Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Electronic Engineering

Analysis and implementation on Zynq FPGA of a pulse blanking core for detection and mitigation of interference in GNSS context

Master Candidate

Nicola Rodighiero

Student ID 2053733

Supervisor

Prof. Daniele Vogrig

University of Padova

Co-supervisors

Eng. Gianluca Gastaldello

Eng. Mauro Fregolent

Qascom srl

Academic Year
2023/2024

Abstract

Nowadays, Global Navigation Satellite Systems (GNSS) play an important role in various applications, ranging from navigation and positioning to timing synchronization. However, due to the nature of those signals, the reliability of GNSS receivers is often compromised under harsh conditions such as in the presence of interference sources. For this reason, this master thesis contains a brief overview on GNSS, interferences and on some mitigation techniques and then focuses on the analysis, design and implementation of a Pulse Blanking (PB) core on a Zynq Field-Programmable Gate Array (FPGA) with the purpose of detection and mitigation of some of those interferences.

Sommario

Al giorno d'oggi, i Global Navigation Satellite Systems (GNSS) giocano un ruolo importante in vari applicativi che vanno dalla navigazione, al posizionamento alla temporizzazione. Tuttavia, per via della natura di questi segnali, l'affidabilità dei ricevitori GNSS è spesso compromessa in condizioni non ottimali come in presenza di sorgenti di interferenza. Per questo motivo, questa tesi contiene una panoramica sul mondo GNSS, sulle interferenze e su alcune delle tecniche di mitigazione per poi concentrarsi nell'analisi, il design e l'implementazione di un core di Pulse Blanking (PB) su Zynq Field-Programmable Gate Array (FPGA) con lo scopo di rilevare e mitigare gli effetti di alcune di queste interferenze.

Contents

List of Figures

List of Tables

List of Acronyms

ADC	analog-to-digital
ADI	Analog Devices Inc.
AGC	automatic gain control
ANF	adaptive notch filtering
AXI	Advance eXtensible Interface
BE	Blanking Events
BPSK	Binary Phase Shift Keying
BR	Blanking Rate
BT	Blanking-Threshold
C/A	coarse acquisition code
CAF	Cross-Ambiguity Function
CDC	Clock Domain Crossing
CDMA	Code Division Multiple Access
CNT	Time interval
CW	Continuous Wave
DFT	discrete Fourier transform
DMAC	Direct Memory Access Controller
DSSS	direct-sequence spread spectrum

EB	End Blanking
EN	ENable parameter
FBPB	Filter Bank Pulse Blanking
FDAF	Frequency Domain Adaptive Filtering
FF	Flip Flop
FFT	Fast Fourier Transform
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GNSS	Global Navigate Satellite System
GPS	Global Positioning System
HDDM	High-Rate DFT-Based Data Manipulator
IDFT	inverse discrete Fourier transform
IF	intermediate frequency
IIR	Infinite Impulse Response
I	In-phase
KLT	Karhunen-Loève transform
LSB	Least Significant Bit
LVDS	Low Voltage Differential Signal
MO	MOde parameter
MSB	Most Significant Bit
MUX	Multiplexer
NF	Notch Filter
PB	pulse blanking

PL	Programmable-Logic
PRI	Pulse Repetition interval
PRN	Pseudo-Random noise
PS	Processing System
PSD	Power Spectral Density
P(Y)	encrypted precision code
Q	Quadrature
RF	Radio Frequency
RHCP	Right-Hand Circularly Polarized
SB	Start Blanking
SDR	Software-Defined-Radio
SOM	System-On-Module
SPI	Serial Peripheral Interface
UHF	Ultra High Frequency
VHDL	Very-High-Speed-Integrated-Circuits (VHSIC) Hardware Description Language

1 Introduction

This thesis is the result of a collaboration with Qascom S.R.L., a company that provide space and cybersecurity engineering services, advanced navigation systems, simulation and test systems, signal intelligence and electromagnetic warfare.
In this context I had the opportunity to collaborate in developing a minor core thought to be integrated in an FPGA inside a Global Navigate Satellite System (GNSS) receiver. This core will be used to enhance the already great robustness against interference.
This thesis have, then, the aim of reporting the developing process, design decisions and results that this internship project produced.
Specifically chapter 2 and chapter 3 contain the starting theoretical part that help to understand the environment and constraint in which the core have to operate and what are interference signals that disturb GNSS communication. Then, in chapter 4 some mitigation techniques have been presented. Moreover, chapter 5 gives the reason why, between all mitigation methods the Pulse Blanker have been chosen.
From there, chapter 6 gives all the details about hardware environment, core functioning principles and design concept that has been used for the implementation of the core itself.
Finally, chapters 7, 8 and 9 report how this design have been verified, tested and how it perform against interference.

2 GNSS signals

In order to understand and mitigate interference in the context of Global Navigate Satellite System (GNSS) signals, it is crucial to know the characteristics of signals and data transmitted from satellites and received by antennas. In this chapter an overview of Global Positioning System (GPS) signals generation and their most important properties is provided. Obviously, other types of GNSS signals, such as Galileo's ones, have different characteristics. However, information that are here provided are fundamental to understand what kind of signals are involved.

2.1 Signals and data

GPS signals are transmitted on two radio frequencies in the UHF band (from 500 MHz to 3 GHz). These frequencies are referred as L1 and L2 and are derived from a common frequency, f_0= *10.23 MHz*:

$$f_{L1} = 154 f_0 = 1575.42\ MHz$$

$$f_{L2} = 120 f_0 = 1227.60\ MHz$$

Signals are composed of the following three parts:

- ***Carrier***: The carrier wave with the frequency f_{L1} or f_{L2}
- ***Navigation data***: The navigation data contain information regarding satellite orbits. Those information are uploaded to all satellites from the ground stations in the GPS Control Segment.

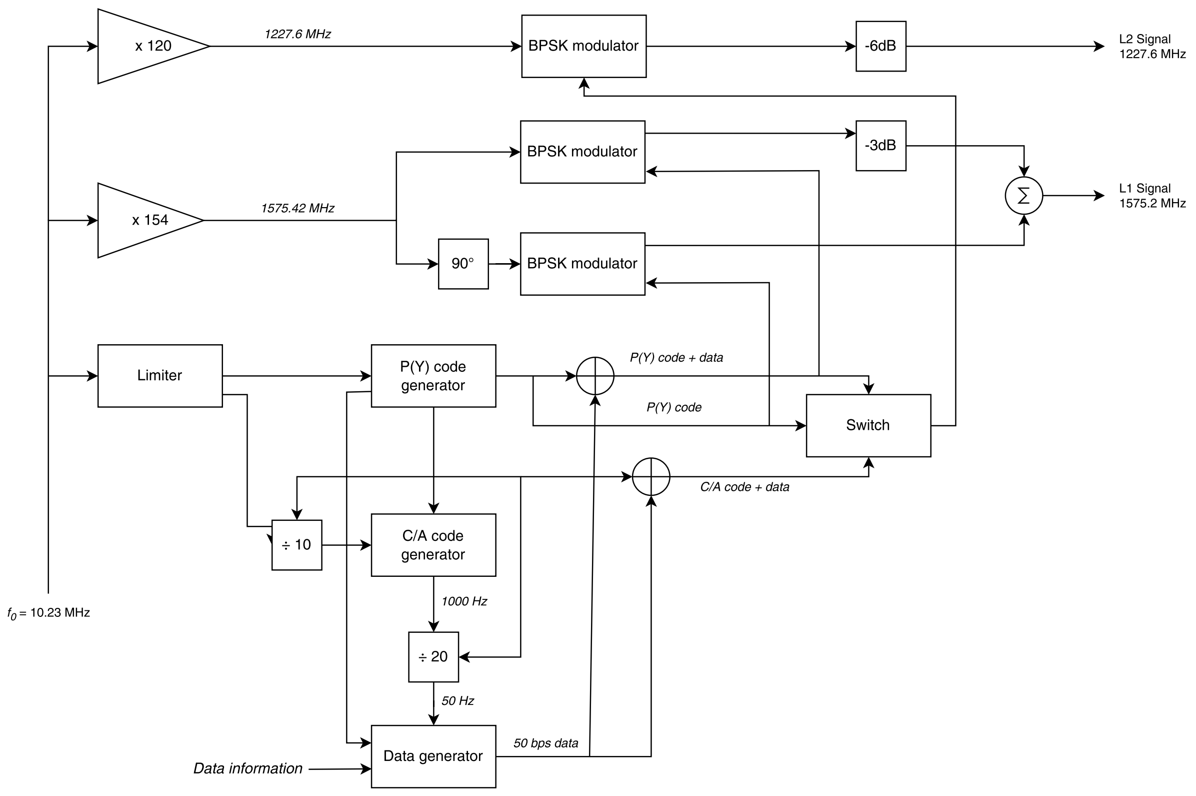


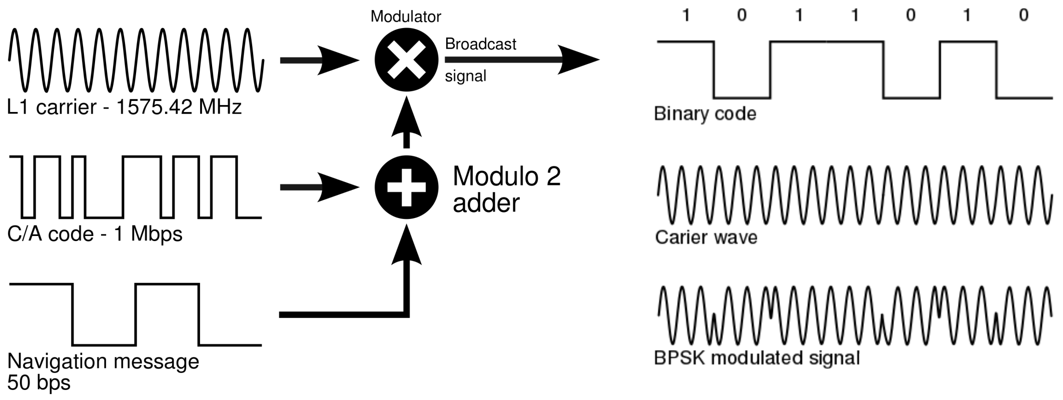
Figure 2.1: Generation of GPS signals at the satellites

- *Spreading sequence*: Each satellites have two unique spreading sequences. The first one is the coarse acquisition code (C/A). It is a 1023 chips long sequence and it is repeated each ms giving a chipping rate of 1.023 MHz. A chip corresponds to a bit but is called differently to emphasize that it does not hold any information. The second one is the encrypted precision code (P(Y)). It is a longer code ($\approx 2.35 \cdot 10^4$ chips) with a chipping rate of 10.23 MHz. It repeats itself each week starting at the beginning of the GPS week which is at Saturday/Sunday midnight.

2.2 GPS SIGNAL SCHEME

In figure 2.1 the sequence of operations used to generate satellite GPS signals is displayed.
In particular, the clock signal has a frequency of 10.23 MHz and is used, once multiplied, to generate L1 and L2 carrier signals. Instead, it is limited to generate P(Y) and C/A codes.
After that, those sequences are combined with the navigation data through modulo-

2 adders. The obtained signal is than supplied to modulators. Binary Phase Shift Keying (BPSK) method is used. This implies that the carrier is instantaneously phase shifted by 180° at the time of chip change. When a navigation data bit transition occurs the phase of the resulting signal is also phase-shifted by 180°.
So, overall we have that for GPS the code length is 1023 chips, 1.023 MHz chipping rate (1 ms period time) and 50 Hz data rate (20 code periods per data bit).
Additional figures follow to better understand how the signal, in the L1 case, is structured and how BPSK modulation behave.



(a) Signal structure GPS L1 case (b) Effect of BPSK modulation

Figure 2.2: GPS L1 signal structure

Spreading sequences, unique for each satellite, are referred to as Pseudo-Random noise (PRN) sequences since are deterministic sequences with noise like properties. Those sequences have two important properties:

- nearly no cross correlation
- nearly no correlation except for zero lag

Those characteristics allow to associate each sequence to a satellite and facilitate the code tracking and acquisition process.

2.3 Doppler frequency shift

In GPS we faced with the Doppler frequency shift caused by the motion of the transmitter (satellite) relative to the GPS receiver. This shift affects both the acquisition and tracking of the GPS signal.

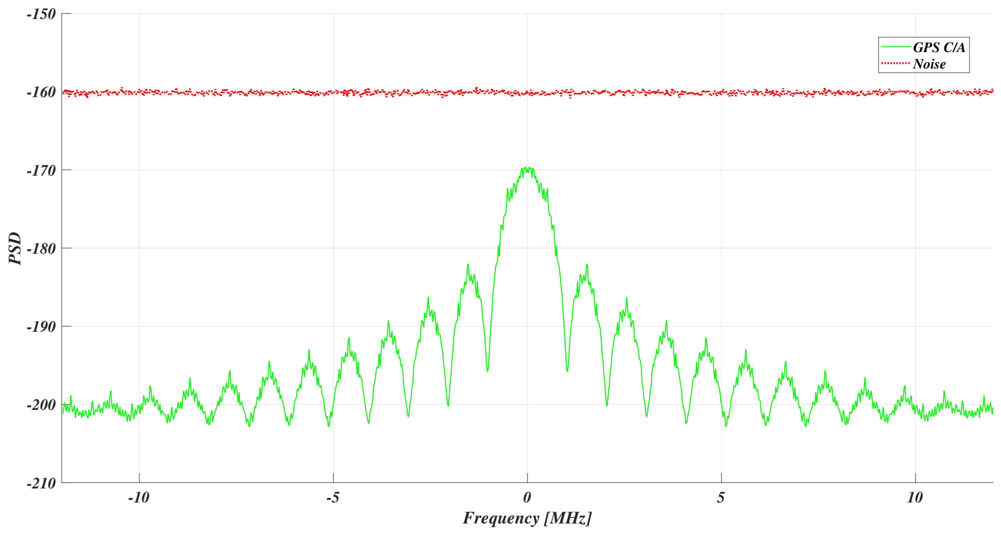


Figure 2.3: Frequency domain depiction of the GPS signal and thermal noise power. B = 24 MHz

The Doppler frequency shift for the L1 frequency can reach up to ± 10 kHz while, due to the slow chip rate, on the C/A code can reach up to ± 6.4 kHz.
Doppler frequency shift on the coarse acquisition code can cause misalignment between the received and the locally generated codes and cause high tracking impediments if not correctly estimated.

2.4 GNSS antenna and front-end

The purpose of this section is to provide some insight into how satellite signals propagating through space can transmit digital data. This is done, of course, via GNSS antenna/front-end. The focus, here, is on the narrowband GNSS L1 signals.

2.4.1 GNSS antenna

The process begins with the GNSS signal, propagating through space, which is incident on a user's GNSS antenna. This induces a voltage within the element that is extremely weak.
Let the Boltzmann's constant be denoted by $k = 1.38{\cdot}10^{-23} J/K$, the absolute temperature

by t in $^\circ K$ and the equivalent noise bandwidth by B in Hz, then, the thermal noise power is:

$$P_{ThermalNoise} = ktB \tag{2.1}$$

Considering typically used bandwidth the received GPS signal power is actually below that of the thermal noise floor, as defined by equation 2.1 and illustrated in figure 2.3. This is quite unique in the field of radio transmission and is a feature of the Code Division Multiple Access (CDMA) spread spectrum signal that requires an appropriate signal processing to acquire and process the signal.
For this reason in the case of the antenna, there is an extensive set of parameters associated that describe its performance. Three fundamental parameters are *frequency/bandwidth*, *polarization*, and *gain pattern*.
The antenna will be design to induce a voltage from radio waves propagating at the GNSS L1 frequency and accommodate the appropriate bandwidth of the desired signal.
Polarization refers to the orientation of the electric field from the radio frequency transmission. Received GNSS signals are Right-Hand Circularly Polarized (RHCP).
The antenna pattern describes the directivity of the antenna. The most basic idea for the antenna pattern would be one that receives signals equally from all directions (an isotropic antenna). However, such a uniform gain pattern does not make sense for GNSS since the signal source, satellites, are overhead for the most applications. So, other pattern like the hemispherical one are preferred.

2.4.2 Filter

The first component within the RF path after the antenna is a filter. Specifically a bandpass filter in order to provide additional frequency selectivity.
Ideally, the antenna would only induce voltages for precisely the frequency band of interest but in reality also frequency components outside that range are captured.
Of course, also filters are not ideal and, for this reason they are characterized by their insertion loss (the attenuation of the desired frequency components) and their bandwidth (typically a 3 dB bandwidth is specified).
Even if filters are not ideal, due to the nature of the GNSS signals, it is very important to try to eliminate any high-power, out-of-band signal sources that could enter the front end and saturate later-stage components.

2.4.3 Amplifier

The goal of the amplifier is to raise the extremely weak incident signal to a level practical for analog-to-digital conversion. Thus, the amount of amplification is based on the specific ADC.
Note that the ideal amplifier would only increase the amplitude of the signal. However, any amplifier will not only increase the amplitude but also add noise to the resulting signal.
Fundamental parameters used to describe an amplifier are:

- ***gain***: usually expressed in dB and often assumed constant over a
- ***specified frequency range***
- ***noise figure***: again, usually specified in dB and indicative of the amount of noise that will be added to the signal being amplified.

Also, the amplification that is usually needed is very high. For this reason it would be unusual to have a single amplifier capable of such gain. Typically, a cascaded stage amplifiers structure is used.

2.4.4 Mixer/Local Oscillator

The basic function of the mixer/local oscillator combination is to translate the input 1575.42 MHz RF carrier to a lower intermediate frequency (IF) and preserve the modulated signal structure. The most obvious reason for this is to bring the frequency to usable ranges in which to operate on the signal and in particular to perform the analog-to-digital conversion.

2.4.5 ADC Converter

The final component in the front-end path is the analog-to-digital converter. This device is responsible for the conversion of the analog signal to digital samples.
Keys parameters to consider are the *number of bits, the maximum sampling frequency, the analog input bandwidth,* and *the analog input range*.
The final amplifier in many GNSS front-end designs will be a variable gain with a feedback signal resulting from processing implemented after the ADC. This is implemented and known within most GNSS receivers as automatic gain control (AGC). The goal of the front-end is to exercise all available bits with the ADC. Thus, if the gain is insufficient to do so and this is determined by monitoring the sampled data stream,

the gain can be increased. Alternatively, if the gain is too high such that the outer ADC bins have an overwhelming number of samples, then the gain can be decreased.

2.4.6 Resulting sampled data

In summary, the scope of the front-end is to adapt the voltage generated by the incident signal to the antenna for sampling by the ADC. In order to accomplish this for most ADCs there are three basis functions which must be accomplished. These are *amplification*, *frequency translation/downconversion*, and *filtering*. This prepare the signal for analog-to-digital conversion, which results in the samples to be later processed.

2.5 GNSS receiver operation overview

Once digital samples are provided, further operations such as acquisition and tracking can take place.
In particular, those operations for satellite navigation systems are based on a channelized structure. This section provides an overview of a receiver channel and the processing that occurs.
Before allocating a channel to a satellite, the receiver must know which satellites are currently visible. There are two common ways of finding the initially visible satellites:

- ***warm start***: the receiver combines information in the stored data and the last position computed by the receiver and then compute which satellites should be visible
- ***cold start***: the receiver does not rely on any stored information. Instead, it starts from scratch searching for satellites.

An acquisition search through all satellites is quite time-consuming. That is, in fact, the reason why a warm start is preferred if possible.
In figure 2.4 an overview of one receiver channel is represented.

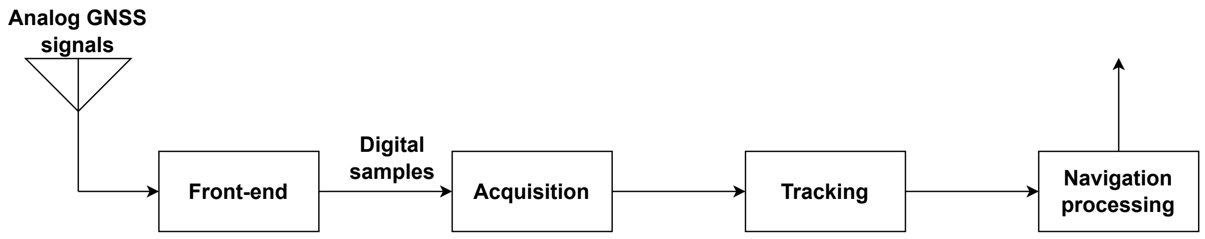


Figure 2.4: Receiver channel

The analog signal captured by the antenna is then amplified, filtered and digitalized by the front-end. The acquisition gives rough estimates of signal parameters. These parameters are refined by the tracking. Finally, the navigation data can be extracted and processed

2.5.1 Acquisition

The purpose of the acquisition is to identify all satellites visible to the user. If a satellite is visible, the acquisition must determine:

- ***Frequency***: As seen previously the frequency of the signal from a specific satellite can differ from its nominal value (that is the intermediate frequency in case of downconversion) due to Doppler frequency shift.
- ***Code phase***: The code phase denotes the point in the current data block where the C/A code begins.

For GPS signal acquisition many different methods have been used and in all of them the C/A code correlation properties are especially important (see section 2.2). Here follow a description of one of the simplest methods that has not the best computational performances but with its simplicity helps to understand what are the steps needed and the reason behind them.
The received signal is a combination of signals from all visible satellites. When acquiring a satellite the incoming signal is multiplied with the local generated C/A code corresponding to that satellite. Due to the correlation characteristics of C/A codes that means that signals from other satellites are removed by this procedure. For the same reason, to avoid removing the desired signal component the logically generated C/A code must be properly aligned in time (must have the correct code phase).
Moreover, the signal must then be mixed with a locally generated carrier wave to remove the carrier of the received signal. To achieve that the frequency of the locally generated signal must be close to the signal carrier frequency. As mentioned previously, the frequency can change considerably from the nominal value due to Doppler

frequency shift for example.
It is now clear that acquisition process is quite similar to a search algorithm. In the interested bandwidth with the nominal value as center for each frequency 1023 different code phases are tried (since C/A code is 1023 chips long).
After the multiplication with the locally generated code and after mixing with the locally generated carrier wave all signal components are squared and summed providing a numerical value as shown in the following diagram.

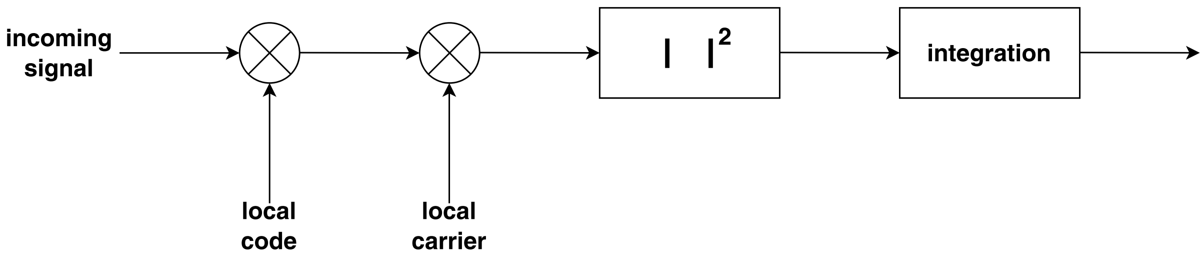


Figure 2.5: Acquisition search procedure

When all possibilities for code phase and frequency are tried, a search for the maximum value is performed. If this value exceeds a determined threshold then the satellite is acquired with the corresponding frequency and phase shift. The following figure shows a typical successful acquisition plot for a satellite. The peak location is related to a C/A code phase and a frequency of the signal.

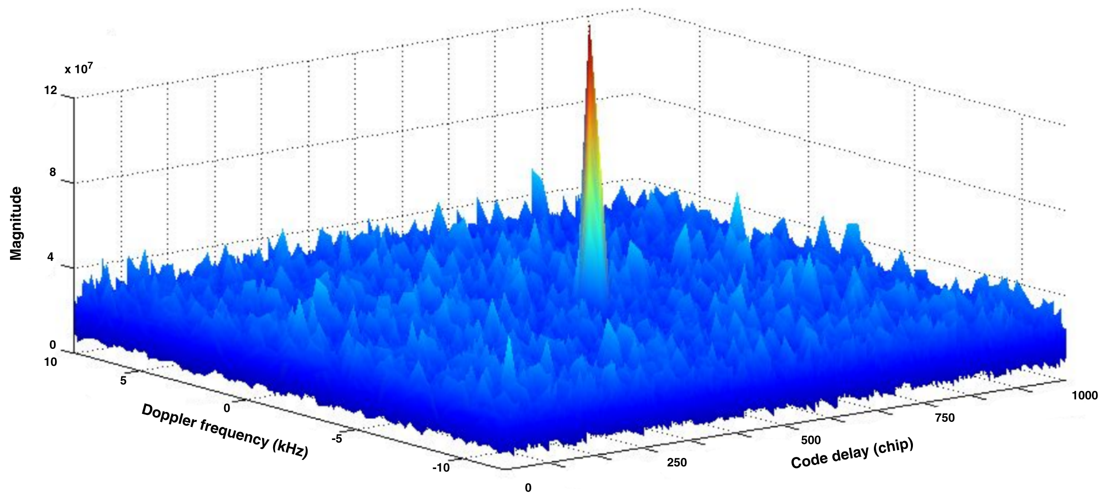


Figure 2.6: Acquisition plot

2.5.2 Tracking

The main purpose of tracking is to refine the coarse values of the code phase and frequency and to keep track of these as the signal properties change over time.
Tracking contains two parts, code tracking and carrier frequency/phase tracking.
This operation is running continuously to follow the changes in frequency as a function of time. If the receiver loses track of the satellite, a new acquisition must be performed for that particular satellite.

2.5.3 Navigation and data extraction

When the signal is properly tracked, the C/A code and the carrier wave can be removed from the signal, only leaving the navigation data bits. The value of the navigation data bit is found by integrating over a navigation bit period of 20 ms (that is the data period).
Thanks to those data and the calculation of the pseudorange (based on the time of transmission from the satellite and the time of arrival at the receiver) the computation of the receiver position can finally take place.

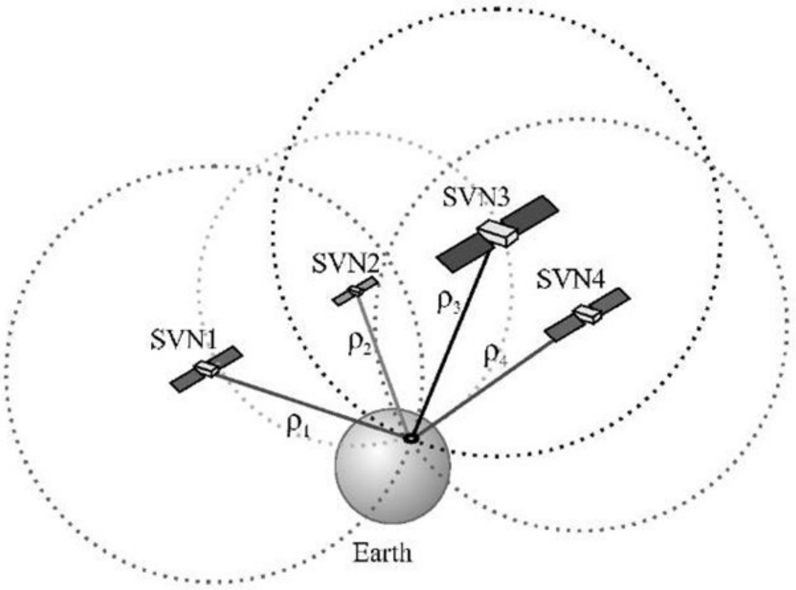


Figure 2.7: The basic principle of GNSS positioning

With known position of four satellites SVN_i and signal travel distance ρ_i the user position can be computed.

3 Interference

Radio frequency signals form any undesired source that are affecting GNSS receivers are considered interference. Even if navigation signals have a direct-sequence spread spectrum (DSSS) signal structure that gives them a certain robustness against interference, this type of signals are received by the antenna at a very low power level, so, RF interference can cause decreased or loss of accuracy, reliability and availability of signals.
Usually, RF interference is categorized into narrowband or wideband depending on whether its bandwidth is large or small relative to the bandwidth of the target GNSS signal but they can also be classified as intentional, unintentional and natural interference according to their sources.

3.1 Unintentional interference

The rapid growth of the wireless telecommunication sector has made the spectrum very crowded and quite saturated. For this reason, transmission of other signals in the GNSS band proximity can cause unintentional interference. Some sources can be medical equipment, cellular communication, radio transmission, Wi-Fi, radars and many more.

