Technical University of Crete
School of Electrical and Computer Engineering

Implementation of Snapshot-Positioning on a PolarFire FPGA/SoC

Sofia Maragkou

Thesis Committee:
Prof. Apostolos Dollas, Technical University of Crete
Prof. Dionisios Pnevmatikatos, Technical University of Crete
Dr Fabio Garzia, Fraunhofer Institute or Integrated Circuits IIS, Nuremberg

A thesis submitted to the Technical University of Crete in accordance with the requirements for the Diploma in Electrical and Computer Engineering

Work for this thesis was conducted at the Fraunhofer Institute for Integrated Circuits IIS in Nuremberg under the supervision of Dr. Fabio Garzia and Mr. Stefan Egerer.

Περίληψη

Η Υπηρεσία Δημόσιου Χαρακτήρα (Public Regulated Service) του δορυφορικού συστήματος Galileo εγγυάται ασφαλή δορυφορική πλοήγηση για υπηρεσιακή χρήση του δημοσίου, πράγμα που κάνει την υλοποίηση απαιτητική σε θέματα ασφαλείας. Η μέθοδος στιγμιαίου δορυφορικού προσδιορισμού θέσης παρέχει ένα γρήγορο και αποδοτικό τρόπο εντοπισμού θέσης ταχύτητας και χρόνου (Position Velocity Time) με την χρήση βοηθητικών δεδομένων, με δείγματα μη επεξεργασμένα και με κώδικες ψευδοτυχαίου θορύβου. Στόχος αυτής της διπλωματικής είναι η υλοποίηση συστήματος στιγμιαίου δορυφορικού προσδιορισμού θέσης σε αναδιατασσόμενη λογική στο ολοκληρωμένο σύστημα PolarFire για ενσωματωμένο δέκτη σε πραγματικό χρόνο. Η επιλογή του ολοκληρωμένου συστήματος PolarFire έγινε βάσει των εγγυήσεων που παρέχει όσο αφορά την χαμηλή κατανάλωση ενέργειας και την ασφάλεια. Ο υπολογισμός της θέσης, της ταχύτητας και του χρόνου γίνεται σε RISC-V πρότυπο με χρήση της RV32IMA αρχιτεκτονικής σετ εντολών σε ενσωματωμένο soft core. Η διπλωματική εργασία αποτελείται από τρία μέρη: την σχεδίαση σε υλικό που υλοποιεί την μέθοδο απόκτησης σήματος, την εφαρμογή που ελέγχει την σχεδίαση υλικού και την εφαρμογή που υλοποιεί τον στιγμιαίο προσδιορισμό θέσης. Η σχεδίαση αποτελείται από τον soft core Mi-V της Microsemi και από ένα μέρος ελεύθερα διαμορφωμένης και προγραμματιζόμενης λογικής που ελέγχεται από μια εφαρμογή bare-metal σε γλώσσα C++. Η εφαρμογή στιγμιαίου προσδιορισμού έχει υλοποιηθεί σε γλώσσα C.

Abstract

Galileo Public Regulated Service (PRS) is a special navigation service which guarantees secure EU satellite navigation for government use and thus it is high-secure demanding. A fast and efficient way to obtain position, velocity and time (PVT) is the method of snapshot positioning by having assistance data and raw data samples together with the pseudo-random noise (PRN) codes. The purpose of this thesis is to implement snapshot positioning on PolarFire FPGA for an embedded real-time receiver. The selection of PolarFire FPGA is based on the low-energy consumption and the security attributes it is offering. The complete PVT calculation is performed in the RISC-V standard RV32IMA instruction set architecture (ISA) embedded soft processor. The thesis consists of three parts the hardware design which implements the acquisition, the control of acquisition and the snapshot positioning application. The hardware design consists of the Mi-V soft core of Microsemi as CPU and a freely configurable and programmable logic part. It is controlled by a bare-metal application in C++. The snapshot positioning application is implemented in C.

Acknowledgements

First of all I would like to express my gratitude to Prof. Panagiotis Partsinevelos. Without his help and inspiration none of the following things would be now reality. I would like to express my deepest appreciation to Alexander Rügamer who took me into his team and gave me the opportunity to work in the facilities of Fraunhofer and to collaborate with all these people. My sincere gratitude to my supervisors from the Fraunhofer Institute Dr Fabio Garzia and Stefan Egerer for their continuous guidance, for their support and patience through all these months of our collaboration and for being available whenever I needed them. I would also like to express my deepest appreciation to the official supervisor of this thesis Prof. Apostolos Dollas for his useful advice and his support but most importantly for the motivation he gave me during this thesis and for my next steps. Also I want to thank Prof. Dionisios Pneumatikatos for serving in my thesis committee. Furthermore I want to thank Inigo Cortes for his help and support. Last but not least, I want to thank all my amazing friends for being next to me all these years. Finally, I want to thank my parents Christoforos and Kalliopi and my sister Anna for helping me selflessly to make my dream come true and supporting my decisions. Thank you all.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Global Navigation Satellite System (GNSS) is the standard generic term for satellite navigation systems that provide autonomous geo-spatial positioning with global coverage. The term is currently used for Global Positioning System (GPS), Galileo, GLONASS and BeiDou systems. The European GNSS Galileo provides three different global navigation services: Open Service (OS),Commercial Service (CS) and PRS.

Galileo PRS is a special navigation service intended for governmental authorized users. It uses strong encryption which guarantees anti-spoofing. The difference between the Galileo PRS and the GPS Precise Positioning Service (PPS), is that the PRS is not only for military users but also for authorized civil users. PRS is primarily intended for use by EU member state government authorized users and each PRS member state decides on its own who is allowed to use the specific service:

- Fire brigades, police, coastguard, border control, health services, humanitarian aid, search and rescue, customs, civil protection units
- Military users
- Critical infrastructure

The properties of the Galileo PRS are:

- Higher robustness
- Higher accuracy
- Better continuity of service

- Anti-Spoofing
- Authentication of signal
- Access control mechanisms

The purpose of this thesis is to implement snapshot positioning on PolarFire FPGA. This implementation is based on the *PRS remote processing server*, as it will be described later on, but targets an embedded real-time PRS receiver. The selection of the specific technology is based on the security features it supports and the low energy consumption that it has. For the snapshot positioning no tracking and data decoding is required. The snapshot PVT can be calculated by performing an acquisition on the digital signal and getting the ephemeris from assisted data. Hence, this method enables a fast and low-power way to calculate PVT. The thesis can be divided into three parts, as can be shown in figure 1.1: the hardware design, which implements the acquisition method, the software that controls the hardware design and the snapshot positioning application. Since the hardware design was already implemented in different technology, important part of this thesis was the hardware system integration in the technology of Microsemi.

More details will be provided in the next chapters.

1.2 Thesis Outline

This thesis consists of five chapters that follow this introduction:

- Chapter 2, *Theoretical background*: contains details on the algorithms and the equations used.
- Chapter 3, *Snapshot Positioning: Related Work and System Modeling*: explains the concept of the snapshot positioning and the differences between a conventional receiver and a snapshot receiver. Furthermore it shows the system modeling used for the proof of concept for this thesis and also refers to implementations close to the one done for this thesis.
- Chapter 4, *Implementation*: describes the software and the hardware parts of the implementation.
- Chapter 5, *Results*: presents the test setup used for the debugging, the resource utilization, the limitations of the design and the results of the software and the hardware implementations.

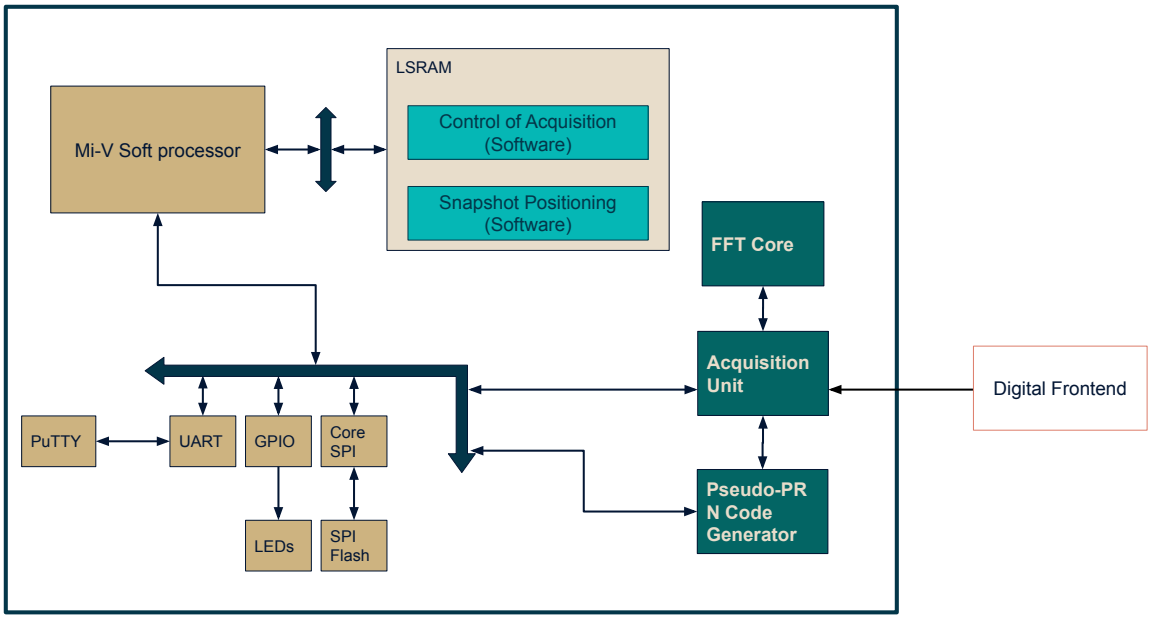

Figure 1.1: Target architecture of the theis work

- Chapter 6, *Conclusion*: provides a summary of the presented work and the future work.

Chapter 2

Theoretical Background

2.1 GNSS Receiver Operation Overview

A GNSS receiver is the user interface to any GNSS. It processes the signals in space (SIS) transmitted by the satellites. A GNSS receiver can be used in a wide range of applications and most of them rely on the receiver navigation solution which is the computation of the PVT. The calculation of the PVT is based on the distance between the receiver and a group of satellites. Nevertheless, the satellites are always in motion and this creates a shift of the expected frequency which is known as the Doppler shift and it is caused by the Doppler effect. With the Doppler shift, the divergence of the frequency can vary from ±6 kHz up to ±10 kHz depending on the dynamic conditions. Since all satellites transmit their signals in the same frequency band, code division multiple access (CDMA) is used to identify each satellite. A PRN code sequence is used to spread the spectrum of the satellite signal. In order to detect a satellite signal the receiver correlates the incoming signal with a locally generated PRN sequence. If there is a peak in the resulting spectrum then the related satellite is visible.

With the objective to determine a position, a conventional GNSS receiver has first to find a rough estimation of the correct frequency and the correct code delay of all visible satellites. This process described is the so called signal acquisition. After acquiring the signals, a GNSS receiver goes into tracking. Only in tracking the receiver can extract the navigation message and decode it. The ephemeris data are part of the navigation message and consist of information about satellite location, timing, and *health*.

As an example, we can consider a GPS signal. The GPS signal can be modeled by the following formula:

$$r[n] = \sqrt{A}d[n]c[n-\tau]cos[2\pi(f_{IF} + f_d)nT_s - \phi] + N \tag{2.1}$$

where A is the carrier power, $d[n]$ is the navigation data , $c[n]$ is the C/A code, f_{IF} denote the Intermediate Frequency (IF) f_d the Doppler shift in Hertz, $T_s = 1/F_s$ the sampling period in seconds, F_s is the sampling frequency in Hertz, ϕ is the initial carrier phase, τ is the initial code delay and N is the additive white Gaussian noise.

And the locally generated carrier signal is expressed by

$$y_c[n] = c[n - \tau] \exp[j2\pi(f_{IF} + f_d)nT_s] \tag{2.2}$$

where, y_c is the locally generated carrier signal at an instant 'n', $c[n]$ represents the C/A code of one GPS satellite, τ is the delay, f_d is the Doppler shift in carrier frequency,f_{IF} is the incoming digitized GPS signal IF frequency, $T_s = 1/F_s$ is the sampling period (s),F_s sampling frequency.

2.2 Acquisition

Acquisition is the first operation performed on the digitized IF signal. The purpose of this process is to identify the satellites visible to the receiver and provide a measurement of the Doppler shift at the frequency of the carrier as well as the the delay at the PRN code. At the acquisition process two things are required, a replica of the PRN code and a replica of the carrier signal.

2.2.1 Serial Search

The serial search acquisition method is mostly referred for its historical importance and has since been only used for receivers focusing on low-resource implementation.

As shown in figure 2.1, the incoming signal IF is multiplied by the locally generated PRN code. The code length is varying from signal to signal, for instance at GPS L1 the code phase varying from 0 to 1022.

After that, the output of the multiplication is correlated with the locally generated carrier signal and also with its 90° phase-shifted version. This way we obtain an in-phase signal I and the quadrature signal Q, which can be used to detect how well the incoming signal is matching with the local carrier phase.

The two output signals I and Q are integrated over an epoch before being squared and summed. The final output, which is a value of correlation between the incoming signal and the locally generated signal, is then compared with a predefined threshold. In case it is exceeded, the frequency and the code phase

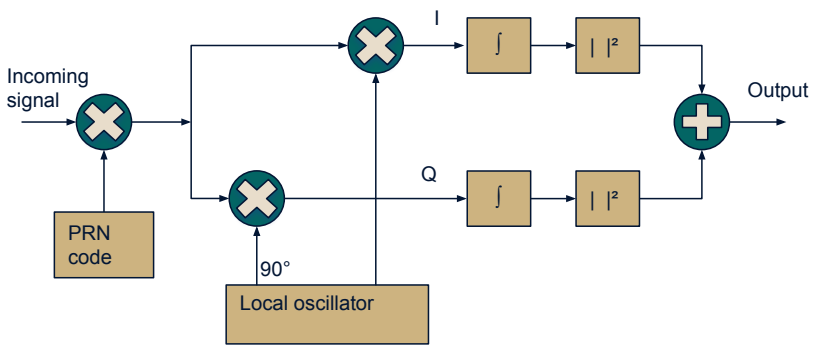

Figure 2.1: Serial search Acquisition block diagram

parameters used are correct. In case it is below the predefined threshold, the replica of the PRN code is shifted by one chip and the process is repeated for the whole code domain i.e. all 1022 chips. If the acquisition is not successful the same process is repeated with a different intermediate frequency.

2.2.2 Parallel Code-Phase Search

Searching through all possible frequency and code phase values is very time consuming. Nevertheless, eliminating one of the two parameters can be very efficient. For instance according to the GPS L1 signal, the PRN code consists of 1023 chips resulting in 1023 possible code phases. If Doppler shift range is ±5 kHz and the search-step is 500 Hz the possible frequencies for searching are 21. In case the search-step is changed to 250 Hz then the possible searching frequencies are 41. So the possible code chips are in a bigger order of magnitude than the possible frequencies. Hence the acquisition should be parallelized in the code phase dimension. In this way the steps that would be performed are equal to the number of the possible Doppler frequencies. Parallel code-phase search converts the incoming signal from time domain to frequency domain and by that way it eliminates one parameter.

As shown in the 2.2, the incoming IF signal is multiplied by a locally generated carrier signal creating signal I and by its 90 ° phase shifted version creating signal Q. The I and Q signals are combined and used as input for the FFT function. The PRN locally generated code is also transformed from the time domain to the

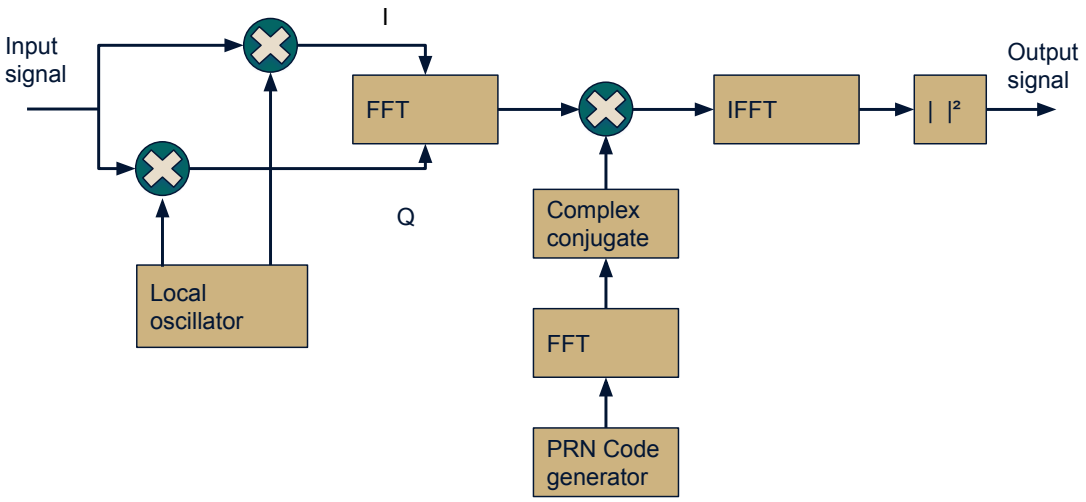

Figure 2.2: Parallel Code Phase Search Acquisition

frequency domain and the result of this transformation is conjugated. The two signals in frequency domain are then multiplied and the result is transformed back to the time domain with an inverse Fourier transform. The highest absolute value of correlation power gives the required Doppler Shift and code phase of the GNSS signal.

