TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Tommi Paakki

Next Generation Multi-System Multi-Frequency GNSS Receivers

Julkaisu 1488 • Publication 1488

Tampere 2017

Tampereen teknillinen yliopisto. Julkaisu 1488
Tampere University of Technology. Publication 1488

Tommi Paakki

Next Generation Multi-System Multi-Frequency GNSS Receivers

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB109, at Tampere University of Technology, on the 1st of September 2017, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2017

Doctoral candidate:	Tommi Paakki Laboratory of Electronics and Communications Engineering Computing and Electrical Engineering Tampere University of Technology Finland
Supervisor:	Jari Nurmi, Dr. Tech, Professor Laboratory of Electronics and Communications Engineering Computing and Electrical Engineering Tampere University of Technology Finland
Pre-examiners:	Marco Lisi, Dottore Ingegnere European Space Research & Technology Centre (ESTEC) The Netherlands Jari Syrjärinne, D.Sc.(Tech) HERE Technologies Finland
Opponents:	Terry Moore, Ph.D., Professor Nottingham Geospatial Institute University of Nottingham The United Kingdom Jari Syrjärinne, D.Sc.(Tech) HERE Technologies Finland

ISBN 978-952-15-3987-9 (printed)
ISBN 978-952-15-4004-2 (PDF)
ISSN 1459-2045

ABSTRACT

Nowadays we have satellites available from GPS, GLONASS, Galileo and BeiDou systems. This will lead to an increased demand for solutions, which utilize multiple Global Navigation Satellite Systems (GNSS). Such solutions can have great market potential since they can be applied in numerous applications involving GNSS navigation, e.g. smartphones and car navigators. The aim of this thesis is to present the issues that arise in modern high sensitivity receivers, and to present research results of navigation algorithms suitable for the next generation multi-system multi-frequency GNSS receivers.

With the availability of multiple satellites systems, the user benefits mostly from the improved visibility of the satellites. The increased availability of satellites naturally increases the computational requirements in the receiver. The main focus of the presented algorithms is on critical factors like provided accuracy versus low cost, low power consumption. In addition, the presented algorithms have been collected into a comprehensive navigation algorithm library where they have additional value for educational purposes.

The presented navigation algorithms focus mainly in the GPS and Galileo systems, with the combination of L1/E1 & L5/E5a frequencies. A novel GPS + Galileo dual frequency receiver was developed by the team over the years. Where applicable, the thesis collects important facts from modern GLONASS and BeiDou systems.

The first part of the thesis introduces all available open service signals from the GNSS systems, revealing how vast the scope of multi-system, multi-frequency receiver design is. The chapter continues with introduction to the basics of GNSS systems, and description of the problems that the receiver designer must overcome. The chapter further continues by describing a basic receiver architecture suitable for multi-system multi-frequency reception. The introductory part also has a short section is dedicated for underlining the importance of testing mechanisms for a novel receiver under development.

The second part of the thesis concentrates on the baseband processing of the GNSS receiver. Topics cover acquisition and tracking, with multi-system multi-frequency implementation details kept in mind. The chapter also contains sections for issues that must be handled in high sensitivity receivers, e.g. cross-correlation and cycle slip detection. The second part of the thesis is concluded with a description how Assisted-GNSS capability would alter many of the design considerations.

The third part of the thesis describes algorithms related to the data bit decoding issues. All the different satellite systems have their own low-level navigation data structure with additional layers of error detection / correction mechanisms. This part of the thesis provides the algorithms for successful decoding of the data.

The final part of the thesis describes the basic navigation solution algorithms suitable for the mass-market receivers. In this part, the method of combining the measurements from the different satellite systems is discussed. Additionally, all the issues of processing multi-system signals are collected here, and in the end the Position, Velocity, and Time (PVT) solution is obtained.

PREFACE

The work presented in this thesis has been carried out in the Laboratory of Electronics and Communications Engineering, and its predecessors, at Tampere University of Technology during the years 2010-2016.

I am grateful to my supervisor Prof. Jari Nurmi, who has been very supportive and encouraging over these years. I was very fortunate to get involved in the GNSS project as early as year 2006. In these early years, he guided me through the M. Sc. degree, and eventually succeeded in convincing me to apply for the Ph. D degree. It has been an exciting 10 years of GNSS business that I have been privileged to take part.

I would also like to give my special thanks to the people with whom we made excellent GNSS products and who shared their ideas and knowledge. These people who especially helped me during the research career are: Dr. Heikki Hurskainen, Jussi Raasakka, Dr. Sarang Thombre, Francescantonio Della Rosa and Ondrej Daniel. Likewise, all the members of Team Nurmi are thanked for the wonderful time spent together.

The research work has received financial support from the Tampere University of Technology, Tampere Doctoral Programme in Information Science and Engineering (TISE), Doctoral training network in Electronics, Telecommunications and Automation (DELTA) and Elisa HPY research Foundation. Part of research has been conducted in a Finnish Funding Agency for Technology and Innovation (TEKES) funded project: Future GNSS Applications and Techniques (FUGAT). Work has also involved European Union Framework Programme (EU FP7) funded project, Galileo Ready Advanced Mass Market Receiver (GRAMMAR).

I would like to thank Dr. Marco Lisi and Dr. Jari Syrjärinne who have been the external reviewers for this thesis. Their review feedback has been valuable and constructive. Further thanks go for Prof. Terry Moore and Dr. Jari Syrjärinne who agreed to be the opponents for my Doctoral dissertation.

Finally, thanks to my family and friend, for all your support during these years!

Tampere, August 2017

Tommi Paakki

TABLE OF CONTENTS

LIST OF PUBLICATIONS

This thesis is a compilation of the following publications, referred to as [P#] and are appended in the concluding half of the thesis.

[P1] **T. Paakki**, J. Raasakka, F. Della Rosa, H. Hurskainen, and J. Nurmi, "TUTGNSS – University Based Hardware/Software GNSS Receiver for Research Purposes", in Proceedings of the *Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS 2010)*, October 2010, in Helsinki (Kirkkonummi), Finland.

[P2] **T. Paakki**, F. Della Rosa, S. Thombre, and J. Nurmi, "Profiling of GNSS Receiver Navigation Software on Embedded Processor", in Proceedings of the *International Conference on Localization and GNSS (ICL-GNSS 2012)*, June 2012, in Starnberg, Germany.

[P3] **T. Paakki**, J. Raasakka, F. Della Rosa, and J. Nurmi, "Efficient Bit Decoding Implementation for Mass Market Multi-Constellation GNSS Receivers", in Proceedings of the *Design & Architectures for Signal & Image Processing (DASIP 2013)*, October 2013, in Cagliari, Italy.

[P4] **T. Paakki**, F. Della Rosa, H. Hurskainen, and J. Nurmi, "Navigation Algorithm Test Environment for GNSS Receivers", in Proceedings of the *European Navigation Conference (ENC GNSS 2010)*, October 2010, in Braunschweig, Germany.

[P5] **T. Paakki**, and J. Nurmi, "Faster Than Real-Time GNSS Receiver Testing", in Proceedings of the *International Conference on Localization and GNSS (ICL-GNSS 2014)*, June 2014, In Helsinki, Finland.

[P6] **T. Paakki**, F. Della Rosa, and J. Nurmi, "Stand-Alone GNSS Time Synchronization Architecture", in Proceedings of the *European Workshop on GNSS Signals and Signal Processing (NAVITEC 2014)*, December 2014, In Noordwijk, The Netherlands.

LIST OF ABBREVIATIONS

ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
A-GNSS	Assisted-GNSS
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BOC	Binary Offset Carrier
BPF	Band Pass Filter
BW	Bandwidth
C/A	Coarse Acquisition
CDMA	Code Division Multiple Access
CN0	Carrier-to-Noise density
CORDIC	Coordinate Rotation Digital Computer
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DFT	Discrete Fourier Transform
DLL	Delay Locked Loop
DOP	Dilution of Precision
ECEF	Earth Centered Earth Fixed
ECI	Earth Centered Inertial
EGNOS	European Geostationary Navigation Overlay System
EKF	Extended Kalman Filter
EPL	Early, Prompt, Late
ESA	European Space Agency
ESD	Electrostatic Discharge
FAA	Federal Aviation Administration
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FFT	Fast Fourier Transform

FLL	Frequency Locked Loop
GAGAN	GPS Aided Geo Augmented Navigation
GDOP	Geometrical Dilution of Precision
GEO	Geosynchronous Satellite Orbit
GGTO	Galileo/GPS Time Offset
GLONASS	Globalnaya Navigatsionnaya Sputnikovaya Sistema
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPST	GPS Time
GST	Galileo System Time
GTRF	Galileo Terrestrial Reference Frame
HW	Hardware
ICD	Interface Control Document
IF	Intermediate Frequency
IGSO	Inclined Geosynchronous Satellite Orbit
IODC	Issue of Data Clock
IODE	Issue of Data Ephemeris
IRNSS	Indian Regional Navigation Satellite System
ITRF	International Terrestrial Reference Frame
KF	Kalman Filter
LFSR	Linear-Feedback Shift Register
LHCP	Left Hand Circularly Polarized
LMS	Land Mobile Satellite
LNA	Low Noise Amplifier
LO	Local Oscillator
LS	Least Squares
MEO	Medium Earth Orbit
MS	Mobile Station
MSAS	Multi-functional Satellite Augmentation System
NCO	Numerically Controlled Oscillator
PLL	Phase locked Loop

PND	Personal Navigation Device
PPM	Part Per Million
PPP	Precise Point Positioning
PRN	Pseudo Random Noise
PVT	Position Velocity Time
QZSS	Japanese Quasi-Zenith Satellite System
RF FE	Radio Frequency Front End
RHCP	Right Hand Circularly Polarized
RMS	Root Mean Square
RTC	Real Time Clock
RTK	Real Time Kinematic
SBAS	Satellite Based Augmentation System
SDCM	System for Differential Corrections and Monitoring
SDR	Software Defined Radio
SISA	Signal in Space Accuracy
SPD	Symmetric Positive Definite
TEC	Total Electron Content
ToA	Time-of-Arrival
TOW	Time-Of-Week
TTFF	Time to First Fix
UERE	User Equivalent Range Error
URA	User Range Accuracy
VE	Very Early
VL	Very Late
WAAS	Wide Area Augmentation System
WN	Week Number

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In these days several Global Navigation Satellite Systems (GNSS) are rapidly being developed. The current fully operational GNSS systems are the American NAVSTAR Global Positioning System (GPS) [1] and the Russian Globalnaya Navigatsionnaya Sputnikovaya Sistema (GLONASS) [2]. In the deployment phase towards the full operational capability are the European Galileo [3] and the Chinese BeiDou systems [4]. In the end of year 2016, an official statement was given that Galileo is now in Initial Services phase [5]. Galileo and BeiDou are expected to reach full completion of their constellation by year 2020.

Meanwhile, also GPS is getting its infrastructure modernized by replacing satellites with next generation models providing new signals. Similarly, GLONASS system has plans to modernize its satellites by adding a new frequency, and providing Code Division Multiple Access (CDMA) capability. As a result, the GNSS users on Earth will have a wide selection of satellites in the sky. The availability of satellites will thus improve dramatically in the coming years. This will be especially useful in the areas where the environment sets limitations to the visibility. The most common case is urban canyons in the cities, where tall buildings block the visibility.

Some of the new satellite navigation systems are planned to be regional systems. The regional satellite system will provide service to certain geographical location on Earth, further increasing the availability of satellites in these regions. One such a regional system is the Indian Regional Navigation Satellite System (IRNSS), and another one is the Japanese Quasi-Zenith Satellite System (QZSS).

There is also a third category of satellite systems, which provide information signals to the users. Satellite based augmentation systems (SBAS) consist of geostationary satellites offering regional augmentation through their signals. These systems are used for improving the positioning accuracy of the typical GNSS receiver. A list of the SBAS systems include: the United States Federal Aviation Administration (FAA) operated Wide Area

Augmentation System (WAAS), and the European Space Agency (ESA) operated European Geostationary Navigation Overlay System (EGNOS), Russian System for Differential Corrections and Monitoring (SDCM), Japanese Multi-functional Satellite Augmentation System (MSAS) and Indian GPS Aided Geo Augmented Navigation (GAGAN).

Tables 1.1 – 1.9 collect the characteristics of the open service signals available from the GNSS systems for the future mass-market receivers [6]. These tables also immediately define the vast scope of this thesis: the complex design of a multi-system, multi-frequency receiver.

Table 1.1 GPS L1 C/A signal characteristics

Carrier frequency	L1: 1575.42 MHz = 154 * 10.23 MHz
Minimum received power	-158.5 dBW
Multiple access	CDMA
Spreading codes	Length 1023 bit Gold codes, duration 1 ms
Data rate | message structure	50 bps | NAV
Data message error correction and detection	Error detection using extended (32,26) Hamming code
Data modulation	50 sps biphase modulation
Pilot and data components	100% power data
Overlay code	None

Table 1.2 GPS L1C signal characteristics

Carrier frequency	L1: 1575.42 MHz = 154 * 10.23 MHz
Minimum received power	-157 dBW
Multiple access	CDMA
Spreading codes	Length 10230 bit Weil-based codes, duration 10 ms
Data rate | message structure	50 bps | CNAV2
Data message error correction and detection	½-rate LDPC coding, block interleaving, and 24-bit CRC
Data modulation	100 sps biphase modulation
Pilot and data components	75% power pilot, 25% power data, in same carrier phase
Overlay code	1800 bit pilot overlay codes, 100 bps, different for each satellite; segments of m-sequences and Gold codes

Table 1.3 GPS L2C signal characteristics

Carrier frequency	L2: 1227.60 MHz = 120 * 10.23 MHz
Minimum received power	Current satellites: -160 dBW. Next generation satellites: -158.5 dBW
Multiple access	CDMA
Spreading codes	M-sequences from a maximal polynomial of degree 27. Pilot component: Length 767250-bit L2CL code, duration 1.5 seconds. Data component: Length 10230-bit L2CM code, duration 20 ms
Data rate | message structure	25 bps | CNAV
Data message error correction and detection	½-rate convolutional coding with constraint length 7, and 24-bit CRC
Data modulation	50 sps biphase modulation
Pilot and data components	50% power pilot, 50% power data, time multiplexed spreading symbol by spreading symbol
Overlay code	None

Table 1.4 GPS L5 signal characteristics

Carrier frequency	L5: 1176.45 MHz = 115 * 10.23 MHz
Minimum received power	Current satellites: -157.9 dBW. Next generation satellites: -157 dBW.
Multiple access	CDMA
Spreading codes	Combination and short cycling of M-sequences. Pilot component: L5Q, Length 10230-bit, duration 1 ms. Data component: L5I, Length 10230-bit, duration 1 ms.
Data rate | message structure	50 bps | CNAV
Data message error correction and detection	½-rate convolutional coding with constraint length 7, and 24-bit CRC
Data modulation	100 sps biphase modulation
Pilot and data components	50% power pilot, 50% power data, phase quadrature multiplexed with pilot component on the quadrature-phase component.
Overlay code	10 bit, 1 kbps Neumann-Hoffman code on L5I. 20 bit, 1 kbps Neumann-Hoffman code on L5Q

Table 1.5 Galileo E1 signal characteristics

Carrier frequency	E1: 1575.42 MHz = 154 * 10.23 MHz
Minimum received power	-157 dBW
Multiple access	CDMA
Spreading codes	Length 4092 memory codes, different for pilot E1-C and data E1-B components, duration 4 ms.
Data rate | message structure	125 bps | I/NAV
Data message error correction and detection	½ -rate convolutional coding with constraint length 7, block interleaving, CRC
Data modulation	250 sps biphase modulation
Pilot and data components	50% power pilot, 50% power data
Overlay code	None on data E1-B. Length 25 sequence on pilot E1-C for tiered code, total duration 100 ms.

Table 1.6 Galileo E5a and E5b signal characteristics

Carrier frequency	E5a: 1176.45 MHz = 115 * 10.23 MHz E5b: 1207.14 MHz = 118 * 10.23 MHz
Minimum received power	-155 dBW
Multiple access	CDMA
Spreading codes	10230-bit codes, different for pilot and data components, duration 1 ms.
Data rate | message structure	E5a-I: 25 bps | F/NAV. E5b-I: 125 bps | I/NAV
Data message error correction and detection	½ -rate convolutional coding with constraint length 7, block interleaving, CRC
Data modulation	E5a-I: 50 sps biphase modulation. E5b-I: 250 sps biphase modulation.
Pilot and data components	50% power pilot, 50% power data
Overlay code	Length 20 sequence on data E5a-I for tiered code, total duration 20 ms. Length 100 sequence on pilot E5a-Q for tiered code, total duration 100 ms. Length 4 sequence on data E5b-I for tiered code, total duration 4 ms. Length 100 sequence on pilot E5b-Q for tiered code, total duration 100 ms.

Table 1.7 GLONASS L1OF and L2OF signal characteristics

Carrier frequency	L1OF: 1598.0625 – 1605.375 MHz, spaced by 0.5625 MHz. L2OF: 1242.9375-1248.625 MHz, spaced by 0.4375 MHz.
Minimum received power	L1OF: -161 dBW. L2OF: -167 dBW.
Multiple access	FDMA
Spreading codes	Same 511 bit m-sequence used for all signals
Data rate | message structure	50 bps | GLONASS
Data message error correction and detection	Error detection using extended (85, 81) hamming code.
Data modulation	50 sps biphase modulation.
Pilot and data components	100% power data
Overlay code	Meander sequence at 1 kbps

Table 1.8 GLONASS L3OC signal characteristics

Carrier frequency	L3: 1202.025 MHz = 1175 * 1.023 MHz.
Minimum received power	-158 dBW
Multiple access	CDMA
Spreading codes	Length 10230 truncated Kasami sequences, duration 1 ms.
Data rate | message structure	100 bps | Modernized GLONASS
Data message error correction and detection	½-rate convolutional coding
Data modulation	200 sps biphase modulation.
Pilot and data components	50% power pilot, 50% power pilot
Overlay code	5 bit, 1 kbps Barker code on data component, 10 bit, 1 kbps Neumann-Hoffman code on pilot component.

Table 1.9 BeiDou B1I and B2I signal characteristics

Carrier frequency	B1: 1561.098 MHz = 1526 * 1.023 MHz B2: 1207.14 MHz = 1180 * 1.023 MHz
Minimum received power	-163 dBW
Multiple access	CDMA
Spreading codes	Length 2046 bit segments of 2047 length Gold codes, duration 1 ms.
Data rate | message structure	MEO and IGSO satellites: approximately 37 bps | BeiDou D1 NAV. GEO satellites: approximately 367 bps | BeiDou D2 NAV
Data message error correction and detection	BCH(15,11) code with block interleaving over two blocks.
Data modulation	MEO and IGSO satellites: 50 sps biphase modulation. GEO satellites: 500 sps biphase modulation.
Pilot and data components	100% power data.
Overlay code	MEO and IGSO satellites: same 1 kbps overlay Neumann-Hoffman code. GEO satellites: None

1.1 Background and Motivation

The introduction to the three available categories of satellite-based systems should highlight the challenge in the next generation GNSS receivers. With so many satellite systems available, in the near future the design of the GNSS receivers gets more complicated. The receivers must get ready to utilize these new signals from the moment they become available. From the receiver point of view, this will mean several design issues:

- Antenna - Must handle various frequencies and different bandwidths of the spectrum.
- Radio Frequency Front End (RF FE) – How to combine the various frequencies, and how to design the filtering with bandwidth (BW) and group delays in mind? What are the Analog to Digital Converter (ADC) requirements?
- Baseband design – Capability of doing the basic acquisition and tracking operations for multiple constellations/signals. Multiple Pseudo Random Noise (PRN) codes available for each system.

- Software (SW) design. – Capability of decoding the navigation data from various systems, and capability of producing a multi-constellation Position, Velocity and Time (PVT) solution.

All the listed features can be implemented on a receiver, but it must be understood that the addition of these new features will come with a cost in form of more expensive hardware (HW), increased power consumption, or increase in computational requirements. The challenge in designing the next generation GNSS receiver lies in both making these capabilities available, and keeping the costs as low as possible. The aim of this study is to conduct research on advanced algorithms best suitable for next generation GNSS receivers.

In the world of hand-held mass-market receivers, it is very important that the new capabilities do not drain the device battery too quickly, and that the cost of the device is still lucrative. For these mass-market receivers it remains to be seen if all the available systems would be supported by a single device. For example, many current smartphones have the support for GPS + GLONASS, effectively doubling the number of satellites in view, and consequently considerably improving positioning in scenarios like deep urban, where tall buildings would block the visibility to most satellites. Additional satellite systems further improve the number of satellites available. Implementing multi-system capability naturally implies increased cost / resource consumption in the device [7]. It will be interesting to see how the markets will develop, once the satellite systems reach their full capabilities in the years to come.

As the complexity of the receiver increases with the multi-system, multi-frequency capabilities, the importance of assisting the receiver externally grows. Many hand-held mass-market devices, especially smartphones, come with the Assisted-GNSS capability (A-GNSS), which helps the receiver to save computational resources, and still achieve quicker first fixes and find weaker satellites. An incomplete list of A-GNSS smartphones can be found in [8]. The availability of A-GNSS in mobile phones is largely explained by government mandates, such as the United States E911 [9] and Europe E112 [10]. In these mandates, it is specified that in case of an emergency call, the handset should be able to deliver the location of the phone, with certain accuracy and within certain time. The location of the handset is quickly available from the cellular network itself, but it fails to provide the necessary accuracy. On the other hand, GNSS technology has the accuracy, but it can sometimes in an adverse environment, take significant amount of time until position is computed. The A-GNSS makes it possible to obtain GNSS position in a much shorter timeframe.

Recent market surveys already show signs for manufactures shifting towards true all-systems support [11]. Figure 1.1 shows how multi-system capability is supported in chipsets from year 2015. Another survey, including also professional grade receivers, shows similarly, how many manufacturers are adopting multi-GNSS support [12].

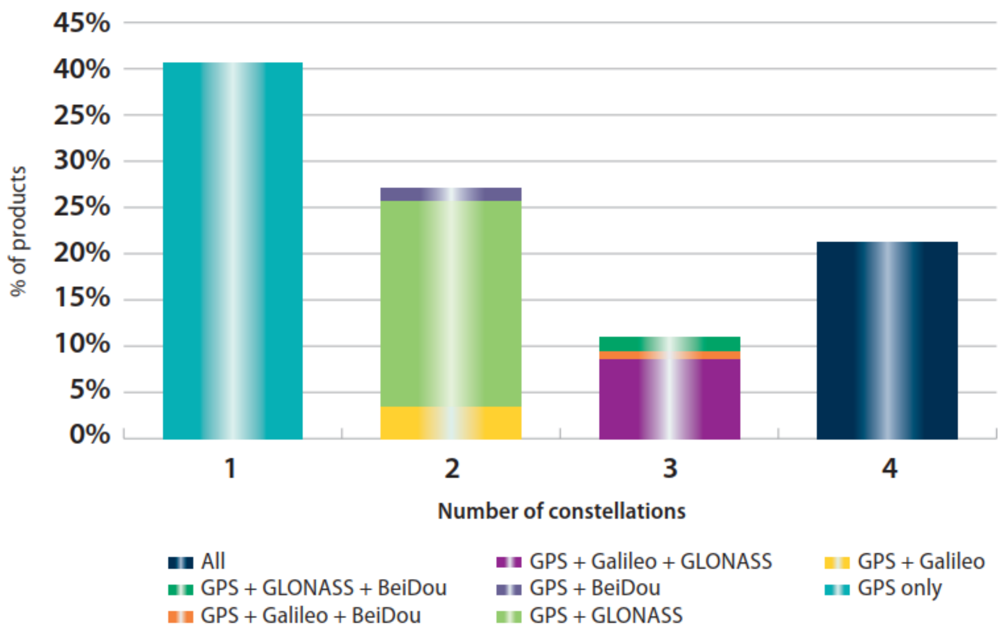


Figure 1.1 Supported constellation by receivers [11]

Multi-frequency support still seems to be only available for professional grade receivers. The multi-frequency capability does imply greater investment in the hardware cost in comparison to the multi-system support. Being able to process multiple frequencies, the receiver must face the challenge of having compatible antenna and front-end design in the receiver. Additionally, the processing of additional frequencies occupies extra tracking channels in the receiver. Benefits for having multi-frequency capability can be listed as:

1) ***Possibility to remove ionosphere effects from measurements***. Discussed in detail in Section 5.10.4.
2) ***Faster navigation data reception.*** For example, Galileo implements frequency interleaved frame structure for E1-B and E5b-I, where it is possible to get the data faster using two frequencies [13].
3) ***Resistance to jamming.*** In a situation where one particular GNSS frequency is affected by jamming or similar interference, it is possible to operate on the data and measurements from the other frequency.

According to the market survey [11], it is most likely automotive industry and road navigation Personal Navigation Device (PND) markets who will adopt the multi-frequency capability.

1.2 Research Objectives and Scope

The main objectives of this research will be improving positioning accuracy, optimization of cost efficiency of hardware (e.g. power saving on hand-held devices) and develop multi-system, multi-frequency algorithms. The research presented in this thesis was strongly driven by the will to develop novel / existing algorithms and to see them implemented in practice [14]. For the actual implementation, the algorithms were targeted for the TUTGNSS reference receiver [15], [16], which represents a platform for a modern multi-constellation, multi-frequency GNSS receiver. A library of implemented algorithms was created, providing easy way of including necessary functions to be tested on the platform. An open source educational approach was adopted [17].

In addition to the actual implementation of algorithms, two additional important topics were paid attention to. First, the implemented algorithms must be benchmarked, evaluated and verified. Verification procedures must be made to ensure that the algorithms are functioning in practice. One must keep in mind that an algorithm is unlikely to be implemented in the final platform if it has poor performance – thus the need for performance evaluations for each separate algorithm.

The last part addressed in this research was dedicated to the methods enhancing the testing phase of the receiver. Since all algorithms must undergo a time-consuming evaluation / verification phase, it was seen beneficial to develop improved testing mechanisms.

In conclusion, the three major topics are listed here:

- Design and implement multi-constellation, multi-frequency algorithms
- Benchmark, evaluate and verify the receiver functions
- Improve the testing mechanisms

In the scope of this thesis, we concentrate on a few particular aspects of GNSS. First, this thesis will concentrate on multi-constellation issues from GPS + Galileo perspective. Multi-frequency combination is selected to be L1/E1, L5/E5a, since it is the optimal combination for GPS + Galileo receivers [18]. The GNSS receiver algorithms are focusing on the navigation data decoding, and navigation solution computations.

Several early receivers exist with either multi-frequency capabilities or Galileo capability [19] - [25]. The selected GPS + Galileo frequency combination studied in this thesis and implemented on TUTGNSS is among the first 50 certified Galileo capable receivers in the world [26].

1.3 Main Contributions

The work presented in this thesis has contributed into three categories of innovations: Describing multi-constellation, multi-frequency GNSS receiver architecture; design and implementation of novel algorithms, and developing new testing methods. The following are the main contributions and outcomes from this thesis work:

A multi-constellation, multi-frequency receiver was developed [P1].

-[P1] Contribution of the author was the development and design of the Navigation Software of the receiver.

The navigation software has been profiled, identifying the bottlenecks of software computations in real-time receivers [P2].

Alternative numerical methods, including CORDIC and fixed-point arithmetic, have been introduced for easy code portability to different processor families [P2].

-[P2] Contribution of the author was the implementation of fixed point arithmetic, selection of relevant algorithms and benchmarking the architecture.

A custom Viterbi decoder suitable for Galileo signal was developed. The choice between hardware and software Viterbi was considered [P3].

A novel on-the-fly block de-interleaving method for Galileo signals was implemented. The on-the-fly method spreads the CPU load evenly, avoiding big bursts in CPU time [P3].

-[P3] Contribution of the author was the implementation of on-the-fly de-interleaver, HW Viterbi and SW Viterbi decoder.

A dual CPU debugging domain was developed. In order to capture internal data from the hardware, without interrupting the normal operations, a second processor was instantiated for handling the data recording via shared memory [P4].

For easier comparison of algorithm performance, the capability of running several navigation solutions in parallel was introduced [P4].

-[P4] Contribution of the author was implementation of algorithms, and design idea of the Test Environment.

Taking advantage of the better computational capabilities of modern desktop PCs, a software implementation of GNSS receiver was presented. It has the capability of significantly reduce the amount of time needed to run test scenarios, when working on recorded GNSS data [P5].

-[P5] Contribution of the author was the idea and implementation of SW testing speed-up.

GNSS receiver architecture capable of synchronizing its measurements with other devices was introduced. Synchronization with GNSS simulator allows instantaneous comparison of parameters, without the need of interpolation or prediction [P6].

The synchronization architecture allows new research on cooperative positioning, where synchronized GNSS receiver can share their measurements [P6].

-[P6] Contribution of the author was the implementation of synchronization mechanism.

1.4 Thesis Outline

The thesis has been divided into several distinct chapters, each one focusing on a different aspect related to GNSS. Figure 1.2 and Table 1.10 are provided for orienting how the chapters are organized and related to each other.

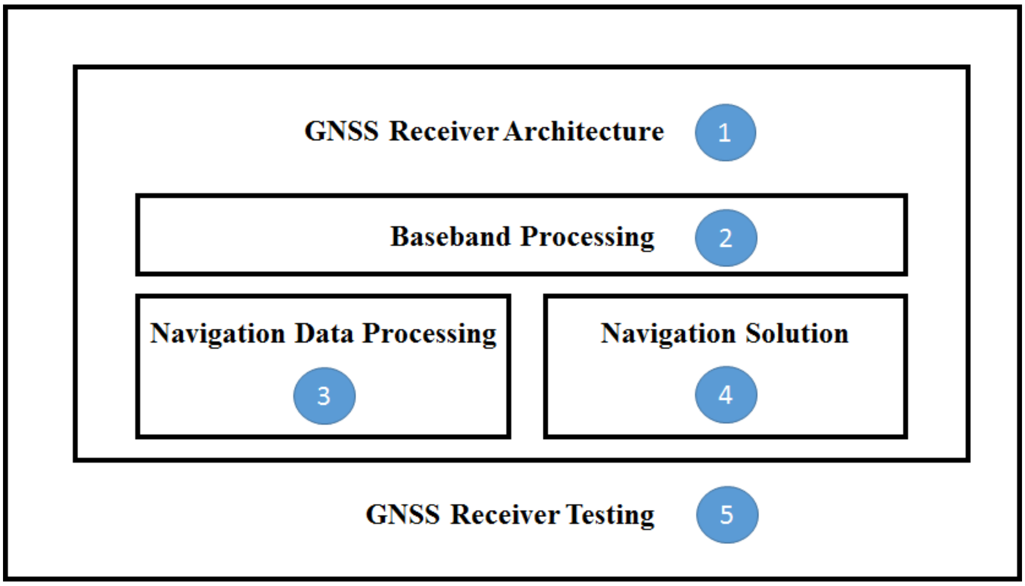


Figure 1.2 Diagram of thesis topics

Table 1.10 Linking the thesis topic to the chapters and publications

Block	Chapters	Publications
Block 1	2	[P1], [P4]
Block 2	3	[P3], [P4]
Block 3	4	[P2], [P3]
Block 4	5	[P2], [P6]
Block 5	2	[P2], [P4], [P5]

Chapter 2 will introduce the basics of satellite navigation. The GNSS system architecture, transmitted signals and error sources are included in the chapter. In addition, the basics of a GNSS receiver are discussed together with a section dedicated to testing of GNSS receivers.

Chapter 3 describes the baseband processing, including acquisition and tracking. Special considerations regarding multi-system, multi-frequency receivers are presented. Issues related to high sensitivity receivers are discussed, and finally it is analyzed how A-GNSS availability affects the receiver performance.

Chapter 4 focuses on the navigation data decoding issues. The requirements of multi-constellation GNSS receiver for data decoding have been taken into consideration. The data decoding performance limits of available open service signal are included.

Chapter 5 describes the wide variety of algorithms related to the navigation solution. Various models have been presented for different use cases of the receiver, depending on how many constellations or frequencies the receiver is processing.

Chapter 6 offers the conclusions and future work directions.

2. BASICS OF GNSS

When talking about navigation many people are mostly interested about seeing their position on the screen of a device. In order to understand how it is possible to obtain the position information, one must understand several basic concepts related to satellite positioning. This chapter aims to provide the reader with the insight to the basics of GNSS knowledge.

2.1 Concept of Satellite Positioning

The GNSS systems are radio navigation systems, where there are satellites sending radio wave signals. These signals travel at a known speed – the speed of light. The basic task of the GNSS receiver on Earth is to measure the time it took for the signal to travel from the satellite to the user receiver. This measured time is then converted into a measure of distance, by Eq. (2.1).

$$r = c \cdot t \tag{2.1}$$

, where

c is speed of light

t is transit time.

With three distance measurements to three transmitters with known locations, the receiver is able to calculate its position unambiguously by the concept of trilateration. GNSS is thus based on Time-of-Arrival (ToA) concept. Before looking into details of trilateration, the GNSS system architecture is introduced.

2.2 GNSS system architecture

The GNSS system architecture is traditionally split into three segments:

1. Control Segment
2. Space Segment
3. User Segment

Figure 2.1 illustrates the GNSS system architecture.

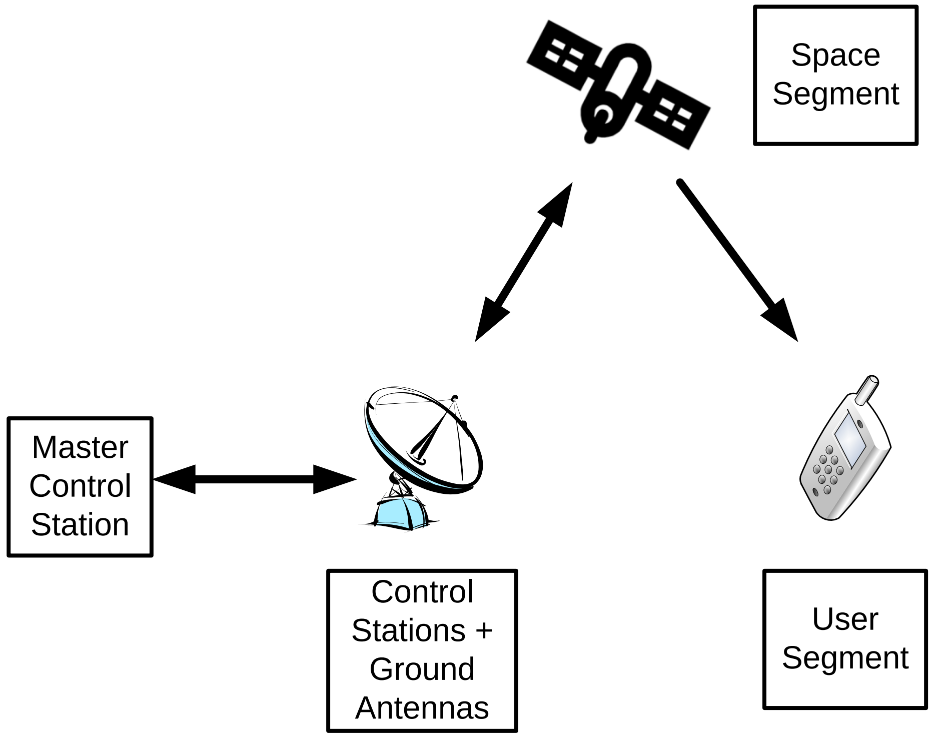


Figure 2.1 GNSS system architecture

The control segment is responsible for monitoring and maintaining the satellite orbits, satellite health and GNSS system time. The control segment also takes care of uploading the navigation message to the satellites. The control segment consists of one Master Control Station, several monitor stations and several ground antennas. The Monitor stations and ground antennas are placed around Earth, so that good coverage to the satellites is achieved.

Space segment consists of the full constellation of satellite vehicles. A nominal constellation for a particular satellite system has 24 satellites in the orbits, plus a few extra satellites for redundancy. The 24 satellites should ensure a global coverage at all times. Currently there are 31 GPS, 27 GLONASS, 18 Galileo and 20 BeiDou satellites in the orbits [27], [28], [29], [30], [31]. All GNSS systems deploy Medium Earth Orbit (MEO) satellites, with approximate radius of 26500 km. BeiDou constellation is special in the sense that the full constellation will also include 5 Geosynchronous Earth Orbit (GEO) and 3 Inclined Geosynchronous Satellite Orbit (IGSO) satellites. This will improve the visibility of satellites over China and the nearby regions. All these satellite constellations are designed such that they provide nearly always a visibility to at least four satellites, from a single GNSS system, to the user on Earth. The satellites are responsible for generating the radio navigation signals, which they will broadcast. A typical GNSS signal consists of several basic elements:

1. Carrier wave
2. Code
3. Navigation data

The carrier wave is a simple sinusoidal wave on the L-band frequencies. The GNSS systems utilize several carrier frequencies, and Figure 2.2 lists these L-band frequencies.

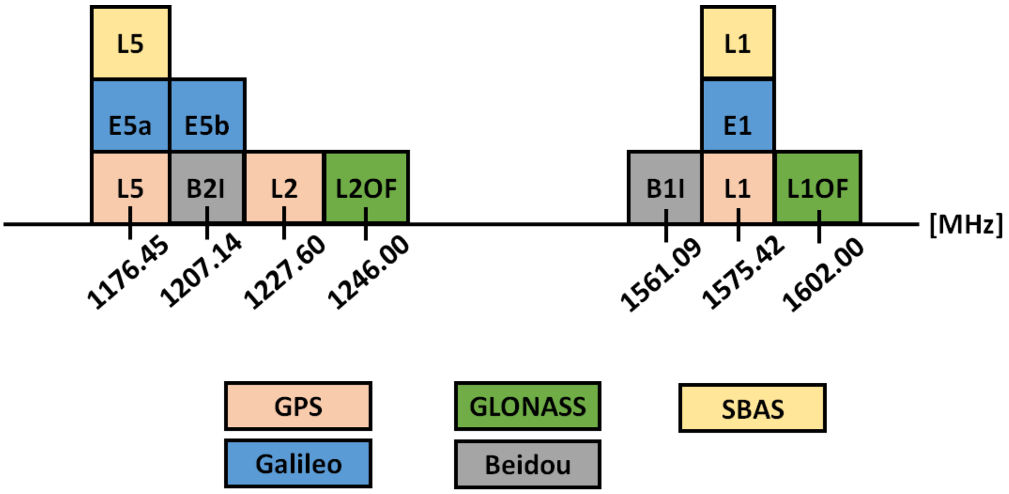


Figure 2.2 Available GNSS open service signals

The second element of the signal is the code. The different satellite systems may utilize additional levels of coding, but a distinct PRN code is found from each. The PRN codes have good autocorrelation properties, allowing the receiver to identify the transmitting satellites. On the other hand, cross-correlation between PRN codes is very low, meaning that the satellites will not be significantly interfere with each other. Cross-correlation is discussed in detail in Chapter 3.2. The use of PRN codes is the principle of CDMA systems. All the existing GNSS systems are CDMA systems with the exception of GLONASS, which is currently a frequency-division multiple access (FDMA) system. In GLONASS FDMA, all the satellites share a common PRN code, but each satellite has been assigned a different frequency slot around the common carrier frequency. In the future, also GLONASS will support CDMA signals with its next generation satellites - simplifying the receiver design, when all systems are CDMA based.