3.2 Intentional interference

As the name suggest, intentional interference is a type of RF signals artificially generated with the aim of disturb GNSS communication. Some reasons for which such techniques are used to disturb or destroy GNSS signals are for military purposes or to defend one's own privacy (even if this is a non-legal practice).
The intentional interference sources can be divided into three main categories: jamming, spoofing and meaconing.
Among them jamming is the most common type. In simple terms, jamming occurs due to transmission of high-decibel radio frequencies close to the GNSS frequency bands. This tend to overload the receivers to such an extent that they lose the Coarse Acquisition (C/A) lock on the satellites.
Instead, spoofing is a deceptive interference which tries to mislead its target from true navigation. In particular, a fake GNSS signal is generated and the receiver will consider the counterfeit signal a real one.
Finally, meaconing consist in receiving, delaying and re-broadcasting the GNSS signal in the same frequency as the real signal to confuse the target receiver.

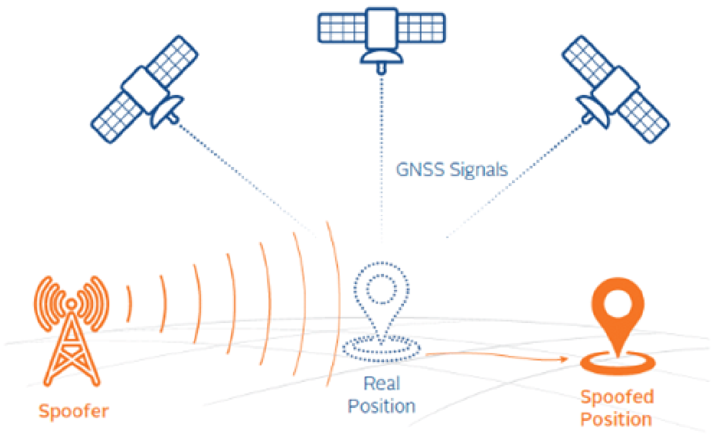


Figure 3.1: Spoofing

3.3 Natural interference

Interference is not only caused by human activities but can be also derive from natural phenomenon like ionospheric scintillation or solar bursts. Ionospheric scintillation is the physical phenomena affecting radio waves coming from the space through the

Ionosphere. Such disturbance is caused by ionospheric electron-density irregularities. Under scintillation, GNSS signals are affected by amplitude and phase variations which mainly compromise the synchronization stage of the receiver.
Furthermore, when strong solar flare events occur an increase in the X-ray, enhanced UV fluxes and solar radio bursts (SRBs) can be observed. The SRBs cover a large range of frequencies, giving rise to signals fades in the GNSS carrier-to-noise ratio and fluctuations in its amplitude and phase.

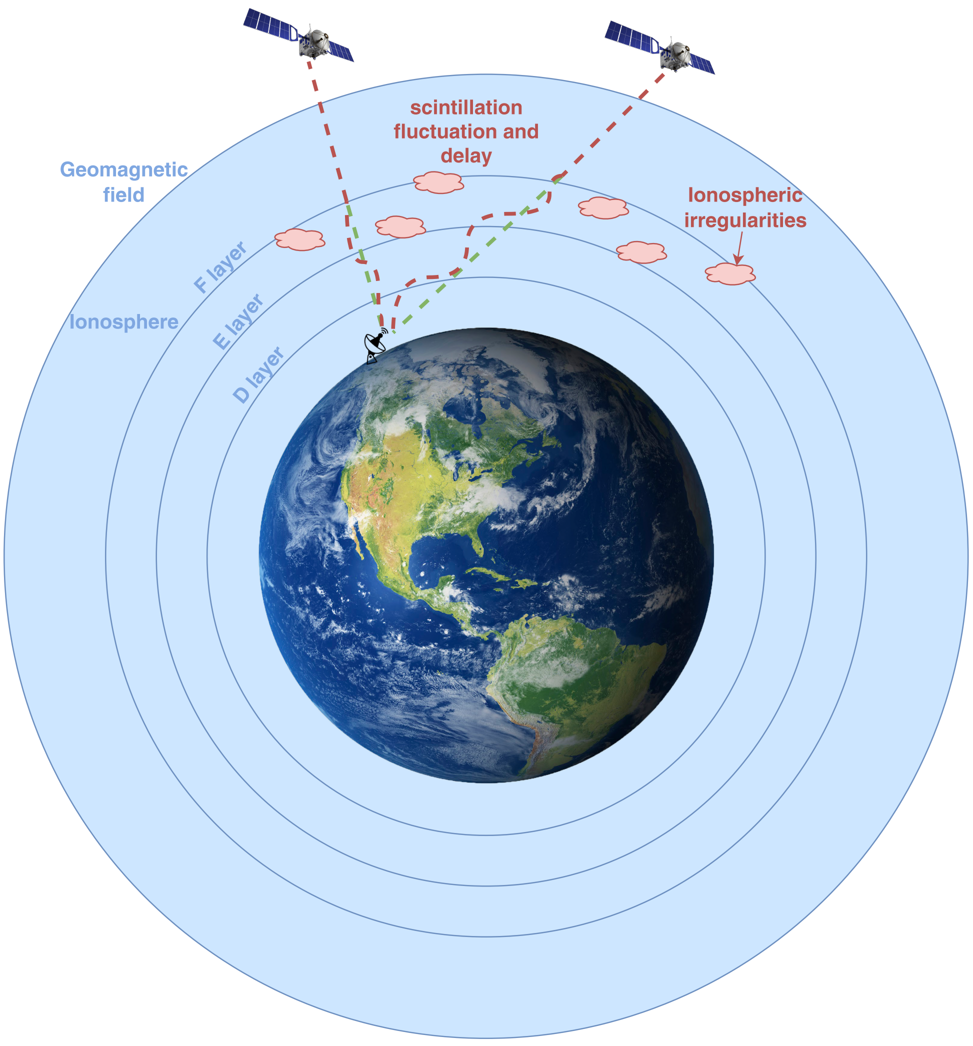


Figure 3.2: Ionospheric scintillation interference

4 Mitigation techniques

Critical government and industry sectors are growing increasingly dependent on GNSS for positioning, navigation and timing. At the same time, the availability of low-cost GNSS jamming devices are presenting a serious threat. For this reason some countermeasures have been implemented in order to mitigate interference effects at the receiver. Some techniques are presented in this chapter.
Inertial aiding and vector tracking improves the receiver robustness by lowering the minimum required C/N_o[1] level for receiver acquisition and tracking. Instead, in the spatial and time-frequency filtering approaches, the incident interfering signals are suppressed before entering the receiver.
Each of these techniques is beneficial and effective by itself. However, they can also be integrated and applied together for greater robustness and protection against interference.

4.1 Inertial aiding

Inertial navigation is a self-contained navigation technique in which measurements provided by accelerometers and gyroscopes are used to help the position and orientation tracking of an object. By processing this kind of data it is possible to alleviate the fact that the GNSS communication can be unstable due to interference.

[1] C/N_o is the ratio between a carrier power and the power of noise per unit bandwidth and is a parameter used to provide and indication of the quality of a received signal.
$C/N_o[dB - Hz] = SNR + 10\log_{10}(bw)$ where bw is the front-end bandwidth in Hz.

The advantage of this technique is that it is not influenced by jamming signals, as it does not use radio signals at all, but the accuracy greatly degrades if interference do not permit communication and become worst and worst over time. Anyway, it is a solution that can help with spot interference by keeping a continuity during navigation without the GNSS signal guidance.
This solution can be still a good improvement in those cases but only when the receiver is inside a system containing already all needed sensors. Otherwise, even if those type of sensors are quite cheap nowadays the overall cost of hardware and software implementation is often not advantageous especially if associated with the limited benefits. Finally, not in all environments this types of sensors can be used limiting one more time the effectiveness of this technique.

4.2 Spatial filtering

Spatial filtering uses antenna arrays to point the receiver antenna beam towards the GNSS signal and away from interference.
Antenna array processing techniques were first used in the field of radar signal processing. The idea is to linearly combine the received signals at the sensors in a weighted fashion to steer the array response in the direction of target signals while spatially filtering out the interference sources.
Many techniques can be used to locate the interference and to redirect antennas in order to improve signal. The simplest way, even if not applicable in all situations, is to have an history map of interference to use as guidance. An example of this is shown in figure 4.1 that report GPS accuracy information by aircraft. It's not necessary a map of where GPS signals are deliberately jammed but looking at where red areas are it's quite obvious that is not due to natural interference. Since this data are based on civilian aircraft data war-zones like Ukraine are white because are no-go areas.
The problem with this method is that more than one antenna need to be used and this is not always possible due to dimension of the device, environment and cost but it can give a substantial improvement in terms of signal quality and position accuracy.

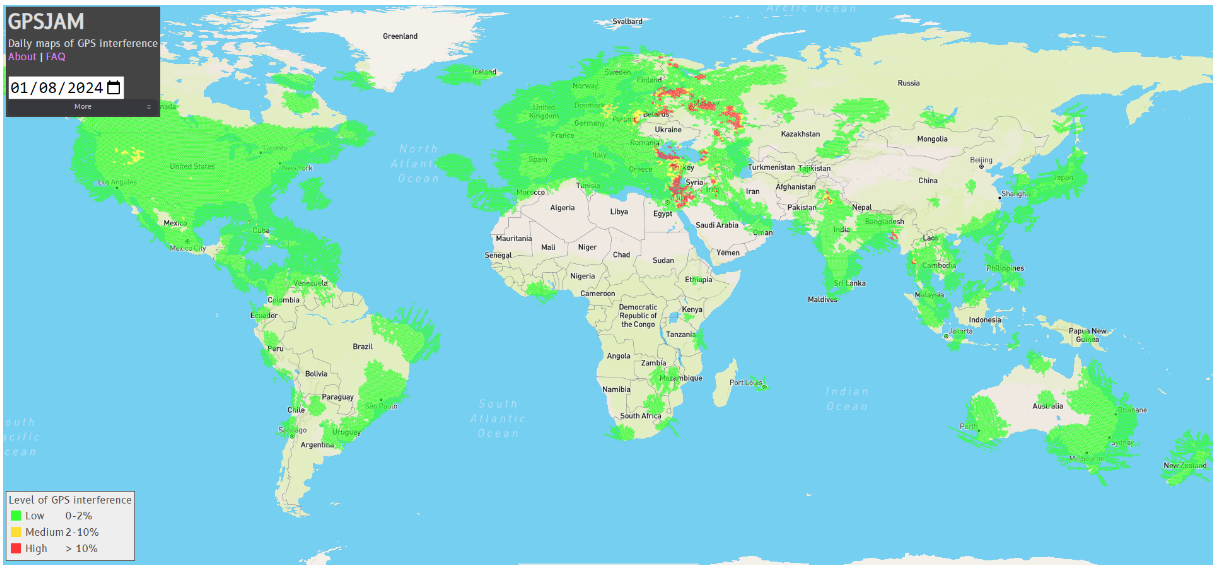


Figure 4.1: GPS interference map

4.3 Time-frequency filtering

This technique consists in the use of adaptive filters that remove the time-frequency varying interference content from the received signal. They differ by the domain in which they operate and in the way they try to separate useful signals from interfering signals. As shown in figure 4.2 this filtering take place after the front-end and before acquisition in order to filter out interference that would affect acquisition and tracking. There are different methods used to filter out interference components from the input signal. This methods mainly differ for the domain in which the filter operate, the implementation of the filter and the blanking rule.

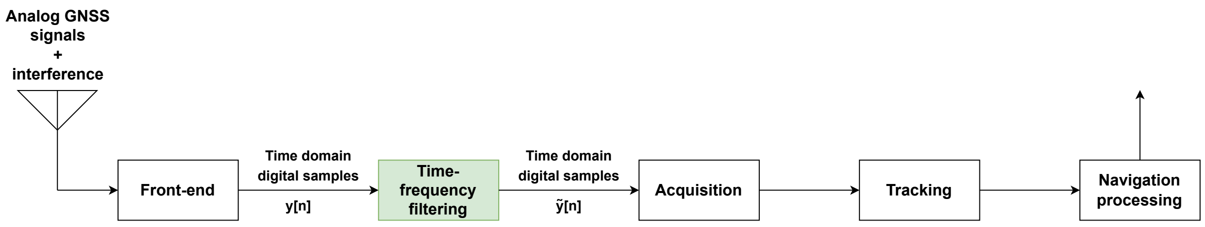

Analog GNSS signals + interference
Front-end
Time domain digital samples
y[n]
Time-frequency filtering
Time domain digital samples
ỹ[n]
Acquisition
Tracking
Navigation processing

Figure 4.2: Time-frequency filtering location

In the following figure an overview of some of the most common used techniques is shown. Some of them are then better described later on this chapter.

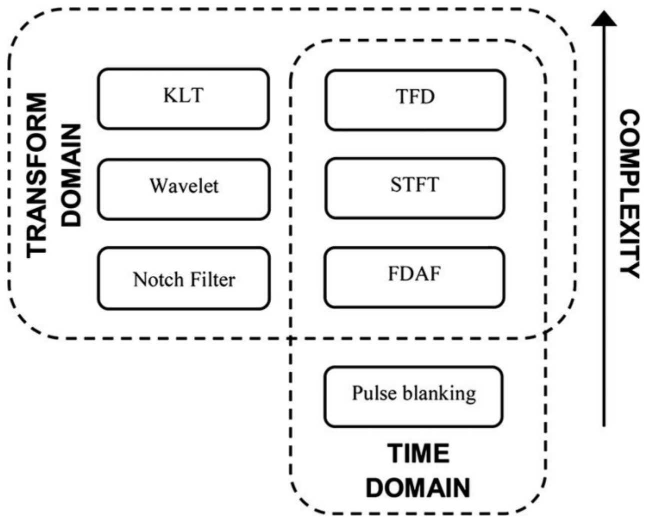


Figure 4.3: Time-frequency techniques overview

4.3.1 Pulse Blanker (PB)

This technique, in its simplest realization, just replaces any large values (in time domain) by zeros. It requires no multiplications and has very low computational complexity respect to other mitigation methods. However, a suitable threshold must be estimated by taking into account typical GNSS signals power. This can be challenging since the power variation of the input signal due to interference and the variable gain due to AGC must be considered. More than that this method has good time selectivity but no frequency one.
In the following figure an example of how the Pulse blanker works on signals is reported.

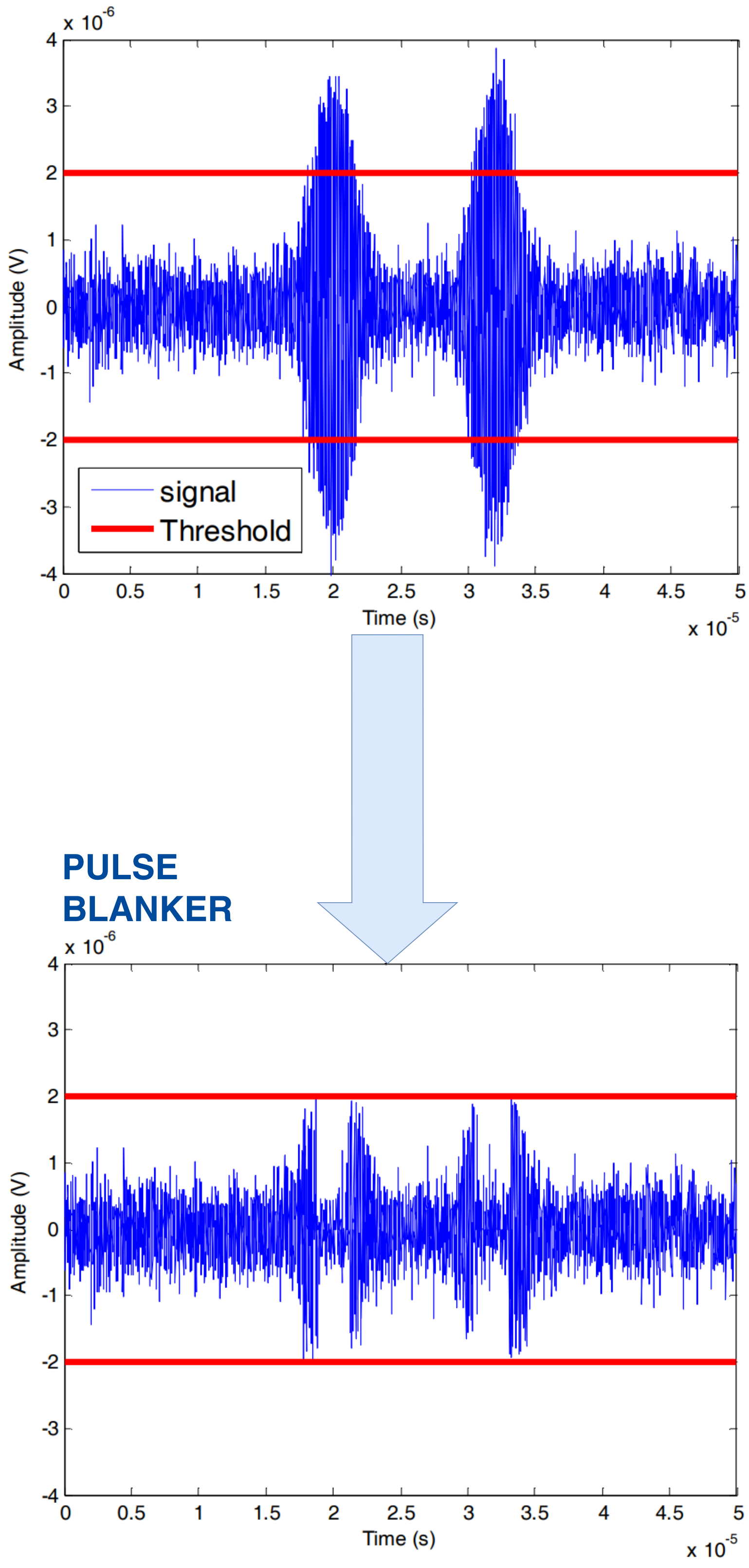


Figure 4.4: Pulse Blanker functioning

4.3.2 Notch Filter (NF)

If the interference signal has a single spectral peak (i.e. a tone or CW signal) then an NF can be used to remove it. NF is usually implemented with an IIR filter such that high interference suppression can be achieved while removing as little as possible from the rest of the unaffected spectrum. By using multiple cascaded notch filters, multi-tone signals can also be removed.
An effective mitigation method against chirp like signals is adaptive notch filtering (ANF). The notch frequency is adapted, such that it can follow the frequency modulation of the interference.
Some limitations with ANF include the adaption time, frequency discontinuities when the interference "jumps" to new frequencies, switching of the filter on or off and the ineffectiveness against other types of interference.

4.3.3 Frequency Domain Adaptive Filtering (FDAF)

The FDAF method transforms the signal to the frequency domain via a discrete Fourier transform (DFT) (often an FFT for processing efficiency). Then, it removes some of the spectral components (higher than a suited threshold) before transforming the signal back to the time domain with an inverse discrete Fourier transform (IDFT) (also with an inverse fast Fourier transform). This method processes a block of data at a time and the size is indicated as N_{FT} since it would be also the DFT size. One downside of FDAF is the limited control of side-lobes. Usually, windowing before the FFT suppresses the side-lobes, but it causes unwanted temporal deformations after reconstruction. A method around this limitation is to use a channelized architecture, such as a poly-phase filter-bank, however it significantly increases computational complexity through the added filter. A schematic view of the method is represented in figure 4.5.

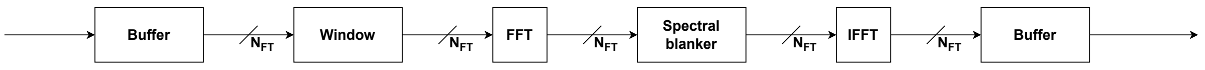


Figure 4.5: Block diagram for FDAF

A figure follow to show an example of how this technique act on the signal step by step in order to fully understand it.

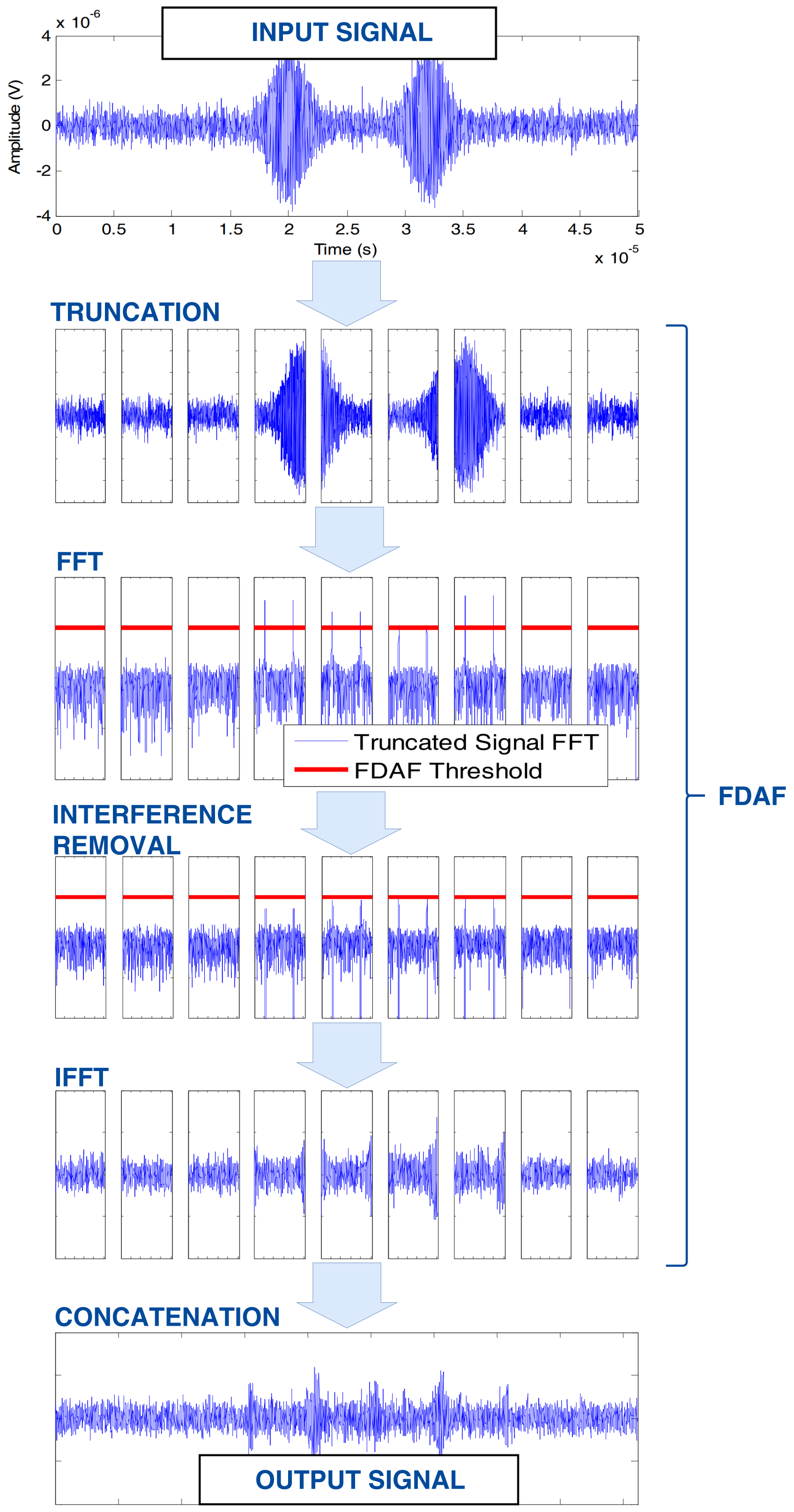


Figure 4.6: FDAF functioning

A performance improvement of FDAF can be obtained using two parallel FDAF processing blocks with an offset (delay). This method is the dual-FDAF. The usage of two processing blocks smooths out any discontinuities, suppresses ringing effects, and allows the efficient use of windowing methods. Furthermore, as the process overlap in time, this method is also more effective against shorter pulses signals than the standard FDAF. The drawback is that it requires approximately twice the processing and memory resources and care need to be taken on the overall signal processing. Indeed, if AGC is used, exist the possibility that (due to the different delay of the branches) when the signal is summed and recovered the two branches can have different amplification. This variable amplitude problem, as seen also for the Pulse Blanker involves also the threshold choice.
A schematic view of the method is represented in figure 4.7

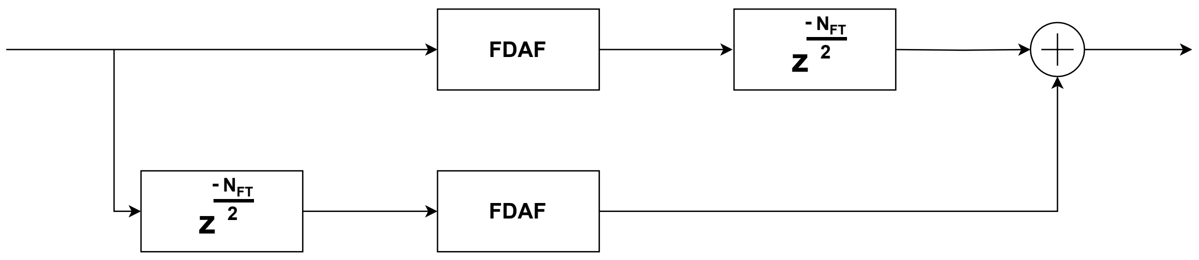


Figure 4.7: Block diagram for dual-FDAF

4.3.4 Advanced mitigation methods

Many other mitigation methods based on time-frequency filtering exists. This include filter banks, Fourier methods, wavelet transforms, the Karhunen-Loève transform (KLT), High-Rate DFT-Based Data Manipulator (HDDM) and interference modeling methods. Although many methods have excellent results, they are often limited by the adaptability to hardware implementations due to non-linear operations, recursive processes, excessive memory and buffering requirements, high processing word-lengths including floating point operations, high computational complexity, or they are only applicable to certain interference types.
Complicated architectures which consists of several simultaneous mitigation algorithms, which work in parallel and only selects the best output also exist. However, these present even more challenges for hardware adaptions.

4.4 Vector tracking

The conventional method of GNSS receiver baseband signal processing is classical scalar tracking where signal parameters of in-view satellites are tracked separately. In other words, there is an individual tracking loop for each satellite, and they operate independently. Due to its simplicity and its effectiveness under benign operative conditions, classical scalar tracking is still widely implemented. However, its performance is limited under signal-deteriorated environments.

Vector tracking achieves enhanced tracking robustness under degraded conditions such as weak signals and interference by acknowledging and utilizing the fact that signal channels are coupled through the shared receiver states of position, velocity and time. This is achieved through feedback of the receiver states from the navigation module to the signal tracking channels as a priory information and constraint. In this manner, channels are jointly tracked through the receiver states. This approach improves tracking and mitigate interference improving quite a lot the GNSS receiver but increase drastically the computation volume.

5 Mitigation choice

As shown in section 4 there are a lot of techniques that are available in order to increase performance and reliability of GNSS receivers under harsh conditions such as interference.
In terms of implementation, time-frequency filtering is often chosen since it is not too complex and it can perform quite well with limited resources. We need also to remember that in many circumstances there is no possibility of having more than one antenna or other sensors like accelerometers and gyroscopes that are needed for some of the techniques previously mentioned.
Moreover, time-frequency techniques gives us multiple approaches to elaborate signals and mitigate interference. We can choose to stay in time domain or transform signals in frequency domain before blanking for example. How to choose and discern the appropriate technique to implement is mainly given by the aim that we have. For example, since each technique has a target interference on which it works best, depending on the environment of the receiver we can predict what can gives us the best result.
In this particular case, instead, since this work has been done during an apprenticeship period in Qascom S.R.L. the choice of what to develop is in part due to internal company reasoning. To be specific Qascom had already developed a GNSS receiver using Analog Devices ADRV9361-Z7035 as a Software-Defined-Radio (SDR) with a dual-FDAF ip implemented. So, a good mitigation technique was already in place but, since for the moment a constant gain is used instead of an AGC a Pulse Blanker can be added in order to both help to avoid saturation and work together with the dual-FDAF to reach better interference mitigation. In the next chapter more details on the Pulse Blanker itself and how it has been integrated with the already implemented core will be given.

6 Implementation

This chapter collects all considerations about the implementation of the Pulse Blanker core. Hardware environment, concepts and actual design choices are discussed here.

