[\alpha]/\mathbb{P}_p$, the set of polynomials with coefficients in $\{0,\ 1\}$ modulo $\mathbb{P}_p$, with $\mathbb{P}_p$ a degree-p irreducible polynomial called the primitive polynomial. Examples of primitive polynomials for a degree $p \in \{1, 2, 3, 4, 6, 8\}$ are presented in Table 7.1. Since any element of GF(q) can be represented as a polynomial with binary coefficients, and since a polynomial can be represented as a vector of its coefficients, a binary representation of GF(q) elements can be easily derived. An example of such representation is given in Table 7.2, for $p = 3$ (thus $q = 8$), and a primitive polynomial $\mathbb{P}_3 = \alpha^3 + \alpha + 1$.

From these considerations, the addition in GF(q) can be reduced to a binary XOR between binary representations, while the multiplication can be performed using the polynomial representation, using the result modulo-$\mathbb{P}_p$ of the multiplication of the polynomials. This can be efficiently implemented using arrays or lookup tables, to leverage the finite size of GF(q) while avoiding the burden of the operation at runtime. It may also be implemented as an addition modulo-q on the power of the symbols.

Table 7.2 – Binary and polynomial representations of GF(8) elements for $\mathbb{P}_3 = \alpha^3 + \alpha + 1$.

Element	Polynomial representation	Binary representation
0	0	000
α^0	1	100
α^1	α	010
α^2	α^2	001
α^3	$1 + \alpha$	110
α^4	$\alpha + \alpha^2$	011
α^5	$1 + \alpha + \alpha^2$	111
α^6	$1 + \alpha^2$	101

Bibliography

[1] « ISM radio band », *Wikipedia*, Oct. 2022.

[2] « NMEA 0183 », *Wikipedia*, Oct. 2022.

[3] *Quasi Cyclic Small Packet – Oct 2019 – Oct 2023.*

[4] C.-S. Choi, J.-D. Jeong, I.-W. Lee, and W.-K. Park, « LoRa based renewable energy monitoring system with open IoT platform », *in 2018 International Conference on Electronics, Information, and Communication (ICEIC)*, Honolulu, HI, USA: IEEE, Jan. 2018, pp. 1–2. DOI: 10.23919/ELINFOCOM.2018.8330550.

[5] L. Kong, M. K. Khan, F. Wu, G. Chen, and P. Zeng, « Millimeter-Wave Wireless Communications for IoT-Cloud Supported Autonomous Vehicles: Overview, Design, and Challenges », *IEEE Communications Magazine*, vol. 55, *1*, pp. 62–68, Jan. 2017, ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600422CM.

[6] M. A. Ben Temim, G. Ferré, and R. Tajan, « A New LoRa-like Transceiver Suited for LEO Satellite Communications », *Sensors*, vol. 22, *5*, p. 1830, Feb. 2022, ISSN: 1424-8220. DOI: 10.3390/s22051830.

[7] A. Koren and D. Šimunić, « Modelling an energy-efficient ZigBee (IEEE 802.15.4) body area network in IoT-based smart homes », *in 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2018, pp. 0356–0360. DOI: 10.23919/MIPRO.2018.8400068.

[8] R. Gallager, « Low-density parity-check codes », *IEEE Transactions on Information Theory*, vol. 8, *1*, pp. 21–28, Jan. 1962, ISSN: 0018-9448. DOI: 10.1109/TIT.1962.1057683.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, « Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1 », *in Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, Geneva, Switzerland: IEEE, 1993, pp. 1064–1070, ISBN: 978-0-7803-0950-0. DOI: 10.1109/ICC.1993.397441.

[10] E. Arikan, « Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels », *IEEE Transactions on Information Theory*, vol. 55, *7*, pp. 3051–3073, Jul. 2009, ISSN: 0018-9448, 1557-9654. DOI: 10.1109/TIT.2009.2021379.

[11] S. Beyme and C. Leung, « Efficient computation of DFT of Zadoff-Chu sequences », *Electronics Letters*, vol. 45, *9*, p. 461, 2009, ISSN: 00135194. DOI: 10.1049/el.2009.3330.

[12] C. Deng *et al.*, « IEEE 802.11be Wi-Fi 7: New Challenges and Opportunities », *IEEE Communications Surveys & Tutorials*, vol. 22, *4*, pp. 2136–2166, 2020, ISSN: 1553-877X. DOI: 10.1109/COMST.2020.3012715.