The final element is the navigation data, which is being sent at a relatively slow data rate, ranging from 50 bps to several hundred bits per second. The navigation data contains information about the GNSS system time, correction parameters for satellite clocks, information about the satellite orbits, satellite health status, and many other parameters. The receiver needs to be able to receive this navigation data before a satellite can be used for solving the user PVT.

These three components of the satellite signal are then modulated together, and broadcast to the users. Figure 2.3 illustrates the modulation of these signal elements, using GPS L1 civil signal as an example [32]. The same principle presented in Figure 2.3 can be extended to all other available GNSS signals. The carrier frequency, PRN code and NAV data rates may be different. Additionally, the signal may contain overlay codes. Tables 1.1 – 1.9 presented the basic facts of open service GNSS signals.

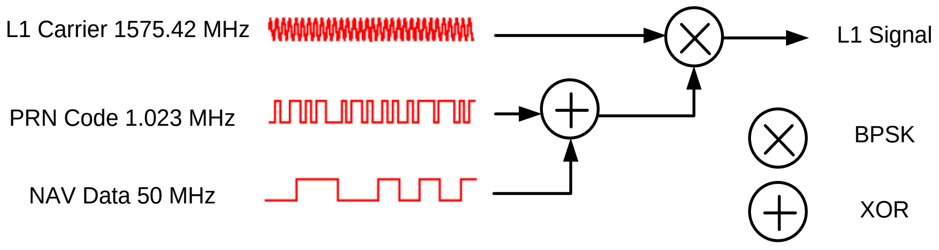


Figure 2.3 GPS L1 civil signal generation

The user segment just means the user receivers. The user receivers passively receive the broadcast satellite signals from which they make the ranging measurements. The receiver also needs to decode the navigation data message, in order to obtain timing parameters, and the necessary parameters for solving the satellite location. Finally, the receiver calculates the PVT solution and displays it in a meaningful format for the user – either as coordinates, or as plotting the position on a map.

2.3 Trilateration

The concept of trilateration is illustrated in Figure 2.4. In trilateration, each distance measurement to the satellites forms a sphere. The intersection of two spheres makes a circle (red lines in the figure). Three sphere intersections consist of two independent points (blue dots). If the system would be perfect, this would allow the user to solve its position unambiguously. In a perfect system, one of the points would appear on the surface of Earth – which would be the correct solution – and the other point could be discarded, since it would not be on the surface.

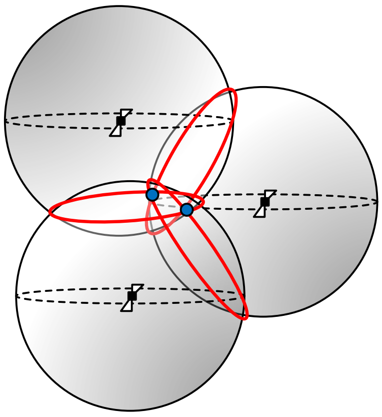


Figure 2.4 Concept of trilateration

Unfortunately, the system cannot be considered as perfect, and thus extra attention must be paid to timing issues. In practice, the clocks involved in the process are not synchronized. There are three types of clock domains involved:

1. GNSS system clock
2. Satellite clocks
3. User receiver clock

Out of the three clocks, the GNSS system time is considered the most accurate. It is a system clock maintained with very high precision equipment on the control stations on Earth.

Each individual satellite vehicle boards very accurate atomic clocks. The atomic clocks are quite expensive, but it makes perfectly sense to have expensive clocks in satellites, since satellites are in overall such expensive investments. Even these satellite atomic clocks have certain time bias and frequency drift when compared to the system time. These values are monitored by the control stations, and correction terms can be provided to compensate the effects.

The user receiver clock is the most inaccurate of the three. Typical receivers have crystal oscillators, and the main reason is that they are cheap. Since the mass-market receivers need to be affordable in price, the clock components cannot be overly expensive.

Conclusion:

The GNSS system clock is considered the reference time. Satellite clocks have small biases to the reference, and the effect can be compensated with correction parameters. The user receiver clock has bigger biases, and to solve the unknown bias, multilateration technique is employed in the receiver.

2.4 Multilateration

Multilateration is similar to trilateration, with the exception that now at least four measurements are needed. The receiver has to solve four unknowns, namely the user x, y and z – coordinates, plus the user clock time bias. For four unknowns four equations are needed. With the introduction of user clock bias, the basic distance measurement changes to a pseudorange measurement. A pseudorange is no longer an exact measure of distance, but it includes the effect of user clock bias. The model for a pseudorange measurement is given in Equation (2.2).

$$\rho = \sqrt{(x_s - x_u)^2 + (y_s - y_u)^2 (z_s - z_u)^2} + b_u \tag{2.2}$$

In Equation (2.2) the 4 unknown parameters are the user coordinates x_u, y_u, z_u and user clock bias b_u. The satellite coordinates x_s, y_s, z_s will be known once the navigation data is decoded, using the algorithm presented in Section 5.7. When the receiver has four simultaneous pseudorange measurements, it can solve PVT. The algorithms for solving the PVT are introduced in Chapter 4.

2.5 GNSS Error Sources

In satellite-based navigation, many error sources will degrade the performance of the receiver. These error sources can be roughly divided into three categories:

1. Space segment errors
2. Propagation channel errors
3. User receiver errors

In the next subsections, details that are more precise will be given about what is included in each of the error categories, and what is their estimated error contribution to the total performance of the receiver.

2.5.1 Space Segment Errors

The control segment constantly monitors the orbits and clock stability of the satellites. Based on the observations a set of parameters is then calculated, called the ephemeris data, which describes the predicted movement of the satellite on its orbit. Similarly, the clock correction parameters are computed for estimating the clock errors. The ephemeris data and clock parameters are more accurate the more recent the predictions are. After long periods, the predictions tend to start accumulating error. Currently the ephemeris data and clock corrections are recalculated and updated to the navigation data of the satellites once in a few hours. The control segment tries to improve the prediction models to reduce these errors. It has been estimated that the error contribution due to ephemeris data + clock correction parameters is approximately $\sigma_{CS} = 3$ m.

2.5.2 Propagation Channel Errors

The satellite orbits are such that the distance which the signal travels from satellite to the receiver varies between 20 000 km and 26 000 km. When a satellite is located directly over the receiver, at zenith = elevation angle of 90°, then the distance between satellite and receiver is the shortest. At lower elevation angles, the distance to the satellite increases. Figure 2.5 illustrates the concept of elevation angle.

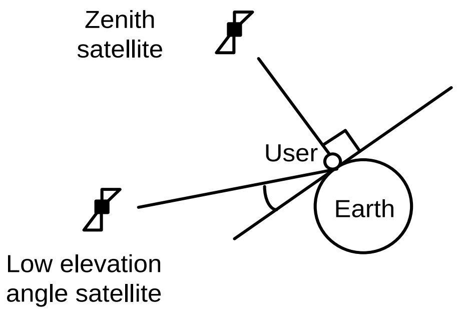


Figure 2.5 Elevation angle

The elevation angle is an important indicator of how good quality the satellite signal can be expected. This becomes apparent when considering the Earth’s atmosphere region. The atmospheric regions relevant to satellite signal are the ionosphere, around 1000 km above Earth, and the troposphere, around 40 km above Earth. These regions will affect the signal travel, thus causing errors to the measurements. As a conclusion: The lower the elevation angle, the longer distance the signal travels in the degrading atmospheric regions. This results in more error in the signal. Thus, high elevation signals are more favorable for positioning. There are models to compensate for the atmospheric errors, but they are not able to eliminate the errors. These models will be discussed in more details in Chapter 5. An average error contribution for propagation channel errors is σ_P = 5 m.

2.5.3 User Receiver Errors

The last error category has two distinct components to it, which will be described in the following subsections:

- *Receiver Noise*

The user receiver has to deal with noise and multipath. The total noise in the receiver is composed from many sources. Any RF signal around the band of interest is considered as noise. In addition, multi-access noise, that is the noise from other GNSS signals, is contributing. Then the receiver components - cables, filters, antenna, etc., generate some noise. Finally, the receiver will sample the incoming signal, which causes quantization noise.

- *Multipath*

The second distinct component is the multipath phenomenon. Multipath refers to a situation where the satellite signal is reflected from a surface, and then reaches the receiver antenna. The reflected signal can cause significant errors if not taken care of. A typical method of reducing the effect of multipath at the receiver level would be utilizing narrow spacing correlators [33]. The benefit of a narrow spacing correlator comes from the fact that the delayed multipath signal arrives later in time than the narrow spacing correlator range.

In a typical multipath scenario, the receiver receives both a direct path signal, and one or more reflected multipath signals. Usually the direct path signal is stronger than the multipath signal and the direct signal always reaches the antenna before the reflected signal. In some case, the direct line of sight signal will be blocked, and only the multipath signals are able to reach the receiver. In such a case, identifying the multipath case becomes more complex.

For multipath, the elevation angle is again a significant indicator. It is more likely to encounter multipath signals for low elevation satellites. This fact can be utilized in the antenna design of a GNSS receiver. The antenna should preferably be directional toward the sky, so that low elevation reflected signals would be attenuated at the antenna level already. Another antenna level multipath mitigation happens with the polarization of GNSS signals. The GNSS signals are Right Hand Circularly Polarized (RHCP), and the antenna is built to receive such signals. Polarity of the reflected signals may change to Left Hand Circularly Polarized (LHCP), and thus being attenuated by the antenna due to the wrong polarization.

As conclusion for the receiver error contribution, an average of σ_U = 1 m can be assumed. It must be noted that for some cases the multipath can cause significantly more error.

2.5.4 Total Error Budget

From the previous sections:

σ_{CS} = 3 m

σ_P = 5 m

σ_U = 1 m

These values can be combined into a characterization of the total errors in the receiver measurements. This combined error is referred as User Equivalent Range Error (UERE) [32].

$$\sigma_{UERE} = \sqrt{\sigma_{CS}^2 + \sigma_P^2 + \sigma_U^2} \approx 6 \text{ m} \tag{2.3}$$

The UERE value tells that the user of GNSS will not achieve perfect positioning solutions, but instead gets a solution where the individual measurements are expected to have error component of 6 m. The UERE value is not the only contributor to the accuracy of the positioning solution. In Chapter 5, another concept called satellite geometry is introduced and its effect together with UERE will be presented.

2.6 Testing of GNSS Receivers

Testing of the GNSS receivers is a continuous process where new features, architectures and algorithms are evaluated. Thorough testing ensures that changes in the receiver do not introduce coding bugs, or other unintentional side effects. Any regression in the performance should be detected, and the source corrected if possible.

In the development phase of the receiver, it is useful to pay attention to the availability of the test data. For debugging purposes, the receiver platform should be designed such that necessary internal process information can be extracted for further analysis. For example, the raw IQ data is often useful for performance analysis. The amount of IQ data depends on the RF FE characteristics, and in general, the data rates tend to be quite high. Thus, high transfer rate interface is needed for recording this data. For example, with the TUTGNSS receiver, a 100 Mbps Ethernet link was utilized for debug data transmission [34]. The next level of debugging data would be the raw measurements that the receiver baseband is making. This contains pseudorange and carrier phase measurements, CN0 levels of signals, and possibly acquisition results. Google has recently announced this level of GNSS measurements will be available for application developers on Android smartphones [35]. The third level of measurements would be the navigation results, i.e. position and velocity solution. Additionally, measures for data decoding bit error rates can be output.

With the help of raw data and measurement output, one can have multiple test metrics that actually define the true performance of the receiver. Rigorous testing using various testing methods is needed to have decent coverage of the receiver capabilities.

Testing methods can be categorized as follows:

1.) Live Sky
2.) Record and replay
3.) Simulators

The live sky testing has two different aspects. First option is to use an outdoor antenna, possibly mounted on the roof of a building. This can be convenient to test stationary scenario, where the receiver under development can be in the laboratory conditions with power supply easily provided. The second case would mean direct mobilization outdoors with the receiver. Depending on the type of the receiver being developed, this may need some extra equipment to provide protection and power supply for the receiver. The antenna would also need to be brought along. A suitable test rack can be built inside a car or a van, for car tests. The antenna can be mounted firmly on the roof of the car. A car can be used to test scenarios with considerable dynamics. Live sky testing is superior way of testing to find surprising conditions, where the receiver may behave unexpectedly. There is also a downside for live sky tests. If something interesting happened once, it may not be possible to replicate it again.

In record and replay testing the GNSS data is first recorded from the RF FE as IQ samples. If something interesting is found in the data, it can be replayed and further studied, thanks to the recording. The record can be used to improve the receiver as long as the receiver passes certain tests. The downside of the record and replay tests is that multi-system, multi-frequency receivers require high data rates for the IQ sampling. Usually quite long data records are desirable, and they tend to require a lot of storage space.

Simulator testing is the most often used testing method. The test cases are only limited by the features that the used simulator offers. Simulators offer testing possibilities, which are either impossible or highly inconvenient for live sky testing. Some examples include, changing time freely, simulating movement with high dynamics, high/low altitudes, high/low latitudes, testing leap seconds, testing interference/jamming, testing with various received signal strengths and the possibility to test satellite systems, which do not exist yet. Many more test cases naturally exist. Replay ability is excellent with most simulators.

Many test metrics can be tested:

1) Time to First Fix
2) Position accuracy
3) Acquisition and tracking sensitivity
4) Re-acquisition time
5) Dynamics tolerance
6) Multipath mitigation / performance
7) Interference mitigation / performance
8) Leap seconds
9) Pseudorange / Carrier phase accuracy

10) CN0 levels
11) Power consumption
12) Required hardware resources
13) Required software code size
14) Receiver meets real-time requirements

Given the long list, which is not even exhaustive, it is clear that testing takes considerable effort during the development of the receiver. It is thus preferred if the testing can be automated as much as possible. An individual test may take a long time to complete, and to get statistically significant amount of data, the test may need to be repeated hundreds of times. The amount of data that is accumulated in the long test session is often very large, and thus it is recommended to utilize automated analysis tools, which do the parsing of the data, and deliver the results in human readable format where it is easy to spot areas where the performance needs to be inspected in detail.

The testing can sometimes be a wild area, and it is not strictly standardized how to test all of the performance metrics listed in this chapter. For the more common metrics there is some effort put in the standardization, for example in [36]. Another issue is that there can be misunderstandings and misinterpretations of the measurement statistics. In reference [37], some of the basic statistical terms are well explained.

2.7 Multi-System Multi-Frequency GNSS Receiver Structure

The GNSS receiver architectures have traditionally been divided into a few separate blocks [38], [39], [40]. In this chapter, each of the blocks will be given a brief introduction. Figure 2.6 illustrates a simplified division of the GNSS blocks. Multi-frequency receivers may utilize multiple antennas, unlike presented in Figure.2.6

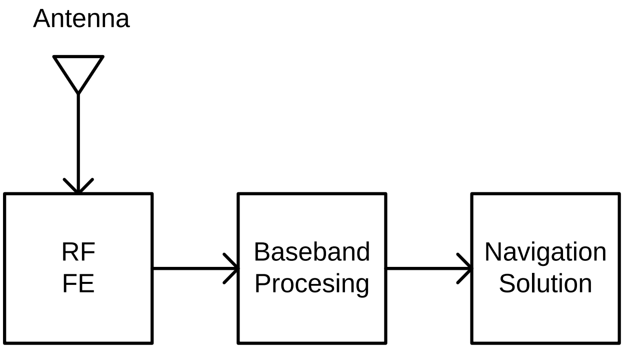


Figure 2.6 GNSS receiver blocks

2.7.1 Antenna

The antenna is the point of the receiver where the GNSS signals are captured [41]. To ensure successful capture of the signals, the antenna has to be able to capture sufficiently large bandwidth around the frequencies of interest. This also means that the antenna is also responsible for rejecting all the uninteresting out of band signals, which would otherwise cause noise to the signal processing. Since the antenna is the first part of the receiver where the signals are captured, the GNSS receiver is always solving the position of the antenna. As previously mentioned in Chapter 2.6, the antenna has a role in multipath rejection issues. In the same chapter, it was mentioned that GNSS antennas should be RHCP polarized. In the handheld mass-market receivers, antenna design is a challenge. The size and cost of the antenna is very restricting. In smartphones, antennas are simple microstrip antennas implemented on printed circuit board trace with negligible costs [42]. These simple antennas have very limited bandwidth, so they are tuned on a particular GNSS frequency. In order to cover the needs of future multi-frequency receivers, it is necessary to place several antennas to cover the needed frequency bands [43]. One example of multi-frequency antennas is presented in [44], where microstrip antennas are stacked together. The antennas of a handheld device should have omnidirectional gain, since the orientation of the device is not fixed. In addition, user body might introduce additional attenuation to the received signal strength.

2.7.2 Radio Frequency Front End

The RF FE block further continues to condition the received signal coming from antenna line. Signal coming from the antenna is filtered with Band Pass Filter (BPF), further narrowing the signal band. In order to prevent damage from harmful interference in the in-band frequency content, a burnout protection implementation can prevent excessive signal levels from reaching further components. A simple burnout protection is based on anti-parallel diodes that act as ground path for Electrostatic Discharge (ESD). Every RF FE solution has a preamplifier, which typically is a Low Noise Amplifier (LNA). The noise generated in the LNA determines the overall noise performance of the system, because noise generated at the beginning of the circuit is amplified together with the signal, by all the later components in the receiver. This is in accordance with Friis formula [45], [46], [47].

After the LNA, the signal can have one more stage of filtering for removing out-of-band interference. At this point, the signal is still a high frequency analogue signal in the GHz range. In order to reduce the computational burden of the later blocks of the chain, the signal is downconverted to a lower Intermediate Frequency (IF). In the downconversion Local Oscillator (LO) generated frequency components are mixed with the incoming signal. This effectively shifts the incoming signal into a lower frequency, but also generates unwanted sideband and harmonics to the spectrum. These effects can be taken care of with proper filtering. Sometimes there might be multiple IF stages in the RF FE block architecture. The RF FE block design follows the basic principles of superheterodyne receivers. Finally, the RF FE block has an ADC, which samples the incoming IF signal. The ADC has two important factors: First, the sampling rate is an essential parameter, as it dictates how much bandwidth the sampled signal has. According to Nyquist sampling theorem, the minimum sampling rate F_s should be greater that two times the highest frequency component of the signal. The second ADC parameter is sampling resolution. The digital samples are represented with a finite number of bits. If using few bits, quantization noise is introduced to the sampled signal. Some front-ends provide complex samples, while others provide only real samples. Finally, the ADC is associated with an Automatic Gain Control (AGC) in order to deal with different incoming signal levels. The AGC measures the input signal level and adaptively adjusts the gain so that the output signal level stays at appropriate level [48], [49].

The RF FE design for multi-frequency multi-system receivers requires additional resources, which naturally increases the cost of implementing such capability to mass-market receivers. A common tactic for enabling multi-frequency multi-system capability is to have separate signal paths for each signal of interest. This allows configuration of individual IF frequencies and filter bandwidths for separate RF paths. One suggested method to reuse parts of the RF chain is to utilize a switching architecture where samples would be taken alternating between the frequency bands [50]. This is illustrated in Figure 2.7.

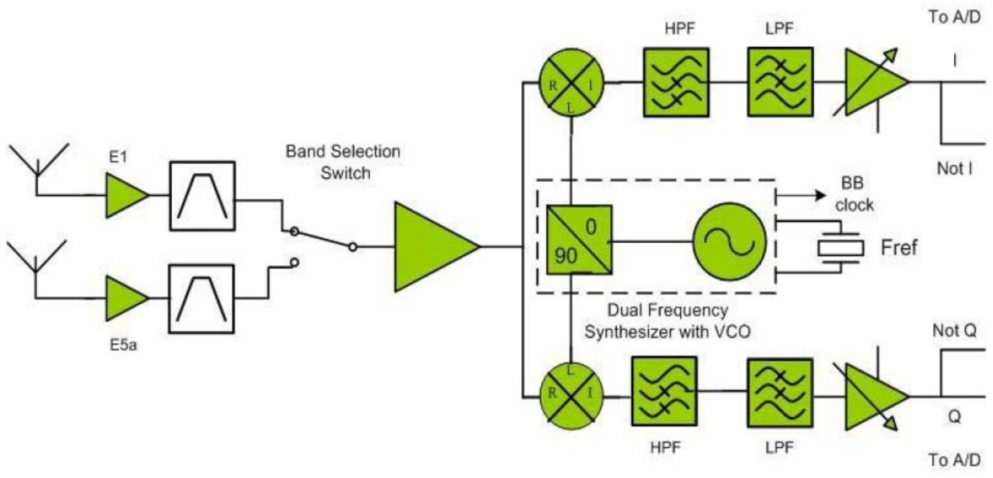


Figure 2.7 E1/L1- E5a/L5 switching receiver block diagram [50]

Another approach for multi-frequency multi-system FE architecture is to fully replicate the RF paths, while still allowing configurability for the paths. An example of a commercial 4-channel GPS/GLONASS/Galileo/BeiDou/IRNSS/QZSS L1/L2/L3/L5 band RF Front End is described in [51].

3. BASEBAND PROCESSING

The baseband block is defined to begin from the point where it gets the quantized digital output from the RF FE block. Typically, the baseband block is composed of many parallel channels, while each of the channels is capable of handling the necessary operations for one satellite signal. The baseband operations are quite computationally heavy and that is why most baseband designs are based on hardware solutions. Current trend is also to research Software Defined Radio (SDR) approach on GNSS field [52]. Figure 3.1 illustrates this transition towards SDR. In SDR design, the principle is to implement as much functionality in software as possible. The number of the parallel channels defines the limit for maximum number of satellite signals that can be processed simultaneously. The number of channels depends on the design needs, and it usually ranges between 5 and 16. Four satellites would be the minimum for producing navigation estimates, while five offers uninterrupted operation should one satellite drop. When implementing multi-system, multi-frequency GNSS receivers, it is desirable to have more than 16 channels available, though the more there are channels, the bigger the hardware unit gets. In addition, power consumption must be considered when selecting the right number of channels for a receiver. In general, the basic 5-16 channels should be multiplied by the number of GNSS systems supported + number of GNSS frequencies. In addition, if support for pilot channels is implemented, these should be budgeted to the total amount of channels.

The baseband processing is responsible for two major tasks, acquisition and tracking.

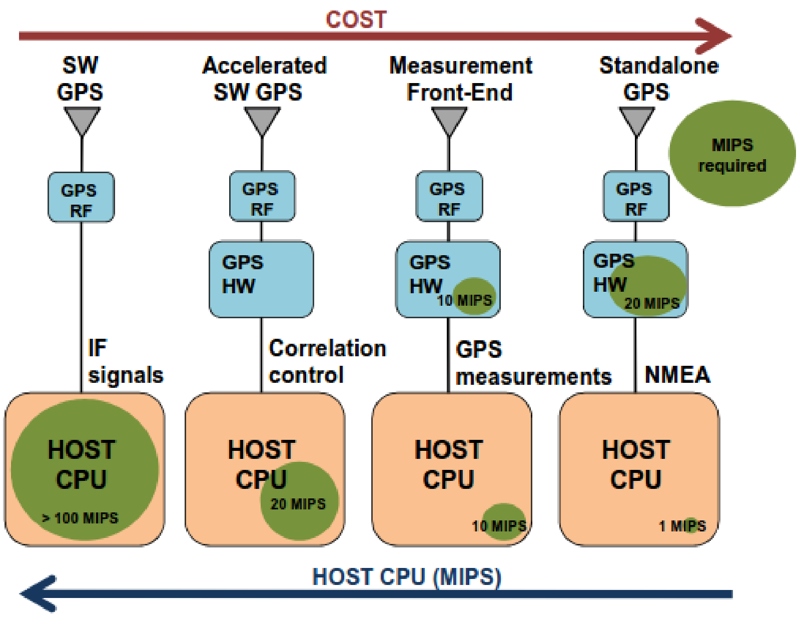


Figure 3.1 Different GPS receiver approaches [52]

3.1 Acquisition

From the incoming digitized signal, the baseband has to detect which satellites are sending a signal. This initial detection phase is called acquisition. The acquisition is a 3-dimensional search problem, with three unknown parameters:

1. Identify satellite PRN
2. Estimate Doppler frequency
3. Estimate code delay

For acquisition there exists the possibility of selecting a traditional hardware correlator based methods, or more software suitable FFT methods.

These methods are traditionally named as [53]:

A. Serial Search Acquisition
B. Parallel Frequency Search Acquisition
C. Parallel Code Search Acquisition

Methods 1. and 2. utilize hardware correlators which are simple multipliers used in the correlation process described by Equation (3.1)

$$z(k) = \sum_{m=0}^{N-1} x(m)h(m+k) \tag{3.1}$$

A) Serial Search Acquisition

In serial acquisition, there is a minimum number of correlators available, and usually that is 3 correlators. An example GPS L1 C/A code serial acquisition would be:

-C/A code with 0.5 chip sample spacing = 2*1023 bins

-1 ms integration

-31 Doppler bins (See Equation (3.3) later for description)

-3 correlators

That would result in 2046/3*31*1 ms = 21.1 seconds of search time per PRN.

Even if serial acquisition works with minimal hardware resources it is obvious that the method is extremely slow, which is why parallel acquisition method is much more preferable.

B) Parallel Frequency Search Acquisition

In parallel frequency acquisition, the number of correlators is dramatically increased. An example configuration is to have N*1023 correlators implemented, for instant coverage of all code delays (N = samples per chip). This correlator structure is called as matched filter structure.

An example with the same specifications as were used in serial acquisition example, would take 1*31*1 ms = 31 ms per PRN search.

A much better execution time has been achieved by simply increasing the number of correlators. The number of correlators is limited by the faster execution / hardware cost tradeoff.

C) Parallel Code Search Acquisition

The third method, parallel code search, also known as Fast Fourier Transform (FFT) acquisition, is justified by the fact that correlation in time domain equals multiplication in frequency domain. FFT is an efficient way of implementing the Discrete Fourier Transform (DFT). Usually FFT is implemented in software, but it can also be implemented in hardware. The implementation of FFT acquisition is illustrated in Figure 3.2.

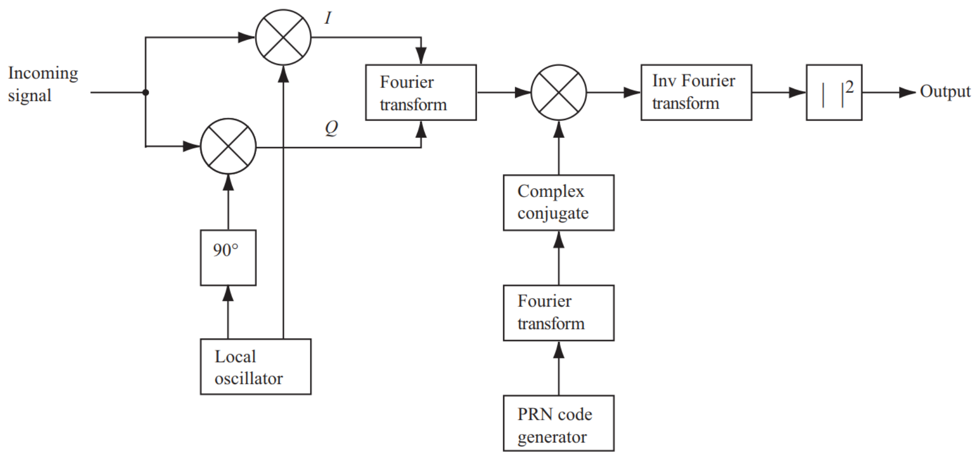


Figure 3.2 Block diagram of the parallel code search algorithm [53]

The search time for FFT algorithm largely depends on the efficiency of the FFT and Inverse FFT (IFFT) functions. The search of each frequency bin requires one FFT of incoming signal, one FFT of PRN code, and one IFFT operation. The FFT of the code can be precomputed and stored in memory if some performance gain is needed with the tradeoff of extra memory consumption. A comprehensive collection of FFT algorithm variations is presented in [54].

3.2 *Hand-over technique*

In multi-frequency receivers, it is possible to avoid the acquisition of some of the frequencies. This can be accomplished by a simple hand-over technique. If the receiver is configured to acquire the usual L1/E1 signal, and consecutively track the signal, then another frequency, e.g. L5/E5 can start tracking based on the L1/E1 Doppler and code phase information. The signals L1/E1 & L5/E5 signals are bit/frame synchronized, so it is possible to know from L1/E1 when the next PRN code epoch is going to start on the L5/E5 frequency. The only adjustment necessary to do is to scale the L1/E1 Doppler to the L5/E5 Doppler. In general, the longer the PRN code of the signal is, the more expensive it is to do regular acquisition. Thus, it is recommended to first acquire L1/E1 frequency, from where the handover can be done for other frequencies. In [55] the handover technique and its suitability with different signals is discussed.

This hand-over acquisition saves significant amount of HW resources and is thus considered efficient implementation of acquisition in multi-frequency receivers. Based on this, the rest of this chapter only focuses on the L1/E1 frequency acquisition details.

Code Generation.

For identifying the satellite, a full set of GNSS PRN codes is needed in the receiver. For the open service signal of GPS, GLONASS and BeiDou the code generation is defined in the Interface Control Documents (ICD), so that they can be generated offline at the receiver with a help of simple Linear-Feedback Shift Register (LFSR). Galileo E1 codes are different type, and they are called memory codes. Galileo E1 codes have no generator structure, and they must be saved in the receiver memory as such. The Galileo memory codes are given in the Galileo ICD. The sequence how the PRNs from various systems are searched is up to the designer of the GNSS chip.

As mentioned in the introduction chapter, GLONASS is an FDMA system, thus having only one PRN code to be generated. The individual satellites are identified by the allocated frequency slot. The GLONASS L1 equivalent center frequency f_c is 1602 MHz. Satellite frequency separation Δf is 562.5 kHz. Finally, individual satellite frequency allocation is given by $f_k = f_c + k \cdot \Delta f$, where $k = -7...6$. There are in total 14 frequencies for GLONASS satellites, although there are 24 GLONASS satellites. This is possible since GLONASS satellites are in antipodal locations on their orbits, i.e. only one satellite with particular frequency will be visible at a time, while the other satellite sharing the frequency will be on the other side of Earth.

Acquisition Search Parameters

A) PRN Selection

If no a-priori knowledge is available, the first parameter to search for is the PRN code. The hardware searches the satellites (PRN and constellation) with certain logic, which may optionally include prioritization of some constellation.

One possibility is to rely on random number generation, which picks a random constellation together with a random PRN to be searched for.

Another deterministic possibility is to sequentially search for the satellites. For example, start with: GPS PRN1, GLONASS k=-7, Galileo PRN1, BeiDou PRN1, next increment PRNs: GPS PRN2, GLONASS k=-6, …, etc.

A prioritized search would search for example first GPS PRNs 1-32, the GLONASS -7 … +6 frequency slots, then all Galileo PRNs, and then all BeiDou satellites.

In the end, the search logic should not affect how fast the receiver is able to find the satellites for positioning. From the locally generated PRN codes, the acquisition tries to find the right PRN code that correlates with the incoming signal. The PRN code is affected by the same Doppler frequency as the carrier wave, so that needs to be adjusted, together with finding the correct code phase.

B) Doppler Frequency

The second parameter is the estimate of the Doppler frequency. The baseband local oscillator tries to generate a carrier wave whose frequency and phase matches with the incoming signal. Unfortunately, the frequency is not exactly the IF, because both the satellite and the user are moving objects, which causes the actual frequency to vary around the IF. This effect is called the Doppler effect, and the Doppler frequencies depend on how two objects are moving in relation to each other.

There are three contributors to Doppler uncertainty:

1. Satellite movement
2. Local oscillator offset
3. Receiver movement

The first contributor is the satellite movement. When the satellite ascends and descends above the receiver’s horizon, there are moments when the satellite to receiver velocity reaches maximum of around ±800 m/s. This maximum satellite to receiver velocity sets the first uncertainty bounds to ±5 kHz [56].

The second contributor is the local oscillator offset. The mass-market devices use cheap TCXO oscillators, which typically have offsets specified to be between 2-5 ppm [57]. Each ppm will contribute 1.5742 kHz of Doppler uncertainty on L1 frequencies.

The third contribution comes from the receive movement. Every 1 km/h of receiver motion increases the Doppler uncertainty on L1 frequency by

$$L1 \cdot (1\,\mathrm{km/h})/c = 1.46\,\mathrm{Hz} \tag{3.2}$$

Mass-market receivers are not expected to encounter high-speed scenarios, and thus the receiver movement is not contributing much, in comparison to the previous two.

Typically, the total Doppler uncertainty for the receiver is around ±10 kHz.

Doppler Bin Size

The acquisition is trying different frequencies in steps called Doppler bins, which typically are a few hundred kHz wide. The more frequency bins there are, the more accurately the true Doppler frequency can be detected, but it is also more computationally demanding to search for more Doppler bins. A rule of thumb acquisition bin size is

$$2/(3 \cdot T_{coh}) \tag{3.3}$$

resulting in less than 1.5 dB loss from frequency mismatch. The equation for frequency mismatch loss is

$$\left|\sin(\pi \cdot f \cdot T_{coh})/(\pi \cdot f \cdot T_{coh})\right| \tag{3.4}$$

These equations already contain the T_{coh} parameter, which is the coherent integration period. This will be discussed in Section 3.2.1 below.

C) Code Phase

The last parameter to estimate is the time delay, i.e. the phase of the code. A natural property of the PRN codes is that they have very weak cross-correlation properties. This means that when correlating two different PRN codes there will hardly be any detectable correlation. With high sensitivity receivers, cross correlation will become an issue, and it is discussed later, in Chapter 3.3. On the other hand, PRN codes have noise-like autocorrelation property. This means that the code correlates with itself only with perfectly matched phase. These two properties enable the acquisition to detect the right PRN of the signal and solve the code phase. The acquisition searches all possible code phases, typically with half a chip bins, in order to find the maximum correlation value for the correct code phase. For example, GPS L1 Coarse Acquisition (C/A) PRN code has length of 1023 chips. If doing code phase search of ½-chip accuracy, it means that 2046 different code phases must be searched. Since autocorrelation peak is two chips wide, the half chip spacing ensures that correct code phase is searched.

Acquisition Result

The output of the acquisition is a 3D matrix, where a peak can be seen if the particular satellite signal is present in the received signal. Figure 3.3 presents acquisition results for a strong signal, which can be easily detected. Figure 3.4 presents a case for a weaker signal. It can be noted that when the signal power gets lower, the more problematic it becomes to detect the signal among the noise. The receiver decides if the signal has been found by comparing the power of the highest peak to the average power of the noise. From the index of the highest peak, correct Doppler and code delay values are then obtained.

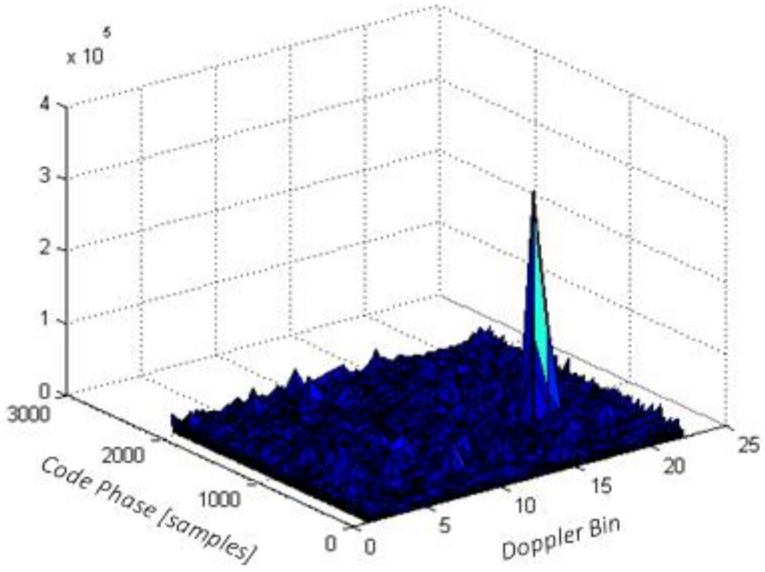


Figure 3.3 Acquisition results of a strong signal

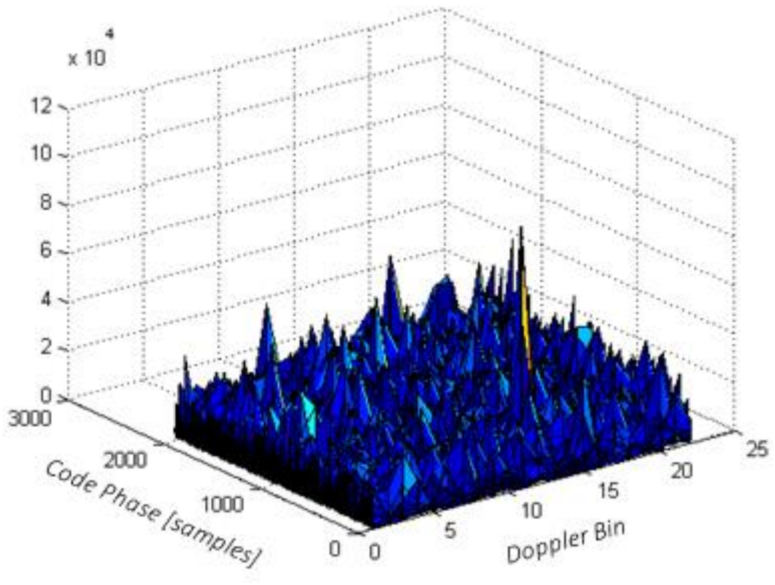


Figure 3.4 Acquisition results of a weak signal

3.2.1 Coherent integration

Coherent integration time was already present in the Equations (3.3) and (3.4). It is defined as the integration time where the phase information is retained. In practice, coherent integration time is limited by two factors:

1) Navigation data or secondary codes
2) Frequency stability

Navigation data bit changes introduce a 180-degree phase reversal into the correlation result. Thus, a data bit change during the coherent correlation process will decrease the correlation result. In acquisition, the data bit transition moments are usually not known, and that must be accounted for. For example, GPS L1 C/A data bit duration is 20 ms. When the bit boundary arrives, there is roughly a 50% probability that the data bit polarity will actually change. In actual data, the probability of a data bit change is less than 50%.