6.1 Hardware environment

As explained in chapter 5 the aim of this core is to mitigate interference in GNSS context. Specifically, it is thought to be added to the Software-Defined-Radio (SDR) developed by Qascom S.R.L. For this reason and to better understand some of the implementation choices an overview of the hardware environment is reported in this section.

6.1.1 Software-Defined-Radios

SDRs represent a revolutionary approach to wireless communication by putting beside traditional hardware-based components a flexible and programmable software. These radios enable the dynamic and real-time reconfiguration of their operating parameters, such as frequency, modulation, and signal processing algorithms, through software control. The primary advantage of SDRs lies in their adaptability, allowing for seamless updates and modifications without requiring changes to the underlying hardware. This flexibility is especially valuable in rapidly evolving communication standards and protocols, as SDRs can easily switch between different modes and frequencies, making them versatile across various applications. Additionally, SDRs contribute to cost-effectiveness by reducing the need for specialized hardware for each communication standard, promoting interoperability, and facilitating innovation in wireless technologies. Furthermore, their ability to accommodate new features through software updates ensures that SDRs remain relevant and efficient in an ever-changing technological landscape.

6.1.2 ADRV9361-Z7035

The specific product used is Analog Devices Inc. ADRV9361-Z7035, a System-On-Module (SOM) that combines the Analog Devices AD9361 integrated RF Agile Transceiver with the Xilinx Z7035 Zynq-7000 All Programmable SoC. This board offers wideband 2x2 receiver and transmit paths in the 70 MHz to 6 GHz range. It combines the RF signal path with the high-speed programmable logic and forms the RF-to-baseband signal processing core of a wireless communication system, allowing the designer to focus on application specific differentiating features.
The board is shown in figure 6.1. Carrier cards are also available.

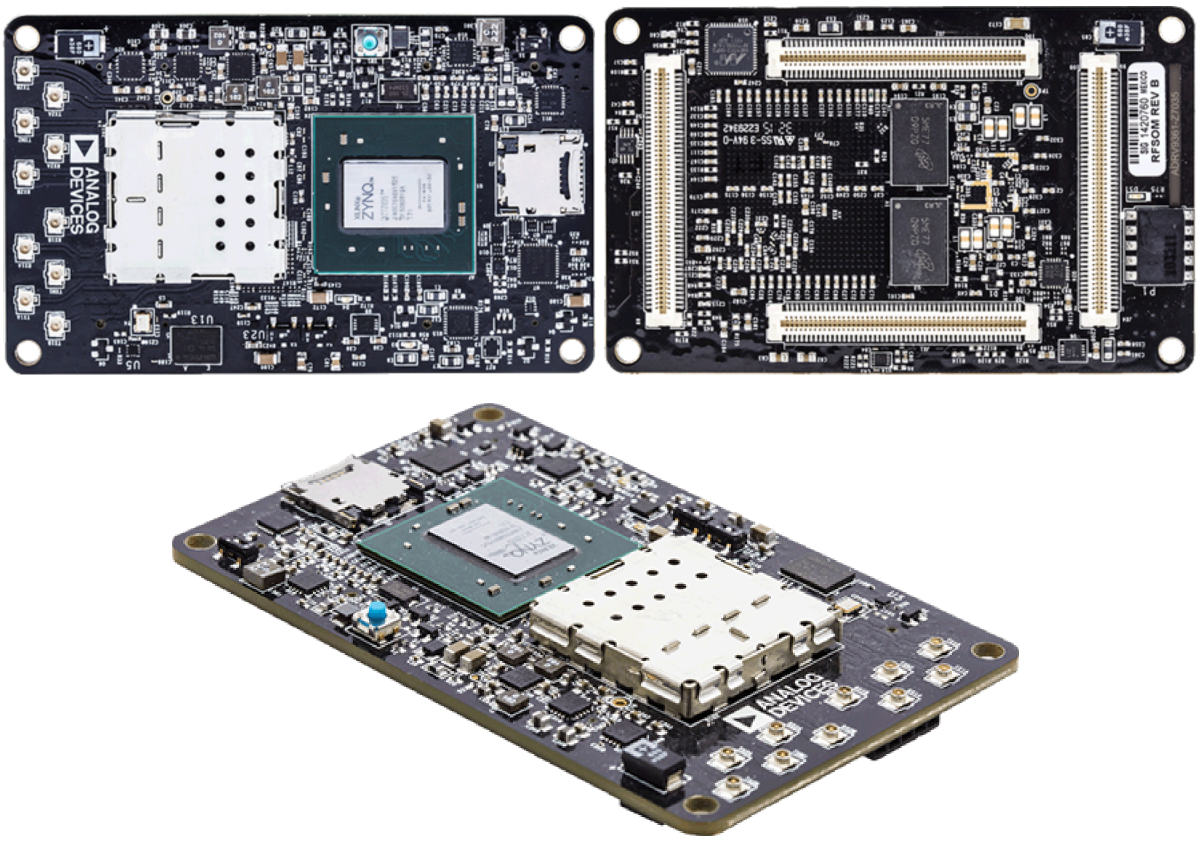


Figure 6.1: ADRV9361-Z7035

Here a description of the main features of the board.

Xilinx Zynq-7000

This board includes a Xilinx Zynq XC7Z035-L2 FBG676I AP SoC. Tight integration between the ARM-based processing system and the on-chip programmable logic creates unlimited possibilities for designers to add virtually any peripheral or create custom accelerators that extend system performance and suite unique application requirements.
This is a -2 speed grade and low power (-L) binned device. All SOM memory and digital interfaces are connected to Zynq through the Processing System (PS) or Programmable-Logic (PL). The Analog Devices AD9361 connects through Zynq PL.
Here a figure with a quick overview of the Zynq-7000 All Programmable SoC.

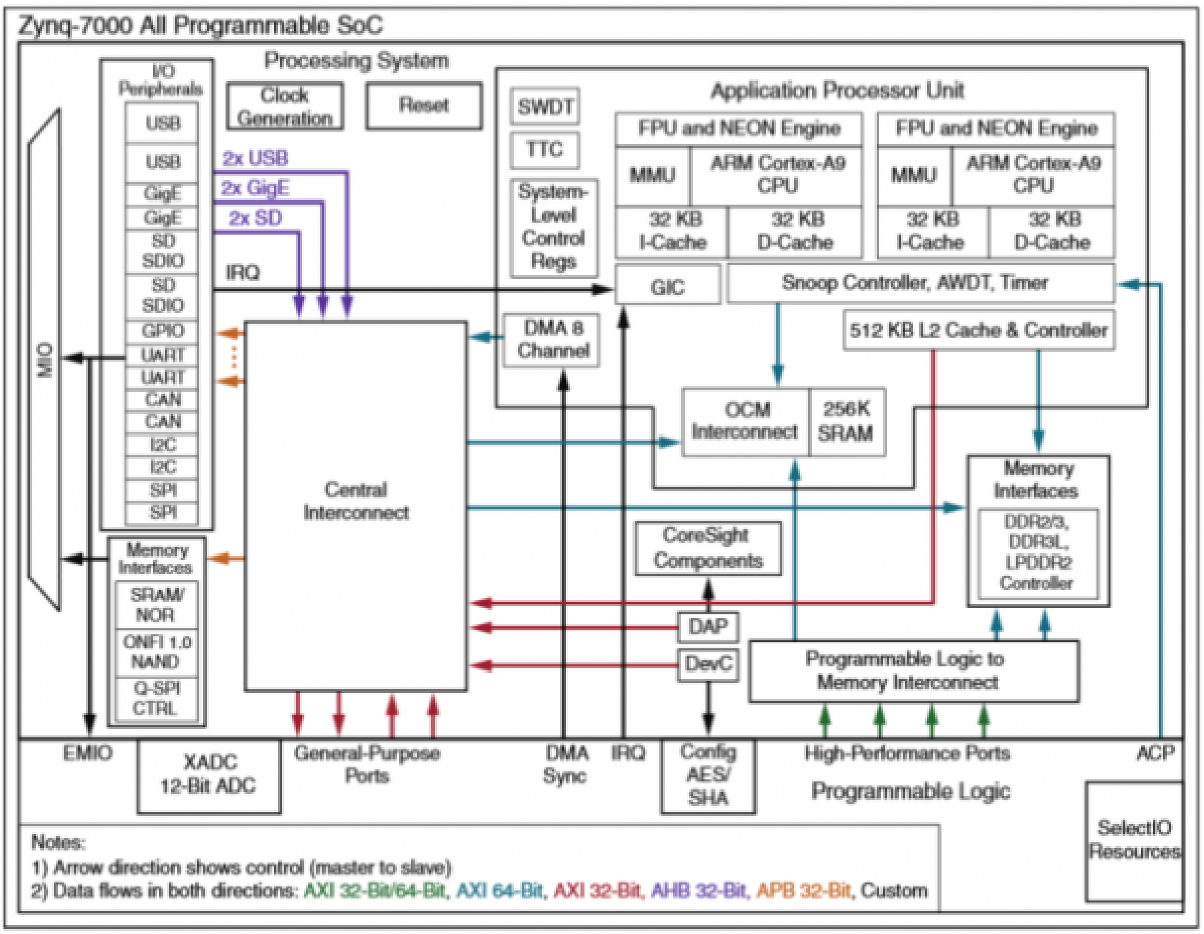


Figure 6.2: Zynq-7000 overview

AD9361 RF Agile Transceiver

An Analog Devices AD9361 RF Agile Transceiver is also included in the board. This is a high performance, highly integrated RF Agile Transceiver. Its programmability and wideband capability make it ideal for a broad range of transceiver applications. The device combines an RF front-end with a flexible mixed-signal baseband section and integrated frequency synthesizers. The AD9361 operates in the 70 MHz to 6 GHz range. Channel bandwidths from less than 200 kHz to 56 MHz are supported.
A scheme of the device is here showed.

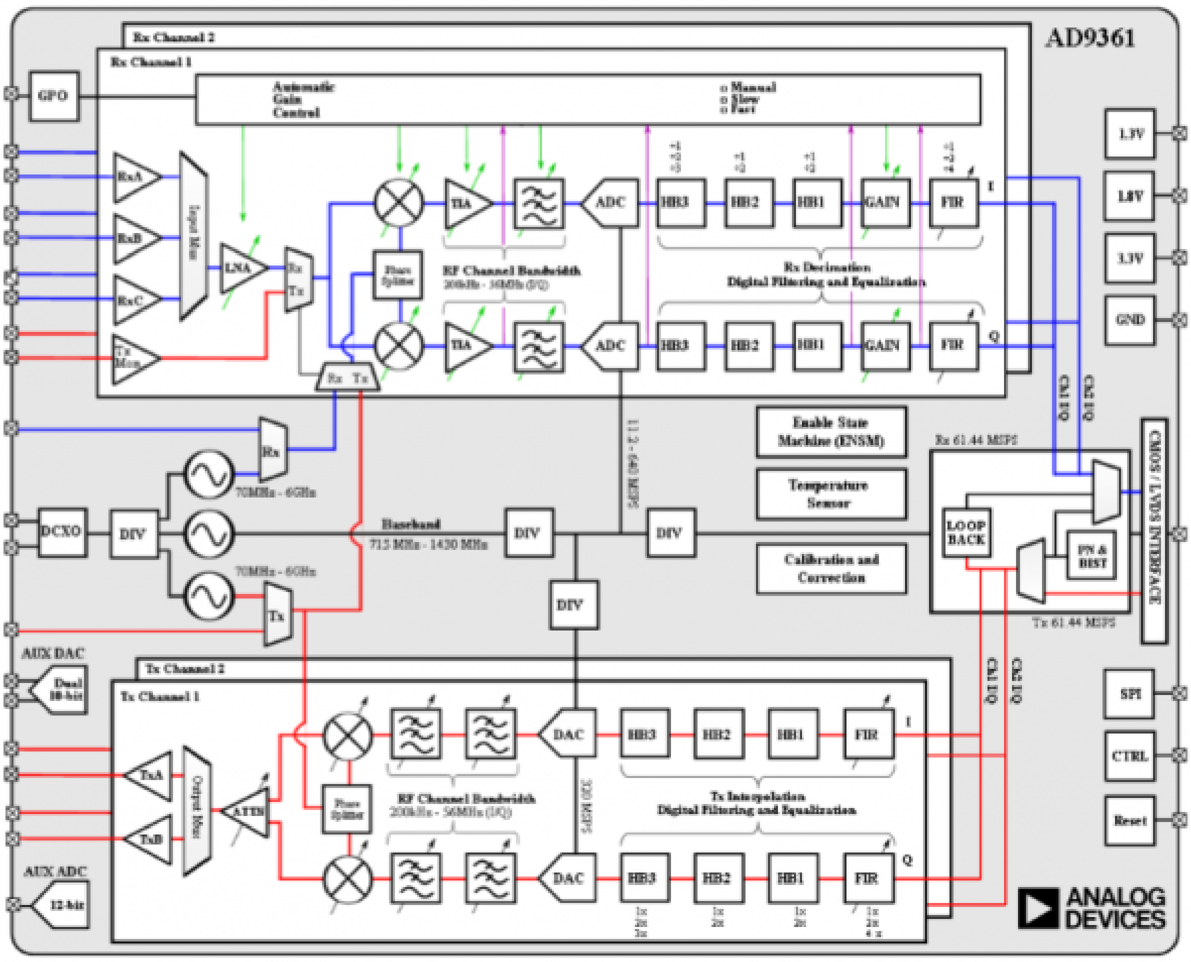


Figure 6.3: AD9361 scheme

AD9361/Zynq SoC connection

AD9361-Z7035 connects the Xilinx Zynq Z-7035 SoC directly to the AD9361 RF Agile Transceiver with dedicated high bandwidth data ports and clocks, an SPI control interface, and other control and framing signals.
The AD9361 digital interface is comprised of two parallel data ports (P0 and P1) and several clock, synchronization, and control signals to transfer samples between the AD9361 and the Zynq SoC. These signals can be configured as single-ended CMOS signals or as LVDS signals for systems that require high speed, low noise data transfer. The system level reference designs that exist for the AD9361-Z7035 all use LVDS, to achieve the maximum data throughput, but can be configured in CMOS mode to better prototype a different hardware subsystem.
In LVDS mode, the interface is operated in double-data rate (DDR) mode. Therefore, 12-bit samples to/from the AD9361 are sent across two 6-bit lanes on differential pairs. A figure showing an overall view of the board follows.

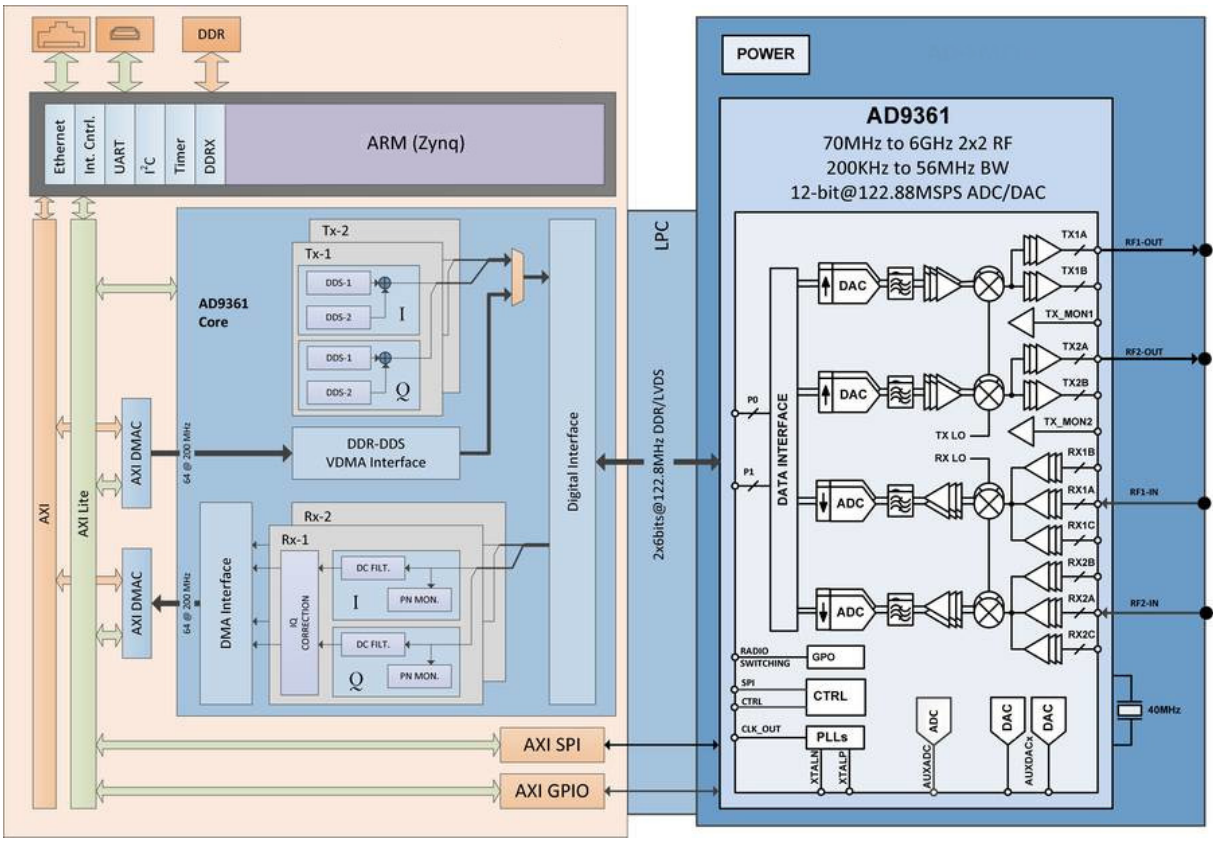


Figure 6.4: Board connections overview

HDL core integration

As outlined by the HDL reference design integration for the ADRV9361-Z7035 a specific location of cores that work along the datapath is considered ideal. Here follows a simplified block diagram that, together with figure 6.4, help understanding where to locate HDL custom logic.

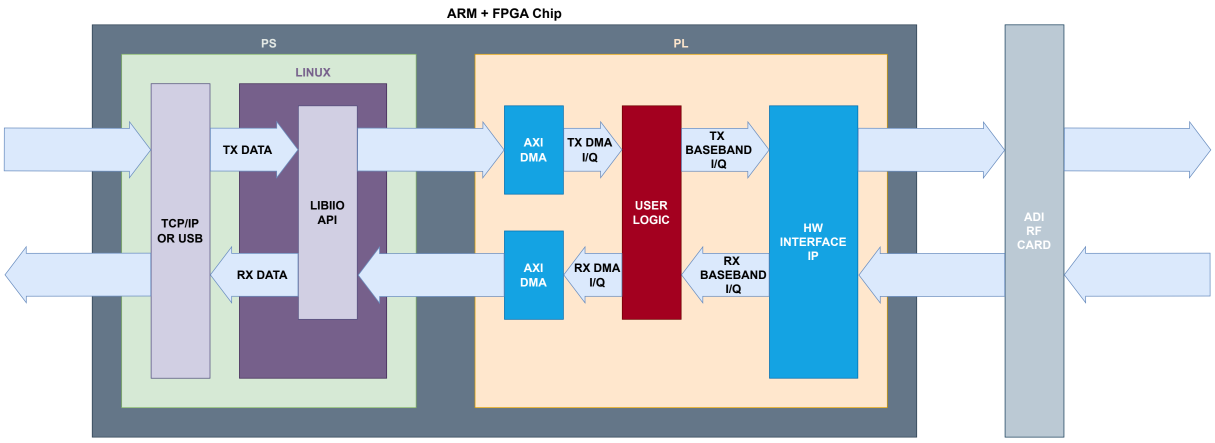


Figure 6.5: Simplified block diagram for HDL custom logic integration

Now, if we look at the reference design for the RF carrier, given by Analog Devices Inc. we can easily understand that in order to be compliant with figure 6.5 we need to locate our custom HDL ip as shown in the following figure.

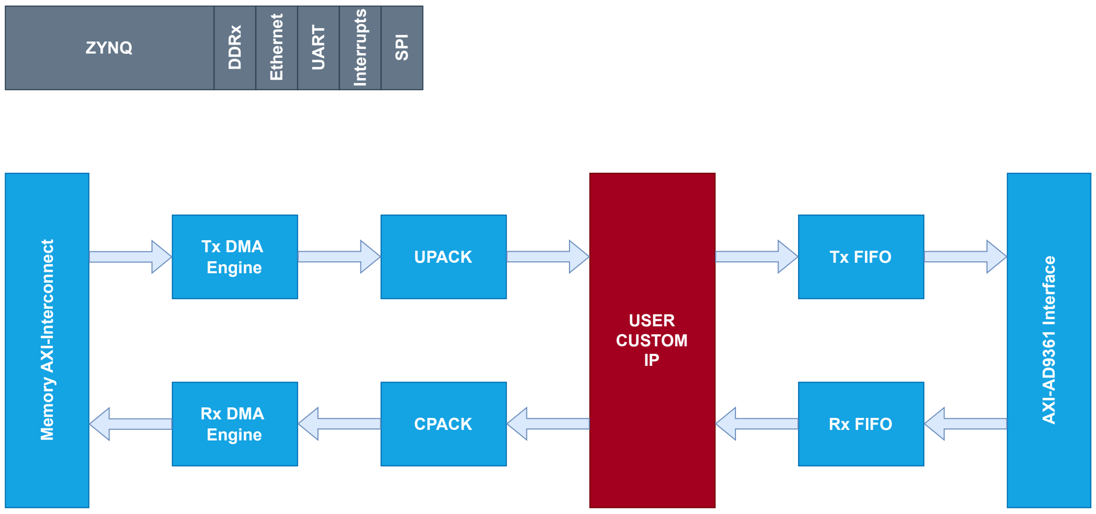


Figure 6.6: HDL reference design with custom ip for the FPGA carrier

As shown, data are intercepted between FIFO and PACK cores.
For what concern received data Rx FIFO core has the role of downscaling the clock rate. There are scenarios when the device clock is too high making a challenge to integrate any processing cores because of the small timing margins. By reducing the clock rate of the data path, the user can easily integrate any custom processing core into the design. Furthermore, CPACK core collects samples from all channels, rearranged them and present them to the DMA. A figure follow to better understand what this core does.

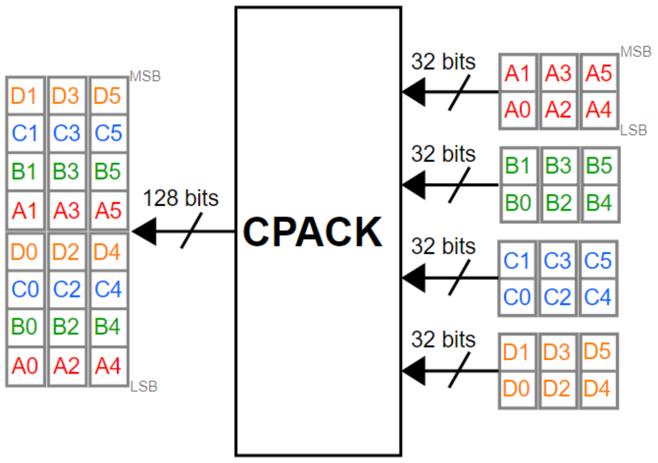


Figure 6.7: CPACK functional description

Of course, for the opposite datapath UPACK and Tx FIFO have the same but complementary roles.
In this case, we need to mitigate interference so, it is quite trivial to say that we are interested about the Rx datapath and here, following Analog Devices Inc. reference design the custom core will be placed.

6.2 Pulse blanker

As seen in chapter 4 there are many ways of realizing a pulse blanker. In this section a description of this particular implementation will be given.

6.2.1 Concept

As discuss the simplest pulse blanker checks samples and blank (set to zero) those that are above a certain threshold. A slightly more advanced version of this idea has been implemented here. Specifically, two modalities can be used in order to perform better depending on the interference. Here follows an explanation of those operative modes.

Sample mode

Sample mode is the simplest way in which the core can operate. Input samples that have the absolute value of the real or the imaginary part bigger than a given threshold are blanked, instead, the other samples remains the same.
This operative mode has the advantage that small amount of data is blanked (if threshold is set properly) and few resources are needed.
On the other hand, if interference such as burst jamming signal is present even if threshold is set properly some peaks will not be blanked and the spectrum will remain altered. For this reason burst mode has been introduced.

Burst mode

This mode introduce a slightly more complex approach at blanking. In particular, if we think at bursts given from a radar, that in our case represents a jamming signal, we can see that even if in the majority of signal time interference is irrelevant, some burst with higher power than the GNSS signal will be present. During that interval, if we introduce a simple sample blanking with threshold, there would be an improvement but still, some peaks of interference will remain, making it harder acquisition and tracking operations. So, we can improve the blanking method by introducing the possibility of blanking also some samples before and after an over-threshold event. In this way, during the burst, with a proper choice of how many samples to blank, all the signal will be set to zero. Therefore, if those bursts are not to frequent in time, a little more signal is blanked but at the same time less distortion will be present in later operations.

To better understand how this two modes works a simple example is reported in the following figure where BT represent the threshold and SB and EB represent how many samples are blanked before and after an over-threshold as described in the next paragraph.

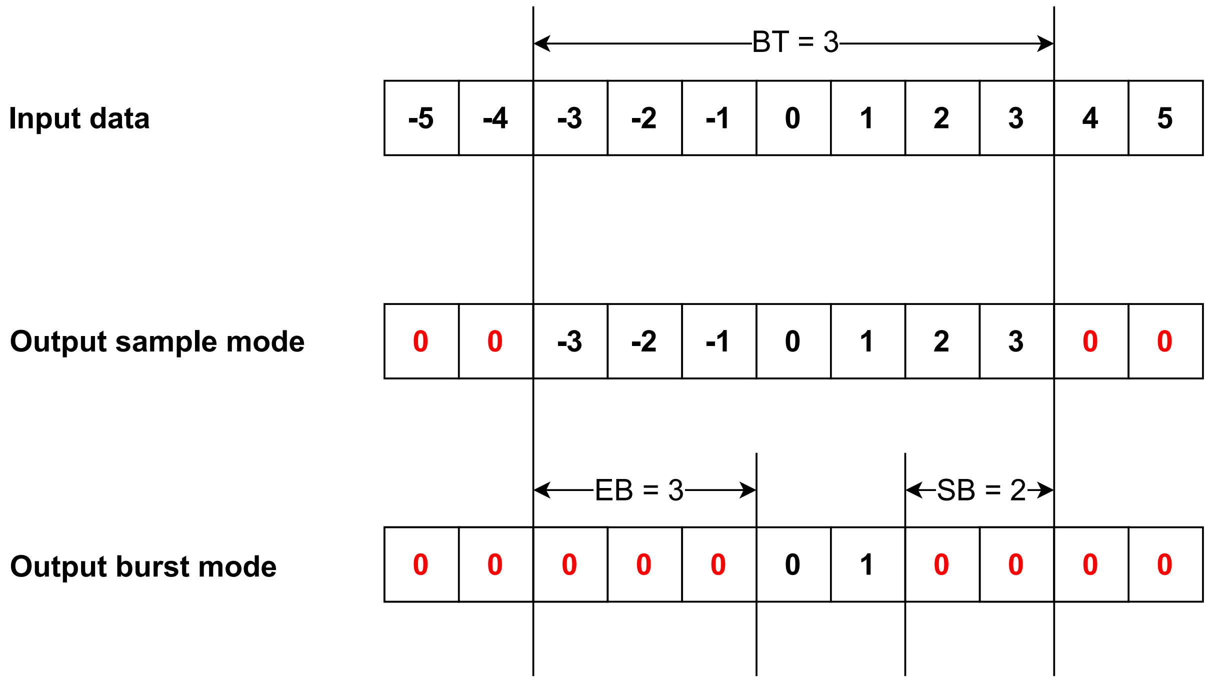


Figure 6.8: Sample/burst mode example

Parameters

In order to work properly some parameters have to be provided to the core.
First of all a Blanking-Threshold (BT) has to be provided in order to set at what level we want to cut the signal. Then, to implement also burst mode, Start Blanking (SB) in order to specify how many samples to blank before an over-threshold and End Blanking (EB) in order to specify how many samples to blank after an over-threshold need to be foreseen.
Furthermore, since we want to choose what the core do, ENable parameter (EN) in order to turn on or off blanking and MOde parameter (MO) in order to select what modality the core have to use are contemplate.
Finally, since it is useful to monitor what the core does in order to set input settings as more effective as possible, some other parameters and monitoring registers have been considered. Time interval (CNT) is an input parameter that set an interval (given as

number of samples per interval). This allow to use register Blanking Rate (BR) to count how many blanking events there has been in the last interval and represent the rate of blanking. This, combined with Blanking Events (BE), that report if from the last time it has been read a blank has occur, gives some information that allow to choose properly what modality is more appropriate and what values are more suitable to obtain the best mitigation possible.