[13] H. Rahbari and M. Krunz, « Exploiting Frame Preamble Waveforms to Support New Physical-Layer Functions in OFDM-Based 802.11 Systems », *IEEE Transactions on Wireless Communications*, vol. 16, *6*, pp. 3775–3786, Jun. 2017, ISSN: 1536-1276. DOI: 10.1109/TWC.2017.2688405.

[14] B. Martinez, F. Adelantado, A. Bartoli, and X. Vilajosana, « Exploring the Performance Boundaries of NB-IoT », *IEEE Internet of Things Journal*, vol. 6, *3*, pp. 5702–5712, Jun. 2019, ISSN: 2327-4662. DOI: 10.1109/JIOT.2019.2904799.

[15] Y. Polyanskiy, « Asynchronous Communication: Exact Synchronization, Universality, and Dispersion », *IEEE Transactions on Information Theory*, vol. 59, *3*, pp. 1256–1270, Mar. 2013, ISSN: 0018-9448, 1557-9654. DOI: 10.1109/TIT.2012.2230682.

[16] C. Monière, K. Saied, B. Le Gal, and E. Boutillon, « Time sliding window for the detection of CCSK frames », *in IEEE Workshop on Signal Processing Systems (SiPS)*, Combria, Portugal: IEEE, Oct. 2021, pp. 99–104, ISBN: 978-1-66540-144-9. DOI: 10.1109/SiPS52927.2021.00026.

[17] K. Saied, « Quasi-Cyclic Short Packet (QCSP) Transmission for IoT », Theses, Université Bretagne Sud, Mar. 2022.

[18] Open Service Signal B2b, *BeiDou Navigation Satellite System Signal In Space Interface Control Document*, Jul. 2020.

[19] G. J. Holzmann and B. Pehrson, *The Early History of Data Networks.* IEEE Computer Society Press, 1995.

[20] M. Shafi *et al.*, « 5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice », *IEEE Journal on Selected Areas in Communications*, vol. 35, *6*, pp. 1201–1221, Jun. 2017, ISSN: 1558-0008. DOI: 10.1109/JSAC.2017.2692307.

[21] M. R. Palattella *et al.*, « Internet of Things in the 5G Era: Enablers, Architecture, and Business Models », *IEEE Journal on Selected Areas in Communications*, vol. 34, *3*, pp. 510–527, Mar. 2016, ISSN: 1558-0008. DOI: 10.1109/JSAC.2016.2525418.

[22] Jim Skea (United Kingdom), Priyadarshi R Shukla (India), Andy Reisinger (New Zealand), Raphael *et al.*, « Climate Change 2022 », Intergovernemantal Panel on Climate Change, Tech. Rep. 6th.

[23] *Cellular IoT Evolution & digitization/Whitepaper*, https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-iot-evolution-for-industry-digitalization, Jan. 2019.

[24] G. Durisi, T. Koch, and P. Popovski, « Toward Massive, Ultrareliable, and Low-Latency Wireless Communication With Short Packets », *Proceedings of the IEEE*, vol. 104, *9*, pp. 1711–1726, Sep. 2016, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2016.2537298.

[25] S. K. Sharma and X. Wang, « Toward Massive Machine Type Communications in Ultra-Dense Cellular IoT Networks: Current Issues and Machine Learning-Assisted Solutions », *IEEE Communications Surveys & Tutorials*, vol. 22, *1*, pp. 426–471, 2020, ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2916177.

[26] R. S. Sinha, Y. Wei, and S.-H. Hwang, « A survey on LPWA technology: LoRa and NB-IoT », *ICT Express*, vol. 3, *1*, pp. 14–21, Mar. 2017, ISSN: 24059595. DOI: 10.1016/j.icte.2017.03.004.

[27] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu, « Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards », *Computer Communications*, vol. 30, *7*, pp. 1655–1695, May 2007, ISSN: 01403664. DOI: 10.1016/j.comcom.2006.12.020.

[28] S. K. Dhurandher, S. Misra, M. S. Obaidat, and N. Gupta, « QDV: A Quality-of-Security-Based Distance Vector Routing Protocol for Wireless Sensor Networks Using Ant Colony Optimization », *in 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Oct. 2008, pp. 598–602. DOI: 10.1109/WiMob.2008.61.