The correlation operation between the carrier signal replica and the digitized input GNSS IF signal is described by the equation:

$$z(n) = \sum_{m=0}^{N-1} y_c(m) y_{IF}(m+n) \tag{2.3}$$

Where m is the index of the sampling time sequence and N is the number of samples.

The circular correlation is defined in Fast Fourier Transform (FFT) as:

$$FFT[z(n)] = FFT\left[\sum_{m=0}^{N-1} y_c(m) y_{IF}(m+n)\right] = FFT[y_c] FFT^*[y_{IF}(n)] \tag{2.4}$$

Where FFT^* is complex conjugate of FFT.

So the above equation in frequency domain can defined as:

$$Z(K) = Y_c(K)Y_{IF}^*(K) \tag{2.5}$$

Correlation function in time domain can be written as:

$$z(n) = IFFT(Z(K)) = IFFT\left(Y_c(K)Y_{IF}^*(K)\right) = IFFT\left(FFT\left[y_c(n)\right]FFT^*\left[y_{IF}(n)\right]\right) \tag{2.6}$$

There is a trade-off between the serial search method and the parallel code-phase search method. The parallel code-phase search method utilizes FFT operations which are more expensive, computationally, than the simple correlation in time domain which are used in serial search.

An alter way to perform the Doppler search, includes the shifting of the FFT of the PRN codes and the computation of the IFFT from the beginning. This way the repetition of the input data and PRN FFT in each iteration is avoided since the same one is always used and just shifted by one sample every time. This method is not a parallel Doppler search.

2.3 Carrier and Code Tracking

This thesis does not deal with either carrier or code tracking, however, they are briefly described here since they are integral part of the conventional receivers. This is the difference between a conventional receiver operation and the snapshot PVT. In snapshot PVT no tracking and message decoding is required, and the receiver acquires the ephemeris data over another channel. After the signal acquisition the conventional receiver tracks the signal by following the changes in carrier and code Doppler shift as described bellow.

2.3.1 Carrier Tracking

After the receiver finds the carrier Doppler frequency of the satellite, it keeps track of the carriers Doppler frequency as shown in figure 2.3.

The most important functions of the carrier tracking loop are the Carrier Loop Discriminator (CLD) and the the carrier loop filter (CLF). The type of CLD defines also the type of the tracking loop which can be either phase lock loop (PLL) or frequency lock loop (FLL). As evidenced by the name, the discriminator of PLL type estimates the phase error and the discriminator of FLL type estimates the

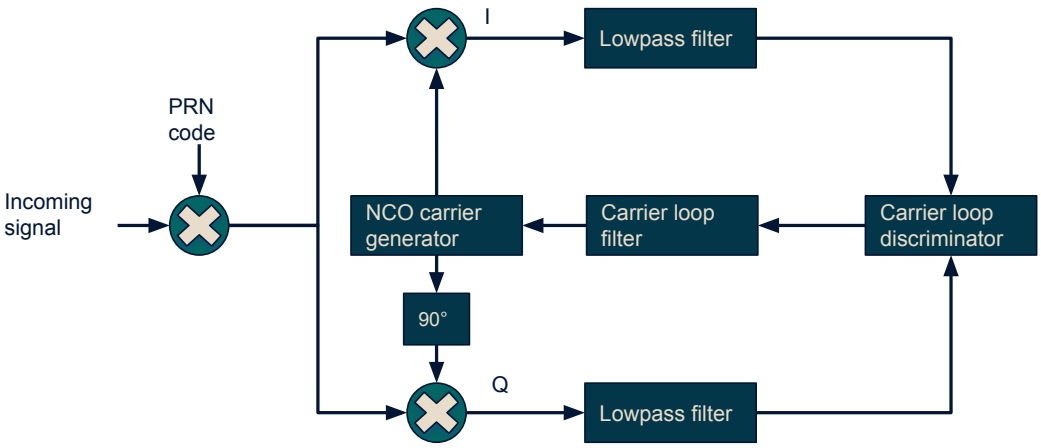

Figure 2.3: Diagram of Phase Lock Loop

frequency error. The output of the discriminator is filtered and used as feedback to the Numerically Controlled Oscillator (NCO), which adjusts the frequency of the local carrier replica. PLLs tend to be more accurate while being more sensitive towards dynamic stress than FLLs.

2.3.2 Code Tracking

In the code tracking loop, the difference between the locally generated PRN code and the PRN code of the signal is tracked and minimized. As shown in the figure 2.4, the code tracking loop consists of three correlators named Prompt, Early and Late for present, delayed and advanced time respectively.

In this case, the most important parts of the module are the code loop discriminator and the code loop filter. The main purpose of the process is to regulate the locally generated PRN code and the PRN of the received signal to have zero phase difference. By doing so the receiver keeps the code component of the signal in track. Each satellite channel needs its own tracking loop.

2.4 Data processing for Positioning

2.4.1 Navigation Data

According to [3], navigation data contains all the information needed for calculating a position. The navigation data includes the ephemeris parameters, in order to

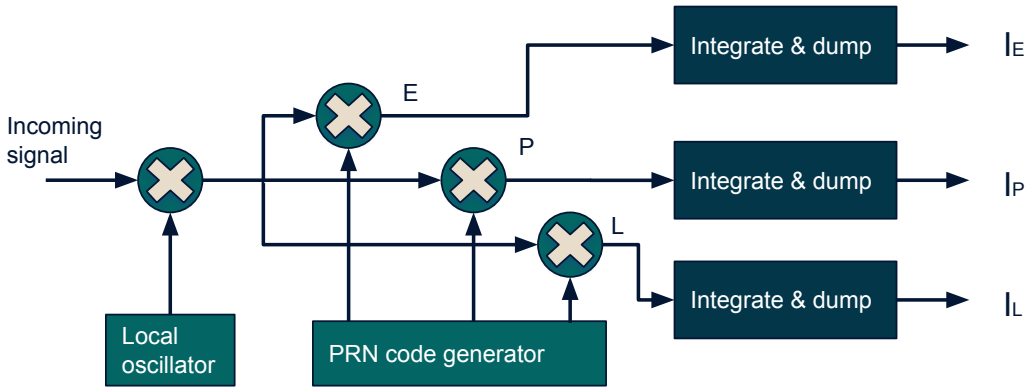

Figure 2.4: Basic code tracking loop block diagram

estimate the satellite position, the time parameters, the clock corrections, service parameters, which indicate the satellite health information, and the almanac information. With the use of the time parameters and the clock corrections the receiver estimates the satellite clock offset and the time conversions.

The navigation data message is represented by binary units which are called bits or symbols. The binary units are called symbols in the case of forward error correction (FEC) otherwise they are referred as bits. One bit or symbol can have more than one primary code period but always an integer number. Also a secondary code sequence can be used. The secondary code sequence alters the sign of primary code period within one bit or symbol. The length of it is equal to the duration of the bit or symbol. The bit or symbol synchronization is also called secondary code synchronization. Once the synchronization is achieved, the secondary code can be removed from the correlation values so the values can later be added up. By this process the integration time gets increased which is good for the tracking process as this increases the sensitivity and the accuracy.

After the process of the bit or symbol synchronization the navigation data message can finally be separated from the received signal. The purpose of the data bit or symbol demodulation is to retrieve the navigation data message. Depending on the tracking phase which has been achieved, there are different scenarios for this process.

The next step is the frame synchronization. The different kinds of navigation messages have their different structure. Though, it is common that all the data is organized in blocks called frames, pages, subframes or in a similar way. The

beginning of such a block is called preamble and it consists of a bit or symbol sequence and it is different for each kind of navigation data. The decoders have the task to recognize this preamble. If demodulation fails to determine the absolute sign of the navigation data, two scenarios have to be taken into consideration for the preamble. The preamble search process has to check for the normal and the inverted preamble. If the preamble is detected, a specific number of bits or symbols that represent the navigation data message block is passed to the decoder. In case of the inverted preamble, the decoder has also to invert the data block. If the block of data is validated, for example by parity checks, the frame synchronization has been achieved. Normally this takes several tries since the preamble is short and part of the data can be falsely interpreted as preamble.

Afterwards the bit error correction takes place. There are many techniques to verify that the bits have been received correctly. Examples for these techniques are parity bits, forward error correction and interleaving. The decoder not only detects errors at the bits or symbols but also corrects them. Since the navigation message changes every few hours, the decoding sensitivity of the receiver can be increased by storing instances of the received data stream. Moreover, it is possible to predict future messages if messages have been received completely.

Finally, the data content can be read. The data consists of bit or symbol fields, according to the kind of the navigation message it belongs to. Every field represents a signed or an unsigned integer number which is converted to a floating point number. These floating point numbers are the data used for the positioning as it is described in the next sections.

2.4.2 Computation of Satellite Position

As it is described in [2], in order to represent the state of both satellite and receiver, a reference coordinate system should be selected. Satellite and receiver are both described by their position and their velocity vectors measured in a Cartesian coordinate system. There are two inertial and rotating Cartesian coordinate systems, the Earth-centered intertial (ECI) system and the Earth-centered and Earth-fixed (ECEF) system. In this thesis the ECEF coordinate system is used, and for that reason an overview is provided.

The ECEF coordinate system is rotating with the Earth. The xy-plane coincident with Earth's equatorial plane, the +x-axis points in the direction of 0° longitude and the +y-axis points in the direction of 90°E longitude. The z-axis is normal to the equatorial plane in the direction of the geographical North Pole. As a result, x-,y-

Keplerian element	Explanation
α	semimajor axis of the ellipse
e	eccentricity of the ellipse
τ	time of perigee passage
i	inclination of orbit
Ω	longitude of the ascending node
ω	argument of perigee

Table 2.1: Keplerian orbital elements

and z-axes rotate with the Earth completing the right-handed coordinate system. The satellite position and the receiver position are computed in the ECEF system and it is typical to transform these Cartesian coordinates to latitude, longitude, and height of the receiver.

To be able to fully describe a satellite position it is important to understand how their orbits are characterized. According to Kepler the following three laws apply to satellites orbiting the earth:

- The orbit of a satellite is an ellipse with the Earth located in one of its foci.
- The radius vector covers a constant area per unit time interval (law of areas).
- The square of the orbital period increases with the third power of the mean distance from the center of the Earth.

Even though there are more forces impacting a satellite's orbit except the Earth gravitation, like the so-called third-body gravitation from the Sun and the Moon, we will take into consideration the Keplerian satellite motion in which the only force acting on the satellite is the point-mass Earth.

There are many possible formulations of the solution to the two-body problem, but the classical solution uses a particular set of six integrals of motion known as the Keplerian orbital elements as shown in table 2.1. The Keplerian orbital elements α,e and τ define the shape of the orbit and give also some time information. The ellipse has semimajor axis α and eccentricity e. As can been seen in figure 2.5, the F point is the center of the mass of the Earth, and as a result the origin of ECEF coordinate system.

Epoch is the time t_0 at which the satellite is at a reference point A. P is the point on the satellite orbit where is closest to the Earth and it is known as *perigee* and the time when satellites passes from *perigee* is mentioned as τ.

The angle in the orbital plane from the perigee to the satellite is called *true anomaly* and it is referred as ν. True anomaly does not vary linearly in time for

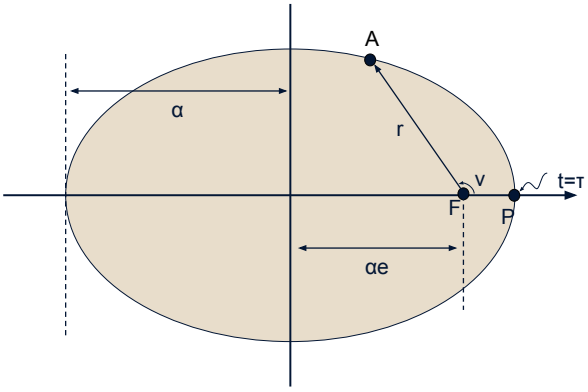

Figure 2.5: Keplerian orbital elements α,e,τ

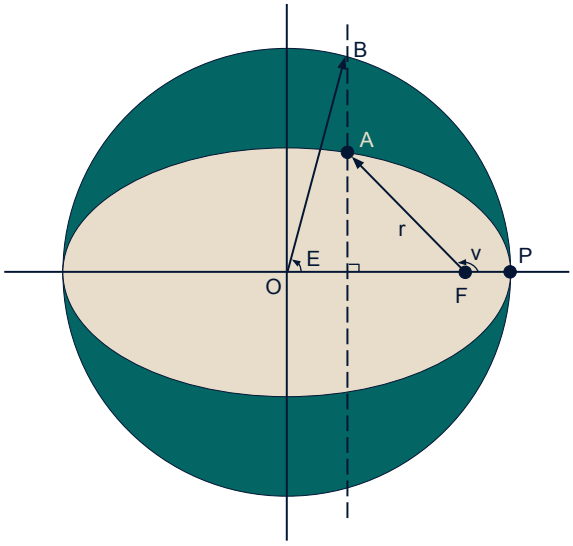

Figure 2.6: Eccentric and true anomaly

noncircular orbits and for that reason two definitions are made to transform true anomaly to mean anomaly, which is linear in time.

The Keplerian orbital elements referred so far define the shape of the elliptical orbit and time relative to perigee. The rest three elements as shown in table 2.1, define the orientation of the orbit in the ECEF coordinate system. The angle between the Earth's equatorial and the satellite orbital plane is the inclination i. The argument of perigee ω and the longitude of the ascending node Ω are defined in relation to the ascending node. The ascending node is the point in the satellite orbit where it crosses the equatorial plane with +z velocity, which implies the direction from the southern to the northern hemisphere. The orbital element that defines the angle between the +x-axis and the direction of the ascending node is called the right ascension of the ascending node. The +x-axis is fixed in the direction of the prime meridian, 0° longitude, in the ECEF coordinate system. As a result, the right ascension of the ascending node (RAAN) is actually the longitude of the ascending node, Ω. The last Keplerian orbital element, the argument of perigee, ω is the angle from the ascending node to the direction of perigee in the orbit. The difference between the longitude of the ascending node and the argument of perigee, is that the Ω is relative to the equatorial plane, but ω is relative to the orbital plane. The orbital parameters differ on a way that the constellation provides coverage of the entire Earth.

When the Keplerian orbital elements of a satellite can be calculated by the true position and the velocity vector at a specific time, they are called *osculating*

orbital elements. However, the osculating orbital elements will change over time because of the perturbing accelerations oppose on the satellite. The Keplerian orbital elements are part of the ephemeris data, with the exception of the time of perigee passage which is converted to mean anomaly at *epoch*. For that reason in the ephemeris the reference time is included, at which the orbital elements are valid. It is called *time of ephemeris* or *time of epoch*. At later times, the true orbital elements are slightly different from the osculating values. The requirement of the ephemeris message to be always very accurate, is covered by the inclusion of the *correction* parameters in the ephemeris data. These corrections provide to the receiver enough information to estimate the Keplerian elements in between updates of the ephemeris message. The ephemeris data with the respectively definitions is shown at table 2.2.

Name of argument	**Explanation**
t_{0_e}	Reference time of ephemeris
$\sqrt{\alpha}$	Square root of semimajor axis
e	Eccentricity
i_0	Inclination angle (at time t_{0_e})
Ω_0	Longitude of the ascending node (at weekly epoch)
ω	Argument of perigee (at time t_{0_e})
M_0	Mean anomaly (at time t_{0_e})
di/dt	Rate of change of inclination angle
$\dot{\Omega}$	Rate of change of longitude of the ascending node
Δn	Mean motion correction
C_{uc}	Amplitude of cosine correction to argument of latitude
C_{us}	Amplitude of sine correction to argument of latitude
C_{rc}	Amplitude of cosine correction to orbital radius
C_{rs}	Amplitude of sine correction to orbital radius
C_{ic}	Amplitude of cosine correction to inclination angle
C_{is}	Amplitude of sine correction to inclination angle

Table 2.2: Table of the ephemeris data

According to the ephemeris data in the table 2.2 and the theory explained above, the position of one satellite can be found by the steps described bellow.

