UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
Ingegneria dell'Informazione
SSD ING/INF-03

Tesi di Dottorato di Ricerca

Programming techniques for efficient and interoperable software defined radios

Autore:

Mario Di Dio

Tutori:
Prof. Marco Luise
Prof. Filippo Giannetti

Anno 2013

Manuscript submitted on February 28th, 2013

Accepted April 9th, 2013

Sommario

Negli ultimi anni la ricerca sui sistemi Software-Defined Radio (SDR) ha rappresentato uno degli argomenti di maggiore interesse nel campo delle comunicazioni wireless. Ciò è dovuto da un lato alla crescente richiesta di sitemi di comunicazione radio riconfigurabili ed interoperabili in grado di imparare dall'ambiente circostante e sfruttare efficacemente lo spettro, realizzando così di fatto il paradigma delle radio cognitive, e dall'altro alla crescente disponibilità di piattaforme hardware riprogrammabili in grado di fornire la potenza di calcolo necessaria per soddisfare i requisiti stringenti di tempo-reale tipici degli standard di comunicazione a larga banda allo stato dell'arte. La maggior parte delle implementazioni SDR sono basate su architetture miste in cui coesistono Field Programmable Gate Arrays (FPGA), Digital Signal Processors (DSP) e General-Purpose Processors (GPP). Sebbene assicurino il massimo grado di flessibilità, le soluzioni architetturali basate esclusivamente su GPP sono tipicamente evitate a causa della loro inefficienza computazionale ed energetica.

Partendo da queste considerazioni, questa tesi si propone di affrontare in maniera congiunta due degli aspetti più importanti nei sistemi SDR GPP-based: l'efficienza computazionale e l'interoperabilità. Nella prima parte della tesi, vengono presentate le potenzialità di una nuova tecnica di programmazione, chiamata Memory Acceleration (MA), in cui le risorse di memoria tipiche dei sistemi GPP-based vengono utilizzate per assistere l'unità centrale di calcolo nell'esecuzione real-time delle operazioni di signal processing. Questa tecnica, appartenente alle tecniche di ottimizzazione tipiche dei sistemi informatici note come Space-Time Trade-Offs, definisce dei nuovi metodi algoritmici in grado di assistere gli sviluppatori nella fase di design di algoritmi di signal processing software-defined. Al fine di dimostrare l'applicabilità di tale tecnica, vengono inoltre descritte alcune implementazioni "mondo-reale" insieme ai fattori di accelerazione ottenuti. Nella seconda parte della tesi, viene analizzato l'aspetto riguardante l'interoperabilità dei sistemi SDR. Le architetture software esistenti, come

la Software Communications Architecture (SCA), astraggono i componenti hardware/software di una di catena di comunicazione radio attraverso l'utilizzo di un middleware, come ad es. CORBA, e forniscono alla catena implementata, chiamata waveform nel linguaggio SCA, il massimo grado di portabilità e interoperabilità tra piattaforme SDR eterogenee. Tale caratteristica implica un aumento del carico computazionale dovuto alla presenza del middleware ed è anche una delle ragioni per cui le implementazioni SDR GPP-based SCA-compliant sono generalmente evitate anche nel caso di waveform a banda stretta come le comunicazioni analogiche in banda VHF. All'interno di questa tesi vengono analizzati l'architettura SCA e il framework di svilupo OSSIE, e vengono indicate alcune linee guida per modificare il suddetto framework ed abilitare la programmazione multithreading component-based e il settaggio della CPU affinity. Viene quindi descritta l'implementazione di una waveform real-time SCA-compliant (transceiver VHF per comunicazioni aeronautiche di tipo voce) sviluppata all'interno di questo framework. Infine, viene fornita la prova di come sia possibile, utilizzando in modo congiunto la tecnica MA e l'architettura SCA, implementare su una piattaforma GPP-based una waveform SCA-compliant tempo-reale a banda larga (AeroMACS) ottenendo ottime prestazioni sia in termini di efficienza computazionale che di interoperabilità.

Abstract

Recently, Software-Defined Radios (SDRs) has became a hot research topic in wireless communications field. This is jointly due to the increasing request of reconfigurable and interoperable multi-standard radio systems able to learn from their surrounding environment and efficiently exploit the available frequency spectrum resources, so realizing the cognitive radio paradigm, and to the availability of reprogrammable hardware architectures providing the computing power necessary to meet the tight real-time constraints typical of the state-of-art wideband communications standards. Most SDR implementations are based on mixed architectures in which Field Programmable Gate Arrays (FPGA), Digital Signal Processors (DSP) and General Purpose Processors (GPP) coexist. GPP-based solutions, even if providing the highest level of flexibility, are typically avoided because of their computational inefficiency and power consumption.

Starting from these assumptions, this thesis tries to jointly face two of the main important issues in GPP-based SDR systems: the computational efficiency and the interoperability capacity. In the first part, this thesis presents the potential of a novel programming technique, named Memory Acceleration (MA), in which the memory resources typical of GPP-based systems are used to assist central processor in executing real-time signal processing operations. This technique, belonging to the classical computer-science optimization techniques known as Space-Time trade-offs, defines novel algorithmic methods to assist developers in designing their software-defined signal processing algorithms. In order to show its applicability some "real-world" case studies are presented together with the acceleration factor obtained. In the second part of the thesis, the interoperability issue in SDR systems is also considered. Existing software architectures, like the Software Communications Architecture (SCA), abstract the hardware/software components of a radio communications chain using a middleware like CORBA for providing full portability and interoperability

to the implemented chain, called waveform in the SCA parlance. This feature is paid in terms of computational overhead introduced by the software communications middleware and this is one of the reasons why GPP-based architecture are generally discarded also for the implementation of narrow-band SCA-compliant communications standards. In this thesis we briefly analyse SCA architecture and an open-source SCA-compliant framework, ie. OSSIE, and provide guidelines to enable component-based multithreading programming and CPU affinity in that framework. We also detail the implementation of a real-time SCA-compliant waveform developed inside this modified framework, i.e. the VHF analogue aeronautical communications transceiver. Finally, we provide the proof of how it is possible to implement an efficient and interoperable real-time wideband SCA-compliant waveform, i.e. the AeroMACS waveform, on a GPP-based architecture by merging the acceleration factor provided by MA technique and the interoperability feature ensured by SCA architecture.

Acknowledgments

Before detailing the challenges and the achievements of this thesis I want to express my sincere gratitude to my first advisor Prof. Marco Luise who offered me the PhD opportunity. In these years, his technical enthusiasm, guidance and supervision were essential during all the activities described in this thesis. It was a real honour for me having him as advisor.

I want also to thank my second advisor Prof. Filippo Giannetti for his precious suggestions and continuous support and also for having lent me some of his fantastic ham radios, becoming crucial for the verification and validation tests on the developed SDR systems.

Then I would like to thank my colleague and friend Dr. Vincenzo Pellegrini with which I have had the honour to share the development and formalization of the Memory Acceleration technique during our daily (and nightly) intense work sessions.

I would like to thank Dr. Marco Moretti for having gave me the possibility of teaching and spreading the SDR paradigm in his courses.

I owe my deepest gratitude to Dr. Giacomo Bacci, a point of reference for me during these years. His deep attitude to problem solving and help, together with an incredible technical knowledge, made working with him an immense opportunity to learn and improve my research method.

I want also to thank Dr. Luca Sanguinetti and Simona Moschini for their tireless support and encouragement, and my DSPCoLa labmates Andrea e Ottavio. Our collaboration was not only a working experience but also an opportunity for building a solid friendship.

Thanks also to my PhD classmates Mr.(Dr.) Riccardo Andreotti and Mr.(Dr.) Loris Gazzarrini for their remote help, especially in the last months.

Then I would like to thank my family for having always supported me, my desires and choices during my life. Nothing would have been possible to achieve without their

infinite help and love.

Finally, I want to thank my sweet Sara, who supported and encouraged me in any difficulty faced during these years and followed me until the "limits of the world" sharing with me the desires, the dreams and the feelings of the "new world". Thank you my love.

Pisa, April 2013 Mario Di Dio

Contents

List of Figures

List of Tables

List of Acronyms

ACGN	Additive Coloured Gaussian Noise
ADC	Analogue to Digital Converter
AM	Amplitude Modulation
AOC	Airline Operation Center
AS	Algorithm Segmentation
ASIC	Application Specific Integrated Circuit
ATC	Air Traffic Control
ATS	Air Traffic Service
AWGN	Additive White Gaussian Noise
B-GAN	Broadband Global Area Network
BOC	Binary Offset Carrier
BPSK	Binary Phase Shift Keying
B4G	Beyond-4G
CDMA	Code Division Multiple Access
CF	Core Framework
COFDM	Coded OFDM
CR	Cognitive Radio
CRB	Cramér-Rao Bound
CORBA	Common Object Request Broker Architecture
CP	Cognitive Positioning
CPS	Cognitive Positioning System
CPU	Central Processor Unit
DAC	Digital to Analogue Converter

DC	Direct Current
DEPE	Delay Estimation through Phase Estimation
DoD	Department of Defense
DSP	Digital Signal Processing
DSPCOLA	DSP for Communication Lab
DVB	Digital Video Broadcasting
DVB-T	Digital Video Broadcasting - Terrestrial version
EPG	Electronic Program Guide
ETSI	European Telecommunication Standards Institute
FBMCM	Filter-Bank Multicarrier Modulation
FCH	Frame Control Header
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
FPGA	Field-Programmable gate Array
FM	Frequency Modulation
FMT	Filtered MultiTone
FSR	Feedback Shift Register
GIOP	General Inter-ORB Protocol
GNU	GNUs Not Unix (Recursive Acronym)
GPP	General Purpose Processor
GPS	Global Positioning System
GSM	Global System for Mobile communications
HW	Hardware
I/Q	In Phase/Quadrature
IBI	Inter Block Interference
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
IID	Independent and Identically Distributed

IIOP	Internet Inter-ORB Protocol
IOR	Interoperable Object Reference
IP	Internet Protocol
ISI	Inter-Symbol Interference
JTRS	Joint Tactical Radio System
MA	Memory Acceleration
MAC	Media Access Control
MC	MultiCarrier
MCRB	Modified Cramér-Rao Bound
MEV	Mean Estimated Value
MIDS	Minimum Independent Data Set
ML	Maximum Likelihood
MPEG2	Motion Picture Experts Group 2
MSE	Mean Square Error
MSEE	Mean Square Estimation Error
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple-Access
OMG	Object Management Group
ORB	Object Request Broker
OpenMP	Open Multi Processing
OPS	Operation Per Second
OS	Operating System
OSSIE	Open-Source SCA Implementation - Embedded
PC	Personal Computer
PID	Process Identifier
PHY	Physical Layer
PN	Pseudo Noise
POA	Portable Object Adapter
PRBS	Pseudo Random Binary Sequence
PSD	Power Spectral Density

PSF	Pulse Shape Factor
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quaternary Phase Shift Keying
RAM	Random Access Memory
RF	Radio Frequency
RMSEE	Root Mean Square Estimation Error
RS	Reed-Solomon (Codes)
RTAR	Recursive Table Aggregation Rule
SANDRA	Seamless Aeronautical Networking through integration of Data links, Radios, an Antennas
SESAR	Single European Sky ATM Research
SIMD	Single Instruction Multiple Data
SCA	Software Communications Architecture
SCPC	Single Channel Per Carrier
SDR	Software Defined Radio
SNR	Signal to Noise Ratio
SRRC	Square Root Raised Cosine
SW	Software
TB	Table Boundary
TDD	Time Division Duplexing
TDE	Time-Delay Estimation
TETRA	TErrestrial Trunked RAdio
TPS	Transmission Parameters Signalling
TS	Transport Stream
TX	Transmitter
VDL	VHF Digital Link
VDL2	VHF Digital Link mode 2
VHF	Very High Frequency
VOLMET	Volume Meteorological

USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
UWB	Ultra Wide Band
XML	Extensible Markup Language
ZF	Zero Forcing
4G	4th Generation

Introduction

Motivations

Since from its formal birth in 1992 [1], the number of research activities on Software Defined Radio (SDR) systems has been increased so that it became one of the hot topic in the wireless communications research field. In particular, this technology has reached a quite solid spreading factor and maturity in the military applications because of its tremendous consequences on systems integration and device convergence [2] [3]. This effect was also due to the increasing availability of reconfigurable hardware (HW) platforms able to satisfy the tight real-time constraints typical of the state-of-art wideband standards. There is not a well-established standard hardware architecture and so SDR platform often use, as computational asset, a custom mixture of Field Programmable Gate Arrays (FPGA), Digital Signal Processors (DSP) and General Purpose Processor (GPP) [4]. In this devices the flexibility is inversely proportional to their computational power.

In fact, due to their high flexibility, GPP-based platforms are the most attractive option for both research and deployment because they slow down both the the development costs and the time-to-market. In fact, costs associated to the hardware design and production typical of the Application Specific Integrated Circuits would be cancel since the reference implementation written in a high-level language (C, C++, Java,..) could already be a finished product ready for deployment. On the other hand, it is recognized that the main limitation of a GPP-based platform is the low computational power and the consequent power inefficiency measured as throughput per Watt. For this reason FPGA-based (hardware-like) implementations are generally preferred, so giving up to the full flexibility feature that is the core of the dream of "Universal Radio" depicted by Mitola [1].

The objective of the first part of this thesis is showing that it is possible to increase

the computational efficiency of the GPP-based SDR systems by exploiting a resource hugely present in GPP platform and not completely used in current SDR implementations, i.e. the memory resources. In fact, starting from classical computer-science approaches [5], we define some new algorithms methods, which we gather under the name of Memory Acceleration (MA) technique, which help the central processor in its processing tasks and so allow the usage of less power-hungry GPP devices or, equivalently, increase the power efficiency of the used GPP. This achievement could have an impact also in the general-purpose hardware industry. In fact, the reduction of the power efficiency gap between GPP-based SDR and HW solutions, could encourage the implementation of GPP architectures tailored for SDR products in with there is a special attention to memory resource management. We could call them *radio processing cores*. Moreover, this would allow to reach greater acceleration factors applying MA technique.

In addition to that, real-time issue in SDR has to be considered also in relation with the software architecture in which it is implemented. In fact, not negligible computational overhead could be due to the usage of a structured, hierarchical software architecture able to provide specific capabilities to the SDR implementation. In particular, in military applications there is a particular attention to the portability and interoperability issues in SDR systems. In fact, the reaching of the device convergence with the creation of a unique reconfigurable SDR terminal is a central objective in the modern military communications. This can be really realized only if the SDR implementations are developed inside a hierarchical and well-established software architecture. Under these assumptions, the US Join Tactical Radio System (JTRS) group formalized a new software architecture tailored for SDR systems, i.e. the Software Communications Architecture (SCA) [3]. The core of this architecture is the presence of a software middleware (CORBA in release 2.2.2) abstracting the hardware/software components and so providing platform-independence to the implemented SDR, called *waveform* in the SCA parlance. Starting from the military field, SCA is being adopted also on civil applications, as witnessed by several European FP7 project like SANDRA [6] and EULER [7] and by the presence in the market of many commercial SDR platforms tagged as SCA-compliant [8] [9]. As said, the success of this architecture is given by its hierarchical and well-organized structure. Unfortunately, a so stratified and hierarchical architecture is paid in terms of computational overhead and this is the reason why the most recognized SCA-compliant platforms are generally based on a

mixture of FPGAs and DSPs, so leaving to the GPP (if present) only some housekeeping functions. For this reason, the other objective of this thesis is providing a more computational efficient SCA architecture starting from a SCA-compliant open-source implementation developed at the WirelessGroup@VirginiaTech, i.e. OSSIE [10]. In fact, we provide guidelines to enable component-based multithreading programming and cpu affinity in the aforementioned framework without affecting the flexibility feature and we merge this with the achievements coming from the application of the MA technique.

Computational efficiency and interoperability can be seen as mutually dependent features of any modern GPP-based SDR. In fact, the optimization of the computational complexity of the signal processing functions composing a SDR can be successful used to face also the computational overhead given by a interoperable software architecture like SCA, so realizing the ultimate goal of this thesis.

Main contributions

The main contribution of this thesis are detailed below:

a. First of all we investigate possible acceleration techniques in the classical field of Space Time trade-off that could be tailored for SDR systems and starting from this we define a novel acceleration algorithm, i.e. the MA technique, that it is based on a smart usage of the memory resources typical of GPP-based system. The rationale behind this is that memory is not a power-hungry device and it can be used as computational asset for helping the central processor in performing its tasks, so allowing the usage of less powerful and not power-hungry GPP devices. Furthermore, we detail some performance evaluation tools for this technique and provide "real-world" test cases showing how this technique is completely generic and it can be applied to any signal processing functions.

b. We investigate also the interoperability issue in SDR systems by analysing in details the Software Communications Architecture and the SDR framework OSSIE. We show how the presence of a middleware can have a non-negligible impact on the computational performance and we try to face this effect exploiting GPP multi-core architectures via multithreading programming. We provide

some modifications to the OSSIE framework and develop a real-time SCA-compliant waveform to validate our approach, obtaining good results both in the computational performance and in the performed interoperability validation tests.

c. We obtain the maximum implementation gain by merging the two paradigms. In fact, we integrate the MA technique in the SCA framework and develop a wideband SCA-compliant waveform, i.e. AeroMACS, able to satisfy tight real-time constraints typical of wideband standards also on a GPP-based platform and provide the interoperability feature ensured by the usage of the SCA architecture.

Outline

This thesis is structured as follows:

a. **Chapter1** details a novel programming technique, called Memory Acceleration, jointly developed with Dr. V. Pellegrini [11]. Its main objective is improving the computational power efficiency of the GPP-based SDR systems and enabling real-time processing also on low-profile computing platforms. The rationale behind this technique and a detailed description of algorithms and performance evaluation tools are presented.

b. **Chapter2** shows the potential of the MA technique describing the computational performance of a "real-world" implementation of a fully-software receiver for ETSI DVB-T signals. We describe the receiver architecture focusing on the computational heaviest signal processing functions working in real-time thanks only to the extensive usage of the MA technique.

c. **Chapter3** analyses the interoperability issue in SDR systems. We concentrate on the SCA architecture, the JTRS project able to provide portability and interoperability to the diverse SDR implementations thanks to the usage of a software middleware abstracting the software/hardware components. In particular, we analyse an open-source SCA-compliant framework developed by WirelessGroup@VirginiaTech, i.e. OSSIE; we concentrate on the exploitation of the multi-core architecture by means of multi-threading programming and

cpu affinity and we provide guidelines to modify the OSSIE framework in order to enable these features.

d. **Chapter4** presents the implementation of a fully-software SCA-compliant waveform implementing a transceiver for VHF voice analogue communications. We detail both the hardware/software architecture, the computational performance and the performed validation tests.

e. **Chapter5** describes how it is possible to merge the acceleration factor provided by MA technique with the interoperability feature given by the SCA architecture. To proof this a wideband SCA-compliant waveform is implemented, i.e. the AeroMACS waveform, a WiMax-inspired standard for aeronautical communications. The computational overhead given by the usage of SCA architecture is balanced by the acceleration factors ensured by MA technique. The resulting waveform is capable of running in real-time on a Intel GPP-based platform with very good computational performance preserving anyway the interoperability feature of a SCA-compliant waveform.

f. **Conclusions** concludes this thesis, highlighting the most relevant achievements in this research as well as the development and research perspectives coming out from this thesis.

Chapter 1

Techniques for improving efficiency in SDR systems: the MA technique

In this chapter we describe a programming technique developed during the Ph.D. studies together with Dr. Vincenzo Pellegrini [11]. The aim of this techinque is showing that, by making use of all the available resources on a GPP-based platform (i.e. not only CPU time, but also Random Access Memory), it is possible to bridge the gap in terms of computational speed and power efficiency that now exists between GPP-based and HW-accelerated SDRs. This efficiency boost is based on revisiting classical concepts already known in computer science under the collective denomination of *space/time tradeoffs*. In previous literature, space/time tradeoffs are intended as a means to reduce the execution time of a certain algorithm either by increasing the degree of HW/SW parallelism of a given implementation (therefore consuming more *space*) [5], [12], or by pre-computing the data produced by some well-determined algorithm and casting it into some tabular form [13] (sacrificing again space, in terms of size of the table to be stored, to gain execution time). Other works recently appeared such as [14], which propose joint usage of look-up tables and Single Instruction Multiple Data (SIMD) programming within a software radio context in order to achieve significant speedups. Based on the assumption that increasing cache size and memory resources on a computing system comes at a much smaller power consumption cost than increasing clock frequency, and therefore offers larger performance improvement margin, we tried instead to focus only on the

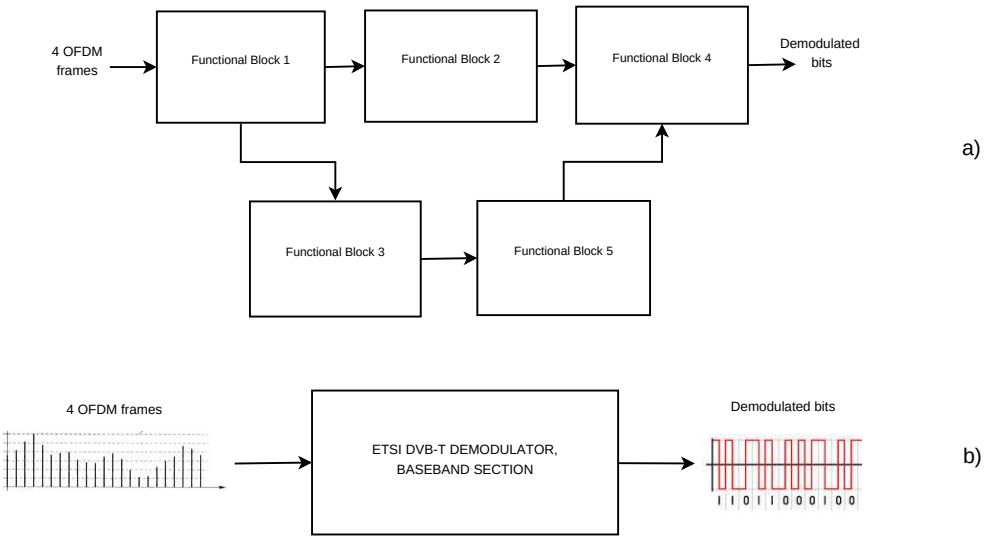

Figure 1.1: *Possible system representations: mesh of constituent blocks a), black box b)*

memory Vs computation trade-off. We did this by exploring the application of such trade-off to radio signal processing in depth in order to provide a convenient and rather general SW design criterion which enables fast implementation of any radio chain in a memory-intensive fashion.
We will introduce introduces the principle of the Memory Acceleration (MA) design rule and then we will discuss some applicative issues trying also to evaluate the performance metrics of such techniques. Some case studies for the MA tecnhique are also reported with the evaluation of the perspectives of this tecnique.

1.1 MA-driven SDR design process

1.1.1 Useful quantities and naming conventions

We start our discussion by observing that any radio terminal and, more generally, any system performing signal processing functions, can be represented as the interconnection of a number of constituent functional blocks. Simple systems are arranged as a straightforward cascaded "chain" of elementary blocks, more complicated schemes (possibly with feedback connections) look more like a "mesh" of components and

connections, as depicted in figure 1.1. Focusing for simplicity on a radio receiver, whatever the mesh of blocks and connection is, the end-to-end signal processing function of our system is equivalent to a mathematical function $f(...)$ which maps a certain amount of soft-valued input symbols (for instance the signal samples collected in a given signal frame) into the corresponding hard-valued demodulated information bits as shown in figure 1.1 b). We call the minimum set of soft channel symbols that can be processed independently from the remainder of the stream the *Minimum Independent Data Set* (MIDS). For the ETSI DVB-T [15] standard (our case study), this would be 4 Orthogonal Frequency-Division Multiplexing (OFDM) frames (i.e. what is called a *superframe* in [15]). We indicate the size (number of items) of the MIDS with symbol l and with A the cardinality of the alphabet each input datum of the MIDS belongs to. The domain of $f(...)$ is then defined as the set of all possible different messages within a MIDS. We also call *input space* the domain of $f(...)$ and C_i the cardinality of such space. Clearly

$$C_i = A^l \tag{1.1}$$

If we could find a convenient analytical expression for the function $f(...)$, we could consider implementing our sample DVB-T demodulator by programming such analytical expression into a computing system via any high-level programming language like C/C++. This would be a *computation-only* implementation of the system, one that is completely located at the *time* end of the time/space trade-off. Such implementation would only (or mainly) take advantage of the *computational* resources being available on a GPP-based platform, with very little attention to the *memory* resources that are available.