[29] « IEEE Standard for Low-Rate Wireless Networks », IEEE, Tech. Rep. DOI: 10.1109/IEEESTD.2020.9144691.

[30] C. Bockelmann *et al.*, « Towards Massive Connectivity Support for Scalable mMTC Communications in 5G Networks », *IEEE Access*, vol. 6, pp. 28 969–28 992, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2837382.

[31] C. Goursaud and J. M. Gorce, « Dedicated networks for IoT: PHY / MAC state of the art and challenges », *EAI Endorsed Transactions on Internet of Things*, vol. 1, *1*, p. 150 597, Oct. 2015, ISSN: 2414-1399. DOI: 10.4108/eai.26-10-2015.150597.

[32] N. Abramson, « THE ALOHA SYSTEM: Another alternative for computer communications », *in Proceedings of the November 17-19, 1970, Fall Joint Computer Conference on - AFIPS '70 (Fall)*, Houston, Texas: ACM Press, 1970, p. 281. DOI: 10.1145/1478462.1478502.

[33] J. Day and H. Zimmermann, « The OSI reference model », *Proceedings of the IEEE*, vol. 71, *12*, pp. 1334–1340, Dec. 1983, ISSN: 1558-2256. DOI: 10.1109/PROC.1983.12775.

[34] D. Chu, « Polyphase codes with good periodic correlation properties (Corresp.) », *IEEE Transactions on Information Theory*, vol. 18, *4*, pp. 531–532, Jul. 1972, ISSN: 1557-9654. DOI: 10.1109/TIT.1972.1054840.

[35] T. Fujita, D. Uchida, Y. Fujino, O. Kagami, and K. Watanabe, « A burst modulation/demodulation method for short-packet wireless communication systems », *in Asia-Pacific Conference On Communications (APCC'2008)*, Akihabara, Japan, Oct. 2008, pp. 1–5.

[36] R. Imad and S. Houcke, « Blind frame synchronization and phase offset estimation for coded systems », *in 2008 IEEE 9th Workshop on Signal Processing Advances in Wireless Communications*, Jul. 2008, pp. 11–15. DOI: 10.1109/SPAWC.2008.4641560.

[37] A. Azari, P. Popovski, G. Miao, and C. Stefanovic, « Grant-Free Radio Access for Short-Packet Communications over 5G Networks », *in IEEE Global Communications Conference (GLOBECOM'2017)*, Singapore: IEEE, Dec. 2017, pp. 1–7, ISBN: 978-1-5090-5019-2. DOI: 10.1109/GLOCOM.2017.8255054.

[38] B. Bloessl and F. Dressler, « mSync: Physical Layer Frame Synchronization without Preamble Symbols », *IEEE Transactions on Mobile Computing*, vol. 17, *10*, pp. 2321–2333, Oct. 2018, ISSN: 1536-1233, 1558-0660, 2161-9875. DOI: 10.1109/TMC.2018.2808968.

[39] K.-H. Ngo, A. Decurninge, M. Guillaud, and S. Yang, « Cube-Split: A Structured Grassmannian Constellation for Non-Coherent SIMO Communications », *arXiv:1905.08745 [cs, math]*, Jun. 2020. arXiv: 1905.08745 [cs, math].

[40] P. Walk, P. Jung, B. Hassibi, and H. Jafarkhani, « MOCZ for Blind Short-Packet Communication: Practical Aspects », *IEEE Transactions on Wireless Communications*, vol. 19, *10*, pp. 6675–6692, Oct. 2020, ISSN: 1536-1276, 1558-2248. DOI: 10.1109/TWC.2020.3004588.

[41] J. Mitola, « Software radio architecture evolution: Foundations, technology tradeoffs, and architecture implications », *IEICE Transactions on Communications*, vol. 83, pp. 1165–1173, 2000.

[42] J. Mitola, « The software radio architecture », *IEEE Communications Magazine*, vol. 33, *5*, pp. 26–38, May 1995, ISSN: 1558-1896. DOI: 10.1109/35.393001.