1 ms of coherent integration has roughly (1/20)*50% = 2.5% probability of including a data bit change.

2ms (2/20)*50% = 5% probability.

4 ms (4/20)*50% = 10% probability

10 ms (10/20)*50% = 25% probability

This sets a limit for a maximal length coherent integration time. A decent performance is achieved by selecting 10 ms of coherent integration time. If 10 ms of coherent integration is done on two successive data sets, one of them is guaranteed of not containing a data bit change [56].

For Galileo the symbol rate is 250 Hz (4 ms per symbol bit), which indicates symbol reversals much more often. Choosing 4 ms of coherent integration is preferable for achieving decent sensitivity levels. This may though lead to missed detections due to symbol polarity changes. Galileo E1 signal does include a pilot component for this reason. A pilot signal does not carry navigation data, and thus data bits do not limit coherent integration of a pilot component. After acquiring Galileo pilot signal, the receiver can do the simple hand-over to E1 data channel tracking.

GLONASS has the same data rate with GPS: 50 Hz (20 ms per data bit). On top of this, GLONASS navigation data is modulated with a 100 Hz Meander sequence. The Meander sequence is simply splitting every data bit into half, first part being same sign as the bit, and last part with opposite sign. This effectively results in bit duration of 10 ms for GLONASS.

BeiDou B1I data rate is 50 Hz. BeiDou has a Neumann-Hoffman (NH) secondary code applied on the navigation data. The NH code is 20 bits long;,with bit duration of 1 ms. It is aligned with the data bit. Thus, one 20 ms data bit of BeiDou is divided into 1 ms parts by the NH code. As 1 ms of coherent integration would be too limiting in the sensitivity sense, advanced acquisition schemes are suggested [58], [59].

3.2.2 Non-coherent integration

Non-coherent integration is used to overcome the limitations imposed by data bit changes. In non-coherent integration, the phase information is no longer preserved. The loss of phase information happens when the coherent integration results are first squared, and then summed non-coherently. The number of coherent integrations that are non-coherently summed depends on the sensitivity requirements of the receiver. In general:

$$\text{Ideal Coherent Gain} = 10 \cdot \log 10(\mathrm{M}_{\mathrm{c}}) \tag{3.5}$$

$$\text{Ideal Non-Coherent Gain } = 10 \cdot \log 10(\mathrm{M}_{\mathrm{nc}}) \tag{3.6}$$

where M_c is number of coherently summed samples,

and M_{nc} is number of non-coherent intervals.

In practice, non-coherent gain is less than coherent gain, since the rounding squaring operation involved in the non-coherent integration causes losses. Nevertheless, increasing the total integration time, by combined means of coherent and non-coherent integration times should lead to better sensitivity of the receiver. Every doubling of integration time ideally increases sensitivity level by 3 dB.

As a conclusion the acquisition part can be considered as a three dimensional searching problem, where correct frequency, phase and PRN code must be found. If successful, the peak of the acquisition matrix gives the unknown Doppler and code phase. After successful acquisition, the next step, tracking, of the satellite can be initiated. In addition, if the acquisition was not successful, that particular satellite can be tried to be acquired again.

3.3 Cross-correlation

Although the thesis has earlier stated that PRN codes have very low cross-correlation, issues may still arise when high-sensitivity receivers are concerned. The cross-correlation analysis of GPS L1 C/A code reveals that there are several weak cross-correlation peaks. These peaks are illustrated in Figure 3.5 and Figure 3.6.

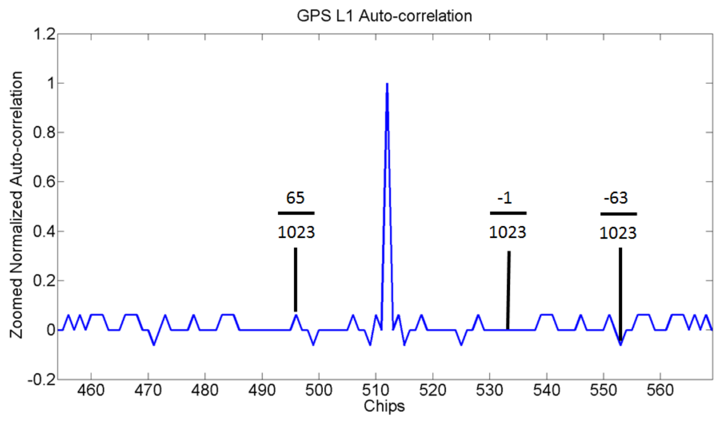


Figure 3.5 Auto-correlation of GPS L1 C/A codes

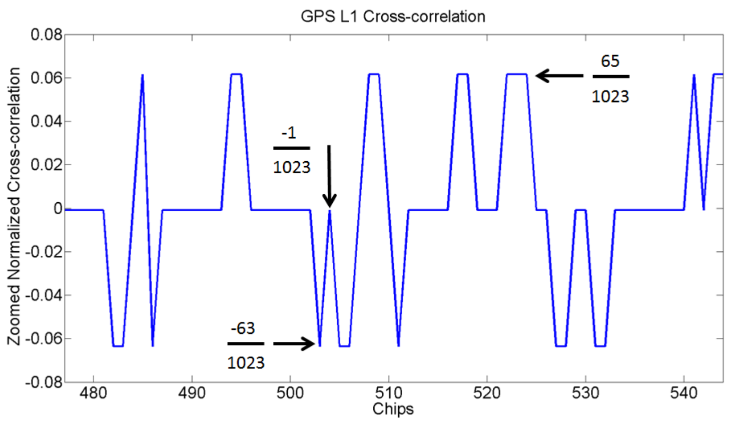


Figure 3.6 Cross-correlation of GPS L1 C/A codes

The figures show normalized correlation functions and their side peak levels. The side peaks in decibels are:

$$20 * \log 10 \left| \frac{63}{1023} \right| \approx -24dB$$

$$20 * \log 10 \left| \frac{65}{1023} \right| \approx -24dB$$

$$20 * \log 10 \left| \frac{-1}{1023} \right| \approx -60dB$$

The side peaks are 24 dB weaker than the main peak. With high sensitivity receivers which can track signals down to -160 dBm range, these weak side peak can cause problems. The nominal signal strength of GNSS signals is around -130 dBm. This means that going into sensitivity level below -150 dBm, care must be taken not to track cross-correlated signal side peaks.

Additionally, the previous simple analysis was done without considering Doppler effects. When full cross-correlation analysis is performed, all possible Doppler frequency combinations must be taken into account [60]. Certain Doppler combinations cause even higher power cross-correlation peaks than those shown in previous figures. For GPS L1CA signal, the worst case cross correlation levels can be as high as -19.2 dB [61].

If the receiver is able to track and decode the navigation data, and consequently include the cross-correlated satellite into navigation solution as valid satellite, then detection methods are needed in order to avoid serious problems in the navigation solution.

Detection of cross-correlated signal tracking can be implemented in several ways, and one fact should always hold – the cross-correlated signal should show clearly lower Carrier-to-Noise density (CN0) levels, around ~20 dB lower. The estimation of CN0 is discussed later in Chapter 3.4.8.

1) The first idea would involve checking the received ephemeris data for the SV number. Unfortunately, this is not possible with GPS L1 C/A data, which does not contain the information to which satellite the signal belongs. This is a clear shortcoming of the legacy GPS L1 C/A data. This information is included in the new L2C and L5 signals of GPS. In Galileo case, the SV number has been included in the E1 ephemeris, and cross-correlation can be more easily detected from there.

2) The second detection method based on navigation data would be to check almanac data. If almanac is available, a rough comparison can be made whether the decoded ephemeris data is comparable with the almanac data. If a large difference is detected and the tracked signal shows low CN0 values, it can be dropped from tracking.

3) The third method of detecting cross-correlation would be Doppler frequency comparison between other satellites. The cross-correlated satellite signals tend to lock on a Doppler frequency which is an integer multiple of 1 kHz of the strong signal with some tolerance [39], [62]. If a match of integer multiple of 1 kHz is found between strong and weak signal, the weak signal can potentially be cross-correlated. Since the satellites are constantly moving in the sky, causing their Doppler values to vary similarly, there are moments when the Doppler values will normally match 1 kHz window, i.e. Doppler collision. That is why the Doppler values need to be evaluated for longer period before making decision. If there was a cross-correlated signal, its Doppler will continue to follow the strong signal with the multiple of 1 kHz difference for a long period.

4) A fourth method, applicable in multi-frequency receivers, is to do Doppler check between multiple frequencies from the same satellite. If for one particular satellite multiple frequencies are being tracked, a simple sanity check between the frequencies can reveal cross-correlated signal. The L1 Doppler frequency, should match with Doppler adjusted L2 and L5 frequencies. The Doppler adjustment is simply to scale L1 Doppler to L2 and L5. To get L2 Doppler from L1, multiply L1 with the factor of the frequencies, which is L2/L1. Similarly, L5 Doppler scale is L5/L1. If the scaled Doppler values do not match, cross-correlation is detected. The new GPS PRN codes for L2C and L5 have better cross-

correlation properties, so they are less likely to be cross-correlated ones. Table 3.1 lists a few relevant cross-correlation levels [63].

Table 3.1 Correlation properties of 10230 long codes

Code Family	Max Cross Sidelobe (dB)
Galileo E5a	-26.4
GPS L5	-27.0
GPS L2C CM	-25.4

Galileo E1-B and Sidelobes

Galileo E1-B PRN codes are memory codes of length 4092. These codes have the same chipping rate as GPS 1.023 MHz. This results in code length of 4 ms, which is the same as Galileo E1-B symbol duration. Longer codes have the good property of offering better cross-correlation performance. Galileo E1-B PRN codes have an additional layer of modulation applied to them called Binary Offset Carrier (BOC). In the simplest form BOC(1,1) modulation splits every PRN code chip into two parts. PRN chip +1 is split into +1 and -1, while PRN chip -1 is split into -1 and +1. This effectively doubles the PRN code rate. BOC modulated signals have a spectral null at the center frequency, thus improving the spectral separation with GPS signals. Other peculiarity of BOC modulation is that it generates distinct side peaks to the auto-correlation function. This side peak can be tracked in a similar manner to cross-correlated signals. Side peak tracking is not that dangerous, since the data, which is decoded from the side peak, would still be valid for the particular satellite. The side peaks appear half a chip away from the main peak. Figure 3.7 illustrates the side peaks. Side peak tracking can be easily detected in the tracker by using multiple correlators. If there is stronger peak in the -½ chip, or +½ from the current peak, then the tracker is tracking the side peak. A simple code adjustment of half a chip in the tracker fixes the tracking to the main peak, and is called as Bump Jump. Figure 3.7 is auto-correlation function of the BOC(1,1) modulated Galileo E1-B PRN code, illustrating the cross-correlation levels and the side peak property. Ignoring the side peaks, the strongest cross-correlation peaks appear at levels of

$$20 * \log 10(0.0479) \approx -26.4 dB,$$

which is less than with GPS L1 C/A codes, as expected with the longer Galileo PRN code. Full analysis of the Galileo E1-B signal with Doppler taken into account, indicate worst-case cross-correlation peaks at level of -23.4 dB [61].

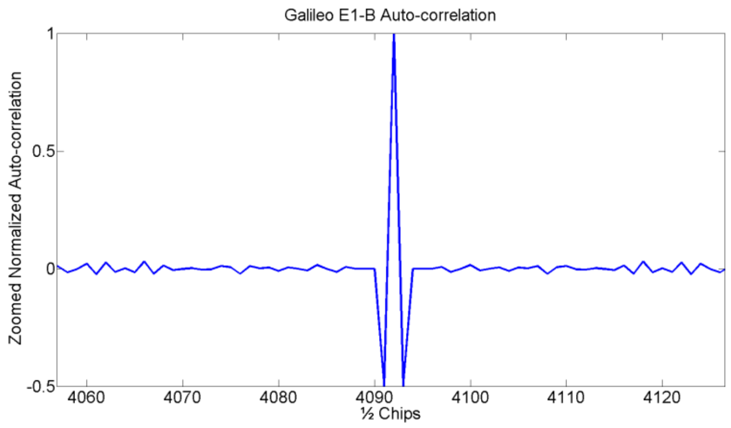


Figure 3.7 Galileo E1-B auto-correlation function zoomed around peak

As a conclusion to cross-correlation topic, it is worth noting that for example the QZSS system will be sending signals from the same Gold code family, and utilizing same navigation data structure as GPS. If the receiver does not have QZSS capability, it can still track cross-correlated QZSS signals as GPS signals. In this particular case, the detection of cross-correlation must happen on the navigation data layer. For QZSS the orbits are different from GPS, and thus some of the ephemeris parameters, like inclination, have different range of values. Thus, if ephemeris data contains parameters, which would not suit for GPS, it should be dropped as a suspect for cross-correlated signal.

A similar problem may can arise with SBAS signals, and the cross-correlation can be detected from navigation data layer. SBAS signals have an additional layer of FEC applied to the signals, so tracker will not be able to decode the data.

Finally, it should be noted that GLONASS FDMA signals are much more resistant to the cross-correlation problems due to the inherent frequency separation. GLONASS FDMA signals shared one PRN code, and the frequency allocation makes the system resistant to cross-correlation.

3.4 Tracking

Acquisition provides the initial rough information about the code delay and Doppler frequency. Based on this information a tracking channel is initiated for the given signal. The usual approach is to start the tracking channel when the next code period is going to start. In this way, the first 1 ms integration period is guaranteed to have no data bit transitions, since data bit transitions are aligned with code periods. The tracking part has two essential tasks.

1. Accurately track code phase and Doppler frequency - Allowing pseudorange and carrier phase measurements.
2. Perform code wipe-off and carrier wipe-off – Allowing samples of navigation data to be collected.

In the tracking channel, the code phase and Doppler frequency are being constantly observed, as these parameters keep changing. The tracking channel has a code wipe-off and a carrier wipe-off stage, where the respective parameters are essentially removed from the signal. A typical tracking channel has two distinct loops:

1. Code tracking loop
2. Carrier tracking loop

The code tracking loop utilizes a Delay Locked Loop (DLL), which tracks the code delay of the incoming signal. A carrier tracking loop can utilize either Phase Locked Loop (PLL), or Frequency Locked Loop (FLL), or a combination of both. The PLL tracks the incoming phase of the carrier, while FLL tracks the incoming carrier frequency.

With the help of the loop structures, the locally generated replicas continue to match the incoming signal with accurate results. Having both code and carrier wiped-off from the incoming signal, the remaining part is a sample of the navigation data [39].

The tracking loops include a few distinct blocks, which are illustrated in Figure 3.8.

1. Doppler Removal block performs the carrier wipe-off.
2. Correlators perform the code wipe-off.
3. Accumulators accumulate the correlation results.
4. Discriminators convert the accumulated I and Q samples into quantities of interest for their respective loops. FLL discriminator tracks frequency error. PLL discriminator tracks phase error. DLL discriminator tracks code delay error.
5. Loop filters filter the noisy discriminator measurements.
6. Numerically Controlled Oscillators (NCO) close the loops and integrate input values for code and carrier wipe-off.

The basic loop structure is the same for all the GNSS systems, and all frequencies. Naturally, the Coder generates the PRN code of the respective GNSS signal. Similarly, carrier frequency of the signal depends on the GNSS system and frequency. The integration time will vary between the systems due to limitations imposed by data bit edges or secondary codes. The approach to start the tracking with minimum integration time of 1 ms, is suitable for all GNSS systems. Integration time is gradually increased once the navigation data bit edges have been found.

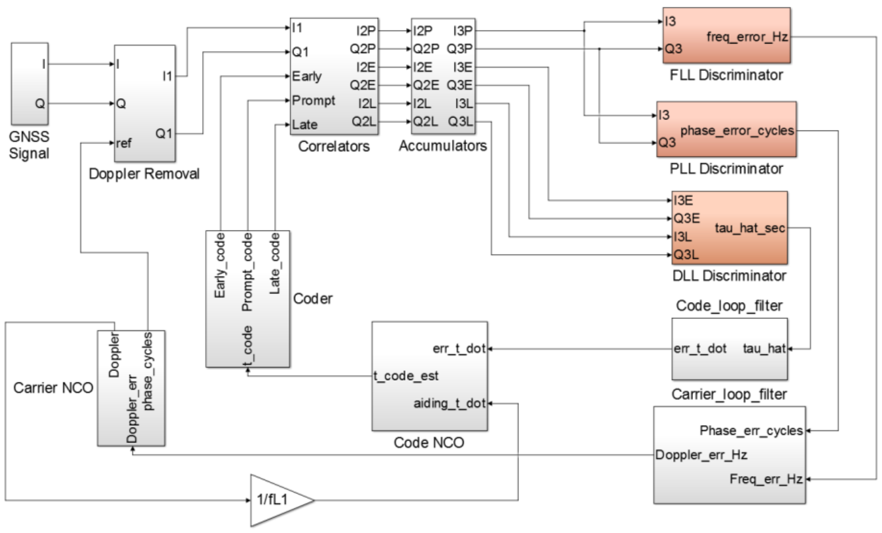


Figure 3.8 GNSS tracking loops with carrier aiding of code loop

The incoming I and Q samples in Figure 3.8 can be presented as (ignoring noise):

$$I_k = \frac{A}{\sqrt{2}} C_k D_k \cos(\varphi_k) \qquad Q_k = \frac{A}{\sqrt{2}} C_k D_k \sin(\varphi_k)$$

$$\varphi_k = 2\pi\left(f_{IF} + \Delta f\right) + \varphi_0$$

Where,

subscript k for *k*th sample,

A is carrier amplitude,

C_k is PRN code sample,

D_k is navigation data bit sample,

f_{IF} is intermediate frequency,

Δf is Doppler frequency.

3.4.1 L2C Tracking

Most of the currently available open service GNSS signals can be tracked with the traditional tracking loop structures. Each signal has its own limitations imposed by the symbol / overlay code durations. The overlay codes can be removed entirely once the receiver is able to synchronize to the frame structure. In GLONASS tracking, channels have only 1 PRN code for each satellite, and the satellites are identified by the frequency slot.

The GPS L2C signal is special in the sense that it is a combination of data channel and pilot channel time multiplexed into one single signal. When using handover technique from the L1 C/A tracking to L2C tracking, it is quite straightforward to implement the needed tracking structure. The L2C signal is split into time multiplexed L2CM and L2CL slots, where L2CM is a medium length code with duration of 20 ms, and L2CL is long code with duration of 1.5 s. The data bit is modulated with the L2CM part of the signal. L2CL is the pilot part without data. A more detailed analysis of the L2C signal can be found in [64]. Figure 3.9 illustrates the timing relationship of L2C signal.

If the L1 C/A tracking has achieved bit synchronization, i.e. the data bit edges are known, it is possible to make the handover to L2C signal L2CM part at the start of every L1 C/A data bit. In this case, one can choose to ignore completely the L2CL part of the signal. L2CL can be ignored by zero filling the L2CL time slots. This simple approach comes with a loss of signal power since essentially half of the signal is then ignored.

In case also L2CL part of the signal is to be tracked properly, it can be done with further synchronization from L1 C/A. The L2CL code repeats every 1.5 seconds and its beginning is synchronized to the start of the L1 C/A subframe, which last 6 seconds. In other words, when frame synchronization is achieved, the handover can be initiated every 1.5 second, i.e. every 75 L1 C/A bits. It should be noted that the first timeslot at the start of subframe is for L2CM, so first L2CL code chip comes after the first L2CM chip. At this point, it is possible to track both L2CM and L2CL together for maximum tracking performance and CNAV navigation data.

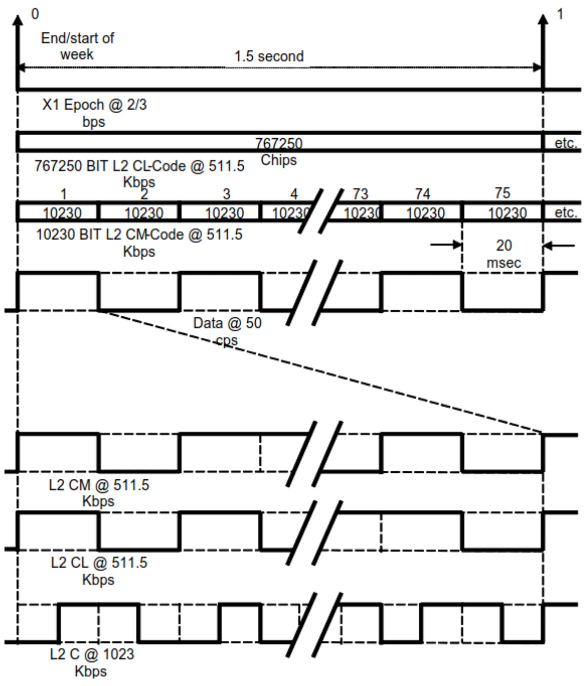


Figure 3.9 L2 CM-/L2 CL-code timing relationships [65]

3.4.2 Doppler Removal

The Doppler removal removes the remaining frequency from the incoming sampled signal. It should be recalled that the incoming signal is already mixed down and sampled on a certain IF frequency by the RF FE. Thus, Doppler removal is going to remove the IF frequency + Doppler frequency. The carrier tracking loop is generating a signal with matching phase, which is used for Doppler removal. Figure 3.10 illustrates this conceptually.

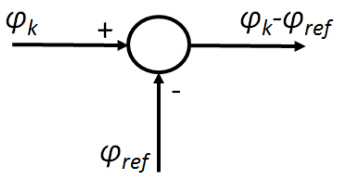


Figure 3.10 Concept of Doppler removal

Mathematically,

$$I_{1k} = \frac{A}{\sqrt{2}} C_k D_k \cos\left(\varphi_k - \varphi_{ref}\right)$$

$$= \frac{A}{\sqrt{2}} C_k D_k \cos(\varphi_k)\cos\left(\varphi_{ref}\right) + \frac{A}{\sqrt{2}} C_k D_k \sin(\varphi_k)\sin\left(\varphi_{ref}\right)$$

$$= I_k \cos\left(\varphi_{ref}\right) + Q_k \sin\left(\varphi_{ref}\right)$$

$$Q_{1k} = \frac{A}{\sqrt{2}} C_k D_k \sin\left(\varphi_k - \varphi_{ref}\right)$$

$$= \frac{A}{\sqrt{2}} C_k D_k \sin(\varphi_k)\cos\left(\varphi_{ref}\right) - \frac{A}{\sqrt{2}} C_k D_k \sin(\varphi_k)\cos\left(\varphi_{ref}\right)$$

$$= Q_k \cos\left(\varphi_{ref}\right) - I_k \sin\left(\varphi_{ref}\right)$$

Once the tracking loop is tracking the signal phase:

$$\varphi_k = \varphi_{ref}$$

$$I_{1k} = \frac{A}{\sqrt{2}} C_k D_k \cos(0) = \frac{A}{\sqrt{2}} C_k D_k$$

$$Q_{1k} = \frac{A}{\sqrt{2}} C_k D_k \sin(0) = 0$$

All the signal power is in the I_{1k} branch.

3.4.3 Correlators

The correlators are simple multipliers where the incoming I_{1k} and Q_{1k} samples are multiplied by the locally generated PRN code samples. The coder produces three different PRN code samples, called Early, Prompt and Late (EPL), as illustrated in Figure 3.11. Based on the Early and Late correlation results, the DLL discriminator can determine the direction of the time adjustment. For this reason, a minimum of three correlators is needed in the receiver. The spacing of the EPL correlators is another design factor, which depends on the signal properties. The EPL correlators should all fit within the autocorrelation peak of the signal. For GPS L1 C/A, the autocorrelation peak is ±1 chip wide, and thus an EP and PL separation of ½ chip would all fit within the peak. On the other hand, when signal PRNs get longer and have higher chipping rate, the peak gets narrower, and thus the EPL must be adjusted to remain within the autocorrelation peak. For Galileo E1-B signal, the BOC modulation also creates the side peaks to ±0.5 chips from the main peak. In order to detect side peak tracking, it is useful to implement extra correlators, for example a total of 5 correlators. The extra correlators would be called Very Early (VE) and Very Late (VL). Then the correlator spacing could be defined such that VE and VL would hit the side peaks, and EPL would hit the main peak. If ever VE or VL would show a higher peak than Prompt correlator would, then bump jump could be initiated.

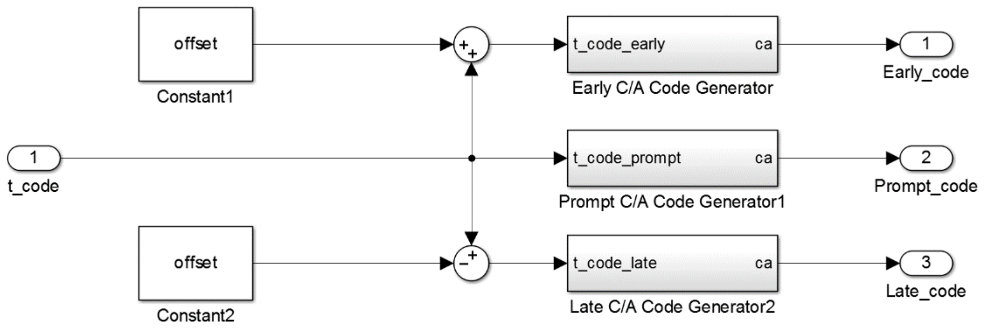


Figure 3.11 EPL code generation

The correlator spacing also affects the tracker pseudorange noise levels and multipath mitigation capability. The narrower the EPL spacing, the less noisy pseudorange measurements are. Similarly, the narrower the EPL spacing, the less effect multipath signals have on the code discriminator [62]. From this perspective, a narrow spacing correlator setup is a superior choice.

Mathematically, the receiver generates:

$C_{kr,e}$ (early), $C_{kr,p}$ (prompt) and $C_{kr,l}$ (late)

Autocorrelation function:

$$E\left[C_k C_{kr,m}\right] = R(\tau_{km}) \qquad \text{(m being either e,p or l)}$$

$$\approx 1 - |\tau_{km}|, \quad |\tau_{km}| \leq 1$$

$$\approx 0, \qquad |\tau_{km}| > 1$$

$$I_{2k} = \frac{A}{\sqrt{2}} C_k C_{kr,m} D_k \cos(\varphi_k - \varphi_{ref})$$

$$Q_{2k} = \frac{A}{\sqrt{2}} C_k C_{kr,m} D_k \sin(\varphi_k - \varphi_{ref})$$

Expected value of the correlator output:

$$I_{2km} = \frac{A}{\sqrt{2}} R(\tau_m) D_k \cos(\varphi_k - \varphi_{ref})$$

$$Q_{2km} = \frac{A}{\sqrt{2}} R(\tau_m) D_k \sin(\varphi_k - \varphi_{ref})$$

When the carrier loop has matching phase, and code loop perfectly matching code delay, then

$$I_{2kp} = \frac{A}{\sqrt{2}} D_k$$

$$Q_{2kp} = 0$$

In other words, all that remains is the navigation data bit sample (+ noise, which was not included in the equations).

3.4.4 Accumulators

The accumulators are used to sum up the I_{2k} and Q_{2k} samples in order to improve the SNR of the signal. Longer integration periods are needed in order to be able to track weak signals, as was explained in Sections 3.2.1 and 3.2.3, dealing with integration time.

3.4.5 Discriminators

There exist various types of discriminators for PLL, FLL and DLL. In general, computational complexity of the discriminators is not a problem these days. In that sense, the best available discriminators should be used. The quality of a discriminator is closely related to how linear its output will be. The more linear the output, the better the SNR will be.

PLL Discriminators

PLL discriminators use accumulated I and Q samples to determine the phase offset. Table 3.2 lists several available discriminators for PLL [39].

Table 3.2 PLL discriminators

Discriminator Algorithm	Costas Loop	Output Phase Error	Notes
sign(I)*Q	Yes	sin(φ)	Near optimal at high SNR. Slope proportional to signal amplitude A.
I*Q	Yes	sin(2φ)	Near optimal at low SNR. Slope proportional to signal amplitude squared A^2
Q/I	Yes	tan(φ)	Suboptimal, but good at high and low SNR. Slope not signal amplitude dependent.
atan(Q/I)	Yes	φ	Optimal at high and low SNR. Slope not signal amplitude dependent.
atan2(Q/I)	No	φ	Optimal at high and low SNR. Slope not amplitude dependent.

Table 3.2 mentions how majority of the discriminators are Costas loops. A Costas discriminator is insensitive to the 180-degree phase reversals caused by the navigation data bit changes. If a given signal does have navigation data, then the receiver must use a Costas loop. Best performing Costas loop is the atan discriminator, with a linear range of ±90 degrees. Other Costas discriminators are linear up to around ±30 degrees. If the tracked signal does not contain navigation data, either due to being a pilot signal, or the navigation

data is wiped-off, then a non-Costas discriminator can be used. The non-Costas discriminator is called coherent discriminator, and it is the atan2 discriminator.

FLL discriminators

FLL discriminators use the I and Q samples to determine the frequency offset. Table 3.3 lists several available discriminators for FLL [39].

Table 3.3 FLL discriminators

Discriminator Algorithm	Output Frequency Error	Characteristics
sign(dot) * cross / $(t_2 - t_1)$ where dot = $I_{k-1}*I_k + Q_{k-1}*Q_k$ cross = $I_{k-1}*Q_k - I_k*Q_{k-1}$	$\sin[2(\varphi_2 - \varphi_1)] / (t_2 - t_1)$	Near optimal at high SNR. Slope proportional to signal amplitude A.
cross / $(t_2 - t_1)$	$\sin[(\varphi_2 - \varphi_1)] / (t_2 - t_1)$	Near optimal at low SNR. Slope proportional to signal amplitude squared A^2.
atan2(cross,dot) / $(t_2 - t_1)$	$(\varphi_2 - \varphi_1) / (t_2 - t_1)$	Four-quadrant arctangent. Optimal at high and low SNR. Slope not signal amplitude dependent.

The atan2 discriminator is the best selection for FLL loop. From Table 3.3 it can be seen that FLL discriminators utilize current and previous I and Q samples to determine the output. This requires the tracking channel to save previous I and Q samples.

An often-neglected fact is that FLL discriminators are likewise sensitive to data bit transitions. With high SNR signals, it is possible to ignore this fact and the FLL loop will still perform adequately. If data bits are ignored, then the FLL loop can be operated with the integration lengths that match the data bit duration. When dealing with weak signals, data bit transitions must be taken into account. In [66] a structure for an FLL loop is presented, where the FLL loop skips the integration period containing the data bit transition, i.e. previous I and Q samples are from previous data bit, and current I and Q samples from a new data bit. Considering for example GPS L1 C/A signal with 20 ms data bit duration, there are a few possible combinations how to take data bit transitions into account with FLL loop. The first possibility is to run the loop with 10 ms integration period, and ignoring every other 10 ms, containing the data bit transition. Next possibility would be to run the loop with 5 ms integration time, add up 3 of those together, and ignore the last 5 ms. This approach would reach 15 ms total integration time. Last example combination would use 1 ms integration times, add 19 of those together and ignore the one after data bit transition.

DLL discriminators

DLL discriminators use the I and Q samples to determine the code phase (time) offset. Table 3.4 lists several available DLL discriminators [39].

Table 3.4 DLL discriminators

Discriminator Algorithm	Coherent	Notes
$I_E - I_L$	Yes	Requires all power to be in the I part of the signal.
$(I_E - I_L)*I_P + (Q_E - Q_L)*Q_P$	No	Uses all three correlators. Some error, but pretty good within 0.5 chip.
$\frac{1}{2}\left[\left(I_E^2 + Q_E^2\right) - \left(I_L^2 + Q_L^2\right)\right]$	No	Good within 0.5 chip.
$\frac{1}{2}\left[\sqrt{I_E^2 + Q_E^2} - \sqrt{I_L^2 + Q_L^2}\right]$	No	Good within 0.5 chip.
$\frac{\sqrt{I_E^2 + Q_E^2} - \sqrt{I_L^2 + Q_L^2}}{\sqrt{I_E^2 + Q_E^2} + \sqrt{I_L^2 + Q_L^2}}$	No	Good within 1.5 chip. (Divide by zero at ±1.5 chip)

These DLL discriminators use a variable number of I and Q correlator samples. The DLL discriminators also have the choice between coherent and non-coherent types. Since, in practice, the tracking loop never has a matching phase at the start of operation, it is mandatory to have a non-coherent DLL discriminator. Then it is optional, if the loop should transition to coherent DLL discriminator once the phase lock has been achieved.

3.4.6 Loop filters

The tracking loop discriminators output the respective quantity of interest with considerable noise. It is necessary to filter out this noise in the loop to generate estimates that are more accurate. Typically, two design parameters must be set for the loop filters:

1. Loop Order
2. Loop Noise Bandwidth

The convention about the loop order is such that the NCO of the tracking loop is counted as one integrator in the loop. Thus, the loop order is: loop filter order + 1 (the NCO). By this convention, a first order loop has zeroth order filter and the end NCO. A second order loop has first order loop filter and the NCO, and so forth.

The loop order describes the loop capability of tolerating dynamic stress. A first order loop is sensitive to velocity, second order is sensitive to acceleration, and third order is sensitive to jerk. Since satellites are always moving with velocity, the minimum order of loops is second order. There is one exception to this rule with the code tracking loop. The method of carrier aiding of the code loop allows the carrier loop to remove dynamics from the code loop. By utilizing carrier aiding of the code loop, the code loop order can be reduced to first order loop. Another benefit of the carrier aiding is that loop bandwidth of the code loop can be significantly reduced [39].

In most mass-market applications, at least some level of acceleration is expected, so third order loops are mostly needed. It is possible to dynamically switch between loops in the tracking, in such a manner that while no acceleration is detected, a second order loop is used, and once acceleration is detected, the loop switches to a third order loop. The practical effect of the loop order is that the computations in the loop are more complex with the higher loop orders.

The second parameter of loop design is the noise bandwidth. This parameter describes how much dynamic stress the loop tolerates, and how quickly the loop converges to stable operation. The wider the noise bandwidth, the better the loop can operate in dynamic environment. Wide noise bandwidth also allows the loop to quickly converge to stable operation. The downside of wide noise bandwidth is that the output is going to be noisier. Thus, the noise bandwidth should be selected so that it is as narrow as possible and still able to tolerate the possible dynamic stress + initial uncertainty in the input. In [39] an analysis is done how much stress the loops can tolerate before starting to lose lock. In addition, [67] explains how to combine the FLL and PLL into FLL-aided-PLL structure.

The usual tracking loop filter designs are based on [39], where loop filters are derived from the analog filter design. These analog filters have then been converted to digital filters by bilinear transform methods. The given parameters in [39] have simplified the analog design in such way that damping ratio of the filters is fixed at 0.707. Utilization of the original analog filter design can be seen in [53] and [56]. The problem that arises with high sensitivity receivers is that the filter designs based on analog counterparts, tend to become unstable when the product of filter bandwidth and integration time grows. Long integration times are needed for weak signals, and thus the filter performance may start to get unstable. In high sensitivity receivers, when the product of filter bandwidth and integration time is expected to be large, it is better to use digital design of loop filters. The digital design of loop filters is presented in papers [68] and [69].

3.4.7 NCOs

The NCOs get their input from the loop filters. Their task is to integrate the respective input so that it will be the input for the carrier and code wipe-off over the next integration period. The carrier NCO gets the frequency estimate as input, and integrates that frequency in order to generate phase for carrier wipe-off. The concept of carrier NCO is illustrated in Figure 3.12.

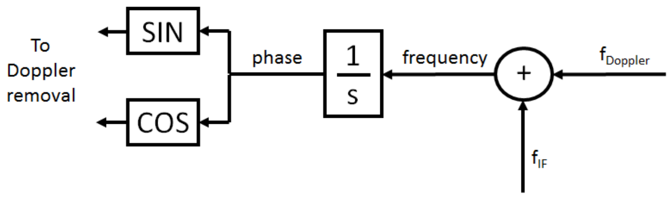


Figure 3.12 Carrier NCO

The code NCO input is the clock rate error due to Doppler effects. A nominal clock rate would be one s/s (one second in one second). With the clock rate error, the code NCO integrates time which is used in the code generators to generate the EPL samples of the code. The concept is illustrated in Figure 3.13.

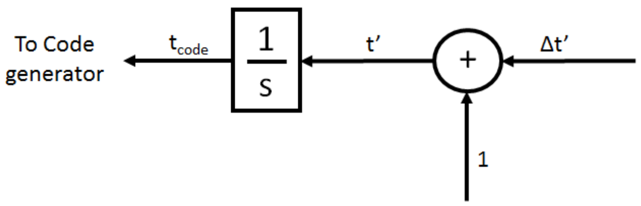


Figure 3.13 Code NCO

Carrier aiding of the code loop is conveniently explained together with the NCO structures. As was explained, the nominal clock rate of the code NCO changes due to Doppler effects. Since the carrier loop is solving the Doppler effect, it can be used to aid the code loop in a way illustrated in Figure 3.14. The carrier frequency f_c depends on the GNSS signal being tracked.