The radius, the inclination and the longitude of node can be estimated by using the following formulas:

Corrected argument of latitude:

$$u_k = \phi_k + \delta_{\phi_k} \tag{2.7}$$

Corrected radius:

$$r_k = \alpha(1 - e\cos E_k) + \delta_{r_k} \tag{2.8}$$

Corrected inclination:

$$i_k = i_o + (d_i/d_t)t_k + \delta_{i_k} \tag{2.9}$$

Corrected longitude of node:

$$\Omega_k = \Omega_o + \left(\dot{\Omega} - \dot{\Omega}_e\right)(t_k) - \dot{\Omega}_e t_{o_e} \tag{2.10}$$

Where $\dot{\Omega}_e$ is the rotation rate of the Earth.

Furthermore, the in-plane x and the in-plane y position can be computed like the equations:

$$x_p = r_k \cos u_k \tag{2.11}$$

$$y_p = r_k \sin u_k \tag{2.12}$$

Finally the x,y and z ECEF coordinates can be estimated.

$$x_s = x_p \cos \Omega_k - y_p \cos i_k \sin \Omega_k \tag{2.13}$$

$$y_s = x_p \sin \Omega_k + y_p \cos i_k \cos \Omega_k \tag{2.14}$$

$$z_s = y_p \sin i_k \tag{2.15}$$

More details on computation of a satellite ECEF position vector can be found at [2] and the whole sequence of formulas used can be found at the appendix B.

2.4.3 Pseudorange Estimation

A problem that needs to be solved, before calculating the receiver position is the determination of satellite-to-user range with nonsynchronized clocks and PRN codes. As is shown in figure 2.7 the r is the vector offset from the user to the satellite, the s represents the position of the satellite relative to the coordinate origin, and u is the receiver position vector with respect to the coordinate system. The receiver's position coordinates are unknown. The vector r can be written as:

$$r = s - u \tag{2.16}$$

Sequentially, the magnitude of the vector **r** is

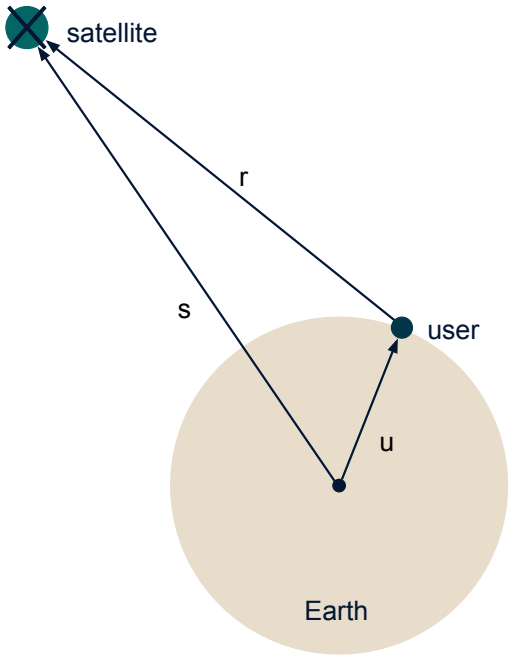

Figure 2.7: Receiver and satellite position vectors

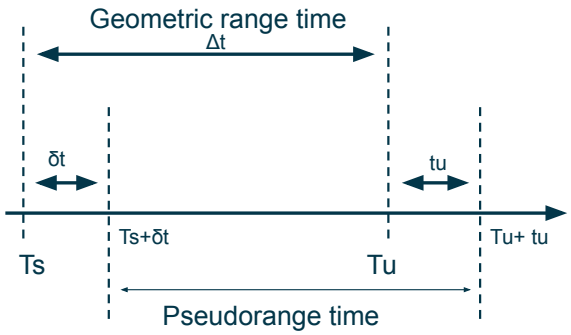

Figure 2.8: Timing relationships

$$\|r\| = \|s - u\| \tag{2.17}$$

In order to calculate the distance r, the measurement of the propagation time is required. As illustrated in the figure 2.8 at the time T_s the signal left the satellite, and at the time T_u the signal reaches the receiver. The propagation time is Δt. At the receiver a replica of the code is generated locally in time t and it is shifted in time until it achieves a correlation with the received code. If the clock of the receiver and the clock of the satellite were synchronized the correlation process would yield the true propagation time. By multiplying this propagation time by the speed of light the real distance between the satellite and the receiver would be resulting. But in reality that is not what is happening. The clock of the satellite has a deviation of the system time and the satellite clock has also an offset from the system clock. As a result, the range determine of the correlation is called *pseudorange* ρ. More specifically, a *pseudorange* is determined by multiplying the signal propagation velocity, which is equals to c, by the time difference of the two non-synchronized clocks. The geometric range can be expressed by the equation:

$$r = c\left(T_u - T_s\right) = c\Delta t \tag{2.18}$$

and the *pseudorange* can calculated like:

$$\rho = c\Big[\big(T_u + t_u\big) - \big(T_s + \delta t\big)\Big] = c\big(T_u - T_s\big) + c\big(t_u - \delta t\big) = r + c\big(t_u - \delta t\big) \quad (2.19)$$

Where δt is the offset of the satellite clock from the system time. When value is positive there is advance and when it is negative there is a delay. The t_u argument is the receiver clock offset from the system clock. $T_s + \delta t$ is the satellite clock reading at the time that the signal left the satellite and $T_u + t_u$ is the receiver clock reading at the time the signal reached the receiver. Finally c is the speed of light. Taking into consideration the formula 2.17 and 2.19 this equation can be written:

$$\rho - c\big(t_u - \delta t\big) = \|\mathbf{s} - \mathbf{u}\| \quad (2.20)$$

The δt consists of bias and drift contributions. The ground monitoring network calculates the corrections of the offset time and transmits the corrections to the satellites. The satellites, broadcast to the receivers the corrections through navigation message. These corrections are applied in the receiver in order to synchronize the transmission of each ranging signal to system time. By this process the δt is consider no longer unknown. Finally the equation 2.20 can expressed as:

$$\rho - ct_u = \|\mathbf{s} - \mathbf{u}\| \quad (2.21)$$

2.4.4 Computation of Receiver Position

In order to determine the receiver position the three dimensions x_u,y_u and z_u and the offset t_u, should be determined. To determine a receiver position, four satellites are needed resulting in the system of equations :

$$\rho_i = \|s_i - u\| + ct_u \quad (2.22)$$

Where i is in range from 0 to 4 implying the number of the satellite. The formula 2.22 for each satellite can be expressed as:

$$
\begin{aligned}
\rho_1 &= \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} + ct_u \\
\rho_2 &= \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} + ct_u \\
\rho_3 &= \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} + ct_u \\
\rho_4 &= \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} + ct_u
\end{aligned}
\tag{2.23}
$$

If we assume that an approximation of the receiver position is known , then, the true position (x_u, y_u, z_u) of the receiver can be expressed as act of the approximated position $(\widehat{x_u}, \widehat{y_u}, \widehat{z_u})$ and a displacement $(\Delta x_u, \Delta y_u, \Delta z_u)$. By writing the equations 2.23 in Taylor series about the approximate position, the position offset $(\Delta x_u, \Delta y_u, \Delta z_u)$ can be obtained as linear functions of the known coordinates and pseudorange measurements.

$$
\begin{aligned}
x_u &= \widehat{x_u} + \Delta x_u \\
y_u &= \widehat{y_u} + \Delta y_u \\
z_u &= \widehat{z_u} + \Delta z_u \\
t_u &= \widehat{t_u} + \Delta t_u
\end{aligned}
\tag{2.24}
$$

According to the formulas 2.23, the pseudorange of a single satellite can be expresses as:

$$
\rho_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} + ct_u = f\left(x_u, y_u, z_u, t_u\right) \tag{2.25}
$$

By the use of the approximate position $(\widehat{x_u}, \widehat{y_u}, \widehat{z_u})$ and the time estimation $\widehat{t_u}$ an approximate pseudorange can be calculated:

$$
\widehat{\rho_i} = \sqrt{(x_i - \widehat{x_u})^2 + (y_i - \widehat{y_u})^2 + (z_i - \widehat{z_u})^2} + c\widehat{t_u} = f\left(\widehat{x_u}, \widehat{y_u}, \widehat{z_u}, \widehat{t_u}\right) \tag{2.26}
$$

Sequentially, the equations 2.24 and the equation 2.26, can give the following relation:

$$
f\left(\widehat{x_u}, \widehat{y_u}, \widehat{z_u}, \widehat{t_u}\right) = f\left(\widehat{x_u} + \Delta x_u, \widehat{y_u} + \Delta y_u, \widehat{z_u} + \Delta z_u, \widehat{t_u} + \Delta t_u\right) \tag{2.27}
$$

The above function can be expanded using a Taylor series by this way:

$$
\begin{aligned}
f\left(\widehat{x}_u + \Delta x_u, \widehat{y}_u + \Delta y_u, \widehat{z}_u + \Delta z_u, \widehat{t}_u + \Delta t_u\right) = f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right) + \\
\frac{\partial f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right)}{\partial \widehat{x}_u}\Delta x_u + \frac{\partial f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right)}{\partial \widehat{y}_u}\Delta y_u + \\
\frac{\partial f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right)}{\partial \widehat{z}_u}\Delta z_u + \frac{\partial f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right)}{\partial \widehat{t}_u}\Delta t_u + \ldots
\end{aligned}
\tag{2.28}
$$

After the first-order partial derivatives to eliminate nonlinear terms, the expansion has been shortened. The partials derivatives evaluate as follows:

$$
\begin{aligned}
\frac{\partial f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right)}{\partial \widehat{x}_u} &= -\frac{x_i - \widehat{x}_u}{\widehat{r}_i} \\
\frac{\partial f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right)}{\partial \widehat{y}_u} &= -\frac{y_i - \widehat{y}_u}{\widehat{r}_i} \\
\frac{\partial f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right)}{\partial \widehat{z}_u} &= -\frac{z_i - \widehat{z}_u}{\widehat{r}_i} \\
\frac{\partial f\left(\widehat{x}_u, \widehat{y}_u, \widehat{z}_u, \widehat{t}_u\right)}{\partial \widehat{t}_u} &= c
\end{aligned}
\tag{2.29}
$$

where

$$
\widehat{r}_i = \sqrt{\left(x_i - \widehat{x}_u\right)^2 + \left(y_i - \widehat{y}_u\right)^2 + \left(z_i - \widehat{z}_u\right)^2} \tag{2.30}
$$

The equation 2.28 under the consideration of the equations 2.29 and 2.26 changes to:

$$
\rho_i = \widehat{\rho}_i - \frac{x_i - \widehat{x}_u}{\widehat{r}_i}\Delta x_u - \frac{y_i - \widehat{y}_u}{\widehat{r}_i}\Delta y_u - \frac{z_i - \widehat{z}_u}{\widehat{r}_i}\Delta z_u + ct_u \tag{2.31}
$$

By the above process the linearization of 2.25 has been succeeded. Separate the known arguments of the expression from the unknown yields:

$$
\widehat{\rho}_i - \rho_i = \frac{x_i - \widehat{x}_u}{\widehat{r}_i}\Delta x_u + \frac{y_i - \widehat{y}_u}{\widehat{r}_i}\Delta y_u + \frac{z_i - \widehat{z}_u}{\widehat{r}_i}\Delta z_u - ct_u \tag{2.32}
$$

To simplify the expression 2.32, some new arguments will be declared as follows:

$$
\begin{aligned}
\Delta\rho &= \widehat{\rho_i} - \rho_i \\
\alpha_{xi} &= \frac{x_i - \widehat{x_u}}{\widehat{r_i}} \\
\alpha_{yi} &= \frac{y_i - \widehat{y_u}}{\widehat{r_i}} \\
\alpha_{zi} &= \frac{z_i - \widehat{z_u}}{\widehat{r_i}}
\end{aligned}
\tag{2.33}
$$

The arguments $\alpha_{xi}, \alpha_{yi}, \alpha_{zi}$ imply the direction cosines of the unit vector from the approximate receiver position to the *i*th satellite. The vector is defined as:

$$
a_i = \left(\alpha_{xi}, \alpha_{yi}, \alpha_{zi}\right) \tag{2.34}
$$

As a result the formula 2.32 can be written as:

$$
\Delta\rho_i = \alpha_{xi}\Delta x_u + \alpha_{yi}\Delta y_u + \alpha_{zi}\Delta z_u - c\Delta t_u \tag{2.35}
$$

The unknowns are now four: $\Delta x_u, \Delta y_u, \Delta z_u, \Delta t_u$ and so are the equations since we need at least four satellites. These equations can be put in matrix form by making the definitions:

$$
\Delta\rho = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \Delta\rho_4 \end{bmatrix} \quad H = \begin{bmatrix} \alpha_{x1} & \alpha_{y1} & \alpha_{z1} & 1 \\ \alpha_{x2} & \alpha_{y2} & \alpha_{z2} & 1 \\ \alpha_{x3} & \alpha_{y3} & \alpha_{z3} & 1 \\ \alpha_{x4} & \alpha_{y4} & \alpha_{z4} & 1 \end{bmatrix} \quad \Delta x = \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ -c\Delta t_u \end{bmatrix}
$$

Finally the solution would be:

$$
\Delta x = H^{-1}\Delta\rho \tag{2.36}
$$

This way of linearization works as long as the displacement $\left(\Delta x_u, \Delta y_u, \Delta z_u\right)$ within the threshold of the linearization point, which is determined by the receiver accuracy requirements. In case the displacement exceeds the threshold the whole process is repeated with approximate solution the calculated coordinates $\left(x_u, y_u, z_u\right)$. The errors that have been ignored at the satellite positioning, are actually translate to errors in the components of vector Δx as is shown bellow:

$$
\epsilon_x = H^{-1}\epsilon_{meas} \tag{2.37}
$$

Where ϵ_{meas} is the vector of the pseudorange measurement errors and ϵ_x is the vector of the errors in the receiver position and receiver clock offset.

To minimize the ϵ_x more than four satellites need to be measured. This results in over determination of the equations system. This format of the solution can be processed by the least square method and improve the estimations of the unknowns.

Chapter 3

Snapshot Positioning: Related Work and System Modeling

3.1 Snapshot algorithms

In contrast with the conventional GNSS receivers described as far, a snapshot PVT can be obtained by having assistance data, like the ephemeris, a rough time or position and raw data samples together with the PRN codes. Tracking and data decoding are not required. This way, the calculation of PVT can be done faster and more efficiently.

Since the snapshot receiver has only few milliseconds of recorded data, the broadcasted ephemeris data and the correction parameters cannot be obtained. For that reason, a secondary channel has to be used as an external data source.

The signal transmission time broadcasted by the satellite is not available, hence the pseudorange computation is not possible as it is based on the transmission and arrival time. In addition the snapshot receiver can detect the code phase within the signal code period.

As it is mentioned in [5], a pseudorange reconstruction algorithm was proposed by Van Diggelen which is based on an a-priori time of transmission t_{ref} and user position X_{RX} given within a certain accuracy and resolves the code period integer ambiguity for each satellite. By setting an arbitrarily chosen integer ambiguity N_{ref} for a reference satellite SV_{ref} and given ephemeris it is possible to compute the expected geometric range ρ and solve the common bias b_{ref}.

$$b_{ref} = N_{ref} \cdot \tau_{E1BC} + P_{E1BC,SVref} - \left(\rho(X_{RX}, X_{SV_{ref}}(t_{ref})) - \delta t_{clk,SV_{ref}}\right) - \epsilon_{SV_{ref}} \quad (3.1)$$

where ϵ_{SV} is the satellite dependent range error and $\delta t_{clk,SV_{ref}}$ the satellite clock error.

Having computed the common bias b_{ref} for the reference satellite, the ambiguity for each one of the rest satellites can be solved by:

$$N_{SV} = \left\lfloor \frac{\rho(X_{RX}, X_{SV}(t_{ref})) - \delta t_{clk,SV} + b_{ref} + \epsilon_{SV} - P_{E1BC,SV}}{\tau_{E1BC}} \right\rceil \tag{3.2}$$

where N_{SV} is rounded to the nearest code duration integer.

If the a-priori time of transmission and user position have been accurate enough, all the ambiguity terms are solved correctly and the so-called full pseudorange can be reconstructed.

Another way to obtain an unambiguous pseudorange measurement, is by the use of GNSS pilot signals' secondary code, when available. Modern GNSS pilot signals have a secondary code on top of the primary code in order to generate a long *tiered code*.

By the conventional acquisition method, only a single primary code sequence is being acquired in order to get the pseudorange measurement. As a result, the point in the secondary code at which the measurement was taken is unknown. In order to solve this ambiguity, a search method is utilized, exploiting that the secondary code sequence is a-priori known. By this method, both the code phase offset and the secondary code are reconstructed and the computation of the full pseudorange is possible without the prediction of the user position. The advantage is that no matter which a priori user position was given, the exact user position can be found.