After this remark on memory resources, it would be natural to think of replacing our function $f(...)$ with a *tabular* implementation of $f(...)$: a table $t(...)$ containing, for each of the A^l items of the overall input space, the associated output value. This would be a *memory-only* implementation, located at the *space* end of the trade-off, and would not require any (or would require negligible) real-time computation. On the other hand, the size C_i of the table would not be practical for any memory technology available today or in the foreseeable future. The table $t(...)$ could be filled up by running once and forever, at instantiation time (i.e., at the time of initialization or configuration of the terminal) any standard, computation-only, implementation of function $f(...)$ (i.e. the traditional radio system chain) over the entire input space.

Such considerations suggest that the path towards optimal SDR implementations lies somewhere in between the two ends, with a hybrid approach that could use *both* computational *and* memory resources to the greatest possible extent.

Let us now come back to the mesh representation of the signal processing functions of our SDR. We call this representation the *0-step* of a procedure underlying MA, that we label *Algorithm Segmentation* (AS). Such *0-step* may be the direct translation of the signal processing functions described in a communications standard, in a reference implementation. The implicit assumption in this decomposition is that each of the functional blocks $f_n(...)$ in the mesh is *atomic*, i.e., impossible to break-up in a further mesh of constituent functional blocks. On the contrary, the aim of our *algorithm segmentation* approach is just coming to a further decomposition of a functional block $f_n(...)$, formerly assumed to be atomic, into a chain (or mesh) of constituent sub-blocks $f_{n,p}(...)$, $p = 1, ..., P_n$ whose end-to-end behavior is equivalent to the original function $f_n(...)$. One advantage of this is that the input spaces of sub blocks $C_{i_{n,p}}$ will be different from and significantly smaller than C_{i_n}, provided that the segmentation is performed correctly. Algorithm segmentation cannot be considered as a form of algorithm re-design: as a consequence, algorithm segmentation *does not change the overall computational cost of the segmented algorithm.* We will describe algorithm segmentation with convenient details in section 1.1.4.

1.1.2 MA implementation design

An expedient visual representation of the SDR signal processing mesh is obtained as follows: we call W_n the *computational cost* of the *n-th* functional block (required number of CPU instructions or Operations Per Second (OPS)), and we use a graphical representation of the SDR in which the size of the functional block is directly proportional to such cost, as in figure 1.2. This gives at a glance an indication of the relative computational weight of each block (function) within the whole radio. Let us also introduce the symbol Ω_m as the total computational cost of *memory management* for table $t_m(...)$, something that has nothing to do with algorithm complexity, but that represents the cost of memory address calculation plus memory access latency (if significant). The latter parameter can be made equivalent to a computational cost by reducing it to CPU time or to equivalent OPS/clock cycles.

We come now to the core of the MA technique. Broadly speaking, the aim of MA

Table 1.1: *MA symbols and taxonomy*

Symbol	*Meaning*
$f_n(...)$	Computation-only implementation of block n
$t_n(...)$	Memory-only implementation of block n
l	Number of items within the MIDS
A	Cardinality of Alphabet for each item of MIDS
C_{i_n}	Cardinality of input space of block $f_n(...)$
C_{o_n}	Cardinality of output space of block $f_n(...)$
W	Total available computational power
W_n	Computational cost of block $f_n(...)$
W_{TB}	Computational cost of subsystem within table boundary
W_m	Computational cost of subsystem replaced by table m
W_r	Computational cost of not yet memory-accelerated subsystem
Ω_m	Computational cost for handling table $t_m(...)$
M	Total size of available memory
M_m	Total memory footprint of table $t_m(...)$
S_m	Data size of items stored in $t_m(...)$
a	Acceleration factor
η	Acceleration efficiency
η_m	Acceleration efficiency of table $t_m(...)$
I	Overall SDR implementation merit parameter

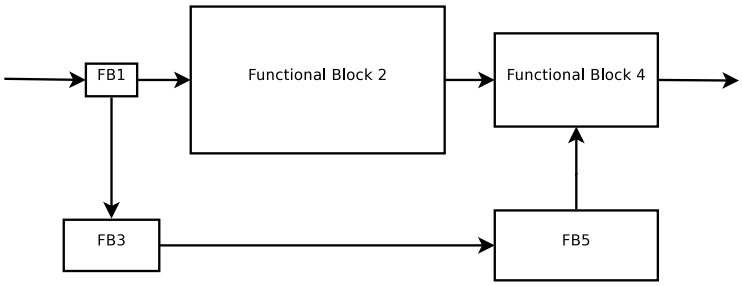

Figure 1.2: *Computational cost weighted functional block representation. Blocks 1 and 4 are* peripheral

is aiding the GPP in processing the informative signal through a proper (extensive) usage of memory resources. Such result is obtained by replacing the functional blocks $f_n(...)$ usually implemented according to a purely-computational (*time*) approach (with marginal usage of memory), with pre-computed tables $t_m(...)$ as introduced in 1.1.1. The replacement is done after one or more steps of algorithm segmentation have been carried out, and on the most demanding blocks in terms of computational power only. In subsection 1.1.3, we will introduce the so-called Recursive Table Aggregation Rule (RTAR) that finalizes tables that will have to be implemented into memory, while also calling for algorithm segmentation to be applied upon the convenient functional blocks. In this respect, notice that the *input space cardinality* C_{i_n} of each $f_n(...)$ is usually unrelated to its *computational cost* W_n. Just to make an example, consider the *data deinterleaver* in a DVB-T demodulator, that performs the inverse operation of the interleaver used in the modulator to scatter around the protected bits of an encoded data block in order to protect them from time-correlated errors. The cardinality of the input space of the deinterleaver is the same as that of its own adjacent binary FEC decoder (they bear the same block length), but the computational complexity of the decoder is definitevely larger than that of the deinterleaver.

We already mentioned that, after segmentation, only the computational heaviest blocks need being implemented in a tabular form. This rule comes from the consideration that the amount of memory resources M is finite, and has to be used in an optimal fashion. Performing memory-acceleration (i.e., tabular implementation) of low-complexity blocks would only result in a waste of memory resources, with

negligible impact on processing speed. The MA rule attains an optimum configuration whenever the available memory resources are exhausted, and the maximum number of operations (or the maximum possible amount of CPU time) has been replaced by memory look-ups implementing the same functions. Replacement of computation-dominated blocks has to follow a hierarchical approach, starting form the most, down to the least demanding, until available memory resources are over.

The computational complexity of a tabular implementation $t_m(...)$ is not zero. We have to consider in fact the memory management cost Ω_m that can be at times non negligible. From this standpoint, it is apparent that the larger is the number of functional blocks $f_n(...)$ that we collapse into a single tabular implementation $t_m(...)$, the smaller is the total memory management cost Ω of our memory-accelerated implementation. In addition to this, we must also consider that GPP-based platforms have a hierarchical memory structure with smaller and faster caches in the proximity of the computing cores and bigger, slower extended memories in a more peripheral location of the system. The general rule to use efficiently such hierarchical memory arrangement is *storing contiguously in memory information which is used contiguously in time*. This means for instance that a series of consecutive blocks in a processing chain are accelerated very efficiently when their processing is aggregated (as far as possible) into a single table $t_m(...)$ whose internal arrangement reproduces the same cascaded structure of the original chain (RTAR is designed in order to yields this). This happens because if such criterion is observed, either the whole table fits into the CPU cache or any subset of the table is fetched into cache only once and never gets used twice, thus maximizing cache friendliness. Once recognized that aggregating blocks in such a structured way is a virtue *per se*, we introduce in the next subsection a Recursive Table Aggregation Rule. Following this rule, we can on one hand provide the aggregation of as many functional blocks as possible into the same table, while on the other we can perform algorithm segmentation –which still remains a demanding design task– only for those blocks where it is really needed and useful.

1.1.3 Recursive Table Aggregation Rule

Assuming that we have an atomic mesh decomposition of our end-to-end algorithm (our SDR), what is the optimum level of break-up to replace computation-intensive blocks with tables? We try to give an answer to this fundamental question through

the RTAR, whose aim is to come to a balanced (optimum) time/space tradeoff in the design of the SDR, possibly providing also cache friendliness.

We start by enclosing the subsystem we intend to memory-accelerate into a closed line that we call the *Table Boundary* (TB). The connections that crosses the TB represent the input/output interface towards the external world of the subsystem under consideration. An atomic block of the subsystem is called *peripheral* if all of its input *or* all of its output connections cross the TB. Once this is defined, the RTAR proceeds as follows:

a. (Initialization) Define the whole radio the as the (sub-)system to be memory-accelerated. This is equivalent to enclosing the entire radio within a TB. Calculate the size C_i of the table that is necessary to memory-implement the selected subsystem (the whole radio). If the table fits into memory, then go to step 3

b. (Reduction) Identify the computationally-lightest block contained within the TB and *reduce* the subsystem by releasing such block (move it outside the TB as in Fig. 1.3). If the released block is not peripheral, then release also all blocks depending on its output, see Fig. 1.5. Calculate the size C_i of the table equivalent to the enclosed system; if the table fits, then go to step 3), otherwise *reduce again* until either i) the table fits (in this case, still go to step 3), or ii) the atomicity limit $f_n(...)$ is reached (figure 1.4). If the latter becomes true, perform *algorithm segmentation* upon the block $f_n(...)$ being currently surrounded by TB and go back to step 2). [1]

c. (Memory-only Implementation) Implement the subsystem being enclosed in the current table boundary by replacing its computation-only functional blocks with an equivalent table $t_m(...)$. If the system still contains blocks not yet

[1] Note that, in the case atomicity limit is reached, the block which undergoes algorithm segmentation is the heaviest block of the whole radio, and therefore the sub-blocks obtained from its segmentation still collectively yield the majority of the computational cost of the SDR. Still, as soon as the first of obtained sub-blocks gets released, we cannot guarantee that this keeps true. Thus, whenever one of the sub-blocks obtained from algorithm segmentation is released, it must be checked whether the table boundary encloses a computational cost W_{TB} which is still greater than the cost of any functional block outside the TB. In case this condition becomes false, the TB must be re-initialized to enclose the entire system and the procedure shall continue from step 2).

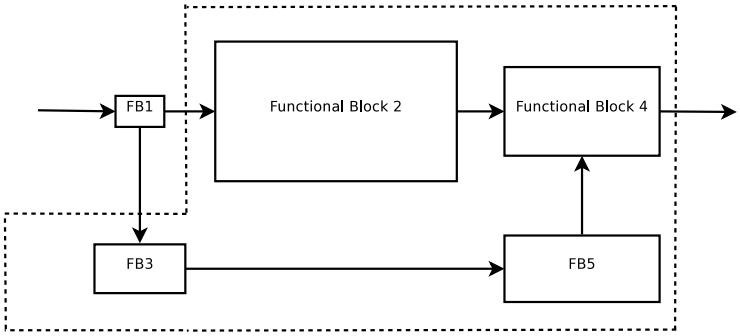

Figure 1.3: *Table boundary after one step, released block is peripheral*

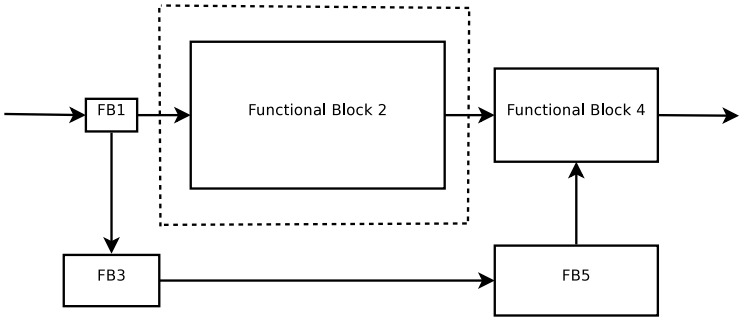

Figure 1.4: *Atomicity limit reached*

implemented in memory, and memory resources are not exhausted, initialize the table boundary for another MA iteration by enclosing all of the remaining computational blocks of the radio system, then go to step 2).

The proposed RTAR rule is admittedly heuristic - an exhaustive approach to algorithm segmentation to find the optimum configuration of the SDR appears unfeasible. Nonetheless, we believe that our rule captures the majority of the achievable MA gain with a manageable approach. In some test cases (conducted upon rather heterogeneous signal processing algorithms), RTAR was shown to provide substantial speedup factors (roughly one order of magnitude) with an acceptable MA design effort.

Cache-friendliness provided by RTAR also constitutes the basis for MA compatibility with parallel programming. Memory access contentions that could indeed happen when loading the required memory table (or table portion) from the external Random

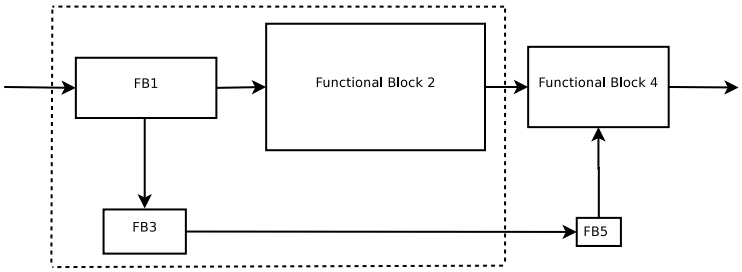

Figure 1.5: *Example of released block (FB5) being non-peripheral. In this case cascaded blocks (FB4) are released as well. At next step of the iterative algorithm, formerly released blocks will be enclosed in the new table boundary*

Access Memory (RAM) into the core-dedicated caches of a multicore computing system can be made extremely sporadic by performing most of the look-ups within the cache, therefore minimizing the number of fetches being necessary from the RAM. For the reader's convenience, an MA flowchart is sketched in in figure 1.6.

1.1.4 Algorithm Segmentation tricks

As previously stated, *Algorithm Segmentation* is the process of breaking down a single functional block $f(...)$ into its constituent functional sub-blocks or *segments*. This process just identifies the segments within a given block $f(...)$ without actually performing any re-design of the segmented algorithm, so that the computational cost of the segmented system $f(...)$ is not changed. A *segment* is any sub-system of $f(...)$ with a specific and identifiable MIDS over one or more input connections. The output yielded by the processing of such MIDS (the segment output) is in its turn input to another segment (with another, possibly different MIDS) which concurs to build up $f(...)$ as a whole. The gain of the process of segmentation lies just in the difference between the cardinality C_i of the overall MIDS of $f(...)$, and those of the constituent segments, $C_{i,m}$. The MIDS of the segments are often (much) smaller than that of $f(...)$. Typically, the overall MIDS has a size that is given by the *Least Common Multiple* of the segments' MIDS. Therefore, when considering (1.1), it turns out immediately that the set of tabular implementations $t_m(...)$ of the segments is dramatically much less demanding than the (global) table $t(...)$ of the whole subsystem, in terms of memory resources.

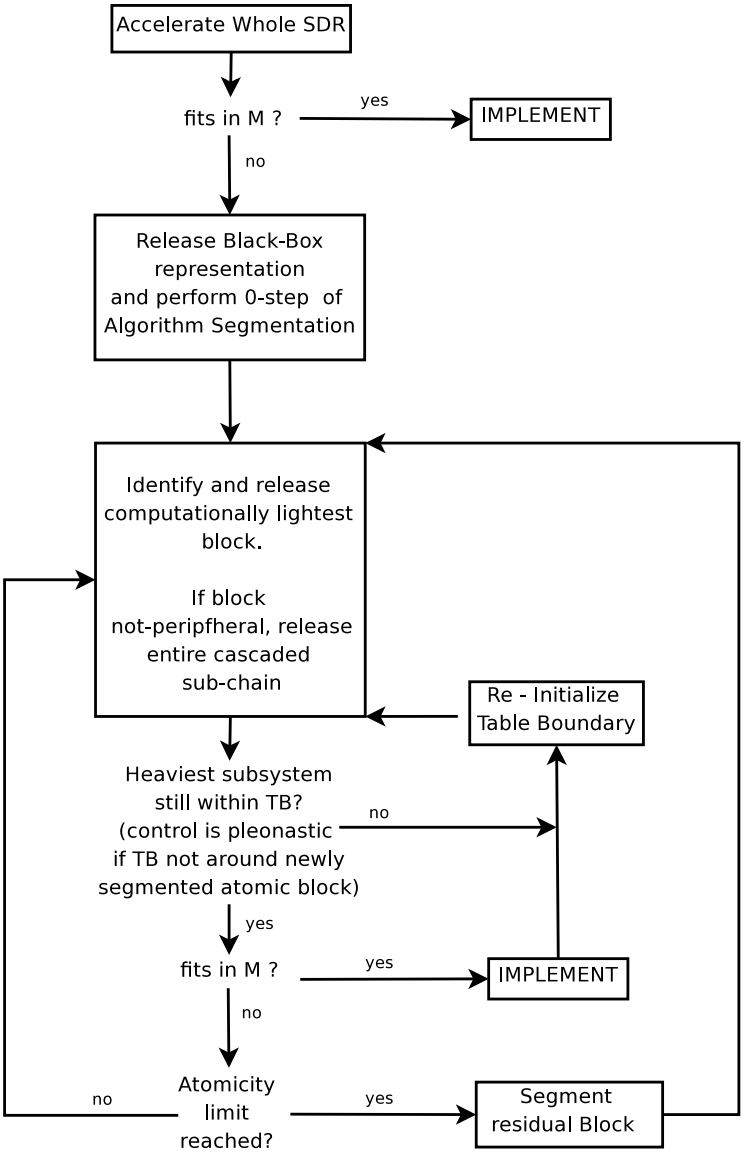

Figure 1.6: *Schematic representation of MA Recursive Table Aggregation Rule. Exit condition on memory exhaustion is not graphically represented for readability*

Considering the extreme variety of architectures and functions of the signal processing algorithms in an SDR terminal, trying to find an optimal segmentation/aggregation configuration is very hard. We can just say that the best algorithm segmentation is the one providing the smallest *granularity of input spaces* of the obtained sub blocks. This is true because the smaller the granularity is, the closer the RTAR will manage to bring the total memory occupancy of the MA-ed SDR to the memory capacity of the computing platform M. Broadly speaking, the more sub-blocks N_{sb} algorithm segmentation obtains from the given block, the better algorithm segmentation was performed.

To sum up, the joint action of algorithm segmentation and RTAR (i.e., the gist of the MA concept) is:

- decomposing the given SDR system down to the finest possible level of computational granularity;
- generating a re-implementation which uses the available memory resources in order to perform as much computation as possible by means of memory lookups;
- doing it with the smallest possible computational cost of memory management

1.2 MA application and performance evaluation

1.2.1 MA Compatibility with different acceleration techniques

All performance results that will be presented were obtained by applying MA alone, i.e. by making use of no other performance enhancement technique such as low level (Assembler) programming or code parallelization. Still, such implementation techniques are fully compatible with MA. Compatibility with low level programming is trivial and does not deserve any discussion. For parallel implementation instead, a possible issue lies in the concurrent access to a certain memory area from multiple computing cores. This may call for some form of collision control and consequent performance bottlenecks. Such problem can be substantially mitigated through the "cache friendliness" aproach that we mentioned above. Multicore/multiprocessor GPP-based platforms often feature cache memories which are dedicated to each single core as depicted in figure 1.7. Such memories are independently accessed by each of

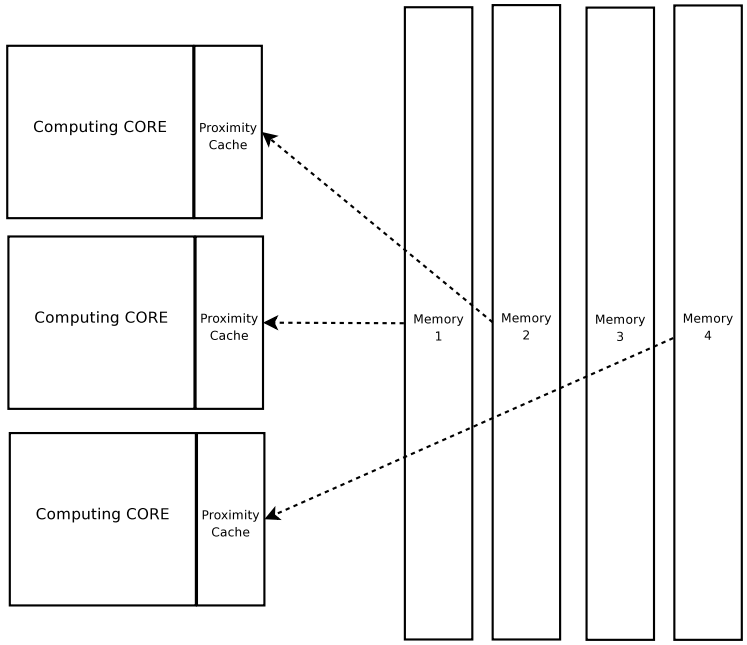

Figure 1.7: *Multiple processing cores, their dedicated caches and table loading from RAM to the core-dedicated cache*

the cores, so that the probability of collision events is zero. Collisions may instead happen when loading the required memory table (or table portion) from the external RAM into the core-dedicated caches as shown in figure 1.7. Appropriate use of a cache-friendly table structure will make these fetches extremely sporadic. Should access contention happen at the RAM-level, it would be rare enough not to degrade system performance. Practical proof of this is provided in [16].

1.2.2 Performance Evaluation

To quantify the performance in terms of processing speed-up of the MA technique, we define the *acceleration efficiency* for any functional block $f(...)$ as

$$\eta = \frac{\sum_{n=0}^{N_{sb}-1} W_n - \sum_{m=0}^{N_t-1} \Omega_m}{\sum_{m=0}^{N_t-1} M_m} \tag{1.2}$$

where N_{sb} is the number of the sub-blocks $f_n(...)$ obtained after algorithm segmentation which are implemented in tabular form, and N_t is the number of tables that will be used to produce such an implementation as resulting from the application of RTAR. In other words, the acceleration efficiency η is the ratio between the computational effort being saved by means of the resulting memory-based implementations (reduced by the amount of computational work needed for table management) and the total memory footprint being required. A negative value for η indicates that the chosen MA design will *reduce* system performance. Once the acceleration process is completed, it is possible to calculate the obtained *acceleration factor* a as

$$a = \frac{W_r + \sum_{n=0}^{N_{sb}-1} W_n}{W_r + \sum_{m=0}^{N_t-1} \Omega_m} = \frac{1 + \sum_{n=0}^{N_{sb}-1} W_n/W_r}{1 + \sum_{m=0}^{N_t-1} \Omega_m/W_r} \tag{1.3}$$

where W_r accounts for the total computational cost of the remaining blocks which where not implemented in memory.

As previously stated, different algorithms offer different opportunities for segmentation. The consequence of this statement is that it is very difficult to give an upper bound for a. Still, a naive and probably loose upper bound for a can be found assuming that the whole radio can be memory-accelerated. In such condition we get

$$a_{max} = \frac{\sum_{n=0}^{N_{sb}-1} W_n}{\sum_{m=0}^{N_t-1} \Omega_m} \leq \frac{\sum_{n=0}^{N_{sb}-1} W_n}{\sum_{m=0}^{N_t-1} [L_m + (i_m - 1)(x + \sigma)]} \tag{1.4}$$

where N_{sb} is now the total number of computational blocks within the segmented system, L_m is the access latency for each table (that depends on the table size and on the chosen implementation platform), i_m is the number of inputs to each table, x is the computational cost for one multiplication by a constant, and σ is the computational cost of one sum with a variable. All such quantities, including L_m, can be expressed in terms of number of equivalent elementary operations, or of required CPU time. The term $(i_m - 1)(x + \sigma)$ is a lower bound for Ω_m that only considers the simplest elementary computation of the memory address.

Chapter 2

Application of MA in real-world SDR systems

In this chapter we detail the application of MA technique described in the previous chapter in a real-world SDR system, a fully-software ETSI DVB-T demodulator running in real-time on a GPP-based platform.

2.1 SR-DVB, a SDR receiver for standard ETSI DVB-T signals

Digital video broadcasting (DVB) standard in its terrestrial version (DVB-T) [15] is the most widely deployed system for standard and high definition digital video delivery to home users worldwide.

DVB-T receivers are commonly based on Application Specific Integrated Circuits (ASICs) implementing synchronization, channel estimation/equalization and demodulation functions which allow the extraction of a Motion Picture Experts Group 2 (MPEG2) Transport Stream (TS) from a Coded Orthogonal Frequency Division Multiplexing (COFDM) radio-frequency (RF) signal.