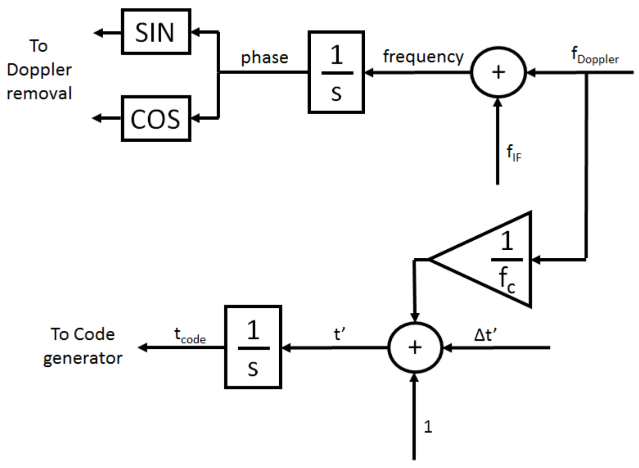


Figure 3.14 Carrier aiding of code loop

3.4.8 CN0 estimation

It is important to constantly monitor the CN0 level of the tracked signal. The CN0 level is an indicator of the signal strength. The tracking loops can have dynamically changing loop noise bandwidth, where based on the CN0 levels, the bandwidth is either increased or decreased. The weaker the signal, the smaller the loop noise bandwidth should be, in order to not lose lock. As a tradeoff, the smaller the loop bandwidth gets, the less dynamic stress the tracking loop can tolerate. The whole control of the tracking loop can be implemented as a state machine where CN0 and acceleration control the loop order, noise bandwidth and FLL-PLL operation combination. A selection of GNSS suitable CN0 estimators is presented in [70].

3.4.9 Alias detection

The typical use of the FLL loop is to achieve initial frequency convergence from it. FLL is less sensitive to bit changes, less sensitive to receiver dynamics, and it has wider range of frequency mismatch range where it can operate. In the start of tracking, the Doppler frequency information from acquisition is very inaccurate, up to hundreds of Hz with short integration times. Still the FLL is able to converge to the correct frequency. Usually when the loop has converged to the correct frequency, the carrier loop control is switched from FLL to PLL. FLL can remain, as a backup solution if the receiver is experiencing dynamic stress. In FLL-assisted-PLL mode, the tracking loop combines the dynamic tolerance of FLL with the tracking accuracy of PLL. It is possible for the FLL to lock into a false frequency of the tracked signal. This phenomenon is called aliasing, and it usually means that FLL falsely locks to true frequency ± ½ of the data bit rate of the signal. For GPS L1 C/A this would be ±25 Hz. Thus depending on the data bit rate of the tracked signal, an alias detector should be configured for detecting tracking of false frequency [39].

3.4.10 Cycle slips

The PLL loop tracks the phase of the incoming signal. If the frequency is correct, then also frequency is tracked while tracking the phase. The PLL has less noise in the carrier phase measurements in comparison to the FLL. In addition, the PLL is more sensitive to dynamics than FLL. When the signal power gets low enough and the dynamics of the receiver get high enough, the PLL starts losing lock.

The loss of PLL lock is often called as cycle slip. There are three main reasons why cycle slips could occur:

1. Weak signal + dynamics
2. Signal gets obstructed
3. Errors in the receiver processing

Weak signal + dynamics

The first cause for cycle slips comes from the PLL tracking thresholds. In [39] the equations are derived for approximate PLL tracking thresholds. It includes also similar thresholds for the FLL and DLL loops, but it is the PLL loop, which usually starts losing lock first. The FLL and DLL can track weaker signals than the PLL. The rule of thumb 1-sigma tracking threshold for PLL is [39]:

$$\sigma_{PLL} = \sqrt{\sigma_{tPLL}^2 + \sigma_v^2 + \sigma_A^2} + \frac{\theta_e}{3} \leq 15^\circ$$

where

σ_{tPLL} is 1-sigma PLL thermal noise in degrees

σ_v is 1-sigma vibration induced oscillator jitter in degrees

σ_A is Allan variance-induced oscillator jitter in degrees

θ_e is dynamic stress error in the PLL tracking loop

The expression can be understood so that while PLL is able to maintain correct phase information within 15 degrees, it is able to track the signal.

PLL thermal noise contribution:

$$\sigma_{tPLL} = \frac{360}{2\pi}\sqrt{\frac{B_n}{CN0}\left(1+\frac{1}{2T*CN0}\right)} \quad \text{(Degrees)}$$

where

B_n is noise bandwidth of the PLL loop

$CN0$ is the carrier to noise power expressed in ratio

T is the integration time

Interpretation:

Small noise bandwidth => weak signal can be tracked.

Long integration time => weak signal can be tracked.

PLL dynamic stress contribution:

For second order PLL loop.

$$\theta_{e2} = 0.2809 \frac{dR^2 / dt^2}{B_n^2} \quad \text{(Degrees)}$$

For third order PLL loop.

$$\theta_{e3} = 0.4828 \frac{dR^3 / dt^3}{B_n^3} \quad \text{(Degrees)}$$

Interpretation:

High noise bandwidth => more dynamic stress tolerated

As a conclusion to the PLL tracking threshold, it can be noted how the PLL loop noise bandwidth has opposite effects on the thermal noise and dynamics tolerance. That is why it is important to control the loop bandwidths dynamically in order to be able to track weak signals in potentially high dynamic scenarios.

Signal obstruction

When the satellites and the receivers move, it is natural that sometimes the signal is obstructed. If a signal is completely blocked by, for example a building, the tracking loops will simply drop the signal. On the other hand, if the obstruction is short, like when going under a bridge, the tracking loops can keep track of the signal. The PLL will though miss several cycles during the time the signal was obstructed.

Errors in the receiver processing

The cause of errors due to receiver processing can be various. As an example, a real-time receiver can sometimes miss the deadlines to process next patch of data. In such cases, a cycle slip could be introduced.

Detection of cycle slips

The smallest possible cycle slip is a half cycle slip. This can be detected in the navigation data as a sudden change in the data polarity. The commonly used PLL discriminators are

Costas discriminators, which are insensitive to the 180° phase reversals naturally induced by the data bit changes.

Second possible step for cycle slip is one full cycle. This cannot be detected in the navigation data anymore, so other means for detecting cycle slips are needed. If pilot channel tracking is possible, the sensitivity of the PLL can be increased by the possibility of using a coherent discriminator. Since pilot channel does not carry data bits, this is possible. The combination of coherent discriminator and increased coherent integration times due to lack of navigation data, allow better sensitivity levels for pilot channel tracking, up to 6 dB [39].

The basic structure for PLL lock detection is presented in [39] and it comprises of pessimistic and optimistic lock detectors. These mechanisms are continuously checking if the PLL is able to maintain the lock. This is a method of noticing when the signal has been lost. The lock detector operates on low-pass filtered I and Q samples, so it takes a moment to detect the loss of lock. Thus, it cannot be used for instantaneous cycle slip detection.

Other means for detecting cycle slips are simple time differencing methods [71]. In differencing, simple time difference is taken and it is observed how many cycles the measurement changed. If Doppler measurement is available, it can be used to predict how many cycles the measurement should have changed [72]. A full cycle slip is related to the carrier frequency of the signal by wavelength, presented in Equation (3.7).

$$\lambda = \frac{c}{f_c} \tag{3.7}$$

where c is speed of light

and f_c is carrier frequency

For L1 carrier frequency $\lambda \approx 19cm$, L2 $\lambda \approx 24cm$ and L5 $\lambda \approx 25cm$

Multi-frequency receivers have the possibility to compare phase measurement from multiple frequencies. The frequency with the highest tracking thresholds can be used as reference to detect slips in the weaker frequency. Conceptually the differencing method is presented in Figure 3.15. A cycle slip of 100 cycles on GPS L1 C/A data was artificially introduced to the data at epoch 50. The slip is much easier to see from the differenced data. The possibility of comparing the difference between carrier smoothed and code based pseudoranges is also mentioned in [72]. This method is not efficient since it can only detect cycle slips that are considerably larger than the noise level on code based pseudorange measurements.

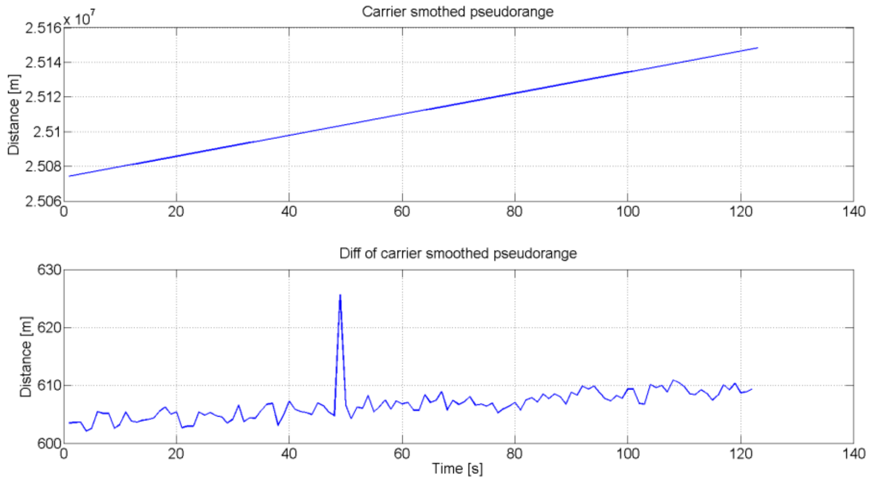


Figure 3.15 Cycle slip detection

To further enhance the power of the differencing method, one can choose to implement further levels of differencing. In [71] an example is shown about how the detection capability is amplified with additional layers of differencing. Naturally, additional levels of differencing causes additional delays to the detection of cycle slips. Table 3.5 illustrates the mechanism of multi-level differencing. Conceptually differencing acts as high pass filters, eliminating the constant parts, thus allowing the change in the input to pass.

Table 3.5 Scheme of differences [71]

t_i	$y(t_i)$	y^1	y^2	y^3	y^4
t_1	0				
		0			
t_2	0		0		
		0		ε	
t_3	0		ε		-3ε
		ε		-2ε	
t_4	ε		$-\varepsilon$		3ε
		0		ε	
t_5	ε		0		$-\varepsilon$
		0		0	
t_6	ε		0		
		0			
t_7	ε				

Polynomial fitting has also been used for cycle slip detection [73]. It identifies situations where carrier phase observable differs from a polynomial model fitted to the phase observable.

Cycle slip detection and correction is especially important in Precise Point Positioning (PPP) and Real Time Kinematic (RTK) applications of GNSS, where accuracy is more important than in mass-market receivers. These application are also trying to solve the integer ambiguity. The solution for integer ambiguity has a convergence time, which will be prolonged by uncorrected cycle slips. An example collection of PPP related cycle slip detection and correction algorithms can be found in [74].

At some occasions, the satellite that was tracked might disappear from the sight, and in these cases, the signal is usually lost from tracking. After a set period, a re-acquisition of the satellite is needed. In re-acquisition, the satellite is searched again, and after successful acquisition, the tracking can be re-initiated again. If the re-acquisition can be done soon after, the signal was lost, usually the Doppler frequency has not deviated much, and thus re-acquisition can do a narrower search concentrating on the known Doppler from previous tracking.

The output information from baseband will be the GNSS navigation data samples, the Doppler frequency, carrier- and code phase measurements. Based on this information the next part, navigation part, can perform its operations.

3.4.11 Assisted-GNSS

The A-GNSS feature may be available for handheld devices such as smartphones. In such a case, it offers important features in order to enhance the receiver's sensitivity and reduce the complexity of the acquisition task.

A-GNSS principle is to provide data to the receiver via an external communications channel, so that the receiver does not need to find the same data from the GNSS satellites. A list of possible assistance data includes:

1) Ephemeris data
2) Approximate position of the receiver
3) Coarse time information

The ephemeris data is the first component. It can consist of the full navigation data payload that the receiver would otherwise have to decode from the satellite signals. Through assistance communication channel it is possible to send this data much faster than the GNSS navigation data rates would allow. The benefit of obtaining ephemeris data comes from several factors:

When combined with an approximate position of the receiver and coarse time, it is possible to predict which satellites should be visible in the sky. Thus, it is possible to focus the acquisition effort on these satellites, cutting the amount of satellites to search for down to 30-40%. Roughly, one third of the satellites of a constellation are visible at a time, somewhat depending on the location and time of the receiver.

The next improvement of the assistance navigation data is that satellite locations are immediately available, together with the Time-Of-Week (TOW) information. Both of these need to be known prior to solving for the receiver position. With assisted data, this is information is immediately available, whereas if the receiver was to decode this information from the satellites, it could take around 30 seconds – assuming that no bit errors were encountered. In other words, A-GNSS enables faster Time to First Fix (TTFF).

The third improvement comes from the reduced Doppler search space for the acquisition. When approximate estimate for the receiver position and coarse time are available, it is possible to predict the Doppler velocity of the satellite. In this way the full Doppler uncertainty range due to satellite velocity ±5 kHz, can be reduced down to a few hundred Hz. Effectively the acquisition algorithm can now dwell longer in the correct Doppler bins for the same time, in order to improve the acquisition sensitivity.

A fourth improvement comes from the tracking loops. With the ephemeris data available from assistance data, it is possible to cancel the navigation data from the incoming satellite signals. By doing such data wipe-off, one practically converts the regular signal into pilot signal. The benefits of a pilot signal are that then coherent PLL discriminator can be used, and the coherent integration time is no longer bound by the data bits – thus, increasing the tracking sensitivity.

The approximate position of the receiver is usually computed in the cellular network from the database of cell tower locations. Depending on the location and density of these towers, the accuracy of the approximate location is some kilometers [75], which is enough for predicting the visible satellites and their Doppler. On the other hand, coarse timing information from cellular network is often not accurate enough to help with the code phase delays in the receiver. For this purpose, the timing information should have sub-millisecond accuracy. It can be assumed that mass-market receivers have Real Time Clock (RTC), which can keep the sub-millisecond precision even when the GNSS tracking is not on. Similarly, the local oscillator offset, which causes Doppler uncertainty to the acquisition, is being saved into non-volatile memory of the receiver. Thus, the accurate RTC clock time and saved oscillator offset can be utilized to further narrow down the search space of the acquisition. If no RTC clock is available and the receiver does not have non-volatile memory, then there exists methods of computing the position of the receiver with the help of the coarse timing information from the assistance data. This coarse time navigation algorithm is described in [75].

The A-GNSS is traditionally separated into two different modes: The Mobile Station (MS) – assisted and MS-based GNSS mode. In MS-assisted GNSS, the position of the receiver is ultimately computed at the cellular network server. In order to do this, the MS needs to send its measurements to the server. This method can further save computational time of the receiver processor.

As conclusion A-GNSS is a feature that bring huge improvements, with the small expense of deploying the supporting architecture in the device. Many smartphones come with this capability, so A-GNSS is a natural way of enhancing the performance of the GNSS receiver in them.

4. MULTI-CONSTELLATION ISSUES IN THE NAVIGATION DATA DECODING

The navigation data decoding is a continuous task, which cooperates with the baseband tracking. The baseband tracking channels provide navigation data samples to the data decoding, while data decoding provides information about data bit boundaries to the tracking channels for increasing the integration time. The process is quite similar for both GPS and Galileo, but some minor issues exist. In this chapter, the necessary steps are introduced for both systems. Figure 4.1 illustrates these steps.

A short listing of the basic data decoding items is as follows:

- GPS Bit Synchronization
- Bit / Symbol Decision
- Subframe / Page Synchronization
- Galileo Block De-Interleaving
- Galileo Compensate Inverter
- Galileo Viterbi Decoding
- Parity / CRC Check
- Data Collection

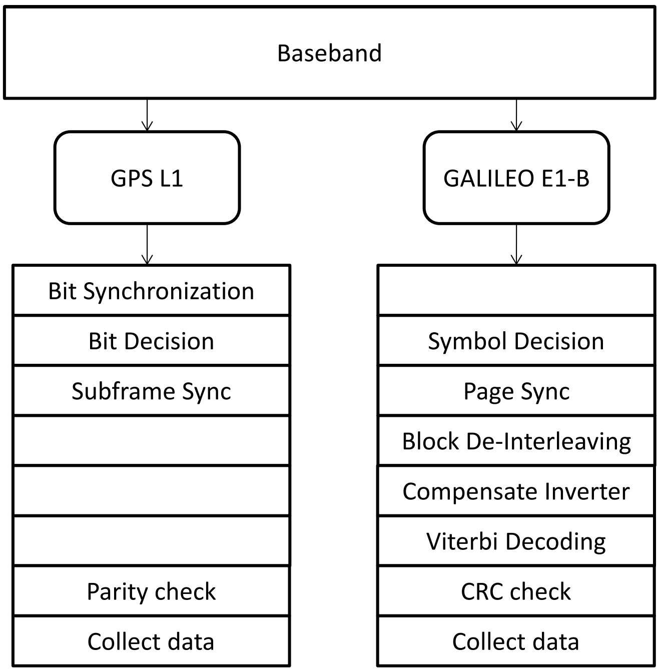


Figure 4.1 Navigation data decoding tasks

4.1 GPS Bit Synchronization

First, bit synchronization is described from GPS L1 signal perspective. In GPS L1 case, the navigation data bit length is 20 ms, and the C/A-code length is 1 ms. In other words, the C/A-code repeats exactly 20 times within a single navigation bit. Now the task is to decide which one of those 20 possible slots is the beginning of a data bit. With good signal strength, the correct slot is easy to spot, since the sign of the samples only changes according to the data bit changes in multiples of 20 ms intervals.

In weak signal conditions, the sign of the tracking sample may arbitrarily vary due to noise - making the decision a little bit more difficult. In such a case, multiple sign changes can be observed during the 20 ms period. For this reason, a statistical method of detecting the correct bit edge is usually utilized. A good example is provided in [62], where the histogram process has 20 slots assigned. Each time a sign change in the samples is detected, the corresponding slot counter is incremented. Eventually, the correct bit edge slot should get clearly the most increments. A detection threshold can be implemented in such a way, that once a certain slot reaches the threshold, it is declared as the correct bit edge slot. Figure 4.2 illustrates such an approach. Additional safety can be added by comparing highest slot against second highest slot. If the difference is big enough, the highest slot is determined as the bit boundary.

The Galileo E1-B signal does not have a similar problem of determining the bit edges. In E1-B case, the data bit length is 4 ms, and the code length is 4 ms as well. This means that the navigation data bit and code are aligned. Once the code starts, a new data bit also starts. In this chapter, we will assume that also for the Galileo implementation the tracking channel provides samples every 1 ms epoch. Thus, for E1-B data bit we get four samples. Further assumption is that every sample is now considered with a value of +1, or 0.

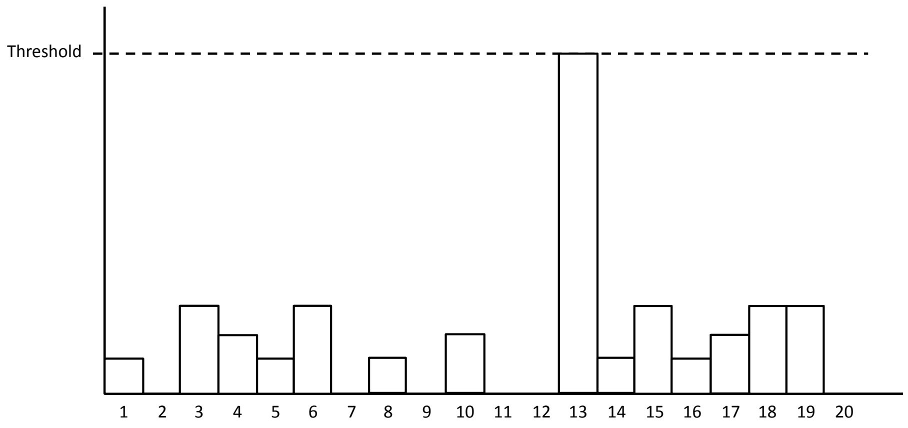


Figure 4.2 Example of bit synchronization histogram

4.2 Bit / Symbol Decision

In this part, the receiver makes the decision of which data bit / symbol was received. The decision is simple to make - majority of samples determines the bit / symbol. For GPS L1 there is 20 samples per bit. When adding these samples from the beginning of a data bit, the sum will be from 0 to 20. If the sum is greater or equal to 10, then the data bit is determined as +1. Otherwise, when the sum remains below 10, it is determined to be a 0 bit.

Galileo E1-B case is similar, but now there is four samples per symbol. When these samples are added, the sum is from zero to four. If the sum is greater or equal to two, then the data symbol is determined as +1. Otherwise, when the sum remains below two, it is determined to be a 0 symbol.

The determined data bits/symbols are then stored to memory for further processing. It is worth mentioning that for Galileo E1-B, the receiver may optionally utilize the tracking channels samples on a soft decision Viterbi decoder. In this case, all the Galileo E1-B symbol samples need to be stored. Viterbi decoders are described later in this chapter.

4.3 Subframe / Page Synchronization

Once the received bits start flowing from the baseband, the next step is to synchronize to the frame structure of the particular GNSS system. The synchronization is based on finding a particular preamble word from the beginning of a subframe (GPS) or a page part (even or odd) (Galileo). This process has two problems, which it must handle:

1. Bit/symbol polarity is still uncertain. In addition to the normal preamble, we also need to search for an inverted preamble. Finding correct inverted preamble means that the data bits/symbols need to be inverted to their correct polarity.
2. Finding the correct preamble, or the inverted one, is not trivial. The preamble pattern might randomly appear within the data too, making the process a bit more complicated. An easy solution is to check if the preamble pattern repeats with the subframe/page part rate. The subframe rate is 6 s for GPS L1, and page part rate is 1 s for Galileo E1-B. If the preamble appears to repeat with the subframe/page part rate, it can be declared as a correct preamble. In very unlucky condition, the selection can still go wrong, but in these cases, the error detection mechanisms will not pass, and the frame synchronization can be re-initiated.

The preambles are as follows:

GPS L1: [1 0 0 0 1 0 1 1]

Galileo E1-B: [0 1 0 1 1 0 0 0 0 0]

4.4 *Galileo Block De-Interleaving*

This section concerns Galileo E1-B data structure, and we describe what happens once the whole page part has been received. For Galileo E1-B the page part it is 250 symbols. One feature in Galileo message structure is the presence of block interleaving. Block interleaving scrambles the message sent by the satellite. The message is then exposed to errors while propagating through the atmosphere to the receiver. These errors tend to appear in bursts. When the message is received, the receiver performs the opposite action, called de-interleaving, which re-orders the original message, and de-scrambles the possible burst errors. The block interleaving, and its effect on error bursts is illustrated in Figure 4.3. The Galileo block interleaving dimensions (n columns and k rows) are 30x8, for the 240 symbols of E1-B message (preamble is not included).

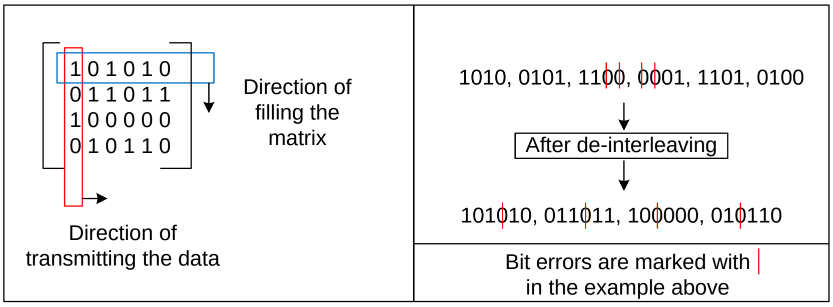


Figure 4.3 The effect of block interleaving on bit error bursts

4.5 *Galileo Compensate Inverter*

Next step after Galileo de-interleaving is to compensate for the inverter effect on the G2 branch of the signal, as shown in Figure 4.4. This is simply accomplished by inverting every second symbol of the received page part. The reason the inverter exists in this structure is not mentioned in the Galileo ICD. One assumption for its existence is that it helps the receiver to detect bit changes in case the transmitted data would for some reason be long periods of constant ones or zeros. In such a case, every other bit would be inverted and the detection of bit edges would still be easy.

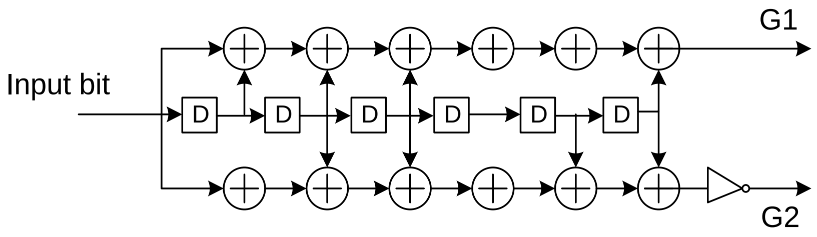


Figure 4.4 Galileo convolutional coding scheme

4.6 Galileo Viterbi Decoding

As mentioned in the previous section, Galileo E1-B utilizes block interleaving to reduce the occurrence of sequential errors in the message. This is beneficial since Galileo uses Forward Error Correction (FEC) encoding to the message. The convolutionally coded message is decoded with a standard Viterbi decoder [76]. The Viterbi decoder is able to correct error bursts of length t.

$$t = \left\lfloor \left(d_{free} - 1\right) / 2 \right\rfloor \tag{4.1}$$

where d_{free} is maximum free distance of the code. For Galileo d_{free} = 10. Based on this equation a Galileo receiver is able to correct error bursts of 4 bits [77], [78]. The implementation of the Viterbi decoder has two options, either a hard decision or soft decision Viterbi decoder. A hard decision Viterbi decoder operates on the symbols, which we determined earlier in Chapter 4.2. A soft decision Viterbi decoder operates on the separate samples from tracking loop. For mass-market receivers the use of hard decision Viterbi decoder is considered sufficient. It is less demanding from the implementation point of view, and still provides excellent performance.

A page part consists of 250 symbols, out of which 10 symbols are for the preamble pattern. The preamble pattern is not encoded, but the remaining 240 symbols must be decoded. From the practical point of view, it is not feasible to implement a Viterbi decoder directly for 240 symbols. Instead, literature recommends using the Viterbi decoder in block mode that operates with traceback length of 5 x the constraint length. Galileo constraint length is 7, so traceback length of 35 would suffice. Since 240 is not a multiple of 35, a slight modification to the decoder is recommended. The closest options are to use traceback depth of 30 or 40 for the decoder. In this way, the decoder does not have to process any partial data blocks. Choosing depth of 30 is cheaper to implement, but might slightly underperform in weak signal conditions. On the other hand choosing depth of 40 is certainly sufficient, but also more expensive to implement.

The Galileo Viterbi decoder always starts from zero state. This is also the reason why every Galileo page part has six zero bits at the end, called the tail bits. These tail bits enable the decoder to run continuously without explicitly resetting the states.

4.7 Parity / CRC Check

Galileo message structure includes block interleaving and FEC for correcting several bit errors. The benefits are obvious, but it also comes with a price: the receiver has to do more computational work for these features. In the very low signal strength situation some errors can still remain. To handle the remaining errors, Galileo utilizes Cyclic Redundancy Check (CRC), and GPS utilizes parity bits, to notice these remaining errors.

4.7.1 Galileo CRC

Galileo message structure has a CRC checksum included into the navigation data, which acts as a final check that no bit errors are present in the decoded data. The CRC checksum is calculated without the synchronization bit pattern and the tail bits. The 24-bit CRC for Galileo is specified by:

Generator polynomial: $G(X) = (1+x)*P(X)$, where

$P(X) = X^{23}+X^{17}+X^{13}+X^{12}+X^{11}+X^{9}+X^{8}+X^{7}+X^{5}+X^{3}+1$

$R(X)$ is defined as the remainder of $G(X)$ divided by $m(X)*X^{24}$, where

$m(X) = m_k + m_{k-1}X + m_{k-2}X^2 + \ldots + m_1X^{k-1}$, and

$m_1, m_2, \ldots ,m_k$ represent the Galileo information bits.

The final CRC checksum will be composed of the coefficients of R(X). Implementation of the CRC algorithm is easily done utilizing the xor-operator. For a received bit sequence of $m_{i,}$ the multiplication with X^{24} means in practice extending the sequence with extra zeros.

The first step is to start applying the xor-operation from the left bitwise with G(X) and m(X). After each round, the G(X) is shifted one step to the right, until it reaches the end of m(X). If m(X) has a zero in the leftmost place being xor'ed with G(X), replace G(X) with zero sequence for that round. When the algorithm stops, all that is left will be the remainder R(X), which is the CRC checksum. This value will be compared to the received value and a decision is made whether CRC check passes or fails. Figure 4.5 illustrates the CRC algorithm with an arbitrary example [79].

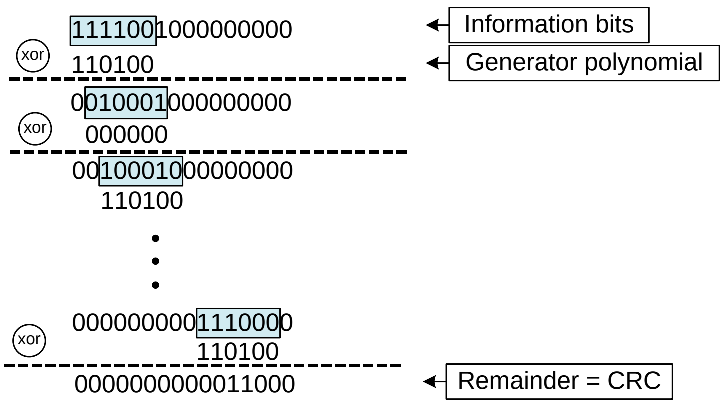


Figure 4.5 Example of the CRC algorithm

4.7.2 GPS Parity Check

The GPS parity check enables the receiver to detect errors in the received message structure. Every GPS subframe is 300 bits long, and is divided into shorter words consisting of 30 bits, as shown in Figure 4.6. Within each word, bits 1-24 are considered as data bits, while bit 25-30 are parity bits.

The actual parity check algorithm operates on complete words. It should also be noted that the algorithm requires two last bits, i.e. bits 29 and 30, from the previous word. The algorithm itself is quite straightforward, and is implemented based on the equations presented in Table 4.1.

The last step of the algorithm is to check whether the calculated parity bits match to the received parity bits. When there does not appear to be any parity mismatch, the parity check passes and the word can be processed. If there are parity mismatches, the whole word will be discarded.

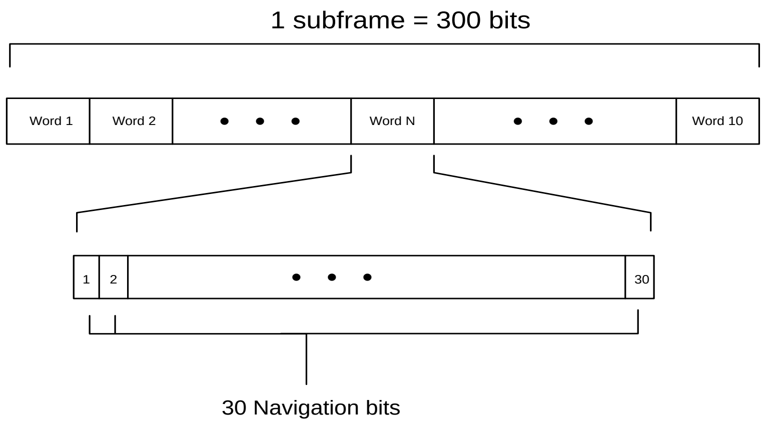


Figure 4.6 The structure of a GPS subframe

Table 4.1 GPS parity check equations

$D_1 = d_1 \oplus D_{30}^*$

$D_2 = d_2 \oplus D_{30}^*$

.

.

.

$D_{24} = d_{24} \oplus D_{30}^*$

$D_{25}=D_{29}^* \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_{10} \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{17} \oplus d_{18} \oplus d_{20} \oplus d_{23}$

$D_{26}=D_{30}^* \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{18} \oplus d_{19} \oplus d_{21} \oplus d_{24}$

$D_{27}=D_{29}^* \oplus d_1 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_8 \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{19} \oplus d_{20} \oplus d_{22}$

$D_{28}=D_{30}^* \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{20} \oplus d_{21} \oplus d_{23}$

$D_{29}=D_{30}^* \oplus d_1 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_9 \oplus d_{10} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{18} \oplus d_{21} \oplus d_{22} \oplus d_{24}$

$D_{30} = D_{29}^* \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus d_{13} \oplus d_{15} \oplus d_{19} \oplus d_{22} \oplus d_{23} \oplus d_{24}$

Where,
$d_1, d_2, ..., d_{24}$ are source data bits;
the symbol * identifies the last 2 bits from previous subframe;
$D_{25}, D_{26}, ..., D_{30}$ are computed parity bits;
$D_1, D_2, ..., D_{30}$ are bits transmitted by the SV;
the symbol $\oplus$ is modulo-2 operator;

4.8 Data Collection

Once the navigation data message has passed through the error detection mechanisms, it can be safely used. For Galileo, one remaining issue is to combine even and odd page parts into a complete message.

The receiver then processes pages for Galileo, and subframes for GPS. The layout of the page/subframe structures are found in the corresponding ICDs. The interesting parameters are:

- Timing information, TOW
- Ephemeris data
- Clock/Ionosphere correction terms
- Health/status parameters
- Optional data, e.g. Almanac data

The parameters usually remain constant for several hours (with exception of timing information, TOW), so it is not necessary to update the information constantly. For this reason, an Issue of Data Ephemeris (IODE) parameter is provided. If the IODE value changes, it indicates that new ephemeris data is available and the receiver should proceed with updating the parameters. Clock correction updates are similarly indicated with an Issue of Data Clock (IODC) parameter.

After successfully receiving all the necessary data for several satellites, the receiver can proceed with navigation solution calculation.

4.9 *Data Decoding Thresholds*

A stand-alone GNSS receiver without the ability to receive navigation data bits via A-GNSS, has to rely on its own processing capabilities. As receiver signal power gets weaker, data bit errors start appearing. After certain thresholds the data bit errors start to dominate, and it becomes, if not impossible, but at least very slow process to receive full subframes without data bit errors.

The properties of the GNSS signal play significant role in determining how weak signals can still provide the necessary navigation data. In [80] a list of contributing factors is identified:

1) Minimum received power level
2) Amount of power dedicated to the data channel (possible split with pilot)
3) Symbol rate of signal
4) Error protection techniques deployed (FEC, interleaving)
5) Channel conditions

The facts for various signals were listed in Tables 1.1 – 1.9. Minimum received signal power directly sets the threshold for the particular signal. It is natural that a signal with high received minimum power has advantage in data decoding thresholds. All other values of the list are directly influencing the decoding threshold. Amount of power dedicated to the data channel is a straightforward term. If all power is dedicated to data channel, i.e. no pilot channel, then there is no loss. If pilot channel shares 50% of the signal power, that is equivalent to 3 dB loss in the data decoding threshold. The third factor is explained with the fact that the slower the symbol rate is the more power there is available for a symbol. More power per symbol reduces the probability of getting a bit error. In other words, slower symbol rates have advantage in data decoding thresholds. Fourth and fifth factors are related in such sense that the error protection technique strongly depends on the type of channel considered. In reference [80] Additive White Gaussian Noise (AWGN) and more realistic Land Mobile Satellite (LMS) channel models are considered. The simulation results include error protection performance for all the GPS and Galileo open service signals. The data decoding thresholds for AWGN channel are presented in Table 4.2.

Table 4.2 Data demodulation performance of GPS and Galileo signal in AWGN channel [80]

	Demodulation threshold @ CED error rate = 10^{-2}					
	GPS				Galileo	
	L1C/A	L2C	L5	L1C	E1 / E5b	E5a
Message	NAV	CNAV	CNAV	CNAV2	I/NAV	F/NAV
Total required CN0 [dB-Hz]	26.5	23.1	26.1	24.5	27.7	20.7

5. MULTI-CONSTELLATION ISSUES IN THE NAVIGATION SOLUTION

In this chapter, the issues that the navigation software must handle in a multisystem GNSS receiver are discussed. For the following topics the chapter is mainly focusing on GPS and Galileo systems.

GPS related topics have been included in order to provide a benchmark for the Galileo discussion. The chapter is started with an introduction to the basic differences that can be expected when combining multiple GNSS systems in the receiver, in contrary to using a single GNSS system. After the short explanation on the system differences, a more detailed look is taken into the tasks, which the navigation software processes. The details start with timing topic, i.e. synchronization to GNSS time. Since both GPS and Galileo use their own time reference, the receiver must make the choice on which GNSS time to synchronize.

After successful initial timing synchronization, the receiver must make periodic measurements, named the pseudorange measurements. These pseudorange measurements need accurate timing information from the baseband tracking loops. The process of constructing the pseudorange measurements for both systems will be described.

In multi-GNSS receivers, the availability of satellites is notably increased. The number of satellites will be very high in open sky conditions. For such a high availability scenario, this chapter presents various criteria for selecting a good subset of satellites from the available satellites, instead of simply using all of them. By utilizing a subset of good satellites, the computational burden will be decreased without reducing the accuracy of the final solution.

The algorithm for computing the satellite position, velocity and acceleration is included in the chapter. Additionally, the Least-Squares (LS) solution for solving the final PVT is being looked at in this chapter. As before, for these algorithms the focus will be in the modifications necessary for the multisystem support.

5.1 GNSS Time Synchronization

When a receiver is turned on for the first time, it has no previous knowledge of its current location or accurate timing information. In such a situation, no almanac data is available either. This is called as a cold start. The first task for the navigation software is to recover a rough estimate of the current time - a process called synchronization to the GNSS time.

In a combined GPS / Galileo receiver, the question is whether the receiver should synchronize to the GPS time, or to the Galileo time. In practice, the choice does not matter much since both systems obviously provide excellent timing accuracy. Furthermore, after the first PVT solution when the user clock bias is solved together with the GPS-to-Galileo time offset, the receiver has essentially synchronized to both of the systems. Regardless of the choice of which system to synchronize to, the basic approach is to select one good candidate satellite from the list of tracked satellites. A good candidate here means usually the satellite with highest CN0 value. CN0 estimates are computed by the baseband, which provides this information to the navigation software [81]. At this point, the receiver does not yet have any information about the satellite location, so the elevation angle is also unknown. Generally high elevation angle would indicate a good candidate for satellite, but since the receiver does not have this information, the selection of the candidate satellite is primarily based on the CN0 estimates.

Table 5.1 GNSS constellations

System	**GPS**	**Galileo**	**GLONASS**	**BeiDou**
Nominal satellites	24 MEO	24 MEO	24 MEO	27 MEO 3 IGSO 5 GEO
Orbital planes	6	3	3	3
Inclination	55°	56°	64.8°	55°
Semi-major axis	26550 km	29600 km	25440 km	27900 km
Average flight altitude	20180 km	23222 km	19100 km	21528 km
Orbital period solar days	11 h 58 min	14 h 7 min	11 h 15 min	12 h 53 min
Orbital period sidereal days	1 / 2	10 / 17	8 / 17	7 / 13

The GNSS satellites have well-defined orbits, as described in Table 5.1. This allows the receiver to accurately approximate the signal travel time, used for the timing synchronization. The average signal travel time is selected to be 72 milliseconds. When multiplied by the speed of light, this time corresponds to a distance of 21600 km between the user and satellite, which is an estimated range of a GPS / Galileo satellite at average flight altitude. An illustration of the differences in the GNSS orbits is presented in Figure 5.1.