6.2.2 Design

Starting from the concept described in subsection 6.2.1 here we explore and extensively report on how the core has been implemented. The idea is to implement it inside the PL part of the Zynq processor in order to have high efficiency and performances.

Integration with existing core

As discussed in chapter 5 this pulse blanker core has not been develop from scratch but it is born with the idea to be integrated with the existing dual-Frequency Domain Adaptive Filtering core developed by Qascom. In order to do that we need to understand some concepts on how FDAF has been implemented.
This core have to be integrated inside the carrier of this complex design explained in section 6.1. Furthermore, we have a precise location where to intercept input data as depicted in figure 6.6.
However, those signals have a peculiar format given by Analog Devices Inc. (ADI) that is not the best for interoperability and for signal processing. For this reason inside the core has been included an entity that provide a First In First Out (FIFO) structure that allow to pass from ADI signals to Advance eXtensible Interface (AXI) ones, in particular AXI4-Stream protocol is used [Appendix A].
Those data are organized and sent in the AXI4-Stream data bus as in the following figure:

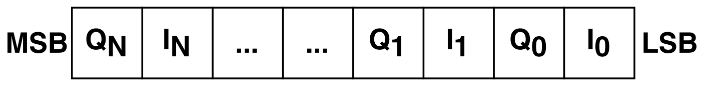


Figure 6.9: Data organization in data bus

Here we can see that data are divided in channels (subscript in the image). Channels are located in increasing channel index order from LSB to MSB of the bus. Each channel has in-phase (I) and quadrature (Q) component slot of bits. In each slot bits are oriented

from LSB to MSB according to the bus orientation.
So, this is the way and the protocol in which data are presented to the dual-FDAF core. Of course, once that data have been elaborated, they are translated back to ADI signals to be forwarded.
Therefore, also parameters and monitoring registers need to be accessed from outside the ip. For this reason an entity (AXI2LocalBus) has been implemented to give the core an interface between the AXI-Lite [Appendix B] protocol and a local bus that is simpler and easier to work with.
A visual summary of this implementation can be found in the following figure.

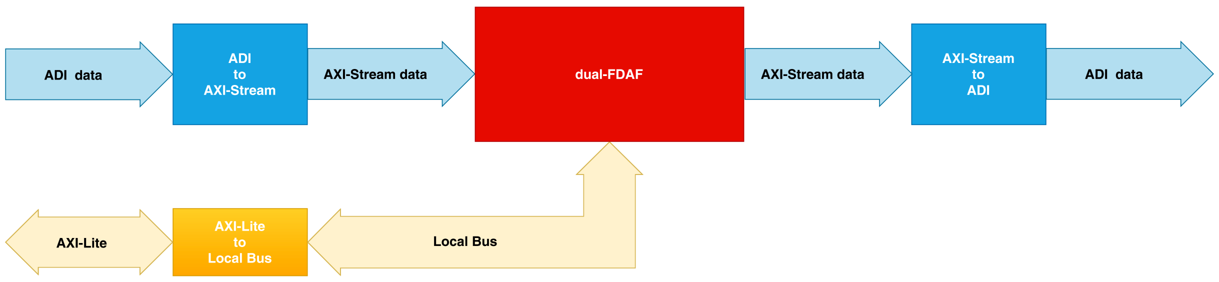


Figure 6.10: dual-FDAF core scheme

Then, in order to add a pulse blanker function to this already developed ip we need to slightly modify this scheme.
For what concern data, we can insert the pulse blanker entity before the dual-FDAF entity, intercepting data from AXI-Stream bus and then forwarding the result of the blanking to it.
Instead, to provide access to the AXI-Lite communication channel to both cores it is convenient to introduce a new entity that works as a local bus switch. In this way, by setting the most significant bit of the address according to the core that we want to reach we can provide parameters and read registers from both entities but by using only one interface to/from the core.
So, by applying all this considerations the scheme will be modified as shown in the following figure:

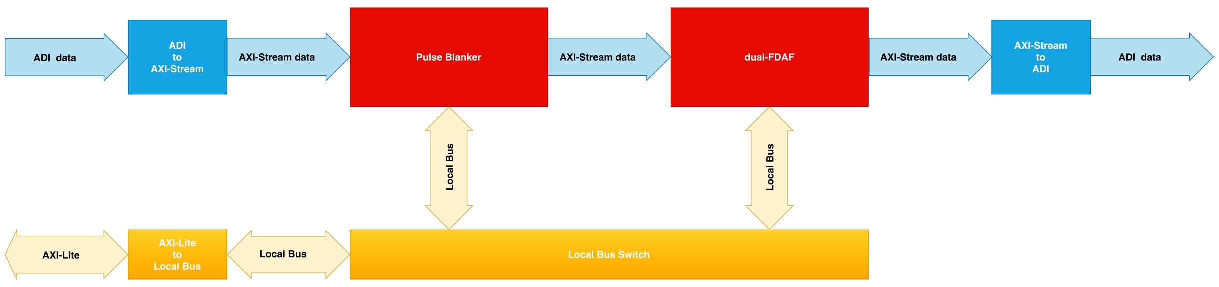
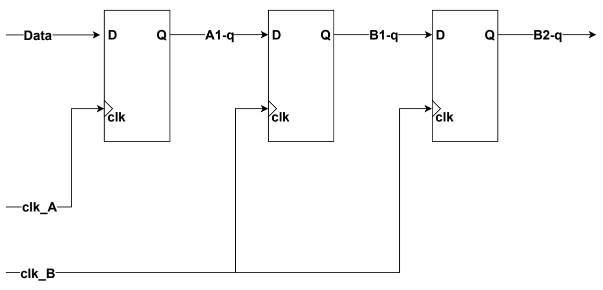


Figure 6.11: dual-FDAF and pulse blanker core scheme

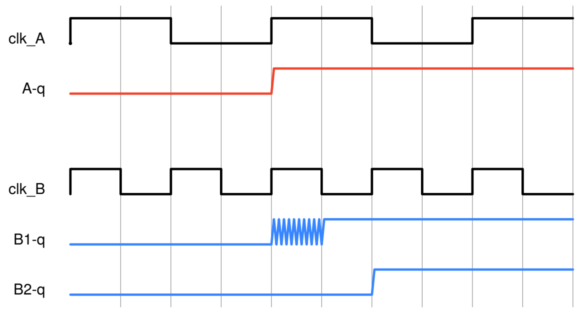
Register controller

In order to access to input and output parameters the core have to save those that are send by the AXI-Lite bus and store those that need to be read if requested. In order to do that an entity inside the Pulse Blanker ip named "Register controller" has been developed since it is not a trivial operation. The same has been previously done for the FDAF ip. The reason why this is not an easy task it that, for the current implementation AXI-Lite bus and Local bus have the same clock but, the core may operate in a different clock domain. This means that care need to be taken on how to manipulate signals in order to avoid meta-stability problems and proper Clock Domain Crossing (CDC) need to be implemented to remove the chance that signals could not be settled while read. To choose the proper way to implement it in each case, source and destination clock domain as well as what data represent need to be considered. Specifically, in our case we have that the local bus clock is slower than the core clock.
Here follows some considerations on how this aspect has been faced while developing the Pulse Blanker ip.

Input parameters: single or multi-bit they contains setup information and goes from the slower clock domain (local bus) to the faster clock domain (core). So, even if multi-bit parameters have correlated bits we have that with the introduction of two cascaded Flip Flop (FF) in the destination clock domain the signal have enough time for the meta-stability to settle down.
A visual representation of this CDC method is depicted in the following figures.



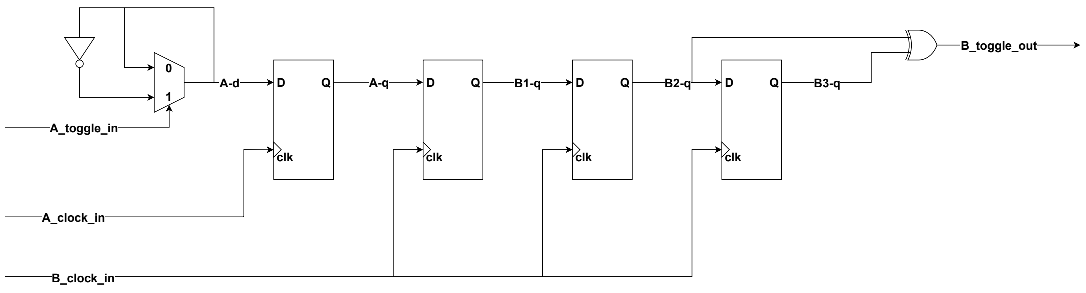
(a) 2-FF CDC scheme



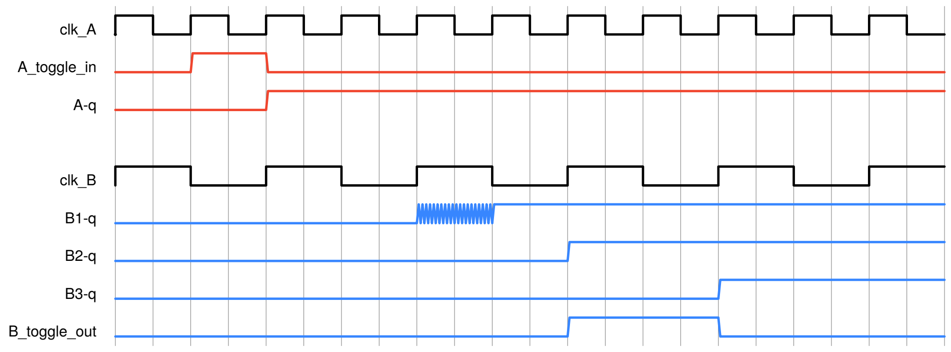
(b) 2-FF CDC timing

Figure 6.12: 2-FF CDC

Monitoring parameters: in this case things are slightly more complicated. Indeed, for these parameters we have the core clock as the source domain (faster) and the local bus clock domain as destination (slower). More than that both BE and BR are multi-bit data. However, we have that BE have non-correlated bits, since each bit refer to an event that occur in the correspondent channel. This is enough to allow us to use the same technique explained in the previous paragraph (input parameters) and represented in figure 6.12 to solve meta-stability issues.
This is not the case for BR. In this case data represent a counter so if data are read at the wrong time unstable data or a number with high difference from the real value can be read. To solve this nontrivial problem some logic must be implemented. A deeper evaluation of the parameter behavior needs to be done.
BR parameter in core clock domain is a register that change value each time CNT samples have been elaborated. This means that typically BR change value with a frequency that is slower than the core clock. This gives us the opportunity to generate a toggle signal for one core clock cycle each time BR change value. However, since the pulse is generated in the faster clock domain a 2-FF synchronizer fails as CDC method as the destination FF may skip the incoming pulse. To solve this, a FF is used to create a continuous high/low signal starting from the pulse in the core clock domain. In this way a following 2-FF CDC configuration allow to take that signal, and once delayed and passed through a rising/falling edge detector in the destination clock domain the pulse can be recreated. The diagram below shows the implementation.



(a) Toggle synchronizer scheme



(b) Toggle synchronizer timing

Figure 6.13: Toggle synchronizer

Once done that, we got a toggle in the destination clock that gives us the information on when the parameter BR is stable.
At this point we can use this signal as a read enable signal. Indeed, by using a recirculation MUX synchronizer (represented in figure 6.14) we can present to the local bus a register that maintain the same value and change to the new BR given in the core clock domain only when permitted by the toggle. In this way we change the value of the register that is presented to the local bus when a read operation is required only when we are sure that BR data is stable.

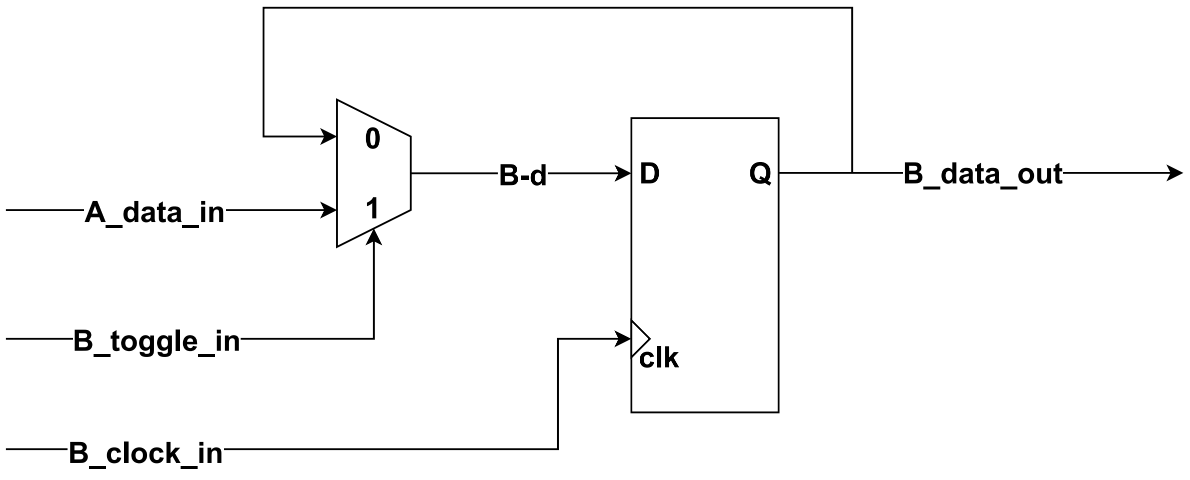


Figure 6.14: Recirculation MUX synchronizer

Practically speaking, this can be implemented with a simple register with enable. This name and representation is instead used in order to emphasize the fact that only when the toggle arrive BR value presented to the local bus is changed according to the its clock rising edge. This ensure that BR is read only when is stable.

Parameters constraints: some considerations need to be discuss about parameters values and width.
Starting from BR we need to consider that, since we need to sync between two clock domain, depending on the two clock frequencies, some time is needed to store this parameter. This means that with a certain data input frequency CNT can be so small that parameter storing will be done on the next change of BR in the core clock domain causing meta-stability risks again. This is highlighted in the following figure.

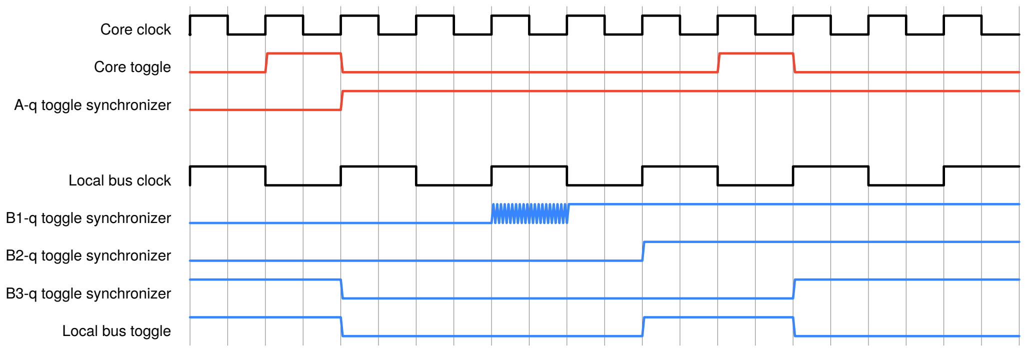


Figure 6.15: CNT constraint

As we can notice we need to ensure that three local bus clock periods fit between each BR toggle in core clock domain.
This may seem like to be a problem but, if we consider that usually data rate and local bus rate are way slower than core clock this problem reduce drastically. If for example we have a data rate of 24 MHz (F_{data}) and a local bus frequency of 50 MHz (F_{lb})we have that:

$$CNT_{min} = \left\lceil \frac{3F_{data}}{F_{lb}} \right\rceil = 2 \tag{6.1}$$

So, even if this remain a constraint if we consider that this parameter is used to set the interval to use for the calculation of Blanking Rate (BR) it will be pretty useless to set CNT this low. Indeed, this rate is useful to understand in a long enough period how many blanking operations have been made to set properly input parameters.
Moreover, a simpler but due consideration would be about CNT and BR width. BR width must be at least as big as CNT width to ensure proper results.
Then, considering that some memory is needed to implement burst mode (as later explained in 6.2.2) we need to be careful while assigning the width of SB parameter. Indeed, if x is SB width, CH is the number of channels and D is the width of each I/Q data then we can derive that a memory of $2 \cdot D \cdot CH \cdot (2^x - 1)$ is needed to store data. It is easy to understand that if we increase x then the memory required will drastically increment. Then a careful evaluation of this parameter need to be carried out. Therefore, even if F_{data} is reasonably high we need to remember that burst mode is thought to mitigate burst interference. This means that the number of samples interested for each burst is still limited and so SB width can be set quite low.
In the end, EN, MO and BE width must be big enough to have a bit for each channel.

Blanking

If only sample mode need to be implemented no memory is needed for data since the check and the blanking is done on the current data. Instead, since we want to implement also bust mode we need to introduce a memory to store sequential data to blank before and after an over-threshold. This memory needs to have a depth that is as big as the maximum value of SB parameter in order to make the core work properly.
This can be realized as a shift register where for each channel we pick I and Q data at the depth index given by SB parameter. This gives us all data to compare over BT and consequentially start to blank. This simple realization is functional but, since with

the increase of the memory size an exponential number of FF will be used a different strategy with a triple port distributed RAM has been used. A FIFO like behavior with pointers has been implemented to allow input and output data and the second output port has been utilized to pick I and Q data at the correct depth given by SB index. This realization have been chosen because it allows to optimize resources and improve timing respect to the shift register implementation.
At this point a counter is utilized. It will restart each time an over-threshold will be found and it gives an high signal (blank) for SB + EB data samples.
Each time a data sample is given as output and the counter signal is high we know that a blank event occurs and so we can count them to report BR and BE.
Of course the need of this register introduce the necessity of a initialization phase (since we need to ensure that we have the maximum number of SB data in order to blank SB maximum data before). Therefore, also data can't exit if no data enter. Anyway, if we consider again the data rate used in 6.1 with a reasonable SB width the time needed for the initialization phase and the data stored are quite insignificant.
So, taking all this into consideration, an overview of the pulse blanker core can be seen in the following figure.

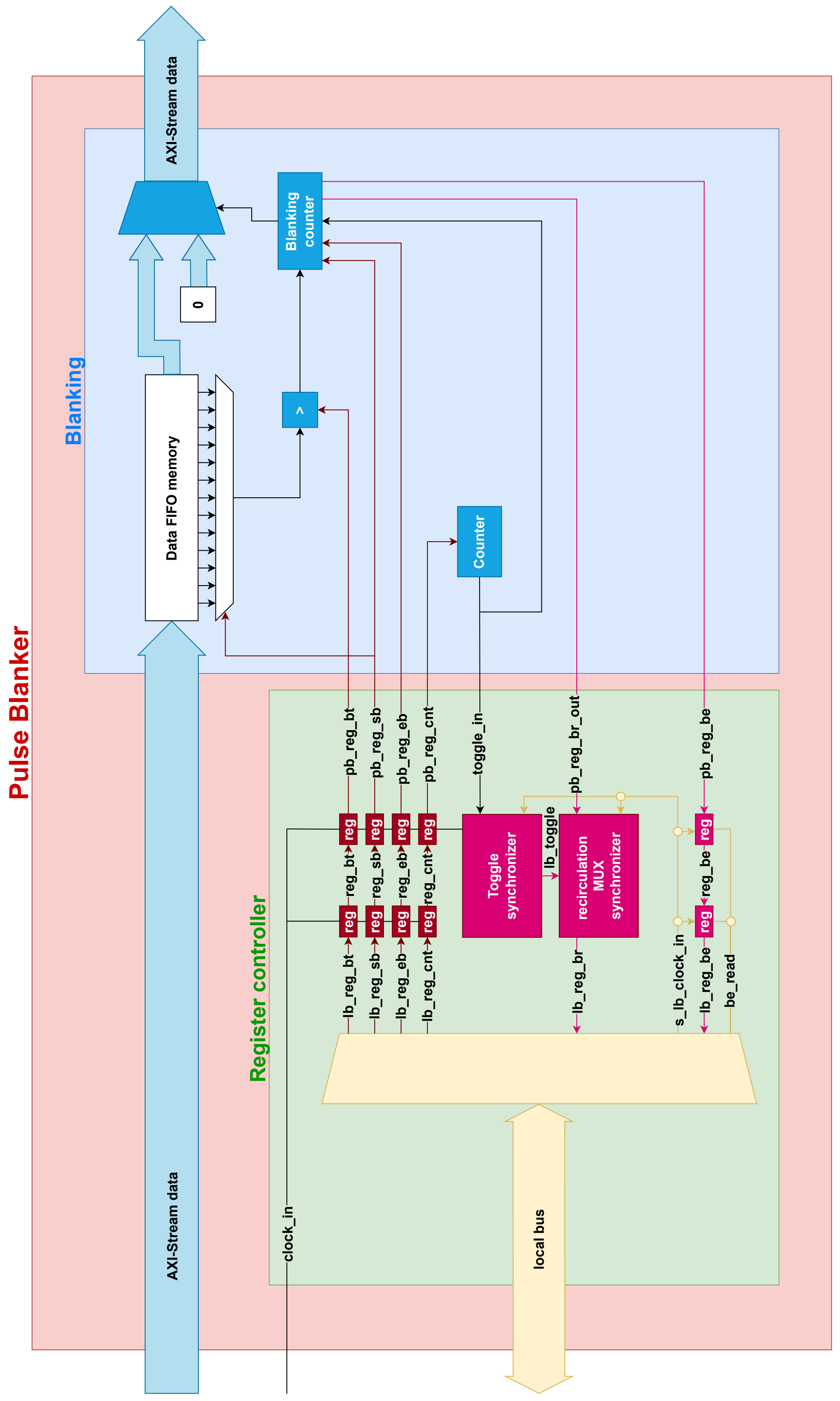


Figure 6.16: Pulse blanker scheme

6.2.3 Synthesis and implementation constraints

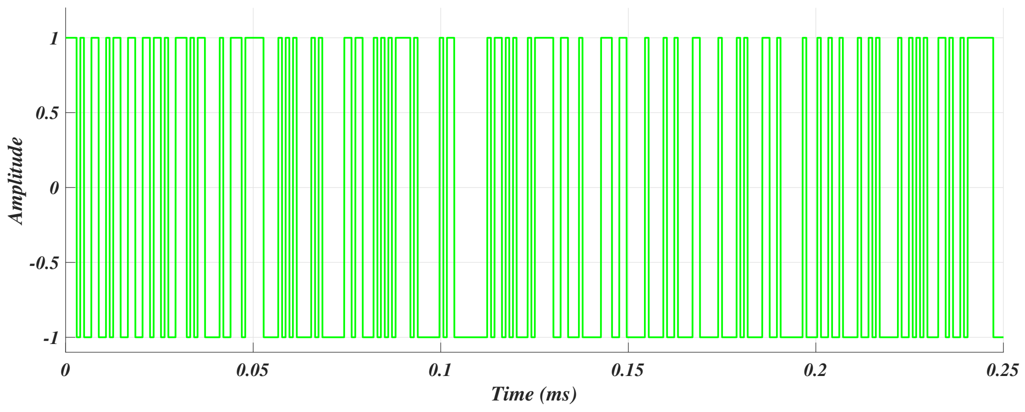
After writing all VHDL code to realize the core we also need to add some constraints in order to proceed with synthesis and implementation phase. This is particularly needed because in this design some Clock Domain Crossing between local bus and processing clock are required. For this reason if we don't add any constraint the tool will rise some timing errors.
To avoid this, since, as explained in previous section, we ensure CDC by how parameters has been managed we can set as *FALSE_PATH* all paths that are interested by Clock Domain Crossing. In this way, once the tool analyse the timing will not rise any error.
More than that, since to guarantee a correct CDC we often need a chain of FF we want to ensure that those registers will be placed one close to the other in order to improve timing. Indeed, *ASYNC_REG* property needs to be set *true* for each interested register.

7 Testing

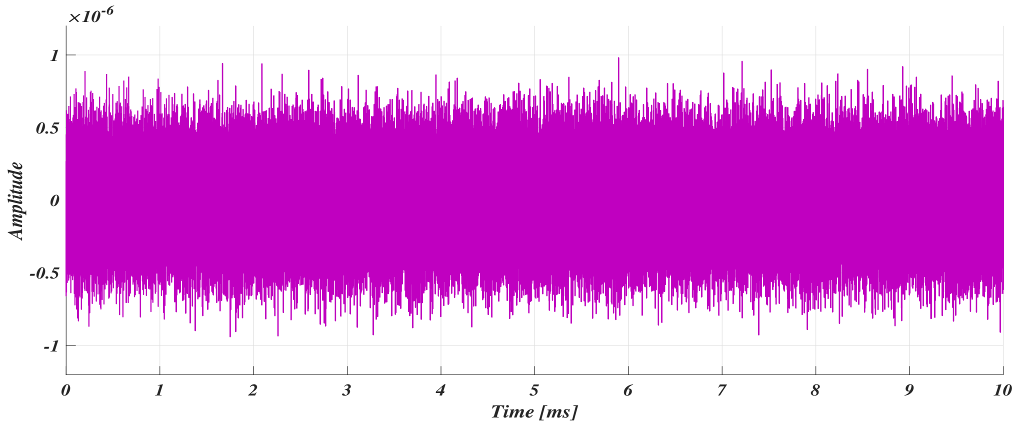
In order to test properly and extrapolate some results on how the pulse blanker work in GNSS context we need to generate some data to use as testbench.
To do that Matlab is the tool that has been used to generate signals as they are at the output of the front-end of a receiver (see chapter 2). Those signals are then reorganized as shown in figure 6.9 and saved in .bin files in order to use them as input for the implemented VHDL design.

7.1 Input samples

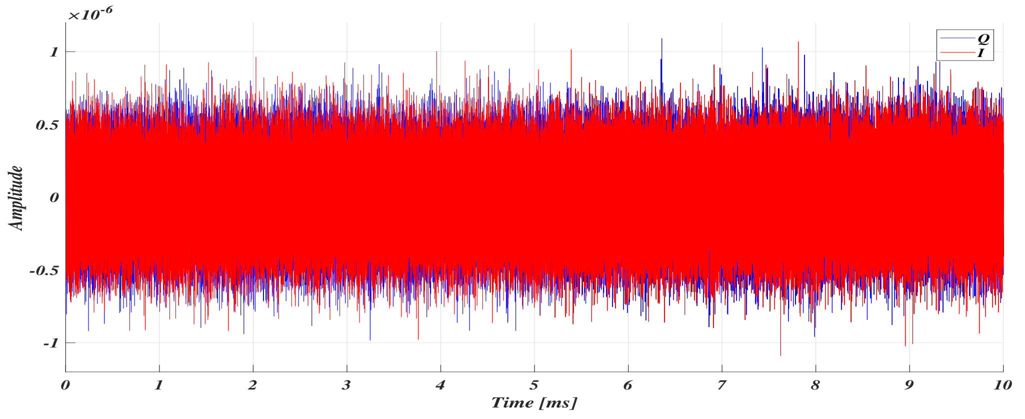
In order to generate test GNSS signals, characteristics of GPS signals L1 band have been used. So, a 1023 PRN code have been modulated with BPSK to generate 1 ms of GNSS signal. In order to have more signal to test many of those 1 ms slots have been putted one after the other. To this signal a white noise has then been added and so a GNSS signal have been created. An example is shown in the following figure.