In such fully-hardware architectures, software applications are restricted to the implementation of some complementary or ancillary functions such as receiver setup, performance monitoring or Electronic Program Guide (EPG).

A fully-software implementation of a DVB-T receiver is generally not considered feasible on a reasonably-priced GPP because of its computational complexity. On the other hand, a fully-software solution might bring some benefits compared to its HW

counterpart in terms of:

- lower development cost;
- quicker time to market;
- greater portability of receiver IP;
- easy upgrade to further DVB-T standard evolution;
- availability of a fully controllable, completely monitored receiver chain intended for signal development and testing on the bench and on the field.

In this chapter we describe the implementation of a proof-of-concept, fully-software receiver for COFDM, ETSI DVB-T signals named SR-DVB, which is based on the well-known Universal Software Radio Peripheral (USRP) HW. Specifically, the USRP provides the RF-to-baseband conversion as well as analogue to digital conversion. All baseband receiver functions are implemented through an efficient C++ code that was developed from scratch at the University of Pisa, Digital Signal Processing for Communication Laboratory (DSPCoLa).

SR-DVB pairs with the fully-software DVB-T modulator *Soft-DVB* [17], the first SDR system in which the MA technique was largely adopted.

2.1.1 The ETSI DVB-T receiver chain

The ETSI DVB-T receiver shown in Fig. 2.1 allows an MPEG2 TS to be extracted from a COFDM signal. The receiver can be seen as the result of the concatenation of two signal processing chains: the synchronization and channel estimation/equalization chain and the channel decoding subsystem, developed in [18] and [19] respectively.

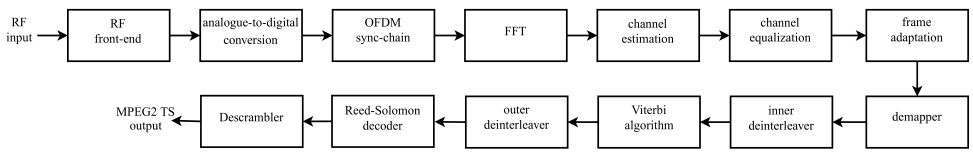

Figure 2.1: *Functional block scheme for the ETSI DVB-T receiver chain*

Implementation quality of the synchronization and channel estimation/equalization blocks has major impact over the sensitivity and the goodness of the receiver. Blocks within channel decoding subsystem, which account for the majority of the computational cost, implement functions ranging from frame adaptation to TS extraction as depicted in Fig. 2.1.

2.1.2 SR-DVB architecture

SR-DVB is a fully-software DVB-T receiver. Therefore HW section consists only of the front-end whereas all the signal processing functions are implemented through SW blocks.

As its modulator counterpart named *Soft-DVB* [17], [20], SR-DVB demodulator lives within an SDR framework called *newRADIO* [20]. Such framework is conceived in order to remove any level of abstraction which is not strictly necessary. The rationale behind this choice is that the complexity within an SDR is always much more computational than logical. Therefore, the switch from an HW implementation to a SW one, though obtained by means of a very basic object abstraction level, provides enough generality and flexibility to the entire SDR system. Development effort is instead needed in minimizing computational overhead as well as in finding practical ways to relieve computing cores from their huge burden in any possible way. Differently from what GNURadio does, within *newRADIO*, C++ is responsible for both functional block implementation and block interconnection. Communication towards and from USRP is obtained by linking to the *libusrp* external C++ library. Synthetically enough, newRADIO delivers us:

- flow control;
- Within-Block and Extra-Block multi-threading with thread synchronization capabilities;
- single programming language (C++) implementation.

The peripheral used to capture the signal in SR-DVB is the USRP system, which was developed by Matt Ettus [21] and was universally adopted by the GNURadio community as its SDR hardware. USRP is an Universal Serial Bus (USB) based board with open design and drivers. It consists of:

- four Analogue-to-Digital Converters (ADC) 64 MSamples/s at a resolution of 12 bit on the receiving side;
- four digital-to-analogue converters (DAC) 128 MSamples/s at a resolution of 14 bit on the transmitting side;
- a Cypress EZ-USB FX2 High-speed USB 2.0 controller;
- 4 extension sockets (2 TX, 2 RX) to connect two to four daughterboards.

Daughterboards are responsible for RF downconversion. In order to cover as many frequency bands as possible, several daughterboards were developed: receivers, transmitters or transceivers do exist which collectively cover the RF spectrum from DC to about 5 GHz. The RF front-end used to acquire the signal in SR-DVB is the transceiver daughterboard RFX900 which works form 800 MHz to 1 GHz. As shown in next sections, the acquired signal has a baud rate of 8 complex Msamples/ s on a 7 MHz DVB-T channel bandwidth and cyclic prefix length of 1/ 4. It is possible to receive this signal at exact Nyquist frequency by sampling it at 8 complex Msamples/ s which is the maximum possible rate over the USB interface of the USRP board. Complex samples are sent over USB interface by interleaving real and imaginary parts, both represented as a signed *short int* on 2 bytes. The resulting rate on USB is 32 MB/ s.

2.1.3 SR-DVB implementation

Each functional block of SR-DVB, except the Fast Fourier Transform (FFT) block which is based on FFTW [22], was implemented from scratch in order to have full control of available system resources and to reach real-time performance.

2.1.3.1 Synchronization chain and channel estimation/equalization functions

Fig. 2.2 focuses on the first part of SR-DVB chain, developed in [18] and described as follows:

- the *RF front-end* shifts the signal from the chosen TV RF channel to baseband;
- the *analogue-to-digital conversion* samples the I/Q (In Phase/Quadrature) signal and provides the interleaved samples to the SW signal processing chain;

- the *OFDM synchronization chain* estimates and corrects the timing and frequency offsets, detects the first available OFDM frame (68 OFDM symbols) by demodulating the Transmission Parameters Signalling (TPS) subcarriers and removes the cyclic prefix;

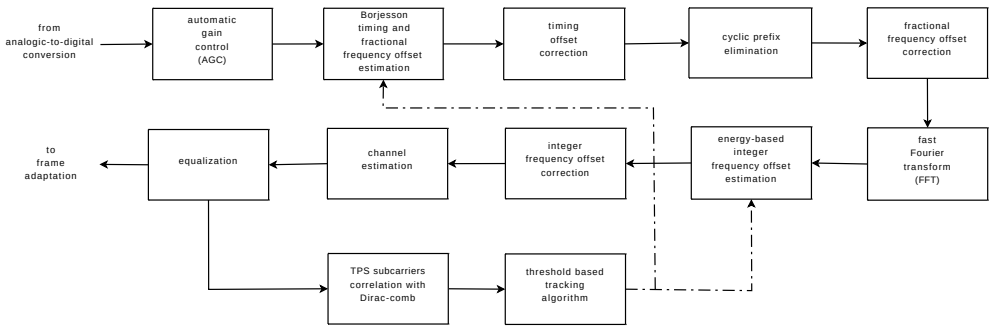

Figure 2.2: *OFDM synchronization and channel estimation/equalization chain in SR-DVB*

- the *Fast Fourier Transform*, with 2048 (2k) or 8192 (8k) points, acts as a matched filter for the OFDM modulation;
- the *channel estimation* estimates channel frequency response by interpolating information carried by boosted pilot subcarriers (both continual and scattered);
- the *channel equalization* uses the calculated channel profile and the *Zero-Forcing* (ZF) technique to equalize the data subcarriers.

The algorithm used for the estimation of timing and *fractional* frequency offsets (i.e. as a fraction of intercarrier spacing) is based upon the algorithm described in [23]. Such algorithm works within the time domain and implements a Maximum Likelihood (ML) open-loop timing and fractional frequency offset estimation by exploiting the inner redundancy contained in OFDM cyclic prefix. In SR-DVB we adopt a modified version of the algorithm using averaged realizations of the log-likelihood function. FFT is performed after the time domain correction of the estimated offsets. The residual *integer* (i.e. as a multiple of intercarrier spacing) frequency offset is estimated with an energy-based, frequency-domain open-loop algorithm as described in [24]. Actually, a mobile window, whose size is as wide as the bandwidth occupied by the active subcarriers (6817 in 8k mode, 1705 in 2k mode) slides over the entire OFDM

symbol calculating, for each position, the energy of the selected part. The difference between the position in which the energy is the highest and the theoretical position of maximum energy (the center of the OFDM symbol) represents the estimation of the integer frequency offset. A tracking algorithm based on a finite-state machine controls the renewal of timing and frequency offset estimations. A combination of the post-equalization TPS subcarriers provides the metric used to trigger such renewal process.

2.1.3.2 Channel decoding subsystem

The second part of SR-DVB chain is the channel decoding subsystem. It was developed in [19] and is shortly described as follows:

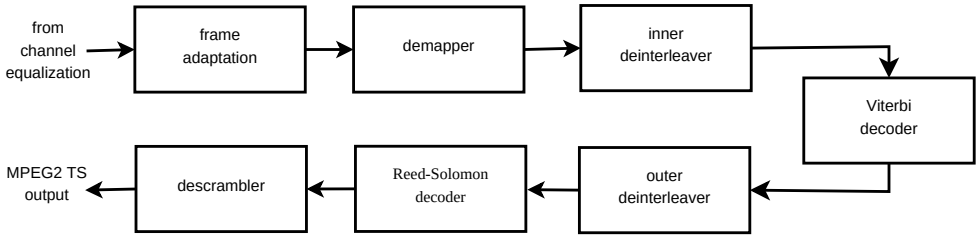

Figure 2.3: *Channel decoding subsystem in SR-DVB*

- the *frame adaptation* performs removal of reference signals, i.e. TPS, continual and scattered pilots as well as virtual subcarriers;
- the *demapper* demodulates the received symbol constellation (QPSK, 16-QAM or 64-QAM) into an encoded bit stream according to the used constellation;
- the *inner deinterleaver* performs a bit-level operation in order to recover the right bit order after shuffling introduced by the inner interleaver on the TX side;
- the *Viterbi algorithm* is the most famous method to decode a convolutional code. Though providing ML decoding, it requires a huge amount of computation which makes it the computationally-heaviest block within the receiver chain having to work at a bitrate of 12.600 Mbps (measured after decoding);

- the *outer deinterleaver* performs a byte-level operation in order to minimize the correlation between data and to scatter possible error bursts at Viterbi decoder output;
- the *Reed-Solomon (RS) decoder* performs the syndrome-based decoding of the RS block-coded data bytes. This mitigates the impact of the error bursts occasionally produced by Viterbi algorithm;
- the *descrambler* derandomizes data stream by applying a bit-wise XOR with a Pseudo Random Binary Sequence (PRBS) and provides input to the MPEG2 decoder.

2.1.4 MA within SR-DVB

Real-time performance achieved while implementing SR-DVB, as described in section 2.1.5, would have never been possible without resorting to the extensive usage of MA. As said in the previous chapter MA technique, which mainly targets GPP based SDRs but would also fit DSP-based systems, is based upon the observation that, on GPP-based machines, memory resources are usually abundant, cheap, and not power-hungry if compared to computing cores.

Within SR-DVB work, MA was applied to the two computationally-heaviest functional blocks of the receiving chain: the ETSI DVB-T, K=7 Viterbi decoder and the OFDM-specific time and fractional frequency offset estimation algorithm described in [23]. For both algorithms, the acceleration factor (with respect to a previous, MA-free but computationally optimized version of such algorithms) obtained is greater than one order of magnitude. We believe that a considerable improvement margin for our MA implementations of such two algorithms still exists and can be the object of further research.

2.1.5 Experimental results

SR-DVB was tested and validated at DSPCoLa, University of Pisa, Italy both for 2k and 8k mode. Signal captured by the antenna is sent via a 50 Ω coaxial cable to the USRP front-end and then (after undergoing just baseband conversion and sampling) via USB 2.0 to an off-the-shelf Personal Computer (PC). As stated in section 2.1.3,

Figure 2.4: *Output of MPEG2 GNOME player of the signal received with SR-DVB*

all the signal processing is performed by the host PC which is equipped with an Intel Q9400 processor (2.66 GHz) and Fedora 12 Operating System (OS).

For the 2k mode, signal transmitted by *Soft-DVB*, the real-time, fully-software modulator presented at WSR08 in Karlsruhe [17], was used to validate the receiver chain. SR-DVB proved able to correctly demodulate the provided transmission in *real-time* while absorbing less than 50% of computational resources available aboard the host PC. Soft-DVB test signal has a baud rate of 8 MSamples/ s on a 7 MHz DVB-T channel bandwidth, cyclic prefix length of 1/4, 16-QAM constellation, 2/3 convolutional coding rate, all yielding a useful bitrate of 11.612 Mbps.

For 8k mode, SR-DVB was validated by receiving the public broadcast signal radiated by the antennas placed at the top of Monte Serra, 13 Km North-East of Pisa. Signal was captured on channel 56 (Italian channelization) with central frequency 754 MHz. Such transmission has a baud rate of 64/ 7 complex Msamples/ s on an 8 MHz DVB-T channel bandwidth, cyclic prefix length of 1/ 32, 64-QAM constellation, 2/3 convolutional coding rate, all yielding a useful bitrate of 24.13 Mbps. Because of the maximum sampling rate of USRP (8 MSamples/s), it was necessary to perform a

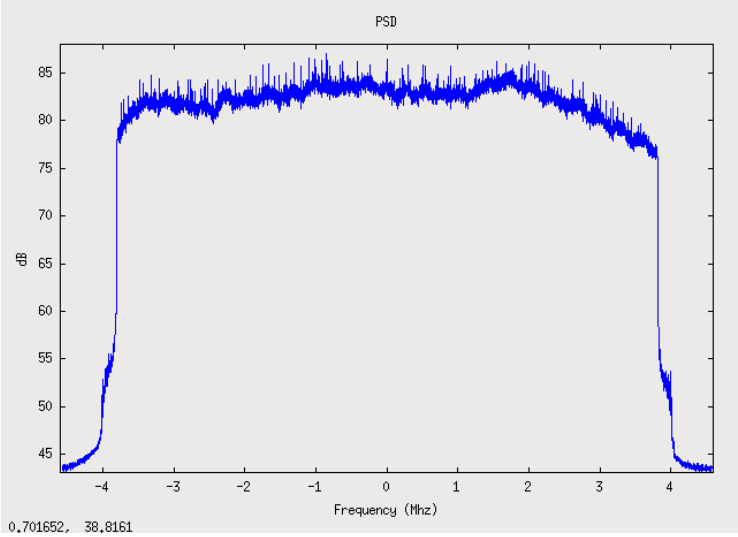

Figure 2.5: *Spectrum of the signal at the input of SR-DVB*

software rational resampling with factor 8/ 7 at the input of SR-DVB. Fig. 2.5 shows the spectrum of the captured DVB-T signal. 8k signal was correctly demodulated and Fig. 2.4 shows one channel from the output MPEG2 TS as played by *mplayer* application. Fig. 2.4 belongs to the transmission of program 155 with PID 255 as shown by the TS analysis presented in Fig. 2.6.

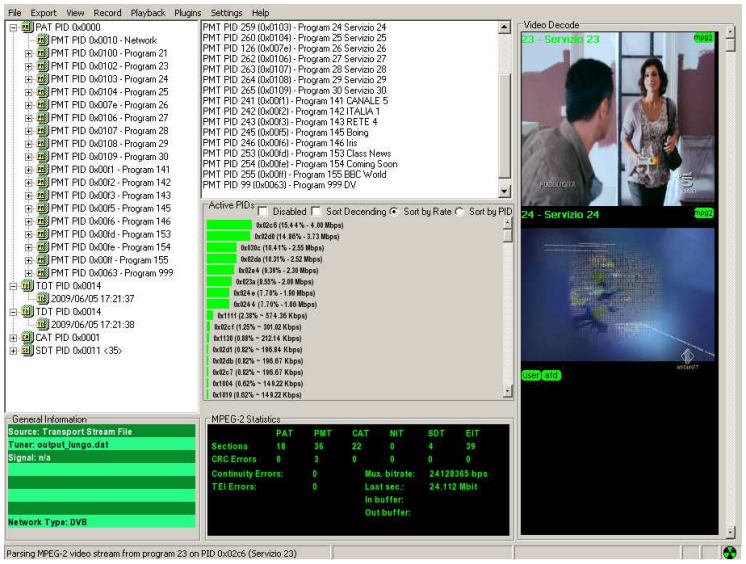

Figure 2.6: *TS analysis of the signal received with SR-DVB*

Chapter 3

Techniques for improving interoperability in SDR systems

In this chapter we analyse some programming techniques and software architectures formalized to increase the portability of the source code and the interoperability among heterogeneous SDR platforms. In particular, we describe the SCA and one of his open-source implementation, i.e. OSSIE, focusing on the research activities performed in order to provide multi-threading support and cpu affinity capabilities to OSSIE SDR framework.

3.1 The Software Communication Architecture

3.1.1 SCA historical notes and JTRS philosophy

In 1997 the United States Department of Defense (DoD) initiated the JTRS program, as a way to try and solve the issue of a programmable, modular, multi-band, multi-mode radio, that would eventually replace over 200 different radio types within the DoD. In late 1998, the first step towards the SCA was taken. The SCA is a non-proprietary, open architecture framework, to help promote the development of interoperable software and hardware. The SCA is not a system specification, as it is intended to be implementation independent, but rather a set of design constraints. If a developer designs a system according to the these design rules, his system will

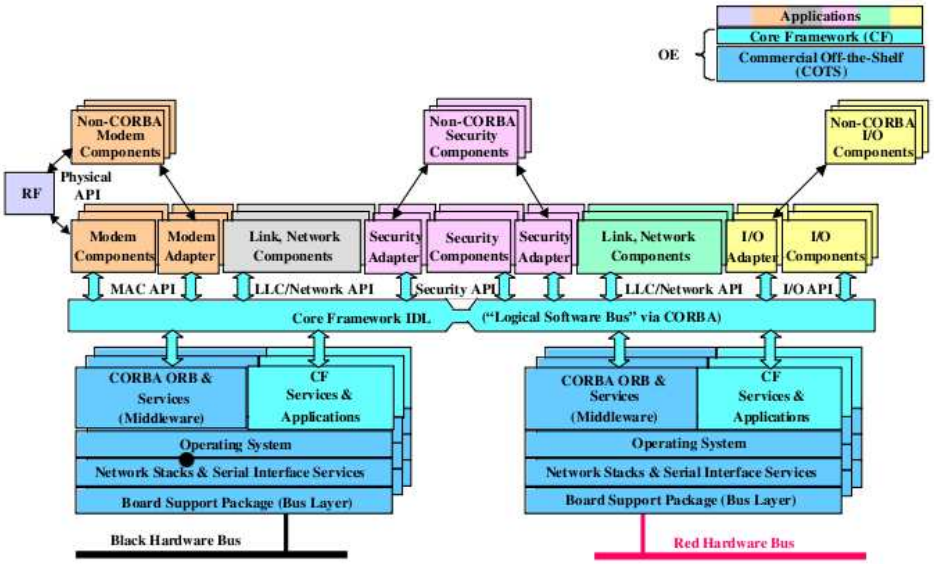

Figure 3.1: *SCA Software Architecture*

be portable with other SCA implementations regardless of what operating system or hardware that implementation is based on. The software structure of the SCA is called the Operating Environment, and consists of the following components:

- a Core Framework (CF). This is the collection of interfaces and services that provide an abstraction of the underlying layers for software application designers. The interfaces are described using CORBA IDL, which will be described in the next section;

- a CORBA middleware that is used for all communications within the software components, called waveforms. A middleware can be described as the glue between software components or between software and the network;

- a POSIX-compliant operating system.

Figure 3.1 shows the general SCA Software Architecture.

3.1.2 The CORBA middleware

CORBA is the acronym for Common Object Request Broker Architecture, and is a standard for producing client/server middleware in a distributed environment. The CORBA standard is created and controlled by the Object Management Group (OMG). The CORBA mechanism allows programs to be running on different machines and written in different programming languages while safely (and portably) exchanging data so that the CORBA mechanism is ideal for classic client/server applications. The Object Request Broker (ORB) is the piece of software running on a machine that handles all server/client requests. The protocol used by CORBA to communicate is the General Inter-ORB Protocol (GIOP). The GIOP maps ORB requests to different network transports, and one such implementation is the Internet Inter-ORB Protocol (IIOP). An important issue for object interoperability is how ORBs address, or locate, objects. An object reference can be thought of as a trustworthy name that always symbolizes a specific object. These references are standardized for all ORBs. Thus, any ORB can invoke operations and calls to objects located with other ORBs. The method used by CORBA to handle this is the Interoperable Object Reference (IOR), which is a sequence of characters that specifies a single CORBA object wherever in the world it is located.

3.1.2.1 Interface Definition Language

CORBA uses OMGs Interface Definition Language (IDL) to specify the interfaces that objects will present to the world. CORBA then specifies a mapping from IDL to a specific implementation language like C++ or Java. Any client that wants to invoke an operation on a CORBA server object, must use the objects IDL interface to specify which operation it wants to perform. It is worth mentioning that IDL is not a programming language - its great for defining interfaces, but it doesnt have the constructs youd need to write a program. The IDL interface defines the contract between the client and server parts of your application, specifying what operations and attributes are available. The programmer then uses an IDL compiler to generate application code, skeletons for the server part, and stubs for the client part. The stubs and skeletons run on top of an ORB, and they work as proxies for servers and clients, respectively. Thanks to this method, the client and server can be written in different languages and/or be running on different platforms and be able to communicate with

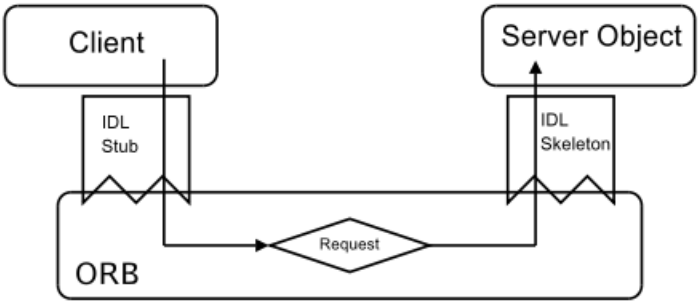

Figure 3.2: *Client-server model in CORBA using a local ORB*

each other in a safe, scalable manner. Every CORBA server object has a unique object reference, and there are several ways in which the client can get a hold of this reference. Once obtained, the client can invoke operations on the server object. The invocation is really forwarded to the client stub, which uses the ORB to forward the request to the servant object, through the server skeleton. Figure 3.2 shows a model of this.

The ORB block contains the necessary methods to pass over the request from client to server. Usually the client and server are not connected to the same ORB, and the GIOP/IIOP protocol is used to forward the request. This process is described in figure 3.3 . This operation is completely transparent from the client and server view. The client needs to know only the reference to the servant, and then the ORBs handle the technical stuff such as load balancing, resource control and error handling of the requests.

A simple example of IDL is shown below. The interface Modulator has one operation called ModulateData which takes one input argument and returns nothing (void), and has one attribute called ModulatorStatus. It also inherits the Resource and Port interfaces from the Core Framework.

```
interface Modulator : CF::Resource, CF::Port{
        void ModulateData(in double incoming_data);
        attribute int ModulatorStatus;
}
```

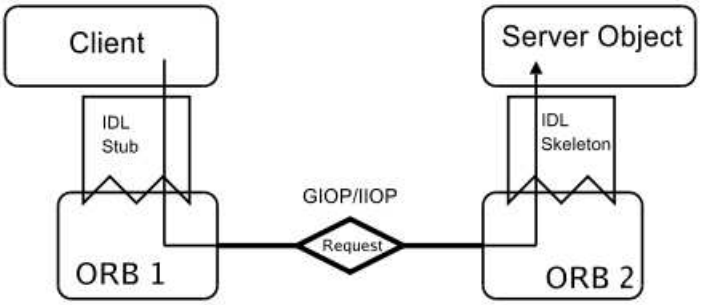

Figure 3.3: *Client-server model in CORBA using GIOP/IIOP*

There are a lot of ORB implementations, TAO ORB is one of the most used. TAO is a real-time ORB based on the SunSoft IIOP protocol engine. TAO is targeted for applications with deterministic and statistical QoS requirements, as well as best effort requirements.

3.1.2.2 Object Adapter

The Portable Object Adapter (POA) is a way of making implementation objects available to the ORB for servicing requests. All CORBA calls on a CORBA object goes through the POA. The POA maps a CORBA object ID to the actual implementation object. Upon a server object initialization, it registers itself with the POA.