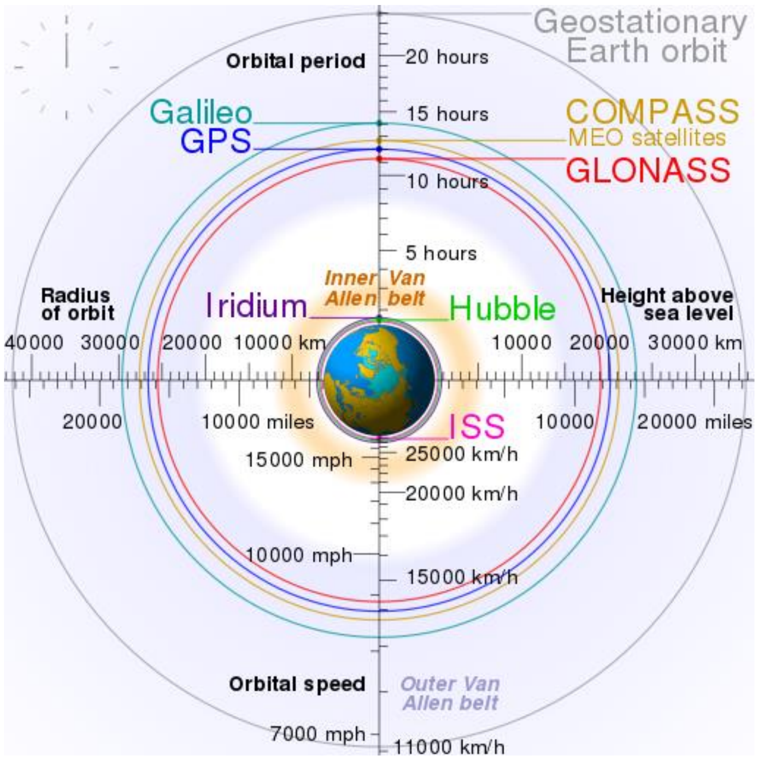


Figure 5.1 GNSS orbits comparison [82]

In next section, the equations for both GPS and Galileo synchronization are provided.

5.1.1 GPS Time Synchronization

$$T_{GPS} = T_{TOW} + T_{GPS\_code} + T_{GPS\_chip} + 0.072\,\mathrm{s} \tag{5.1}$$

where T_{TOW} is the current TOW count decoded from the navigation data bits. In GPS case, the TOW is found in the beginning of each subframe, thus being updated every 6 seconds. The TOW reports how many seconds has elapsed since the start of the current GPS week. By the definition, both GPS and Galileo weeks start in the midnight between Saturday and Sunday. GPS week numbering was started on 6th of January 1980, whereas Galileo week numbering was started on 22nd August 1999 [13], [65].

T_{GPS_code} is the integer number of full GPS C/A-code epochs since the beginning of current subframe. In GPS case the C/A-codes are 1 ms long. The integer number of milliseconds is converted to seconds in the equation.

T_{GPS_chip} is composed of 2 parts. The first part is the integer number of full code chips in the current GPS C/A-code (0-1022). The second part is the fraction of the current chip. The fraction information is provided by the baseband tracking loops. This combined timing information is then similarly converted into seconds.

5.1.2 Galileo Time Synchronization

$$T_{Galileo} = T_{TOW} + T_{Gal\_code} + T_{Gal\_chip} + 0.072\,\mathrm{s} \tag{5.2}$$

where T_{TOW} is the current TOW decoded from the navigation data. In Galileo I/NAV messages TOW is recovered at nominal page rate of 2 seconds. This is a faster rate than the GPS TOW update rate, which was every 6 seconds.

T_{Gal_code} is the integer number of Galileo codes since the start of a current page. In E1-B signal case, code length is 4 ms. This is converted into seconds in the equation.

T_{Gal_chip} is composed of 2 parts. The first part is the integer number of code chips (0-4091), while the second part is the fraction of the code chip, obtained from the baseband tracking loops. Similarly, this is converted to seconds in the equation.

5.1.3 Time Synchronization - Conclusions

As the previous synchronization Equations (5.1) and (5.2) suggest, the GNSS time is composed of four distinct components. Additionally, a fifth parameter called Week Number (WN) is included in the GNSS time keeping. Next paragraph concludes the significance of each timing component.

- First part of the GNSS time is Week Number, which is, at minimum, needed to handle the crossover of the week boundaries. The WN is decoded from the navigation message.
- Second major part of the composite time is the TOW, which is also decoded from the navigation data. The TOW counts full seconds, 0 – 604799, since the beginning of the current GNSS week. TOW is reset to zero at the end of each week.
- Third component is the integer number of C/A-codes elapsed since the beginning of current subframe. This ranges between 0 – 6 seconds in GPS case, and between 0 – 2 seconds in Galileo I/NAV case.
- Fourth time component is the fraction of a code chip. This part ranges between 0 – 1 millisecond in GPS case, and between 0 – 4 milliseconds for Galileo. Now this part of composite time has already a lot smaller impact on the initial synchronization time.
- The last component of the time is the approximate signal travel time, for which 72 milliseconds was selected. In practice, the true travel time might be several milliseconds different from the chosen one. This explains why the code fraction component does not play a major role in the coarse time synchronization.

The described GNSS synchronization technique should provide a rough estimate of the GNSS time. This initial estimate is expected to be within several milliseconds from the true GNSS time. Time estimate error of several milliseconds will still provide sufficient accuracy for the PVT solution, which allows further fine-tuning of the time by applying the clock bias correction. Once the clock bias is corrected, the receiver is accurately synchronized to the GNSS time.

5.1.4 Time synchronization – Multi-GNSS

GPS and Galileo time scales are handled in a very similar manner in the receiver. The difference between these timescales can be solved in the navigation solution, or its estimate can be decoded from the Galileo navigation data. This topic will be discussed later in Section 5.11.

GLONASS time scale differs from GPS and Galileo in such sense that GLONASS time does include leap seconds. In other words, GLONASS time differs from GPS and Galileo time by the number of leap seconds, which is 18 seconds as of January 2017. BeiDou time scale does not include leap seconds, but the time scale was initiated with the amount of leap seconds as of January 2006. In other words, BeiDou has constant offset of 14 leap seconds to GPS and Galileo time scales.

5.2 Satellite Selection for Navigation Solution

The modern GNSS receivers can track many satellites simultaneously. Once Galileo and BeiDou reach full constellation of satellites, it is possible to have a situation where the receiver could be tracking 30+ GNSS satellites. From the navigation solution point of view, this abundance of satellite signals has its pros and cons. The benefit of having many satellites tracked comes from signal diversity. With many satellites, it is more likely to have a good satellite geometry, leading to a more accurate navigation solution. This topic is discussed more in Section 5.12. With many satellites, it is easier to exclude satellites with weak signal from the navigation solution. Favoring good quality satellite signals ensures better accuracy of the navigation solution. On the other hand, a large number of satellites being tracked has its downside, which comes from the increased computational demands of the receiver. Extra satellites occupy tracking channels of the receiver, and the navigation solution needs to perform more computations with the increased number of satellites.

As conclusion, there can be redundancy in the available signals, even to the extent where it is perfectly ok not to use some of the satellites in PVT solution. On the other hand, in a challenging environment like the urban canyon, the availability of satellites is still going to remain low. Another issue with the urban canyon scenario is that the geometry of the satellites is likely to be quite bad.

5.2.1 Listing Accepted Satellites

A conventional single system receiver can operate with a minimum of 4 satellites accepted for the navigation solution. In general, this requirement will change for a dual system receiver where a minimum of 5 satellites is needed for the solution. The requirement of the

5th satellite comes from the fact that the two satellite systems are not perfectly synchronized. The recommended method for the dual system navigation solution is to use 5 satellites, e.g. containing signals from both GPS and Galileo systems and solve the time difference between the system times. It is also good to remember that a dual system receiver can always operate in a single system mode with a minimum of 4 satellites from a single system, e.g. in a case where no satellites are available from the other system.

Now the minimum number of satellites for a dual system navigation solution has been defined. Another topic is to consider the effect of including additional signals to the PVT solution. Adding more signals might first sound like a very good idea, but in practice, the benefits of extra signals quickly become negligible. A rule of thumb would be to limit the number of selected signals around 8. According to our test results, adding more satellites beyond 8 does not significantly improve the accuracy of the PVT solution. When the best subset of satellite signals is selected, any additional satellite signals will likely increase the computational burden without significantly improving the overall positioning quality. The accuracy issue can be explained with the Dilution of Precision (DOP) parameter, which is calculated in the navigation solution. The DOP value describes how good is the geometry that the selected satellites provide. This geometry factor is a direct indicator of the expected accuracy of the navigation solution, and it is not significantly improved by additional satellites beyond the rule of thumb 8 satellites. Section 5.12 describes the DOP contribution and computation in greater detail.

5.2.2 Satellite Selection Criteria

A satellite can be considered usable for the navigation solution once it passes certain criteria. There are both mandatory and optional criteria. A short list of mandatory criteria:

- Full set of valid ephemeris data must be obtained
- Satellite health status decoded, and indicates healthy satellite
- Accuracy estimate from navigation data is below user defined threshold
- Synchronization to the TOW
- Code phase measurements availability

Some of these criteria are specified in the respective ICDs. The details of the mandatory criteria are as follows.
First, the signal must be tracked for a certain time in order to decode all the necessary ephemeris and clock correction data. The ephemeris data is needed for the satellite position computation. Details of the ephemeris data are found in Section 5.6, and satellite position computation is found in Section 5.7. GPS and Galileo navigation data rates and data layout are quite different and low-level details are found in the respective ICDs [13], [65].

Second, the satellites provide a health status indicator flags, which are used to warn if any of satellite signals appear unhealthy. Naturally, an unhealthy signal should not be used.

Third, the accuracy estimates must be reasonable. For GPS, the accuracy estimate is called User Range Accuracy (URA), and equivalently for Galileo it is Signal in Space Accuracy (SISA). These values are estimated by the control segment, and account for the estimated total error contribution for which the space and control segments are responsible. A check must be made that the indicated value is at acceptable levels. Naturally, satellites which indicate high-expected error in the signal are to be avoided. The functionality of these values is the same for both systems. Complete list of URA / SISA values is presented in [13], [65].

Fourth, synchronization to the TOW. Synchronization means that the receiver has found the preamble pattern, synchronized to the frame structure, and has been able to retrieve the TOW value from the navigation data. Usually the TOW decoding step is closely related to the decoding of ephemeris data. Eventually the TOW value is needed for pseudorange construction.

Fifth, the availability of code phase measurements. The code phase measurements will be available from the baseband once the tracking loops have converged and stabilized. The code phase measurements are likewise needed for the pseudorange construction.

Once the mandatory criteria are met, further ranking of the satellite signals can be made with optional criteria:

- Elevation angle
- CN0 estimate
- BER
- Pseudorange noise estimate
- DOP contribution of the chosen satellite

The elevation angle is a very significant value for determining the usability of a satellite. It indicates how long distance the signal must travel in the Earth atmosphere. The lower the elevation angle, the longer the distance traveled in the atmosphere. Longer distances will induce greater variation to the signal travel time due to the troposphere and ionosphere delays. An elevation mask can be used to discard low-elevation satellites from the navigation solution. Typical elevation mask values are around 5-15 degrees. The elevation angle is unknown until the ephemeris data for calculating the satellite position has been received. In addition, the receiver location should be known. When the receiver is turned on for the first time, it should have a mechanism to handle the initial PVT calculation. If receiver position is unknown it is not possible to compute elevation angle, thus elevation mask criteria could be ignored in the first position fix.

CN0 estimate is based on the accumulated I/Q samples from the baseband. It gives a simple estimate of the overall quality of the satellite signal. If some satellites have considerably lower CN0 estimates, it is a good reason to avoid those satellites in the navigation solution. In general, if the receiver is not able to track many satellites, then is may be necessary to include satellites with low CN0 to the navigation solution. This is to enable the position fix. If a position fix is done with low quality satellites, it should be clearly flagged as a fix with bad satellites. In the end, the CN0 threshold can be set dynamically, depending on the availability of satellite signals.

Bit Error Rate (BER) is obtained from the navigation data decoding, and it is very similar indicator to the CN0 estimate. BER indicates how often bit errors occur for certain signal.

The bit errors cause the CRC checks on the data frames to fail. These erroneous frames will be rejected, causing the risk of not being able to notice critical updates in the ephemeris data / signal health status indicator. As conclusion, satellites signals with high BER are to be avoided.

Pseudorange noise estimation can be done either in the navigation software, simply by observing the continuous pseudorange measurements over a time window, or in the baseband, where the variance of the code loop discriminator is estimated. If significant variation is noticed in the measurements, it will indicate that the baseband tracking loop is not able to track the signal perfectly. Once again, a high value in pseudorange noise estimation is a good reason to avoid using such a satellite signal.

Once the receiver has made an initial PVT solution with available satellites, the position of the receiver is known. With the position information, it is possible to calculate the DOP value for different combinations of satellites. The DOP value indicates how good accuracy the chosen satellites can provide. The DOP value can be solved from the geometry matrix, which is introduced in Section 5.11. The DOP value can be used as one factor in reducing the number of satellites without significantly decreasing the accuracy of PVT solution.

5.2.3 Listing Low-Quality Satellites

The previous section listed many factors how to evaluate the quality of a satellite signal. Sometimes it may be useful to make a list of the low-quality satellites. Such a list is useful for saving resources in the receiver. For example, if a satellite is constantly causing problems, or is known to be unhealthy, it can be added to the list. Information about the low-quality satellites can then be used in the baseband. The baseband can then stop the tracking and acquisition of such satellites for a certain period, thus freeing the channel for other satellites, or just to save power in the baseband. It should be kept in mind that from cross-correlation perspective all visible satellites should be always tracked, so that traditional cross-correlation detection mechanisms can work.

5.3 Code Phase Pseudorange Measurements

The code phase pseudoranges are the fundamental measurements of a standalone GNSS receiver made by tracking the phase of the received signal PRN code. Code phase measurements are ambiguous for the length of a code, which would correspond to 1 ms for C/A-codes. Code phases can be constructed to unambiguous pseudoranges by combining information from the navigation data structure. The pseudoranges are usually measured with respect to the beginning of the latest received subframe. A time parameter is broadcast in the navigation message, indicating the time at the beginning of a subframe. After the beginning of the frame, the receiver counts how many full C/A-codes have been received. Finally, the code phase measurement represents the fractional part of the current C/A-code. The process of combining this information is presented in the following paragraph.

The pseudorange represents the measured distance between the satellite and the receiver. It is not an exact range, due to unknown receiver clock bias and other errors sources. The GNSS receivers usually have cheap crystal oscillator clocks, which are not very accurately synchronized to the GNSS time, and they are not very stable. The receiver clock can be compensated with the clock bias that is solved in the PVT solution. The other measurement errors, caused by, e.g. control segment or atmospheric errors, cannot be immediately accounted for, and thus remain in the pseudorange measurement. In order to minimize the contribution of these errors, the criteria presented previously in Section 5.2.2 should be used for selecting the most reliable set of satellites.

Pseudoranges are composite measurements, combined from different variables in the receiver. One common way of making pseudorange measurements is to define certain points of GNSS time when measurements are made for all tracked satellites simultaneously. In the following, *t* is the common time of taking the pseudorange measurements. Then pseudorange measurements will be defined as follows:

$$\rho_i(t) = c \cdot \left[t_u(t) - t_i(t - \tau_i) \right] \tag{5.3}$$

where

subscript i	marks satellite i,
$\rho_i(t)$	pseudorange measurement for satellite i,
c	the speed of light,
$t_u(t)$	receiver generated time value.
τ_i	the transit time

$$t_i(t-\tau_i) = t_i^{TOW} + t_i^{ms} + t_i^{chip} \tag{5.4}$$

t_i^{TOW}	latest TOW received from the navigation data,
t_i^{ms}	total number of C/A-codes passed since the last frame, expressed as integer multiple of milliseconds,
t_i^{chip}	fraction of current C/A-code.

Looking at the last components of the pseudorange measurements, 3 main differences between GPS and Galileo pseudorange construction can be noticed.

First, the TOW count is received every 6 seconds for GPS C/A and for Galileo I/NAV messages TOW is received every 2 seconds.

Second, the number of C/A-codes received since the beginning of the frame. For GPS case, each GPS C/A-code adds 1 millisecond to the value, whereas for Galileo each C/A-code adds 4 milliseconds.

Third, the fraction of the current C/A-code. For GPS these values will be between 0 – 1 ms, and for Galileo between 0 – 4 ms.

As conclusion, pseudoranges are generated for all channels simultaneously, meaning both GPS and Galileo satellites, at the same internal time t.

The receiver clock at time t is showing $t_u(t)$. The receiver time can be synchronized to either GPS or Galileo time, since the difference is negligible. At time instant t, the real measurement information is given by the baseband in the three aforementioned parameters: the TOW parameter provides the coarse estimate of the time; the number of C/A-codes provides a time component with more accuracy; and finally the last component, the phase of the C/A-code, provides a time component of the finest precision.

These three levels of timing accuracy have analogy to an analog clock, where the TOW count represents the hour hand of a clock. The number of C/A-codes represents the minute hand, and the fraction of C/A-code measurement represents the second hand. This is a very useful analogy in understanding how the pseudorange timing is composed of these parameters [83].

5.4 Carrier Phase Measurements

The code phase pseudorange measurements have the good property of being unambiguous; thanks to the timing information retrieved from the navigation data. The receiver is also able to continuously track the carrier phase of the signal. The carrier phase measurements are much less noisy, and they would provide better accuracy measurements. There is just a slight problem that the carrier phase measurements are ambiguous measurements, and there is no easy method for solving the ambiguity. Disregarding the clock bias and measurement errors, the carrier phase measurement can be represented (in units of cycles) as:

$$\phi(t) = \phi_u(t) - \phi_s(t-\tau) + N, \tag{5.5}$$

Where

$\phi_u(t)$ is the phase of the receiver-generated signal

$\phi_s(t-\tau)$ is the phase of the signal received from the satellite at time t

τ is the transit time

N is the integer ambiguity.

The problematic terms here is the integer ambiguity, which corresponds to the full cycles of the carrier wave between the satellite and the receiver. While there are methods for solving the integer ambiguity [84], [85], [86], it is easier to measure only the changes in the carrier phase measurement. While carrier tracking is continuous, the carrier phase measurements offer precise and unambiguous measurements of change in pseudorange. This is often referred to as integrated Doppler or delta pseudorange. The next section will show how to utilize carrier phase measurements.

5.5 Combining Code and Carrier Measurements

In this section, the precise, but ambiguous carrier phase measurements will be combined with the noisy, but unambiguous code phase pseudoranges. Before deriving the algorithm, it is important to mention a few relevant properties of the Earth atmosphere. The Earth atmosphere consists of ionosphere and troposphere regions, both being refractive mediums. A refractive medium bends the signal travel path, thus making it travel a longer distance than a direct line would have been. This causes an error when making the measurements. Besides both regions being refractive, the ionosphere region is also a dispersive medium. In a dispersive medium, signal bending is frequency dependent. These errors are discussed more in the following sections. How this all affects our carrier and code measurements, is that troposphere will delay both measurements similarly, while ionosphere will introduce code phase delay, and carrier phase advance. Code delay and carrier advance have the same magnitude, but a different sign. This phenomenon is called code-carrier divergence [32]. With this in mind, the algorithm for combining code and carrier measurements will be derived.

$$I_\rho(t) = -I_\phi(t), \;\; T_\rho(t) = T_\phi(t)$$

Let us define: $I_\rho(t) = d_I(t)$ and $T_\rho(t) = d_T(t)$

Pseudorange and carrier phase measurement will be (in units of length):

$$\rho(t) = r(t) + c(\Delta t_u(t) - \Delta t^s(t-\tau)) + d_I(t) + d_T(t) + \varepsilon_\rho(t) \tag{5.6}$$

$$\Phi(t) = \lambda\phi(t) = r(t) + c(\Delta t_u(t) - \Delta t^s(t-\tau)) - d_I(t) + d_T(t) + \lambda N + \varepsilon_\phi(t) \tag{5.7}$$

Let us define ionosphere-free pseudorange $\rho^*(t)$

=>

$$\rho^*(t) = r(t) + c(\Delta t_u(t) - \Delta t^s(t-\tau)) + d_T(t) \tag{5.8}$$

Now pseudorange and carrier phase measurements can be rewritten as:

$$\rho(t) = \rho^*(t) + d_I(t) + \varepsilon_\rho(t) \tag{5.9}$$

$$\Phi(t) = \rho^*(t) - d_I(t) + \lambda N + \varepsilon_\phi(t) \tag{5.10}$$

The change in the code and carrier phase measurements between two measurement epochs can be written as:

$$\Delta\rho(t_i) = \rho(t_i) - \rho(t_{i-i}) = \Delta\rho^*(t_i) + \Delta I(t_i) + \Delta\varepsilon_\rho(t_i) \tag{5.11}$$

$$\Delta\Phi(t_i) = \Phi(t_i) - \Phi(t_{i-i}) = \Delta\rho^*(t_i) - \Delta I(t_i) + \Delta\varepsilon_\phi(t_i) \tag{5.12}$$

From the above equation, two important observations can be made:

- Ionosphere delay has opposite sign in (5.11) and (5.12)
- The ambiguity term disappears in (5.12)

The concept of the combination of unambiguous code measurements with the ambiguous carrier phase measurements is explained next. With the help of code-based pseudoranges we solve the initial ambiguity, and start to smoothen the code measurements with the carrier measurements.

Initial estimate of $\rho(t_0)$ is available from each epoch as:

$$\hat{\rho}(t_0)_i = \rho(t_i) - [\Phi(t_i) - \Phi(t_0)] \tag{5.13}$$

These estimates can be averaged over n epochs:

$$\bar{\rho}(t_0) = \frac{1}{n}\sum_i \hat{\rho}(t_0)_i \tag{5.14}$$

The final combination of code and carrier measurements is done with the help of a specific recursive filter called the Hatch filter [87].

A recursive filter of length M:

$$\bar{\rho}(t_i) = \frac{1}{M}\rho(t_i) + \frac{(M-1)}{M}[\bar{\rho}(t_{i-1}) + (\Phi(t_i) - \Phi(t_{i-1}))] \tag{5.15}$$

$$\bar{\rho}(t_0) = \rho(t_0)$$

A typical filter length could be M = 100. How the filter actually works is that it is initialized with M = 1, and then afterwards M gets incremented by one each round until reaching 100. When M reaches its maximum value, it will no more be incremented and the filter has reached a steady state operation. The filter design is quite flexible about the maximum length, and how big is the increment each round. Bigger increments reach the steady state faster, but can have problems resolving the ambiguity accurately. When utilizing carrier smoothing of code phase measurements, it is important to realize that any interruption in the carrier tracking, e.g. a cycle slip, will result in loss of accumulated phase information between the measurements. In such event, the filter must be reset, by initializing M back to 1. Another issue worth considering is the code-carrier divergence. In long continuous tracking the code-carrier ionosphere divergence will get grow significantly, and it is advised to reset the filter occasionally. For mass-market receiver operations, according to our experience, doing a filter reset roughly every 10 minutes ensures operation without significant degradation in accuracy due to code-carrier divergence [88]. Carrier-smoothed pseudoranges are significantly less noisy than the code based pseudoranges. Results are presented in Figure 5.2. When the respective measurements are plotted on normal scale, i.e. around 20 000 km, the few meters of noise will not be visible without significant zoom, as seen in Figure 5.2 top plot. The difference between the measurements is shown in the mid plot. Finally, the bottom plot presents the carrier-smoothed pseudorange rates vs. pseudorange rates in scale.

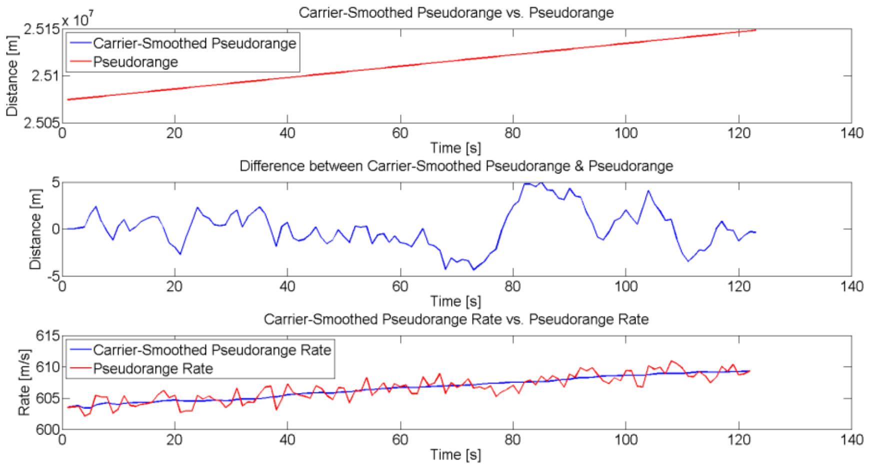


Figure 5.2 Carrier-smoothing of pseudoranges

5.6 Satellite Related Calculations

Satellite related calculations start once the receiver has been able to acquire and track enough of the GNSS signals. For those satellites, full set of ephemeris from the navigation data should have been obtained. The navigation data frame structure is quite different for GPS and Galileo, but it is very convenient that both systems support the same ephemeris data set for describing the satellite location on orbits at any given time. The ephemeris data is composed of Keplerian parameters listed in Table 5.2.

BeiDou MEO satellites will use similar Keplerian ephemeris data for satellite location calculations. On the other hand, GLONASS ephemeris data is completely different from the Keplerian ephemeris data. In GLONASS the ephemeris data consists of the satellite x, y, z position, velocity and acceleration parameters. From these, the position of the satellite is computed with an algorithm involving Runge-Kutta integrations, as presented in [89]. Finally, GLONASS satellites are referenced to PZ-90 coordinate frame, which should be converted to WGS-84, to be compatible with GPS.

The Table 5.2 lists the 6 basic Keplerian elements, 9 correction terms of time perturbations and 1 time epoch parameter. Additionally, there are 3 clock correction parameters and a navigation data timestamp IODE, which indicates any changes in the ephemeris data, as mentioned in Section 4.8. For an individual satellite, all used ephemeris must be time-stamped with same IODE. The algorithm for solving the satellite location at a given time uses a few additional constants, which are listed in Table 5.3.

Table 5.2 Ephemeris parameters

Parameter	Number of bits		Scale factor		Units
	GPS	GAL	GPS	GAL	
t_{0e}	16	14	16	60	seconds
C_{rs}	16		2^{-5}		meters
Δn	16^*		2^{-43}		semicircles/sec
M_0	32^*		2^{-31}		semicircles
C_{uc}	16^*		2^{-29}		radians
e	32		2^{-33}		unitless
C_{us}	16^*		2^{-29}		radians
$\sqrt{A}$	32		2^{-19}		meters$^{1/2}$
C_{ic}	16^*		2^{-29}		radians
Ω_0	32^*		2^{-31}		semicircles

C_{is}	16*	2^{-29}	radians
i_0	32*	2^{-31}	semicircles
C_{rc}	16*	2^{-5}	meters
ω	32*	2^{31}	semicircles
$\dot{\Omega}$	24*	2^{-43}	semicircles/sec
$\dot{i}$	14*	2^{-43}	semicircles/sec
Parameters indicated with * are two's complement numbers			

Table 5.3 Parameter definitions

Parameter	**Value**
t_u	User receiver time. Generated in the receiver.
ρ^s	Pseudorange measurement for satellite *s*. Generated in the receiver.
c	Speed of light. 299792458 m/s
μ	Earth's gravitational constant. Note difference in the respective ICDs. GPS: 3.986005e14 m^3/s^2 Galileo: 3.986004418e14 m^3/s^2
$F = \frac{-2 \cdot \sqrt{\mu}}{c^2}$	Minor difference between systems due to Earth's gravitational constant. GPS: -4.442807633e-10 $s/m^{1/2}$ Galileo: -4.442807309e-10 $s/m^{1/2}$
$\dot{\mathbf{\Omega}}_e$	Earth's rotation rate 7.2921151467e-5 rad/s

5.7 Satellite Location, Velocity and Acceleration

The complete algorithm for solving satellite positions is described step-by-step in this section. Parameters used in this algorithm are those that can be found in Table 5.2 and Table 5.3. Some of the parameters slightly differ between GPS and Galileo, but the algorithm still uses the same approach for both systems.

The last remaining issue in satellite location calculation is about the GPS and Galileo geodetic coordinate reference frames. Both systems use their own coordinate reference frames. For GPS the reference frame is WGS-84, and for Galileo, Galileo Terrestrial Reference Frame (GTRF). The differences between the two models are expected to be within a few centimeters, as both WGS-84 and GTRF are supposed to follow closely the International Terrestrial Reference Frame (ITRF). Therefore, this error is hidden by the higher satellite orbital and clock estimation errors in the ephemerides and its impact in accuracy is negligible for a standard navigation service. No further action is needed for correcting the coordinate differences in mass-market receivers [90], [91].

Table 5.4 Satellite Position, Velocity and Acceleration calculation [13], [65], [92]

Parameter	**Value**	**1st Time Derivative**	**2nd Time Derivative**
Coarse Time of Transmission	$t_c = t_u - \frac{\rho^s}{c}$		
Semimajor Axis	$\mathrm{A} = (\sqrt{\mathrm{A}})^2$		
Corrected Mean Motion	$\mathrm{n} = \sqrt{\frac{\mu}{\mathrm{A}^3}} + \Delta\mathrm{n}$		
Time from Ephemeris Epoch	$\mathrm{t_k} = \mathrm{t_c} - \mathrm{t_{oe}}$	$\mathrm{t_k}' = 1$	
Mean Anomaly	$\mathrm{M} = \mathrm{M_0} + \mathrm{n} \cdot \mathrm{t_k}$	$\mathrm{M}' = \mathrm{n}$	
Eccentric Anomaly*	$\mathrm{E} = \mathrm{M} + \mathrm{e} \cdot \sin(\mathrm{E})$	$\mathrm{E}' = \frac{\mathrm{M}'}{1 - \mathrm{e} \cdot \cos(\mathrm{E})}$	$\mathrm{E}'' = \frac{-(\mathrm{E}')^2 \cdot \mathrm{e} \cdot \sin(\mathrm{E})}{1 - \mathrm{e} \cdot \cos(\mathrm{E})}$

True Anomaly	$v = \mathrm{atan}(\frac{\sqrt{1-e^2}\cdot\sin(E)}{\cos(E)-e})$	$v' = \frac{\sqrt{1-e^2}\cdot E'}{1-e\cdot\cos(E)}$	$v'' = \frac{2v'\cdot E''}{E'}$
Argument of Latitude	$\Phi = v+\omega$	$\Phi' = v'$	$\Phi'' = v''$
Argument of Latitude Correction	$\delta u = C_{us}\cdot\sin(2\Phi) + C_{uc}\cdot\cos(2\Phi)$	$\delta u' = 2\Phi'\cdot C_{us} \cdot\cos(2\Phi) - 2\Phi' \cdot C_{uc}\cdot\sin(2\Phi)$	$\delta u'' = -4(\Phi')^2\cdot\delta u + \frac{\Phi''}{\Phi'}\delta u'$
Radius Correction	$\delta r = C_{rs}\cdot\sin(2\Phi) + C_{rc}\cdot\cos(2\Phi)$	$\delta r' = 2\Phi'\cdot C_{rs} \cdot\cos(2\Phi) - 2\Phi' \cdot C_{rc}\cdot\sin(2\Phi)$	$\delta r'' = -4(\Phi')^2\cdot\delta r + \frac{\Phi''}{\Phi'}\delta r'$
Inclination Correction	$\delta i = C_{is}\cdot\sin(2\Phi) + C_{ic}\cdot\cos(2\Phi)$	$\delta i' = 2\Phi'\cdot C_{is} \cdot\cos(2\Phi) - 2\Phi' \cdot C_{ic}\cdot\sin(2\Phi)$	$\delta i'' = -4(\Phi')^2\cdot\delta i + \frac{\Phi''}{\Phi'}\delta i'$
Corrected Argument of Latitude	$u = \Phi + \delta u$	$u' = \Phi' + \delta u'$	$u'' = \Phi'' + \delta u''$
Corrected Radius	$r = A\cdot(1 - e\cdot\cos(E)) + \delta r$	$r' = A\cdot e\cdot E' \cdot\sin(E) + \delta r'$	$r'' = A\cdot e\cdot(E')^2 \cdot\cos(E) + A\cdot e\cdot E'' \cdot\sin(E) + \delta r''$
Corrected Inclination	$i = i_0 + \delta i + \dot{i}\cdot t_k$	$i' = \dot{i} + \delta i'$	$i'' = \delta i''$
Corrected Longitude of Node	$\Omega = \Omega_0 + \left(\dot{\Omega} - \dot{\Omega}_e\right)\cdot t_k - \dot{\Omega}_e\cdot t_{0e}$	$\Omega' = \dot{\Omega} - \dot{\Omega}_e$	
Satellite x-Position in Orbital Plane	$x_p = r\cdot\cos(u)$	$x_p' = -r\cdot u' \cdot\sin(u) + r'\cdot\cos(u)$	$x_p'' = -(u')^2\cdot x_p - u''\cdot y_p - 2u'\cdot r' \cdot\sin(u) + r''\cdot\cos(u)$

Satellite y-Position in Orbital Plane	$y_p = r \cdot \sin(u)$	$y_p{}' = r \cdot u' \cdot \cos(u) + r' \cdot \sin(u)$	$y_p{}'' = -(u')^2 \cdot y_p + u'' \cdot x_p + 2u' \cdot r' \cdot \cos(u) + r'' \cdot \sin(u)$
ECEF x-Coordinate	$x = x_p \cdot \cos(\Omega) - y_p \cdot \cos(i) \cdot \sin(\Omega)$	$x' = -\Omega' \cdot y + \sin(\Omega) \cdot (z \cdot i' - \cos(i) \cdot y_p{}') + x_p{}' \cdot \cos(\Omega)$	$x'' = -\Omega' \cdot y' + \sin(\Omega) \cdot (z' \cdot i' - \Omega' \cdot x_p{}' + y_p \cdot i'' \cdot \sin(i) - y_p{}'' \cdot \cos(i) + i' \cdot y_p{}' \cdot \sin(i)) + \cos(\Omega) \cdot (x_p{}'' + y_p \cdot \Omega' \cdot i' \cdot \sin(i) - \Omega' \cdot y_p{}' \cdot \cos(i))$
ECEF y-Coordinate	$y = x_p \cdot \cos(\Omega) + y_p \cdot \cos(i) \cdot \cos(\Omega)$	$y' = \Omega' \cdot x + \cos(\Omega)(-z \cdot i' + \cos(i) \cdot y_p{}') + x_p{}' \cdot \sin(\Omega)$	$y'' = \Omega' \cdot x' - \cos(\Omega) \cdot (-z' \cdot i' + \Omega' \cdot x_p{}' - y_p \cdot i'' \cdot \sin(i) + y_p{}'' \cdot \cos(i) - i' \cdot y_p{}' \cdot \sin(i)) + \sin(\Omega) \cdot (x_p{}'' + y_p \cdot \Omega' \cdot i' \cdot \sin(i) - \Omega' \cdot y_p{}' \cdot \cos(i))$
ECEF z-Coordinate	$z = y_p \cdot \sin(i)$	$z' = y_p i' \cos(i) + y_p{}' \cdot \sin(i)$	$z'' = \sin(i) \cdot (-y_p \cdot (i)^2 + y_p{}'') + \cos(i) \cdot (y_p \cdot i'' + 2i' \cdot y_p{}')$

* Equation for eccentric anomaly. This equation cannot be solved in closed form, and thus iterative methods should be used for solving E. One suitable candidate is the Newton's method:

Newton’s method algorithm: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0,1,2,...$

=>

```
$E_n = M$
$for\ n = 0:10$
    $E_{n+1} = E_n - \frac{E_n - e \cdot \sin E_n}{1 - e \cdot \cos E_n}$
    $if\ |E_n - E_{n+1}| < \text{tolerance}$
        $E = E_{n+1}$                                        (5.16)
        $break$
    $end$
    $E_n = E_{n+1}$
$end$
```

The algorithm (5.16) will quickly converge to the correct value with a few iterations.

5.8 Earth Rotation Correction

The Satellite location has now been solved in the Earth Centered Earth Fixed (ECEF) coordinate frame with the algorithm presented in Table 5.4. There is one more important adjustment to be done with the coordinate frames: Earth rotation correction. The satellite signal will travel approximately 72 ms before it reaches the receiver antenna on Earth’s surface. During that time, Earth has rotated slightly around its axis. This effect is called the Sagnac effect. The compensation is done with a simple matrix rotation from ECEF to Earth

Centered Inertial (ECI) coordinate frame. Figure 5.3 illustrates the Sagnac effect. The algorithm is presented with Equations 5.17 and 5.18.

$$dt = \frac{\rho_s}{c} \tag{5.17}$$

$$\begin{bmatrix} x_{SV} \\ y_{SV} \\ z_{SV} \end{bmatrix} = \begin{bmatrix} \cos(\dot{\Omega} \cdot dt) & -\sin(\dot{\Omega} \cdot dt) & 0 \\ \sin(\dot{\Omega} \cdot dt) & \cos(\dot{\Omega} \cdot dt) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} \tag{5.18}$$

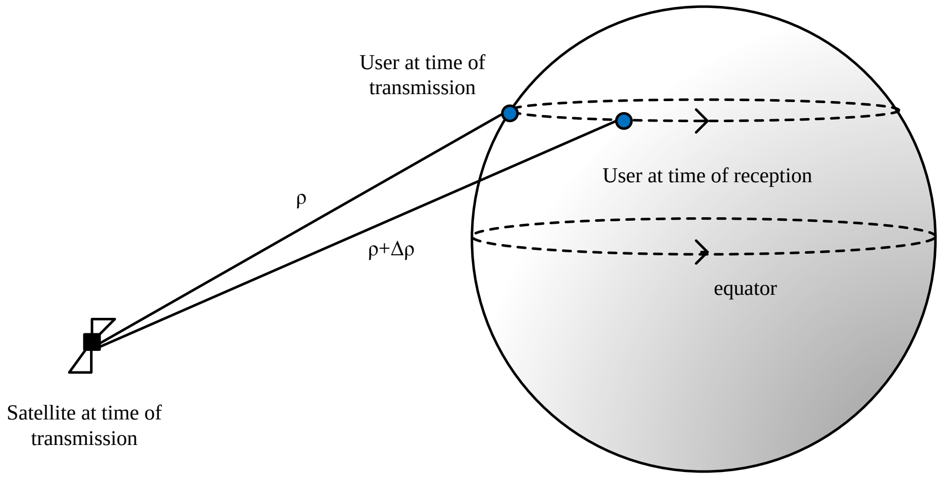

User at time of transmission
User at time of reception
ρ
ρ+Δρ
equator
Satellite at time of transmission

Figure 5.3 Sagnac effect

5.9 Satellite Clock Error

The GNSS satellites have highly accurate atomic clocks, but still each satellite has a minor bias and frequency drift, which must be accounted for. This difference between each individual GNSS satellite and the respective GNSS system time is called satellite clock error. The control segment monitors the satellite clocks and accurately measures the behavior. Correction parameters are then uploaded to the satellites, from where they are broadcast to the GNSS users through the navigation message. The broadcast parameters are listed in Table 5.5. The systems share the same parameters, but the parameters are defined with different number of bits, and scale factors.