(a) PRN signal



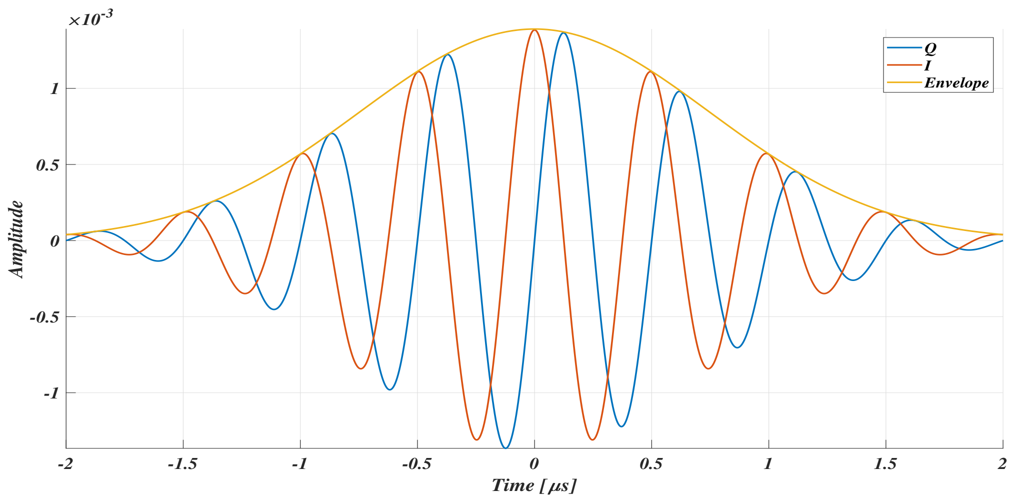
(b) GNSS signal



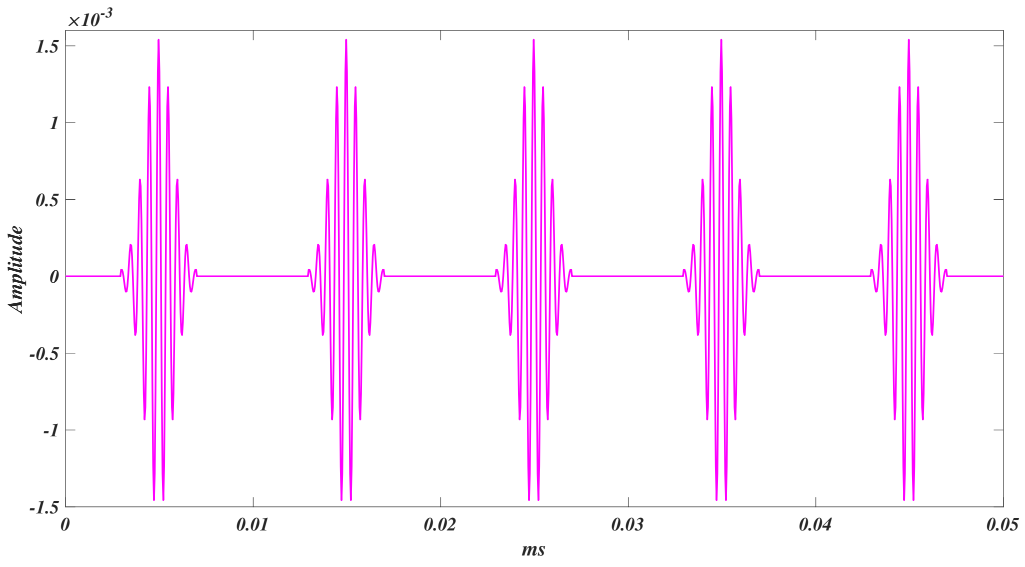
(c) GNSS signal I/Q

Figure 7.1: GNSS example

Above this signal a interference has been added. In particular, a sinusoidal burst signal composed by some Gaussian shaped bursts have been used (as it is the main interference target of the Pulse Blanker implementation). An example interference is reported in the following figure.



(a) Interference burst



(b) Interference

Figure 7.2: Interference example

If we look at the involved signals power the order of magnitude would be similar to what it is represented in figure 7.3.

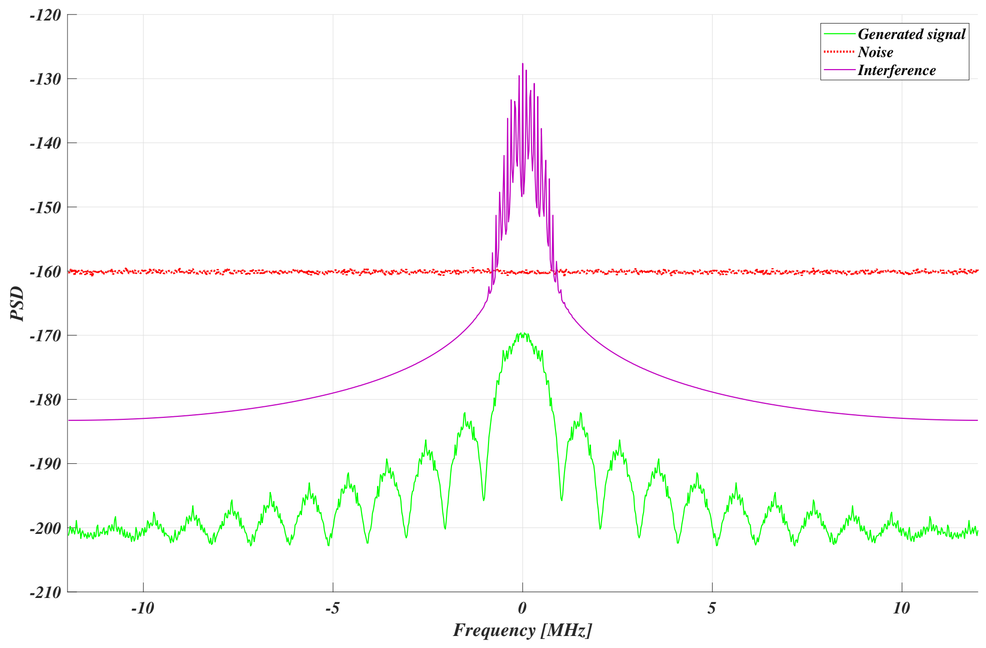
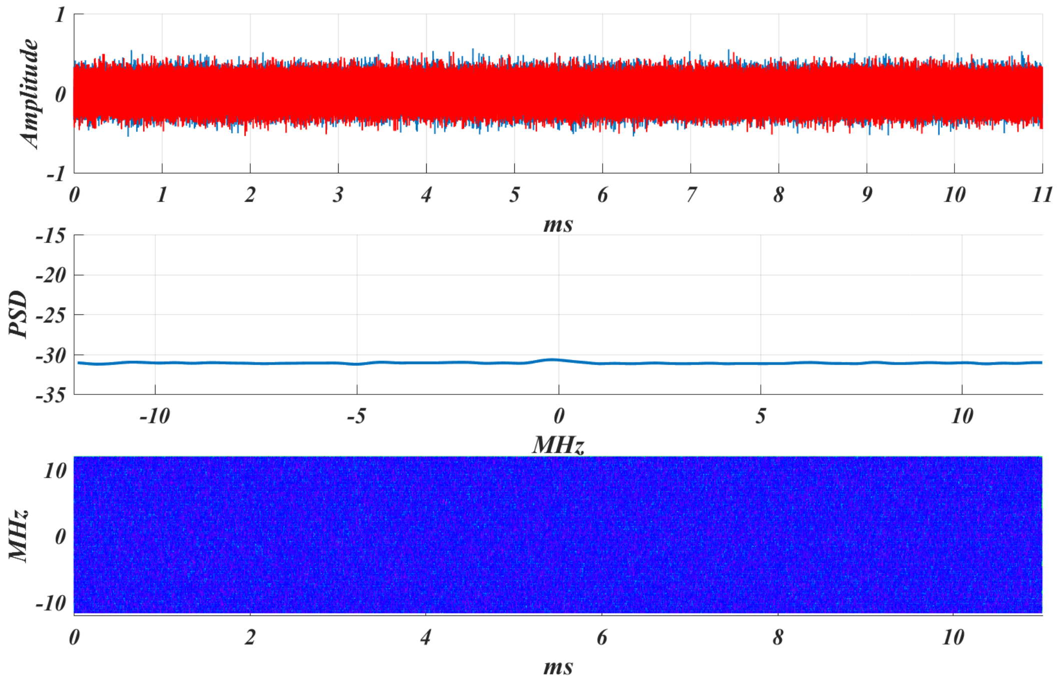
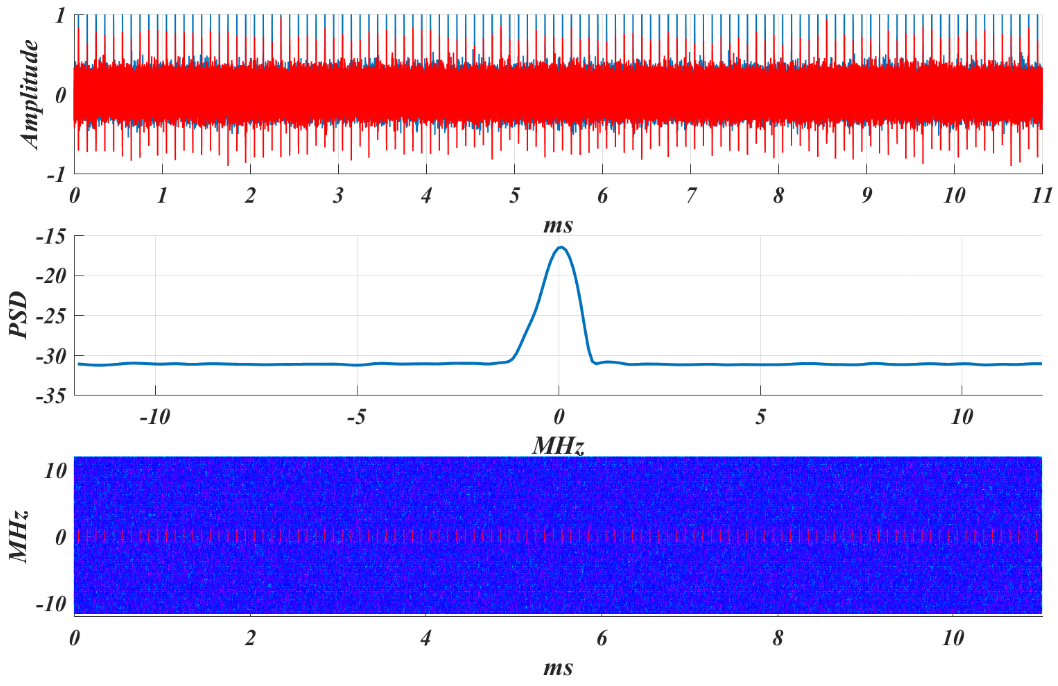


Figure 7.3: PSD GNSS signals example

Once GNSS signal with interference is generated, front-end behavior is simulated by amplifying it to fit properly (without interference) in the range (-1, 1). Amplification is kept constant for all the signal.
The result of all this is used as testing input and an example with an interference given by a sinusoidal carrier with frequency 50 KHz and bursts with a width of 4 μs and Pulse Repetition interval (PRI) of 10 kHz is given in the following figure.



(a) Test GNSS signal example without interference

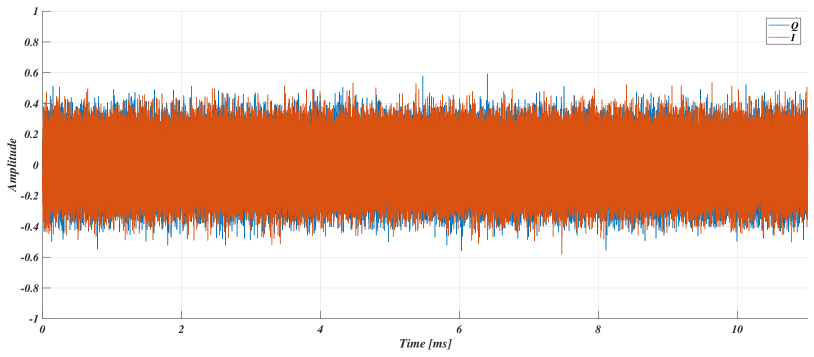


(b) Test GNSS signal example with interference

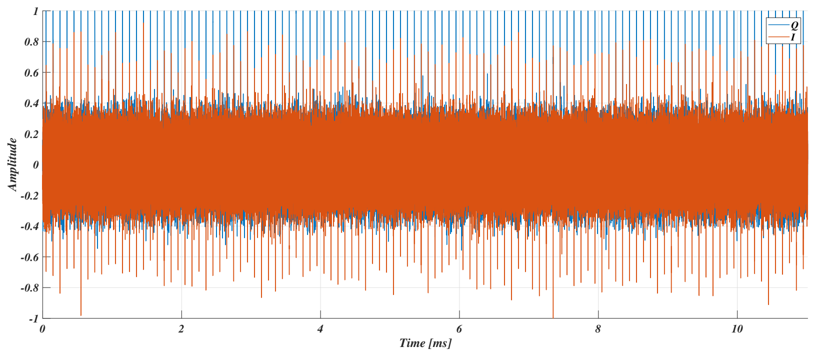
Figure 7.4: Test GNSS signal example

7.2 Verification

In order to ensure the correct core behavior a bit-true verification has been carried out. In particular, a test vector containing samples generated as explained in section 7.1 have been send and collected with a VHDL testbench through busses. The result of this has then been compared bit by bit with the result given by using the same input in a Matlab function that replicate the behavior of the Pulse Blanker functionalities. The same have been done for monitoring parameters.
Using a long input signal and random valid and ready signals for the busses gives us a great confidence to have tested most of the possible cases and a certain security that all works as expected.
Here follows some figures that shows how results given by Matlab and by the core are the same both in sample and in burst mode and both with and without interference.

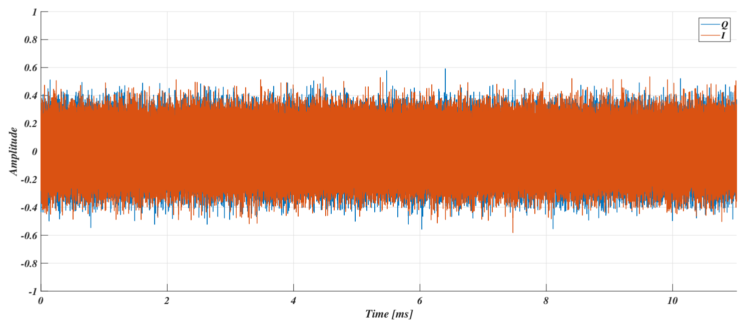


(a) Input GNSS signal result (without interference)

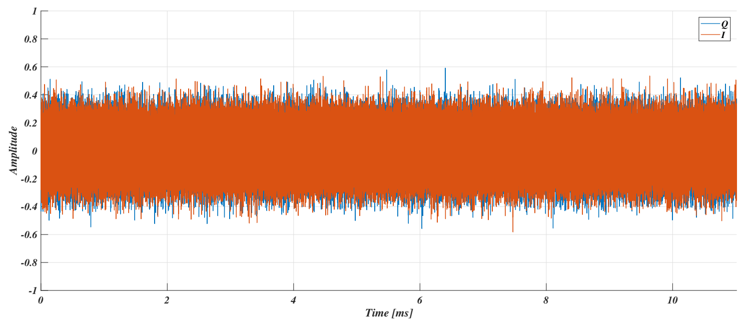


(b) Input GNSS signal result (with interference)

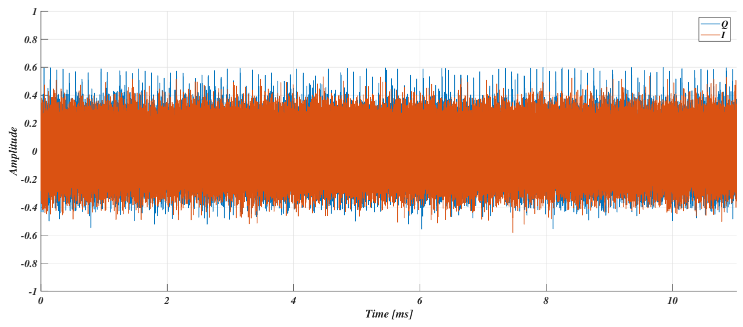
Figure 7.5: Input GNSS signals for verification



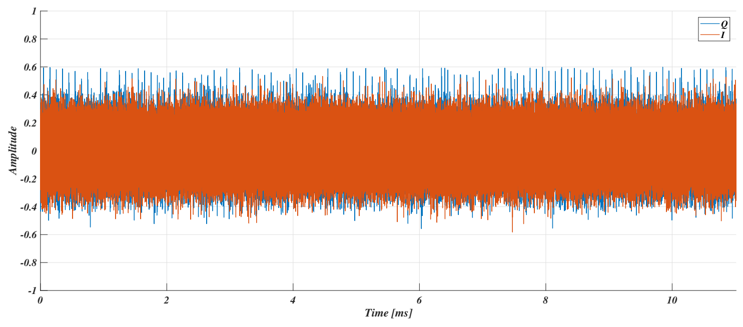
(a) Matlab GNSS signal result (without interference)



(b) Core GNSS signal results (without interference)

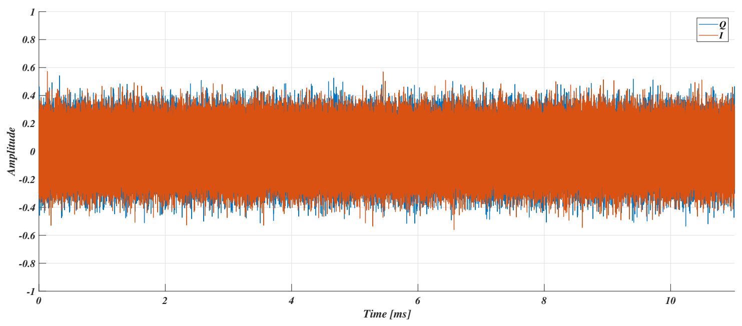


(c) Matlab GNSS signal result (with interference)

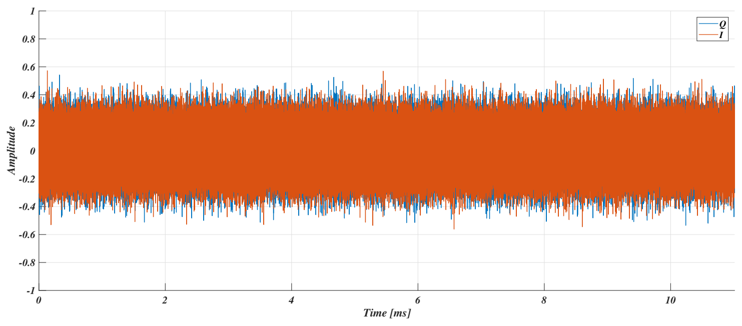


(d) Core GNSS signal results (with interference)

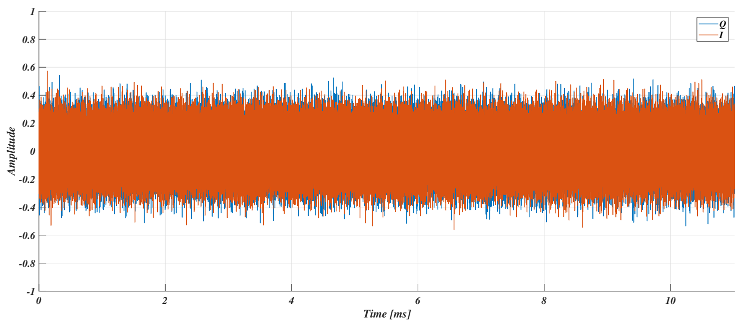
Figure 7.6: Sample mode verification



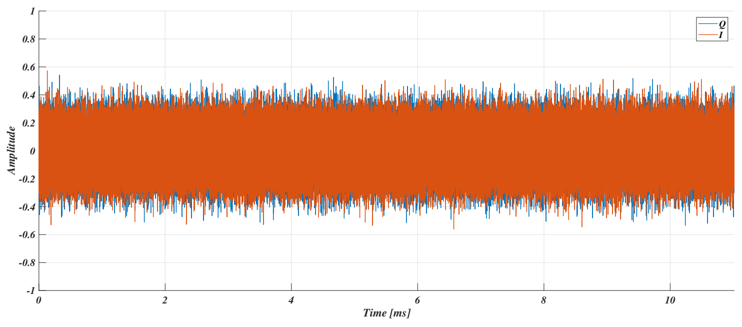
(a) Matlab GNSS signal result (without interference)



(b) Core GNSS signal results (without interference)



(c) Matlab GNSS signal result (with interference)



(d) Core GNSS signal results (with interference)

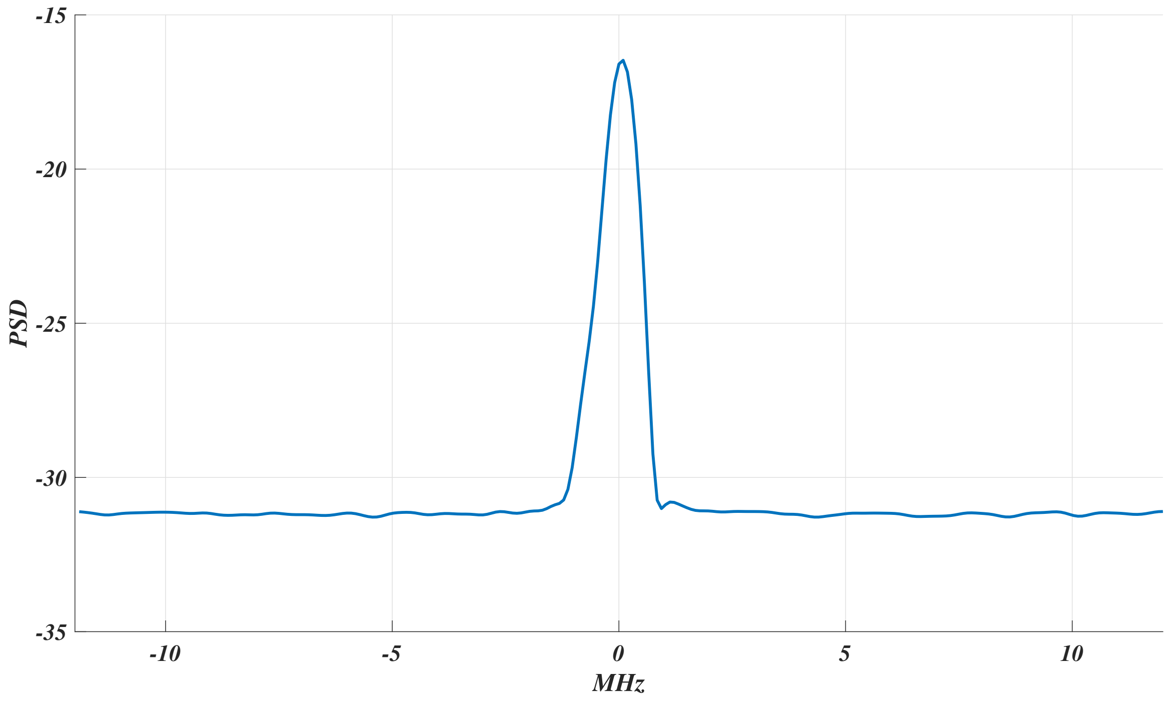
Figure 7.7: Burst mode verification

8 Results

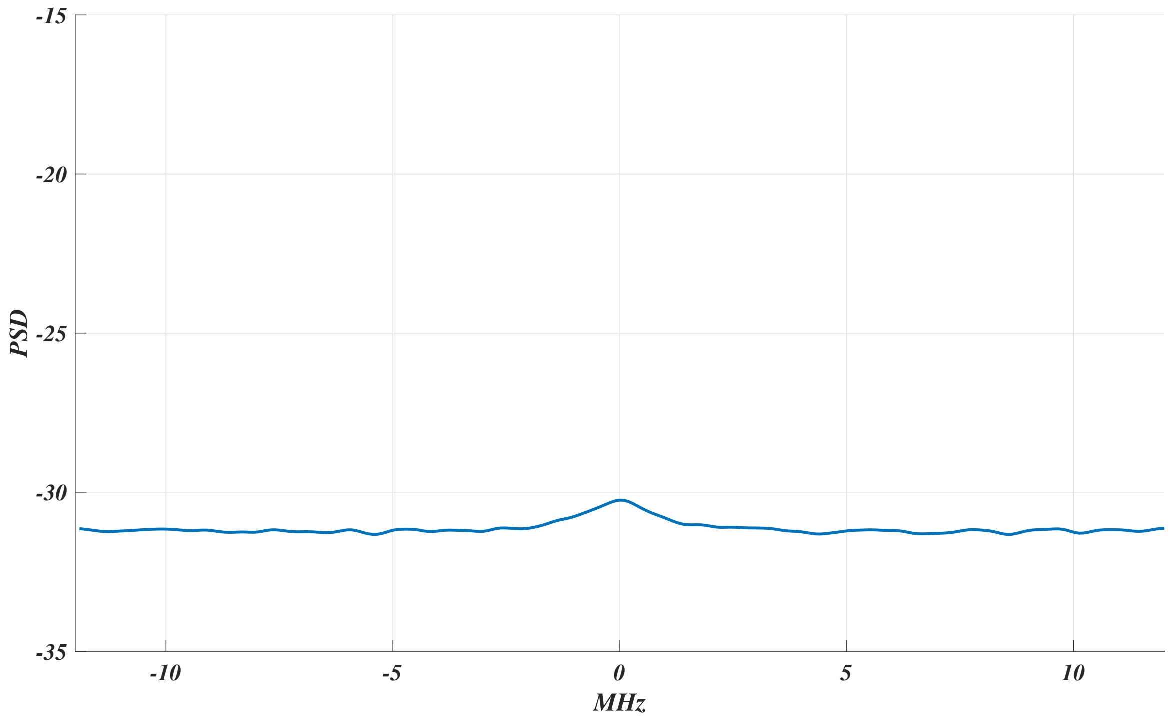
In this chapter a further analysis starting from inputs data described in chapter 7 will be carried on in order to evaluate filter performances in terms of interference mitigation.

8.1 Mitigation evaluation method

A possible evaluation of the Pulse Blanker interference mitigation property can be given by looking at what are the effects of the filter in the PSD and in the spectrogram. Here follows an example with interference carrier frequency of 50 kHz, a PRI of 10 kHz, a burst width of 4 μs and a power of the interference signal 10 dB over the noise power.