[43] A. M. Wyglinski, D. P. Orofino, M. N. Ettus, and T. W. Rondeau, « Revolutionizing software defined radio: Case studies in hardware, software, and education », *IEEE Communications Magazine*, vol. 54, *1*, pp. 68–75, Jan. 2016, ISSN: 1558-1896. DOI: 10.1109/MCOM.2016.7378428.

[44] *HackRF One - Great Scott Gadgets*, https://greatscottgadgets.com/hackrf/one/.

[45] A. R., G. Xavier, H. V., N. Prasannan, R. Peter, and S. K.P., « GNU Radio Based Control System », *in 2012 International Conference on Advances in Computing and Communications*, Aug. 2012, pp. 259–262. DOI: 10.1109/ICACC.2012.59.

[46] M. Sever and B. Tavlı, « Use of GNU Radio as a Validation and Visualization Tool in Communications Electronic Support Project », *in 2020 28th Signal Processing and Communications Applications Conference (SIU)*, Oct. 2020, pp. 1–5. DOI: 10.1109/SIU49456.2020.9302461.

[47] R. Dhar, G. George, A. Malani, and P. Steenkiste, « Supporting Integrated MAC and PHY Software Development for the USRP SDR », *in 2006 1st IEEE Workshop on Networking Technologies for Software Defined Radio Networks*, Sep. 2006, pp. 68–77. DOI: 10.1109/SDR.2006.4286328.

[48] T. J. Rouphael, « High-Level Requirements and Link Budget Analysis », *in Signal Processing for Software-Defined Radio*, Elsevier, 2009, pp. 87–122, ISBN: 978-0-7506-8210-7. DOI: 10.1016/B978-0-7506-8210-7.00004-7.

[49] M. Robert and B. A. Fette, « The Software-Defined Radio as a Platform for Cognitive Radio », *in Cognitive Radio Technology*, Elsevier, 2009, pp. 65–103, ISBN: 978-0-12-374535-4. DOI: 10.1016/B978-0-12-374535-4.00003-5.

[50] N. Nikaein *et al.*, « OpenAirInterface: An open LTE network in a PC », *in Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, 2014, pp. 305–308.

[51] J. Hoydis *et al.*, « Sionna: An Open-Source Library for Next-Generation Physical Layer Research », 2022. DOI: 10.48550/ARXIV.2203.11854.

[52] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming, and A. Burg, « An Open-Source LoRa Physical Layer Prototype on GNU Radio », *in 2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, May 2020, pp. 1–5. DOI: 10.1109/SPAWC48557.2020.9154273.

[53] A. Lavric, A. I. Petrariu, and V. Popa, « Long Range SigFox Communication Protocol Scalability Analysis Under Large-Scale, High-Density Conditions », *IEEE Access*, vol. 7, pp. 35 816–35 825, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2903157.

[54] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker, « Sora: High-performance software radio using general-purpose multi-core processors », *Communications of the ACM*, vol. 54, *1*, pp. 99–107, 2011.

[55] C.-L. I, J. Huang, R. Duan, C. Cui, J. Jiang, and L. Li, « Recent Progress on C-RAN Centralization and Cloudification », *IEEE Access*, vol. 2, pp. 1030–1039, Aug. 2014, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2351411.

[56] A. Checko *et al.*, « Cloud RAN for Mobile Networks—A Technology Overview », *IEEE Communications Surveys & Tutorials*, vol. 17, *1*, pp. 405–426, Sep. 2015, ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2355255.

[57] C. Mavromoustakis, G. Mastorakis, and C. Dobre, Eds., *Advances in Mobile Cloud Computing and Big Data in the 5G Era* (Studies in Big Data). Springer International Publishing, 2017, ISBN: 978-3-319-45143-5. DOI: 10.1007/978-3-319-45145-9.

[58] M. Flynn, « Very high-speed computing systems », *Proceedings of the IEEE*, vol. 54, *12*, pp. 1901–1909, 1966, ISSN: 0018-9219. DOI: 10.1109/PROC.1966.5273.

[59] M. J. Flynn, « Some Computer Organizations and Their Effectiveness », *IEEE Transactions on Computers*, vol. C-21, *9*, pp. 948–960, Sep. 1972, ISSN: 0018-9340. DOI: 10.1109/TC.1972.5009071.

[60] ARM, *NEON Programmer's Guide*, 1.0. ARM, 2013.

[61] Intel®, *Architecture Instruction Set Extensions Programming Reference*, Jun. 2022.