As mentioned before, it is not possible to obtain the signal time of transmission because the snapshot is only few milliseconds. For that reason time of transmission can only be estimated, so the time error occurred is called *coarse time error* δ_{tc}. This fact causes an error at the satellite position which is used in the least squares solution of positioning. So the pseudorange equation also needs the product of the coarse time error and the pseudorange rate. Wording:

$$\begin{aligned} P_{E1BC,SV} + N_{SV} \cdot \tau_{E1BC} = \\ \left(\rho(X_{RX}, X_{SV}(t_{ref})) - \delta t_{clk,SV}\right) + b_{ref} + \dot{\rho}(X_{RX}, X_{SV}(t_{ref})) \cdot \delta_{tc} + \epsilon_{SV} \end{aligned} \tag{3.3}$$

After extending the state of unknowns, it is necessary to receive and acquire at least five satellite signals for three dimensional positioning. The state is given through the unknown receiver position, the common bias and the coarse time error:

$$state = [X_{RX}, Y_{RY}, Z_{RZ}, b_{ref}, \delta_{tc}]^T \tag{3.4}$$

For the PRS signals no pseudorange reconstruction is needed because PRS signals are non-periodic and the PRS code length exceeds the signal propagation time. Since the PRS signals are non-periodic, the PRS PRN sequences are valid only for some specific time period. If there is a match between the snapshot and a PRN sequence, the time of transmission for the specific satellite can be determined. Hence, there is no reason for the use of a-priori time of transmission and the equation of positioning with the five unknowns can be reduced down to four unknowns since the coarse time error will not be included in the equation. The challenge of positioning with Galileo PRS results from the fact that the PRS spreading code is non-periodic. That way, the time frame in which the acquisition can performed is strictly predefined and precise time information is required. Otherwise the complexity of acquisition increases. As a solution to that, a dedicated direct PRS acquisition module has to be used, or time has to be obtained from the OS components or transmitted by a secondary channel. Every uncertainty can result to an increase to the code phase search space.

3.2 Related work

In the experiment [13], the snapshot of data record was tested to find out the influence of integration time, sampling frequency, Doppler frequency shift and quantization on the possibility to acquire five or more satellites and the position solution accuracy. This experiment is referred to a software defined receiver (SDR). In a SDR the incoming signal is digitized and all the rest of the calculations are implemented in software, minimizing the hardware part and maximizing the flexibility of the receiver.

As a result the experiment concludes to the following:

- With the increase of the integration time more satellite can be acquired and also the Doppler frequency shift and the code delay can be determined more accurately. More specifically in order to acquire more than five satellites, the integration time should be at least 4 ms.
- In case the sampling frequency is reduced by two times, the length of data record has to be increased in two times. Though the reduction of the sampling frequency is not reasonable since signal-to-noise ratio (SNR) will stay low.
- The quantization from the RF front-end does not effect the satellites acquired, and it is not reasonable to compress data.

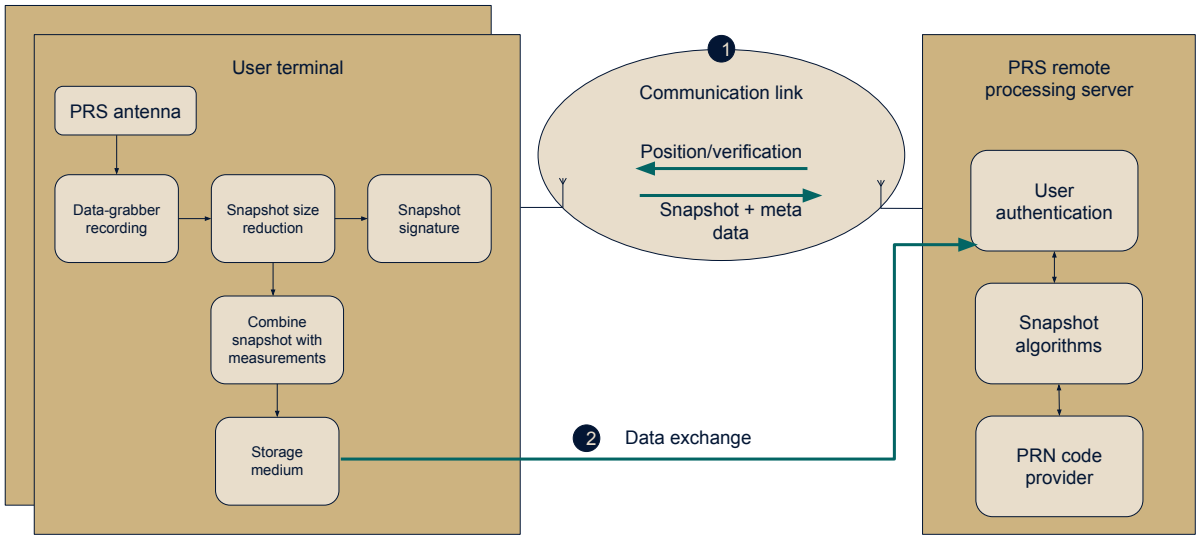

Figure 3.1: System architecture with communication link (1) and data transfer link for postprocessing (2)

Based on the above information, related work based on snapshot positioning will be described below. There are basically two different system concepts where the snapshot approach is used. The one system concept is server-based authentication positioning and the other one is ultra-low-power GNSS receivers.

3.2.1 Server-based authentication positioning

On the concept of the server-based authenticated position there is an embedded device which performs the snapshot. Then the device sends the snapshot to a server where the PVT is calculated. Some of the implementations that are based on this concept will be described below.

3.2.1.1 Architecture of Galileo OS/PRS Snapshot receiver

According to the [5], the main challenge of PRS snapshot receivers are to make sure that the process of the signal for positioning, takes place in a secure server environment. Only authorized user terminals can communicate with the server, and make use of all its features without endangering the security of the Galileo PRS as is shown in figure 3.1.

The antenna receives OS or PRS signals, on E1A, E6A or both frequency bands.

Then a raw data snapshot is recorded by a front-end, and if necessary can be reduced, on the application implemented. It should be mentioned that the PRS signal has a dual-band capability to provide a higher jamming protection. Even if one of signal bands is jammed, the other band may still be untouched.

The size of the raw data snapshot is given by the sampling rate f_s, the quantization bits Q and the snapshot length l as can be seen in the equation:

$$size = f_s \cdot Q \cdot l \tag{3.5}$$

Data can be compressed by resampling and filtering the received signals, reducing Q or limiting the recording length l. The trade-offs for reducing the size are: probability of detection and loss of accuracy.

The raw PRS snapshot can be used as digital fingerprint for authentication with time and georeference or just georeference on measurement data, files or documents.

In order to create a snapshot signature, the raw PRS snapshot has to be cryptographically combined with the measurement in a way that the time and position information can be authenticated using PRS. One solution for this problem is to exchange public/private key also referred as asymmetric cryptography. The user terminal uses a private key to sign a hash value of both the raw PRS snapshot and the measurement. The public key is then shared with the customer. The public key, the hash values and the actual measurement with the raw PRS snapshot data are stored on the server. For the verification of the measurement with the raw PRS snapshot, a 3rd party can compare the transmitted hash sums with hashes generated by the provided data. The provided hash sums can be verified using the user's device public key.

The communication channel can be either uni-directional or bi-directional depending on the application. For the uni-directional link all snapshot data must be stored on a local storage volume of the user terminal. The forwarding of the raw data snapshots to the PRS remote processing server may then happen at any time a secure connection to the server can be established. For that reason the tracking is not real-time capable, but can still provide a proof of a georeferenced action or time with a PRS guaranteed location and time.

By using a bi-directional link, the user terminal can forward the PRS snapshot samples including the corresponding meta-data to the processing server, the server can then compute the snapshot PVT using the raw samples and send them back to the user terminal. Depending on the application, parameters like the commu-

nication channel used, snapshot length and assistance data should be taken into account.

If a mobile communication link is used, snapshot size has to be reduced as much as possible because the bandwidth is limited and expensive.

The OS/PRS remote processing server has to ensure first that only authorized user terminals can use its services. Having guaranteed that the environment is secure, the server can then calculate the PVT solution using Galileo PRS. For the calculation of the PVT solution the snapshot algorithm block has to be connected with a PRN code provider. The PRN code provider of the PRS has a crucial meaning since it includes PRS security modules with the classified algorithms to generate PRS PRN streams.

3.2.1.2 PROSPA: Open Service Authentication

PRS/Open Service Positioning and Authentication (PROSPA) is a pilot study, [6], supported by the UK space agency (UKSA), who are currently investigating solutions for key management within the Galileo PRS. The objective of PROSPA is to demonstrate the proof of concept for a Galileo open service authentication system using PRS signals. PROSPA uses *snippets*, time sliced subsections of the encrypted PRS signals, which are distributed to authorized user receivers over a communications link. The received snippets are then used to authenticate the received Open Service signals. PROSPA maintains benefits like authenticity, anti-spoofing and access control but without having the disadvantage of classified key distribution to the user receivers. It is a cost-effective way of providing an authenticated Open Service to government authorized users, and therefore it could provide benefits to public services and user groups which could otherwise not have used PRS. The final PROSPA system will include snippet generators located at secure centres. Each generator will essentially be an enhanced PRS receiver. Snippets of the encrypted PRS signal are generated by a proprietary algorithm which does not reveal the encrypted code. As the snippets contain only small sections of the PRS signal that have already been transmitted globally by the Galileo satellites, they are unclassified. The snippets are checked in the service centre by a snipper validation receiver and then they are distributed to the user receivers by a communication channel. The user receivers perform a time aligned correlation with the PRS snippet in order to authenticate the open service signals. A strong correlation shows that the PRS signal is present and hence the signal is authentic and suitable for use.

PROSPA targets at the minimization of the communication capacity that each user receiver requires. For this purpose existing communication networks are used, for example 3G or 4G, for snippet distribution. For the delivery of snippets from the server, standard commercial encryption and user authentication can be used. Hence, smartphones can acquire PRS positioning information.

The demonstration of PROSPA uses an RF simulation of the Galileo PRS and Open Service signals. For the PRS emulation, a random stream of chips which is unknown to the receiver was used. Since the signals are designed ahead, the snippers are loaded in advance on to an file transfer protocol (FTP) server. The snippets are acquired by an Android terminal and sent to the user receiver via Bluetooth. Subsequently, the receiver returns to the terminal an authenticated navigation solution. The demonstration of the PROSPA proves that authentication is possible by the use of snippets of the PRS signal.

3.2.1.3 Signal Authentication: A secure civil GNSS for today

As mentioned before, some GNSS signals are encrypted or obscured, which implies they are designed to prevent spoofing or to deny unauthorized access. In [7] it is proposed a method to maintain the anti-spoofing benefits of the secret codes without the direct access to the codes. This can happen by joining the signal received in a specific location, with a nearly synchronous signal received at a remote station. This article uses the GPS L1/P(Y) code as the secret code, and L1P(Y) is used as well at the illustrative example below. Though the techniques described can be applied to other navigation signals as well.

The processing of the signals to most of the commercial GNSS receivers is common with small variations. The innovation suggested and proposed on this project, is the joint processing of signals received at two separate locations to access the secret code sequence for authentication. This authentication process recognizes and exploits the fact that the P(Y)-code sequence received at the one location is identical to the sequence received to the other location except the differential satellite-to-receiver signal travel time Δt. The method can be seen at the figure 3.2.

The steps required for joint processing are the following:

- Record GPS raw data at location number 1 and location number 2.
- Transmit a data snapshot and a timestamp of the data snapshot from the user device to the reference station for processing for each satellite for both user

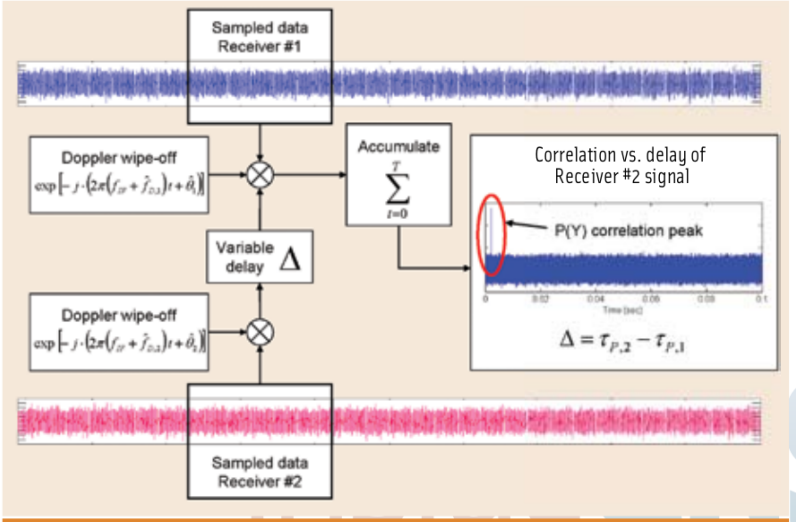

Figure 3.2: Two-receiver correlation with unknown CDMA sequence. The presence of a P(Y)-code correlation peak authenticates the signal. The peak timing for several satellites allows relative positioning (and timing) of receiver number 1 vs. receiver number 2.

device and reference station.

- Perform Doppler frequency wipe-off.
- Estimate carrier-phase in order to separate to in-phase and quadrature components.
- Correlate the Q-channel from the user device with the Q channel from the reference station: slide the correlation window until a peak of sufficient magnitude appears.

The correlation peak shows the presence of components $s_{P(Y)}(t)$ in receiver number 1 and receiver number 2 signals, and is accomplished by sliding the correlation window until $\Delta t = \tau_{p,2} - \tau_{p,1}$ also the correlation peak establishes authenticity. The value of Δt accounts both for receiver clock offset and for different satellite-to-receiver signal travel times. Therefore, having the Δt measurement on several satellites, the presence of the correlation peaks confirms the authenticity of each receiver signal observation and also locate receiver number 1 with respect to receiver number 2 in an analogous manner to GPS carrier-phase differential positioning.

By this method the authentication architecture essentially transfers the security and possible the navigation functionality from the user device to the trusted authentication processor. The novelty here is that there is no PVT calculation but just

a correlation between snapshots. This architecture supports many PNT security applications at very reasonable cost.

3.2.1.4 Ultra low-cost PRS receiver

The main purpose of ULTRA project, [10], is to develop an ultra low-cost PRS receiver capable of addressing low-end applications. This project has been proposed as an accelerator to provide stimulus to the support and uptake of PRS across a range of users and applications. In this proposal, the mobile system grabs the raw GNSS data. Then a secured communication channel is used to transmit the raw GNSS digital samples to the secure server. The server includes a secured software receiver and the software receiver calculates an off-line PVT solution for each mobile, using algorithms for GNSS signal processing.

The features of the ULTRA PRS receiver are the following:

- Easy access: The mobile receiver does not need a security module so the access is easier to PRS services to a large community.
- Low-Cost: The components and the system architecture aim at reducing manufacturing costs.
- Distributed architecture: The PNT processing is distributed between fields unit and the secured back-office which does not suffer any computer resource constraints, enabling the implementation of advanced algorithms.
- Portable, low-power: The specific design takes into consideration the need to mount the receiver on a variety of platforms and the need to convert installations. Also with the innovative power management functions preserve battery in order to have long life operations.
- Reliable: In order to ensure reliable position fix, the design is based on multiple software receiver architectures.

The possible end application areas for the ULTRA PRS receiver include: emergency services like search and rescue operations, critical transport services like mass transport, critical energy services like timing and synchronization of civil and governmental telecommunications networks, strategic activities, economic activities and commercial activities.

3.2.1.5 Data Compression for Assisted - GPS Signal Processing

In this research, [8], a concept for assisted-GPS (A-GPS) server-based systems is suggested. The receiver records sampled IF data and transfers it directly to the server. The acquisition search as well as the position solution computation are taking place on the network server. As a result, the data transfer between the client and the server requires a high bandwidth. Therefore, data compression changes the efficiency of this implementation significantly. The purpose of this research is to determine the feasibility and optimal values for the parameters of the system by minimizing the data size while preserving reasonable detection sensibility and accuracy.

The fact that the acquisition search and the computation of the position solution are taking place at the server simplifies the hardware in the client, reduces computation load and allows the receiver client to capture more signals besides the GPS. There is a strong relation between the sensitivity of a GPS signal and the amount of captured IF data in a given snapshot of time. The accuracy of the position solution in the server, relies on the capability of the network channel for data transmission. The performance of the transmission network channel depends on the amount of raw IF data that needs to be transferred from the client to the server.

The trade-off, between the receiver sensibility and the amount of captured data from the receiver, is determined by the selection of parameters in the signal processing like the front-end bandwidth, the sampling rate, the number of quantization bits and the observation time.

3.2.2 Ultra low-power GNSS receivers

On the concept of the ultra-low-power GNSS receivers, the receiver calculates the position only at certain time slots without keeping signals in tracking, using the snapshot algorithm. The main constraint in this concept is the power consumption and for that reason these implementations use dedicated processors and ASIC instead of FPGAs. More details about implementations on this concept will be described bellow.