3.1.2.3 Naming Service

The OMG Naming Service is one of CORBAs standardized services. The Naming Service provides the principal mechanism used by clients of an ORB-based system to locate objects that they intend to use. The basic function of the naming service is the association of names with object references. A server object creates associations between a name and its object reference, and registers this information in the Naming Service. Then a client that knows the name of an object can retrieve its object reference by querying the Naming Service.

3.1.3 SCA architecture

As said before, SCA is designed to be an open, standardized architecture providing interoperability, quick upgrade capability, software reuse and scalability. When designing the structure of a SCA compliant system, all these factors should be kept in mind. A SCA waveform or Application consists of one or more software components, called Resources, using heterogenous computational hardware assets, called Devices. A Device is a type of Resource used by applications as software proxies for actual hardware component. The Application is created in an Application Factory module, usually by the DomainManager. Every component in an SCA compliant waveform inherits the interfaces from the Core Framework. These interfaces are described in CORBA IDL language, and then compiled into the preferred programming language. A central role in the SCA architecture is given to the XML language. In fact, the Domain Profile is a set of XML files used to describe the characteristics of the system, the different interfaces it is composed of and their functional capabilities and interdependencies. It is managed by the Domain Manager module. Legacy software components that dont have CORBA support can be incorporated in an SCA system by the use of CORBA adapters, which are proxies to wrap the functionality of the component in the CORBA environment. Figure 3.1 shows the relationship between the software component implementing a SCA-compliant waveform at different level of the ISO/OSI protocol stack and the Operating Environment which consists of the Core Framework, the CORBA middleware, and the operating system.

Although the SCA uses the CORBA middleware for its software bus, the application layer can reach the OS by other means. However, waveform access to the OS is highly restricted.

3.1.3.1 Core Framework

As mentioned in section 2.1, the CF is one of the key components in an SCA system. The CF consists of:

- Base Application Interfaces (Port, LifeCycle, TestableObject, PropertySet, PortSupplier, ResourceFactory and Resource) that can be used by all software applications.
- Framework Control Interfaces (Application, ApplicationFactory, DomainMan-

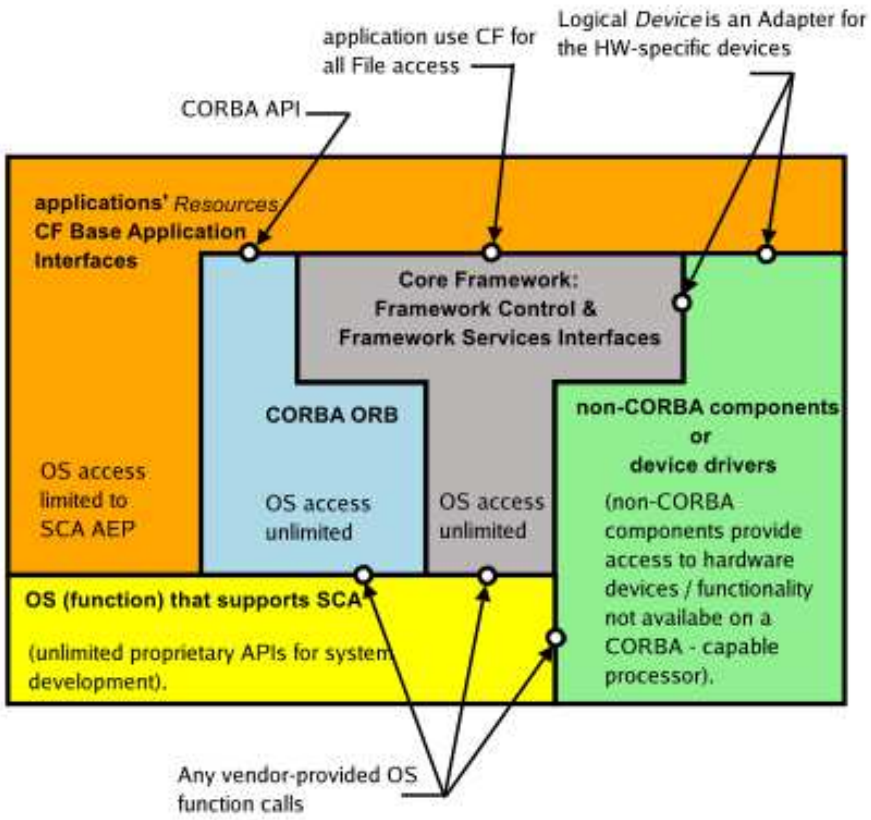

Figure 3.4: *Relationship between structures in SCA environment*

ager, Device, LoadableDevice, ExecutableDevice, AggregateDevice and DeviceManager) used to control the system.

- A Domain Profile describing the properties of hardware devices (Device Profile) and software components (Software Profile) in the system.

3.1.4 SCA and CORBA

The Core Framework interfaces are expressed in CORBA IDL. When implementing a waveform, all components interfaces which constitute the waveform are also described in CORBA IDL. As written in Sec. 3.1.2, IDL is a language that defines interfaces. So basically, an SCA waveform is a bunch of interfaces. When compiling the CORBA IDL, server skeletons and client stubs are created. This means, on a deeper level, that all the waveform components such as Resources and Devices are actually CORBA servants and/or clients. The and/or has to be emphasized. A component can be (usually is) both server and client at the same time. For example take some components, A, B and C. A has an interface that B is connected to. B uses the interface provided by A. This makes A a CORBA servant and B the CORBA client that connects to the servant. On the other hand, C has an interface A is connected to. In this case, C is the server and A the client. Thus, A is both server and client at the same time. The various components of the waveform uses the CF Port interface (remember this is also a CORBA IDL interface) to make connections with each other. The terms uses port and provides port are commonly used in SCA development. A uses port requests data or service from another component, while a provides port returns requested data or performs a requested service. Under this model, software assumes the role of a CORBA client when it is calling through a uses port, and the role of a CORBA servant when it is answering at a provides port.

3.2 SCA and multithreaded applications

The aim of this section is to focus on the use of GPP based multi-core platforms for the implementation of SDR systems. In this way parallel (multithreading) programming technique represents the enabler technology to maximally exploit the computational resources of a multi-core system. In particular, we will focuse on processor affinity method, a very interesting modification of the native scheduling algorithm of the

Operating System. This method makes it possible to determine the set of cores of a multi-core system on which a task (thread) has to be run. The application of this method on SCA-compliant SDR system can be useful in order to improve the load balancing between CPU loads and to allocate the software components of a SCA waveform on specific cores (eg. PHY → core1, MAC → core2, IP → cor3, etc..), thus increasing the control of the system at run-time. This method has to be ported into a SCA-compliant framework like OSSIE, an open-source SCA-compliant SDR framework developed by WirelessGroup@Virginia Tech. In fact, in its most recent version (0.8.2), it does not support the explicit setting of the processor affinity on the software components. In the last part of the chapter we provide the guidelines to modify OSSIE, showing the source code that should be changed and providing the functions and the macros that should be used in order to help the scheduler in allocating a software component on a specific core of a multi-core platform

3.3 Notes on multi-core processor architectures

Until the end of 90's processors were equipped with only one core so that they could process only one instruction at a time. For this reason, processors utilize pipelines in order to process several instructions together even if they are still consumed into the pipeline one at a time. A multi-core processor is a processing system in which it is possible to process several instructions at a time thanks to the presence of two or more independent cores. They are typically integrated in a single integrated circuit die or in multiple dies in a single chip package. Developers generally uses the term multi-core to refer only to multi-core processors which are locate in the same integrated circuit die and refers to separate processors dies as multi-chip module. Besides, the term multi-CPU refers to multiple physically separate processing-units (which often contain special circuitry to facilitate communication between each other). When the number of cores increases (several tens of cores) the classical multi-core techniques are no longer efficient so that is necessary a network on chip. A multi-core processor implements multiprocessor in a single physical package. Designers may couple cores in a multi-core device together tightly or loosely. In fact, cores can share caches and implement inter-core communication methods based on message passing or shared memory method. Network connecting the cores can be based on bus, ring, 2-dimensional mesh, and crossbar. In particular, a multi-core system is called

homogeneous if include only identical cores. Otherwise it is called heterogeneous.

Software algorithms can obtain large gain by the use of a multi-core device, even if this gain is limited by the possibility of parallelizing the software. In the best case, named *embarrassingly parallel*, the speed-up factor can be near the number of cores. Typically this speed-up factor is not reached because algorithm parallelization is often very complex and represents by itself an ongoing current research topic.

3.3.1 Advantages and disadvantages in using multi-core architectures

Thanks to the proximity of multiple cores on the same die the cache coherency circuitry can operate at higher clock-rate and the performance of cache/bus snooping are improved. Finally, signals between different cores travel shorter distances and degrade less so that more data are sent, individual signals are shorter and do not need to be repeated as often. In order to maximize the utilization of the resources provided by a multi-core processor both the operating system and the application software require adjustments. Furthermore, the use of multiple threads within an application has a great impact on the increase of performance on a multi-core system. In fact, bus and memory bandwidth can result in a limiting factor. If a single core is close to being memory-bandwidth limited, going to dual-core might only give 30% to 70% improvement. If memory bandwidth is not a problem, a 90% improvement can be expected. On the server side, multi-core processors are perfect because they allow many users to connect to a site simultaneously and have independent threads of execution. So Web servers and application servers can have much better throughput.

3.4 Impact of multithreading programming techniques in software development

Multithreaded programming often requires a very difficult thread synchronization and can easily introduce bugs due to the interleaving of processing on data shared between threads (thread-safety). For this reason, a multithreaded code is generally more difficult to debug than single-threaded one. On he other hand, parallel programming techniques directly benefit from multi-core architecture. For these reason several parallel programming models, such as OpenMP, Boost Threads, POSIX threads and

Intel Thread Building Block, exist and can be used on multi-core architectures. The most difficult aspect of the parallel application is managing concurrency. For this reason, parallel software development is usually divided in several steps:

- **Partitioning.** In this step the initial problem is divided in smaller tasks and then are checked the opportunities for parallel execution.
- **Communication.** In this step is designed the communication flowgraph between the concurrent threads. In particular, it focuses on thread-safety, looking at the data resources shared among the concurrent threads.
- **Agglomeration.** In this step, the small tasks are re-aggregated in order to avoid the excessive segmentation of the presence of replicated data that makes it increasing the overall computational load.
- **Mapping.** In this step each task is assigned to a computational resource (core, CPU, DSP, etc..)

3.5 OSSIE framework and processor affinity method

OSSIE is an open source SDR framework developed at Virginia Tech. The aim of OSSIE is to provide a valid tool to research and education in SDR systems. The SDR core framework of OSSIE is based on JTRS SCA so that it represents also a good validation platform to build and test SCA-compliant waveforms (Fig. 3.5).

The software package includes:

- an SDR core framework based on SCA;
- the Waveform Workshop, a set of tools for rapid development of SDR components and waveforms applications;
- an evolving library of pre-built components and waveform applications.

The framework is conceived to provide a valid development environment and remove the complexity related to the implementation of the SCA hierarchical software architecture. Within *OSSIE*,unlike other SDR frameworks like GNURadio, C++ is responsible for the single functional block implementations, while the interconnection among such blocks is provided, as stated by the SCA standard [3], by means of

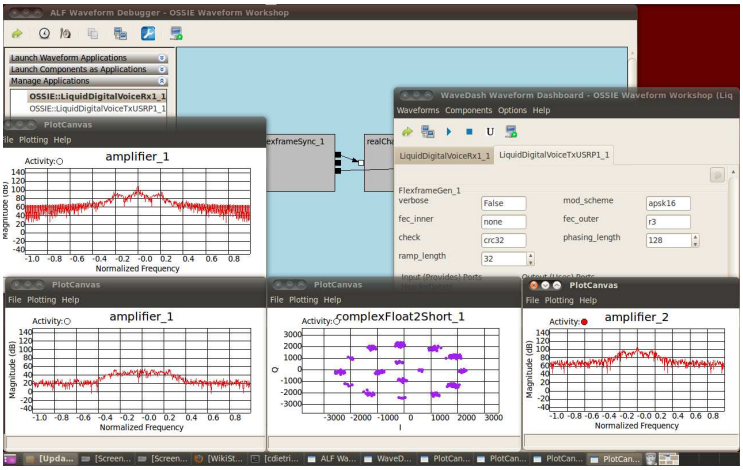

Figure 3.5: *OSSIE SDR development environment*

CORBA interfaces and *ports* communication. Communication from/to SDR platform, like the Ettus USRPs [21], is obtained by including the *libusrp* external C++ library in a suited *software device* abstracting HW platform. Furthermore, a certain amount of ad-hoc modifications to the OSSIE primitives that manage communications from/to the USRPs were also introduced to exploit the full functionalities of the HW platforms.

3.5.1 Processor affinity and affinity mask

Processor affinity is a method to modify the native scheduling algorithm. In fact, each task has a label showing its preferred (kin) processor so that, at allocation time, each scheduler attempts to assign the task to its kin processor in preference to others. This approach can lead to significant increase in software performance because some remnants of a process may remain in one processor's state (in particular, in its cache) from the last time the process ran, and so scheduling it to run on the same processor the next time could result in the process running more efficiently. The different level of adherence to processor affinity characterizes the particular scheduling algorithm implementation. In fact, under certain conditions, some scheduling algorithms allow a task to change to another processor in order to increase the overall efficiency of the system, eg. two processor-intensive tasks (X & Y) having affinity to one processor while another processor lies unused. In this case the affinity of one of the two tasks

has to be changed to have affinity with the second processor while the other continues to have affinity with the original processor. Processor affinity technique can reduce the cache problems while it cannot resolve a persistent load-balancing problem. In system with non-uniform architecture processor affinity becomes more complicated. For example, in a system with dual-core hyper-threaded CPUs, there is complete affinity between two virtual CPUs implemented via hyper-threading on the same core; partial affinity between two cores on the same physical chip (core can share some, but not all, cache) and no affinity between separate physical chips.

It worth noting that processor affinity alone cannot represent a complete scheduling algorithm. For example, if a process has affinity with one virtual hyper-threaded CPU in a given core, and that virtual CPU is currently busy, cache affinity would suggest allocating the process on the idle virtual CPU partner but, since the two virtual CPUs share all computing, cache, and memory resources, it typically results more efficient to assign the process to a different core or CPU if one is available. In fact, even if the process loses its cache affinity, the overall performance is higher because the process does not have to compete for resources such as functional units within the CPU, memory, etc..

In Unix the processor affinity can be modified by changing the affinity mask, a bit mask indicating the set of processors in which a process or a thread is eligible to run.

Thus, a process's CPU affinity mask determines the set of cores on which a task can be run. Setting the core affinity mask can allow performance benefits on a multi-core system. In fact, setting the affinity mask of a process to specify a single core, and setting the affinity mask of all other processes to exclude that core ensures the maximum execution speed for that process. This approach also eliminates the performance cost caused by the cache invalidation, occurring when the execution of a process is moved from one core to another one. The set of functions and macros that are used to set the processor affinity are listed below:

- #include <sched.h>
- void CPU_CLR(int cpu, cpu_set_t *set);
- int CPU_ISSET(int cpu, cpu_set_t *set);
- void CPU_SET(int cpu, cpu_set_t *set);
- void CPU_ZERO(cpu_set_t *set);

- int sched_getaffinity(pid_t pid, unsigned int cpusetsize, cpu_set_t *mask);
- int sched_setaffinity(pid_t pid, unsigned int cpusetsize, cpu_set_t *mask);

cpu_set_t structure represents the core affinity mask and is pointed to by *mask*. In order to manipulate the CPU sets four macros are provided:

- *CPU_ZERO()*. It clears a set;
- *CPU_SET()* and *CPU_CLR()*. They respectively add and remove a given core from a set;
- *CPU_ISSET()*. It checks if a core is part of the given set. This macros is useful after the results of *sched_getaffinity()*.

To the first available core on the system is assigned the *cpu* value of 0, *cpu* value of 1 is assigned to the next and so on. The constant *CPU_SETSIZE (1024)* specifies a value one greater than the maximum core number that can be stored in a CPU set.
sched_getaffinity() writes the affinity mask of the process with ID *pid* into the *cpu_set_t* pointed to by *mask*. The *cpusetsize* argument specifies the size (in bytes) of *mask*. Note that a child process created via *fork()* method inherits its parent's core affinity *mask*. The value returned from a call to *gettid()* can be passed in the argument *pid*. If *pid* is zero, then the mask of the calling process is returned.
sched_setaffinity() sets the core affinity mask of the process with ID *pid* to the value specified by *mask*. If *pid* is zero, then the calling process is used. The argument *cpusetsize* is the length (in bytes) of *mask*. This argument can be specified as *sizeof(cpu_set_t)*. If the process specified by *pid* is not currently running on one of the cores specified in *mask* the process is migrated to one of the cores specified in *mask*.

3.5.2 Modifications on OSSIE framework

Low-cost, flexibility and easy possibility of reconfiguration make GPP multi-core processors optimum candidates for the development of demonstrators of SCA-compliant waveform applications. The different abstraction layers of a wireless system (PHY, MAC, IP,...) are often implemented in a SCA-compliant waveform as different software components that communicates through the concept of *port interfaces*. The

real-time constraint implies the utilization of programming method, such as parallel programming, multithreading and processor affinity to increase the overall efficiency and performance of the system. In particular, the possibility of setting the processor affinity gives also major control on the computational resources of the system. This feature can be useful to demonstrate, for example, the deterministic behaviour of the code, eg. for safety certification in avionics application. Unfortunately the most recent version of OSSIE framework (0.8.2) does not support the allocation of a software component on a specific core of a multi-core platform. The access to a GPP device is specified in OSSIE framework by files *GPP.h* and textitGPP.cpp, both located into the directory /home/ossie/src/ossie-0.8.2/platform/GPP.

It is important to notice that GPP_i class refers to ExecutableDevice_impl class in the constructor of the class. The definition of the ExecutableDevice_impl class is contained in the file *ExecutableDevice_impl.h* in the directory /home/ossie/src/ossie0.8.2/system/ossie/include/ossie.

In the above header file are listed the definitions of the constructor/destructor, public and private functions of the *ExecutableDevice_impl* class. The implementations of the functions, described in the header file, are contained in the file *ExecutableDevice_impl.cpp* in directory /home/ossie/src/ossie-0.8.0/system/ossie/framework. In particular we are interested in public function called *execute()*.

In this function *fork()* method is used to create a new process and execute it on the device, in this case a GPP device. Without modifications on this code it is impossible to address a specific core of a multi-core processor and it is the native scheduler algorithm that decides the core in which the new process will be run. In order to enable the allocation of a software component on a specific processor is necessary to modify the part of the code regarding the child process and the GPP device. In fact, it is necessary to pass to the *execute()* function an additional option including the information about the affinity mask chosen for the child process. The affinity mask can be passed to the function as an additional property added by the programmer to the software components or as a further settable option of the GPP device. In both cases architectural modifications to the OSSIE framework are needed, and we believe that the second way is simpler to implement than the first one. In fact, with this approach is possible to create a node with a number of *virtual* GPPs equal to the available

cores on the machine. Each *virtual* GPP is characterized by a specific affinity mask so that software components of the waveform can be allocated treating the different cores as distinct processors (PHY → Core1, MAC → Core2, IP → Core3,...). Once obtained the affinity mask information, it is sufficient to call the *sched_setaffinity()* function in the part of the code of the *execute()* function regarding the creation of the child process, using as pid the *new_pid* obtained by *fork()* function.

Chapter 4

SCA-compliant real-world interoperable waveforms

In this chapter we describe the implementation of a real-time and SCA-compliant waveform, developed inside the OSSIE SDR framework modified according to the guidelines provided in the previous chapter. In particular, we focus on the computational performance of the implemented waveform and on the performed validation tests.

4.1 A real-time tx/rx waveform for VHF aeronautical communications

Aeronautical radio communications is the subject of several analogue and digital standards, like VHF telephones and VHF data link (VDL) [25], in order to transfer vocal and data information between the aircraft and the ground stations during all the phases of the flight (landing, take-off,..).

As described in [25], the frequency spectrum from 118 MHz to 137 MHz is assigned to aeronautical communications. According to such standard, an analogue voice channel has a bandwidth of 8.3 KHz, resulting in 2280 available voice channels on the whole aeronautical frequency spectrum. The modulation used is legacy Amplitude-Modulation (AM) with modulation index 0.85. VHF aeronautical equipment is commonly based on ASICs implementing all the digital signal processing functions in Hardware. In such HW architecture, SW is used only for house-keeping functions and user interfaces. A fully-software implementation of a tx/rx chain for VHF aeronautical

communications is not generally considered because of the large economies of scale of those single-standard products.

On the other hand, the implementation of a new flexible and reconfigurable radio terminal capable of switching among different and heterogeneous radio communications standards is actually investigated in some research projects, like SANDRA (Seamless Aeronautical Networking through integration of Data links, Radios, an Antennas) [6], financed by the European Commission and strongly supported by manufacturers. The key-concept of this approach is the SDR paradigm in which easily-reconfigurable hardware like GPP, DSP and FPGA are used as a computational assets for executing all the software-defined signal processing functions. Also, the SW framework has to provide flexibility and interoperability between the software modules and the hardware platforms. This is exactly the aim of SCA architecture [3].

As said, the main advantages of this approach are:

- greater portability of source/object code;
- interoperability among the SCA-compliant software-defined radios;
- easy upgrade to further standard evolution;

This feature encourages the implementation of SCA-compliant waveforms implementing heterogeneous and interoperable radio communications standards (VHF maritime and aeronautical, GSM, TETRA, and even broadcasting standards like Digital Video Broadcasting (DVB)).

The aeronautical waveform described in this section is based on theettus USRP2 hardware [21] . As for the project described in Sec.2, the USRP2 provides the digital to analogue conversion (DAC) and baseband-to-RF conversion in the transmitter and the RF-to-baseband conversion including ADC in the receiver. According to the fully-software paradigm, all baseband transmitter/receiver functions are implemented through efficient C++ modules that were developed from scratch.

4.1.1 The OSSIE/USRP2 SDR platform

Any SDR system can be segmented into its own SW and HW sections that are customized on the basis of project requirements. In this project, both the transmitter and the receiver are developed on an open-source SW/HW platform that we will shortly describe.

The SW section of the project was implemented inside the OSSIE framework, a SCA-cmpliant framework described in Sec. 3.5. The peripheral used to transmit/acquire the signal in this project is the well-known USRP2 platform. Differently from the USRP, it consists of:

- 4 ADC at 100 MSamples/s with a resolution of 14 bit;
- 4 DAC at 400 MSamples/s with a resolution of 16 bit;
- Digital downconverters/upconverters with programmable decimation/interpolation rates;
- 1 Gigabit Ethernet interface;
- 2 Gbps high-speed serial interface for expansion;
- extension sockets to connect a wide variety of RF daughterboards;

The RF front-end used in the transmitter is the transceiver daughterboard WBX which operates from 50 MHz to 2.2 GHz while on the receiving side the RF front-end is provided by the receiver daughterboard TVRX covering the frequency spectrum from 50 MHz to 860 MHz. In both cases complex-valued I-Q samples are sent over a Gigabit Ethernet interface (interleaved real/imaginary parts, both represented as a signed *short int* on 2 bytes). The maximum sampling frequency sustained by the Gigabit Ethernet interface is 25 Msamples/ s corresponding to 800 Mb/s over the interface, with which a full spectral window of 25 MHz can be processed with no loss.

4.1.2 Transmitter implementation

We describe in this section the main constituent functions of the transmitter. The functional blocks are represented in Fig. 4.1, and all of them, except for the FFT block which is based on a standard routine [22], were implemented from scratch in order to have full control of the available system resources and to reach real-time performance.

4.1.2.1 Controller

The first block is the *Controller*. Its main function is to dynamically allocate the computational resources with respect to the number of active channels we intend to

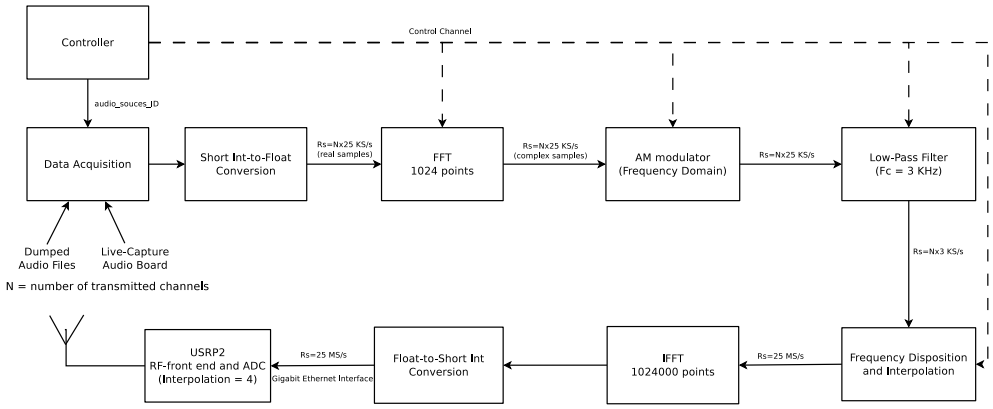

Figure 4.1: *Functional block scheme of the implemented transmitter chain.*

transmit. This block represents the interface with the user which sets the number of the active channels, the center frequencies of the channels to be used for transmissions, and the location of the audio sources.