[62] M. D. Hill and M. R. Marty, « Amdahl's Law in the Multicore Era », *Computer*, vol. 41, *7*, pp. 33–38, Jul. 2008, ISSN: 1558-0814. DOI: 10.1109/MC.2008.209.

[63] I. Zecena, M. Burtscher, T. Jin, and Z. Zong, « Evaluating the performance and energy efficiency of n-body codes on multi-core CPUs and GPUs », *in 2013 IEEE 32nd International Performance Computing and Communications Conference (IPCCC)*, Dec. 2013, pp. 1–8. DOI: 10.1109/PCCC.2013.6742789.

[64] B. Le Gal and C. Jego, « High-throughput multi-core LDPC decoders based on x86 processor », *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 27, *5*, pp. 1373–1386, May 2016.

[65] A. Cassagne *et al.*, « A Flexible and Portable Real-time DVB-S2 Transceiver using Multicore and SIMD CPUs », *in 2021 11th International Symposium on Topics in Coding (ISTC)*, Aug. 2021, pp. 1–5. DOI: 10.1109/ISTC49272.2021.9594063.

[66] B. Le Gal, C. Leroux, and C. Jego, « Multi-Gb/s software decoding of polar codes », *IEEE Transactions on Signal Processing (TSP)*, vol. 63, *2*, pp. 349–359, Jan. 2015.

[67] B. Le Gal and C. Jego, « Low-latency and high-throughput software turbo-decoders on multi-core architectures », *Annals of Telecommunications, Springer*, vol. 75, pp. 27–42, Aug. 2019.

[68] B. Le Gal and C. Jego, « High-Throughput FFT-SPA Decoder Implementation for Non-Binary LDPC Codes on x86 Multicore Processors », *Journal of Signal Processing Systems*, vol. 92, *1*, pp. 37–53, Jan. 2020, ISSN: 1939-8018, 1939-8115. DOI: 10.1007/s11265-019-01447-8.

[69] R. Wilson, *Intel FlexRAN reference designs deployed in 5G infrastructure*, Jun. 2018.

[70] R. Maiden, C. Lanzani, and A. Vora, *Build more cost-effective and more efficient 5G radios with Intel Agilex FPGAs (WP-01312-1.0)*, Manual, INTEL, Intel Programmable Solution Group.

[71] V. Pignoly, B. Le Gal, C. Jego, B. Gadat, and L. Barthe, « Fair comparison of hardware and software LDPC decoder implementations for SDR space links », *in 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Nov. 2020, pp. 1–4. DOI: 10.1109/ICECS49266.2020.9294906.

[72] G. Zhou, A. Liu, K. Yang, T. Wang, and Z. Li, « An Embedded Solution to Visual Mapping for Consumer Drones », *in 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2014, pp. 670–675. DOI: 10.1109/CVPRW.2014.102.

[73] Zhenzhen Ye, Chunjie Duan, P. V. Orlik, Jinyun Zhang, and A. A. Abouzeid, « A Synchronization Design for UWB-Based Wireless Multimedia Systems », *IEEE Transactions on Broadcasting*, vol. 56, *2*, pp. 211–225, Jun. 2010, ISSN: 0018-9316, 1557-9611. DOI: 10.1109/TBC.2010.2042499.

[74] M. Schlüter, M. Dörpinghaus, and G. P. Fettweis, « Bounds on Phase, Frequency, and Timing Synchronization in Fully Digital Receivers With 1-bit Quantization and Oversampling », *IEEE Transactions on Communications*, vol. 68, *10*, pp. 6499–6513, Oct. 2020, ISSN: 1558-0857. DOI: 10.1109/TCOMM.2020.3005738.

[75] R. Chauvat, A. Garcia-Pena, and M. Paonni, « Efficient LDPC-coded CCSK Links for Robust High Data Rates GNSS », *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–13, 2022, ISSN: 1557-9603. DOI: 10.1109/TAES.2022.3190819.

[76] R. Gupta and F. Brewer, « High-Level Synthesis: A Retrospective », *in High-Level Synthesis*, P. Coussy and A. Morawiec, Eds., Dordrecht: Springer Netherlands, 2008, pp. 13–28, ISBN: 978-1-4020-8587-1 978-1-4020-8588-8. DOI: 10.1007/978-1-4020-8588-8_2.