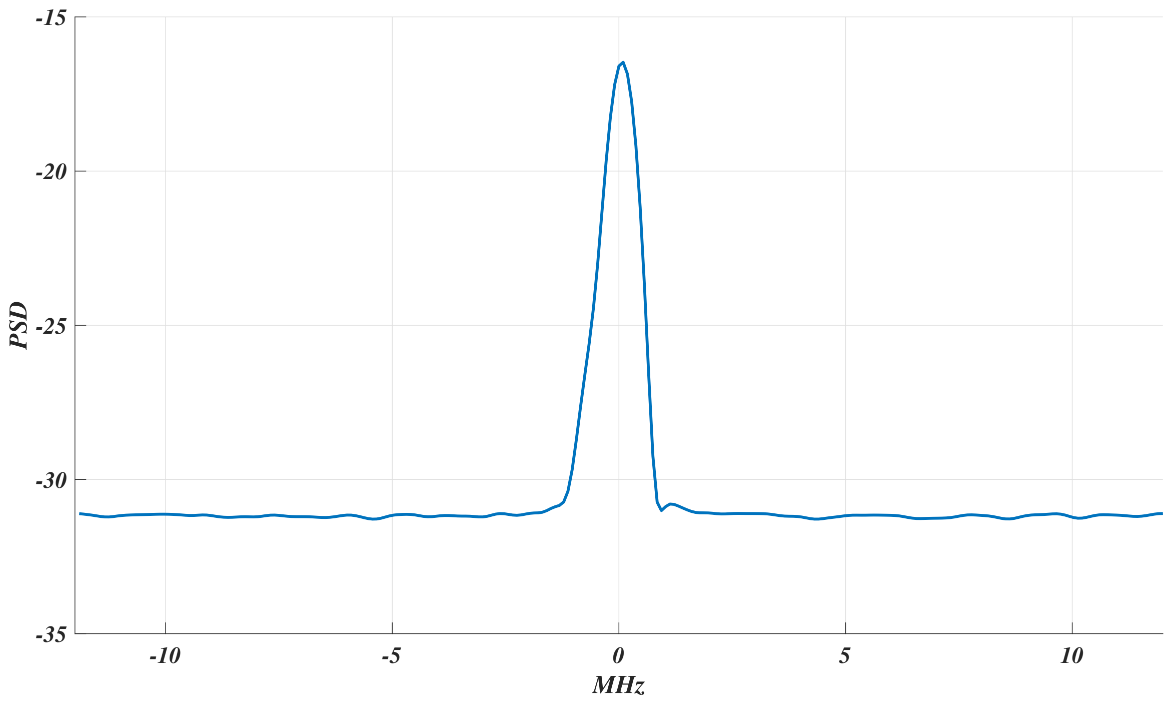


(a) GNSS signal without Pulse Blanker

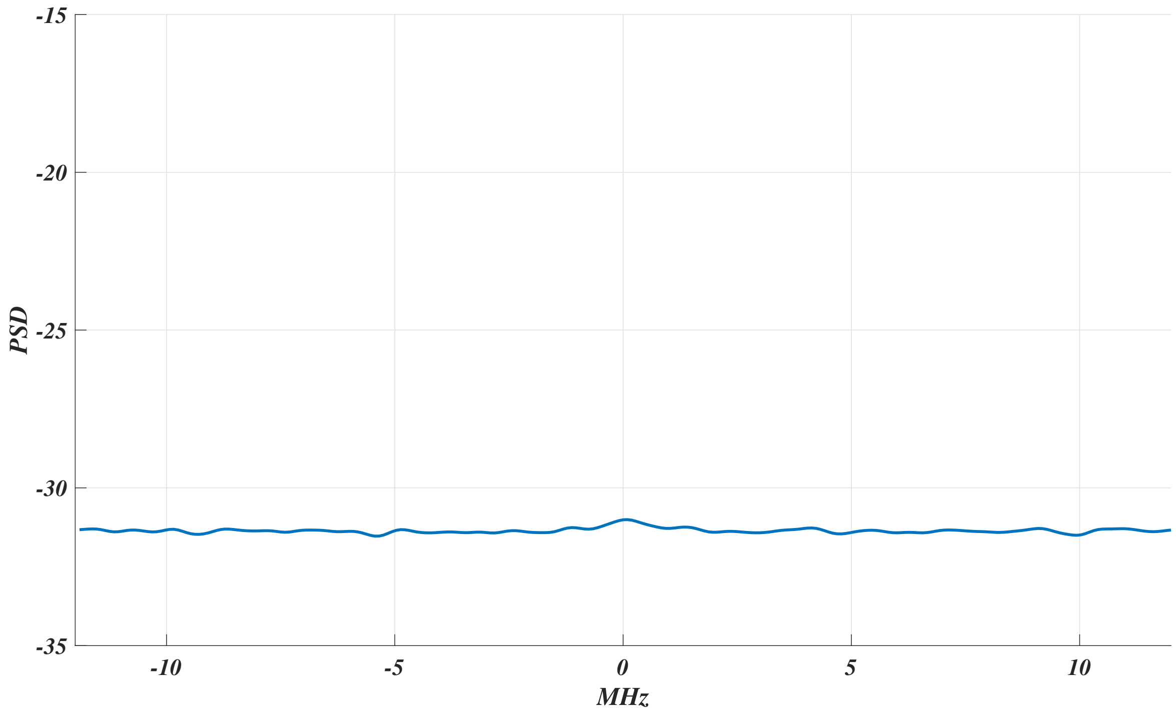


(b) GNSS signal with Pulse Blanker

Figure 8.1: PSD of GNSS signal with interference (sample mode)

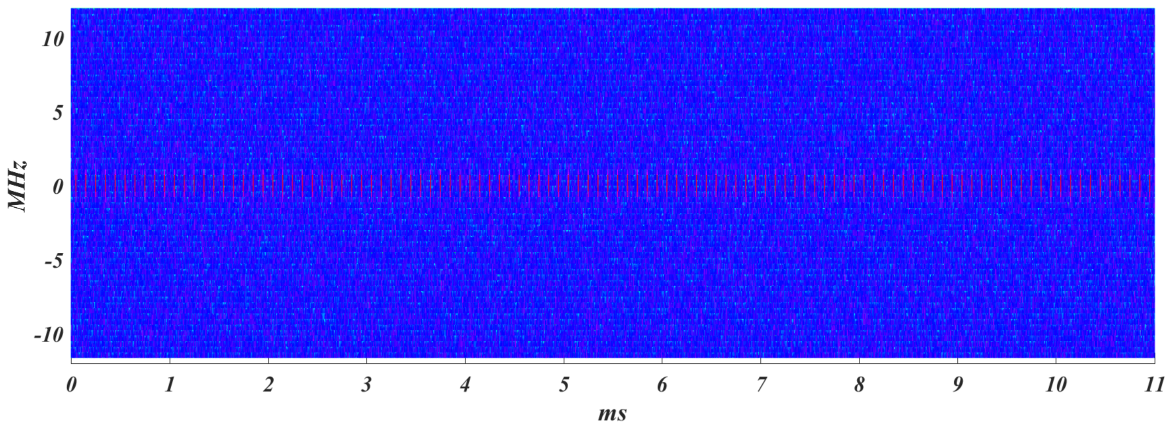


(a) GNSS signal without Pulse Blanker

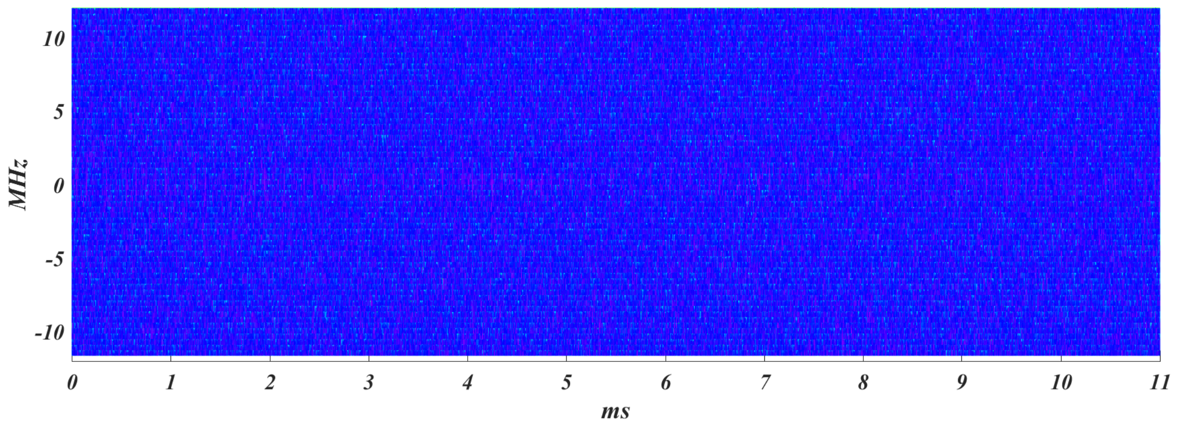


(b) GNSS signal with Pulse Blanker

Figure 8.2: PSD of GNSS signal with interference (burst mode)

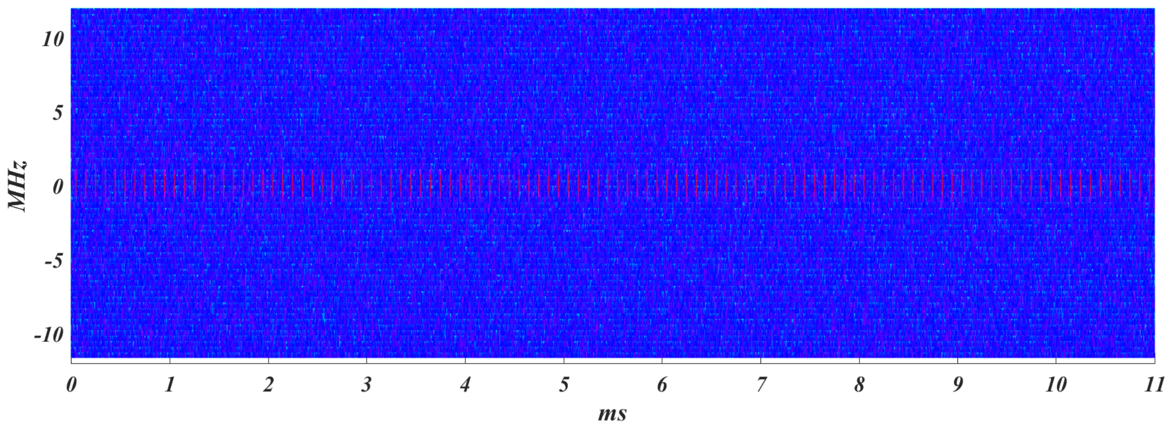


(a) GNSS signal without Pulse Blanker

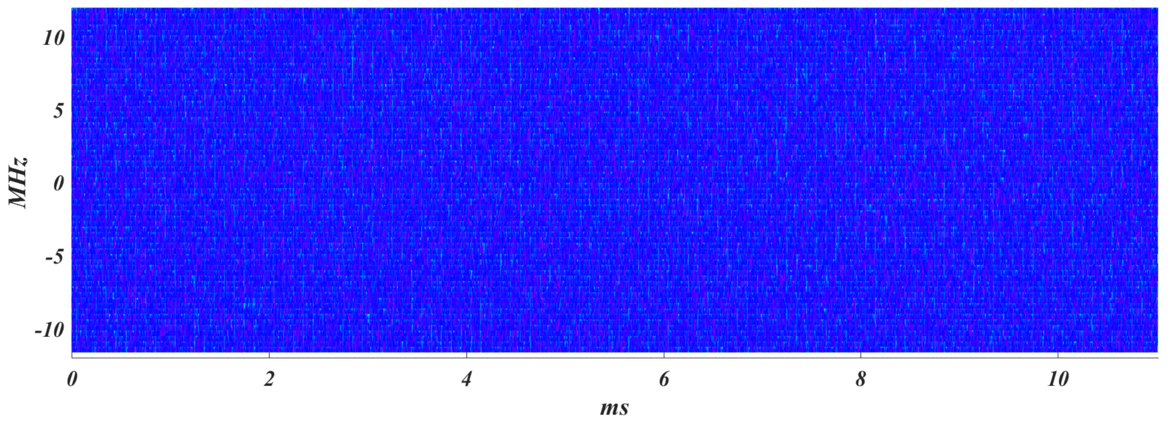


(b) GNSS signal with Pulse Blanker

Figure 8.3: Spectrogram of GNSS signal with interference (sample mode)



(a) GNSS signal without Pulse Blanker

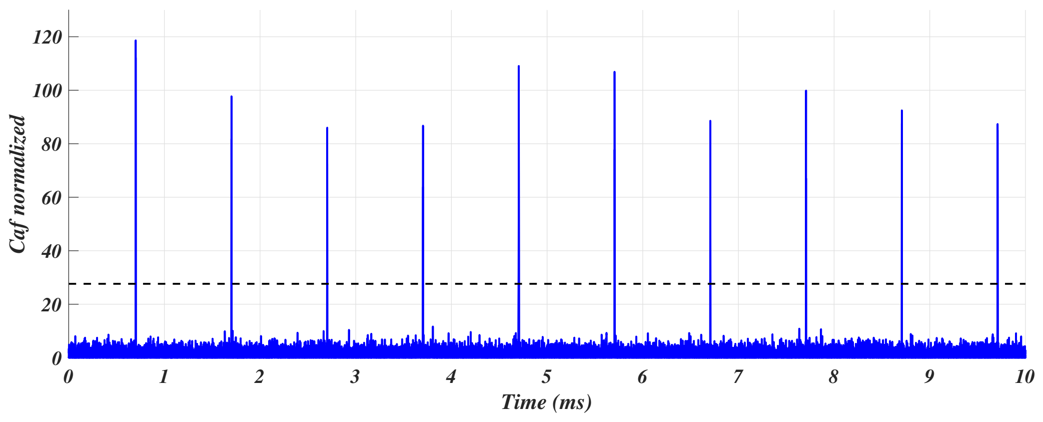


(b) GNSS signal with Pulse Blanker

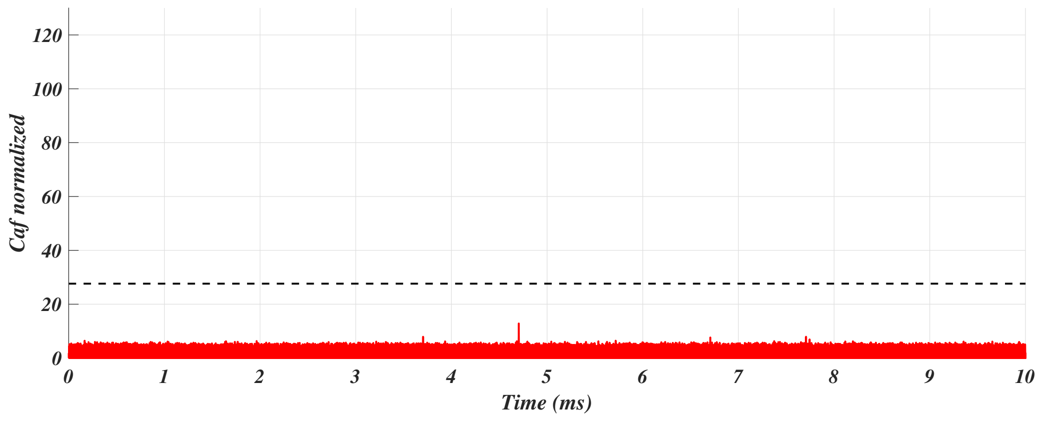
Figure 8.4: Spectrogram of GNSS signal with interference (burst mode)

This gives a qualitative idea but in order to quantify how the Pulse Blanker affect the signal quality by mitigating interference some work have to be done. By looking at figure 4.2 it results quite obvious to think that by simulating acquisition phase a measurable result about signal quality can be extrapolated. However, given the generation method used (explained in section 7.1) a simplified version of what has been described in section 2.5.1 can be adopted. Indeed, since the input signal has been created by simulation no different C/A codes correspondent to different satellites but only the one that has been used to generate the GNSS signal need to be checked and, furthermore, also Doppler frequency shift can be neglected. Therefore, this operation consists in calculating the normalized Cross-Ambiguity Function (CAF) [Appendix C] between signal and C/A code. By doing this we expect to have some peaks as result.

This is due to the fact that calculating CAF over time is like exploiting code phase in the signal and doing so the normalized CAF value will result low unless the phase shift is zero. Finally, by estimating a false alarm probability an acquisition threshold has been found in order to check if signal quality is enough to acquire or not.
Since we have GNSS signal with and without interference and both before and after the filter, we can do a more complete analysis and extrapolate some results. Specifically, for each signal, since CAF peaks are periodic of period equal to C/A length we can generate a long signal and calculate the mean value of CAF peaks. Once done that, we can estimate the quality of the signal by taking the difference of this value with the acquisition threshold (in logarithmic scale). Then, we can compare this value across all different cases and finally evaluate filter interference mitigation performances.
Matlab code of acquisition function that calculate normalized CAF can be found in Appendix D.
Here follows a figure where CAF results of GNSS signals with or without strong interference have been plotted. Dashed horizontal line represents acquisition threshold.



(a) GNSS signal without interference



(b) GNSS signal with interference

Figure 8.5: Acquisition

8.2 Filter mitigation capability evaluation and results

Following what is described in section 8.1 gives us the possibility to extrapolate results data like those reported in the following tables. Specifically, those results have been obtained with a carrier frequency of 50 kHz, a PRI of 10 kHz, a burst width of 4 μs and a power of the interference signal 10 dB over the noise power. Security of acquisition is given by the difference between the mean peak value of the CAF normalized and the acquisition threshold.

Results without Pulse Blanker			
		Without interference	**With interference**
Sample mode	**Mean peak value CAF normalized**	19.43 dB	10.33 dB
	Acquisition threshold	14.41 dB	14.41 dB
	Security of acquisition	5.02 dB	-4.08 dB
Burst mode	**Mean peak value CAF normalized**	19.51 dB	9.51 dB
	Acquisition threshold	14.41 dB	14.41 dB
	Security of acquisition	5.09 dB	-4.90 dB

Table 8.1: Results example without Pulse Blanker

Results with Pulse Blanker			
		Without interference	**With interference**
Sample mode	**Mean peak value CAF normalized**	19.43 dB	18.67 dB
	Acquisition threshold	14.41 dB	14.41 dB
	Security of acquisition	5.02 dB	4.26 dB
Burst mode	**Mean peak value CAF normalized**	19.51 dB	19.31 dB
	Acquisition threshold	14.41 dB	14.41 dB
	Security of acquisition	5.09 dB	4.89 dB

Table 8.2: Results example with Pulse Blanker

Results comparison			
		Without interference	**With interference**
Sample mode	**Security of acquisition (without Pulse Blanker)**	5.02 dB	-4.08 dB
	Security of acquisition (with Pulse Blanker)	5.02 dB	4.26 dB
	Security of acquisition (gain)	0.00 dB	8.34 dB
Burst mode	**Security of acquisition (without Pulse Blanker)**	5.09 dB	-4.90 dB
	Security of acquisition (with Pulse Blanker)	5.09 dB	4.89 dB
	Security of acquisition (gain)	0.00 dB	9.79 dB

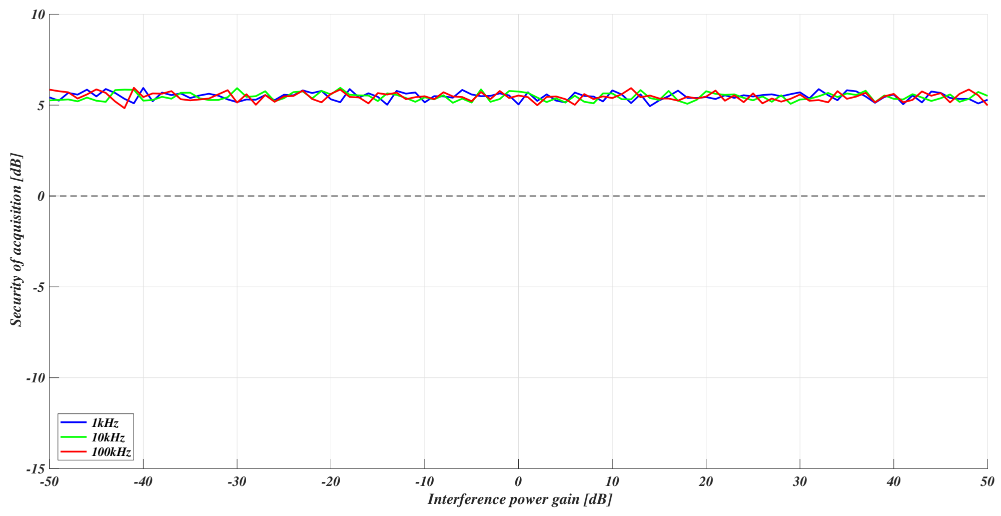
Table 8.3: Results example without Pulse Blanker

From this tables it is clear that we can evaluate Pulse Blanker behavior and quantify its beneficial effect on GNSS signal quality.
In order to do further analysis this type of results have been obtained for a range of different interference power. In particular, the power of the GNSS signal noise have been taken as reference and the interference added have been multiplied to a gain that allow to set interference power with a defined dB difference from that reference. In this

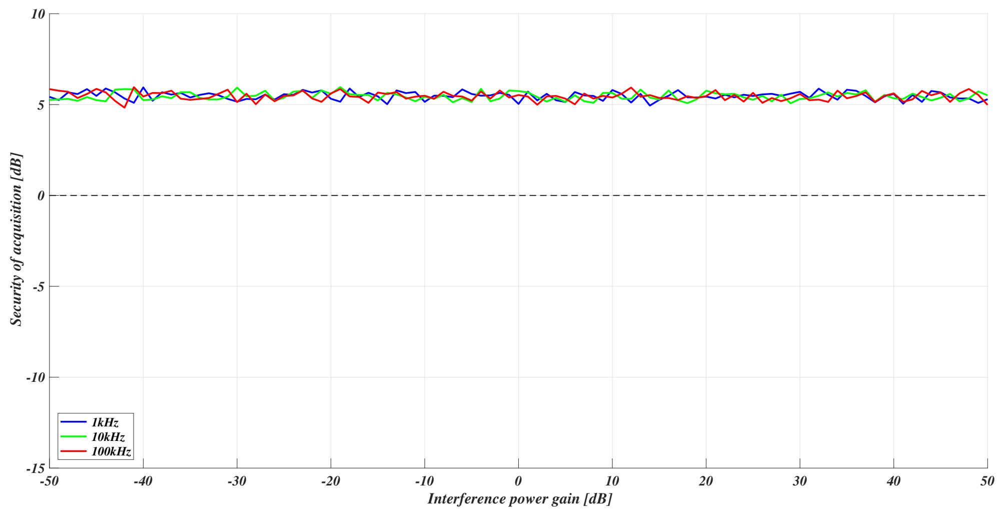
way results in range noise power ± 50 dB of interference power have been exploited. This have been then repeated for 1, 10 and 100 kHz of PRI. In the following sections are reported all results obtained.

8.3 Sample mode results

In this section results regarding Pulse Blanker sample mode are reported.

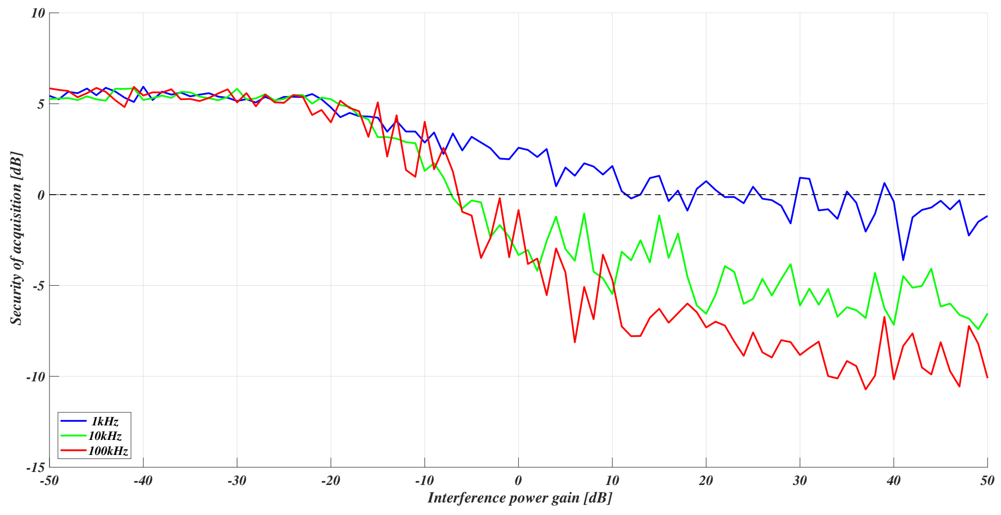


(a) Security of acquisition without Pulse Blanker

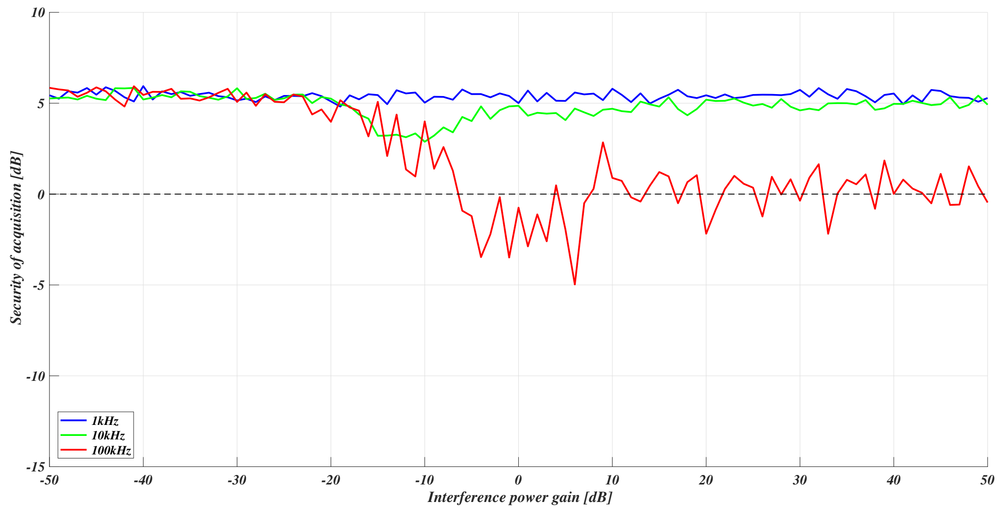


(b) Security of acquisition with Pulse Blanker

Figure 8.6: Security of acquisition with sample mode (without interference)



(a) Security of acquisition without Pulse Blanker

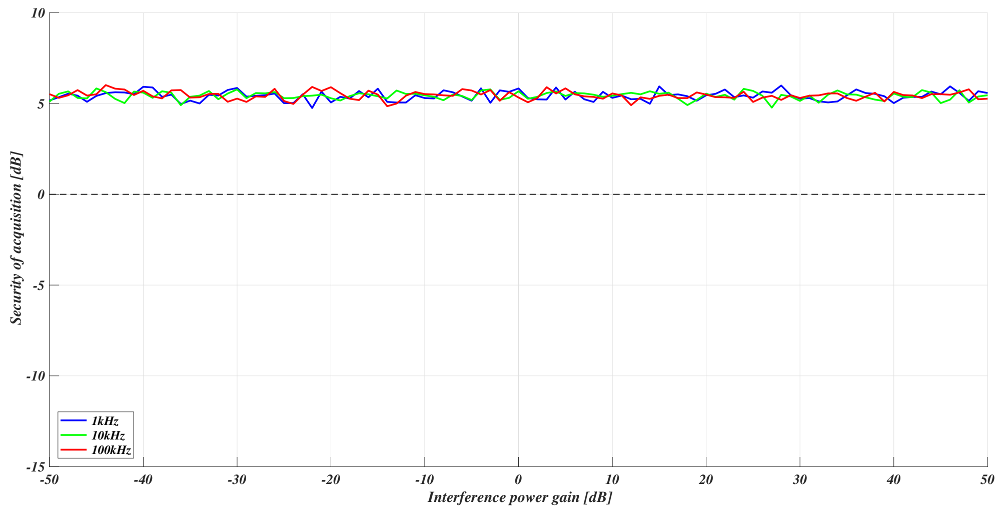


(b) Security of acquisition with Pulse Blanker

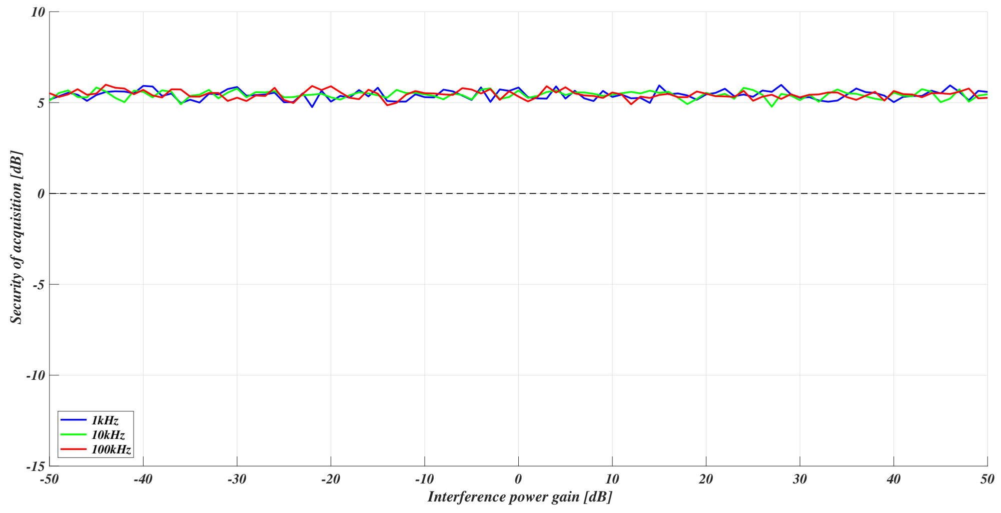
Figure 8.7: Security of acquisition with sample mode (with interference)

8.4 Burst mode results

In this section results regarding Pulse Blanker burst mode are reported.