Table 5.5 Clock correction parameters

Parameter	**Number of bits**		**Scale factor**		**Units**
	GPS	**GAL**	**GPS**	**GAL**	
T_{GD}	8*	10*	2^{-31}	2^{-32}	seconds
t_{0c}	16	14	2^{4}	60	seconds
a_{f2}	8*	6*	2^{-55}	2^{-59}	sec/sec^2
a_{f1}	16*	21*	2^{-43}	2^{-46}	sec/sec
a_{f0}	22*	31*	2^{-31}	2^{-34}	seconds
Parameters indicated with * are two's complement numbers					

$$\Delta t_r = F \cdot e \cdot \sqrt{a} \cdot \sin E_k \tag{5.19}$$

$$\Delta t = a_{f0} + a_{f1} \cdot (t - t_{oc}) + a_{f2} \cdot (t - t_{oc})^2 + \Delta t_r - T_{GD} \tag{5.20}$$

Equation (5.19) presents the correction term for relativistic effects, and it is usually associated with the clock correction. The satellites are moving with such high velocities on the orbits, that relativistic effects cannot be ignored. The relativistic effects affect the clock frequency of the satellite.

Equation (5.20) collects the total clock correction into a single equation. This clock correction is applied to the measured pseudoranges.

5.10 Atmospheric Corrections

The GNSS satellites are located on orbits approximately above 26 000 km from the center of the Earth. The assumption for typical mass-market GNSS receiver is that it is located on the surface of Earth. Due to this, the satellites are typically sending their signal at least

20 000 km from the receiver. These signals travel mostly in space, which can be considered as a vacuum. On the proximity of Earth, the signal enters Earth's atmosphere, which no longer can be considered as vacuum. In Earth's atmosphere, signals are subject to atmospheric refraction, which affects the signal propagation. For GNSS, the two most important layers of the atmosphere are called the ionosphere and troposphere. Next sections will investigate the effects of these layers, and introduce the methods of mitigating these effects in GNSS receivers.

5.10.1 Single Frequency Ionosphere Correction

The first atmospheric layer the GNSS signals encounter is called the ionosphere. Within the ionosphere, free electrons affect the signal propagation. The number of free electrons varies with time, and the common goal in mitigating the ionosphere effect is to estimate the Total Electron Content (TEC) on the path between the user and the satellite. If not accounted for, ionosphere can cause pseudorange errors of tens of meters [32]. Ionosphere effects have greater magnitude than those of the troposphere, which will be discussed in Section 5.10.5.

TEC can be expressed in TEC units (TECU), and 1 TECU = 10^{16} electrons per m^2. The delay caused by the ionosphere is related to the TEC value according to the following equation [71]:

$$I = \frac{40.3}{f^2} \cdot TEC \tag{5.21}$$

where,

I is the delay caused by ionosphere in meters,

f is the center frequency of the used GNSS band in Hertz.

GPS and Galileo both have their own models for mitigating the ionosphere errors. Both systems rely on the broadcasted parameters within the navigation data. Next, both ionosphere models are investigated.

5.10.2 GPS Klobuchar Model:

The GPS Klobuchar model is a simple model from the receiver design point of view [93]. It solely relies on the inputs received from the satellites (Satellite Transmitted Terms) and Receiver Generated Terms. Complete procedure for computing the ionosphere delay is directly presented in GPS ICD, and will not be repeated here [65]. Using the algorithm of [65] it is expected to reduce the ionosphere error with 50% for single frequency GPS measurements.

5.10.3 Galileo NeQuick Model:

The Galileo NeQuick model is another model for the ionosphere effects [94]. Implementation of NeQuick capability on receivers needs a bit more preparation, since the model assumes availability of so-called CCIR maps, and dip latitudes grid file. It must be noted that the Galileo system itself does not provide this information on the navigation data. That is one major difference to the GPS Klobuchar model, which can operate solely on the data obtained from the GPS system. On the other hand, the benefit of the NeQuick will be an improvement in the ionosphere corrections. A 70% reduction of ionosphere effects is estimated for single frequency Galileo NeQuick model user [95].

Galileo navigation data provides a few transmitted terms to enable the ionosphere corrections. The 3 broadcast parameters are presented in Table 5.6.

Table 5.6 Galileo ionosphere correction terms

Parameter	**Description**
a_{i0}	Effective ionization level 1st order parameter
a_{i1}	Effective ionization level 2nd order parameter
a_{i2}	Effective ionization level 3rd order parameter

Using the parameters of Table 5.6, the effective ionization level is computed according to the following equation:

$$Az = a_{i0} + a_{i1}\mu + a_{i2}\mu^2 \tag{5.22}$$

where,

μ is the Modified Dip latitude (MODIP).

The solved Az is used to replace the NeQuick model F10.7 value. Running NeQuick now provides the TEC value, which eventually leads to the ionosphere delay through the use of Equation (5.22).

For stand-alone GPS/Galileo receiver without availability to CCIR maps, and dip latitudes grid file, it is possible to use GPS Klobuchar model for Galileo satellites. This approach expects that the receiver will retrieve the Klobuchar correction parameters from GPS satellite navigation data, and then apply Klobuchar corrections for both systems. This method avoids the requirement of extra files in Galileo NeQuick model, with the expense of losing some accuracy in the ionosphere correction for Galileo signals. The Klobuchar model does not work with full potential for Galileo, since Galileo orbits are different from GPS orbits, as indicated in Table 5.1. On the other hand, if the receiver has NeQuick capability, it can be used instead of Klobuchar model for GPS also.

Since ionosphere error is a slowly varying error, it is possible to save some computational load without sacrificing accuracy. In [96] it has been evaluated that a single ionosphere correction computed by NeQuick is valid for up to 15 min. In other words, it is not necessary to constantly compute the ionosphere corrections, but rather update the values at certain intervals, up to the 15 min limit.

Interestingly, GLONASS satellites do not send any ionosphere correction data. When using GLONASS satellites, the ionosphere correction should rely on other GNSS systems, SBAS, or A-GNSS.

5.10.4 Dual Frequency Ionosphere Correction

The multi-frequency capability of the receiver gives access to ionosphere-free pseudorange measurements by utilizing two simultaneous pseudorange measurements made on different frequencies. Derivation of the algorithm begins with definition of ionosphere-free pseudorange:

$$\rho_{IF} = r + c(\Delta t_u - \Delta t_s) + T \tag{5.23}$$

Where,

r is geometric user to satellite range,

Δt_u is user clock bias,

Δt_s is satellite clock bias,

T is tropospheric delay.

With the help of ionosphere-free pseudorange, the pseudorange measurements made on two different frequencies can be presented as:

$$\rho_{F1} = \rho_{IF} + I_{F1} + \varepsilon_{F1} \tag{5.24}$$

$$\rho_{F2} = \rho_{IF} + I_{F2} + \varepsilon_{F2} \tag{5.25}$$

Where,

ρ_{F1}, ρ_{F2} are pseudorange measurements on frequencies F1 and F2,

I_{F1}, I_{F2} are ionosphere delays,

$\varepsilon_{F1}, \varepsilon_{F2}$ are noise terms.

The ionosphere delays are directly proportional to unknown TEC, and inversely proportional to the square of the carrier frequency. According to Equation 5.21, ionosphere delays can be modeled as:

$$I_{F1} = \frac{40.3 * TEC}{f_{F1}^{\ 2}} \tag{5.26}$$

$$I_{F2} = \frac{40.3 * TEC}{f_{F2}^{\ 2}} \tag{5.27}$$

Where

TEC is Total Electron Count,

f_{F1}, f_{F2} are the two carrier frequencies.

Substituting these values into equations (4.24) and (4.25) yields:

$$\rho_{F1} = \rho_{IF} + \frac{40.3 * TEC}{f_{F1}^{\ 2}} + \varepsilon_{F1} \tag{5.28}$$

$$\rho_{F2} = \rho_{IF} + \frac{40.3 * TEC}{f_{F2}^{\ 2}} + \varepsilon_{F2} \tag{5.29}$$

From these two equations, the unknown TEC can be eliminated:

$$f_{F1}^{\ 2}(\rho_{F1} - \rho_{IF} - \varepsilon_{F1}) = 40.3 * TEC$$

$$f_{F2}^{\ 2}(\rho_{F2} - \rho_{IF} - \varepsilon_{F2}) = 40.3 * TEC$$

$$\Rightarrow f_{F1}^{\ 2}(\rho_{F1} - \rho_{IF} - \varepsilon_{F1}) = f_{F2}^{\ 2}(\rho_{F2} - \rho_{IF} - \varepsilon_{F2})$$

$$\Rightarrow f_{F1}^{\ 2}\rho_{F1} - f_{F1}^{\ 2}\varepsilon_{F1} - f_{F2}^{\ 2}\rho_{F2} - f_{F2}^{\ 2}\varepsilon_{F2} = f_{F1}^{\ 2}\rho_{IF} - f_{F2}^{\ 2}\rho_{IF}$$

$$\Rightarrow \rho_{IF} = \frac{f_{F1}^{\ 2}}{f_{F1}^{\ 2} - f_{F2}^{\ 2}}\rho_{F1} - \frac{f_{F2}^{\ 2}}{f_{F1}^{\ 2} - f_{F2}^{\ 2}}\rho_{F2} - \frac{f_{F1}^{\ 2}}{f_{F1}^{\ 2} - f_{F2}^{\ 2}}\varepsilon_{F2} + \frac{f_{F2}^{\ 2}}{f_{F1}^{\ 2} - f_{F2}^{\ 2}}\varepsilon_{F1}$$

. (5.30)

It is very important to note that the ionosphere-free equations still contain the error terms, which cannot be ignored. This error term consists of errors due to satellite clock, ephemeris data and tropospheric delay. Direct consequence is that we have achieved the ionosphere-free pseudorange measurements at a cost – significantly increased noise in the measurements. When modeling multipath and receiver noise uncorrelated on the two used frequencies, and with same variance σ_ρ, then the RMS noise of the ionosphere free pseudorange would be defined by:

$$\sigma_{\rho_{IF}} = \frac{\sqrt{f_{F1}^{\;4} + f_{F2}^{\;4}}}{f_{F1}^{\;2} - f_{F2}^{\;2}} \sigma_\rho \qquad (5.31)$$

Conclusions:

The estimates of ionosphere-free pseudoranges (ignoring the noise terms), from Eq. (5.30) are listed in Table 5.7 together with noise gain from Eq. (5.31). When using the GPS notations for frequency bands L1, L2 and L5, we get:

Table 5.7 Dual-frequency ionosphere combinations

Frequency combination	**Ionosphere free estimate**	**RMS noise**
L1-L2	$\rho_{IF} = 2.546\rho_{L1} - 1.546_{L2}$	$\sigma_{\rho_{IF}} = 2.978\,\sigma_\rho$
L1-L5	$\rho_{IF} = 2.261\rho_{L1} - 1.261_{L5}$	$\sigma_{\rho_{IF}} = 2.588\,\sigma_\rho$
L2-L5	$\rho_{IF} = 12.255\rho_{L2} - 11.255_{L5}$	$\sigma_{\rho_{IF}} = 16.640\,\sigma_\rho$

From Table 5.7 it can be seen how dual-frequency ionosphere correction benefits from large frequency separation of the carrier waves. The best dual-frequency combination from ionosphere correction point of view would be L1-L5, which provides the correction with least increased noise. The results also indicate how poor combination L2-L5 would be (smallest frequency separation), since the noise is amplified by a factor of 16.

The receiver noise is usually a negligible factor in the noise amplification, and thus the remaining issue is with multipath errors. In order to benefit from dual-frequency correction, multipath mitigation techniques should be employed in the receiver. With multipath effects taken care of, it is recommended to utilize dual frequency corrections over the single-frequency models.

5.10.5 Troposphere correction

The troposphere is the second important layer in Earth’s atmosphere causing propagation delay to the received GNSS signals. Tropospheric delay is generally less significant than ionosphere delay. On low elevation satellites, the tropospheric delay will be considerable, but as suggested earlier in the Section 5.2.2, an elevation mask is used to exclude the low elevation satellite because of significant troposphere delays.

From implementation point of view, the GNSS systems do not send the navigation data for correcting the tropospheric delays. In the literature, there is a wide variety of algorithms presented for correcting the tropospheric delay. For standard GNSS user even the simplest models are usually enough to correct most of the delay. The simplest models work as a function of elevation angle to the satellite, whereas the more sophisticated models can utilize information about local air pressure, temperature, and other parameters used to describe the atmospheric conditions.

A simple model for correcting tropospheric delay [62]:

$$\mathrm{t}_{\mathrm{tropo}} = \frac{2.47}{\sin(E) + 0.0121} \tag{5.32}$$

where the tropospheric delay will be in meters, and elevation angle E is given in radians. Naturally, one should avoid division by zero problems when utilizing this equation with low elevation satellites.

An example of more sophisticated model based on Saastamoinen model is presented in following Table 5.8 [62].

Table 5.8 Saastamoinen troposphere model

$T_{z,d} = 0.002277 \cdot (1 + 0.0026 \cdot \cos(2\phi) + 0.00028 \cdot H) \cdot P0$	Dry zenith delay
$T_{z,w} = 0.002277 \left(\frac{1255}{T0} + 0.05 \right) \cdot e0$	Wet zenith delay
$m_d(E) = \frac{1}{\sin(E) + \frac{0.00143}{\tan(E) + 0.0445}}$	Mapping function for dry delay
$m_w(E) = \frac{1}{\sin(E) + \frac{0.00035}{\tan(E) + 0.017}}$	Mapping function for wet delay
$t_{tropo} = T_{z,d} \cdot m_d(E) + T_{z,w} \cdot m_w(E)$	Total zenith delay for troposphere

In Table 5.8 equations ϕ is user latitude in radians, H is orthometric height of the antenna in kilometers, $P0$ is total pressure in millibars, $T0$ is temperature in Kelvin, and $e0$ is partial pressure due to water vapor in millibars. A typical stand-alone GNSS receiver does not have access to atmospheric pressure values, and temperatures. As such, a receiver can substitute local averages for these values, in order to use the Saastamoinen model. Best results will be achieved if the receiver would have access to the present values over network assistance [97].

5.11 Navigation Solution

The navigation solution will solve the final PVT. In a multi-system receiver where measurements are available from both systems, e.g. GPS and Galileo, there is one new issue to be taken care of. The issue arises with the different system times of GPS and Galileo.

GPS satellites transmit their data based on GPS Time (GPST) and Galileo satellites transmit data based on Galileo System Time (GST). GPS and Galileo time systems have a certain bias between them, and this bias is one more unknown, which must be solved. For this problem, two methods are described in the following section.

- Solution 1 – Utilize broadcast Galileo/GPS Time Offset (GGTO)

Parameters to solve GGTO value will be broadcast in the Galileo data message. These parameters are listed with Equation (5.33). Once these parameters are obtained, the Equation (5.33) can be used to get the GGTO value for correcting all measurements to a common time reference. The GGTO can be used for either fixing GPS time to Galileo or vice versa. While this approach is a simple way of fixing the time domain differences, the only remaining issue will be in the accuracy of the broadcasted parameters for the GGTO algorithm. The accuracy is nevertheless expected to provide sufficient accuracy for standard GNSS applications [98].

$$GGTO = t_{gal} – t_{gps}$$

$$GGTO = A_{0G} + A_{1G}*[TOW – t_{0G} + 604800*((WN – W_{0G}) \bmod 64)] \qquad (5.33)$$

where

A_{0G}	constant term describing the offset to GGTO,
A_{1G}	rate of change of the offset GGTO,
t_{0G}	reference time for GGTO data,
WN	GST Week Number,
WN_{0G}	Week Number of the GGTO reference.

In reference

- Solution 2 – Solve the two time biases - one from each system

The approach of solving the time bias for both systems comes with a drawback. The usual way of solving the PVT is to have a minimum of 4 satellites for solving the 4 unknowns (x, y, z and bias). When having two biases to solve, a minimum of 5 satellites is needed.

Naturally, those 5 satellites must contain satellites from both systems, or otherwise the receiver can make a standard single system PVT solution with minimum of 4 satellites.

In reference [99], an analysis of the GGTO performance is presented. The broadcast GGTO performs adequately but is outperformed by the separate bias estimation method. This would be as expected, and the use of broadcast GGTO is a tradeoff between accuracy and computational load.

In the following part, a short introduction to standard Least-Squares solution is derived. The standard method needs to be modified to be able to handle dual system navigation solution.

5.11.1 Single-System Least-Squares Solution

At this point, the receiver has obtained pseudorange measurements, modelled as nonlinear measurements in Equation (2.2), and satellite locations in Table 5.4. The pseudorange equations still contain four unknowns, the user coordinates and clock bias term, which need to be solved. The first step is to linearize the pseudorange equations near an approximate user position, referred to as nominal point of linearization.

$$\mathbf{x}_{u0} = \left[x_{u0}, y_{u0}, z_{u0}\right]^T$$

The linearization works as long as the nominal point of linearization is sufficiently close to the true user location. If an estimate is available, it is recommended to use it. In case no a priori information about location is available, it is sufficient to use center of Earth as an initial guess [53]. In the actual implementation, the first navigation solution is produced with the nominal point of linearization, and subsequent solutions use previously solved user location as refined approximation. The measurement linearization is an iterative process and thus it is beneficial to use best available guess for user location, as it reduces the number of iterations needed.

Let the initial values of user location and clock bias be $\mathbf{x}_{u0}$ and b_0 correspondingly.

Then the approximate pseudorange is:

$$\boldsymbol{\rho}_i = \left\| \mathbf{x}_{SV}^i - \mathbf{x}_{u0} \right\| + b_0$$

The true user location and clock bias are:

$$\mathbf{x} = \mathbf{x}_{u\mathbf{0}} + \delta\mathbf{x}$$

$$b = b_0 + \delta b$$

Now a system of linear equations is written from where $\delta\mathbf{x}$ and δb can be solved

$$\delta\boldsymbol{\rho}_i = \boldsymbol{\rho}_{ic} - \boldsymbol{\rho}_{i0}$$

$$= \left\| \mathbf{x}_{SV}^i - x_{u0} - \delta\mathbf{x} \right\| - \left\| \mathbf{x}_{SV}^i - x_{u0} \right\| + \left(b - b_0 \right) + \tilde{\varepsilon}_{\rho}^{i} \tag{5.34}$$

The following approximation can be used

$$\left\| \mathbf{x}_{SV}^i - x_{u0} - \delta\mathbf{x} \right\| \approx \left\| \mathbf{x}_{SV}^i - x_{u0} \right\| - \frac{\left(\mathbf{x}_{SV}^i - x_{u0} \right)}{\left\| \mathbf{x}_{SV}^i - x_{u0} \right\|} \cdot \delta\mathbf{x} \tag{5.35}$$

Combining Equations (4.34) and (4.35) yields

$$\delta\boldsymbol{\rho}_i = -\frac{\left(\mathbf{x}_{SV}^i - x_{u0} \right)}{\left\| \mathbf{x}_{SV}^i - x_{u0} \right\|} \cdot \delta\mathbf{x} + \delta b + \tilde{\varepsilon}_{\rho}^{i} \tag{5.36}$$

It is useful to define direction cosines, which are present in Equation (5.36). A direction cosine is a unit vector pointing from user location to the satellite location. Direction cosines are presented as

$$\mathbf{1}_i = \frac{1}{\left\| \mathbf{x}_{SV}^i - \mathbf{x}_{u0} \right\|} \cdot \left(x_{SV}^i - x_{uo} \,,\, y_{SV}^i - y_{uo} \,,\, z_{SV}^i - z_{uo} \right)^T \tag{5.37}$$

With the help of direction cosine notation, the Equation (5.36) can be presented in matrix form.

$$\delta\mathbf{p} = \begin{bmatrix} \delta\rho_1 \\ \delta\rho_2 \\ \vdots \\ \delta\rho_N \end{bmatrix} = \underbrace{\begin{bmatrix} (-\mathbf{1}_1)^T & 1 \\ (-\mathbf{1}_2)^T & 1 \\ \vdots & \\ (-\mathbf{1}_N)^T & 1 \end{bmatrix}}_{\mathbf{G}} \cdot \underbrace{\begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ \delta b \end{bmatrix}}_{\Delta\mathbf{x}} + \tilde{\varepsilon}_\rho \qquad (5.38)$$

The matrix $\mathbf{G}$ is called the geometry matrix.

In a more compact matrix form of Equation (5.38) is

$$\delta\mathbf{p} = \mathbf{G} \cdot \Delta\mathbf{x} + \tilde{\varepsilon}_\rho \qquad (5.39)$$

The Equation (5.39) represents a system of linear measurements. The error model for the measurement errors are considered having zero mean, so

$$\mathbf{E}\left(\tilde{\varepsilon}_\rho\right) = 0$$

An optimal solution for Δx can be obtained with little manipulation of Equation (5.39):

$$\mathbf{G}^T \cdot \Big\| \quad \delta\mathbf{p} = \mathbf{G} \cdot \Delta\hat{\mathbf{x}}$$

=>

$$\mathbf{G}^T \cdot \delta\mathbf{p} = \mathbf{G}^T \cdot \mathbf{G} \cdot \Delta\hat{\mathbf{x}} \qquad (5.40)$$

=>

$$\Delta\hat{\mathbf{x}} = \left(\mathbf{G}^T \cdot \mathbf{G}\right)^{-1} \cdot \mathbf{G}^T \cdot \delta\mathbf{p} \qquad (5.41)$$

From the solution of Equation (5.41) the new estimates for user position and clock bias are

$$\hat{\mathbf{x}} = \mathbf{x}_{u0} + \delta\hat{\mathbf{x}}$$

$$\hat{b} = b_0 + \delta\hat{b}$$

At this point, the measurement Equation (5.36) can be once again linearized with these new values and this process is iterated until the solution converges to the sufficiently accurate values. Usually LS will converge quickly to correct estimate, e.g. around 5 iterations, even if starting from the center of Earth.

5.11.2 Computational Note about Least-Squares

Observing Equation (5.41) it can be seen that it provides the needed solution. The computational issue arises with the matrix inversion needed for solving the equation. In general, matrix inversions are computationally demanding operations, and are best to be avoided when performance is needed. An alternate way of solving Equation (5.41) is to use matrix decomposition methods.

- *LU decomposition*

The description of the LU decomposition algorithm begins from Equation (5.40). First, it must be noted that the term $\mathbf{G}^T \cdot \mathbf{G}$ in Equation (5.40) is always a square matrix with a size of 4x4.

$$\underbrace{\mathbf{G}^T \cdot \mathbf{G}}_{\mathbf{P}} \cdot \Delta\hat{\mathbf{x}} = \underbrace{\mathbf{G}^T \cdot \delta\boldsymbol{\rho}}_{\mathbf{X}}$$

$$\mathbf{P} \cdot \Delta\hat{\mathbf{x}} = \mathbf{X}$$

The matrix $\mathbf{P}$ can be decomposed into two matrices L and U, where L is lower triangular matrix and U is upper triangular matrix correspondingly.

$$\mathbf{L} \cdot \underbrace{\mathbf{U} \cdot \Delta\hat{\mathbf{x}}}_{\mathbf{Y}} = \mathbf{X}$$

$$\mathbf{L} \cdot \mathbf{Y} = \mathbf{X}$$

At this point only, the Y is unknown in the equation above, and it can be solved easily using forward substitution method. The situation above is best explained with an example.

$$\begin{bmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

The elements of Y are solved as follows:

$$X_1 = L_{11} \cdot Y_1 \Rightarrow Y_1 = \frac{X_1}{L_{11}}$$

$$X_2 = L_{21} \cdot Y_1 + L_{22} \cdot Y_2 \Rightarrow Y_2 = \frac{X_2 - L_{21} \cdot Y_1}{L_{22}}$$

$$X_3 = \ldots$$

After solving every element of Y, the problem can be solved by looking at how Y was defined.

$$\mathbf{U} \cdot \Delta\hat{\mathbf{x}} = \mathbf{Y}$$

Now the only unknown is the $\Delta\hat{\mathbf{x}}$ term, and it can be solved with a similar method as Y was solved.

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ 0 & U_{22} & U_{23} & U_{24} \\ 0 & 0 & U_{33} & U_{34} \\ 0 & 0 & 0 & U_{44} \end{bmatrix} \cdot \begin{bmatrix} \Delta\hat{x}_1 \\ \Delta\hat{x}_2 \\ \Delta\hat{x}_3 \\ \Delta\hat{x}_4 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix}$$

$\Delta\hat{\mathbf{x}}$ has now been solved using LU method. The fact worth noticing is that the inverse of $\mathbf{G}^T \cdot \mathbf{G}$ was not needed after all. This helps reducing the computation amount significantly. The whole detailed LU decomposition algorithm is described in [100] and the complexity of the algorithm is stated to be (2/3)n^3.

- *Cholesky Decomposition*

One step further in optimizing the process of solving the $\Delta\hat{\mathbf{x}}$ is to utilize the fact that the matrix P will be a Symmetric Positive Definite (SPD) matrix. For SPD matrices, it is possible to decompose the matrix into Cholesky factors as follows:

$$\mathbf{P} = \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{U}^T$$

Here U is again an upper triangular matrix and D is a diagonal matrix. The benefit of Cholesky decomposition is that based on the symmetry, it is enough to solve just half of the terms in comparison to LU decomposition. This gives a little improvement in the algorithm efficiency. The Cholesky factorization is described also in [100] and the complexity is $(1/3)n^3$.

As a conclusion, the decomposition methods are superior to the direct matrix inverse calculation methods, in terms of efficiency. Cholesky method is the recommended method of solving $\Delta\hat{\mathbf{x}}$.

5.11.3 Dual-System Least-Squares

Having both GPS and Galileo in the navigation solution, leads to the situation where 5th unknown for the second time bias is needed. Following the Equation (5.38) notation, a LS solution for an example case with 3 GPS satellites and 2 Galileo satellites is:

$$\mathbf{G} \cdot \Delta\mathbf{x} = \begin{bmatrix} \left(-\mathbf{1}_1\right)_{GPS}^T & 1 & 0 \\ \left(-\mathbf{1}_1\right)_{GPS}^T & 1 & 0 \\ \left(-\mathbf{1}_1\right)_{GPS}^T & 1 & 0 \\ \left(-\mathbf{1}_1\right)_{GAL}^T & 0 & 1 \\ \left(-\mathbf{1}_1\right)_{GAL}^T & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \\ c \cdot \Delta t_u(GPS) \\ c \cdot \Delta t_u(GAL) \end{bmatrix} = \delta\boldsymbol{\rho} - \tilde{e}_\rho \tag{5.42}$$

Solving the dual-system LS can be done similarly as with single-system.

5.11.4 Kalman Filter

The ability to produce LS solutions is a minimum requirement for a GNSS navigation solution. Unfortunately, the LS solutions can contain a fair amount of noise, which corrupts the quality of the results. The disadvantage of LS method is that it is unable to utilize any previous results for augmenting the current results. The motivation for selecting Kalman Filter (KF) is the fact that it is desirable to introduce a kind of a memory system where previous results would help producing new results. In other words, Kalman filter utilizes a priori information of the system in order to eliminate the randomness that noise causes to the results [101], [102], [103], [104], [105].

From the implementation point of view, a Kalman filter is not a traditional filter. It is rather an adaptive algorithm, which smoothens out the random effects from the final PVT results. In order to get the Kalman filter working, a measurement model must be described. It is also necessary to provide approximate estimates of measurement and process noise statistics. Based on the information provided, the Kalman filter will perform three distinct steps:

1. Predict the next state
2. Compute Kalman gain
3. Update the output

Some important parameters need to be defined before introducing the functionality of Kalman filter:

$\hat{\mathbf{x}}$ is the state vector. In the case of a stand-alone GNSS receiver the states might include the user location, user velocity, user acceleration, clock bias and clock drift rate. Depending on the how the modelling is done, different sets of states might be used.

$\mathbf{P}$ is the error covariance matrix, which indicates the accuracy of the state estimate.

$\mathbf{G}$ is the measurement model matrix, which connect measurements to the states. The measurement model matrix is very similar to the geometry matrix introduced in LS method section.

$\boldsymbol{\rho}$ is the measurement vector. Possible measurements in GNSS case are pseudoranges, pseudorange rates.

$\boldsymbol{\Phi}$ is the state transition matrix, which represents how the estimates propagate from one epoch to another.

$\mathbf{Q}$ is the process noise covariance matrix.

$\mathbf{R}$ is the measurement noise covariance matrix.

A basic flow chart of a discrete Kalman filter is illustrated in Figure 5.4. In these equations states and error covariance marked with superscript "+" are values after the measurement update; while "-" indicates values projected to the next round.

The equations presented in Figure 5.4 form the baseline for any Kalman filter. In GNSS case, the Kalman filter is usually implemented as an Extended Kalman Filter (EKF). The extension in this context just means that the filter is non-linear. The non-linearity of the filter comes from the fact that on each round the linearization takes place around the filter's estimated values, rather than around the nominal values.

In the next sections, two commonly used different implementations for the Kalman filter are described. The determining factor when choosing a Kalman filter will be the complexity of the modelling, and the scenario where it will be used.

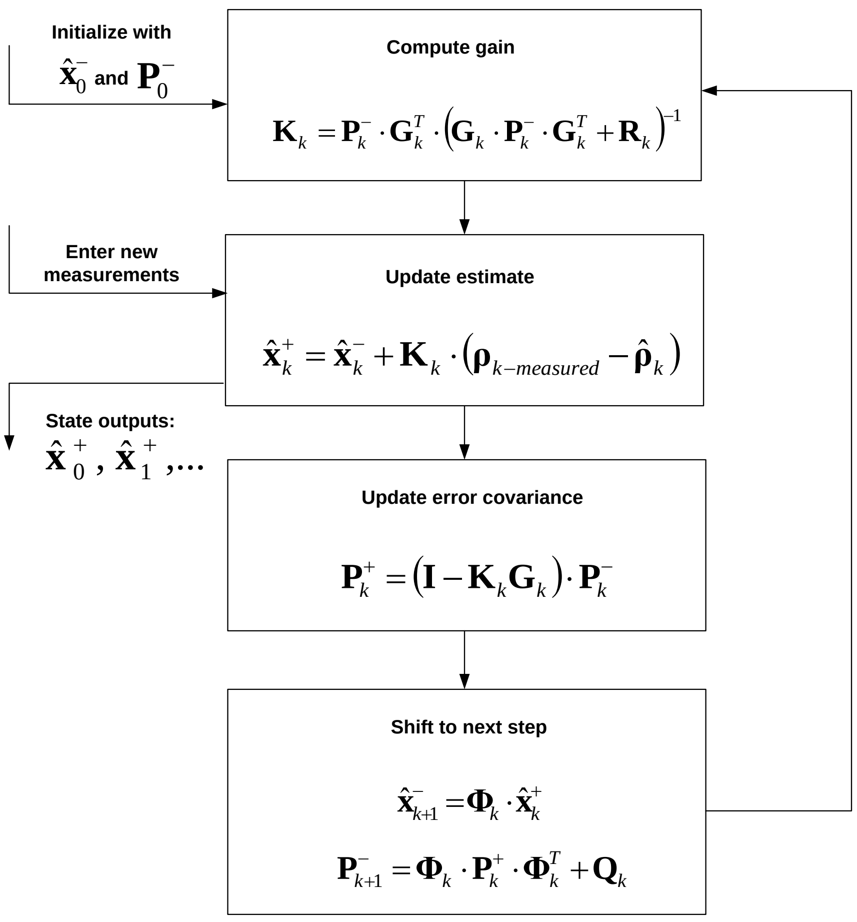

Initialize with
$\hat{\mathbf{x}}_0^-$ and $\mathbf{P}_0^-$
Compute gain
$\mathbf{K}_k = \mathbf{P}_k^- \cdot \mathbf{G}_k^T \cdot \left(\mathbf{G}_k \cdot \mathbf{P}_k^- \cdot \mathbf{G}_k^T + \mathbf{R}_k\right)^{-1}$
Enter new measurements
Update estimate
$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \cdot \left(\boldsymbol{\rho}_{k-measured} - \hat{\boldsymbol{\rho}}_k\right)$
State outputs:
$\hat{\mathbf{x}}_0^+, \hat{\mathbf{x}}_1^+, \ldots$
Update error covariance
$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k \mathbf{G}_k\right) \cdot \mathbf{P}_k^-$
Shift to next step
$\hat{\mathbf{x}}_{k+1}^- = \mathbf{\Phi}_k \cdot \hat{\mathbf{x}}_k^+$
$\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k \cdot \mathbf{P}_k^+ \cdot \mathbf{\Phi}_k^T + \mathbf{Q}_k$

Figure 5.4 Kalman filter algorithm [62]

- *Static Kalman Filter Model*

The first implemented Kalman filter was designed to be a very simple model of the GPS scenario. The simplest possible state vector would then consist only of the user location. Even the clock bias parameter can be considered to be absorbed as a bias in the user location, and thus it can be omitted from the state vector. This results in a three-state Kalman filter $\hat{\mathbf{x}} = \left[x_u, y_u, z_u\right]^T$. This particular model is called as P-model (Position) [102].

The next decision to be made is the measurement model. For a simple measurement model, the output location from LS method is selected as the measurements $\hat{\boldsymbol{\rho}}_k = [x_u, y_u, z_u]^T$. With having one-to-one correspondence between states and measurements, the measurement model matrix is easy to generate, as it will simply be an identity matrix, $\mathbf{G} = \mathbf{I}$.

As this simple model is completely neglecting the user velocity in the modelling, it is necessary that the update frequency for this model is kept quite high. The simulations have proven that the usual 1Hz navigation result rate is still suitable even with moderate user velocities. User velocity can be neglected because with high update rate and moderate user velocities, the situation is essentially static, i.e. there is not much user displacement between two quick consecutive navigation results. The neglecting of user velocity also allows using identity matrix as the state transition matrix $\mathbf{\Phi} = \mathbf{I}$.

The process noise covariance matrix can be presented as an identity matrix as well, $\mathbf{Q} = \mathbf{I}$. Then there is the measurement noise matrix left to be determined. Since the expected UERE from Equation (2.3) is ~6m, and it is suggested that somewhat larger values should be used for noise covariance [102], thus for example $\mathbf{R} = \mathbf{I} \cdot 128$, performs well.

The initial value for $\hat{\mathbf{x}}$ is simply selected as the output from the LS method and thus its initialization should not pose any problems. As for the error covariance matrix $\mathbf{P}$ it is harder to decide a good initial value. Almost anything can practically be selected as the algorithm itself will correct it, but extreme initial values can cause problems. It is mentioned in [102] that initializing $\mathbf{P}$ with too high values might even cause a divergence in the Kalman filtering due to numerical problems. Divergence is necessary to be avoided and thus small initial values for $\mathbf{P}$ are preferred. Small values cause suboptimal operation of the algorithm for the first rounds until it converges close to the true error covariance. An additional point mentioned by [102] is that if at any point $\mathbf{P}$ should be a non-positive-definite matrix, it is most likely that the algorithm will diverge. This problem can be remedied by replacing the original error covariance update equation with the following equation:

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}_k\right) \cdot \mathbf{P}_k^- \cdot \left(\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}_k\right)^T + \mathbf{K}_k \cdot \mathbf{R}_k \cdot \mathbf{K}_k^T \tag{5.43}$$

Equation (5.43) is called the Joseph form. Following the above-mentioned rules, $\mathbf{P}$ is initialized as identity matrix.

- *Kalman Filter for Mobile Environment*

In order to benefit more from the Kalman filter potential, it is needed to model the scenario with detail. The first step in improving the model is to add velocity to the observed states. The addition of velocity is ideal for hinting the direction of movement, and naturally, the next location is most likely found along the direction of movement.

Another substantial change will be made to the measurement model. Unlike the previous version which utilized the LS method results for measurements, we are now about to almost completely forfeit the use of LS method. In this more detailed version of Kalman filter pseudoranges and pseudorange rates are used as measurements. It is interesting to note that the pseudorange measurements alone are enough information for the filter to solve the user location.

The implemented state vector comprises of eight states, consisting of position, velocity and timing information, so that $\hat{\mathbf{x}} = \left[x_u, \dot{x}_u, y_u, \dot{y}_u, z_u, \dot{z}_u, t, \dot{t}\right]^T$, where t is time bias and $\dot{t}$ is clock drift rate. This particular model is called PV-model (Position, Velocity) [102].

Assuming that N satellites are included in the navigation solution, then the measurement vector would be $\hat{\boldsymbol{\rho}}_k = \left[\rho_1, \rho_2, \cdots, \rho_N, \dot{\rho}_1, \dot{\rho}_2, \cdots, \dot{\rho}_N\right]^T$.