4.1.2.2 Data acquisition and AM modulation

The first part of the chain is described as follows:

- the *Data Acquisition* block acquires from the controller the location of the audio sources and reads from local mass storage or from the audio board the digital audio file sampled at a sampling frequency of 25 KHz;
- the *Short Int-to-Float Conversion* is necessary to provide more accuracy to the subsequent processing blocks;
- the *Fast Fourier Transform* block performs a 1024-point FFT at a sampling rate of 25 KSamples/s;
- the *AM modulator* operates in the frequency domain and performs AM modulation of the input data with a modulation index of 0.85;
- the *Low-Pass Filter* limits the bandwidth of the signal to 3 kHz in order to be compliant with the strict constraints described on the standard [25]. For each

channel, the low-frequency 123 frequency samples out of the total of 1024 are retained.

4.1.2.3 Channel multiplexing and RF-front end

The second part of the chain is shortly described as follows:

- the *Frequency Disposition and Interpolation* block collects the frequency samples of the different channels and arrange them on their spectral positions according to the specifications delivered by the controller block;
- the *Inverse Fast Fourier Transform (IFFT)* performs a 1024000-point IFFT. The output stream has a sampling rate of 25 MS/s corresponding to a 25 MHz bandwidth;
- the *USRP2* board performs data interpolation with interpolation factor 4 in order to meet the sampling rate of its DAC (100 MSamples/s) and then performs frequency up-conversion with central frequency equal to 127.5 MHz.

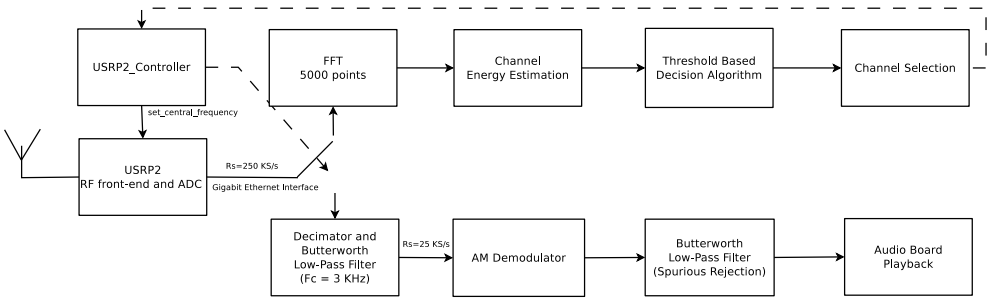

Figure 4.2: *Functional block scheme of the implemented receiver chain.*

4.1.3 Receiver implementation

In this section we describe the implementation of the *receiver* section that pairs with the transmitter described in 4.1.2. The receiver has been split in two sub-chains: the *channel sensing* algorithm and the *channel demodulation*. In the first phase of reception it is necessary to detect the active channels in the raster of the 2280 VHF

available channels; afterwards, the user selects the desired channel(s), the USRPs sets its reception parameters according to this choice, and finally demodulation starts.

4.1.3.1 The USRP2_Controller

The *USRP2_Controller* block is responsible for the dynamic change of the reception parameters of the USRP2 board. The correspondence between the channel name and its central frequency is stored in a table, called *channel map*. According to the information received from the *channel selection*, it sets on the USRP2 the proper center frequency, the HW decimation factor, the gain of the RF front-end, and redirects the output stream of the USRP2 towards the demodulation chain.

4.1.3.2 Channel sensing algorithm

The aim of this sub-chain is to reveal the active channels scanning all the channel positions on the VHF aeronautical spectrum. In order to match the accuracy of the algorithm with the available computational resources, the spectrum (19 MHz) has been divided into 76 sub-bands of 250 kHz which are iteratively analysed. The energy-based algorithm is described as follows:

- the *FFT* module receives 5000 complex samples from the USRP2 buffer at the rate of 25 KSamples/s and performs a 5000-points FFT;
- the *Channel Energy Estimation* block isolates the contributions of the different channels (166 complex samples per channel) and calculates the energy associated to each channel as an average of 10 consecutive FFT observations. Then the energy threshold is calculated as the arithmetical average of the 2280 estimated channel energies. A channel is considered as *active* if its estimated channel energy is larger than that threshold;
- the *Channel Selection* module presents to the end-user the list of the active channels. After the user choice, this block communicates the channel number to demodulate to the *USRP2_Cotroller*.

4.1.3.3 Signal demodulation and audio playback

After channel selection, the USRP2 output stream is redirected to the demodulation chain, whose components are as follows:

- the *Decimation and Butterworth Low-Pass Filter* component performs a decimation operation with a factor 10 reducing the sampling rate from 250 KSamples/s to 25 KSamples/s and so representing a 25 kHz frequency bandwdith. This sampling frequency was chosen in order to satisfy the requirements of the audio card on the host PC. To select the desired channel, signal samples are filtered by a tenth-order Butterworth low-pass filter with a cut-off frequency of 3 kHz (the same cut-off frequency used on the transmitter);
- the *AM Demodulator* performs classical AM envelope demodulation in the time domain by extracting the module of the complex-valued received samples;
- the *Butterworth Low-Pass Post-Filter* was introduced to reject the out-of-band spurious components generated by AM demodulation and to improve the audio quality of the demodulated signal;
- the *Audio Board Playback* component receives the data stream from input port, performs the DAC operation and plays the audio signal on the loudspeakers.

4.1.4 Validation tests and experimental results

The implemented waveform was developed, tested and validated at DSPCoLa, University of Pisa, Italy both for the transmitter and receiver side.

At the transmitter side, as stated in section 4.1.2, all the signal processing is performed by an off-the-shelf PC equipped with an Intel Core i7 2670QM processor (2.2 GHz), Ubuntu 11.10 Operating System and OSSIE 0.8.2. The transmitter was validated using some commercial radio devices (ICOM,..) capable to receive aeronautical communications. The main feature of the transmitter is the possibility to concurrently transmit in real-time several audio streams, coming from dumped audio files or from real-time acquisition devices (audio cards), placing them on arbitrary RF channels located anywhere in the 25 MHz spectrum window. Our measurements showed that our transmitter is fully compliant to the strict constraints on adjacent channel interference described in the standard [25]. For example, Fig.4.3 shows the spectrum of a parallel transmission of 5 audio streams. In this case, a low-cost radio scanner was capable to receive all of the active channels with no interference and a very good audio quality. The main design choice to implement all signal processing functions in the frequency domain makes the computational load practically independent of

the number of active streams: the computational heaviest function is the 1024000-point IFFT whose complexity is independent of the number of active channels. To demonstrate this, we stressed the chain by transmitting more than 20 parallel channels without obtaining any significant change on the computational load of the system, which turns out to be only 20% of the available CPU resources.

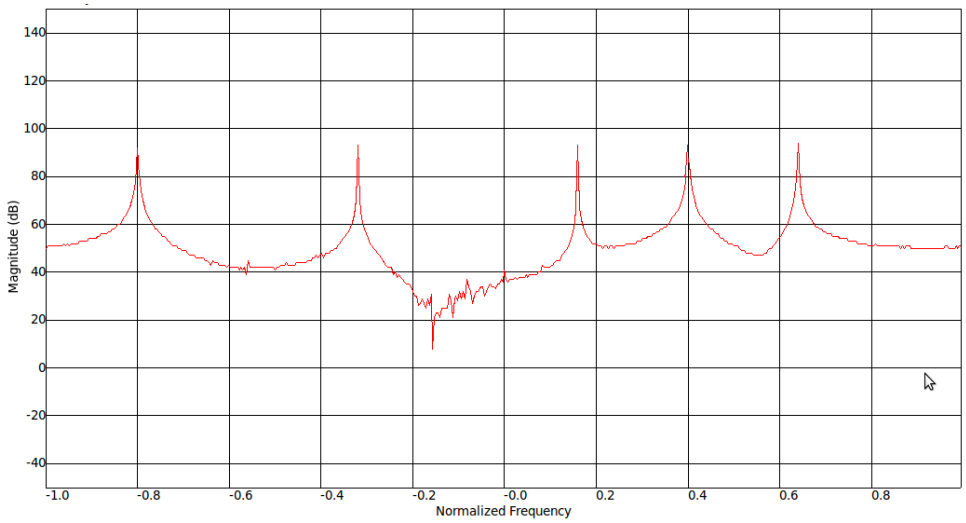

Figure 4.3: *Spectrum of the parallel transmission of several audio streams on a 25 MHz spectrum window.*

At the receiver side, as described in section 4.1.3, the signal captured by the antenna is sent to the USRP2 front-end and then, after baseband conversion and ADC conversion, it is routed via the Gigabit Ethernet interface to the host PC. In our lab, we used an off-the-shelf PC with an Intel Pentium IV, Ubuntu 10.04 Operating System and OSSIE 0.8.1. As described in [25], we tested the receiver using a modulated pilot tone at 2 kHz which was transmitted at different power levels by an Agilent Signal Generator E4438C. Tests showed that the occupied channel was correctly revealed as active, in any of the allowed spectral positions, and it was demodulated with very good audio quality even in low Signal-to-Noise Ratio (SNR) conditions. Fig. 4.4 shows the spectrum of the signal at the output of the low-pass filter ready to be played on the loudspeakers. The computational load of the receiver is about 25% of

the available computational power.

In addition to the instrumental tests above, we also performed some subjective, "real-world" listening sessions, in particular by demodulating the Volume Meteorological (VOLMET) channel transmitted by the Pisa Airport at 128.4 MHz, as well as other analogue voice channels, with very good perceived SNR.

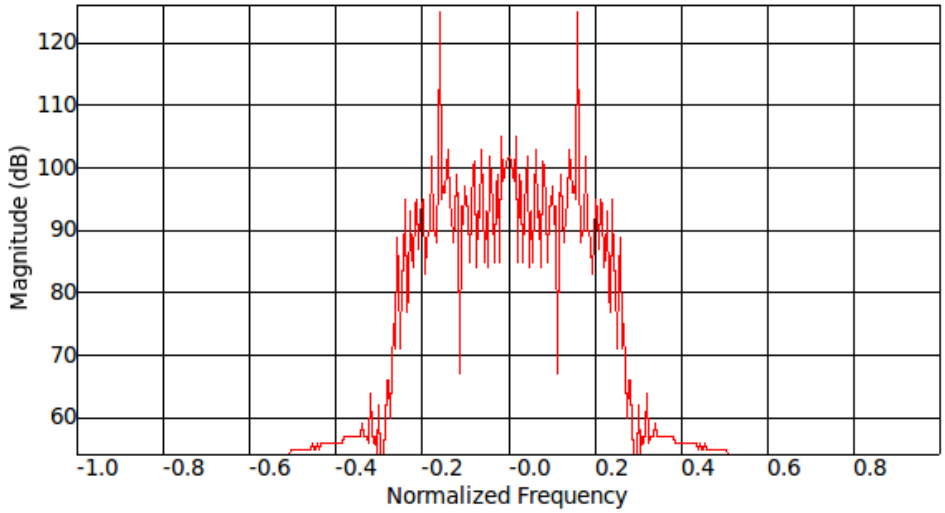

Figure 4.4: *Spectrum of the processed signal at the input of audio board*

Chapter 5

A SCA-compliant MA-based SDR: the AeroMACS waveform

In this chapter we detail the implementation of a real-time fully-software AeroMACS waveform capable of merging the two aspects discussed in this thesis, i.e. the computational efficiency provided by the MA technique described in Ch. 1 and the interoperability typical of SCA-compliant waveforms, like the one detailed in Ch. 4. The realization of this waveform is the proof-of-concept that it is possible to find an optimal trade-off between flexibility and computational efficiency, that it is also the ultimate goal of this thesis.

5.1 The AeroMACS waveform

The air transportation market is expected to double by the 2025 and the current air traffic systems will not be capable to satisfy this growth. In particular, new security requirements are requested, for example, to efficiently move people and cargo. Focusing on communications issues, it is possible to identify some critical aspects to improve: pilots situation awareness, Airline Operation Center (AOC) data traffic capacity, passengers and cabin communications systems. The solution for these critical aspects is the convergence of protocols and interfaces towards a new open system that is the result of the collection of different communications technologies tailored for a specific operational setting. In this scenario, the key-concepts are the

flexibility and the interoperability among the legacy communications systems and the new high-capacity communications standards. The challenge of this approach is to increase the capacity, security and efficiency of the aeronautical communications with no or small increase of the complexity and cost of the on-board equipments.

This has been also the objective of the already aforementioned SANDRA project [6], a research project financed by the European Commission involving some prestigious academic research centres (like the University of Pisa) and some of the major players of the European telecommunications market. The final scope of the SANDRA project has been the integration at different levels, from antenna to the network layer, of several communications standards (analogue VHF, VDL2, B-GAN, AeroMACS) on a reconfigurable Integrated Modular Router. The core of this project is the SDR approach in which reliable communications and interoperability among different standards, on the same reconfigurable hardware platform, can be easily provided by the SCA architecture.

The implemented waveform can be considered as the joint result of the analysis of AeroMACS waveform, the WiMAX IEEE 802.16e standard [26] for ATS/AOC communications studied in the SANDRA project, and the well-established background in implementing real-time fully-software SCA-compliant waveforms, as showed in Ch. 4. In particular, we will describe the implementation of a real-time, fully-software SCA-compliant AeroMACS waveform (PHY layer) [27]. As indicated in the SESAR/FCI recommendations, AeroMACS will provide the airport connectivity in the near future. Our implementation is based on the well-known Ettus USRP2 hardware [21]. As usual, the USRP2 provides the digital to analogue conversion and baseband-to-RF conversion at the transmitting side and the RF-to-baseband conversion including analogue to digital conversion at the receiving side. All baseband functions, except for the FFT block which is based on a standard routine [22], are implemented through an efficient C++ code that was developed from scratch inside the OSSIE SDR framework. Thus, the resulting development platform is the same used in Ch.4.

The AeroMACS standard [27] is based on the IEEE 802.16 - 2009 standard [26], selecting from this standard the parameters suitable for ATC and AOC communication in the airport surface environment. As IEEE 802.16, AeroMACS is based on a Orthogonal Frequency-Division Multiple-Access (OFDMA) system employing a Time Division Duplexing (TDD) protocol. There are two possible transmission mode: 5 MHz and 10 MHz. In this work we implemented the 5 MHz mode. The most

important system parameters of the implemented waveform are listed in the following table.

Parameter	*Required value*
Center frequency	5.091-5.150 GHz
System profile	OFDMA
Duplexing mode	TDD
Transmission Bandwidth	5 MHz
FFT size	512
Sampling factor	28/25
Sampling frequency	5.6 MSamples/s
Frame length	5 ms
OFDM symbol duration without guard interval	91.4 us
Guard interval	11.42 us
OFDM symbol duration with guard interval	102.84 us
Subcarrier spacing	10.94 kHz
# symbols per frame	51
Net useful bit-rate	12.25 Mb/s

Table 5.1: *AeroMACS waveform system parameters*

Fig.5.1 shows the time-frequency logical structure of an OFDMA AeroMACS frame in TDD mode. Each frame contains several logical data region belonging to the different users associated to the cell. So it is necessary to define a hierarchical grouping of the subcarriers in order to identify the resources allocated to the users.

The active subcarriers (carrying data or pilot) are divided into physical clusters containing 14 adjacent subcarriers over 2 consecutive symbols. In each symbol of this cluster 12 subcarriers are allocated for data transmission, while the remaining 2 are used as pilots. The physical clusters are renumbered into logical clusters and allocated to the active users. The subchannel, that is the minimum frequency-time resource unit, is composed by two clusters for a total of 48 subcarriers. Clusters allocated to a specific user are typically not adjacent to each other in the frequency domain and the spaces between them are not regular. For this reason, it is necessary to perform channel estimation on a cluster-by-cluster basis. In fact, channel knowledge on a

given cluster does not provide any information about the channel realization over the other clusters assigned to the same user because they are normally located in different frequency positions.

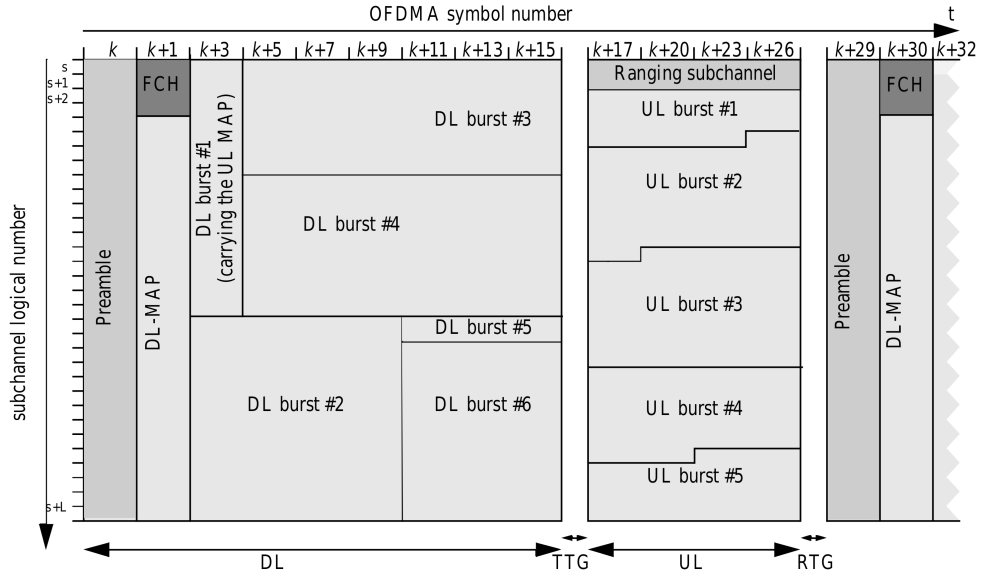

Figure 5.1: *AeroMACS OFDMA frame in TDD mode*

5.1.1 Transmitter implementation

Fig.5.2 depicts the constituent signal processing functions on the transmitter side. As already said, all of them, except for the FFT block which is based on an external standard routine [22], were implemented in order to optimize the usage of the computational resources and to reach real-time performance.

The implemented chain is described as follows:

- the *randomizer* removes time domain correlation in the binary data to transmit by performing a XOR operation with a PRBS generated via a generator polynomial and a Feedback Shift Register (FSR);
- the *convolutional encoder* performs convolutional encoding with rate 1/2, constraint length $K = 7$ and generator polynomial $g_1 = 177$ and $g_2 = 133$. The shift

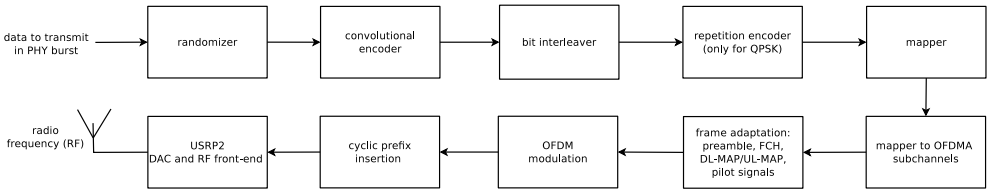

Figure 5.2: *Block scheme of the implemented transmitter signal processing blocks*

registers are initialized with the last seven bits of a Forward Error Correction (FEC) block, so realizing a FEC block-independent encoding. This technique is named *tail biting*;

- the *bit interleaver* prevents bits from original bitstream from being always associated with the same OFDM subcarriers that can offer, in general, a low SNR;

- the *repetition encoder* block performs a repetition encoding only if the used constellation is the QPSK, as in the case of the Frame Control Header (FCH) reference signal;

- the *mapper* block groups encoded interleaved bits and maps them into the constellation symbols (BPSK, QPSK, 16-QAM and 64-QAM);

- the *mapper to OFDMA subchannels* arranges the symbols in the OFDMA subchannels according to the allocation scheme fixed by the MAC layer (Fig.5.1);

- the *frame adaptation* block inserts in the OFDMA burst the reference signals (boosted pilot subcarriers, preamble, ...) necessary to synchronization steps (time/frequency) at the receiving side;

- the *OFDM modulation* block inserts the 92 virtual subcarriers and performs an IFFT operation with $N = 512$ points;

- the *cyclic prefix insertion* block inserts the guard interval in order to annul the time dispersion of the channel and avoid the Inter-Symbol Interference (ISI) and the Inter-Block Interference (IBI);

- the *USRP2* performs the digital-to-analogue conversion by interpolating the digital complex samples and then shifts the analogue baseband I/Q signal to the radio-frequency channel defined in the AeroMACS standard (5091-5150 MHz).

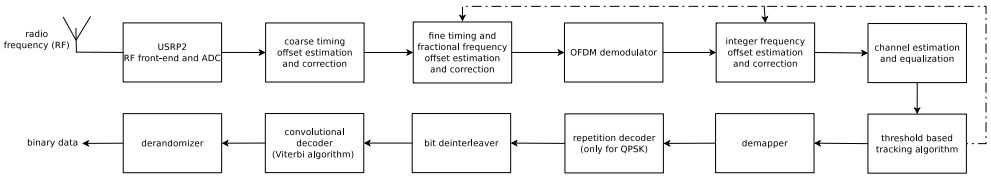

Figure 5.3: *Block scheme of the implemented receiver signal processing blocks*

5.1.2 Receiver signal processing blocks

In this section we describe the constituent blocks of the receiving section that pairs with the transmitter signal processing functions described in 5.1.1. As shown in Fig.5.3, the receiver chain can be divided in two sub-chains: the *synchronization and channel estimation/equalization* chain and the *channel decoding* subsystem.

The two subchains can be described as follows:

- the *USRP2* provides to the system the RF front-end/down-conversion to base-band and the analogue-to-digital conversion. The resulting signal is a stream of complex interleaved short samples;

- the *coarse timing offset estimation and correction* is performed by exploiting the characteristic frequency shape of the preamble. In fact, the preamble is transmitted by using *N/3* equispaced subcarriers while the remaining ones are forced to be null, so that two segments of the resulting signal in the time-domain with length *N/3* are highly-correlated. After the estimation is possible to identify and align to the start of the PHY data burst;

- the *fine timing and fractional frequency offset estimation and correction* is based on the algorithm described in [23]. This algorithm operates in the time domain and implements a ML open-loop timing and fractional (i.e a fraction of the inter-carrier spacing) frequency offset estimation by exploiting the inner redundancy

contained in OFDM cyclic prefix. In our work we adopted a modified version of this algorithm that uses averaged realizations of the log-likelihood function. After the estimation is possible to align to the first useful OFDM symbol and to correct in the time-domain the fractional frequency offset;

- the *OFDM demodulator* removes the cyclic prefix, performs a FFT operation with 512 points, acting as a matched filter for the OFDM modulation, and removes the virtual subcarriers;
- the *integer frequency offset estimation and correction* block jointly estimates the integer, i.e. multiple of intercarrier spacing, frequency offset and identify the training preamble. The algorithm maximizes a correlation function which depends on the frequency offset and the training preamble;
- the *channel estimation and equalization* block estimates channel frequency response by interpolating information carried by boosted pilot subcarriers and equalizes data subcarriers using the calculated channel profile and the ZF technique;
- the *threshold based tracking algorithm* is a finite-state machine that controls the renewal of timing and frequency offset estimations. It monitors the power level of the central subcarrier (DC) that, as stated in the standard, should be zero, and if the power goes over a threshold orders the renewal of the timing and frequency offset estimations;
- the *demapper* demodulates the received symbol constellation (BPSK, QPSK, 16-QAM or 64-QAM) into an encoded bit stream according to the used constellation;
- the *repetition decoder* block is enabled only for QPSK modulation and provides the decoded bitstream based on the classical majority decision-maker;
- the *bit deinterleaver* performs a bit-level operation in order to recover the right bit order after shuffling introduced by the bit interleaver on the TX side;
- the *convolutional decoder* performs the Viterbi decoding algorithm, the most famous method to decode a convolutional code. It is the computationally-heaviest block within the receiver chain having to work at a bitrate of 12.5 Mbps (measured after decoding);

- the *derandomizer* descrambles data stream by applying a XOR operation with a PRBS sequence.