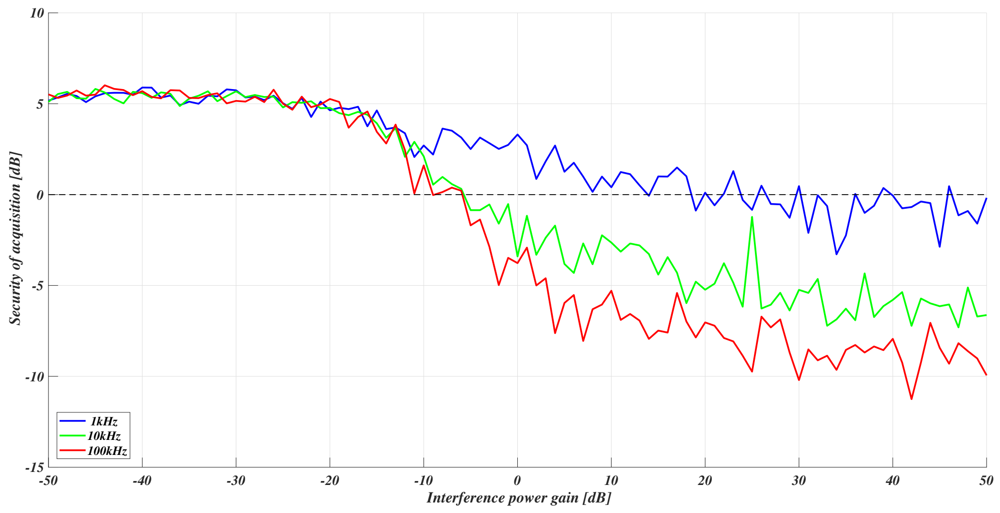


(a) Security of acquisition without Pulse Blanker

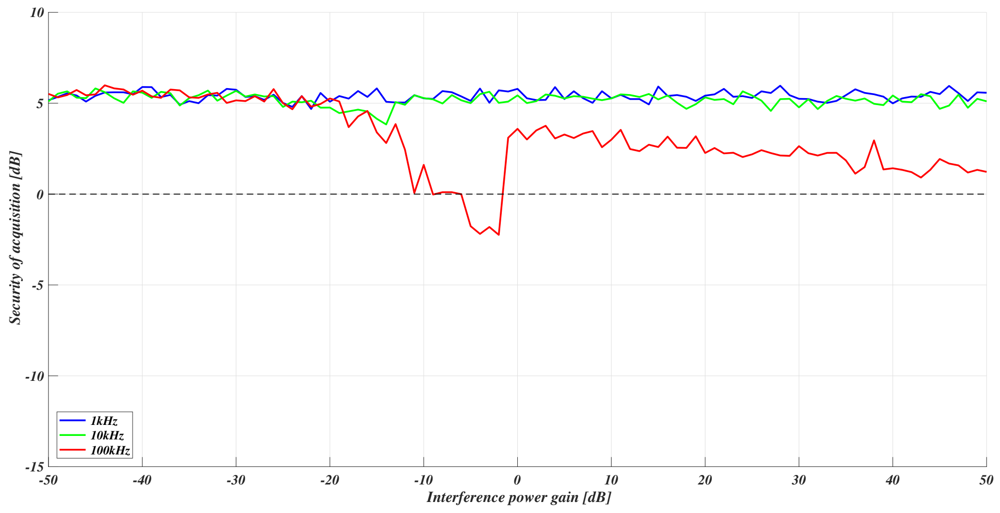


(b) Security of acquisition with Pulse Blanker

Figure 8.8: Security of acquisition with burst mode (without interference)



(a) Security of acquisition without Pulse Blanker



(b) Security of acquisition with Pulse Blanker

Figure 8.9: Security of acquisition with burst mode (with interference)

9 Conclusions

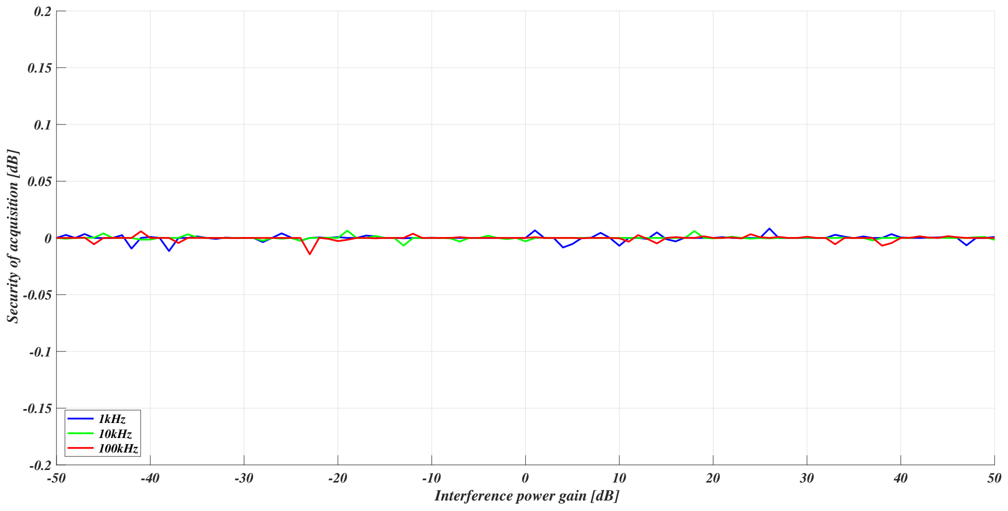
To start we can look at figures 8.1, 8.2, 8.3 and 8.4 to notice that for sure interference have been mitigated. Peak in PSD and pink lines in spectrogram (bursts of the interference) have been reduced or canceled totally.
A further analysis have been than made to evaluate filter performance by starting from results collected in chapter 8.
Looking at figure 8.6 and 8.8 it is quite easy to understand that in presence of a signal without interference the quality of the signal itself is not modified. Indeed, if we plot the difference in security of acquisition with and without the Pulse Blanker we obtain figure 9.1 and it is almost everywhere zero. This shows how the Pulse Blanker doesn't affect the signal when interference is not present if a proper blanking threshold (BT) is chosen.
Behavior under interference need to be then analyzed. In this case if we collect results from figure 8.7 and 8.9 we can compare results for the same PRI and we obtain figure 9.2, 9.3 and 9.4. Those graphs show clearly the effectiveness of the filter that cuts interference peaks and allow to increase the security of acquisition. The last acquisition possible is in fact done under harsher conditions in terms of interference power (represented by the zero crossing point of the curve that moves to the right in the case with the Pulse Blanker).
Another thing worth noticing is that with the increase of the PRI the effectiveness of the filter decrease. This is because, obviously, the more bursts of interference there are in a time interval, the more the filter blank. This means that at a certain PRI frequency samples that are blanked are so many that acquisition is not possible anyway. Therefore, this filter just cut overthreshold samples and it is not that useful when bursts have an

high frequency and a power level not that high respect to the GNSS signal. This can be seen clearly in graph 9.4. Here, when interference power is close to the noise one, filter effect is almost unnoticeable. This is because peaks of interference are present with a frequency such that acquisition is compromised but the power of those interference peaks is approximately within signal range. That means that Pulse Blanker can't do nothing (if threshold is lowered then it will cut also most of the signal).
With that been sad, if we compare sample and burst mode we can see that the performance of the last is clearly better even in this condition. This is because, as soon as interference power became big enough to let us set a proper threshold, burst mode blank more signal but all interference is cutted from the signal helping with the later acquisition phase. The more the frequency of the interference bursts is high the bigger the difference become visible. Sample mode increase the security of acquisition slowly after that moment, instead, burst mode allow to increase it by a lot more and with the need of less interference power than sample mode. This difference can be seen also in PSD plot. If we zoom 8.1 and 8.2 figures we can appreciate the difference in quality of signal that the two modes brings by lowering the peak. This is shown in figure 9.5. Moreover, a direct comparison between security of acquisition of both modes can be seen in figure 9.6 and 9.7. The trend previously described can be observed in those plots.
Last but not least, we need to evaluate how many resources are allocated in order to implement the Pulse Blanker inside the PL of the Zynq. This is because even if good mitigation results have been shown we need to ensure that not too much logic is needed. Indeed, this can compromise receiver overall performances, timing and efficiency.
Considering only the Pulse Blanker core the number of resources that need to be added respect to the previous implementation that use only the dual-FDAF are reported in the following table:

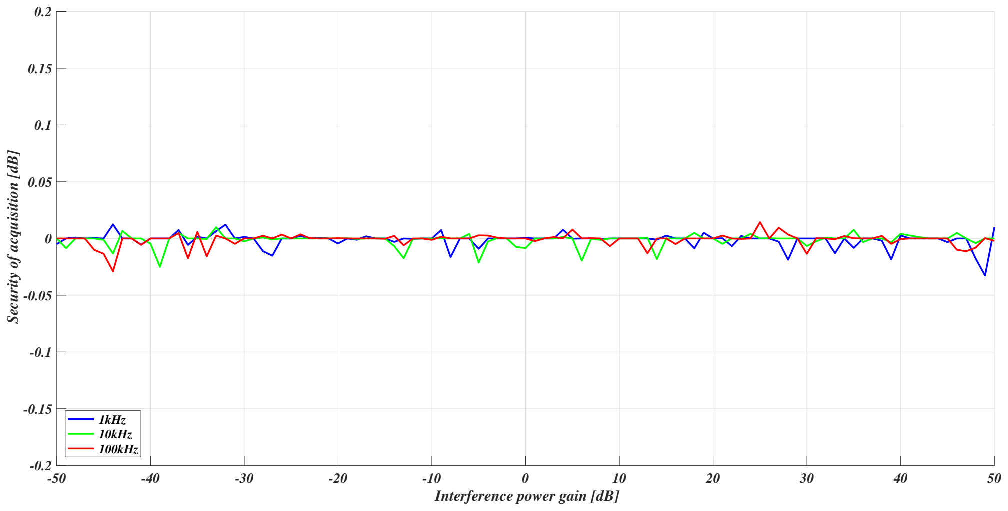
Pulse Blanker resources utilization		
Resource type	**Utilization [$n°$]**	**FPGA utilization [%]**
FF	1153	0.335
LUT	1496	0.870
LUTRAM	704	0.001

Table 9.1: Pulse Blanker resources utilization

Looking at those numbers we can clearly asses that, even if some memories, registers and logic resources are involved the overall impact is quite irrelevant.
This means that, by adding this core, with a little increment in the resources involved, a good mitigation against spot interference can be achieved reaching the main goal of the thesis project.

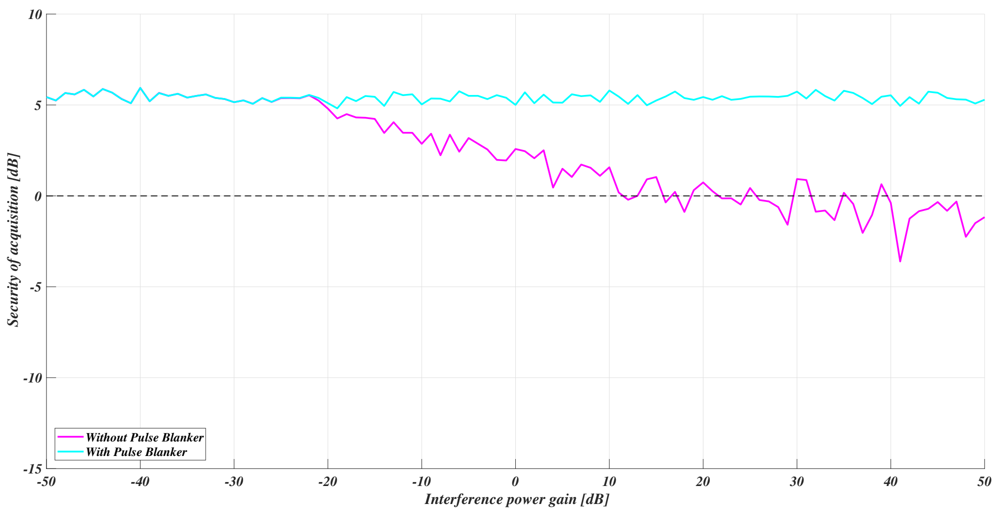


(a) Sample mode

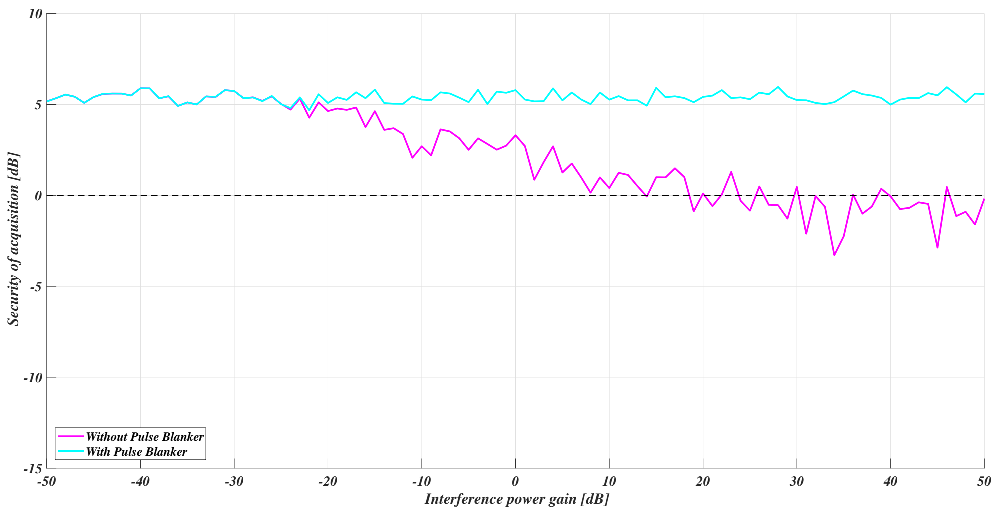


(b) Burst mode

Figure 9.1: Security of acquisition with Pulse Blanker - security of acquisition without Pulse Blanker (without interference)

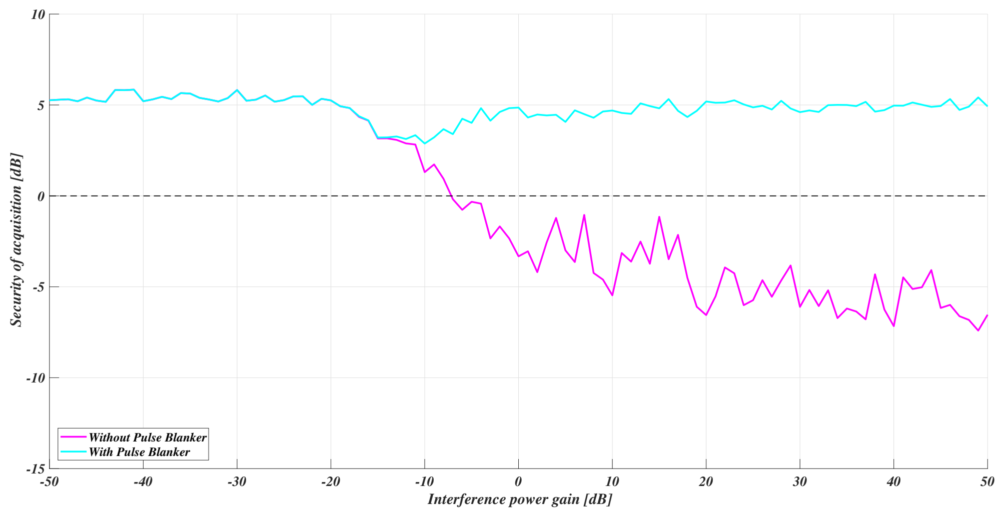


(a) Sample mode

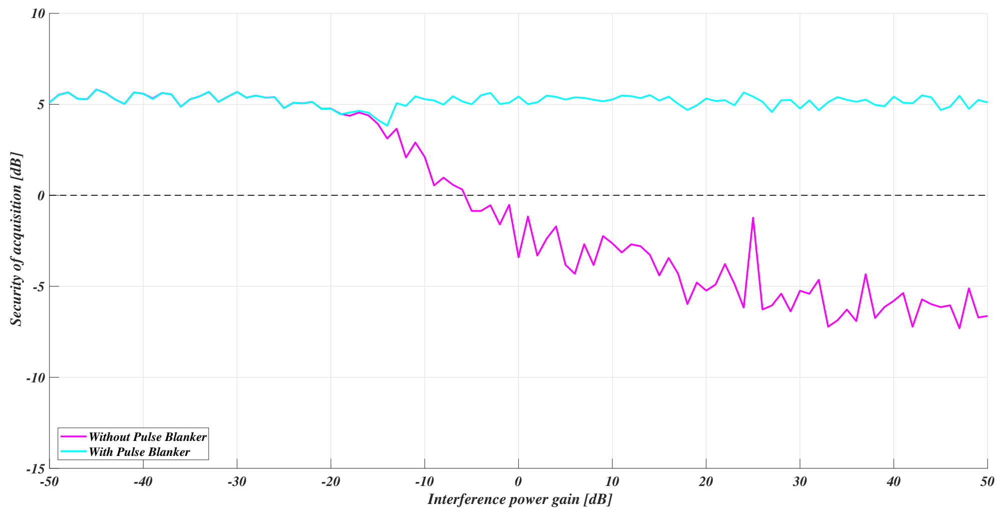


(b) Burst mode

Figure 9.2: Security of acquisition with Pulse Blanker and interference PRI = 1 kHz

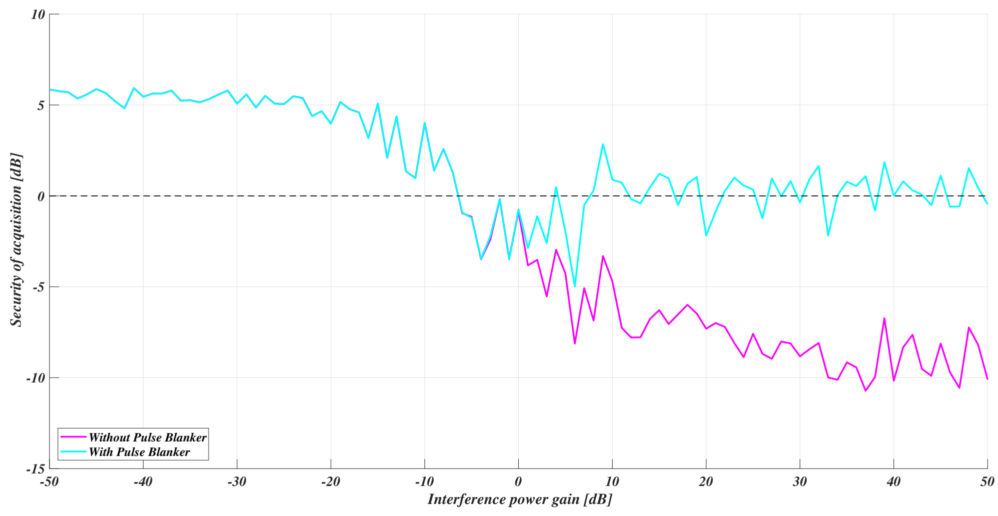


(a) Sample mode

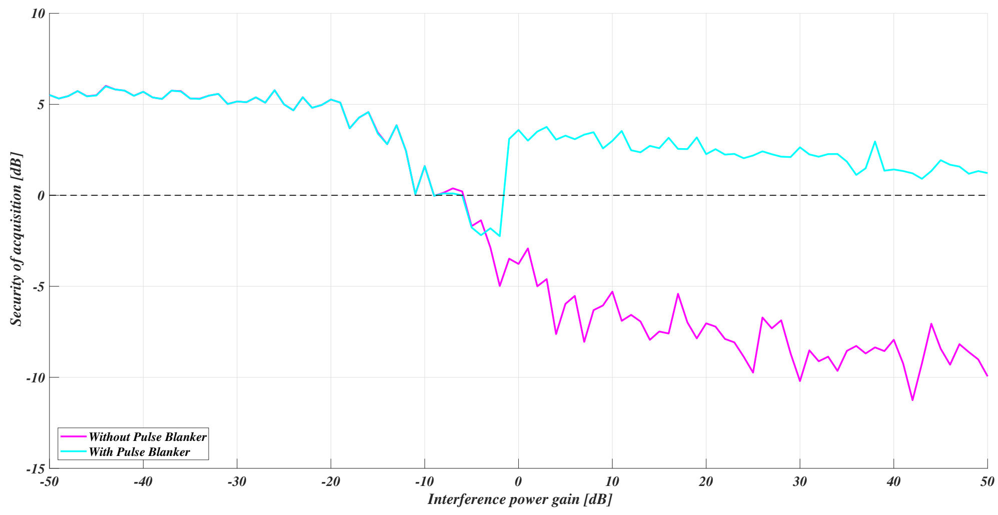


(b) Burst mode

Figure 9.3: Security of acquisition with Pulse Blanker and interference PRI = 10 kHz

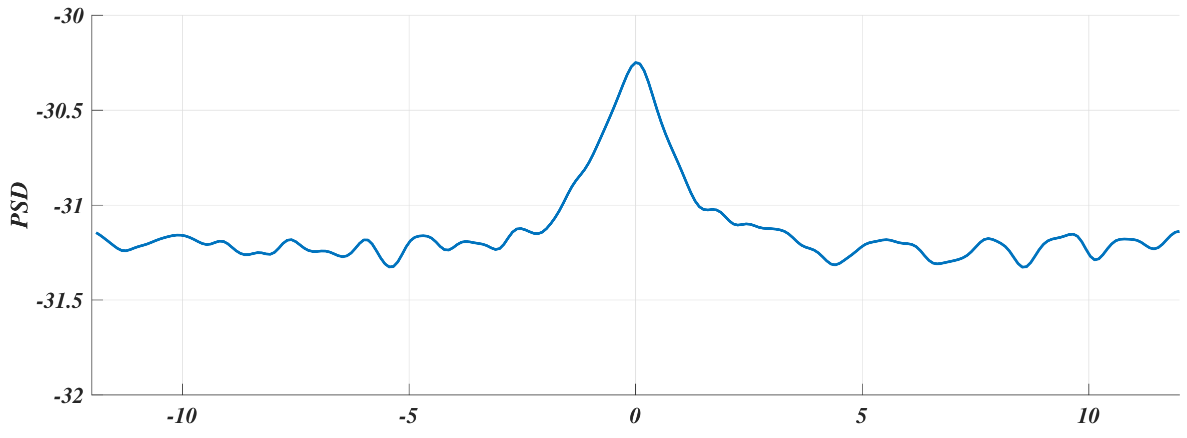


(a) Sample mode

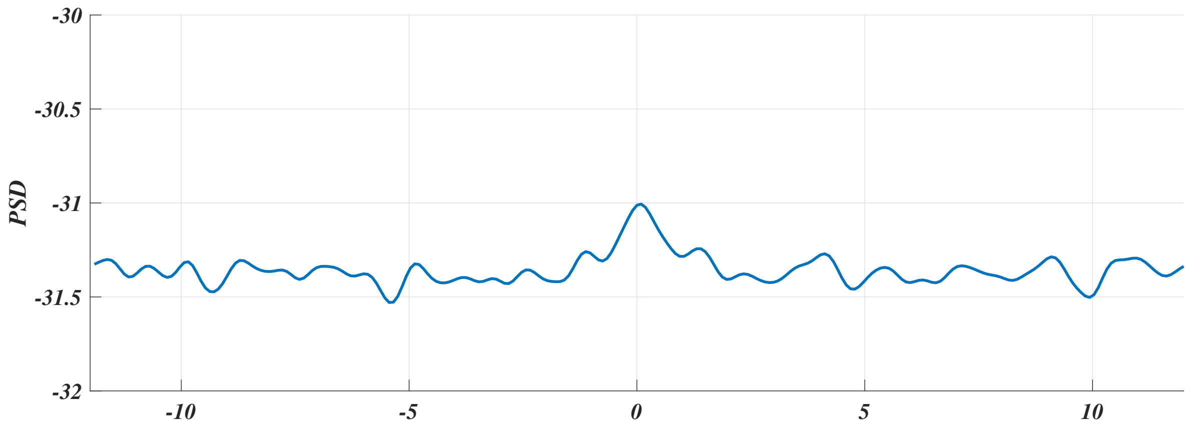


(b) Burst mode

Figure 9.4: Security of acquisition with Pulse Blanker and interference PRI = 100 kHz

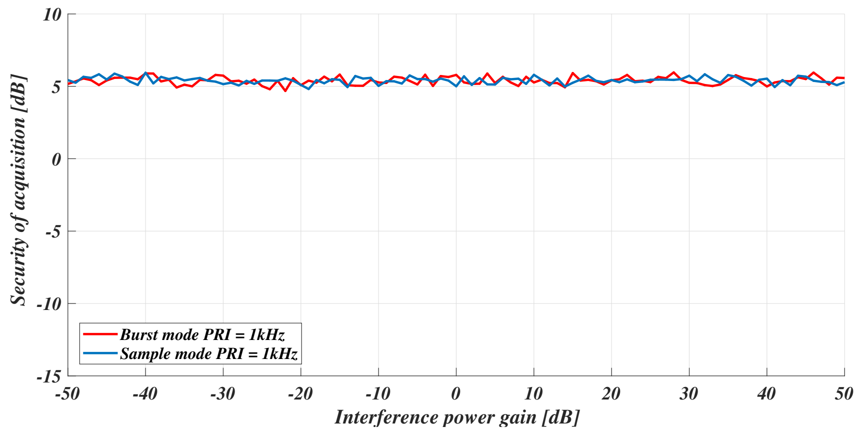


(a) Sample mode

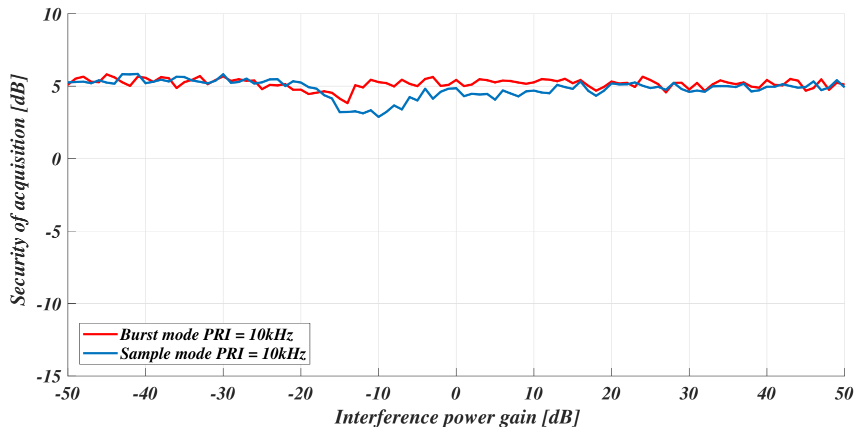


(b) Burst mode

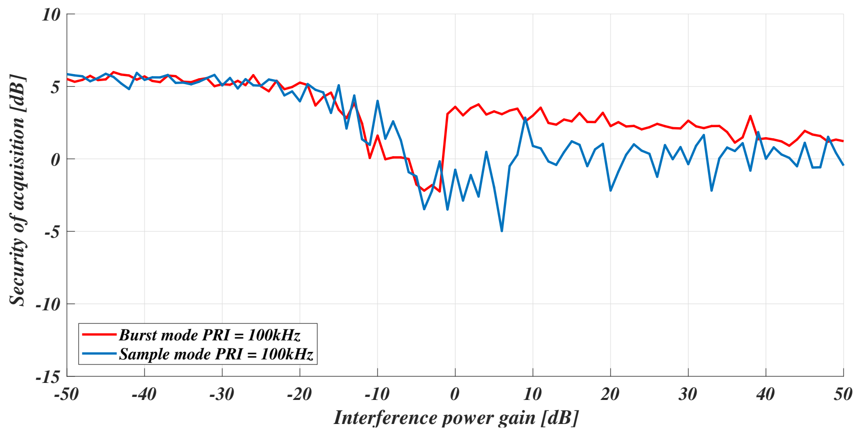
Figure 9.5: PSD zoom of figure 8.1 and 8.2



(a) PRI = 1kHz



(b) PRI = 10kHz



(c) PRI = 100kHz

Figure 9.6: Sample mode vs burst mode

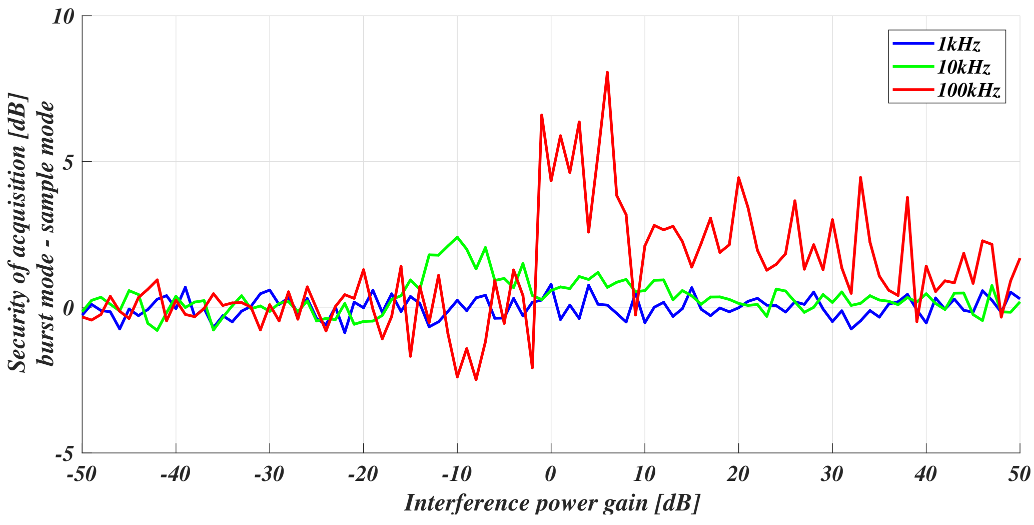


Figure 9.7: Security of acquisition difference between burst mode and sample mode

References

GPS signal references

[1] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, *A SOFTWARE-DEFINED GPS AND GALILEO RECEIVER A Single-Frequency Approach*. Birkhäuser, 2007, isbn: 978-0-8176-4390-4. doi: 10.1007/978-0-8176-4540-3.