3.2.2.1 A Novel Design of Low Power Consumption GPS Positioning Solution Based on Snapshot Technique

The proposition of [12] addressed the problem of high energy consumption in mobile applications or applications in regions where GPS signal is present only for short time. The design they suggest, is a fast GPS receiver which only needs

several milliseconds of GNSS data and is based on the snapshot technique. The power consumption of the design they suggest is about 23% of the consumption of a typical GPS receiver. The design includes two parts. The GNSS grabber which collects the IF digitized data and the server which post-processes the digitized data in order to estimate the PVT of the GNSS grabber. The snapshot processing takes advantage of the 1-bit quantization front-end and the Doppler positioning in order to achieve low power consumption. The position solution is calculated without a-prior knowledge of the initial position. The accuracy achieved is about 14 m in horizontal position. According to the results, the design proposed reduces the size of the dataset while having a higher performance than related studies.

3.2.2.2 Snapshot positioning for low-power miniaturised spaceborne GNSS receivers

In the paper [11] an experiment carried out by NASA called micro-GPS is described. The GPS receiver is a SDR.It is mostly discussed as a non-terrestrial, space, receiver for *cubesats*. It supports different modes depending on the battery status. The modes are: autonomous, assisted and passive.

In autonomous mode, the receiver will perform all the standard tasks like acquisition, tracking, navigation data decoding and positioning. This mode is the most energy consuming mode. It is computationally more expensive than the other modes but it is required in the launch and deployment phase of the *cubesat*. In assisted mode the receiver works in snapshot mode. For that reason long term ephemeris are uploaded into memory by the ground station. The receiver processes 20 ms of data every epoch, and in between the measurements the receiver is sleeping. This keeps the balance between power consumption and performance and it can be enabled as soon as the *cubesat* connects with the ground control station for the first time when an up-to-date ephemeris is uploaded. In passive mode the receiver records snapshots of RF data every epoch and broadcasts it to the ground station whenever this is possible. This mode is enabled typically at the end of the *cubesats* mission, when the orbit determination is carried out on the ground and the receiver consumes almost no power.

3.2.2.3 Snapshot positioning: Next Generation GNSS Receiver for Low Power Applications by Baseband Technologies

This snapshot positioning implementation is complementary to existing WIFI positioning services. WIFI positioning works best, in dense urban environments with closely spaced wireless access points. In contrast, in large open spaces away

from wireless access points, WIFI positioning cannot be efficient. In these cases, snapshot positioning using GNSS signals works best. At this implementation, the power consumption can be eliminated enough, to extend the battery life by several orders of magnitude. This can happen by the use of different modes for the snapshot positioning engine. In the energy-efficient mode, the GNSS enabled device turns on the GNSS hardware only for a short period of time, like 2 ms and can offload all the energy and computationally demanding tasks to a remote server, such as a cloud-based server, at a convenient time in the future. In this mode, the snapshot receiver can operate up to several weeks on a single coin cell battery, or years on a typical mobile phone battery. For real-time applications, the GNSS enabled device sends in real-time the digitized GNSS data to a cloud-based server by the use of GSM or Bluetooth or WIFI. Likewise in this mode, the energy and computationally demanding tasks are offloaded to the server which has no energy constraints like the mobile device.

3.2.2.4 A Novel Multi-Step Algorithm for Low-Energy Positioning Using GPS

At the low-energy GPS prototype implemented from [14] the signal transmitted by the satellite is only sampled for 2 ms. The algorithm used increases the robustness by filtering estimated residuals instead of using a different database and can work with fixed and moving targets. The system includes a device that samples the GPS signals and a server that utilizes Doppler navigation and coarse time navigation in order to compute the positions. The median positioning error is 40 meters even if the receiver is moving at 80 km per hour. The two factors of the energy consumption in a receiver are the turning on radio in order to sample the signals transmitted by the satellites and the processing of the sampled signals in order to estimate the position. In this implementation these two tasks are performed on two physically different devices described above the receiver and the server. Since receivers work on batteries they have limited energy constraints and the server has no energy constraints and higher computational power. Based on that, the computations are taking place at the server, and also it can store generic information like the ephemeris. In this way, the time the radio has to be turned on is lowered by a factor of between 3000 and 15000. Furthermore by the implementation described at this paper, the time of transmission is not needed to be decoded from the incoming signal so the required sampling time is decreased.

3.2.2.5 A GPS/Galileo Software Snap-Shot Receiver for Mobile Phones

The paper [15] describes a software based receiver, which uses the snapshot technique and it is implemented for mobile phones. Snapshot positioning is a fast and efficient method specially when the necessary assistance data can be provided by the mobile phone communication links. By the approach described, the power consumption of the device and the time to first fix are reduced. The architecture of the software proposed is flexible. It can use signals from multiple systems using the definition of the reference codes. This system uses assistance data to reduce the Doppler search space at the signal acquisition. The results of this research shows that the approach proposed in this paper delivers good performance in reasonable computation time.

3.2.3 Security vulnerabilities of FPGAs

This thesis targets at a system with strong security. Hence, the security vulnerabilities of the FPGAs are considerable.

Side-channel attack is any attack based on information gained from the implementation of a computer system, rather than weakness in the implemented algorithm itself. Timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information which can be exploited.

Even though power side-channel attacks require specialized equipment and physical access, according to [17], for integrated FPGAs this is not true. The integrated FPGAs enable software-based power side-channel attacks without physical proximity. Side-channel attacks are based on information gained from the implementation of the computer system rather than weaknesses in the implemented algorithm. According to the paper [17], an on-chip power monitor can be built by the use of ring oscillator (RO). A RO observes the power consumption of other modules on the FPGA or the SoC. Based on that a RO can perform a power analysis attack on a RSA cryptomodule and power consumption observation on the FPGA soft core. A power side-channel attack on the FPGA soft core can break timing-channel protection for a RSA program running on that soft core.

Another vulnerability is showed at the paper [16]. According to this paper, the encryption mechanism can be completely broken with moderate effort. It is proved that the key extraction was possible only by the measurements of a single power-up. Access to the key allows cloning, design manipulation and fraud. Possible threats are also the theft of IPs, reverse engineering and the introduction of hardware

Frequency band	Chip rate	FFT core frequency
E1A	2.5575 MHz	171 MHz
E1B	1.023 MHz	167 MHz
E6A	5.115 MHz	179 MHz

Table 3.1: Estimated FFT core frequency for 10KHz search range

Trojans.

3.3 System modeling

The approach used for this thesis is in between the two system concepts described. The target of this thesis is to provide strong security as the applications of server-based authenticated positioning concept system but also in a real-time system with low power consumption as the concept of ultra-low-power GNSS receivers. At the implementation of this thesis the complete PVT calculation is done on a RISC-V embedded processor. Therefore there are much more performance and resource constraints. The fact that the Mi-V core used is a soft core in a low-power FPGA, *PolarFire*, makes the implementation more challenging. Since the security requirements are in greater interest than the low-energy consumption, a flash-based FPGA was used because both ASIC and RAM-based FPGAs have vulnerabilities.

Since this thesis is a part of a bigger project, there is no full-scale functional system modeling in the scope of this thesis. The hardware part of this design, has been already developed in *Xilinx* technology, and part of the purpose of this thesis is to test the functionality of this design on the Microsemi technology. For that reason IP cores implemented in Fraunhofer and Microsemi IP cores were used and a hardware system integration had to be performed. A significant aspect of this thesis has been to convert an existing design based on Xilinx technology to Microsemi technology in order to take advantage of its ultra-low power and security attributes.

As it is already known from the theory, for the parallel code phase search method FFT calculations have to be performed. As it will be described more detailed later on, the Doppler search range is proportionate with the frequency of the FFT core. To achieve the typical range of 10 kHz depending on the frequency band used, the frequency of the FFT core is shown in table 3.1.

The library that controls the acquisition is developed in Fraunhofer and it is designed to work on operating system. Part of this thesis was to change this library in order to work on bare-metal applications. For the snapshot positioning

application a reference script in python was used. The reference script can achieve accuracy 0.0023 m.

Chapter 4

System design

4.1 Hardware Platform

The target device selected for this project is a PolarFire FPGA. Those FPGAs offer features like low energy consumption and also have advanced security features. The features that allow these devices to have low energy consumption are the use of non-volatile technology. Moreover the Microsemi provides low power devices with up to 35 percent lower static power with identical electrical specifications like standard and speed grade. Security at the PolarFire FPGAs is issued by many features like supply chain assurance, IP protection and data security. Furthermore, the FPGAs have integrated tamper detectors and tamper responses, enabling users to monitor the environment and the operating parameters of the design and in case of an event detection the user can perform some of the provided actions The FPGA targeted in this work is the MPF300TS PolarFire of Microsemi. This device has 300 K logic elements, 924 math blocks and 20.6 Mbit total RAM Mbits.

4.2 FPGA Design

The hardware system implemented for this thesis consists of a RISC-V softcore as CPU and a freely configurable and programmable logic part. As shown in figure 4.1, it is based on a Mi-V processor subsystem which includes the Mi-V softcore, a memory unit and GPIO, UART, SPI core peripherals. The communication between the soft core and the other IPs is based on the AXI protocol. The rest of the system consists of the cores that perform the Fast Acquisition method, namely the PRN Code Generator, the Fast Acquisition unit and the FFT core.

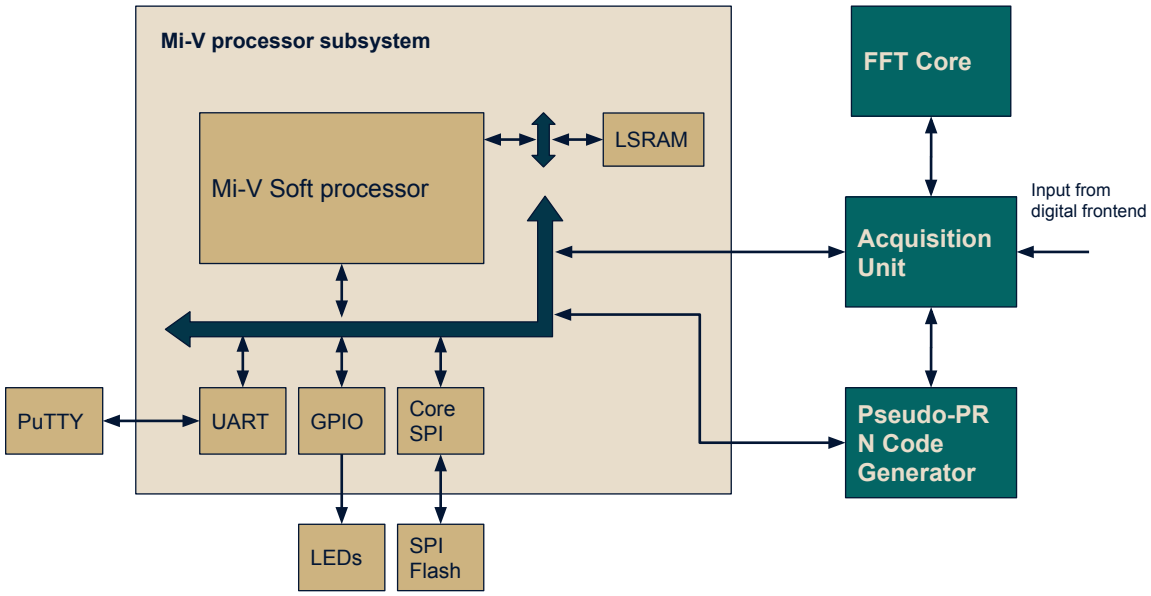

Figure 4.1: Hardware design of the thesis

4.2.1 Mi-V

RISC-V is an open source ISA based on established reduced instruction set computer (RISC) principles. The Mi-V core is a softcore processor designed to implement the RISC-V standard RV32IMA ISA for FPGA soft-core implementations. The diagram block of the Mi-V is shown in figure 4.2. The processor is based on the Coreplex E31 by SiFive and provides a single hardware thread. The Mi-V core allows a single-issue in-order 32-bit execution pipeline with a peak sustainable execution rate of one instruction per clock cycle. An example of the pipeline and its timing is as shown in figure 4.3 and in table 4.1. The core includes a RISC-V standard platform-level interrupt controller (PLIC) configured to support up to 31 interrupts with a single priority level. In this design two interrupts are connected to this core one from UART and another one from the Fast Acquisition unit. The core has full external debugger support over an industry- standard JTAG interface supporting two hardware breakpoints. Mi-V incorporates two external AHB interfaces, bridged from the internal TileLink interfaces. The AHB interface for the memory, is used by the cache controller to refill the instruction and data caches. The AHB interface for the I/O is used for uncached accesses to I/O peripherals. The reset vector address of the core is 0x70000000.

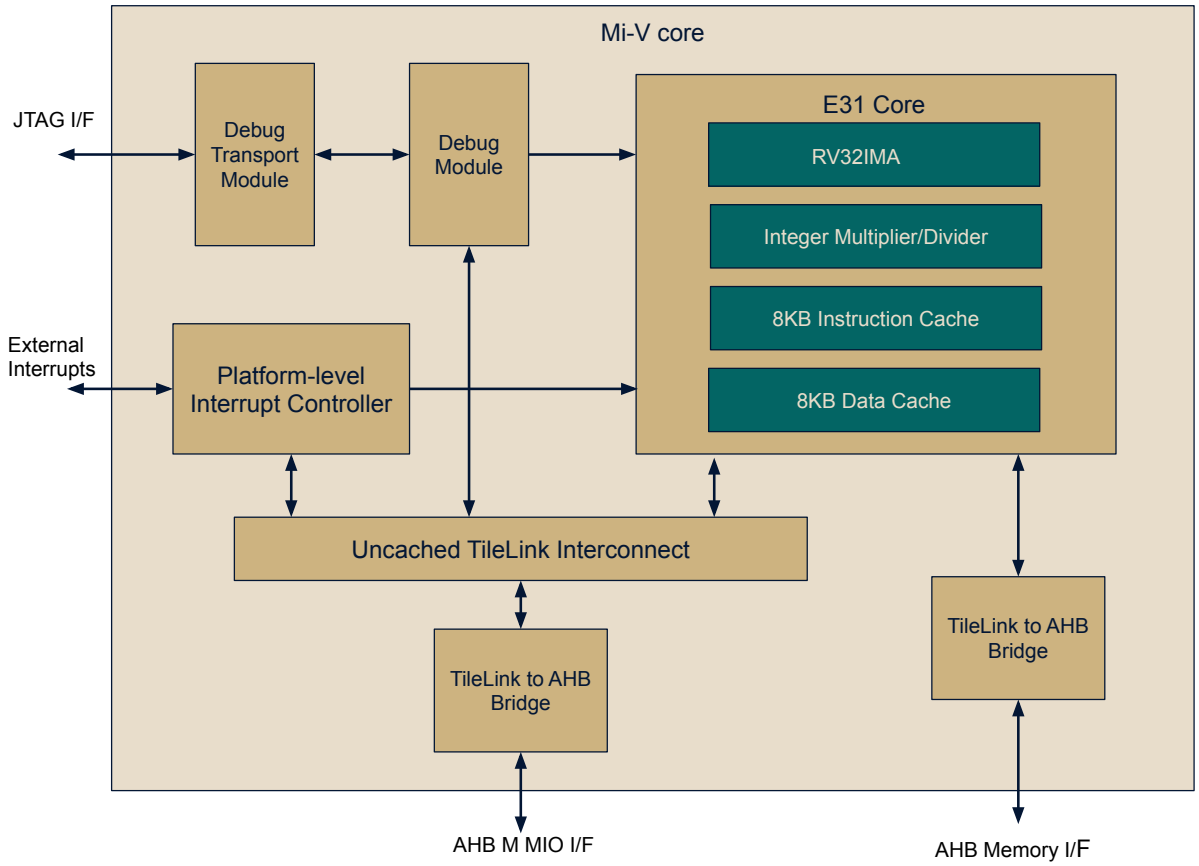

Figure 4.2: Mi-V block diagram

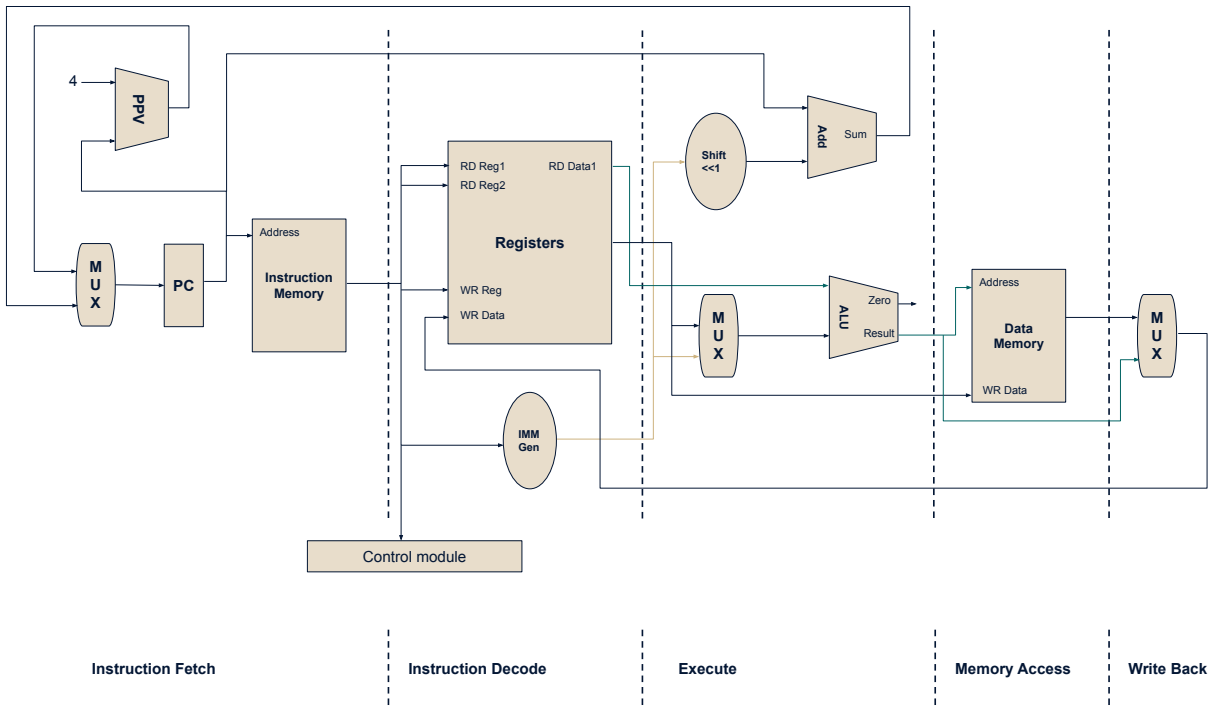

Figure 4.3: Example Five Stage Pipelined Architecture

Clock cycle	1	2	3	4	5	6		n
Fetch	Instr. 1	Instr. 2	Instr. 3	Instr. 4	Instr. 5	Instr. 6	...	Instr. n
Decode		Decode Instr. 1	Decode Instr. 2	Decode Instr. 3	Decode Instr. 4	Decode Instr. 5	...	Decode Instr. n-1
Execute			Execute Instr. 1	Execute Instr. 2	Execute Instr. 3	Execute Instr. 4	...	Execute Instr. n-2
Mem access				RD/WR Mem 1	Mem Access 2	Mem Access 3	...	Mem Access n-3
Write back					Write Back 1	Write Back 2	...	Write back n-4