5.1.3 Computational results

At the time of writing the implemented waveform is being tested and validated at DSPCoLa, University of Pisa, Italy. As stated in section 5.1.1, all the signal processing is performed by an off-the-shelf PC equipped with an Intel Core i7 2670QM processor (2.2 GHz), Ubuntu 11.10 Operating System and OSSIE 0.8.2.

As stated in section 5.1, AeroMACS signal has a baud rate of 5.6 MSamples/s on a 5 MHz channel bandwidth, cyclic prefix length of 1/8, 64-QAM constellation for the user data, 1/2 convolutional coding rate, all yielding a useful bitrate of 12.5 Mbps. AeroMACS waveform proved able to correctly modulate and demodulate the signal in real-time with a maximum throughput of ≈16 Mbps at the transmission side and ≈13 Mbps at the reception side and absorbing less than two of the eight (virtual) processors available on the i7-2670QM (25% of total computational power). It is clear that, the use of a multithreaded software architecture, ie. the parallel execution of the diverse functional blocks, can increase less than linearly the total throughput and equally distribute the load among the available cores. Nevertheless, the threads synchronization on a GPP can result in a more complicated and abstract software architecture. For this reason, in order to match the real-time constraints as fast as possible, we first profiled the processing blocks and then we implemented the waveform with the minimum number of threads.

It was also planned to implement basic MAC operations like the network identification and association in order to validated the implemented waveform with commercial devices. These are on-going activities at the time of writing.

5.1.4 MA within the AeroMACS waveform

Reaching real-time performance described in Sec.5.1.3 could not be possible without an extensive usage of MA programming technique detailed in Ch. 1

In particular, MA was applied to the two computationally-heaviest functional blocks of the receiving chain: the timing and fractional frequency offset estimation algorithm described in [23] and the $K = 7$ Viterbi decoder. At the time of writing, the acceleration factor for both the algorithms was greater than one order of magnitude

wrt a previous MA-free but computationally optimized version of such algorithms. We strongly believe that further improvements for these MA-based algorithms can be possible and could be the object of further research.

Chapter 6

Conclusions and perspectives

In the first chapter, we introduced a novel SDR programming technique, that we called Memory Acceleration, providing high acceleration factors in fully-software SDR implementation on GPP-based platform. We detailed the two algorithmic tools composing this technique, i.e. the Algorithm Segmentation (AS) and the Recursive Table Aggregation Rule (RTAR). We showed how it is possible to segment a signal processing algorithm, theoretically a full radio chain, and to re-aggregate its atomic blocks into look-up tables implemented in memory in a smart and cache friendly way.In order to show the potential of the MA technique we illustrated some test cases showing that it is possible to reach acceleration factors of *one order of magnitude* wrt an optimized computation-only reference implementation. Thanks to this it is possible to increase the power efficiency of the GPP-based SDR implementations without affecting their flexibility and reconfigurability.

In the second chapter, we showed a real-world application of the MA techinque. In fact, we detailed the architecture of a real-time, fully-software, ETSI DVB-T receiver named SR-DVB, based on the USRP peripheral and an ad-hoc software framework called *newRadio*. Architecture and implementation details are given together with achieved computational performance. In particular, we focused on the two heaviest signal processing blocks, i.e. the OFDM time/frequency synchronization and the Viterbi decoder. This receiver has to be intended as the proof-of-concept of feasibility and convenience of fully-software, GPP-based demodulation of even highly demanding, high throughput communication standards such as ETSI DVB-T. Meeting the real-time constraints was possible only relying on the extensive application of the MA technique on the computational heaviest block of the receiving chain.

In the third chapter, we faced the issue of interoperability in the SDR systems. In

particular, we focused on the analysis of the SCA, the software architecture defined by JTRS for providing enhanced portability and interoperability features to the military SDR implementations, called waveform in the SCA parlance. The key-concept is the usage of a software middleware, CORBA in release 2.2.2, which abstracts the HW/SW components and ideally makes the waveform platform-independent. Then we analysed the relation of SCA with the GPP-based implementation and the opportunity for increasing its computational efficiency exploiting the multi-core GPP platforms. Starting from this, we deeply analysed an open-source SCA-compliant framework, i.e. OSSIE, and its relation with the multicore architectures and we investigated the possibility of applying component-based multithreading programming and setting the cpu affinity. In particular, cpu affinity method has been studied since the application of it in SDR systems can result in performance benefits and major system control. In fact, the possibility of helping the scheduler in deciding in which core set a task is eligible to run, gives to the developer more control on a multi-core system at run-time. We provide also some guidelines to modify OSSIE in order to support these feature.

In the fourth chapter, we showed the development of a SCA-compliant waveform running in real-time on a GPP-based multi-core platform. In particular, we detailed the architecture of a fully-software SCA-compliant transceiver waveform for VHF aeronautical communications based on the USRP2/OSSIE platform. We focused on some key-choices in the design of the chain and on the most critical aspect from a computational point of view. Validation tests showed the developed waveform to be fully-compliant to the specifications of the standard both at the transmitter and at the receiver side with very good computational performance. We reported also the results of validation tests we performed in order to check the interoperability of the implemented waveform. The waveform should be considered as a proof-of-concept, the first step towards the realization of a low-cost, high-efficiency, flexible and reconfigurable SDR node in which heterogeneous aeronautical communications standards can easily coexist.

In the fifth chapter, we tried to merge the advantages of the aspects analysed in this thesis. We illustrated the implementation of a MA-based SCA-compliant waveform realizing a wideband communications standard, i.e. the AeroMACS waveform, a WiMax-like communications standard tailored for future aeronautical digital communications. We started from a brief description of the standard and then we concentrated on the signal processing blocks composing the tx/rx chain. We highlighted

the peculiarity of this waveform and the computational heaviest blocks in that chain and we showed the computational results before and after the application of MA technique. We realized that it is possible to integrate the MA technique inside the SCA framework without loosing flexibility and we proofed that, thanks to the usage of MA technique and the exploitation of the multi-core architectures it is possible to realize a real-time SCA-compliant wideband waveform capable of running in real-time on computing hardware platform exclusively based on GPP devices.

Before concluding this thesis let us list some possible development and research perspectives that can be followed in the future starting from the results presented in this thesis. The application of MA technique in GPP-based SDR systems showed that MA could have a big potential in the future development of fully-reconfigurable SDRs. In particular, it could be potentially used on current GPP platforms for the realization and deployment of SDR systems in which we do not have tight power constraints (base stations, vehicular devices, etc..). Regarding the mobile devices, MA contributed in reducing the power efficiency gap between SDRs and ASICs but we believe that this is still not enough for enabling a wide deployment of SDR mobile devices based on GPPs in next future. Anyway the increasing availability of jointly energy efficient and powerful GPP devices can help this process. Moreover, the acceleration factors presented in this thesis were obtained with computing architectures and compilers (GNU g++) that are totally unaware of the MA approach and therefore tend to disfavor memory access to privilege pure computation. This suggests the possibility of developing GPP devices tailored for MA-based SDR in which memory access and management is optimized, so resulting in greater acceleration factor. From a research perspective, there are still some open points in the MA technique formalization. In fact, the current algorithm segmentation method, differently from the RTAR that can be easily automatized, still needs of human support in defining the correct granularity level and the reaching of the atomicity in the algorithm decomposition. We believe that it is possible to find some additional semantics and metrics able to automatize the algorithm segmentation process. Finally, considering we showed that the MA can be integrated into SCA-compliant framework, this could open new perspectives also in the world of interoperable military (or not) waveforms. In fact, the computational overhead introduced by SCA could be counterbalanced by the extensive application of MA technique, so enabling the deployment of SCA-compliant SDRs running on GPP-based platform and enhancing the flexibility of the SCA-compliant waveforms.

In this ideal MA-accelerated SCA-compliant architecture, the MA technique could be seen as an Acceleration Abstraction Layer (AAL) for the current SCA architecture.

Appendix A

A time delay estimation algorithm for SDR-enabled cognitive positioning systems

In this appendix we describe the definition of a new time estimation algorithm for cognitive positioning systems. Although this argument is not directly linked with the topics discussed in this thesis, we decide to insert this because it represents not only a considerable part of the performed research activity but also because we strongly believe that a cognitive positioning system can be naturally enabled only by relying on the SDR paradigm. In this approach, the flexibility feature discussed in the previous chapter assumes here a central role and the depicted estimation algorithm cab be considered as a further proof of the potential of our fully-software SDR approach.

A.1 Cognitive radio and cognitive positioning paradigms

Cognitive Radio (CR) is a paradigm for wireless communication in which either a network or a wireless node changes its transmission and/or reception parameters (signal format and bandwidth, frequency band etc.) to communicate efficiently avoiding interference with licensed or unlicensed users [28]. This paradigm is a well-known paradigm in the telecommunications research society, whilst it is not so famous the "sister" concept of *Cognitive Positioning* (CP) [29].

Cognitive systems strive for optimum spectrum efficiency by allocating capacity as requested in different, possibly disjoint frequency bands. Such approach is naturally

enabled by the joint effect of software-defined systems and the adoption of flexible MultiCarrier (MC) technologies, in all of its flavors: traditional OFDM, Filter-Bank Multicarrier Modulation (FBMCM),and possibly non-orthogonal formats with full time/frequency resource allocation [30, 31]. On the other hand, modern wireless networks more and more expect availability of location information about the wireless terminals, driven by requirements coming from applications, or just for better network resources allocation [32,33]. Thus, signal-intrinsic capability for accurate localization is a goal of 4th Generation (4G) as well as Beyond-4G (B4G) networks [34,35].

In the following, we will see that a multicarrier signal format, possibly split in (two or more) non-contiguous bands, gives also new opportunities in terms of enhanced-accuracy time delay estimation that ultimately translates into enhanced accuracy positioning.

We will start from Section A.2 with a review of the Modified Cramér-Rao Bound (MCRB), its (approximated) frequency-domain computation, as well as the study of the impact on the bound of the location of the received signal spectrum within the receiver bandwidth. In Section A.3 we will present the general structure of a bandlimited multicarrier ranging signal, and in Section A.4 we will discuss how to optimize such signal format through minimization of the MCRB, to come to the description of Cognitive Positioning opportunities (optimization of the MC signal format through minimization of the MCRB, and design of cognitive algorithms). The customary *Conclusions* section wraps up the paper.

A.2 Frequency-domain computation of the MCRB for delay estimation

A.2.1 The (M)CRB for delay estimation

The Cramér-Rao Bound (CRB) is a fundamental lower bound on the variance of any estimator [36,37] and, as such, it serves as a benchmark for the performance of actual parameter estimators [38–40]. It is well known and widely adopted for its simple computation, but its close-form evaluation becomes mathematically intractable when the vector of observables contains, in addition to the parameter to be estimated, also some *nuisance parameters*, i.e., other unknown random quantities whose values are not the subject of the estimator (information data, random chips of the code of

a ranging signal, multipath channel parameters etc.), but that concur to shape the actual values of the signal (of the observables themselves). The MCRB for a received signal $x(t)$ embedded in complex-valued Additive White Gaussian Noise (AWGN) with two-sided Power Spectral Density (PSD) $2N_0$ is found to be [41]

$$MCRB\left(\lambda\right) = \frac{N_0}{\mathbb{E}_{\mathbf{u}}\left\{\int_{T_{obs}}\left|\frac{\partial x(t)}{\partial\lambda}\right|^2 dt\right\}} \tag{A.1}$$

where λ is the unknown (scalar) parameter to be estimated, $\mathbf{u}$ is the vector of the nuisance parameters, T_{obs} is the observation time-interval, and $\mathbb{E}_{\mathbf{u}}$ indicates statistical expectation wrt $\mathbf{u}$.

As stated before, providing enhanced-accuracy location information for a wireless terminal basically amounts to providing enhanced accuracy estimate of the time-of-arrival of the data or ranging signal from/to the terminal to/from the wireless network access point. So we will focus in the following onto the issue of *Time-Delay Estimation* (TDE) for a data or ranging signal. Assuming ideal coherent demodulation (i.e., assuming that during signal tracking the carrier frequency and the carrier phase are known to a sufficient accuracy), the baseband-equivalent of the received signal is

$$r\left(t\right) = x\left(t-\tau\right) + n\left(t\right), \tag{A.2}$$

where τ is the group delay experienced by the radio signal when propagating from the transmitter to the receiver (as seen in the reference time-frame of the receiver). The MCRB for this specific case is easily found to be [41]

$$MCRB\left(\tau\right) = \frac{N_0}{\mathbb{E}_{\mathbf{u}}\left\{\int_{T_{obs}}\left|\frac{dx(t-\tau)}{d\tau}\right|^2 dt\right\}}. \tag{A.3}$$

From this, we can devise a simple criterion for optimal signal design: finding that specific waveform that, across a pre-set bandwidth and for a certain SNR, gives the minimum MCRB value []. We will not consider here any aspects related to a possible *bias* of the estimator arising in a severe multipath propagation environment, to concentrate on the main issue that we have just stated. Before providing the solution to this, we will make a short detour to gain better insight into the relation between the spectral shape of a signal and its TDE MCRB.

A.2.2 Computation of the MCRB(τ) in the frequency domain

The issue of evaluating the ultimate performance of TDE dates back to more than six decades, and plenty of contributions dealing with this problem can be found [42], [43]. A recent formalization of the problem is contained in [40], but with specific assumptions on the cyclostationarity of the signal and on the piecewise-constant nature of the parameters to estimate. We intend here to review and simplify what has already been done in the past, to come to a "clean" formulation of the TDE MCRB in the frequency domain that gives much insight into the opportunities to solve the problem of signal optimization.

Assume we are receiving a generic pilot ranging signal $x(t;\mathbf{c})$ bearing no information data, but containing a pseudo-random ranging code $\mathbf{c}$ whose chips are considered as binary ($\in\{\pm 1\}$) (Independent and Identically Distributed) IID nuisance parameters. This signal turns out to be a *parametric random process*, for which each time-unlimited sample function is a signal $x(t;\mathbf{c})$ with finite power P_x and chip rate $R_c = 1/T_c$. We recall that the PSD of a (parametric) random process $x(t)$ like ours is defined to be

$$S_x(f) \triangleq \lim_{T_{obs}\to\infty} \frac{\mathbb{E}_{\mathbf{c}}\left\{|X_{T_{obs}}(f;\mathbf{c})|^2\right\}}{T_{obs}} \tag{A.4}$$

where $X_{T_{obs}}(f;\mathbf{c})$ is the Fourier transform of the finite-energy *windowed* signal $x_{T_{obs}}(t;\mathbf{c}) = x(t;\mathbf{c})\cdot \mathrm{rect}(t/T_{obs})$ truncated in the time interval $[-T_{obs}/2; T_{obs}/2]$, and where $\mathbb{E}_{\mathbf{c}}\{\cdot\}$ denotes statistical expectation over the (random) code chips. The power P_x of the band-pass signal $x_{BP}(t;\mathbf{c}) = \mathfrak{Re}\{x(t;\mathbf{c})\exp(\jmath 2\pi f_0 t)\}$ (f_0 the carrier frequency) is

$$P_x = \frac{1}{2}\int_{-\infty}^{\infty} S_x(f)df. \tag{A.5}$$

With the signal model (A.2), the MCRB is found to be

$$MCRB(\tau) = \frac{N_0}{T_{obs}\int_{-\infty}^{\infty} \frac{1}{T_{obs}}\mathbb{E}_{\mathbf{c}}\left\{\left|\frac{\partial x_{T_{obs}}(t-\tau;\mathbf{c})}{\partial t}\right|^2\right\}dt} \tag{A.6}$$

Using Parseval's relation we get

$$MCRB(\tau) = \frac{N_0}{T_{obs}\int_{-\infty}^{\infty} 4\pi^2 f^2 \frac{\mathbb{E}_{\mathbf{c}}\left\{\left|X_{T_{obs}}(f;\mathbf{c})\right|^2\right\}}{T_{obs}}df} \tag{A.7}$$

We now adopt a crucial assumption that leads to an accurate approximation of the bound. Specifically, we assume T_{obs} very large, so that $\mathbb{E}_{\mathbf{c}}\left\{\left|X_{T_{obs}}(f;\mathbf{c})\right|^2\right\}\Big/T_{obs} \cong S_x(f)$. Under this hypothesis,

$$MCRB(\tau) = \frac{N_0}{T_{obs}4\pi^2 \int\limits_{-\infty}^{\infty} f^2 S_X(f)\, df}$$

$$= \frac{1}{8\pi^2 \cdot N \cdot \frac{E_c}{N_0}\beta_x^2} = \frac{B_{eq}T_c}{4\pi^2 \cdot \frac{E_c}{N_0}\beta_x^2} \tag{A.8}$$

where $T_{obs} = N \cdot T_c$ (N very large), β_x^2 is the normalized second-order moment of the PSD of the complex signal

$$\beta_x^2 \triangleq \frac{1}{2P_x} \int\limits_{-\infty}^{\infty} f^2 S_X(f)\, df \quad , \tag{A.9}$$

$E_c = P_x \cdot T_c$ is the average signal energy per chip symbol of the ranging code, and finally $B_{eq} = 1/(2NT_c)$ is the (one-sided) noise bandwidth of a closed-loop estimator equivalent to an open-loop estimator operating on an observation time equal to NT_c. From (A.8)-(A.9), we conclude that, as known, the MCRB depends on the second-order moment of the PSD of the complex signal β_x^2, independent of the type of signal format (modulation,spreading, etc.) that is adopted.

A.2.3 Dependence of the MCRB(τ) on spectrum location

Since the MCRB is proportional to the inverse of the second order moment of the PSD, some considerations need to be done on the effect of the location of the *center of gravity* or *center frequency* of the signal spectrum. The center frequency f_G of the signal spectrum is given by [40]

$$f_G \triangleq \frac{1}{2P_x} \int\limits_{-\infty}^{\infty} f \cdot S_x(f)\, df \tag{A.10}$$

The (squared) Gabor bandwidth of signal x is also defined as

$$(B_G)^2 = \frac{1}{2P_x} \int\limits_{-\infty}^{\infty} (f - f_G)^2 \cdot S_x(f)\, df. \tag{A.11}$$

When the PSD is not even-symmetric, the center frequency of the PSD is not null, and the normalized second-order moment β_x^2 is generally larger than the squared Gabor bandwidth:

$$\beta_x^2 = f_G^2 + (B_G)^2. \tag{A.12}$$

As an example, consider the bandpass received signal

$$r_{BP}(t) = \mathfrak{Re}\left\{x(t-\tau)\, e^{j(2\pi f_0(t-\tau)+\varphi)}\right\} + n_{BP}(t), \tag{A.13}$$

$n_{BP}(t)$ being the band-pass AWGN with power spectral density $N_0/2$. The PSD of the (bandpass) received signal is

$$S_{r_{BP}}(f) = \frac{1}{4}S_x(f-f_0) + \frac{1}{4}S_x(-f-f_0) + \frac{N_0}{2}. \tag{A.14}$$

Assume also that $S_x(f)$ is strictly bandlimited within $[-B, B]$. Instead of demodulating the signal wrt the nominal carrier frequency f_0, we may use the frequency

$$f_B = f_0 - B \tag{A.15}$$

so that the resulting demodulated I/Q complex signal is

$$r(t) = x(t-\tau)\, e^{j(2\pi B(t-\tau)+\varphi)} + n(t), \tag{A.16}$$

whose signal component has a spectrum that is asymmetric (as shown in Fig. A.1) even when $S_x(f)$ is indeed symmetric. In particular, $z(t) = x(t-\tau)\, e^{j(2\pi B(t-\tau)+\varphi)}$ is an *analytic signal* with no spectral components on $f < 0$. This means that $z(t) = z_I(t) + \check{z}_I(t)$ where $\check{z}_I(t)$ is the Hilbert transform of $z_I(t)$, and also means that both $z_I(t)$ and $z_Q(t) = \check{z}_I(t)$ are bandlimited to $2B$. For such situation

$$\beta_x^2 = B^2 + (B_G)^2 > (B_G)^2. \tag{A.17}$$

The inevitable conclusion is that $z(t)$ is *better* than $x(t)$ for delay estimation. The price to be paid is the increased *receive* bandwidth: the receiver needs twice as much processing bandwidth wrt the case of conventional demodulation ($f_B = f_0$). The most striking aspect of this computation is that the *transmit* bandwidth is *unchanged*.

Global Positioning System (GPS) HW designers know very well about the classical modes of operation for a GPS receiver: the term $(B_G)^2$ in (A.12) applies to conventional *code tracking* on the symmetric baseband spectrum, whilst f_G^2 can be though of

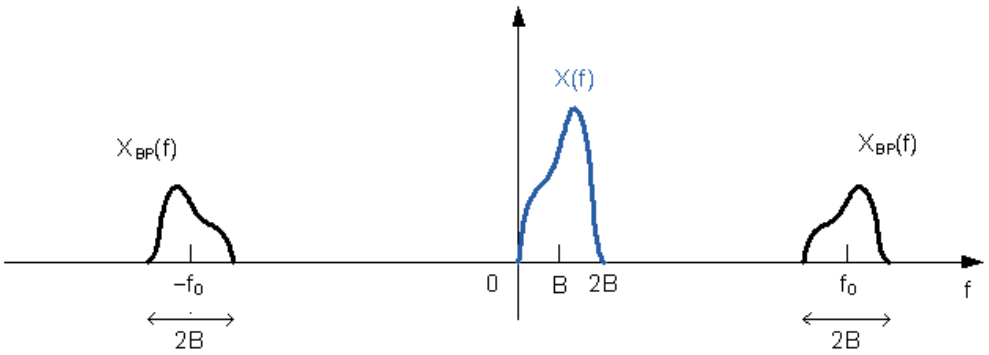

Figure A.1: *Single-Side Band Baseband Conversion of a Bandpass Spectrum*

as relevant to *carrier navigation* [44], where delay estimation is performed by tracking the cycles of the carrier directly on the bandpass signal. Tracking the signal carrier ensures higher-precision TDE (just because the duration of a carrier cycle is smaller than the duration of a ranging code chip), but with the drawback of high *estimation ambiguity*, that is related to the *periodicity* of the signal. The period of the ranging code is usually large (theoretically infinite if the chips are random - it is actually 1 ms for the GPS C/A code, and about 1 week for the P code), so that it is relatively easy to tell each code period apart and have no ambiguity in TDE. On the contrary, the period of the carrier is just $T_0 = 1/f_0$ (about 0.63 ns for GPS) so that complicated procedures of *phase unwrapping* are to be adopted (very often in a post-processing mode in virtual time) not to run into ambiguity problems. Ambiguity introduces an unknown estimation bias in terms on integer multiples of $T - 0$ that harm estimation accuracy. In our example (A.16), ambiguity is caused by the "virtual subcarrier" at the frequency B in the asymmetric demodulated spectrum. The zero-crossings of the virtual carrier can be used to track the signal and improve delay estimation, at the price of introducing an ambiguity equal to $1/B$. This is again a well-known issue for all positioning signal containing a subcarrier like Binary-Offset Carrier (BOC) modulations [45].

The preceding discussion shows that the *location in frequency* of the signal spectrum as well as its shape across the occupied bandwidth affects TDE accuracy and so, positioning accuracy. The two factors can be easily controlled assuming the usage of a multicarrier format for the ranging signal and of a software-defined system to

support the required flexibility. This is our starting point in the discussion of the opportunities for cognitive positioning.

A.3 Filter-bank multicarrier ranging signals

We will now examine in detail the format of a bandlimited multicarrier signal with a binary ranging code to perform (cognitive) positioning. We will assume that the chip rate of the ranging code is $R_c = 1/T_c$, and that the code repetition length L (the code period LT_c) is very large.