A couple of helpful variable definitions:

$$\hat{d}_i = \sqrt{\left(x_{SV}^i - x_u\right)^2 + \left(y_{SV}^i - y_u\right)^2 + \left(z_{SV}^i - z_u\right)^2}$$

$$LOSX_i = \frac{x_{SV}^i - x_u}{\hat{d}_i}$$

$$LOSY_i = \frac{y_{SV}^i - y_u}{\hat{d}_i}$$

$$LOSZ_i = \frac{z_{SV}^i - z_u}{\hat{d}_i}$$

Now, $\rho_i = \hat{d}_i + t$,

and $\dot{\rho}_i = LOSX_i \cdot \left(\dot{x}_{SV}^i - \dot{x}_u\right) + LOSY_i \cdot \left(\dot{y}_{SV}^i - \dot{y}_u\right) + LOSZ_i \cdot \left(\dot{z}_{SV}^i - \dot{z}_u\right) + \dot{t}$

The corresponding

$$\boldsymbol{\rho}_k = [\rho_1, \rho_2, \cdots, \rho_N, \dot{\rho}_1, \dot{\rho}_2, \cdots, \dot{\rho}_N]^T$$

The pseudorange measurements are pseudoranges corrected by the navigation functions, and the pseudorange rate measurements can derived from the carrier phase measurements.

It is worth noting that all the needed parameters for the measurements are obtained via the navigation solution operations. The parameter t is used for solving the time bias between GNSS system times. Parameter $\dot{t}$ represents the estimated clock drift, which can be used in re-acquisition process.

The measurement model matrix $\mathbf{G}$ resembles quite much the geometry matrix defined for LS method in the Sections 5.11.1 – 5.11.3, and confusion between these two should be avoided. The measurement model matrix that connects states to the measurements will be:

$$\mathbf{G} = \begin{bmatrix} -LOSX_1 & 0 & -LOSY_1 & 0 & -LOSZ_1 & 0 & 1 & 0 \\ -LOSX_2 & 0 & -LOSY_2 & 0 & -LOSZ_2 & 0 & 1 & 0 \\ \vdots & & \vdots & & \vdots & & \vdots & \\ -LOSX_N & 0 & -LOSY_N & 0 & -LOSZ_N & 0 & 1 & 0 \\ 0 & -LOSX_1 & 0 & -LOSY_1 & 0 & -LOSZ_1 & 0 & 1 \\ 0 & -LOSX_2 & 0 & -LOSY_2 & 0 & -LOSZ_2 & 0 & 1 \\ \vdots & & \vdots & & \vdots & & \vdots & \\ 0 & -LOSX_N & 0 & -LOSY_N & 0 & -LOSZ_N & 0 & 1 \end{bmatrix}$$

The operation of the previous version was limited to small time intervals. Now this problem will be addressed with defining state transition matrix differently. The update rate of the system is now defined with symbol Δt . Utilizing this symbol the state transition matrix is formed as follows:

$$\Phi = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

It is mentioned that even the precise modelling of process noise covariance matrices are “guesstimates” based on rough expected vehicle dynamics at best [102], and it was noted in the testing that leaving the $\mathbf{Q}$ as identity matrix still performs sufficiently.

For the noise covariance $\mathbf{R}$, the diagonal elements consist of the measurements noise of pseudoranges and pseudorange rates. Again, it is convenient to choose a little higher noise values than what is actually expected, and the implemented matrix is:

$$\mathbf{R} = \begin{bmatrix} r_\rho & & & & & & & \\ & r_\rho & & & & & & \\ & & \ddots & & & \mathbf{0} & & \\ & & & r_\rho & & & & \\ & & & & r_{\dot\rho} & & & \\ & & \mathbf{0} & & & r_{\dot\rho} & & \\ & & & & & & \ddots & \\ & & & & & & & r_{\dot\rho} \end{bmatrix}$$

where r_ρ is pseudorange error variance and $r_{\dot\rho}$ is pseudorange rate error variance. Suitable values for example r_ρ = 128 and $r_{\dot\rho}$ = 10.

The initialization of the state vector will be taken from the output of LS method. It should be emphasized that this is the only moment where the LS method is needed in this Kalman filter approach. After getting the initial values from LS method, we do not need any further results from there, and thus it is unnecessary to calculate anything with LS method on later epochs. The reason for taking the first initial values from LS method is purely that now the Kalman filter converges to optimal performance faster than with initializing every value with zeros.

5.12 Dilution of Precision

Two factors affect GNSS accuracy: measurement errors and satellite geometry. The measurement error budget for GNSS was calculated in Chapter 2. The effect of satellite geometry on the accuracy is presented with the concept of Dilution of Precision.

A DOP value indicates how good positioning accuracy can be expected from a satellite combination, which the receiver sees in the sky. Geometrical Dilution of Precision (GDOP) is the most common of the DOP indicators, which describes the overall quality of the satellite geometry. The more the satellites are spread above the horizon, the smaller is the GDOP, and consequentially, the better is the expected positioning accuracy. Figure 5.5

illustrates the effect of geometry on the positioning accuracy. In the figure, the larger the gray area, the worse the geometry is for the situation.

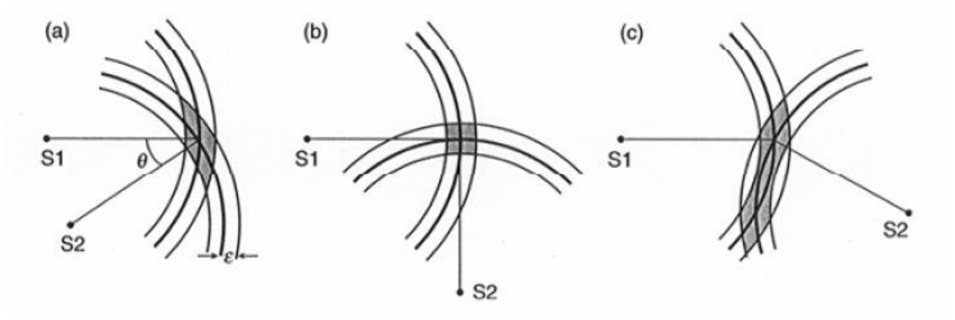


Figure 5.5 A simple 2-D satellite geometry example [32]

Using the earlier notation, GDOP is found from the geometry matrix G (introduced in Equation 5.44)

$$GDOP = \sqrt{trace\left(\mathbf{G}^T \cdot \mathbf{G}\right)^{-1}} \tag{5.44}$$

The GDOP value is useful to have, when evaluating navigation results. A combination of satellites that have minimal GDOP is preferable to use in the PVT solution. That means that if there are many satellites available for the navigation solutions, for example more than 8 satellites, it might be desirable not to include some of the satellites into the PVT solution. Having fewer satellites reduces the computational demands in the navigation algorithm.

Conclusion about GNSS accuracy:

The total accuracy of GNSS is the combination of the two components previously described:

Root Mean Square (RMS) error = $\sigma_{UERE} \cdot GDOP$

5.13 Additional PVT Calculation Topics

The Chapter 5 has listed many algorithms that the PVT calculation has to perform for obtaining the solution. In a receiver the actual rate at which the PVT computations are performed, depends on the application. For mass-market receivers it is not necessary to calculate the PVT with higher than 1 Hz rates. Some parameters are such slowly varying that they can be considered as nearly constant over short intervals of time. A prime example would be the atmospheric corrections for ionosphere and troposphere. The atmospheric conditions do not change such dramatically over short periods. Because of this, it is perfectly valid to calculate these corrections at a much slower rate than 1 Hz. The rate depends on how much error the application target can tolerate. Rates of 1 update per minute, to rate of 1 update per 15 minutes are examples of suitable time frames [96].

The atmospheric corrections are quite little computationally demanding. A bigger impact can be achieved by estimating the satellite orbits with a much simpler equation than the algorithm presented in Table 5.4. If the position, velocity, and acceleration of a satellite are once calculated with the algorithm of Table 5.4, it is possible to estimate the location of the satellite with a simple linear model [92], [106]:

$$x = x_0 + v \cdot \Delta t + a \cdot \Delta t^2 \tag{5.45}$$

This approximation saves a lot of computational effort, and is recommended to be used for saving processing power on a mass-market device. The approximations for satellite location degrade much faster than the atmospheric approximations, so it is recommended to utilize rates ranging from 1 update per 30 s, to 1 update per 1 min.

When deciding suitable update rates for a target platform, it is important to perform thorough performance analysis of the PVT algorithms in order to find out the possible tolerances for accuracy performance vs. computational savings.

6. CONCLUSIONS

Main Contributions of the Research

The main contribution of the presented research work is to provide reference collection of multi-system multi-frequency suitable algorithms for GNSS receivers.

-The implementation of acquisition module and tracking channels suitable for modern GNSS signals.

-The capability of dealing with secondary codes and pilot channels. How to deal with different navigation messages and their encoding.

-How to combine signals from multiple frequencies and systems, and to provide the PVT solution.

In the end, the GNSS receiver designer must make a choice between the complexity and accuracy of the algorithms needed. Most mass-market receivers must meet the real-time requirements, limiting some of the most complex algorithms on mass-market processor platforms. The simple algorithms are less computationally demanding, thus saving the processor time and power consumption, which is an asset for handheld devices.

The second contribution is that with the presented algorithms, it is easy to understand the requirements of GLONASS and BeiDou receivers. The ideas presented can be extended to concern these systems as well as SBAS, QZSS and IRNSS. For each implementation of a receiver, a careful benchmarking of the algorithms must be conducted. This involves evaluating the processing time of each individual satellite signal and understanding how much processing is needed in the average and worst-case scenarios.

The third contribution is to present improved testing methods for thorough testing procedures. Using a combination of simulator data, recorded data, and live sky data is essential. In [P5] the recorded data sets have been shown to have the fastest execution time.

Future Work

For the future work, one remaining task is clear: to improve the TUTGNSS receiver with GLONASS and BeiDou capabilities. In addition to the new system support, the new signals from the other frequencies could be more efficiently utilized. The implementation of a true every-systems every-frequency receiver is the ultimate goal.

A receiver with the huge capability of processing many potential signals from many sources could benefit from greater parallelism in the software processes. New architectures for a multiprocessor platform emerged as a future research topic, to evaluate how much benefits the addition of new processors would bring.

Assisted GNSS is not a new topic, but in addition to the regular assistance over network, the receivers could perform cooperative positioning in the future. Sharing data and measurements between the receivers could improve the performance at least in the limited visibility scenarios like indoors. This topic was partially started with [P6].

The complete design and development of an own multi-frequency, multi-system RF FE was seen as one possible study direction. Such design would provide much knowhow in the field of electronics design.

Finally, the process of developing the algorithms should be clearly documented and consecutively new research and educational material should continue to be published.

BIBLIOGRAPHY

[1] Official U.S. Government information about the Global Positioning System (GPS) and related topics, Available at: http://www.gps.gov/

[2] Federal Space Agency, Information-Analytical Centre, Available at: https://www.glonass-iac.ru/en/

[3] European Space Agency, Available at: http://www.esa.int/Our_Activities /Navigation/The_future_-_Galileo/What_is_Galileo

[4] BeiDou Navigation Satellite System, Available at: http://en.beidou.gov.cn /index.html

[5] European Commission, "*Galileo goes live!*", Press release, December 14th, 2016, Brussels. Available at: http://europa.eu/rapid/press-release_IP-16-4366_en.htm

[6] J. Betz, "Engineering Satellite-Based Navigation and Timing – Global Navigation Satellite Systems, Signals, and Receivers", John Wiley & Sons, 2016, ISBN: 978-1-118-61597-3.

[7] NovAtel, "*Multi-GNSS Integration: The Challenges of Diversity*", InsideGNSS, May/June 2012, pp 34-35.

[8] Wikipedia, "*List of devices with assisted GPS*", November 2016, Available at: https://en.wikipedia.org/wiki/List_of_devices_with_assisted_GPS

[9] Federal Communications Commission, "*FCC docket No. 94-102*", September 8, 2000.

[10] EU law and publications, *"The EU Directive E112"*, July 25, 2003.

[11] European Global Navigation Satellite Systems Agency, "*GNSS Market Report*", Issue 4, March 2015. Available at: https://www.gsa.europa.eu/market/market-report

[12] GPS World, "*GPS World Receiver Survey 2017*", January 2017. Available at: http://gpsworld.com/resources/gps-world-receiver-survey/

[13] European Commission, "*Galileo Open Service, Signal in Space Interface Control Document, (OS SIS ICD)",* v1.2, November 2015.

[14] E. Zemskov and J. Nurmi, "*Performance Enhancements for Embedded Software Implementation of GNSS Navigation Algorithms*," in Proc. of the Industrial Embedded Systems, IES '06, Oct. 18-20, 2006.

[15] T. Paakki, J. Raasakka, F. Della Rosa, H. Hurskainen and J. Nurmi, "*TUTGNSS – University Based Hardware/Software GNSS Receiver for Research Purposes*", in Proc. of the Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS 2010), October 2010, Helsinki (Kirkkonummi), Finland.

[16] Altera Nios II Processor URL: http://www.altera.com/products/ip/processors/nios2/ni2-index.html

[17] T. Danielsen, "*Creating a GNSS Receiver from Free Software Components*", in Proc. of ION GNSS 2007. Fort Worth, Texas.

[18] H. Hurskainen, E.-S. Lohan, J. Nurmi, S. Sand, C. Mensing and M. Detratti, "*Optimal Dual Frequency Combination for Galileo Mass Market Receiver Baseband*," in Proc. of the IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS). Tampere, Finland, October 7—9, 2009.

[19] T. Lück, J. Winkel, M. Bodenbach, E. Göhler, N. Falk, A. Consoli, F. Piazza, D. Gerna, R. Granger, P. Readman, S. Simpson and H-J. Euler, "*Artus – A second generation Galileo/GPS receiver*," in Proc of ION GNSS 2007, September 25-28, 2007, Fort Worth, Texas.

[20] P. Berglez, "*Development of a GPS, Galileo and SBAS receiver*," in Proc. of the ELMAR-2008, 10-12 September 2008, Zadar, Croatia.

[21] N. Gerein and M. Olynik, "*Prototype Galileo Receiver Development*," in Proc. of the ENC GNSS 2004, 16-19 May 2004, Rotterdam, The Netherlands.

[22] C. Mongrédien, M. E, Cannon and G. Lachapelle, "*Performance Evaluation of a GPS L5 Software Receiver Using a Hardware Simulator*," in Proc. of the ENC 07, Geneva, Switzerland, 29 May-June 1, 2007.

[23] L. Xiaoli, L. Jingnan, L. Tao and H. Xi, "*Design of Software-Based GPS/Galileo Receiver for Applications*" in China Communications, Management & Technology Forum, 2006.

[24] T. Ma, X. Cui, M. Lu, and Z. Feng, "*The Optimization of Algorithm and Implementation for Dual-Constellation Navigation Receiver*," in Proc. of the ION GNSS 2010, September 21-24, 2010, Portland, OR.

[25] S. A. Nik, and M. G. Petovello, "*Implementation of a Dual-Frequency GLONASS and GPS L1 Software Receiver*", in the Journal of Navigation, 63, the Royal Institute of Navigation, 2010.

[26] http://galileognss.eu/first-50-galileo-fixes-certified/

[27] Live Real Time Satellite Tracking and Predictions, Available at: http://www.n2yo.com

[28] Official U.S. Government information about the Global Positioning System (GPS) and related topics, "Space Segment", Available at: http://www.gps.gov /systems/gps/space/

[29] Federal Space Agency, Information-Analytical Centre, "GLONASS Status", Available at: https://glonass-iac.ru/en/GLONASS/

[30] European GNSS Service Centre, "*Constellation Information*". Available at: https://www.gsc-europa.eu/system-status/Constellation-Information

[31] International GNSS Service, "*BeiDou Constellation Status Information*". Available at: https://igscb.jpl.nasa.gov/projects/mgex/Status_BDS.htm

[32] P. Misra and P. Enge, "*Global Positioning System: Signals, Measurements, and Performance*", 2nd ed., 2006, ISBN: 0-9709544-1-7.

[33] A. J. Van Dierendonck, P. Fenton and T. Ford, "*Theory and Performance of Narrow Correlator Spacing in GPS Receiver*", in Journal of Navigation, Vol. 39, No. 3, 1992.

[34] S. Thombre, J. Raasakka, T. Paakki, F. Della Rosa, M. Valkama, and J. Nurmi, *"Automated Test-bench Infrastructure for GNSS Receivers – Case Study of the TUTGNSS Receiver"*, In Proc. of the ION GNSS+ 2013, Nashville, Tennessee, USA, September 16-20, 2013.

[35] S. Banville and F. van Diggelen, "*Innovation: Precise positioning using raw GPS measurements from Android smartphones*", In GPS World, November 7, 2016.

[36] A. Mitelman, P-L. Normark, M. Reidevall, and S. Strickland, *"Apples to Apples: A Standardized Testing Methodology for High Sensitivity GNSS Receivers"*, In Proc. of the ION GNSS 2007, Fort Worth, TX, September 25-28, 2007.

[37] F. van Diggelen, *"GNSS accuracy, Lies, Damn Lies, and Statistics"*, In GPS World, Vol. 18, No. 1, 2007.

[38] M. G. Petovello, C. O'Driscoll, G. Lachapelle, D. Borio and H. Murtaza, "*Architecture and benefits of an advanced GNSS software receiver*," Journal of Global Positioning Systems (2008), Vol. 7, No. 2: 156-168.

[39] E. D. Kaplan and C. J. Hegarty, "*Understanding GPS – Principles and Applications*," 2nd ed., Norwood, Artech House, 2006, ISBN-10:1-58053-894-0.

[40] M. S. Braasch and A. J. Van Dierendonck, "*GPS Receiver Architectures And Measurements*", Invited paper, Proceedings of the IEEE, Vol. 87, No. 1, January 1999.

[41] W. Kunysz, "*Effect of Antenna Performance on the GPS Signal Accuracy*", in Proc. of the 1998 National Technical Meeting of the Institute of Navigation, Long Beach, CA, January 1998, pp. 575- 580.

[42] F. van Diggelen, C. Abraham J. De Salas, and R. Silva, "*GNSS Inside Mobile Phones – GPS, GLONASS, QZSS, and SBAS in a Single Chip*", Inside GNSS, March/April 2011.

[43] I. Gupta, "*GNSS Antennas: The Link between Satellites and Receivers*", Inside GNSS, May/June 2016.

[44] T. Hekmat, and G. Heirichs, "*Dual-Frequency Receiver Technology for Mass Market Applications*" In Proc. of ION GPS 2001, 11-14 September 2001, Salt Lake City, UT.

[45] N. Storey, "*Electronics – A Systems Approach*", 2nd ed. Pearson Education Limited, 1998, ISBN 0-201-17796-X.

[46] J. M. Samper, R. B. Pérez and J. M. Lagunilla, "*GPS & Galileo: Dual RF Front-end Receiver and Design, Fabrication, and Test*", McGraw Hill, 2009, ISBN 978-0-07-159869-9.

[47] H. T. Friis, "*Noise Figures of Radio Receivers*", in Proc. of I.R.E, July, 1944.

[48] R.M. Weiler, P. Blunt, M. Unwin and S. Hodgart, "*A Direct-Conversion Receiver for Wide-Band GNSS Signals*", in Proc. of the ENC-GNSS 2008, Toulouse, France, April 2008.

[49] F. Piazza and Q, Huang, "*A 1.57 GHz RF Front-end for Triple Conversion GPS Receiver*", IEEE Journal of Solid-State Circuits, Vol. 33, February 1998.

[50] M. Detratti, E. López, E. Pérez, and R. Palacio, "*Dual-Band RF Front-End Solution for Hybrid Galileo/GPS Mass Market Receivers*", In Proc. of IEEE CCNC, January 10-12, 2008, Las Vegas, Nevada.

[51] NTLab, "*New 4-channel multi-frequency RF front-end supporting 5 GNSS systems*", Company News, April 17, 2015. Datasheet Available: ntlab.com/sitefiles/1/6/21228/NT 1065 SE DS v0.26.pdf

[52] S. Söderholm, T. Jokitalo, K. Kaisti, H. Kuusniemi and H. Naukkarinen, "*Smart Positioning with Fastrax's Software GPS Receiver Solution*," in Proc. of ION GNSS 2008, Savannah, Georgia, Sep. 16-19, 2008.

[53] K. Borre, D. Akos, N. Bertelsen, P. Rinder and S. Jensen, "*A Software Defined GPS and Galileo Receiver: A Single-Frequency Approach*," Boston, Birkhäuser, 2007, ISBN 978-0-8176-4390-4.

[54] D. M. Lin, and J. B-Y. Tsui, "*Comparison of Acquisition Methods for Software GPS Receiver*", In Proc. of ION GPS 2000, 19-22 September 2000, Salt Lake City.

[55] M. Paonni, M. Bavaro, M. Anghileri, and B. Eissfeller, *"On the Design of a GNSS Acquisition Aiding Signal"*, in Proc. of ION GNSS+ 2013, Nashville, Tennessee, September 16-20, 2013.

[56] J. B-Y, Tsui, "*Fundamentals of Global Positioning System Receivers – A Software Approach*", John Wiley & Sons, Inc., 2005, ISBN 0-471-70647-7.

[57] R. M. Cerda, "*Understanding TCXOs*", Microwave Product Digest, April, 2005

[58] Z. Xiangxiang, Y. Jing, and Z. Zhang, "*A B1I Signal Fast Acquisition Scheme of Beidou Soft Receiver*", in Proc. of 2014 IEEE Chinese Guidance, Navigation and Control Conference, August 8-10, 2014, Yantai, China.

[59] M. Z. H. Bhuiyan, S. Söderholm, S. Thombre, L. Ruotsalainen, and H. Kuusniemi, "*Overcoming the Challenges of BeiDou Receiver Implementation*", Sensors 2014, 14, 22082-22098; doi:10.3390/s141122082.

[60] A.T. Balaei, and D.M. Akos, "*Cross Correlation Impacts and Observations in GNSS Receivers*", Journal of the Institute of Navigation, Vol.58, No.4, winter 2011.

[61] M. Foucras, J. Leclére, C. Botteron, O. Julien, C. Macabiau, P.-A. Farine, and B. Ekambi, "*Study on the cross-correlation of GNSS signals and typical approximations*", GPS Solutions, August 2016, DOI: 10.1007/s10291-016-0556-7.

[62] B. Parkinson and J. Spilker (Editors), "*Global Positioning System: Theory and Applications*", American Institute of Aeronautics and Astronautics (AIAA), Volume 1, 1996, ISBN: 1-56347-106-X.

[63] J. Betz, et.al., "*Description of the L1C Signal*", In Proc. of ION GNSS 2006, 26-29 September 2006, Fort Worth, Texas.

[64] S. U. Qaisar, "*Receiver Strategies for GPS L2C Signal Processing*", Ph.D. Thesis, School of Surveying & Spatial Information Systems, The University of New South Wales, Australia, March 2010.

[65] Navstar GPS, "Global Positioning System Interface Control Document, IS-GPS-200H", 9th December, 2015.

[66] F. D. Natali, "*AFC Tracking Algorithms*", IEEE Transactions on Communications, Vol. Com-32, NO 8, August 1984.

[67] P.W. Ward, "*Performance Comparisons between FLL, PLL and a Novel FLL-assisted-PLL Carrier Tracking Loop Under RF Interference Conditions*", in Proc. of the International Technical Meeting of the Institute of Navigation, September, 1998.

[68] J. T. Curran, G. Lachapelle, and C. C. Murphy, "*Improving the Design of Frequency Lock Loops for GNSS Receivers*", IEEE Transactions on Aerospace and Electronic Systems, Vol. 48, NO.1, January 2012.

[69] S.A. Stephens and J.B. Thomas, "*Controlled-Root Formulation for Digital Phase-Locked Loops*", IEEE Transactions on Aerospace and Electronic Systems, Vol. 31, NO. 1, January, 1995.

[70] N. C. Beaulieu, A. S. Toms, and D. R. Pauluzzi, "*Comparison of Four SNR Estimators for QPSK Modulations*", IEEE Communications Letters, Vol. 4, No. 2, February 2000.

[71] B. Hoffman-Wellenhof, H. Lichtenegger, and E. Wasle, “*GNSS – Global Navigation Satellite Systems*”, Springer, 2008, ISBN 978-3-211-73012-6.

[72] C. Ouzeau, C. Macabiau, A.C. Escher, and B. Roturier, “*Compliance of single frequency ionospheric delay estimation and cycle slip detection with civil aviation requirements*”, In Proc. of ION NTM 2007, San Diego, CA.

[73] T.F. Fath-Allah, “*A new approach for cycle slips repairing using GPS single frequency data*”, World Applied Sciences Journal, 8, 2010.

[74] S. Banville, “*Improved Convergence for GNSS Precise Point Positioning*”, Ph.D. dissertation, Department of Geodesy and Geomatics Engineering, Technical Report No. 294, University of New Brunswick, Fredericton, New Brunswick, Canada, 2014.

[75] F. van Diggelen, “*A-GPS – Assisted GPS, GNSS and SBAS*”, Artech House 2009, ISBN-13: 978-1-59693-374-3.

[76] A. J. Viterbi, “*Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm*”, IEEE Trans. on Information Theory, April 1967, pp. 260-269.

[77] E. A. Lee and D. G. Messerschmitt, “*Digital Communication*”, 2nd ed., Kluwer Academic Publishers, 1994, ISBN 0-7923-9391-0.

[78] J. G. Proakis, “*Digital Communications”*, 3rd ed., McCraw-Hill, 1995, ISBN 0-07-051726-6.

[79] T. V. Ramabadran and S. S. Gaitonde, “*A Tutorial on CRC Computations*”, IEEE Micro, 1988.

[80] M. Anghileri, M. Paonni, D. Fontanella, and B. Eissfeller, ”*Assessing GNSS Data Message Performance*”, InsideGNSS, March/April, 2013.

[81] R. B. Langley, “*GPS Receiver System Noise*”, in GPS World, Innovation Column, June 1997, pp. 40-45.

[82] G. Swan, Wikipedia, “*Satellite Navigation*”, June 2015, Available at: https://en.wikipedia.org/wiki/Satellite_navigation.

[83] J. Syrjärinne, "*Time Recovery through Fusion of Inaccurate Network Timing Assistance with GPS Measurements*," in Proc. 3rd Int. Conference on Information Fusion, Paris, France, July 10-13, 2000.

[84] P. Teunissen, "*Least Squares Estimation of the Integer Ambiguity Estimation*", in IAG General Meet., Inv. Lecture, Section IV: Theory and Methodology., Beijing, China, 1993.

[85] A. Leick, "*GPS Satellite Surveying*", 3rd ed., John Wiley & Sons, 2004, ISBN 0-471-05930-7.

[86] C. C. Counselman and S. A. Gourevitch, "*Miniature Interferometer Terminals for Earth Surveying: Ambiguity and Multipath with Global Positioning System*", IEEE Transactions on Geoscience and Remote Sensing, 1981.

[87] R. R. Hatch, "*The synergism of GPS code and carrier measurements*", in Proc. of the Third International Geodetic Symposium on Satellite Doppler Positioning, New Mexico, 1982.

[88] D. W. Simili and B. Pervan, "*Code-Carrier Divergence Monitoring for the GPS Local Area Augmentation System*", in Proc. of the IEEE/ION PLANS 2006, April 25-27, 2006, San Diego, California, pp- 483-493.

[89] Russian Space Systems, "GLONASS Interface Control Document*",* Edition 5.1, 2008.

[90] G. Gendt, Z. Altamimi, R. Dach, W. Sohne and T. Springer, "*GGSP: Realisation and Maintenance of the Galileo Terrestial Reference Frame*", Advances in Space Research, Vol. 47, January 2011.

[91] M. J. Merrigan, "*A Refinement of the World Geodetic System 1984 Reference Frame*", in Proc. of the ION GPS 2002, Portland, OR, September 2002.

[92] P. Korvenoja and R. Piché, "*Efficient Satellite Orbit Approximation*," in Proc. of ION GPS 2000, Salt Lake City, USA, Sept. 19-22, 2000.

[93] J. Klobuchar, "*Ionospheric Time-Delay Algorithm for Single-Frequency GPS Users*", IEEE Transactions on Aerospace and Electronics Systems, Vol. AES-23, May 1987.

[94] B. Bidaine, R. Prieto-Cerdeira and R. Orus, "*NeQuick: In-depth Analysis and New Developments*," in Proc. of the 3rd ESA Workshop on Satellite Navigation User Equipment Technologies, NAVITEC 2006, Noordwijk, The Netherlands, 2006.

[95] European Commission, "*Ionosphere correction algorithm for Galileo single frequency users*", Issue 1.2, September 2016.

[96] A. Angrisano, S. Gaglione, C. Gioia and S. Troisi, "*Validity Period of NeQuick (Galileo Version) Corrections: Trade-off between Accuracy and Computational Load*", in Proc. of the International Conference on Localization and GNSS (ICL-GNSS 2014), June 2014, Helsinki, Finland.

[97] J. LaMance, J. DeSalas and J. Järvinen, "*Assisted GPS – A Low-Infrastructure Approach*", in GPS World, March, 2002.

[98] J. H. Hahn and E. D. Powers, "*Implementation of the GPS to Galileo Time Offset (GGTO)*", in Proc. of Frequency Control Symposium and Exposition, August 29-31, 2005.

[99] C. Gioia, J. Fortuny-Guasch, and F. Pisoni, "*Estimation of the GPS to Galileo Time Offset and its validation on a mass market receiver*", in Proc. of the European Workshop on GNSS Signals and Signal Processing (NAVITEC 2014), Noordwijk, The Netherlands, December 2014.

[100] G. H. Golub and C. F. Van Loan, "*Matrix Computations*", 3rd ed., the Johns Hopkins University Press, 1996, ISBN 0-8018-5413-X.

[101] M. Grewal, A. Andrews and C. Bartone, "*Global Navigation Satellite Systems, Inertial Navigation, and Integration*", 3rd ed., John Wiley & Sons, 2013, ISBN: 978-1-118-44700-0.

[102] R.G. Brown and P.Y.C Hwang, "*Introduction to Random Signals and Applied Kalman Filtering*," 3rd ed., John Wiley & Sons, 1997, ISBN 0-471-12839-2.

[103] M. Grewal and A. Andrews, "*Kalman Filtering: Theory and Practice – Using MATLAB*", 2nd ed., John Wiley & Sons, 2001, ISBN: 0-471-39254-5.

[104] J. A. Farrell and M. Barth, "*The Global Positioning System & Inertial Navigation*", McGraw Hill, 1999, ISBN 0-07-022045-X.

[105] Y. Bar-Shalom, X. R. Li and T. Kirubarajan, "*Estimation with Applications to Tracking and Navigation*", John Wiley & Sons, Inc., 2001, ISBN 0-471-41655-X.

[106] H. D. Young and R. A. Freedman, "*University Physics*", 9th ed., Addison-Wesley Publishing Company, Inc., 1996, ISBN 0-201-31132-1.

PUBLICATIONS

PUBLICATION 1

T. Paakki, J. Raasakka, F. Della Rosa, H. Hurskainen, and J. Nurmi, "TUTGNSS – University Based Hardware/Software GNSS Receiver for Research Purposes", in Proceedings of the *Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS 2010),* October 2010, in Helsinki (Kirkkonummi), Finland.

TUTGNSS – University Based Hardware/Software GNSS Receiver for Research Purposes

Tommi Paakki, Jussi Raasakka, Francescantonio Della Rosa, Heikki Hurskainen, and Jari Nurmi
Department of Computer Systems
Tampere University of Technology
Tampere, Finland
Email: {fistname.lastname@tut.fi}

Abstract—**Recently a lot of research has been going on in the field of Global Navigation Satellite System receivers. Only few projects have been providing open source hardware receivers. In this paper we present a university based GNSS receiver intended for open source release. The highly modular design of the receiver is described in detail. We also provide description of the software parts and advanced user interface of the receiver. In the end of the paper a test scenario is provided proving the functionality of the receiver design. The presented GNSS receiver architecture is expected to be valuable prototype for research purposes.**

Keywords- GNSS, GPS, GALILEO, Receivers, Education

I. Introduction

At the moment the Global Navigation Satellite System (GNSS) field is rapidly advancing into new modern technologies. The American Global Navigation System (GPS) and the Russian GLONASS systems are both ongoing a modernization phase where the existing constellations are being improved. The GPS will soon have new Satellite Vehicles (SV) sending new signals with better modulation techniques. In addition the European Galileo positioning system is ongoing a validation phase and first satellites are expected to be launched in the near future. Also China has plans to extend their regional Beidou satellite system into global navigation system.

With all the emerging new GNSSs a lot of research is ongoing on the receiver technology. The market potential in the GNSS field is expected to grow. It is important to develop receivers capable of operating with the new GNSS signals. From commercial point of view this would mean that companies can provide such receivers to the customers as soon as the new systems become available.

In this paper we are presenting the current status of the TUTGNSS Reference Receiver developed at Tampere University of Technology (TUT) [1]. The TUTGNSS is a fully functional GNSS receiver, providing significant platform for developing new algorithms for the multi-system, multi-frequency GNSS field. The key emphasis in the design has been openness, high modularity and the educational aspects in terms of getting debugging information from the receiver.

This paper is divided into the following sections. First, Section II gives a brief overview of the TUTGNSS Reference Receiver architecture and specifications. Then, Section III provides detailed block by block description of the implemented architecture. Section IV provides results obtained with the TUTGNSS Reference Receiver. And finally, Section V concludes the presented work and future plans.

II. TUTGNSS Reference Receiver

The objective in the development of TUTGNSS Reference Receiver has been to make an educational hardware/software platform for GNSS receiver algorithm testing. The importance of the TUTGNSS is that there are no “black boxes” in the design, allowing users to have full control over its further development. Currently, we are looking for possibility to release basic version of TUTGNSS as open software [2].

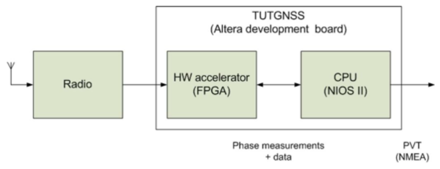


Figure 1. Architecture of TUTGNSS

The TUTGNSS Reference Receiver uses hardware-accelerated software receiver architecture, where hardware logic performs correlations for acquisition and tracking, and software controls all events that take place in the hardware. Figure 1 shows the high level architecture of TUTGNSS Reference Receiver. Current Field Programmable Gate Array (FPGA) used for development of TUTGNSS is an Altera Stratix II DSP development board (EP2S180) [3], but in principle the design should be platform independent. The hardware parts are implemented in VHDL hardware description language.

The software processes are executed on NIOS II softcore processor implemented within the FPGA. Software processes are handled and driven by control process which implements interrupt service routines according to priorities. The main software processes of the receiver are identified by:

- Receiver Control Process
- Acquisition Control Process
- Tracking Control Process
- Navigation Solution Process

The source codes for software have been implemented in C-language. A detailed picture of the hardware/software architecture of the platform is shown in Fig. 2.

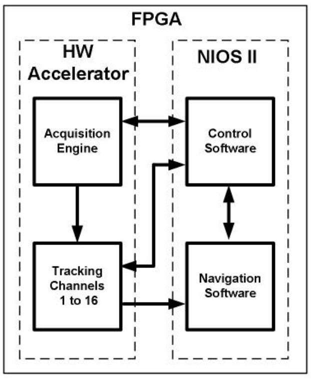


Figure 2. Hardware/Software communications in TUTGNSS

III. Building A GNSS Receiver

The architecture of a standard GNSS receiver has been presented in the literature [4], [5]. Our TUTGNSS design follows the standard architecture illustrated in Fig. 3. Next we present detailed description of each block designed for TUTGNSS receiver.

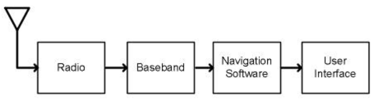


Figure 3. Block diagram of a general GNSS receiver

A. Radio Front End

The radio front end is first component in the GNSS receiver chain, and thus it has critical effect on the performance of the rest of the receiver chain. There are several requirements that the radio front end must fulfill:

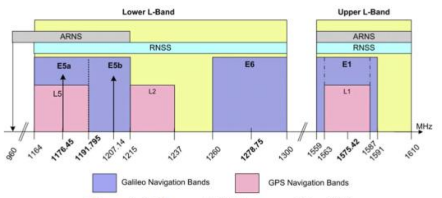


Figure 4. Galileo and GPS Frequency Plan [12]

- Compatibility with the interesting GNSS frequency bands: L1/E1 band for single frequency operation, L1/E1 + L5/E5a bands for dual frequency mode. GNSS frequency bands are illustrated in Fig. 4.
- Easy interfacing to the FPGA board.
- Low cost

We have not had the resources to design our own radio front end, so we rely on third party hardware in our design. Several suitable commercial front ends exist but they still differ with some design critical parameters, such as sampling frequency, Intermediate Frequency (IF) and sample format after analog to digital conversion.

In past, TUTGNSS has been successfully integrated with Atmel ATR0603 GPS L1 RF FE [6], Sige SE4120L [7] and NEMERIX GPS L1 Evaluation Board [8].

B. Baseband

The TUTGNSS is an entity composed of hardware blocks and software. Current design supports only standalone GNSS navigation. 16 flexible tracking channels with 8 correlators per channel have been implemented. The baseband has modular design enabling rapid prototyping of advanced algorithms. Figure 5 illustrates the modularity of the baseband design. In next part we will describe the modules in detail.

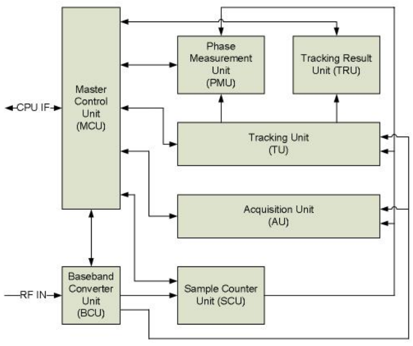

Master Control Unit (MCU)
Phase Measurement Unit (PMU)
Tracking Result Unit (TRU)
CPU IF
Tracking Unit (TU)
Acquisition Unit (AU)
Baseband Converter Unit (BCU)
RF IN
Sample Counter Unit (SCU)

Figure 5. Hardware modules of TUTGNSS

1) Baseband Converter Unit (BCU)

Baseband converter unit changes any type of RF-Front end to common three bit I&Q signed presentation. Also its job is to remove the intermediate frequency (IF) so that acquisition and tracking units need only to account the Doppler frequency of the incoming satellite signal. BCU architecture is given in Fig. 6. The BCU also contains optional decimation for input, separately for both acquisition and tracking.

BCU is always updated to meet the parameters of used radio. BCU also has debugging mode where RF-Front end data can be emulated through software. This eases debugging of the system as the input data parameters (Satellites/Doppler/Delays) are known and thus it's easy to verify if the hardware works correctly. Dataset for RF-Front end emulation can be easily made using Matlab for example.