Table 4.1: Example Pipeline Timing

4.2.2 AXI bus interconnect

The AXI interconnect bus is a configurable core that connects the soft processor with the rest of the modules and it is compliant with AXI protocol. Inside the AXI interconnect core a AXI4 crossbar core routes traffic between the slave ports (SP) and master ports (MP). In this design the core connects the two memory-mapped masters from the Mi-V to four memory-mapped slave devices. The two masters are the AHB interfaces for I/O and memory access. The master IF are specified for AHB-Lite communication. The four slave devices are the LSRAM, the PRN Code Generator, the Fast Acquisition core and the peripherals GPIO, UART and SPI Core. The slaves are type AXI4 for the LSRAM core,AXI4-Lite for the PRS Code Generator and the Fast Acquisition Unit and AXI3 for the peripherals. However, all three peripheral devices have APB slaves and therefore four intermediate bridge

Devices	Protocol	Address range	Data width (bits)
LSRAM	AXI4	0x70000000-0x7fffffff	64
Acquisition Unit	AXI4-Lite	0x60030000-0x60003fff	32
PRS Code Generator	AXI4-Lite	0x60004000-0x60004fff	32
Peripherals	AXI3	0x6000000-0x60003fff	32

Table 4.2: Table of the slaves configurations at AXI bus

modules were used in order the communication protocols to be compatible. More details about this module can be found in [21]. At the table 4.2 there are details of the AXI slaves.

4.2.3 LSRAM

The large static RAM (LSRAM) core has the purpose of the main memory of the RISC-V. The core has write and read interface. The address width of the memory is 32 bit and the width of ID is 8 bit.

4.2.4 UART

The UART Core is a serial communication controller with a serial data interface. For the purpose of transmitting and receiving data, the FIFO mode was enabled. According to the documentation, [22], when transmitting data, the data are loaded into the transmit FIFO until the TXRDY signal is driven inactive. When the signal is driven inactive the transmission begins until the transmission FIFO is empty. In order to receive data, the activity of RX signal is monitored. When a START bit is detected the receive state machine begins to store the data in the receive FIFO and when the transaction is complete the RXRDY signal indicates that there are data available.

4.2.5 GPIO

GPIO core is the controller of a General-purpose I/O. In the specific design it is used to control the LEDs for debugging reasons. The APB data width is 32 bit and the number of I/Os used is 4.

4.2.6 SPI core

SPI Core is a controller core used for synchronous serial communication using Motorola serial peripheral interface (SPI). The FIFO depth of the core is 32 bit and

the frame width is 8 bit. In the specific design it controls the external SPI Flash memory and it is also used mainly for debugging reasons.

4.2.7 Fast Acquisition unit

The Fast Acquisition unit controls the FFT core and the PRS Code Generator core. It implements the parallel code phase search method as it is described at 2.2.2 and also as it is shown in figure 2.2. It has input from a digital frontend, two 4-bit vectors, one for the real part of the signal and one for the imaginary part of the signal. Also the PRN Code Generator provides the Fast Acquisition unit with 128-bit chip sequences. The unit sends the signal input and the PRN code sequence input to the FFT core, where the transformation from time to frequency domain takes place, and then processes the output of the FFT, respectively to the method that implements. One of the modules that is included in the Fast Acquisition unit is the FFT Controller. This core generates the FFT configuration and manages the FFT inputs. In addition, it processes the FFT output by multiplying the samples FFT and the PRN FFT and detects the peak of the result. More details about the FFT controller are provided in figure 4.4.

4.2.8 FFT core

The FFT core provided by Fraunhofer, computes the FFT operations and sends the result to the Fast Acquisition Unit. The core handles the input samples in a bit-reversed order, but for compatibility reasons, additional logic has been added between the FFT controller and the FFT core to convert the samples from the one format to the other. The additional logic consists of two buffer units. Each buffer unit is able to store a full set of input samples for a L-FFT and two finite state machines, one for writing the buffers and one for reading from the buffers the samples.

The FFT core is based on a radix-2 architecture. According to the radix-2 approach, a FFT of the length N and $Nmod2 = 0$ can be broken down to $M = N \log_2 N$ stages and each stage consisting of $\frac{N}{2}$ butterfly-structures. The butterfly-structure performs a 2-point FFT and consists of two complex adders and one complex multiplier with a twiddle factor W_N^k with $W_N^k = e^{-i\frac{2\pi}{N}k}$, where $0 \leq k \leq \frac{N}{2}$. The radix-2 architecture of the FFT algorithm can be seen in figure 4.5. The core, computes one FFT of PRN code, one FFT of averaged samples and x FFTs of complex multiplication for Doppler search, plus three dummy FFTs that

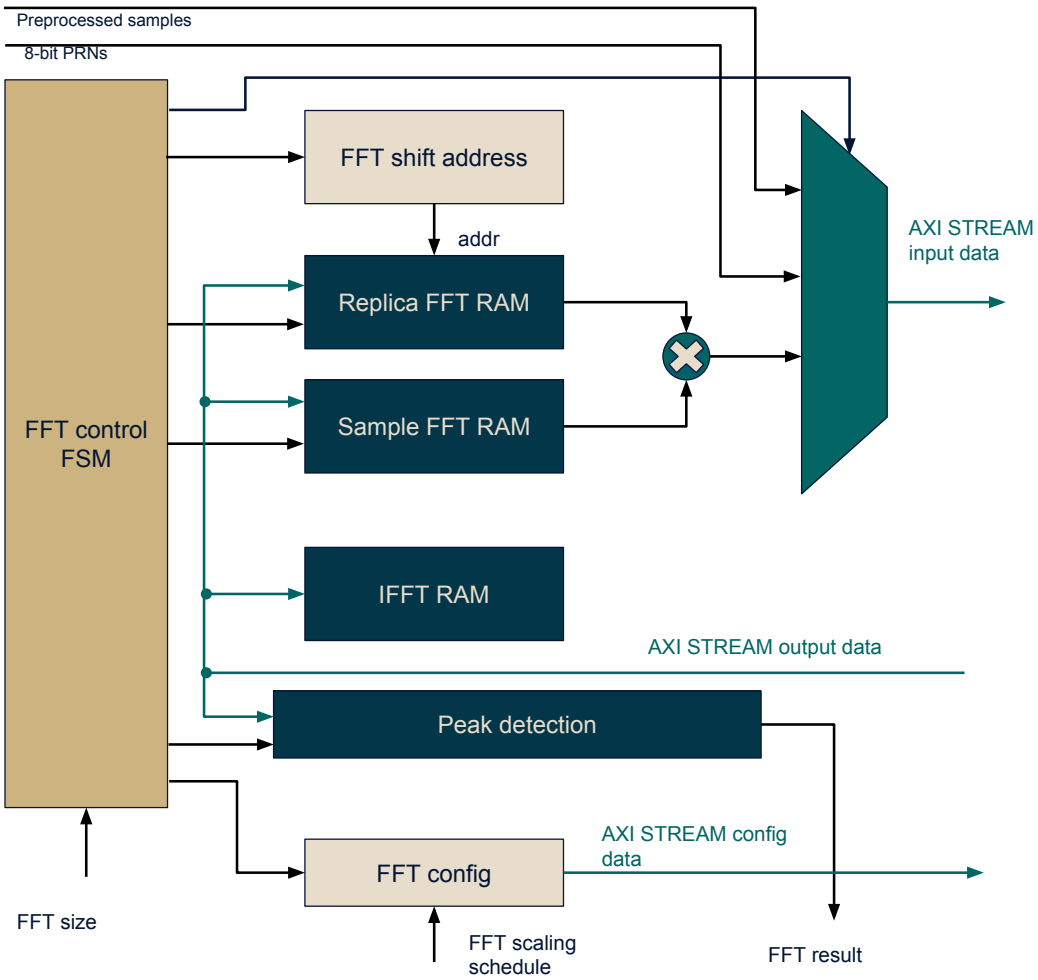

Figure 4.4: Diagram of FFT controller

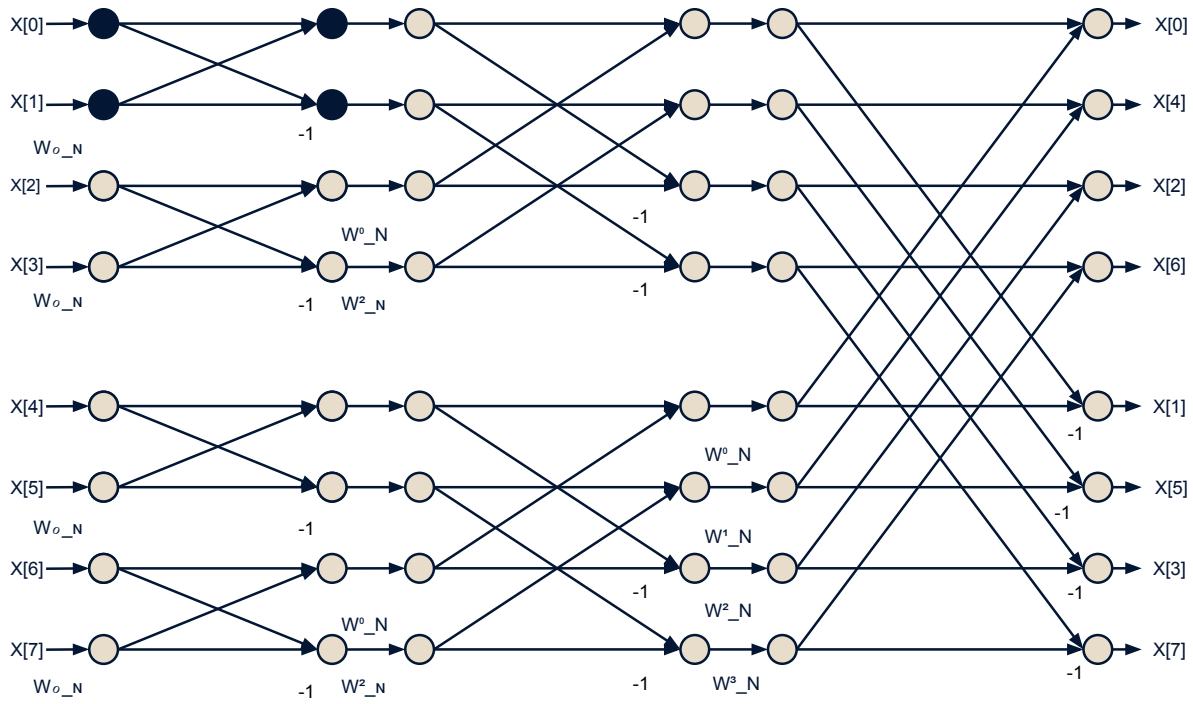

Figure 4.5: Radix-2 architecture of FFT

keep the internal pipeline of the core running and absorb the FFT core latency.

4.2.9 Pseudo-PRS code generator

The purpose of this core is to generate PRN sequences based on GPS P-code which can be used to track the *PRS noise* signal generated by the SPIRENT signal generator.

4.3 Integration of system

The integration of the system can be divided in two parts. Firstly, the integration of a Mi-V processor subsystem took place and later on the cores that implement the fast acquisition were added in the design. The part of the Mi-V processor subsystem includes the cores of Mi-V soft processor, the LSRAM, the AXI bus interconnect and the peripheral devices and it is shown in figure 4.6. The connection of these

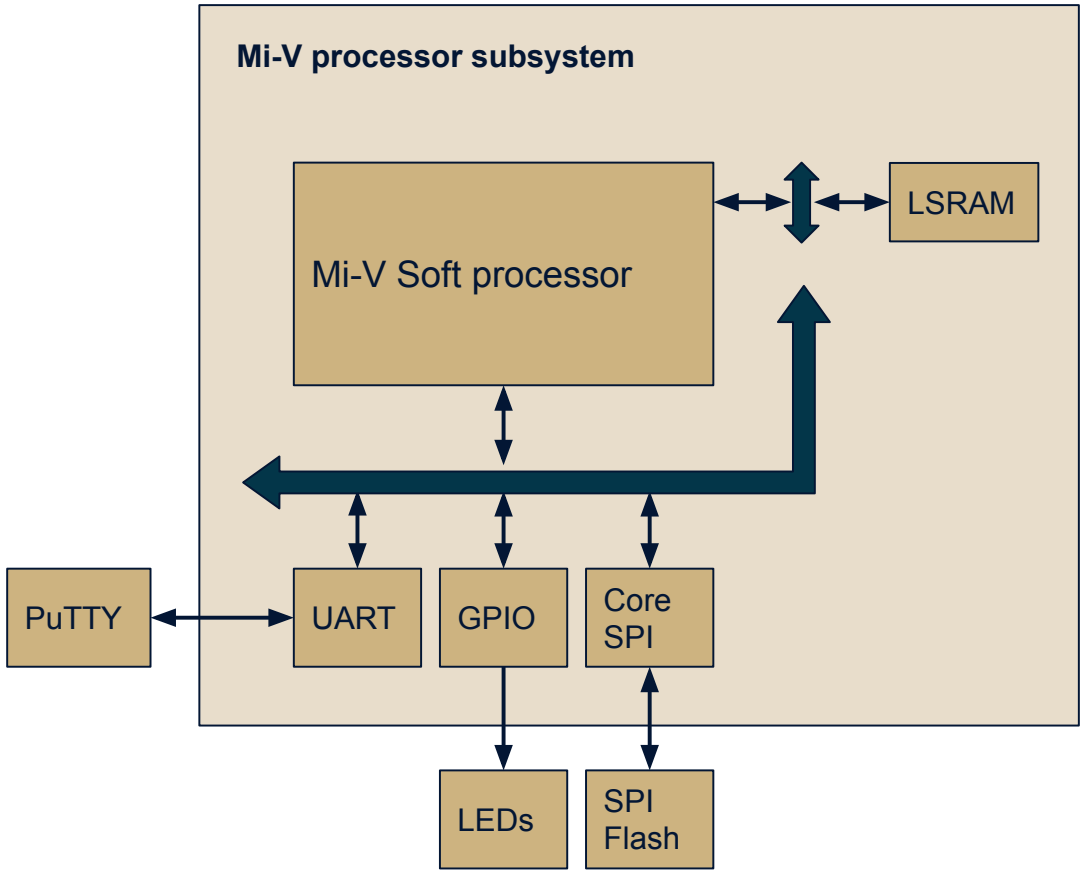

Figure 4.6: Mi-V processor subsystem

cores was mostly straight forward except of the peripheral devices, UART, GPIO and Core SPI which were not compatible with the AXI protocol and for that reason *protocol bridges* were used. To complete the design as it is shown in figure 4.1 three more cores were added in the design the FFT core, the Fast Acquisition Unit and the Pseudo-PRN Code Generator. For the integration of these cores their configurations had to be compatible with each other and also compatible with the AXI interconnect bus.