A basic ranging MC signal can be constructed following the general arrangement of multicarrier modulation: the input chip stream $c[i]$ of the ranging code is parallelized into N substreams with a MC symbol rate $R_s = R_c/N = 1/(NT_c) = 1/T_s$, where T_s is the time duration of the "slow-motion" ranging chips in the N parallel substreams. We can use a *polyphase* notation for the $k-$th ranging subcode ($k = 0, 1, , N-1$) in the $k-$th substream (subcarrier) as $c^{(k)}[n] \triangleq c[nN + k]$ where k, $0 \leqslant k \leqslant N-1$, is the subcode identifier and/or the subcarrier index, whilst n is a time index that addresses the $n-$th MC symbol (block) of time length $T_s = NT_c$. The substreams are then modulated onto a raster of evenly-spaced subcarriers with frequency spacing f_{sc} and the resulting modulated signals are added to give the (baseband equivalent of the) overall ranging signal. In Filter-Bank Multicarrier Modulation (FBMCM) (also called Filtered MultiTone (FMT)) the spectrum on all subcarriers are strictly bandlimited and nonoverlapping, akin to conventional Single Channel Per Carrier (SCPC). The resulting signal is

$$x(t) = \sqrt{\frac{2 \cdot P_T}{N}} \sum_n \sum_{k=0}^{N-1} c^{(k)}[n] g(t - nT_s)\, e^{j2\pi k(1+\alpha)t/T_s} \tag{A.18}$$

where P_T is the signal power, and $g(t)$ is a bandlimited pulse, for instance a square-root raised cosine pulse with roll-off factor α - in this case the subcarrier spacing is $(1+\alpha)/T_s$. Figure A.2 shows the arrangement of a multicarrier modulator implementing (A.18). It is well known that this arrangement has an efficient realization based on the usual IFFT processing of multicarrieer modulators, followed by a suited polyphase filterbank based on the prototype filter $g(t)$ [46].

Contrary to OFDM, subcarrier orthogonality is attained in the frequency domain and holds irrespective of the signal observation time.

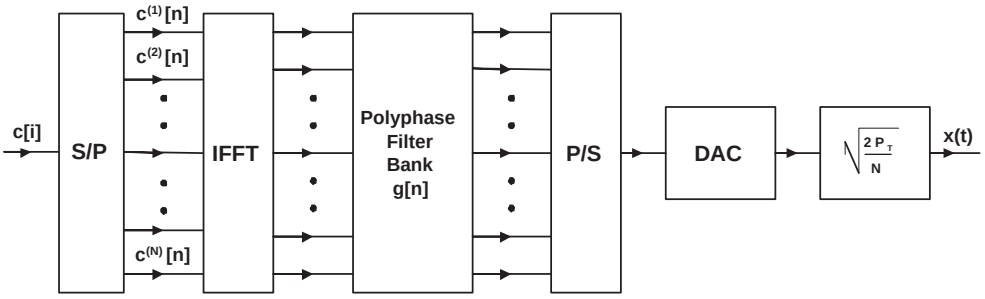

Figure A.2: *Filter-Bank MultiCarrier Modulator*

The bandwidth occupancy of an FBMCM signal can be easily calculated under the hypothesis that the chips of the ranging code are IID, as is the case for a very long Pseudo Noise (PN) code. The PSD $S_x(f)$ of $x(t)$ is

$$S_x(f) = \frac{2 \cdot P_T}{N} \sum_{k=0}^{N-1} S\left(f - k\frac{(1+\alpha)}{T_s}\right), \tag{A.19}$$

where

$$S(f) = \frac{\mathbb{E}\{|c^{(k)}[n]|^2\}}{T_s} |G(f)|^2 = G_N(f) \tag{A.20}$$

is the PSD of each sub-stream with chip rate $1/T_s$, in which $G(f)$ is the Fourier transform of the pulse$g(t)$, and $G_N(f)$ is a Nyquist frequency-raised-cosine function with roll-off factor α, and with $G_N(0) = T_s$. Figure A.3 shows a sample FBMCM spectrum to show its bandlimitation feature. Differently from OFDM, FBMCM does not need virtual carriers, since the spectrum is strictly bandlimited and the sampling frequency in the modulators/demodulators obey Nyquist's rule.

When T_{obs} is sufficiently large, $T_{obs} = N_m T_s$, $N_m \gg 1$, the MCRB for such a signal can be easily computed:

$$MCRB(\tau) = \frac{T_c^2}{8\pi^2 \frac{E_c}{N_0}\frac{N_m}{N}\left[\xi_g + \frac{(1+\alpha)^2}{N}\sum_k k^2\right]} \tag{A.21}$$

where $E_c = P_T \cdot T_c$ and ξ_g is the so-called Pulse Shape Factor (PSF)[1], a normalized

[1]For a square-root raised-cosine pulse with roll-off factor α, $\xi_g = 1/12 + \alpha^2(1/4 - 2/\pi^2)$

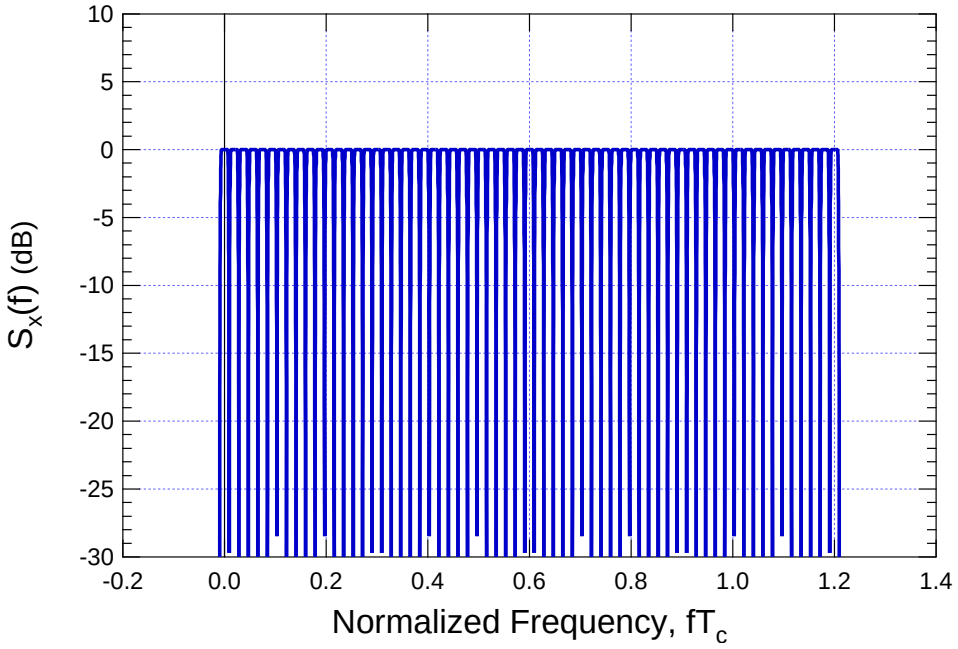

Figure A.3: *PSD of a FBMCM signal with $N = 64$ carriers and a squared-root raised cosine pulse with $\alpha = 0.2$*

version of the Gabor bandwidth of pulse $g(t)$ [39]:

$$\xi_g \triangleq \frac{T_s^2 \cdot \int_{-\infty}^{\infty} f^2 \cdot |G(f)|^2 df}{\int_{-\infty}^{\infty} |G(f)|^2 df} . \tag{A.22}$$

We can easily show that this bound is exactly coincident with the one obtained by the general frequency-domain formulation using (A.8)-(A.9) with the expression (A.19) for the signal PSD.

A.4 Cognitive bounds and algorithms with multicarrier signals

A.4.1 MCRB with uneven power allocation and the relevant bounds

A multicarrier signal naturally allows for selective allocation of signal power across a wide bandwidth. To accommodate this feature, we only need to further introduce in (A.18) an amplitude coefficient on each subcarrier:

$$x(t) = \sqrt{\frac{2P_T}{N}} \sum_n \sum_{k=0}^{N-1} p_k \, c^{(k)}[n] g(t - nT_s) \, e^{j2\pi k(1+\alpha)t/T_s} \tag{A.23}$$

where p_k^2 is the relative power weight of carrier k ($p_k \geq 0$), that satisfies

$$\sum_k p_k^2 = N \tag{A.24}$$

to give the nominal total transmitted power P_T. Some p_k's can also be 0 indicating that the relevant subcarrier or even a whole subband is not being used. The relevant MCRB is found to be

$$MCRB\,(\tau) = \frac{T_c^2}{8\pi^2 \frac{E_c}{N_0} \frac{N_m}{N} \left[\xi_g + \frac{(1+\alpha)^2}{N} \sum_k k^2 p_k^2 \right]} \tag{A.25}$$

A nice problem is now finding the *power distribution* that provides the best timing estimation accuracy, that is, minimizes (A.25) through maximization of

$$\sum_k k^2 p_k^2 \tag{A.26}$$

with the constraint (A.24). When N is fixed and the spectrum is symmetric (f_G=0), (A.26) is maximized for the optimal power scheme

$$\begin{cases} p_k = \sqrt{\frac{N}{2}} & k = \pm\frac{N-1}{2} \\ p_k = 0 & k \neq \pm\frac{N-1}{2} \end{cases} \tag{A.27}$$

indicating a configuration in which the power of the signal is concentrated at the edge of the bandwidth. This is a well-known results of Gabor bandwidth theory. When

comparing the optimal distribution with what we get with the uniform (flat) power allocation $p_k \equiv 1$, we see for N very large that

$$MCRB|_{\text{optimal}}(\tau) \cong \frac{MCRB|_{\text{flat}}(\tau)}{3} \tag{A.28}$$

An MC signal with uneven, adaptive power distribution can be adopted to implement a *Cognitive Positioning System* (CPS) [29]. In our envisioned FBMCM scheme for positioning, the proper power allocation allows to reach the desired positioning accuracy, not only in an AWGN channel, but also in an ACGN channel (colored noise). Colored noise arises from variable levels of interference produced by co-existing (possibly primary) systems on different frequency bands. The key assumption is that such interference can be modelled as a Gaussian process. This is certainly justified in wireless networks with unregulated multiple access techniques such as Code Division Multiple Access (CDMA) and/or Ultra Wide Band (UWB).

A simple case study for a CPS may start from the assumption to use only M out of the N available subcarriers on which the total assigned bandwidth can be partitioned, and to use them all at the same power level. The signal format is (A.23) with $p_k = \sqrt{N/M}$ for M subcarriers, and $p_k = 0$ for the remaining $N - M$ elements. The corresponding MCRB (symmetric spectrum) is minimized by a configuration in which the M subcarriers are split in two groups of $M/2$ subcarriers each, and placed at the edges of the bandwidth. Figure A.4 depicts the appearance of the PSD of such signal with $N = 64$ and $M = 16$. The (optimal) MCRB can be easily computed and turns out to be

$$MCRB(\tau) = \frac{T_c^2}{8\pi^2 \frac{E_c}{N_0}\frac{N_m}{N}} \cdot \frac{1}{\left[\xi_g + \frac{(1+\alpha)^2}{M}\frac{(M+2)(M^2+4M-3MN-6N+3N^2+3)}{12}\right]} \tag{A.29}$$

Let us compare this result with the MCRB that applies to M contiguous carriers (symmetric around $f = 0$), that is, for the same total net spectral occupancy. We only need to replace N with M in (A.21) and do the summation:

$$\frac{T_c^2}{8\pi^2 \frac{E_c}{N_0}\frac{N_m}{M}\left[\xi_g + \frac{(1+\alpha)^2}{12}(M^2-1)\right]} \cong \frac{T_c^2}{2\pi^2 \frac{E_c}{N_0}\frac{(1+\alpha)^2}{3} N_m M} \text{ (large } M\text{).} \tag{A.30}$$

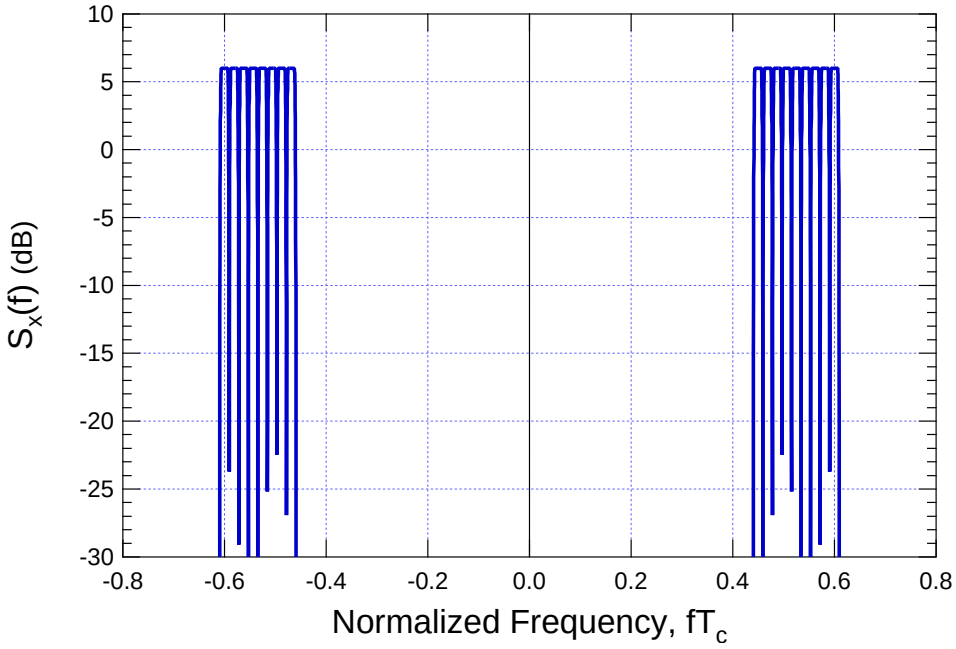

Figure A.4: *PSD of a multicarrier signal with even power allocation on two non-contiguous bands*

If we assume $M \ll N$, (A.29) reads

$$\cong \frac{T_c^2}{2\pi^2 \frac{E_c}{N_0} (1+\alpha)^2 N_m \cdot N} \text{ (large } M) \tag{A.31}$$

that is considerably smaller than (A.30). The effect of having the two bands for positioning far apart in the frequency domain is apparent.

A.4.2 Bounds for CP in the ACGN channel

We have to revise the criteria for optimal signal design if we drop the assumption of white background noise. To be specific, we investigate the issue of finding the power allocation scheme that gives the minimum MCRB [47] for TDE in a Gaussian channel whose (additive) noise has a variable PSD $S_n(f)$. Skipping some details, our starting point is the computation of $CRB(\tau)|_{ACGN}$ as a function of the "partial"

CRBs $CRB_k(\tau)$ we would get observing each subcarrier separately from the others on its own k-th subband only. By virtue of signal orthogonality we get

$$CRB(\tau)|_{ACGN} = \frac{1}{\sum\limits_k CRB_k^{-1}(\tau)} \tag{A.32}$$

Doing the computations, we have

$$CRB(\tau)|_{ACGN} = \frac{1}{4\pi^2 T_{obs}(\Delta f)^3} \cdot \left(\sum_k k^2 \frac{S_x\left(k\Delta f\right)}{S_n(k\Delta f)} \right)^{-1}, \tag{A.33}$$

where $\Delta f = f_{sc}$ is the subcarrier spacing, and where the PSD of the transmitted signal and of the noise were considered constant across each subband. A fundamental result is obtained if we let $N \to \infty$ (thus $\Delta f \to df$ and $\sum \to \int$) [48] :

$$CRB(\tau)|_{ACGN} = \frac{1}{4\pi^2 T_{obs}} \cdot \left(\int_B f^2 \frac{S_x\left(f\right)}{S_n(f)} df \right)^{-1}. \tag{A.34}$$

A.4.3 Optimum signal design for CP in the ACGN channel

Coming back to the problem of enhancing TDE accuracy, and sticking for simplicity to the finite-subcarriers version of the problem, we have to minimize the MCRB (A.33) with the constraint (A.24) on the total signal power. Considering that $S_x(k\Delta f)$ is proportional to p_k^2, we are to find the power distribution p_k^2 that maximizes

$$\sum_k \frac{k^2 p_k^2}{S_n(k\Delta f)} \tag{A.35}$$

subject to the constraints $\sum p_k^2 = N$ and, of course, $p_k \geq 0$. The optimal distribution is easily found to be

$$\begin{cases} p_k = 0 & k \neq k_M \\ p_{k_M} = \sqrt{N} & \end{cases},$$

$$k_M = \underset{k}{\operatorname{argmax}} \frac{k^2}{S_n(k\Delta f)} \tag{A.36}$$

that corresponds to placing all the power onto the sub-band for which the *squared-frequency to noise ratio* (SFNR) $k^2/S_n(k\Delta f)$ is maximum.

A more realistic case study for CP in ACGN takes also into account possible power limitations on each subcarrier that prevents from concentrating all of the signal power

onto the edge subcarriers (for AWGN) or on the subcarrier with the best SFNR as above. We have thus the further constraint

$$0 \leq p_k^2 \leq P_{max} < N. \tag{A.37}$$

The solution to this new power allocation problem can be easily found via linear programming:

a. Order the square-frequency-to-noise-ratios $SFNR_k$ from the highest to the lowest; set the currently allocated power to zero; mark all carriers available;

b. Find the available power as the difference between the total power N and the currently allocated power. If it's null, then STOP, else, if it's larger then P_{max}, then put the maximum power P_{max} on the available carrier with the highest SFNR; else put on the same carrier the (residual) available power;

c. Update the currently allocated power by adding the one just allocated, and remove the just allocated carrier from the list of available carriers. If the list is empty, then STOP, else goto 2)

This results in a set of bounded-power subcarriers that gives the optimum power allocation (minimum TDE MCRB) with ACGN.

A.4.4 Algorithms for cognitive positioning

How can the opportunities for CP be exploited? Conventional delay estimators are not directly applicable to a multicarrier signal whose subcarriers are scattered over non-contiguous bands [49]. Let us examine a simple case study of an FBMCM signal in AWGN, where the active subcarriers are concentrated at the two band edges, and all bear the same power, as in Fig. A.4, but with an asymmetric spectrum on positive frequencies only for simplicity. The signal model, after baseband conversion, matched filtering on each subcarrier, sampling at the multicarrier symbol rate $1/T_s = 1/NT_c$ on each subcarrier, and removal of the ranging chip $c^{(k)}[n]$ on each subcarrier (despreading) is

$$\begin{aligned} r^{(k)}[n] = &\sqrt{\frac{2P_T}{M}} g_N(\tau) \exp\{j2\pi k(1+\alpha)\tau/T_s\} \\ &+ IChI^{(k)}(\mathbf{c}, \tau; n] + w^{(k)}[n] \end{aligned} \tag{A.38}$$

where $w^{(k)}[n]$ is the n-th noise component on the k-th subcarrier, whose I/Q (mutually independent) components both have variance $N_0/T_s = N_0/(NT_c)$, and where $ICh I^{(k)}(\mathbf{c}, \tau; n]$ is the interchip-interference term arising on subcarrier k due to the sampling offset τ. Assuming that coarse delay acquisition has already taken place, so that the (residual) timing offset is comparable to the chip time T_c, we can assume in (A.38) $g_N(\tau) \simeq 1$ and $ICh I^{(k)}(\mathbf{c}, \tau; n] \simeq 0$, so that the approximated, simplified signal model turns out to be

$$r^{(k)}[n] = \sqrt{\frac{2P_T}{M}} \exp\{j2\pi k(1+\alpha)\tau/T_s\} + w^{(k)}[n] \tag{A.39}$$

The subcarrier index k runs across the union of the two disjoint bands $B_L = [-N/2, -N/2+M/2-1]$ and $B_R = [N/2-M/2, N/2-1]$, so that the relevant CRB is (A.29).

A sample "cognitive" delay estimator for this signal structure is really simple: we start by computing two subcarrier *phase* estimates, the one on the left-edge section B_L, and the other on the right-edge section B_R, using the standard (low complexity) maximum-likelihood algorithm:

$$\hat{\theta}_L = \angle \sum_{n=0}^{N_m-1} \sum_{k \in B_L} r^{(k)}[n]$$

$$\hat{\theta}_R = \angle \sum_{n=0}^{N_m-1} \sum_{k \in B_R} r^{(k)}[n] \tag{A.40}$$

where the operator $\angle\{\cdot\}$ denotes the phase of its complex-valued argument. Each estimate is derived after observing $M/2$ values in the frequency domain, and repeating such observations for N_m multicarrier symbol periods in time. We also conventionally associate the two phase estimates to the two *center-frequencies* of the left-edge and right-edge sections, respectively, whose mutual frequency distance is equal to $(N - M/2)(1+\alpha)/T_s$. After this is done, we derive the delay estimate as the *slope* of the line that connects the two points $(-N/2+M/4-1/2, \hat{\theta}_L)$ and $(N/2-M/4-1/2, \hat{\theta}_R)$ on the (*frequency,phase*) plane:

$$\hat{\tau} = \frac{T_s}{1+\alpha} \frac{\left|\left|\hat{\theta}_R\right|_{2\pi} - \left|\hat{\theta}_L\right|_{2\pi}\right|_{2\pi}}{2\pi\,(N - M/2)} \tag{A.41}$$

or, in "vector" form

$$\exp\left\{\jmath\, 2\pi \frac{\hat{\tau}(1+\alpha)(N-M/2)}{T_s}\right\}$$

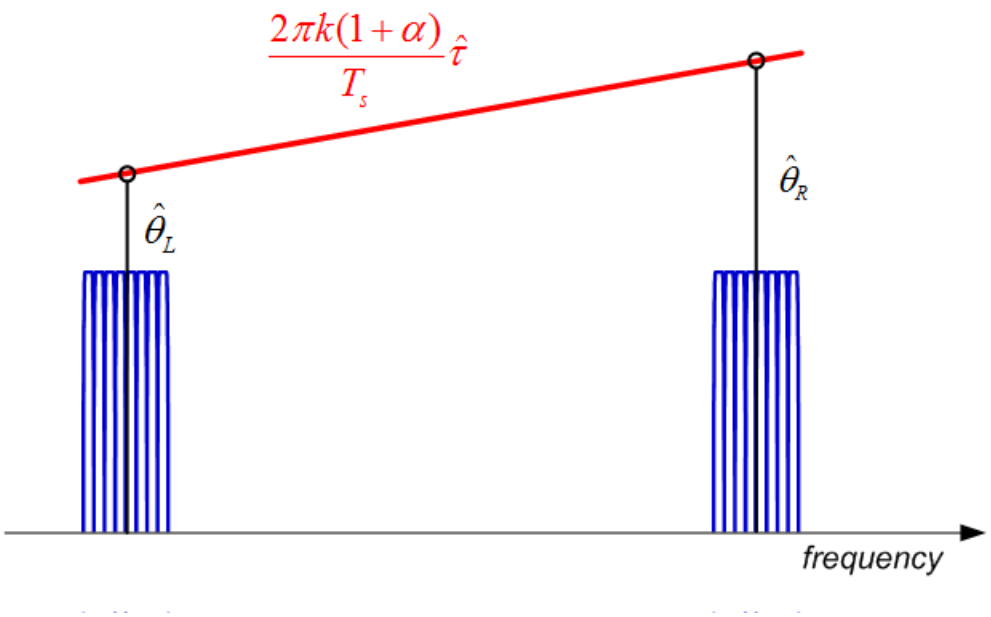

Figure A.5: *Pictorial representation of the DEPE algorithm*

$$= \exp\left\{\jmath\,\hat{\theta}_R\right\} \cdot \exp\left\{-\jmath\,\hat{\theta}_L\right\} \tag{A.42}$$

This simple algorithm, which is pictorially described in Fig. A.5, will be called DEPE (Delay Estimation through Phase Estimation).

The operator $|x|_{2\pi}$ returns the value of x modulo 2π, in order to avoid phase ambiguities, and is trivial to implement when operating with fixed-point arithmetic on a digital hardware.

It is clear that the operating range of the estimator is quite narrow. In order not to have estimation ambiguities, we have to make sure that

$$-\pi \leq |\hat{\theta}_R|_{2\pi} - |\hat{\theta}_L|_{2\pi} < \pi$$

therefore the range is bounded to

$$\frac{|\tau|}{T_c} \leq \frac{N}{2(1+\alpha)\,(N - M/2)} \tag{A.43}$$

When $N \gg M \gg 1$, this interval is basically equal to a fraction $1/(1+\alpha)$ of the chip time, so that the use of the DEPE algorithm is restricted to fine estimation of the residual time offset after coarse acquisition is over.