[2] M. Foucras, J. Leclère, C. Botteron, O. Julien, C. Macabiau, P.-A. Farine, and B. Ekambi, "Study on the cross-correlation of gnss signals and typical approximations," *GPS Solutions*, Aug. 2016. doi: 10.1007/s10291-016-0556-7. [Online]. Available: https://enac.hal.science/hal-01353985.

Interference references

[3] N. Fadaei, *Detection, characterization and mitigation of gnss jamming interference using pre-correlation methods*, 2016. doi: 10.11575/PRISM/25598. [Online]. Available: https://prism.ucalgary.ca/server/api/core/bitstreams/bb0542a9-7d27-4c15-9222-4a61c41c25c4/content.

[4] J. Vilà-Valls, N. Linty, P. Closas, F. Dovis, and J. T. Curran, "Survey on signal processing for gnss under ionospheric scintillation: Detection, monitoring, and mitigation," *NAVIGATION: Journal of the Institute of Navigation*, vol. 67, no. 3, pp. 511–535, 2020, issn: 0028-1522. doi: 10.1002/navi.379. eprint: https://navi.ion.org/content/67/3/511.full.pdf. [Online]. Available: https://navi.ion.org/content/67/3/511.

[5] E. R. de Paula, A. R. F. Martinon, C. Carrano, A. O. Moraes, J. A. C. F. Neri, J. R. Cecatto, M. A. Abdu, A. C. Neto, J. F. G. Monico, W. da Costa Silva, B. C. Vani, I. S. Batista, O. Mendes, J. R. de Souza, and A. L. A. Silva, "Solar flare and radio burst effects on gnss signals and the ionosphere during september 2017," *Radio*

Science, 2022, e2021RS007418 2021RS007418. doi: https://doi.org/10.1029/2021RS007418. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2021RS007418. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021RS007418.

Mitigation references

[6] M. Raimondi, O. Julien, C. Macabiau, and F. Bastide, "Mitigating pulsed interference using frequency domain adaptive filtering," pp 2251 –2260, Sep. 2006. [Online]. Available: https://enac.hal.science/hal-01021790.

[3] N. Fadaei, *Detection, characterization and mitigation of gnss jamming interference using pre-correlation methods*, 2016. doi: 10.11575/PRISM/25598. [Online]. Available: https://prism.ucalgary.ca/server/api/core/bitstreams/bb0542a9-7d27-4c15-9222-4a61c41c25c4/content.

[7] G. X. Gao, M. Sgammini, M. Lu, and N. Kubo, "Protecting gnss receivers from jamming and interference," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1327–1338, Jun. 2016, issn: 1558-2256. doi: 10.1109/JPROC.2016.2525938.

[8] H. Li, D. Borio, and P. Closas, "Dual-domain robust gnss interference mitigation," 2019. doi: 10.33012/2019.16991.

[9] D. Borio and C. Gioia, "Impact of robust interference mitigation on gnss timing," pp. 1–13, Nov. 2020. doi: 10.23919/ENC48637.2020.9317331.

[10] F. Garzia, J. R. van der Merwe, A. Rügamer, S. Urquijo, and W. Felber, "Hddm hardware evaluation for robust interference mitigation," *Sensors*, vol. 20, no. 22, 2020, issn: 1424-8220. doi: 10.3390/s20226492. [Online]. Available: https://www.mdpi.com/1424-8220/20/22/6492.

[11] D. Borio and C. Gioia, "Gnss interference mitigation: A measurement and position domain assessment," *NAVIGATION: Journal of the Institute of Navigation*, vol. 68, no. 1, pp. 93–114, 2021, issn: 0028-1522. doi: 10.1002/navi.391. eprint: https://navi.ion.org/content/68/1/93.full.pdf. [Online]. Available: https://navi.ion.org/content/68/1/93.

[12] J. R. van der Merwe, F. Garzia, A. Rügamer, and W. Felber, "Advanced and versatile signal conditioning for gnss receivers using the high-rate dft-based data manipulator (hddm)," *NAVIGATION: Journal of the Institute of Navigation*, vol. 68, no. 4, pp. 779–797, 2021, ISSN: 0028-1522. DOI: 10.1002/navi.441. eprint: https://navi.ion.org/content/68/4/779.full.pdf. [Online]. Available: https://navi.ion.org/content/68/4/779.

[13] W. Syam, "The difference between c/n_0 and snr in gnss signal processing," 2023. [Online]. Available: https://www.wasyresearch.com/the-difference-between-c-n0-and-snr-in-gnss-signal-processing/.

Implementation references

[14] (2024). "Cross-correlaton," [Online]. Available: https://en.wikipedia.org/wiki/Cross-correlation.

[15] *Ad9361 adc data sheet*, Analog Devices Inc. [Online]. Available: https://opencpi.github.io/releases/1.5.0/assets/AD9361_ADC_Sub.pdf.

[16] *Ad9361 data sheet*, Analog Devices Inc. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ad9361.pdf.

[17] *Ad9361 reference manual*, Analog Devices Inc. [Online]. Available: https://ez.analog.com/cfs-file/__key/telligent-evolution-components-attachments/00-441-00-00-00-07-91-97/AD9361_5F00_Reference_5F00_Manual_5F00_UG_2D00_570.pdf.

[18] *Adrv9361-z7035 user guide - electrical specification*, Analog Devices Inc. [Online]. Available: https://wiki.analog.com/resources/eval/user-guides/adrv9361-z7035/electrical-specifications.

[19] ARM, "Amba axi protocol specification," version 1.0, [Online]. Available: https://documentation-service.arm.com/static/5f915920f86e16515cdc3342?token=.

[20] ARM, "Amba axi-stream protocol specification," version 1.0, [Online]. Available: https://documentation-service.arm.com/static/64819f1516f0f201aa6b963c?token=.

[21] *Building hdl tutorial*, Analog Devices Inc. [Online]. Available: https://wiki.analog.com/resources/fpga/docs/build.

[22] *Hdl reference design repository*, Analog Devices Inc. [Online]. Available: https://github.com/analogdevicesinc/hdl.

[23] *High-speed dma controller peripheral*, Analog Devices Inc. [Online]. Available: https://wiki.analog.com/resources/fpga/docs/axi_dmac.

[24] *Portable radio reference design features*, Analog Devices Inc. [Online]. Available: https://wiki.analog.com/resources/eval/user-guides/pzsdr/carriers/portable-radio-reference-design/features.

[25] Xilinx, *7-series fpgas selectio resources ug471*, AMD, [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug471_7Series_SelectIO.

[26] Xilinx, *Vivado design flows overview ug888*, AMD, [Online]. Available: https://docs.xilinx.com/r/en-US/ug888-vivado-design-flows-overview-tutorial/Vivado-Design-Flows-Overview.

[27] Xilinx, *Vivado design suite tutorial ug871*, AMD, [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug871-vivado-high-level-synthesis-tutorial.

[28] Xilinx, *Vivado design suite tutorial: Creating and packaging custom ip ug1119*, AMD, [Online]. Available: https://docs.xilinx.com/r/en-US/ug1119-vivado-creating-packaging-ip-tutorial.

[29] Xilinx, *Vivado design suite tutorial: Designing ip subsystems using ip integrator ug995*, AMD, [Online]. Available: https://docs.xilinx.com/r/en-US/ug995-vivado-ip-subsystems-tutorial.

[30] Xilinx, *Vivado design suite tutorial: Embedded processor hardware design ug940*, AMD, [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug940-vivado-tutorial-embedded-design.

[31] Xilinx, *Vivado design suite user guide: Embedded processor hardware design ug898*, AMD, [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug898-vivado-embedded-design.

[32] Xilinx, *Vivado design suite user guide: Ip-centric design flow ug896*, AMD, [Online]. Available: https://docs.xilinx.com/r/en-US/ug896-vivado-ip/IP-Centric-Design-Flow.

[33] Xilinx, *Vivado design suite user guide: Logic simulation ug937*, AMD, [Online]. Available: https://docs.xilinx.com/r/en-US/ug937-vivado-design-suite-simulation-tutorial.

[34] Xilinx, *Vivado design suite user guide: Using constraints ug903*, AMD, [Online]. Available: https://docs.xilinx.com/r/en-US/ug903-vivado-using-constraints.

[35] Xilinx, *Vivado design suite: Axi reference guide ug1037*, AMD, [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug1037-vivado-axi-reference-guide.

[36] Xilinx, *Zynq 7000 soc technical reference manual ug585*, AMD, [Online]. Available: https://docs.xilinx.com/r/en-US/ug585-zynq-7000-SoC-TRM/Zynq-7000-SoC-Technical-Reference-Manual.

[37] Xilinx, *Zynq-7000 soc pcb design guide ug933*, AMD, [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug933-Zynq-7000-PCB.

Images references

[38] *Adrv9361-z7035 board overview*, Analog Devices Inc. [Online]. Available: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/ADRV9361-Z7035.html#eb-overview.

[39] *Board connections overview*, Analog Devices Inc. [Online]. Available: https://www.analog.com/en/education/education-library/videos/2753072929001.html.

[40] *Comparison convolution correlation*, Cmglee. [Online]. Available: https://commons.wikimedia.org/wiki/File:Comparison_convolution_correlation.svg.

[41] Enemy, *Phase_modulation_bpsk_gps*. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/f/f8/Phase_modulation_BPSK_GPS.svg.

[42] *Gnss receiver operation overview (gps and galileo receiver)*, what-when-how.com. [Online]. Available: https://what-when-how.com/a-software-defined-gps-and-galileo-receiver/gnss-receiver-operation-overview-gps-and-galileo-receiver/.

[43] P. F. Lammertsma, *Gps_signal_modulation_scheme*. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/3/34/GPS_signal_modulation_scheme.svg.

[44] *Map of gps interference*, gpsjam.org. [Online]. Available: https://gpsjam.org/?lat=5.25389&lon=15.15232&z=2.0&date=2024-01-08&proj=equirectangular.

[45] *Maritime navigation under threat: Gnss spoofing raises security concerns*, www.riskintelligence.eu. [Online]. Available: `https://www.riskintelligence.eu/analyst-briefings/maritime-navigation-under-threat`.

[46] *Zynq-7000 overview*, Analog Devices Inc. [Online]. Available: `https://wiki.analog.com/resources/eval/user-guides/adrv9361-z7035/electrical-specifications`.

Appendices

A AXI4-Stream

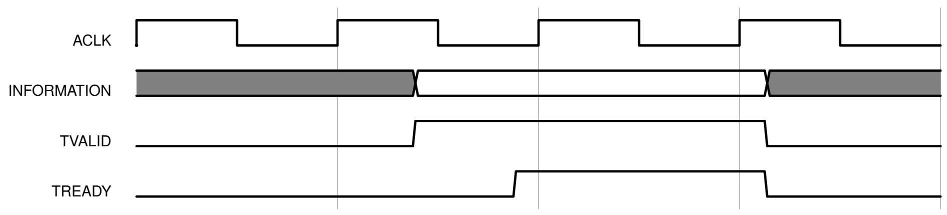
The AXI4-Stream protocol is used as a standard interface to connect components that wish to exchange data. The interface can be used to connect a single master, that generates data to a single slave, that receives data. The protocol can also be used when connecting larger numbers of master and slave components and enables many forms of data streams. The interface signals are listed in table A.

A single transfer of data across an AXI4-Stream interface is defined by a single **TVALID**, **TREADY** handshake. A two-way flow control mechanism enables both the master and the slave to control the rate at which the data and control information is transmitted across the interface. For a transfer to occur both the **TVALID** and **TREADY** signals must be asserted. Either **TVALID** or **TREADY** can be asserted first or both can be asserted in the same **ACLK** cycle.
A master is not permitted to wait until **TREADY** is asserted before asserting **TVALID**. Once **TVALID** is asserted it must remain asserted until the handshake occur.
A slave is permitted to wait for **TVALID** to be asserted before asserting the corresponding **TREADY**.
If a slave asserts **TREADY**, it is permitted to deassert **TREADY** before **TVALID** is asserted.
Some examples of handshake sequence are given in figure H.

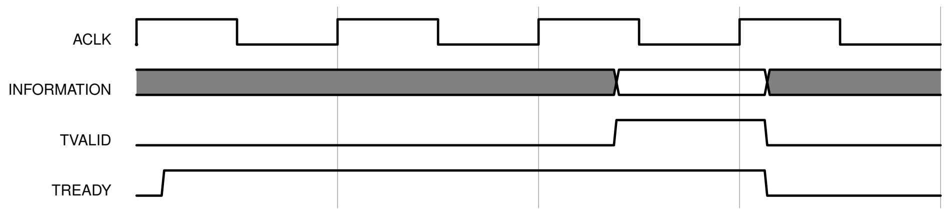
Not all signals are always required by the protocol. **TREADY** signal can be omitted in certain circumstances. However, it is recommended that **TREADY** is always implemented. **TKEEP** and **TSTRB** signals are optional signals and are not required by all types of data stream. For data streams that have no concept of packets or frames also **TLAST** is optional. Furthermore, **TID**, **TDEST** and **TUSER** are also optional.
Most applications of the AXI4-Stream interface will transfer a data payload but it is allowable to implement an interface that does not have a **TDATA** data payload.

Interface signals list		
Signal	**Source**	**Description**
ACLK	Clock source	The global clock signal. All signals are sampled on the rising edge of **ACLK**
ARESETn	Reset source	The global reset signal. **ARESETn** is ACTIVE-LOW
TVALID	Master	**TVALID** indicates that the master is driving a valid transfer. A transfer takes place when both **TVALID** and **TREADY** are asserted.
TREADY	Slave	**TREADY** indicates that the slave can accept a transfer in the current cycle.
TDATA[(8N-1):0]	Master	**TDATA** is the primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
TSTRB[(n-1):0]	Master	**TSTRB** is the byte qualifier that indicates whether the content of the associated byte of **TDATA** is processed as a data byte or a position byte.
TKEEP[(n-1):0]	Master	**TKEEP** is the byte qualifier that indicates whether the content of the associated byte of **TDATA** is processed as part of the data stream. Associated bytes that have the **TKEEP** byte qualifier deasserted are null bytes and can be removed from the data stream.
TLAST	Master	**TLAST** indicates the boundary of a packet
TID[(i-1):0]	Master	**TID** is the data stream identifier that indicates different streams of data.
TDEST[(d-1):0]	Master	**TDEST** provides routing information for the data stream.
TUSER[(u-1):0]	Master	**TUSER** is user defined sideband information that can be transmitted alongside the data stream.

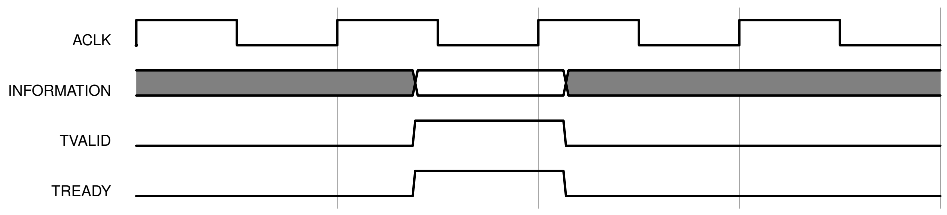
Table B: AXI4-Stream interface signals list



(a) **TVALID** before **TREADY** handshake



(b) **TREADY** before **TVALID** handshake



(c) **TVALID** with **TREADY** handshake

Figure H: Handshake sequence examples AXI4-Stream

B AXI-Lite

The AXI protocol supports high-performance, high-frequency system designs as specified in the Amba AXI Protocol specification [19]. It is burst-based and defines the following independent transaction channels:

- read address
- read data
- write address
- write data
- write response

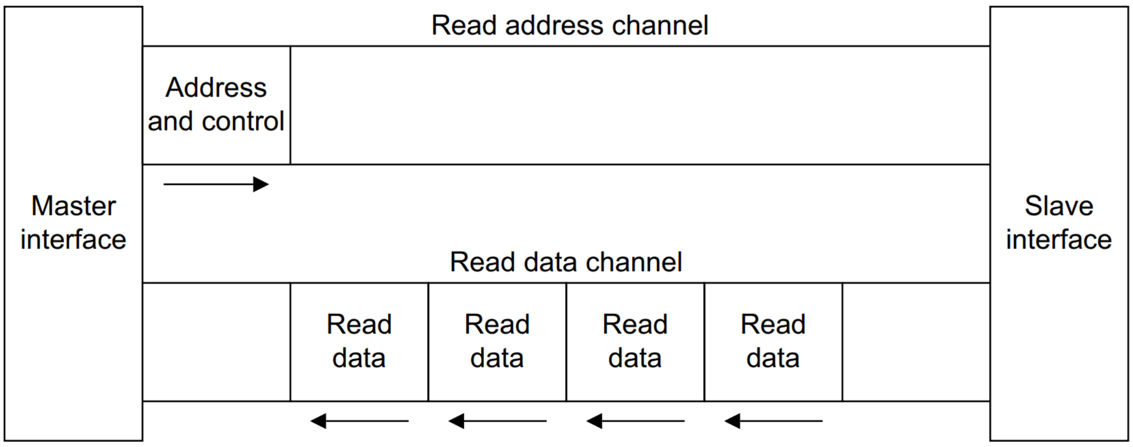
An address channel carries control information that describes the nature of the data to be transferred. The data is transferred between master and slave using either:

- A write data channel to transfer data from the master to the slave. In a write transaction, the slave uses the write response channel to signal the completion of the transfer to the master.
- A read data channel to transfer data from the slave to the master.

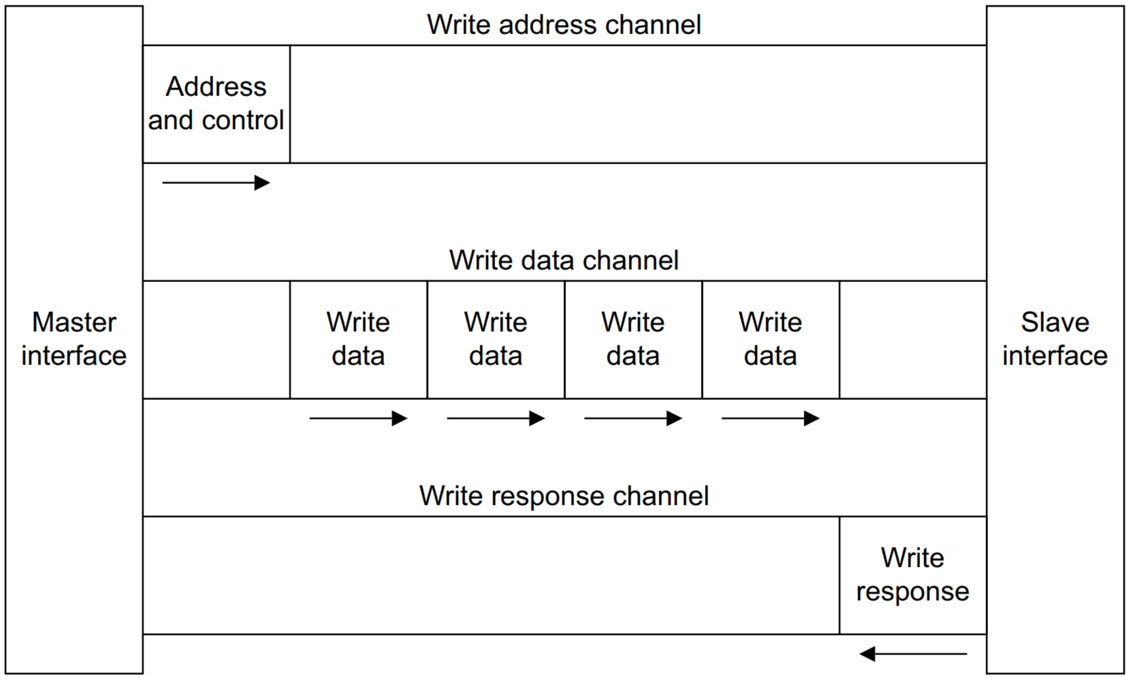
The AXI protocol:

- permits address information to be issued ahead of the actual data transfer
- supports multiple outstanding transactions
- supports out-of-order completion of transactions.

Figure I shows how a read transaction uses the read address and read data channels and how a write transactions uses the write address, write data and write response channels. Each of the independent channels consists of a set of information signals and **VALID** and **READY** signals that provide a two-way handshake mechanism.
The information source uses the **VALID** signal to show when valid address, data or control information is available on the channel. The destination uses the **READY** signal to show when it can accept the information. Both the read data channel and the write data channel also include a **LAST** signal to indicate the transfer of the final data item in a transaction.
Read and write transactions each have their own address channel. The appropriate address channel carries all of the required address and control information for a transaction.



(a) Read transaction



(b) Write transaction

Figure I: AXI read and write architecture

The read data channel carries both the read data and the read response information from the slave to the master, and includes:

- the data bus, that can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide
- a read response signal indicating the completion status of the read transaction.

The write data channel carries the write data from the master to the slave and includes:

- the data bus, that can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide
- a byte lane strobe signal for every eight data bits, indicating which bytes of the data are valid.

Write data channel information is always treated as buffered, so that the master can perform write transactions without slave acknowledgement of previous write transactions.
A slave uses the write response channel to respond to write transactions. All write transactions require completion signaling on the write response channel.

The AXI protocol also includes the AXI4-Lite specifications, a subset of AXI4 for communication with simpler control register style interfaces within components. It has a fixed data bus width and all transactions are the same width as the data bus. The data bus width must be, either 32-bits or 64-bits.
Signals supported and involved in AXI4-Lite specifications are listed in the following table.

Interface signals list					
Global	**Write address channel**	**Write data channel**	**Write response channel**	**Read address channel**	**Read data channel**
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
-	AWADDR	WDATA	BRESP	ARADDR	RDATA
-	AWPROT	WSTRB	-	ARPROT	RRESP

Table C: AXI4-Lite interface signals list

C Cross Ambiguity Function (CAF)

In signal processing, cross-correlation is a measure of similarity of two series as a function of the dispacement of one relative to the other. This is also known as a sliding dot product or sliding inner product. It is commonly used for searching a long signal for a shorter, known feature. It has applications in pattern recognition, single particle analysis, electron tomography, averaging, cryptanalysis, and neurophysiology. The cross-correlation is similar in nature to the convolution of two functions. In an autocorrelation, which is the cross-correlation of a signal with itself, there will always be a peak at a lag of zero, and its size will be the signal energy.
For continuous funcions f and g, the cross-correlation is defined as:

$$(f \star g)(\tau) \triangleq \int_{-\infty}^{\infty} \overline{f(t)} g(t+\tau) dt \tag{1}$$

Where $\overline{f(t)}$ denotes the complex conjugate of $f(t)$ and τ is called displacement or lag.
A figure showing a visual comparison of convolution, cross correlation and autocorrelation follows. For the operations involving function f, and assuming the height of f is 1.0, the value of the result at 5 different points is indicated by the shaded area below each point. Also, the vertical symmetry of f is the reason $f * g$ and $f \star g$ are identical in this example.

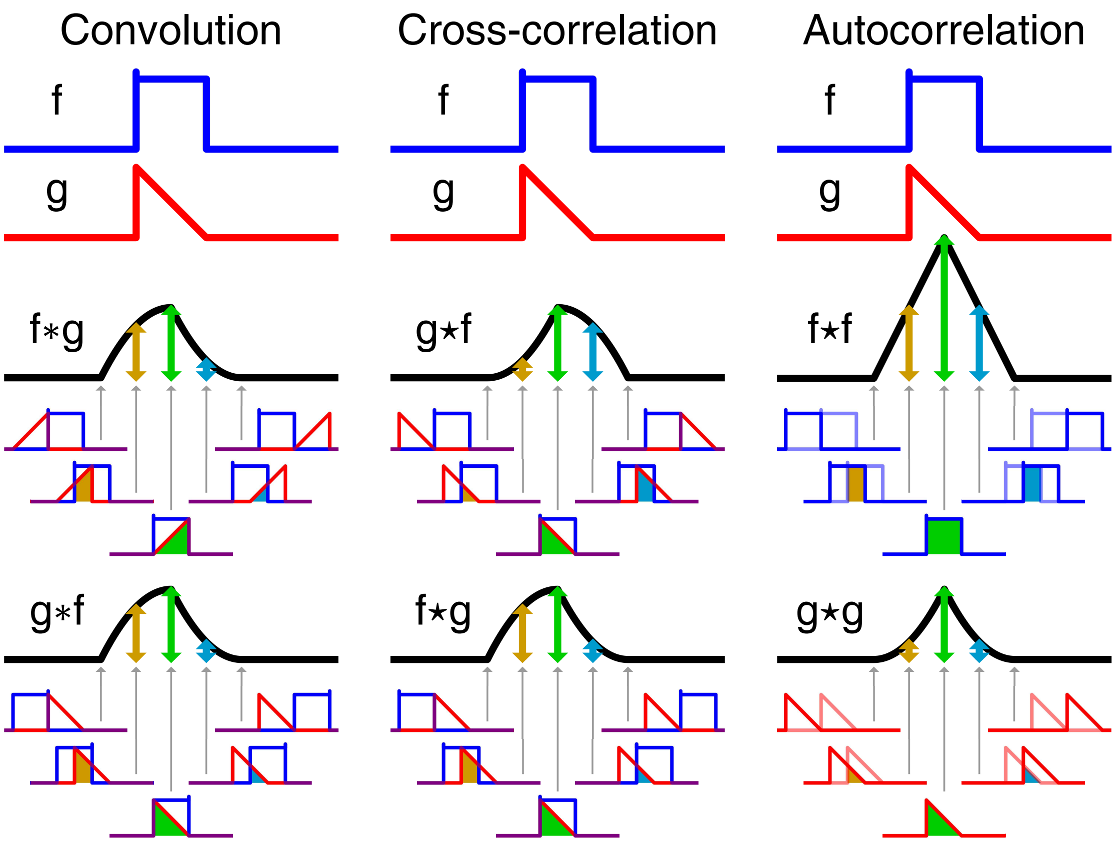


Figure J: Visual comparison of convolution, cross correlation and autocorrelation

D Acquisition function

In this section Matlab code used to calculate normalized CAF on a GNSS signal is reported in order to better describe the acquisition process that have been carried on.

```matlab
function [cafNormal] = acquisition(settings, replicaSignal, inputSignal)

    % Add an aquisition window to calculate also last window caf
    inputSignal = [inputSignal zeros(1,settings.gnss.acqSamples)];

    % Reshape input signal to calculate caf at each column
    acqBuffer = buffer(inputSignal, 2*settings.gnss.acqSamples, settings.gnss.acqSamples,'nodelay');

    % Initialize vectors and variables
    cafNormal = zeros(1, settings.gen.totSamples);

    for i = 1:size(acqBuffer,2)
        % Calculate cross-correlation function
        transformedReplicaSignal = zeros(1, 2*settings.gnss.acqSamples);
        transformedReplicaSignal(1 : settings.gnss.acqSamples) = replicaSignal;
        transformedReplicaSignal = conj(fft(transformedReplicaSignal));
        ccf = ifft(transformedReplicaSignal .* fft(acqBuffer(:,i)'));
        ccf = ccf(1, 1 : settings.gnss.acqSamples);

        % Calculate caf(cross ambiguity function)
        caf = abs(ccf).^2;
        cafAverage = mean(caf);
        cafN = caf ./ cafAverage;

        % Save caf for this acquisition window
        cafNormal(1,(i-1)*settings.gnss.acqSamples+1:i*settings.gnss.acqSamples) = cafN;
    end

    % Eliminate the last acquisition window
    cafNormal = cafNormal(1, 1 : settings.gen.totSamples-settings.gnss.acqSamples);

end
```

No particularly hard procedures are used. A buffer function has been used in order to facilitate signal and replica signal (PRN code) comparison and once CAF have been calculated the normalized one is found by dividing it by its mean value.
The only caution to have is that, since GNSS signal that is simulated is not an infinite stream but only a few milliseconds the last ms that is compared with the PRN code will result in a less big of a peak since further signal samples are missing. This mean that when testing the core, if we want to find the mean peak value in the normalized CAF we need to not consider the last peak.