4.4 Clock Domains

At this design two asynchronous clocks are used as can been seen in figure 4.7. The one clock domain has frequency 83.333 MHz and the other clock domain has frequency 54 MHz.The high frequency domain is the domain of the processor and

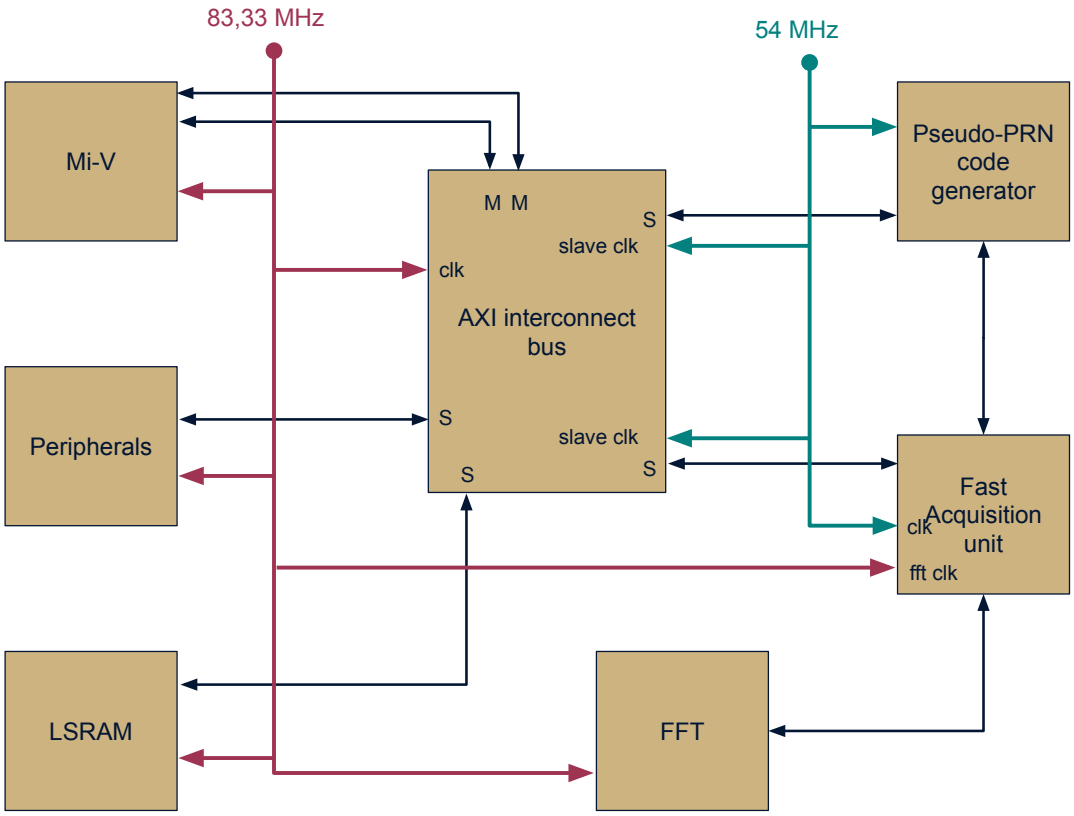

Figure 4.7: Clock crossing of the main design

the FFT core. The lower frequency domain is used at the PRN Code Generator core and at the Fast Acquisition Unit and it is the clock frequency connected to the digital frontend. The FFT core is connected to the faster clock to allow a larger number of FFT operations for the same snapshot of samples. This is happening because for every data set, according to the method of parallel code phase search, at least three FFT computations are taking place. For more details about the number of the calculations needed see section 4.5.2.

4.5 Software

The tool used for development and debugging the applications of this thesis is the SoftConsole v6. For all applications developed for this thesis the drivers for the peripheral devices of the hardware design, have been used. More specifically, the drivers of the UART, Core SPI and the GPIO are used. In addition the library with

the hardware abstraction layer functions for Mi-V soft processor is included as well. This tool is an Eclipse-based graphical integrated development environment (IDE) supported by Microsemi.

4.5.1 Debug software

For debugging reasons an application was developed with the purpose of reading values from registers and storing values in registers. The supported commands for the user interface are the commands *peek* and *poke*. The program operates with the use of interrupts, using an Interrupt Handler based on the interrupt signal of UART RXRDY. The *peek* command keyword is followed by an 32-bit address in hexadecimal format and returns the value of the register to the corresponding address. The *poke* command keyword is also followed by an 32-bit address in hexadecimal format accompanied by an additional value with a maximum width of 32-bits in the same format. These commands are entered by the user via a terminal console and are received by the UART core as an array of bytes. The syntax of the commands is the following:

- *peek 0xXXXXXXXX*
  returns the value of the register in the address given
- *poke 0xXXXXXXXX 0xXXXX*
  stores the given value in the address given

During the development of that application, it was found the UART can only receive 16 bytes instead of the 256 bytes as specified in Microsemi's documentation. This behaviour was verified after extensive debugging and communication with Microsemi's support.

4.5.2 Control of Acquisition

This software implementation is a library, which controls the FFT acquisition module. It provides the configuration for the parallel code phase search method. The library configures the FFT size as well as requesting the FFT acquisition results from the hardware design. The library was already implemented in C++ and it was designed to work on an operating system. Part of this thesis was to make the library compatible for bare-metal applications. The processes for reading registers and writing registers of the Fast Acquisition unit, are implemented using functions provided by the drivers of Microsemi for hardware register access.

All the system calls and the exceptions as well as the main call arguments were removed. This library can support the control of the Fast Acquisition unit for all the signals. The only exception is the PRS signal of Galileo, which needs extra timing information from the operating system. For that reason, the bare-metal application needs to acquire the timing information over another way. The communication between the hardware and the software is achieved through specific registers. The specific library can support two modes of the parallel code phase search method on hardware, the fast and the sensitive mode. The challenge of such systems is to speed up the Doppler search. In the fast mode, knowing the size of the FFT and the chip rate, the Doppler bin size can be calculated by $\frac{f_{chip}}{size_{fft}}$. The number of the FFT operations for the Doppler search, is given by $\frac{f_{fft}}{f_{chip}} - 3$. The number three subtracted in the relation is referred to the one FFT calculation of PRN code and the two dummy cycles needed because of the internal latency of the FFT core. The Doppler range achieved for searching can be found by the multiplication of the Doppler bin size and the number of FFTs needed. That range is centered on the intermediate frequency. If the Doppler shift is bigger than the search space, the intermediate frequency is moved in order to change the search space. In the case of the sensitive mode, one Doppler bin is searched each time. The result of the inverted FFT is added with the previous one so the peak will be doubled in case it is found. By this way the peak is easier noticeable. This mode is used for weak signals.

The most important function of the library is the $acquire\_satellite$ function. Before using this function the acquisition module should be initialized, the PRN code is configured and a threshold has to be determined using the corresponding function. The function's inputs are the signal to be acquired, the acquisition mode, and the intermediate frequency. The sensitive mode however needs a hint for the Doppler frequency, the Doppler bin size and the Doppler search width. The output of the function is the hardware timestamp that will be used in other functions, the Doppler frequency of the satellite and the value of the peak that was found. When the function is operating in sensitive mode, or when the signal is the PRS of Galileo, the function outputs the average peak which is the mean value of all FFT peak values used as threshold to find peaks. The function returns true when the acquisition was successful.

4.5.3 Snapshot-PVT Positioning

This part of the software developed for this thesis is a standalone application that implements the snapshot PVT positioning algorithm in C. For the calculation of snapshot-PVT, assistance data is needed. The assistance data used in this program are ephemeris information found in a receiver independent exchange format (RINEX) navigation file. A RINEX file is an ASCII file of GNSS broadcast ephemeris data conforming to the RINEX standard. RINEX navigation files contain position, velocity and clock information for all the satellites of the GNSS constellation. The time system used is in GNSS system time format and positions and velocities are in the ECEF reference format. Even though the RINEX data should be provided by the use of a dedicated interface, for example UART, in this case, we assume that the ephemeris data is already received and stored in the local memory. For the specific software implementation a RINEX filer parser was created. The application, having the list of the visible satellites, the ephemeris data of them, their full pseudoranges and the reception time as its input, starts an iteration loop which implements the least squares algorithm to find the receivers position. The loop can break either after exceeding a specific threshold or after the limit of the iterations is reached. In every iteration the algorithm computes the satellite position and stores it in a lest squares compatible form. The solution calculated is getting more and more accurate in every iteration. For more details about the satellite positioning see section 2.4.2.

Chapter 5

Verification and Results

5.1 Test setup

For development and debugging, the evaluation board MPF300-EVAL-KIT-ES was used. At the image 5.1 there is a picture of the board with all the peripheral devices mentioned.

The setup of the hardware design requires a connection to a digital frontend in order to obtain the input samples of the real and the imaginary part of the signal as well as the input clock of 54 MHz. For the connection of these signals two different banks were used. The clock signal is connected to the Bank 2, using LVCMOS18 I/O standard and the eight bits of the incoming signal vectors were connected to Bank 4, using LVCMOS33 I/O standard. For debugging and for communication through the UART terminal, the evaluation board was connected through the USB-UART terminal to a Windows 7 computer and a PuTTY terminal has to be connected to the corresponding port. The speed, *baud*, of the serial connection was set to 115200 and the flow control has to be set to *None*. Also about the line discipline options, the local echo and the local line editing are set to *Force on*.

The two tools used to program the FPGA are the Libero v12 software and the FlashPro PolarFire v2.3 programmer. The device can be programmed using the on-chip system controller through its dedicated JTAG or SPI interface. Based on the interface used three programming modes are supported: JTAG, SPI master, and SPI slave as it is shown in figure 5.2.

For the debugging of the applications the tool SoftConsole v.6 was used. More details for the specific tool are provided in section 4.5.

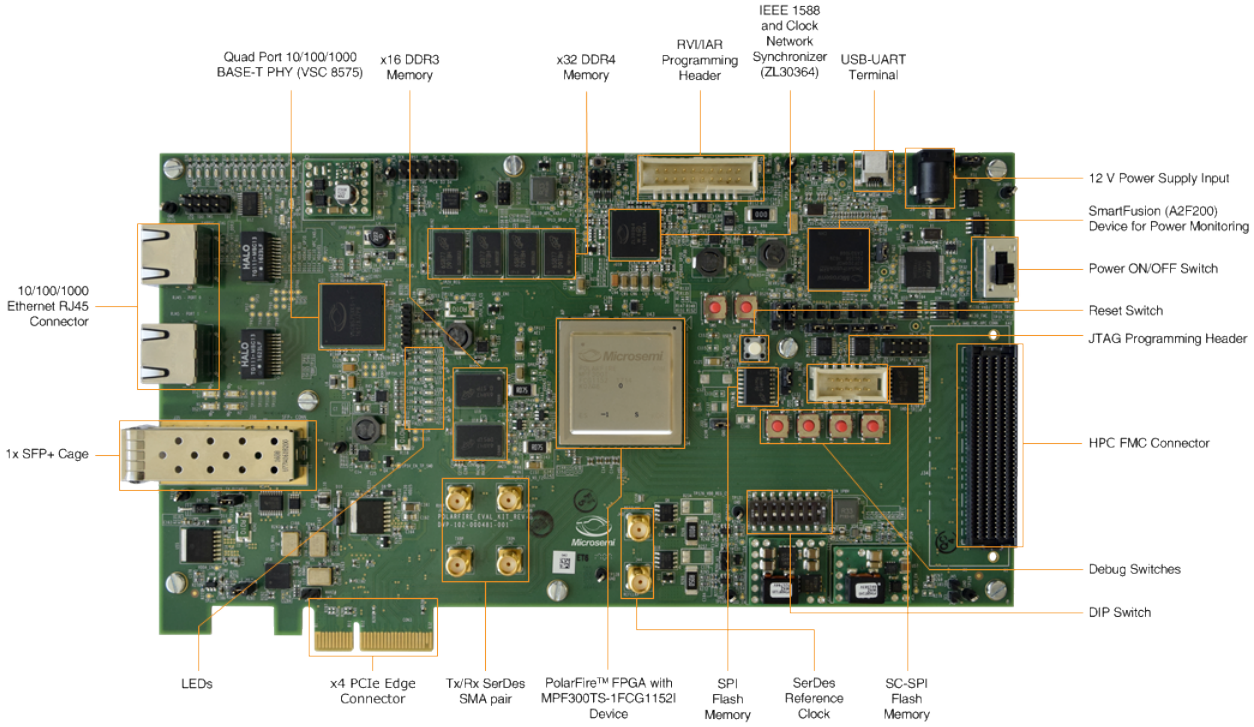

Figure 5.1: Evaluation board

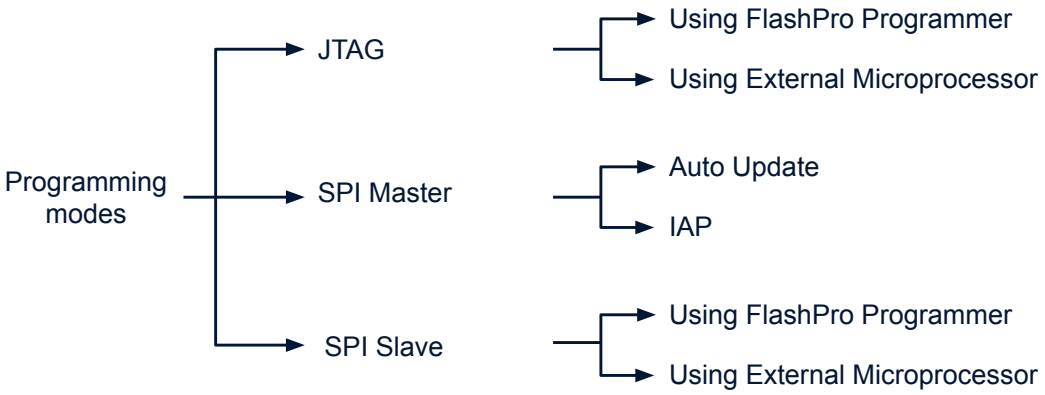

Figure 5.2: PolarFire FPGA Programming Modes

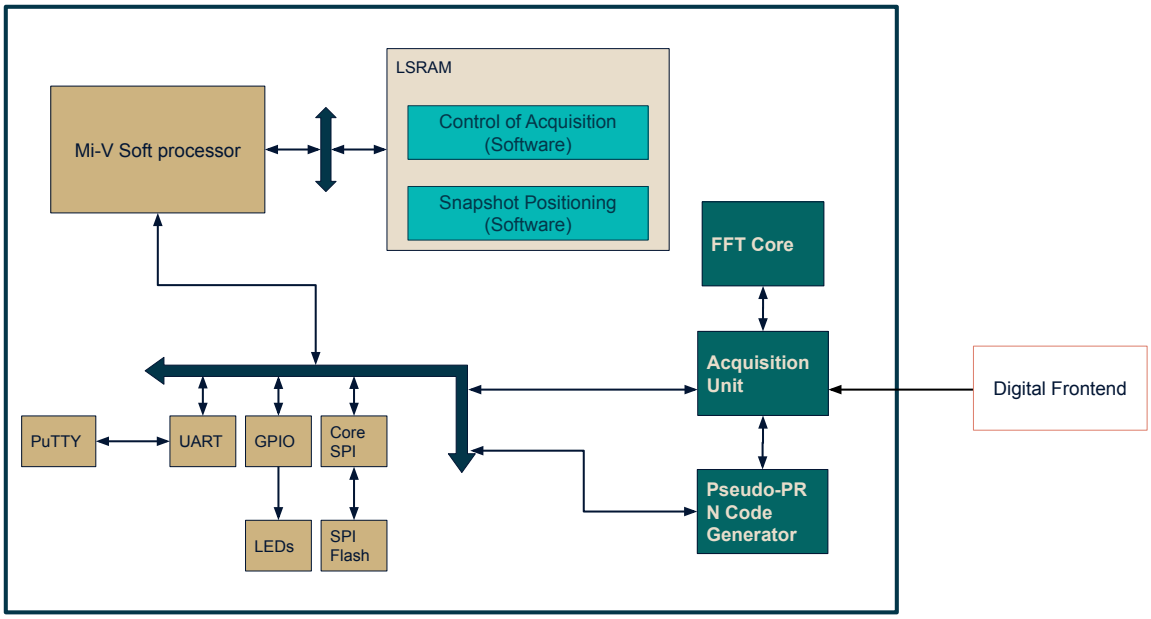

Figure 5.3: Target architecture of the thesis work.