[77] P. Coussy and A. Morawiec, *High-Level Synthesis: From Algorithm to Digital Circuit*, 1. éd. Berlin: Springer Science + Business media B.V, 2008, ISBN: 978-1-4020-8588-8.

[78] G. Martin and G. Smith, « High-Level Synthesis: Past, Present, and Future », *IEEE Design & Test of Computers*, vol. 26, *4*, pp. 18–25, Jul. 2009, ISSN: 1558-1918. DOI: 10.1109/MDT.2009.83.

[79] M. Fingeroff, *High-Level Synthesis Blue Book*. New Jersey: Xlibris Corporation, 2010, ISBN: 978-1-4500-9724-6 978-1-4500-9723-9.

[80] G. D. Micheli, « High-Level Synthesis of Digital Circuits », *in Advances in Computers*, vol. 37, Elsevier, 1993, pp. 207–283, ISBN: 978-0-12-012137-3. DOI: 10.1016/S0065-2458(08)60406-4.

[81] E. Martin, O. Sentieys, H. Dubois, and J. Philippe, « GAUT: An architectural synthesis tool for dedicated signal processors », *in Proceedings of EURO-DAC 93 and EURO-VHDL 93- European Design Automation Conference*, Sep. 1993, pp. 14–19. DOI: 10.1109/EURDAC.1993.410610.

[82] E. Casseau, B. Le Gal, P. Bomel, C. Jego, S. Huet, and E. Martin, « C- BASED RAPID PROTOTYPING FOR DIGITAL SIGNAL PROCESSING », *in EUSIPCO*, Turkey: EUSIPCO, 2005, pp. 1–4.

[83] Xilinx, *Vivado Design Suite User Guide: High-Level Synthesis (UG902)*, 2019.

[84] Xilinx, *Vitis High-Level Synthesis User Guide UG1399 (v2021.1)*, 2021.

[85] B. Khailany *et al.*, « A modular digital VLSI flow for high-productivity SoC design », *in Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18, New York, NY, USA: Association for Computing Machinery, 2018, ISBN: 978-1-4503-5700-5. DOI: 10.1145/3195970.3199846.

[86] M.-T. Tran, « Towards hardware synthesis of a flexible radio from a high-level language », These de Doctorat, Rennes 1, Nov. 2018.

[87] R. Kastner, J. Matai, and S. Neuendorffer, *Parallel Programming for FPGAs*. arXiv, May 2018.

[88] Y. Delomier, B. Le Gal, J. Crenne, and C. Jego, « Model-Based Design of Flexible and Efficient LDPC Decoders on FPGA Devices », *Journal of Signal Processing Systems*, Feb. 2020. DOI: 10.1007/s11265-020-01519-0.

[89] Y. Delomier, B. Le Gal, J. Crenne, and C. Jego, « Model-based Design of Hardware SC Polar Decoders for FPGAs », *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 13, *2*, May 2020.

[90] T. Bjerregaard and S. Mahadevan, « A survey of research and practices of Network-on-chip », *ACM Computing Surveys*, vol. 38, *1*, p. 1, Jun. 2006, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/1132952.1132953.

[91] M. Braun, J. Pendlum, and M. Ettus, « RFNoC: RF Network-on-Chip », *Proceedings of the GNU Radio Conference*, vol. 1, *1*, Sep. 2016.

[92] E. Kreinar, « RFNoC Neural Network Library using Vivado HLS », *Proceedings of the GNU Radio Conference*, vol. 2, *1*, pp. 7–7, Sep. 2017.

[93] K. Saied, A. C. A. Ghouwayel, and E. Boutillon, « Short Frame Transmission at Very Low SNR by Associating CCSK Modulation With NB-Code », *IEEE Transactions on Wireless Communications*, vol. 21, *9*, pp. 7194–7206, Sep. 2022, ISSN: 1536-1276, 1558-2248. DOI: 10.1109/TWC.2022.3156628.

[94] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, « Low-complexity decoding for non-binary LDPC codes in high order fields », *IEEE Transactions on Communications*, vol. 58, *5*, pp. 1365–1375, May 2010, ISSN: 0090-6778. DOI: 10.1109/TCOMM.2010.05.070096.