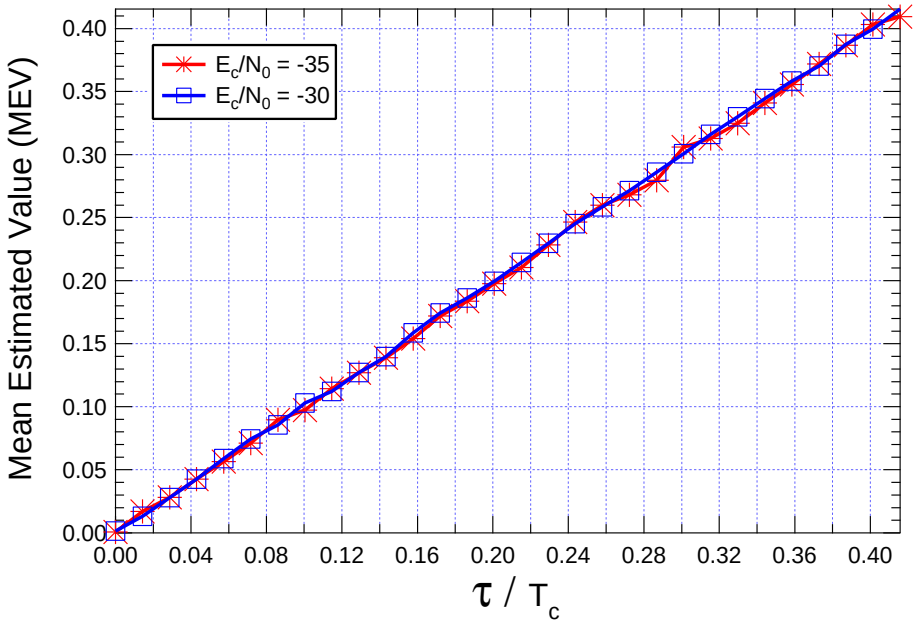

Figure A.6: *Bias curves of the DEPE algorithm*

The DEPE algorithm can be shown to be unbiased, apart from the usual issue of the discontinuity of the $\angle$ function across $\pm\pi$, that may become annoying for low values of C/N_0. Fig. A.6 depicts the normalized MEV (Mean Estimated Value) curves of the DEPE algorithm (i.e., the average estimated value $E\{\hat{\tau}\}$ as a function of the true delay τ for different values of E_c/N_0) as derived by simulation. The parameter N is equal to 2048, whilst $M = 128$. Such curves show, as mentioned above, that the algorithm is unbiased in a broad range around the true value.

It is also easy to evaluate the estimation error variance of the DEPE estimator. It is known that $\hat{\theta}_L$ and $\hat{\theta}_R$ in (A.41) have an estimation variance $\sigma^2_{\hat{\theta}}$ that (for sufficiently high E_s/N_0 achieves its own CRB. Considering the signal expression in (A.39), and that the estimation window for each estimate is equal to $M/2$, the variance is given by

$$\sigma^2_{\hat{\theta}} = \frac{1}{N_m} CRB(\theta) = \frac{1}{N \cdot N_m} \frac{1}{E_c/N_0} \tag{A.44}$$

Since the two phase estimates in (A.41) are independent, we get

$$\sigma^2_{DEPE}(\hat{\tau}) = \left(\frac{NT_c}{1+\alpha}\right)^2 \frac{2\cdot\sigma^2_{\hat{\theta}}}{4\pi^2(N-M/2)^2}$$

$$= \frac{T_c^2}{2\pi^2\left[(N-M/2)(1+\alpha)\right]^2}\frac{N}{N_m\cdot E_c/N_0} \tag{A.45}$$

Assuming $N \gg M \gg 1$, this variance is approximated by

$$= \frac{T_c^2}{2\pi^2 N\cdot N_m(1+\alpha)^2}\frac{1}{E_c/N_0} \tag{A.46}$$

than turns out to be exactly equal to (A.31). This shows that DEPE attains its CRB in most practical situations. We report anyway in Fig. A.7 the ratio between the estimation variance (A.45) and the actual MCRB (A.29). It is seen that this ratio is close to 1 for any practical values of N and M.

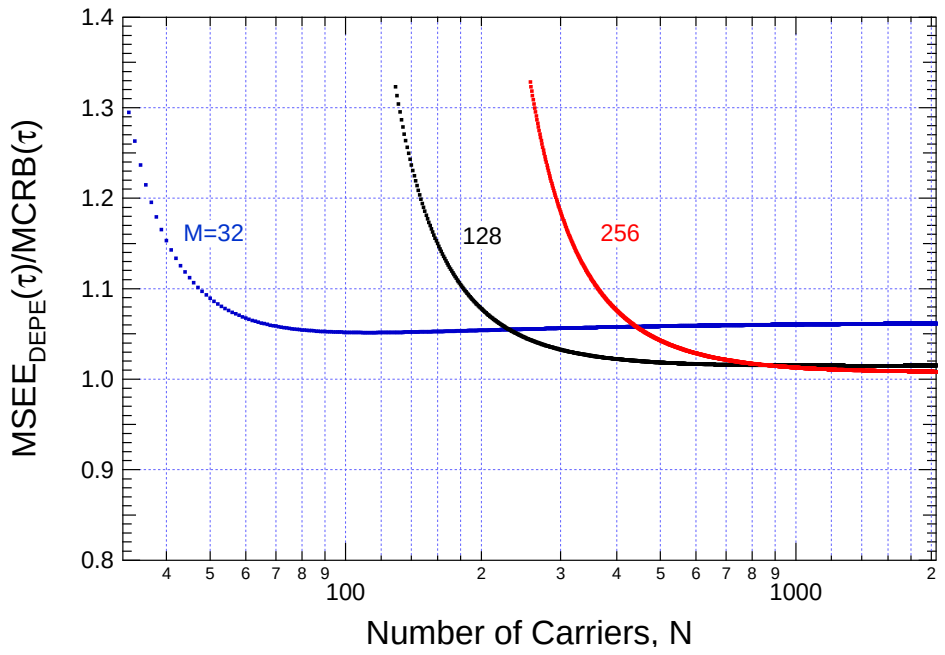

Figure A.7: *Comparison between the RMSEE and the MCRB of the DEPE algorithm - SRRC pulse with* $\alpha = 0.2$

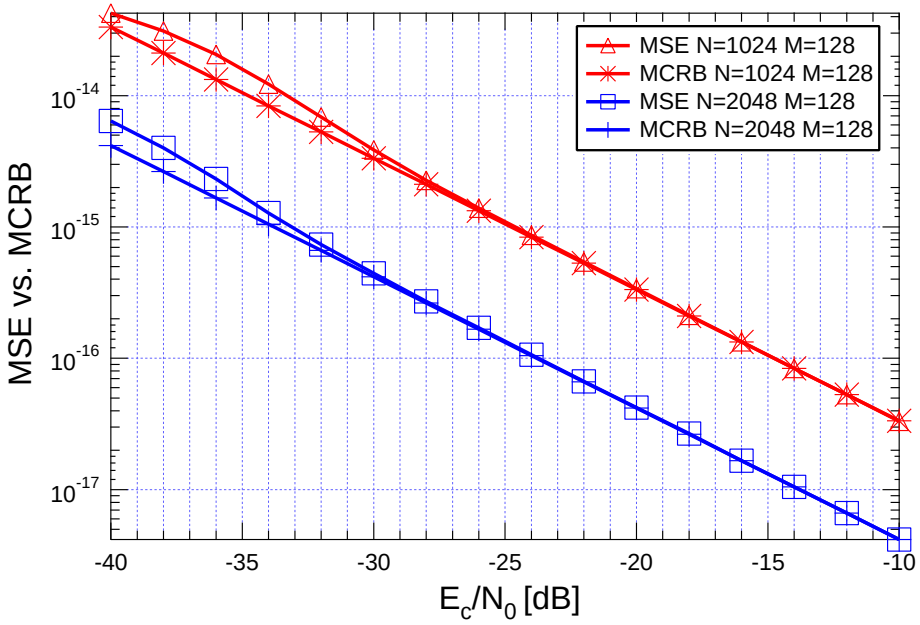

Figure A.8: *Comparison between MCRB and measured MSE of the DEPE algortihm - SRRC pulse with* $\alpha = 0.2$

A.4.5 Simulation results

The performance of the DEPE algorithm has been evaluated by simulation in terms of its *Mean Square Estimation Error (MSEE)*. We used the simplified signal model (A.39) in which the time delay τ satisfies the limitation (A.43). Fig. A.8 shows our simulation results for $N = 1024$ or 2048 and with $M = 128$. As predicted by Fig. A.7, the DEPE algorithm attains its own MCRB (A.29).

The basic strategy of DEPE can be generalized to a ranging signal that spans more than two subbands. The starting point is still the derivation of a single phase estimate for each subband, followed by linear regression to fit a line across the phase values. The estimated signal delay $\hat{\tau}$ is trivially the slope of the regression line. We will not enter here into the details of such computation, but we wish to give a few numerical results. For another simple test case of a multicarrier ranging signal of $M = 128$ active subcarriers split into four equi-spaced and equi-span subbands. The signal format is

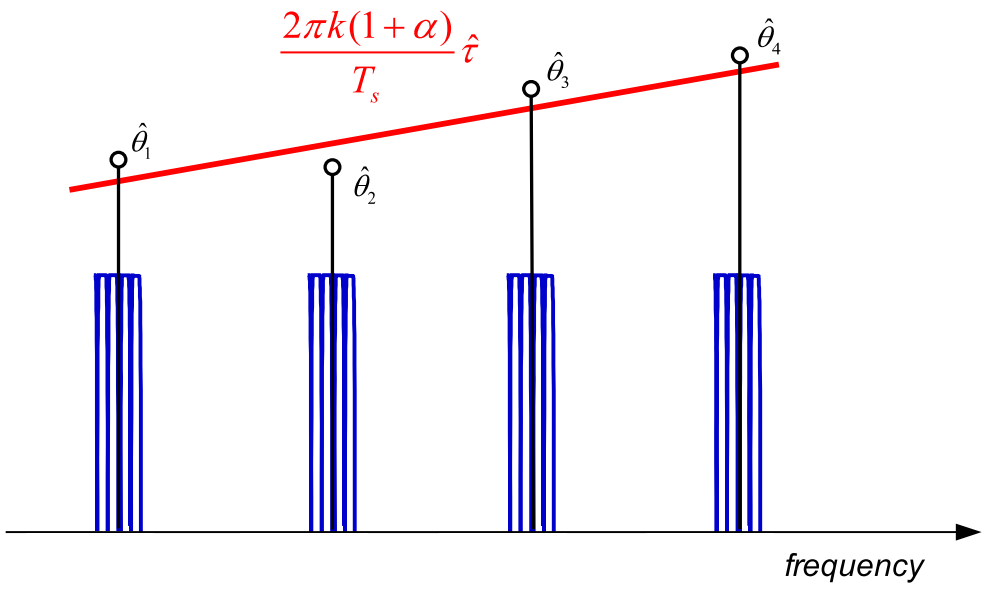

Figure A.9: *Pictorial representation of the extended DEPE algorithm applied on a signal divided into four subbands*

(A.23) with $p_k = \sqrt{N/M}$ for M active subcarriers, and $p_k = 0$ for the remaining $N-M$ elements, so that the relevant MCRB is easily computed by using (A.25). The MSEE performance of the algorithm, which is again sketched in Fig. A.9, are shown in Fig. A.10 for $N = 1024$ and 2048 and $M = 128$. We notice that the extended DEPE algorithm still attains its own MCRB.

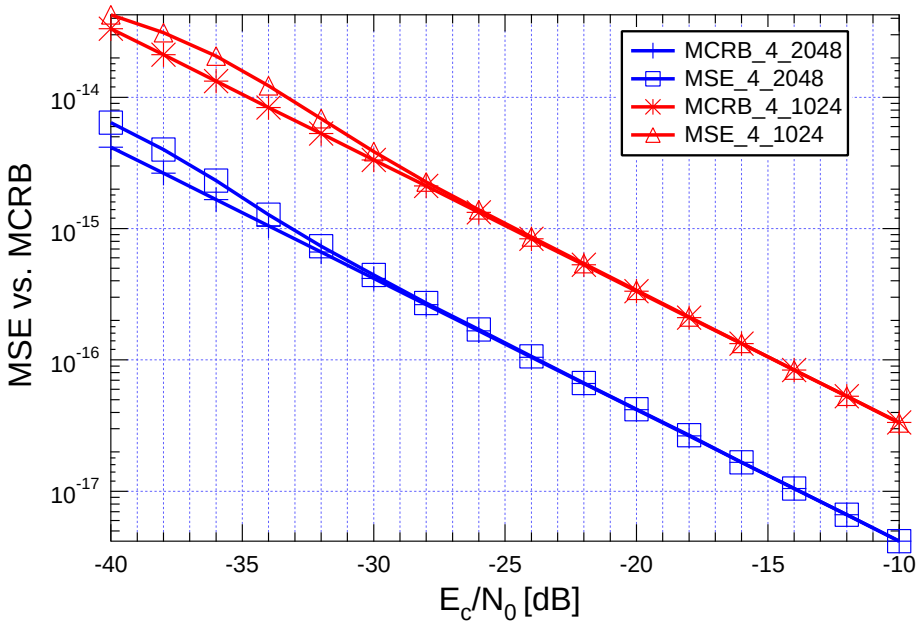

Figure A.10: *Comparison between the MCRB and the simulated MSEE of the extended DEPE algortihm - SRRC pulse with* $\alpha = 0.2$

Bibliography

[1] J. Mitola, "The software radio," in *In Proc. IEEE National Telesystems Conference*, Washington, DC, USA, May 1992, pp. 15–23.

[2] P. Cook and W. Bonser, "Architectural overview of the speakeasy system," *Selected Areas in Communications, IEEE Journal on*, vol. 17, no. 4, pp. 650 –661, apr 1999.

[3] C. A. Gonzalez, C. Dietrich, and J. Reed, "Understanding the software communications architecture," *IEEE Communications Magazine*, vol. 47, no. 9, pp. 50–57, Sept. 2009. [Online]. Available: http://sca.jpeojtrs.mil/

[4] M. Safadi and D. Ndzi, "Digital hardware choices for software radio (sdr) baseband implementation," in *In Proc. ICTTA 06 Information and Communication Technologiese*, Damascus, Syria, May 2006.

[5] D. Cociorva, G., C. G. L. Baumgartner, P. Sadayappan, J. Ramanujam, M. Nooijen, D. E. Bernholdt, and R. Harrison, "Space-time trade-off optimization for a class of electronic structure calculations," *ACM SIGPLAN Notices*, vol. 37, pp. 177–186, 2002.

[6] S. project. [Online]. Available: http://www.sandra.aero

[7] G. Baldini, O. Picchi, M. Luise, T. Sturman, F. Vergari, C. Moy, T. Braysy, and R. Dopico, "The euler project: application of software defined radio in joint security operations," *Communications Magazine, IEEE*, vol. 49, no. 10, pp. 55 –62, oct. 2011.

[8] P. P. C. Middleware. [Online]. Available: http://www.prismtech.com/

[9] S. S. Processing. [Online]. Available: http://www.spectrumsignal.com/

[10] "Ossie, sca-based open source software defined radio." [Online]. Available: http://ossie.wireless.vt.edu/

[11] V. Pellegrini, "Fast memory-based processing in software-defined radios," *PhD Thesis, University of Pisa*, Apr. 2013.

[12] C. D. Walter, "Space/time trade-offs for higher radix modular multiplication using repeated addition," *IEEE Transactions on Computers*, vol. 46, no. 2, pp. 139–141, Feb. 1997.

[13] M. Stamp, "Once upon a time-memory tradeoff."

[14] 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," *Commun. ACM*, vol. 54, pp. 99–107, Jan. 2011.

[15] "Digital video broadcasting (dvb); framing structure, channel coding and modulation for digital terrestrial television," *ETSI EN 300 744 V1.5.1*, Jan. 2009.

[16] V. Pellegrini, M. D. Dio, L. Rose, and M. Luise, "A real time, fully software, etsi dvb-t receiver based on the usrp," in *WSR10*, Karlsruhe, Germany, Mar. 2010.

[17] V. Pellegrini, G. Bacci, and M. Luise, "Soft-dvb: a fully software, gnuradio based etsi dvb-t modulator," in *International Workshop on Software defined Radio (WSR08)*, Karlsruhe, Germany, Mar. 2008, pp. 163–168.

[18] M. D. Dio, "Signal synchronization and channel estimation/equalization functions for dvb-t software-defined receivers," *Master's Thesis, University of Pisa*, Sept. 2009. [Online]. Available: http://etd.adm.unipi.it/theses/available/etd-07292009-191145/unrestricted/sync_chain.pdf

[19] L. Rose, "R-dvb: Software defined radio implementation of dvb-t signal detection functions for digital terrestrial television," *Master's Thesis, University of Pisa*, Apr. 2009. [Online]. Available: http://etd.adm.unipi.it/theses/available/etd-04032009-134746/unrestricted/R_dvb.pdf

[20] V. Pellegrini and M.Luise, "Fully software ofdm modulation in vehicular, highly time-variant channels. an implemented technology and its results," in *International Symposium on Wireless Communication Systems (ISWCS'09)*, Siena, Italy, Sept. 2009, pp. 550–554.

[21] "Ettus research llc website." [Online]. Available: http://www.ettus.com

[22] M. Frigo and S. G. Johnson, "Fftw: An adaptive software architecture for the fft," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Seattle, WA, May 1998, pp. 1381–1384.

[23] J. V. D. Beek, M. Sandell, and P. O. Borjesson, "Ml estimation of time and frequency offset in ofdm systems," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1800–1805, Jul. 1997.

[24] W. Li, X. Youyun, and C. Yueming, "Robust and fast frequency offset estimation for ofdm based satellite communication," *AJournal of Eletronics (CHINA)*, vol. 26, no. 1, pp. 45–51, 2009.

[25] "Ground-based vhf hand-held, mobile and fixed radio transmitters, receivers and transceivers for the vhf aeronautical mobile service using amplitude modulation," *ETSI Std.*, 2011.

[26] "Wimax 802.16e." [Online]. Available: http://standards.ieee.org

[27] "Aeromacs." [Online]. Available: http://www.eurocontrol.int

[28] J. I. Mitola and G. J. Maguire, "Cognitive radio: making software radios more personal," *Personal Communications, IEEE*, vol. 6, no. 4, pp. 13–18, Aug. 1999.

[29] H. Celebi and H. Arslan, "Cognitive positioning systems," *IEEE J COM*, vol. 6, no. 12, Dec. 2007.

[30] M. P. Wylie-Green, "A multi-carrier communication technique for interference-free spectrum sharing in cognitive radios," in *Waveform Diversity and Design Conference, 2009 International*, Aug. 2009.

[31] T. Luo, T. Jiang, W. Xiang, and H. Chen, "A subcarriers allocation scheme for cognitive radio systems based on multi-carrier modulation," *Wireless Communications, IEEE Transactions on*, vol. 7, no. 9, pp. 3335–3340, Sept. 2008.

[32] H. Celebi and H. Arslan, "Utilization of location information in cognitive wireless networks," *Wireless Communications, IEEE*, vol. 14, no. 4, pp. 6–13, Aug. 2007.

[33] P. Jia, M. Vu, and T. Le-Ngoc, "Capacity impact of location-aware cognitive sensing," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, Nov. 2009, pp. 1–6.

[34] C. Mayorga, F. della Rosa, S. Wardana, G. Simone, M. Raynal, J. Figueiras, and S. Frattasi, "Cooperative positioning techniques for mobile localization in 4g cellular networks," in *Pervasive Services, IEEE International Conference on*, 2007, pp. 39–44.

[35] S. Frattasi, M. Monti, and R. Prasad, "Cooperative positioning techniques for mobile localization in 4g cellular networks," in *A cooperative localization scheme for 4G wireless communications*, 2006, pp. 287–290.

[36] H. L. V. Trees, *Detection, Estimation and Modulation Theory.* New York, NY: Wiley, 1968.

[37] H. Cramér, *Mathematical methods of statistics.* New York, NY: Princeton Univ. Press, 1946.

[38] S. M. Kay, *Fundamentals of statistical signal processing: Estimation theory.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[39] U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers.* New York: Plenum Press, 1997.

[40] T. Alberty, "Frequency domain interpretation of the Cramér-Rao bound for carrier and clock synchronization," *IEEE J COM*, vol. 43, no. 2-3-4, Feb.-Apr. 1995.

[41] A. N. D'Andrea, U. Mengali, and R. Reggiannini, "The modified Cramér-Rao bound and its application to synchronization problems," *IEEE J COM*, vol. 42, no. 2-4, pp. 1391–1399, Feb.-Apr. 1994.

[42] A. Mallinckrodt and T. Sollenberger, "Optimum pulse-time determination," *IRE Trans.*, no. PGIT-3, pp. 151–159, Mar. 1954.

[43] M. Skolnik, *Introduction to radar systems.* McGraw-Hill, 1980.

[44] E. D. Kaplan, *Understanding GPS: Principles and applications.* Boston, MA - London, UK: Artech House, 1996.

[45] J. W. Betz, "Binary offset carrier modulations for radionavigation," *J. Inst. Navigation*, vol. 48, no. 4, pp. 227–246, Winter 2001-2002.

[46] Y. Gao, Z. Gao, W. Zhu, and X. Yang, "The research on the design of filter banks in filtered multitone modulation," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 1, 2005, pp. 584–588.

[47] F. Zanier, G. Bacci, and M. Luise, "Criteria to improve time-delay estimation of spread spectrum signals in satellite positioning," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 3, no. 5, pp. 748–763, Oct. 2009.

[48] A. Zeira and A. Nehorai, "Frequency domain cramer-rao bound for gaussian processes," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 6, pp. 1063–1066, Jun. 1990.

[49] F. Kocak, H. Celebi, S. Gezici, K. A. Qaraqe, H. Arslan, and H. V. Poor, "Time-delay estimation in dispersed spectrum cognitive radio systems," *EURASIP J. on Adv. in Signal Processing*, vol. 2010, no. Special Issue on Advanced Signal Processing for Cognitive Radio Networks.

List of Publications

Book Chapters

- M. Di Dio et al., "Innovative Signal Processing Techniques for Wireless Positioning," chapter 5 of "Satellite and Terrestrial Radio Positioning Techiniques: A signal processing perspective", edited by D. Dardari, E. Faletti, M. Luise, Elsevier.

International Journals

- A. Mallat, J. Louveaux, L. Vandendorpe, M. Di Dio and M. Luise, "Discrete Fourier transform based TOA estimation in UWB systems," submitted to *EURASIP Journal of Wireless Communications and Networking 2012*, Mar. 2012.
- V. Pellegrini, M. Di Dio, L. Rose and M. Luise, "On Accelerating Signal Processing Functions in Software Defined Radio through Efficient use of Available Memory Resources," submitted to *IEEE Transactions on Signal Processing*.
- V. Pellegrini, M. Di Dio, L. Rose and M. Luise, "On the usage of Memory Resources for Achieving Real-Time in Wideband Embedded SDRs," submitted to *EURASIP Journal on Embedded Systems*.

International Conferences

- M. Di Dio, D. Bolognesi, M. Francone and M. Luise, "On Future Aeronautical Communications Standards: a Real-Time, Fully-Software AeroMACS waveform

implementation based on the SCA-compliant OSSIE/USRP2 platform," in *Proc. of DMD track of SDR-WinnComm 2013*, Washington, USA, Jan. 2013.

- M. Di Dio, A. Morelli and M. Luise, "A Real-Time, Fully-Software Aeronautical VHF tx/rx waveform based on the SCA-compliant OSSIE/USRP2," in *Proc. of 7th Karlsruhe Workshop on Software Radios (WSR12)*, Karlsruhe, Germany, Mar. 2012.

- A. Mallat, J. Louveaux, L. Vandendorpe, M. Di Dio, et al., "Discrete Fourier transform based TOA estimation in UWB systems," in *Proc. of JNCW 2011 NEWCOM++/COST 2100 joint workshop*, Paris, France, Mar. 2011.

- M. Di Dio, M. Luise, F. Zanier, et al., "On Delay Estimation with Multicarrier Signals and its relation with Cognitive Positioning," in *Proc. of JNCW 2011 NEWCOM++/COST 2100 joint workshop*, Paris, France, Mar. 2011.

- V. Pellegrini, M. Di Dio, L. Rose and M. Luise, "Computation Potential of the MA technique," in *Demo session held at IEEE Global Communications Conference 2010*, Miami, Florida, USA, Dec. 2010.

- V. Pellegrini, M. Di Dio, L. Rose and M. Luise, "On the Computation/Memory Trade-Off in Software Defined Radios," in *Proc. IEEE Global Communications Conference 2010*, Miami, Florida, USA, Dec. 2010.

- V. Pellegrini, M. Di Dio, L. Rose and M. Luise, "A real-time, fully-software receiver for DVB-T signals based on the USRP," in *Proc. Karlsruhe Workshop on Software Radios (WSR)*, Karlsruhe, Germany, Mar. 2010.

National Conference

- V. Pellegrini, M. Di Dio, L. Rose, and M. Luise, "On the Computation/Memory Trade-Off in Software Defined Radios," in *Riunione Annuale GTTI*, Brescia, Italy, June 2010