5.2 Verification

The parts of this thesis: the hardware design, the controlling acquisition library and the snapshot positioning application are all complete in principle. The parts are highlighted in figure 5.3. The verification of the IP cores used in the hardware design was beyond the scope of this thesis. According to the integration of the system described already, firstly the verification of the Mi-V processor subsystem was performed. For this task, the software developed for debugging was used and it was possible to read and write values at the registers. Due to memory problems described below, it was not possible to integrate the bare-metal application that controls the acquisition, with the hardware design. Hence, there is no functional verification of the complete hardware design. For the verification of the snapshot algorithm, a reference script in Python was used. Due to memory problems, this application was developed and tested on the Mi-V processor subsystem hardware design. The accuracy of the positioning achieved in the snapshot positioning application was 0.64 m.

5.3 Hardware

5.3.1 Resource usage

The resource usage is shown in the table 5.1 and the detailed logic resource usage is shown in the table 5.2.

Type	Used	Total	Percentage
4LUT	121643	299544	40.61
DFF	104246	299544	34.80
I/O Register	0	1536	0.00
User I/O	22	512	4.30
–Single-ended I/O	22	512	4.30
–Differential I/O Pairs	0	256	0.00
uSRAM	201	2772	7.25
LSRAM	826	952	86.76
Math	168	924	18.18
H-Chip Global	8	48	16.67
PLL	2	8	25.00
DLL	0	8	0.00
UJTAG	1	1	100.00
INIT	1	1	100.00
Transceiver Lanes	0	16	0.00
Transceiver PCIe	0	2	0.00
ICB_CLKINT	3	72	4.17

Table 5.1: Resource usage

Type	4LUT	DFF
Fabric Logic	83447	66050
uSRAM Interface Logic	2412	2412
LSRAM Interface Logic	29736	29736
Math Interface Logic	6048	6048
Total Used	121643	104246

Table 5.2: Detailed resource usage

5.3.2 Limitations

Depending on the frequency of FFT module and the chip rate of the signal, the number of FFT operations for the Doppler search varies. The Doppler bin size can be found by $\frac{chip_{rate}}{fft_{size}}$ and the number of FFT operations can be found by $\frac{f_{fft}}{chip_{rate}} - 3$. The multiplication of the Doppler bin size and the number of FFT operations gives the Doppler search range. The range of the Doppler search is proportional with the frequency of the FFT module. According to the relations mentioned, the higher the frequency of the FFT core is, the bigger the Doppler search space becomes. The

higher frequency achieved in the hardware design for the FFT core is 83.333 MHz. More details are provided on section 4.5.2.

For instance, for the E1A frequency band of the PRS signal of Galileo, the chip rate is $f_{chip} = 2.5575\,\text{MHz}$ and the size of the FFT is $2^{14} = 16384$, that means that the Doppler bin is 156 Hz. If the frequency of the FFT core is $f_{fft} = 83\,\text{MHz}$, then the number of the FFT calculated will be 29. With the parameters mentioned above the achieved Doppler search range is 4.5 kHz. If the intermediate frequency is 0 Hz, the range of search would be from −2.25 kHz to 2.25 kHz.

For each frequency band, the chip rate is different and therefore, the Doppler bin size and the Doppler range is changing. The table 5.3 shows the differences in each case.

Frequency band	Chip rate	Doppler bin size	Doppler search range
E1A	2.5575 MHz	156 Hz	4.5 kHz
E6A	5.115 MHz	312 Hz	4 kHz

Table 5.3: Doppler bin size and Doppler range for each frequency band

The typical Doppler search range is ±5 kHz. To achieve 10 kHz search range, the frequency at the FFT core has to be higher. Depending on the frequency band the needed frequency for the FFT core can be seen in table 3.1. Frequency band E1B has repeating PRN codes so it is not possible to exploit the whole FFT size. In this case we have to limit the FFT size to the size of the FFT code length. The code length is 4092 therefore a 4K FFT is used. Subsequently the Doppler bin size is 250 Hz and the Doppler search range is 19.5 kHz which means in this case the Doppler search range is covered.

5.4 Software

The algorithm of the PVT positioning followed a reference implementation in Python, done by Fraunhofer. The application implemented for this thesis, has an accuracy of *double* floating point for the input data of the RINEX file and for all the operations that take place in the algorithm. The number of visible satellites used for the positioning is eleven. The iteration loop for the receiver positioning, has a limit of hundred iterations and the threshold used is $1e-3$. The accuracy that has been achieved is 0.64 m, whereas the accuracy achieved by the reference Python implementation is 0.0023 m.

As declared in linker script section the memory length for allocation is 1 MB and the length of the stack and the heap size is 128 kB each. According to the results

of building, the application needs for text segment, which contains the executable instructions 58 kB. For the data segment, which contains the global variables and the static variables 2 kB are used. The length of the *bss* segment, which is often called uninitialized data segment, is 263 kB. In total the bytes needed for the application are 323 kB.

For the library and the corresponding bare-metal application controlling the acquisition, 500 kB of RAM were allocated. For each the heap and the stack 2 kB were allocated. According to the build report, The size of the text segment is 466 kB, the size of the data segment is 4 kB and the size of uninitialized data segment is 11 kB. In total the bytes needed for the library are 481 kB.

Subsequently, the memory in total needed for the software part of this thesis is about 804 kB since the values have been rounded.

5.5 Explanation of the results

As it is mentioned previously, due to the lack of memory it was not possible to integrate the software parts of this thesis and the hardware design, which is described in section 4.2. According to the reports, the LSRAM has 952 blocks of 20 kbit, which means the available memory is 2.38 MB and the 826 of them are used. Subsequently, the free memory is 315 kB. To integrate both, the control of acquisition and the snapshot positioning application in the FPGA extra 489 kB are needed. For that reason, the snapshot positioning application was developed and tested on the Mi-V processor subsystem and not in the main hardware design of this thesis. The reason the accuracy is not as high as the result of the reference script, is because of the lack of floating point unit in the hardware implementation. By adding the floating point unit to the hardware design the accuracy will change noticeably. Also, it was not possible to store the RINEX navigation file on the evaluation board. As a result, the RINEX parser was not used, and the ephemeris data was stored in a constant C-structure in the application. Another problem that came up during the implementation is that the registers used for the communication of the hardware and software were not accessible, due to problems that the tools used for the hardware development cannot detect.

Chapter 6

Conclusion

6.1 Summary

The purpose of this thesis is to implement snapshot positioning on a Microsemi PolarFire system on chip.

The hardware design of the thesis implements the parallel code phase acquisition method. The design includes cores from the Fraunhofer institute and Microsemi. It was developed on the Libero v.12 tool of Microsemi, and tested on the MPF300-EVAL-KIT-ES evaluation board, including the MPF300TS device. The design is based on a Mi-V processor subsystem, which communicates with the modules through an AXI protocol. The most important parts of it are the PRN code generator, the FFT unit and the Fast Acquisition unit. The Fast Acquisition unit receives a 4-bit vector for the real part of the signal and a 4-bit vector for the imaginary part of the signal from a digital frontend and the PRN sequence generated by the corresponding module as an input. The Fast Acquisition module provides the input to the FFT core for the FFT computations.

The software part includes a library that controls the Fast Acquisition unit and a snapshot PVT application calibrating the position of the receiver and the error range of the position. Both parts are based on applications developed by the Fraunhofer institute. The library that controls the Fast Acquisıtion unit is in C++ and it had to be changed in order to be compatible with stand-alone applications. For the snapshot PVT application, there was a reference script on Python. The application was developed in C. The position error range achieved was 0.64 m without the use of floating point unit in hardware. For all the software parts of this thesis the tool SoftConsole of Microsemi was used.

Because of memory lack and error in the design flow of the software provided by Microsemi a complete hardware design of this thesis and for that reason a

minimized design was used for debugging.

6.2 Future work

The work carried out in this thesis will be implemented on a larger Polarfire FPGA. This allows to overcome the limitations encountered in this work. The device will provide enough on-chip memory to accommodate both the requirements of the local buffers of the FFT acquisition unit and the size of the SW stack/heap needed by the PVT calculation. In addition, a better routing of the resource might also allow a higher operating frequency, which can have an impact on the acquisition performance. A dedicated channel will also be used to transmit the ephemeris data in real time using the RINEX protocol. This way there won't be any constraint posed on the local memory depending on the amount of satellite signals to be processed.

Fraunhofer IIS intends to integrate PVT-snapshot FPGA design in one of its receiver platforms. The samples can be provided by an additional FPGA or ASIC, where downconversion, filtering and additional interference mitigation techniques are performed. The usage of the Polarfire device will allow to comply with higher security requirements and harder low power constraints.

$\alpha = \left(\sqrt{\alpha}\right)^2$	**Semimajor axis**
$n = \frac{\sqrt{\mu}}{\sqrt{\alpha^3}} + \Delta_n$	**Corrected mean motion**
$t_k = t - t_{0_e}$	**Time from ephemeris epoch**
$M_k = M_o + nt_k$	**Mean anomaly**
$E_k = M_k + \boldsymbol{e} \sin E_k$	**Eccentric anomaly**
$v_k = \arctan\left(\frac{\sqrt{1-\boldsymbol{e}^2}\sin E_k}{\cos E_k - \boldsymbol{e}}\right)$	**True anomaly**
$\phi_k = v_k + \omega$	**Argument of latitude**
$\delta_{\phi_k} = C_{us}\sin(2\phi_k) + C_{uc}\cos(2\phi_k)$	**Argument of latitude correction**
$\delta_{r_k} = C_{rs}\sin(2\phi_k) + C_{rc}\cos(2\phi_k)$	**Radius correction**
$\delta_{i_k} = C_{is}\sin(2\phi_k) + C_{ic}\cos(2\phi_k)$	**Inclination correction**
$u_k = \phi_k + \delta_{\phi_k}$	**Corrected argument of latitude**
$r_k = \alpha(1 - \boldsymbol{e}\cos E_k) + \delta_{r_k}$	**Corrected radius**
$i_k = i_o + (d_i/d_t)t_k + \delta_{i_k}$	**Corrected inclination**
$\Omega_k = \Omega_o + \left(\dot{\Omega} - \dot{\Omega}_e\right)(t_k) - \dot{\Omega}_e t_{o_e}$	**Corrected longitude of node**
$x_p = r_k \cos u_k$	**In-plane x position**
$y_p = r_k \sin u_k$	**In-plane y position**
$x_s = x_p\cos\Omega_k - y_p\cos i_k\sin\Omega_k$	**ECEF x-coordinate**
$y_s = x_p\sin\Omega_k + y_p\cos i_k\cos\Omega_k$	**ECEF y-coordinate**
$z_s = y_p \sin i_k$	**ECEF z-coordinate**

Acronyms

A-GPS assisted-GPS. 31

CDMA code division multiple access. 4

CLD Carrier Loop Discriminator. 8

CLF carrier loop filter. 8

CS Commercial Service. 1

ECEF Earth-centered and Earth-fixed. 11–13, 15, 49

ECI Earth-centered intertial. 11

FEC forward error correction. 10

FFT Fast Fourier Transform. 7, 8, 35, 42, 44, 47, 48, 53, 54, 56

FLL frequency lock loop. 8, 9

FTP file transfer protocol. 28

GNSS Global Navigation Satellite System. 1, 4, 7, 22, 23, 25, 28, 30–33, 35, 49

GPS Global Positioning System. 1, 4, 6, 28, 29, 31–33

IDE integrated development environment. 47

IF Intermediate Frequency. 5–7, 31, 32

ISA instruction set architecture. 38

LSRAM large static RAM. 41

MP master ports. 40

NCO Numerically Controlled Oscillator. 9

OS Open Service. 1, 25

PLIC platform-level interrupt controller. 38

PLL phase lock loop. 8, 9

PPS Precise Positioning Service. 1

PRN pseudo-random noise. 4–6, 8, 9, 22, 24, 27, 42, 44, 48, 56

PRS Public Regulated Service. 1, 2, 24–28, 30, 48, 54

PVT position, velocity and time. 2, 4, 8, 22, 25, 27, 32, 35, 49, 56

RAAN right ascension of the ascending node. 13

RINEX receiver independent exchange format. 49, 54, 55

RISC reduced instruction set computer. 38

RO ring oscillator. 34

SDR software defined receiver. 24, 32

SIS signals in space. 4

SNR signal-to-noise ratio. 24

SP slave ports. 40

SPI serial peripheral interface. 41

UKSA UK space agency. 27

Bibliography

[1] Frank van Diggelen. *A-GPS Assisted GPS,GNSS and SBAS*. ARTECH HOUSE, 2009.

[2] Elliot D. Kaplan, Christopher J. Hegarty. *Understanding GPS Principles and Applications*. ARTECH HOUSE, 2006.

[3] Peter J.G. Teunissen, Oliver Montenbruck. *Springer Handbook of Global Navigation Satellite Systems*. Springer, 2017.

[4] Kai Borre, Dennis M. Akos, Nikolaj Bertelsen, Peter Rinder, Soren Holdt Jensen. *A Software-Defined GPS and Galileo Receiver - A signle frequency approach* Birkhäuser, 2007.

[5] Alexander Rügamer, Daniel Rubino, Ivana Lukčin, Simon Taschke, Manuel Stahl, Wolfgang Felber, Fraunhofer Institute for Integrated Circuits IIS, Nuremberg, Germany. *Secure Position and Time Information by Server Side PRS Snapshot Processing* ION GNSS 2016

[6] M. Turner, A. Chambers, E. Mak, Astrium Ltd L.E. Aguado, B. Wales, M. Dumville, NSL *PROSPA: Open Service Authentication* ION GNSS+ 2013

[7] Sherman Lo, David De Lorenzo and Per Enge from Stanford University and Zanio, Inc. Dennis Akos from University of Colorado Paul Bradley from DAFCA, Inc. *Signal Authentication - A Secure Civil GNSS for Today* InsideGNSS September/October 2009

[8] Jinghui Wu, Andrew G. Dempster, School of Surveying and Spatial Information Systems, University of New South Wales *Data Compression for Assisted - GPS Signal Processing* ION GNSS 2011

[9] Logan Scott, LS Consulting *Proving Location Using GPS Location Signatures: Why it is Needed and a Way to Do It* ION GNSS+ 2013

[10] The ULTRA project is co-funded by the European Union's Seventh *Ultra Low-Cost PRS Receiver* https://www.gsa.europa.eu/ultra-low-cost-prs-receiver

[11] Ben Wales from Bavaro Nottingham Scientific Ltd, Luis Tarazona from Department of Engineering Faculty of Technology De Montford University, Michele Bavaro from Bavaro Nottingham Scientific Ltd. *Snapshot positioning for low-power miniaturised spaceborne GNSS receivers* 5th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)2010

[12] Thuan Nguyen Dinh, Vinh La The from NAVIS Centre Hanoi University Of Science and Technology Vietnam *A Novel Design of Low Power Consumption GPS Positioning Solution Based on Snapshot Technique* International Conference on Advanced Technologies for Communications 2017

[13] S. V. Shafran, E.A. Gizatulova, L.A. Kudryavtsev from Samara National Research University, Russia *SNAPSHOT TECHNOLOGY IN GNSS RECEIVERS* 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS) 2018

[14] Daniel Örn, Martin Szilassy, Bram Dil, Fredrik Gustafsson from Department of Electrical Engineering, Linköping University, Link¨oping, Sweden *A Novel Multi-Step Algorithm for Low-Energy Positioning Using GPS* 19th International Conference on Information Fusion (FUSION) 2016

[15] Dominik Dötterböck and Bernd Eissfeller Institute of Geodesy and Navigation University FAF Munich *A GPS/Galileo Software Snap-Shot Receiver for Mobile Phones* 2019

[16] Amir Moradi, Markus Kasper, Christof Paar *On the Portability of Side-Channel Attacks* 2011

[17] Mark Zhao and G. Edward Suh from Computer Systems Laboratory, Cornell University Ithaca, New York *FPGA-Based Remote Power Side-Channel Attacks* IEEE Symposium on Security and Privacy 2018

[18] Handbook of Microsemi. *User Guide: PolarFire FPGA Programming*

[19] Handbook of Microsemi. *User Guide: PolarFire FPGA Design Flow*

[20] Handbook of Microsemi. *MiV RV32IMA L1 AHB v2.1*

[21] Handbook of Microsemi. *CoreAXI4Interconnect v2.5*

[22] Handbook of Microsemi. *CoreUART v5.6*

[23] User Guide of Microsemi. *PolarFire FPGA User I/O*

[24] User Guide of Microsemi. *PolarFire FPGA Evaluation Kit*

[25] Release Notes of Microsemi *Microsemi SoftConsole v6.0*

[26] Handbook of Microsemi. *Product Overview: PolarFire FPGA*

[27] Werner Gurtner, Astronomical Institute, University of Bern. *The Receiver Independent Exchange Format*

[28] Andrew Waterman, Krste Asanović *The RISC-V Instruction Set Manual: Volume I: User-Level ISA*

[29] Tutorial of Microsemi. *PolarFire FPGA: Building a Mi-V Processor Subsystem*

[30] Mark Petovello *What Is Snapshot Positioning and What Advantages Does It Offer?* https://insidegnss.com/what-is-snapshot-positioning-and-what-advantages-does-it-offer/