[95] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, « Short Turbo Codes over High Order Fields », *IEEE Transactions on Communications*, vol. 61, *6*, pp. 2201–2211, Jun. 2013, ISSN: 0090-6778. DOI: 10.1109/TCOMM.2013.041113.120539.

[96] R. Mori and T. Tanaka, « Non-binary polar codes using Reed-Solomon codes and algebraic geometry codes », *in 2010 IEEE Information Theory Workshop*, Dublin, Ireland: IEEE, Aug. 2010, pp. 1–5, ISBN: 978-1-4244-8262-7. DOI: 10.1109/CIG.2010.5592755.

[97] M. Davey and D. MacKay, « Low-density parity check codes over GF(q) », *IEEE Communications Letters*, vol. 2, *6*, pp. 165–167, Jun. 1998, ISSN: 1089-7798. DOI: 10.1109/4234.681360.

[98] S. Pfletschinger and D. Declercq, « Getting Closer to MIMO Capacity with Non-Binary Codes and Spatial Multiplexing », *in 2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Miami, FL, USA: IEEE, Dec. 2010, pp. 1–5, ISBN: 978-1-4244-5636-9. DOI: 10.1109/GLOCOM.2010.5684077.

[99] O. Abassi, L. Conde-Canencia, M. Mansour, and E. Boutillon, « Non-binary low-density parity-check coded cyclic code-shift keying », *in IEEE Wireless Communications and Networking Conference (WCNC'2013)*, Shanghai, China: IEEE, Apr. 2013, pp. 3890–3894. DOI: 10.1109/WCNC.2013.6555196.

[100] G. Liva, E. Paolini, T. De Cola, and M. Chiani, « Codes on high-order fields for the CCSDS next generation uplink », *in 2012 6th Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC)*, Sep. 2012, pp. 44–48. DOI: 10.1109/ASMS-SPSC.2012.6333104.

[101] J. Castiñeira Moreira and P. G. Farrell, *Essentials of Error-Control Coding*. West Sussex, England: John Wiley & Sons, 2006, ISBN: 978-0-470-03571-9.

[102] O. Abassi, « Etude des décodeurs LDPC non-binaires », Ph.D. dissertation, 2014.

[103] G. Dillard, M. Reuter, J. Zeiddler, and B. Zeidler, « Cyclic code shift keying: A low probability of intercept communication technique », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, *3*, pp. 786–798, Jul. 2003, ISSN: 0018-9251. DOI: 10.1109/TAES.2003.1238736.

[104] R. Imad, C. Poulliat, and S. Houcke, « Frame Synchronization Techniques for Non-Binary LDPC Codes over GF(q) », *in 2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Miami, FL, USA: IEEE, Dec. 2010, pp. 1–6, ISBN: 978-1-4244-5636-9. DOI: 10.1109/GLOCOM.2010.5683422.

[105] D. Akopian, « Fast FFT based GPS satellite acquisition methods », *IEE Proceedings - Radar, Sonar and Navigation*, vol. 152, *4*, pp. 277–286, Aug. 2005, ISSN: 1350-2395. DOI: 10.1049/ip-rsn:20045096.

[106] « ISO/IEC/IEEE International Standard - Floating-point arithmetic », *ISO/IEC 60559:2020(E) IEEE Std 754-2019*, pp. 1–86, May 2020. DOI: 10.1109/IEEESTD.2020.9091348.

[107] M. Frigo and S. Johnson, « The Design and Implementation of FFTW3 », *Proceedings of the IEEE*, vol. 93, *2*, pp. 216–231, Feb. 2005, ISSN: 1558-2256. DOI: 10.1109/JPROC.2004.840301.

[108] Volpin, Léa, Le Gal, Bertrand, and Ferré, Guillaume, « Efficient LoRa-like transmitter stacks for SDR applications », *in Proceedings of the IEEE International Conference on Circuits and Systems (ICECS)*, Glasgow, UK, Oct. 2022, P–P.

[109] K. Trifunovic, D. Nuzman, A. Cohen, A. Zaks, and I. Rosen, « Polyhedral-Model Guided Loop-Nest Auto-Vectorization », *in 2009 18th International Conference on Parallel Architectures and Compilation Techniques*, Sep. 2009, pp. 327–337. DOI: 10.1109/PACT.2009.18.

[110] J. S. Walther, « A unified algorithm for elementary functions », *in Proceedings of the May 18-20, 1971, Spring Joint Computer Conference on - AFIPS '71 (Spring)*, Atlantic City, New Jersey: ACM Press, 1971, p. 379. DOI: 10.1145/1478786.1478840.

[111] Ettus Research, *USRP Hardware Driver and USRP Manual: Table Of Contents*, https://files.ettus.com/manual/.

[112] F. Cerqueira and B. B. Brandenburg, « A comparison of scheduling latency in linux, PREEMPT-RT, and LITMUS RT », 2013.

[113] L. S. Blackford *et al.*, « An updated set of basic linear algebra subprograms (BLAS) », *ACM Transactions on Mathematical Software*, vol. 28, *2*, pp. 135–151, 2002.

[114] C. Marchand, E. Boutillon, H. Harb, L. Conde-Canencia, and A. Al Ghouwayel, « Hybrid Check Node Architectures for NB-LDPC Decoders », *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, *2*, pp. 869–880, Feb. 2019, ISSN: 1558-0806. DOI: 10.1109/TCSI.2018.2866882.

[115] *OpenStreetMap*, https://www.openstreetmap.org/copyright.

[116] M. Rady *et al.*, « A Historical Twist on Long-Range Wireless: Building a 103 km Multi-Hop Network Replicating Claude Chappe's Telegraph », *Sensors*, vol. 22, *19*, p. 7586, Oct. 2022, ISSN: 1424-8220. DOI: 10.3390/s22197586.

[117] C. Poulliat, M. Fossorier, and D. Declercq, « Design of non binary LDPC codes using their binary image: Algebraic properties », *in 2006 IEEE International Symposium on Information Theory*, Seattle, WA: IEEE, Jul. 2006, pp. 93–97, ISBN: 978-1-4244-0505-3 978-1-4244-0504-6. DOI: 10.1109/ISIT.2006.261681.

Titre : Implémentation Temps-Réel d'un Récepteur Quasi Cyclic Short Packet

Mot clés : Implémentation Temps Réel, CCSK, Paquets Courts, Matériel, Logiciel, Réseaux Étendus à Faible Puissance

Résumé : Dans les communications sans fil, la détection et la synchronisation des trames sont généralement effectuées à l'aide d'un préambule, ce qui consomme une quantité de bande passante et de ressources non négligeables lors de l'envoie de petits paquets de données. Récemment, un nouveau type de trame sans préambule appelé Quasi Cyclic Short Packet (QCSP) a été proposé. Cette thèse étudie les possibilités de mise en œuvre temps-réel de la chaine de communication QCSP. À cette fin, les algorithmes sont détaillés, tant en émission qu'en réception, puis, lorsque cela est possible, optimisés. De plus, la tâche la plus critique du récepteur, la détection, est étudiée en profondeur. Différents niveaux de parallélisme et stratégies d'implémentation sont détaillés pour des implémentations logicielles, mais aussi matérielles. Plusieurs compromis entre le débit et l'utilisation des ressources sont également discutés. Enfin, des expériences grandeur nature sont présentés. Ainsi, la thèse démontre que le processus d'émission/-réception d'une trame QCSP est réalisable à un faible coût matériel, ce qui rend la trame QCSP attrayante pour les réseaux étendus à faible puissance (LPWAN).

Title: Real-Time Implementation of a Quasi Cyclic Short Packet Receiver

Keywords: Real-Time Implementation, CCSK, Small Packets, Hardware, Software, Low Power Wide Area Network

Abstract: In wireless communications, frame detection and synchronization are usually performed using a preamble, which consumes a significant amount of bandwidth and resources when sending small data packets. Recently, a new kind of preamble-less frame called Quasi Cyclic Short Packet (QCSP) has been proposed. This thesis investigates the possibilities of real-time implementation of the QCSP communication chain. To this end, the algorithms are detailed, both in transmission and reception, and then, where possible, optimized. In addition, the most critical task of the receiver, the detection, is studied in depth. Different levels of parallelism and implementation strategies are detailed for both software and hardware implementations. Several trade-offs between throughput and resource utilization are also discussed. Finally, full-scale experiments are presented. Thus, the thesis demonstrates that the process of transmitting/receiving a QCSP frame is feasible at a low hardware cost, which makes the QCSP frame attractive for low power wide area networks (LPWAN).