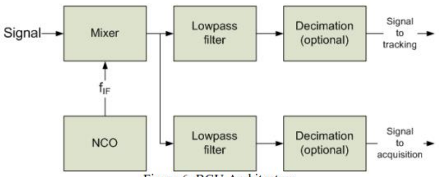


Figure 6. BCU Architecture

2) Master Control Unit (MCU)

Master control unit handles all information passing between CPU and the GNSS HW accelerator. From the SW programming point of view it has only one control register which is reserved for the moment. Control register may be used in the later versions to control transfers between CPU and the GNSS HW accelerator. Idea for having a MCU is that this way we may change transfer protocol without having to change any other unit in the system.

3) Sample Counter Unit (SCU)

Sample counter unit measures time by counting the samples coming from the RF-Front end. This is very important because this is our local time reference that we need to use to calculate user PVT (Position, Velocity, and Time) information. SCU is used also to synchronize all units together, mainly transformation between acquisition and tracking stages. SCU takes its input from the BCU and updates its counter whenever new sample arrives. When software knows the incoming sample rate this information can be used as very high precision clock, which accuracy is depended on the RF-Front end oscillator. If software uses debugging mode of the BCU then time SCU increments only when user drives new sample to the BCU input.

4) Acquisition Unit (AU)

Acquisition unit searches satellites from the incoming data stream. Acquisition unit is used by setting desired satellites to be searched with software, desirable Doppler values and then starting the acquisition by writing any value to acquisition start register. Acquisition result (Amplitude/Doppler/Delay) can be read from the appropriate registers after acquisition has been done. Decision whether acquisition has been successful is made by software.

Currently, AU is implemented as parallel matched filter structure, but by using same AU block interface it is possible to replace it with other type of acquisition, e.g. FFT.

5) Tracking Unit (TU)

Tracking Unit handles the tracking of each satellite found in the sky. Tracking Unit performs basic signal tracking tasks: data extraction and phase measurement for pseudorange composition. Software register interface is designed to support maximum of 32 tracking channels, but hardware may be limited to less (space available in the FPGA). Current design uses 16 channels.

Tracking channels themselves have very little control registers since the programming interface has been made very simple to ease software control of the system. It has to be noted that NCO values for each channel are only updated at each epoch edge.

6) Tracking Result Unit (TRU)

Tracking result unit collects all results from the TU. It is basically a FIFO which collects all the integrated results from the TU so that software can later read the results when it has time to do it. These results can be then used to update TU NCO registers and demodulate data information concerning each satellite signal. It is left for the software how to use this information, but results are generated at each epoch edge so regarding to different GNSS systems/signals integration time may be different. Interface supports up to 8 correlators.

7) Phase Measurement Unit (PMU)

Phase measurement unit measures the phases of the tracking channel NCO's at software specified intervals. This phase measuring is related to local time information received from the SCU. Phases are measured from every tracking channel simultaneously so software can derive pseudorange information for each satellite with the aid of demodulated data from the satellites. PMU stores information about code and carrier phases so advanced methods on pseudorange calculation are also possible.

C. Navigation Software

The navigation software is responsible for producing the final PVT solution. TUTGNSS navigation software has been divided into two independent tasks, which are illustrated in Fig. 7.

1) Bit Processing Task

The Main purpose for bit processing task is to collect the extracted navigation data bits coming from the baseband. For each tracked satellite the navigation data is continuously collected. Bit process task synchronizes to the navigation data frame structure (either GPS or Galileo) by looking for the respective synchronization patterns in the stream. After successful synchronization the bit processing task performs further data integrity checks and data decoding according to Fig. 8. After the necessary bit manipulation, if the data was declared as valid, the important data is stored into memory. The important data includes ephemeris data, clock correction terms, almanac data and ionosphere model terms.

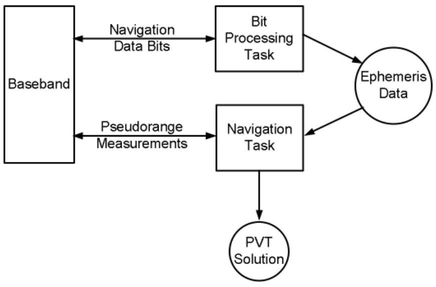


Figure 7. TUTGNSS Navigation software tasks

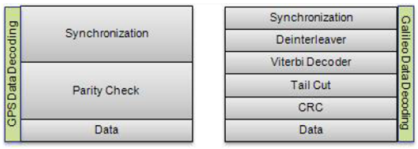


Figure 8. Data decoding process for GPS and Galileo

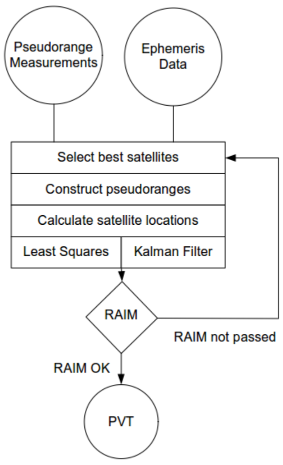


Figure 9. TUTGNSS navigation algorithm

2) Navigation Task

The navigation task is responsible of producing the PVT solution. The default rate for producing a PVT solution is 1 Hz. The rate can be changed if needed. The navigation task software is written in standard C language. In order to improve the portability of the code we have decided to implement our own fixed point arithmetic library. With the fixed point library we could run the code even on processors which do not have native floating point support. Furthermore we have implemented the needed mathematical functions with our own fixed point arithmetic, so that standard math library is not needed to be included into the project. Most of the needed mathematical functions have been implemented using the CORDIC method [9]. Currently we have two versions of the fixed point software: a 64-bit and a 96-bit version for extra precision. The functional parts of the navigation task are illustrated in Fig. 9. Next we are going to look at each part in more detail.

Select best satellites: A satellite will be considered to be included in the navigation solution after it has started receiving pseudorange measurement from the baseband, and it has valid ephemeris data in the memory. The next step is to check that the candidate satellite has reasonable C/N$_0$ level, is above predefined elevation angle, and that is has not been excluded by the RAIM/FDE (Receiver Autonomous Integrity Check / Fault Detection and Exclusion). In the third step we check that we have 4 or more good satellites before we give the permission to proceed with the navigation solution.

Construct pseudoranges: The construction of pseudoranges is done with the standard method presented in the literature [10]. The pseudorange is defined as:

$$\rho(t) = c \cdot \left[t_u(t) - t^{(s)}(t-\tau)\right] \tag{1}$$

where:

$$\begin{aligned} t^{(s)}(t-\tau) = \; & \text{Z-count} \\ & + \text{number of navigation bits} \\ & + \text{number of C/A codes} \\ & + \text{number of C/A code chips} \\ & + \text{fraction of C/A code chip,} \end{aligned}$$

$t_u(t)$ = receiver time,

c = speed of light.

The *Z-count* and *number of navigation bits* is received from the Bit process task, while baseband provides the information about *number of C/A codes*, *number of C/A code chips* and *fraction of C/A code chip.*

Calculate Satellite Locations: By default the satellite location calculations are recalculated at every new PVT solution. The standard method described in the GPS and Galileo Interface Control Documents (ICD) [11],[12], are computationally quite demanding algorithms, so we have decided to optimize the computational burden by approximating the satellite locations over short periods with linear models [13].

Least Squares / Kalman Filter: The PVT solution is produced in this step. The TUTGNSS navigation task currently runs 2 different methods in parallel, namely the Least Squares (LS), and Extended Kalman Filter (EKF). The purpose of running 2 different algorithms is purely educational point of view. The LS method is widely accepted as the basic method of producing the PVT solution. In TUTGNSS we use LS to form the basic solution, and then we try to improve our EKF models. The EKF model performance is always monitored against the LS solution, and we expect the well modeled EKF to yield higher accuracy solution than LS. Furthermore, if for some reason the LS and EKF solutions differ by a large amount, we can easily log the debugging information and analyze what caused the differences in the algorithms. Currently TUTGNSS has 3 models for EKF: P-model, for stationary user, PV-model for dynamic scenario, and PVA-model for scenarios with highly varying acceleration. All the models have been implemented by processing scalar measurement one at a time, i.e. without need of matrix algebra [14].

RAIM: With the RAIM we ensure that the provided PVT solution can be trusted. We check the measurement residuals if the navigation solutions and if set thresholds are exceeded, certain satellites will be excluded from the list of accepted satellites. The RAIM also checks basic sanity checks that e.g. ranges to satellites are around 24 000 km. If the RAIM is not passed, we try to exclude a satellite which is most probably causing the problems. If we still have the required 4 satellites the PVT solution is reproduced with the new set of satellites.

D. User Interface

The user interface is used for visualizing the results that the user is interested in. TUTGNSS provides 2 different user interfaces.

TUTGNSS Debugging domain: The debugging domain has been developed for the purposes of debugging the hardware parts of TUGNSS receiver. Traditionally it is very hard to get information out of the hardware parts, so we decided to implement a conventional way of doing that. The debugging domain is based on implementing a second dedicated NIOS II processor into the design. The second processor is interfaced to the design via shared memory from where it can read the debugging information without affecting the rest of the system. The read debugging info can then be stored into compact flash memory for post processing, or it can be sent directly to a web browser via an Ethernet connection. The principle of the debugging domain is presented in Fig. 10. The design is based on Altera Web server [15].

Graphical user interface: For the graphical user interface we can use freely available software which takes NMEA format data as input. One interesting possibility is directly displaying the PVT solution on Google Maps [16]. Of course for post processing reasons logging the data from TUTGNSS, and then analyzing it on Matlab has lots of educational aspects.

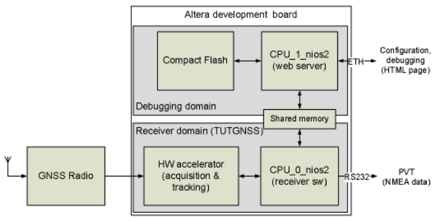

Altera development board
Compact Flash
CPU_1_nios2 (web server)
ETH
Configuration, debugging (HTML page)
Debugging domain
Shared memory
Receiver domain (TUTGNSS)
GNSS Radio
HW accelerator (acquisition & tracking)
CPU_0_nios2 (receiver sw)
RS232
PVT (NMEA data)

Figure 10. TUTGNSS architecture with debugging domain

IV. Results

In this chapter we provide results that we have achieved with the TUGNSS receiver. First Fig. 11 presents the current resource consumption of the design on the Altera Stratix II FPGA. It can be seen that the AU is the biggest single unit consuming 70% of the design resources. The current 16 TUs consume roughly 1/3 of the amount of AU. The whole design consumes 70% of the resources available on Stratix II.

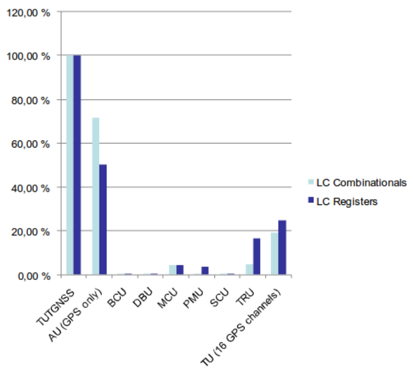


Figure 11. TUTGNSS hardware synthesis results

To evaluate the whole receiver performance we have conducted testing campaign in the Tampere area, Finland. We used a patch antenna attached to the roof of a car. The recorded PVT solution was later displayed on Google Maps. Figure 12 presents a zoomed part of the route. No obvious plunders were encountered on the test route.

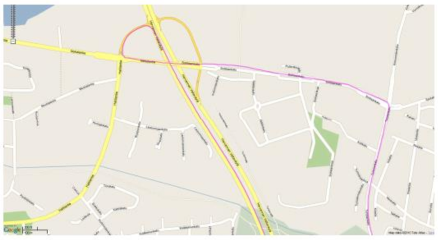


Figure 12. Ground track of the dynamic test viewed on Google Maps.

V. CONCLUSIONS AND FUTURE WORK

In this paper we presented the architecture of the TUTGNSS, a university based GNSS receiver intended to be released as open source. We have described the modular design approach in detail for both hardware and software parts.

The TUTGNSS Reference Receiver is now fully functional real-time single-frequency GNSS receiver. Hardware synthesis results and a sample test scenario are provided in the paper. The presented architecture is expected to be a valuable platform for future research on GNSS receiver technology.

Future work will consist of finalization of the open source tool prior releasing it to public. Also integrating the design with dual-frequency RF front end to enable the dual-frequency testing on the platform is foreseen. Further testing with Galileo satellites will be expected in the future. Also careful timing analysis for the selected RAIM complexity is needed in order to avoid any real-time requirement violations.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°227890 (GRAMMAR project).

REFERENCES

[1] H. Hurskainen, T. Paakki, Z. Liu, J. Raasakka, and J. Nurmi, "GNSS receiver reference design," in Proc. of the ASMS 2008 conference, Aug. 26-28, 2008, Bologna, Italy.

[2] Homepage of DCS GNSS Receiver Group, URL: http://www.tkt.cs.tut.fi/research/gnss/

[3] Altera Stratix II URL: http://www.altera.com/literature/manual /mnl_stx2_pro_dsp_dev_kit_ep2s180.pdf

[4] E. D. Kaplan, and C. J. Hegarty, "Understanding GPS – Principles and applications," 2nd ed., Norwood, Artech House, 2006, ISBN-10:1-58053-894-0.

[5] M. S. Braasch and A. J. Van Dierendonck, "GPS receiver architectures and measurements," Proceedings of the IEEE, vol. 87, no. 1, pp. 48-64, 1999

[6] Atmel, "ATR0603 – GPS Front End IC," Datasheet. Rev 3.5, Nov. 2006.

[7] Sige, "SE4120L – GNSS Receiver IC", datasheet. Rev 3.5, May 2009.

[8] NEMERIX. Datasheet "EB1006A – Evaluation Board for NJ1006A". October 2005. Rev 1.2.

[9] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Computers, vol. EC-8, pp. 330–334, Sept. 1959.

[10] P. Misra, and P. Enge, "Global Positioning System: Signals, measurements, and performance," Lincoln, Ganga-Jamuna Press, 2004, ISBN 0-9708544-0-9.

[11] Navstar Global Positioning System. Interface Specification, IS GPS-200, Revision D. 2004

[12] Galileo open service, signal in space interface control document, OS SIS ICD, Feb. 2008, Draft 1.

[13] P. Korvenoja, and R. Piché, "Efficient satellite orbit approximation," in Proc. of ION GPS 2000, Salt Lake City, USA, Sept. 19-22, 2000.

[14] R.G. Brown, and P.Y.C Hwang, "Introduction to random signals and applied kalman filtering," 3rd ed., John Wiley & Sons, 1997, ISBN 0-471-12839-2.

[15] Altera Web Server URL: http://www.altera.com/support/examples/nios2 /exm-micro_tutorial.html

[16] Google Maps, URL: http://maps.google.com/

PUBLICATION 2

T. Paakki, F. Della Rosa, S. Thombre, and J. Nurmi, “Profiling of GNSS Receiver Navigation Software on Embedded Processor”, in Proceedings of the *International Conference on Localization and GNSS (ICL-GNSS 2012)*, June 2012, in Starnberg, Germany.

Profiling of GNSS Receiver Navigation Software on Embedded Processor

Tommi Paakki, Francescantonio Della Rosa, Sarang Thombre, and Jari Nurmi
Department of Computer Systems
Tampere University of Technology
Tampere, Finland
{firstname.lastname@tut.fi}

Abstract— **New Global Navigation Satellite Systems like Galileo are expected to be fully operational in the future, thus increasing the availability of satellite signals for the GNSS receiver. As modern receivers have the capability of tracking different signals simultaneously, in this paper we evaluate how much stress all the available satellite signals will put on the processor responsible for calculating the PVT solution. We present C-language fixed-point method which does not require floating point support from the target processor, resulting in a target processor independent source code. The performance of the presented algorithms is shown in terms of clock cycles. In the end we demonstrate that it is possible to maintain a traditional 1Hz update rate on the processor even with high amount of satellites included into the navigation solution.**

Keywords - GNSS, GPS, Galileo, Receivers;

I. Introduction

Current Global Navigation Satellite Systems (GNSS) are the American GPS and the Russian Glonass. With the European Galileo and the Chinese Beidou systems under development, we expect a huge availability of signals in the future. Also regional signals like European Geostationary Navigation Overlay Service (EGNOS) and Wide Area Augmentation System (WAAS) will be of interest. This abundance of signals allows the GNSS users to enjoy of the increased availability of signals, especially in urban environment where the visibility of satellites is severely hindered. On the other hand, the GNSS receivers will be put on greater stress due to the processing of the available signals.

In this paper we concentrate on the computational burden that the navigation algorithms present on the GNSS receiver processor. GNSS related papers available in literature systematically leave the details of navigation algorithm open [1]. Another popular way to handle the navigation algorithm is to utilize external processor for final calculation [2]. One good reason for such solution is for example the lack of double precision variables on the processor, like in [3]. The availability of floating point support is an important factor when developing a GNSS application. A recent comparison of microprocessors for space-based navigation applications indicates that only two out of five studied microprocessors have floating point unit available [4]. One popular method to cope with the lack of floating point unit is to include a floating point co-processor to handle the floating point calculation on the receiver platform [5].

In this paper we present a software solution for the floating point requirement, aiming to improve the GNSS receiver source code portability for easier comparison of different target processors. We compare the native double precision arithmetic performance with full custom integer arithmetic instructions aimed for processors without floating point support. The increased availability of satellite signals also increases the processing time needed in the navigation software. This leads to the tradeoff between meeting the real time requirements, and choosing the optimal Receiver Autonomous Integrity Monitoring (RAIM) / Fault Detection and Exclusion (FDE) algorithms.

The paper is organized as follows: Section II describes the platform where the developed algorithms have been tested; Section III provides details of state of the art configuration in GNSS receivers for floating point support. The section also describes the software configuration implemented; Section IV introduces the software algorithms; Section V sums up the results, and finally, Section VI provides the conclusions.

II. TUTGNSS Development Platform

The TUTGNSS Reference Receiver is a platform developed at Tampere University of Technology (TUT), Department of Computer Systems (DCS) [6]. It is a university based hardware/software GNSS receiver where we have full control over its development, i.e. it contains no “black boxes” in the design. A driving factor has been to emphasize the research oriented aspects in the development, including the ability of getting results and debugging information from the platform.

The Radio Frequency (RF) Front-End is a third party hardware. TUTGNSS can be configured with a variety of front-ends, thanks to the internal baseband converter unit, which changes any type of RF Front-End input to common three bit I&Q signed presentation. It also removes the Intermediate Frequency (IF) so that acquisition and tracking only need to account for the Doppler frequency. Current configuration uses Maxim 2769 RF Front-End [7].

TUTGNSS is running on a Field Programmable Gate Array (FPGA), and current design uses Altera Stratix II DSP development board (EP2S180) [8]. The design is platform independent, and can be mapped to different FPGA with minor modifications. The hardware parts of TUTGNSS are implemented in VHDL language. For the software processes a

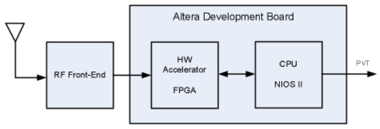


Figure 1. Architecture of TUTGNSS

NIOS II softcore processor is implemented on the FPGA. The clock frequency for the system runs at 100MHz. An overview of the TUTGNSS structure is given in Figure 1.

The receiver follows hardware accelerated software receiver architecture, where hardware logic performs correlations for acquisition and tracking, and software controls all events that take place in the hardware. The selected Real-Time Operating System (RTOS) is MicroC/OS-II from Micrium Inc [9].

The operating system handles and drives software processes by control process which implements Interrupt Service Routines (ISR) according to priorities. The main software processes of the receiver are identified in Table I. The source codes for software are coded in C-language, and the used compiler is a Nios II gcc. Algorithms performances are presented both without compiler optimization and with optimization level 3.

III. Software Configuration

In the early stages of TUTGNSS development we had noticed that 32 bit floating point accuracy was not sufficiently accurate for GNSS navigation algorithm application. The current platform with NIOS II softcore does support the 64 bit double precision arithmetic, which is now sufficient for the navigation algorithm. While the development of the algorithm in double precision is straightforward, we feel that the existence of double precision arithmetic is a limiting factor of the GNSS design. Moreover, for embedded systems, the preferred solution would be the use of integer arithmetic. From these starting points we wanted to take the additional challenge of developing navigation software independent of the target processor. The main requirements for the processor would be: existing C-compiler, and 32 bit integer variables.

A. TUTGNSS Fixed-Point Variables

The starting point for selecting the fixed-point format is to figure out what range of values are encountered in typical navigation algorithm calculations. Analysis of the navigation algorithms revealed that most of the numbers we deal with are small, like most of the ephemeris parameters with scale factors less than one [10]. From the constants speed of light is example of a big number, *C = 299 792 458 m / s*. Speed of light fits into 32 bit integer variable, hence the decision of maximum number was set to 2^{32}. The amount of bits we dedicate for the decimal parts set the final granularity, i.e. the accuracy of the fixed-point variables. The final version of implemented fixed-point arithmetic has 2*32 = 64 bits for decimal part, setting the granularity to 2^{-64}, providing very good accuracy. Figure 2 illustrates the structure of TUTGNSS fixed-point variable.

B. Fixed-Point Arithmetic

In order to utilize the presented fixed-point variables a whole set of basic operations need to be written. A list of available functions for these arithmetic operations is presented in Table II.

Short elaboration for the division operation is needed. We have implemented a function for converting the division into a multiplication using binomial theorem. Assume we need to calculate z / x. First step is to solve 1 / x = y, which is exactly what our *Div* function does. Then z / x = z * y, where division has now been converted into a multiplication.

The *Mul2* and *Div2* functions are for quick multiplication / division with 2 to the power of N. In other words these are shift functions. Whenever possible, these should be utilized instead of the ordinary *Mul* / *Div* functions.

Table I. TUTGNSS Processes with associated priorities

Process name	Priority (Note: lower value for higher priority)
Receiver Control	1
Tracking	2
Acquisition	3
Bit Decoding	4
Navigation Solution	5

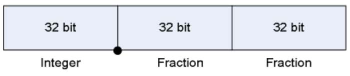


Figure 2. TUTGNSS Fixed-Point Format

Table II. Fixed-Point Arithmetic

Operation	Function
Add	Add(x,y)
Sub	Sub(x,y)
Mul	Mul(x,y)
Div	Div(x)
Mul2	Mul2(x)
Div2	Div2(x)

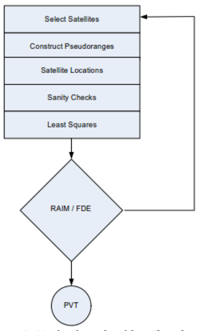


Figure 3. Navigation Algorithm Flowchart

IV. Implemented Algorithms

In this paper we concentrate on the basic algorithms for calculating the final Position, Velocity, and Time (PVT) solution. The basic textbook combination of algorithms needed is illustrated in Fig. 3 [11].

As it can be seen from the equations of [11], the calculation of satellite position is the most computationally demanding task of the PVT software. This part requires heavy use of mathematical functions such as *sqrt()*, *sin()*, *cos()* and *atan2()*. These mathematical functions are provided by the C math library, but they cannot be used directly with fixed-point arithmetic, like in TUTGNSS. The workaround for this problem is to implement the aforementioned mathematical functions with the help of Coordinate Rotation Digital Computer (CORDIC) [12].

The basic principle of CORDIC is to calculate trigonometric functions using only addition, subtraction, bit shift and table-lookup operations. Also the CORDIC is an iterative algorithm where each iteration is expected to increase the accuracy of the solution by 1 decimal. A safe accuracy of the results is achieved with 32 or more iterations.

V. Results

The presented algorithms have been evaluated with the Altera Performance Counter unit, a block provided by the Quartus II System-On-a-Programmable-Chip (SOPC) builder. The performance counter allows measuring the performance of code section in terms of clock cycles, and consecutively time, spent in the section. Otherwise the TUTGNSS setup is kept in the normal configuration. Additionally, the performance of the algorithms is compared with having the compiler option for optimization turned on and off.

A. Arithmetic Operations

First evaluation is the performance of the basic arithmetic operations in the chosen fixed-point format. Each arithmetic operation was evaluated in own section in the performance counter. Figure 4 and Figure 5 show the non-optimized results and the improvement with optimization respectively, highlighting how the optimization does great job in reducing the amount of cycles needed to calculate these operations. Furthermore, as expected, multiplication and especially division are operations that need considerably more processing than other operations.

B. CORDIC Functions

The fixed-point arithmetic cannot directly use the ready available math library, thus CORDIC implementations are needed for *sqrt(), cos(), sin()* and *atan()* functions. CORDIC is an iterative method, so the performance of each function was evaluated with variable iteration amount. The tested iteration amount ranges from 30 to 57. Performance results can be seen in Figure 6. As expected, the amount of cycles needed grows with the iteration rounds. The cost of extra iterations is much higher on the non optimized code.

C. Satellite Locations

For the navigation algorithm part we are interested in the effect of having more satellites in the navigation solution. Any additional satellite in the solution increases the processing time needed for these algorithms. This becomes an important issue when the receiver is capable of receiving signals from multiple systems, like combined GPS, Galileo and Glonass, effectively multiplying the available amount of satellites. Figure 7 shows how the non-optimized fixed-point algorithm performs against the double precision version. The used iteration amount for CORDIC is 51, yielding maximum accuracy. The optimized version results are given in Figure 8, showing that the optimization does not have considerable effect on the native double precision version. In this test satellite location was calculated for 8 satellites.

D. Sanity Checks

After performed the calculation of satellite location we have implemented an additional sanity check step. A check is made, that the calculated satellite location is approximately 20000 km from the user location. Pseudorange measurements are also corrected with the timing information we have obtained. Pseudoranges are expected to match the same sanity levels. Should any of these sanity checks fail, we can still exclude that particular satellite from the list of accepted satellites before entering the Least Squares (LS) part of the software. Last important issue included in this phase is the Sagnac effect correction for the satellite location [11]. The Sagnac correction contains *sin()* and *cos()* functions, thus somewhat increasing the processing time of this part. The performance report for non-optimized version is given in Fig. 9, and the optimized version in Fig. 10.

E. Least Squares

The LS is the final part in our analysis. In LS algorithm we utilize matrix structures to obtain the PVT solution. The size of the matrices increase with each additional satellite included in the LS algorithm, thus consequentially increasing the processing time of the algorithm. The minimum amount of satellites needed to obtain the PVT is 4. On the other hand 8 satellites present a situation with high amount of satellites.

LS is an iterative algorithm, and the amount of iterations needed is not always fixed. In most cases 5 iterations are enough for the LS to converge. It is safe to assume that 5 iterations are more than LS usually needs, thus setting a worst case performance scenario. The performance result for the LS algorithm is presented with different amount of satellites included in the LS part. This amount is set between 4 and 8. Results are shown in Fig. 11.

F. *Summary*

To sum previous results, the worst case scenario for computational burden is when:

- No compiler optimization is used
- Maximum amount of iterations for CORDIC
- Maximum amount of satellite location calculations
- Maximum amount of satellites in LS with maximum Iterations

The computational burden budget for this setup is presented in Table III. Similarly, the budget for optimized code is listed in Table IV.

Finally, the whole TUTGNSS functionality with the custom fixed-point arithmetic is tested with a Spirent STR4500 GPS/SBAS Simulator System [13]. The scenario consisted of normal GPS constellation with 8 satellites in view for significant computational burden. The x, y, z and altitude errors have been plotted for 4 minute test run on figure 12 and 13 respectively. The exact coordinates of the receiver location are as follows:

X = 2795201.83 m
Y = 1235745.80 m
Z = 5579480.80 m
Altitude = 0.0 m

```
--Performance Counter Report--
Total Time: 0.00045391 seconds  (45391 clock-cycles)
+----------------+-----+-----------+---------------+-----------+
| Section        |  %  | Time (sec)|  Time (clocks)|Occurrences|
+----------------+-----+-----------+---------------+-----------+
|ADD             | 1.76|    0.00001|            801|          1|
+----------------+-----+-----------+---------------+-----------+
|SUB             | 2.13|    0.00001|            965|          1|
+----------------+-----+-----------+---------------+-----------+
|MUL_LONG        |  7.6|    0.00003|           3450|          1|
+----------------+-----+-----------+---------------+-----------+
|MUL_SHORT       |  5.9|    0.00003|           2677|          1|
+----------------+-----+-----------+---------------+-----------+
|DIV             |   81|    0.00037|          36747|          1|
+----------------+-----+-----------+---------------+-----------+
|MUL2            |0.822|    0.00000|            373|          1|
+----------------+-----+-----------+---------------+-----------+
|DIV2            |0.762|    0.00000|            346|          1|
+----------------+-----+-----------+---------------+-----------+
```

Figure 4. Performance report for non-optimized arithmetic functions

```
--Performance Counter Report--
Total Time: 6.741E-05 seconds  (6741 clock-cycles)
+----------------+-----+-----------+---------------+-----------+
| Section        |  %  | Time (sec)|  Time (clocks)|Occurrences|
+----------------+-----+-----------+---------------+-----------+
|ADD             | 2.45|    0.00000|            165|          1|
+----------------+-----+-----------+---------------+-----------+
|SUB             |  3.1|    0.00000|            209|          1|
+----------------+-----+-----------+---------------+-----------+
|MUL_LONG        | 6.68|    0.00000|            450|          1|
+----------------+-----+-----------+---------------+-----------+
|MUL_SHORT       | 5.82|    0.00000|            392|          1|
+----------------+-----+-----------+---------------+-----------+
|DIV             | 79.4|    0.00005|           5350|          1|
+----------------+-----+-----------+---------------+-----------+
|MUL2            |  1.1|    0.00000|             74|          1|
+----------------+-----+-----------+---------------+-----------+
|DIV2            | 1.05|    0.00000|             71|          1|
+----------------+-----+-----------+---------------+-----------+
```

Figure 5. Performance report for optimized arithmetic functions

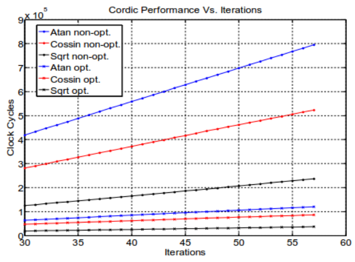


Figure 6. Cordic performance vs. iterations

```
--Performance Counter Report--
Total Time: 0.590451 seconds  (59045077 clock-cycles)
+----------------+-----+-----------+---------------+-----------+
| Section        |  %  | Time (sec)|  Time (clocks)|Occurrences|
+----------------+-----+-----------+---------------+-----------+
|Fixed-Point     | 95.2|    0.56185|       56185493|          1|
+----------------+-----+-----------+---------------+-----------+
|Double          | 4.84|    0.02860|        2859572|          1|
+----------------+-----+-----------+---------------+-----------+
```

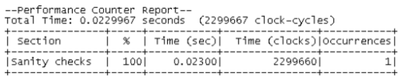
Figure 7. Performance report for calculating satellite position for 8 satellites Non-optimized code

```
--Performance Counter Report--
Total Time: 0.120255 seconds  (12025512 clock-cycles)
+----------------+-----+-----------+---------------+-----------+
| Section        |  %  | Time (sec)|  Time (clocks)|Occurrences|
+----------------+-----+-----------+---------------+-----------+
|Fixed-Point     | 76.1|    0.09152|        9151529|          1|
+----------------+-----+-----------+---------------+-----------+
|Double          | 23.9|    0.02874|        2873973|          1|
+----------------+-----+-----------+---------------+-----------+
```

Figure 8. Performance report for calculating satellite position for 8 satellites Optimized code

```
--Performance Counter Report--
Total Time: 0.139145 seconds  (13914493 clock-cycles)
+----------------+-----+-----------+---------------+-----------+
| Section        |  %  | Time (sec)|  Time (clocks)|Occurrences|
+----------------+-----+-----------+---------------+-----------+
|Sanity checks   |  100|    0.13914|       13914485|          1|
+----------------+-----+-----------+---------------+-----------+
```

Figure 9. Performance report for sanity checks and Sagnac correction Non-optimized code



```
--Performance Counter Report--
Total Time: 0.0229967 seconds  (2299667 clock-cycles)
+----------------+-----+------------+---------------+-----------+
| Section        |  %  | Time (sec)|  Time (clocks)|Occurrences|
+----------------+-----+------------+---------------+-----------+
|Sanity checks   |  100|     0.02300|        2299660|          1|
+----------------+-----+------------+---------------+-----------+
```

Figure 10. Performance report for sanity checks and Sagnac correction Optimized code

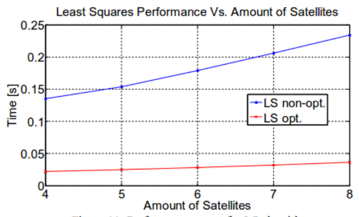


Figure 11. Performance report for LS algorithm (5 iterations) vs. amount of satellites included

TABLE III. TOTAL BUDGET FOR NON-OPTIMIZED PVT CODE

Task	Time [s]
Satellite location	0.57
Sanity checks	0.14
LS	0.23
Total =	0.94

TABLE IV. TOTAL BUDGET FOR OPTIMIZED PVT CODE

Task	Time [s]
Satellite location	0.10
Sanity checks	0.03
LS	0.04
Total =	0.17

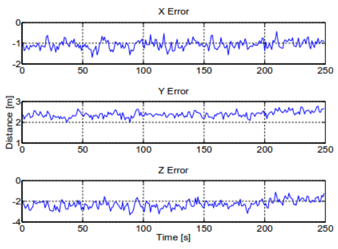


Figure 12. x, y, z – coordinate error

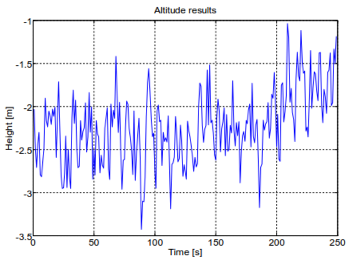


Figure 13. Altitude error

VI. CONCLUSIONS

In this paper we presented the computational demands of various parts of the navigation software. The source codes were written in standard C-language using custom fixed-point arithmetic resulting in target processor independent solution. Only assumption is that the target processor supports 32 bit integer variables.

The performance results show that it is possible to produce PVT solution at standard 1 Hz rate, even without using any compiler optimization. Running applications with single CPU platform, like TUTGNSS, this is not possible due to other processes, like acquisition and tracking control, occupying the processor part of the time. Hence, it is strongly recommended to use the compiler optimization option. In our case the compiler succeeded in reducing the computational amount by 80%. A potential alternative solution is to implement second dedicated processor for the navigation software only. The Stratix II FPGA has plenty of resources free for such implementation.

The optimized code performance leaves safe amount of overhead for future extension to multi-system receiver with operability with Galileo. With Galileo satellites the amount of satellites in view is effectively doubled, and consequentially computational demands increase. With more than 8 satellites in view, it may be favored solution to not to use all the available satellites in view, but select an optimal subset of satellites. It is good to keep in mind that the implemented RAIM / FDE algorithm take some processing time and should be accounted for in the total computational burden budget in order to meet the critical real-time capability.

For future optimization, advanced features like satellite orbit approximation methods can be used [14]. Also, the number of iterations needed in a particular spot in the navigation algorithm can be further investigated, further reducing computational demand. Finally, the fixed-point method performance can also be compared to floating point coprocessor implementation in terms of FPGA resource consumption.

REFERENCES

[1] P. Mumford, K. Parkinson, and A. Dempster, “An Open GNSS Receiver Research Platform,” In IGNSS Symposium 2006, 2006.

[2] T. Lück, J. Winkel, M. Bodenbach, E. Göhler, N. Falk, A. Consoli, F. Piazza, D. Gerna, R. Granger, P. Readman, S. Simpson, H-J. Euler, “Artus – A second generation Galileo/GPS receiver,” in Proc of ION GNSS 2007, September 25-28, 2007, Fort Worth, TX.

[3] B. Sauriol, and R. Landry, “FPGA-Based Architecture for High Throughput, Fleixble and Compact Real-Time GNSS Software Defined Receiver,” in Proc. Of the ION NTM 2007, Jan. 22-24, 2007, San Diego, CA.

[4] S. De Florio, E. Gill, S. D’Amino, and A. Grillenberger, “Performance Comparison of Microprocessors for Space-Based Navigation Applications,” in Proc of 7th IAA Symposium on Small Satellites for Earth Observation, May 2009

[5] G. Kappen, S. Bahri, O. Priebem and T. G. Noll, “Implementation of a CORDIC based Double Precision Floating Point Unit Used in an ASIP Based GNSS Receiver,” Timenav’07, Geneva, 29.5-1.6.2007.

[6] T. Paakki, J. Raasakka, F. Della Rosa, H. Hurskainen, and J. Nurmi, ”TUTGNSS – University Based Hardware/Software GNSS Receiver for research Purposes,” in Proc. Of the UPINLBS 2010 conference, Oct. 14-15, 2010, Kirkkonummi, Finland.

[7] Maxim 2769, URL: http://www.maxim-ic.com/

[8] Altera Stratix II, URL: http://www.altera.com/

[9] MicroC/OS-II, URL: http://www.micrium.com/

[10] Navstar Global Positioning System. Interface Specification, IS GPS-200, Revision D. 2004.

[11] E. D. Kaplan, and C. J. Hegarty, “Understanding GPS – Principles and applications,” 2nd ed., Norwood, Artech House, 2006, ISBN-10:1-58053-894-0.

[12] J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Trans. Electron. Computers, vol. EC-8, pp. 330–334, Sept. 1959.

[13] Spirent STR4500, URL: http://www.spirent.com/

[14] P. Korvenoja, and R. Piché, “Efficient satellite orbit approximation,” in Proc. of ION GPS 2000, Salt Lake City, USA, Sept. 19-22, 2000.

PUBLICATION 3

T. Paakki, J. Raasakka, F. Della Rosa, and J. Nurmi, “Efficient Bit Decoding Implementation for Mass Market Multi-Constellation GNSS Receivers”, in Proceedings of the *Design & Architectures for Signal & Image Processing (DASIP 2013),* October 2013, in Cagliari, Italy.

Efficient Bit Decoding Implementation for Mass Market Multi-Constellation GNSS Receivers

Tommi Paakki, Jussi Raasakka, Francescantonio Della Rosa, and Jari Nurmi
Department of Electronics and Communications Engineering
Tampere University of Technology
Tampere, Finland
{firstname.lastname@tut.fi}

Abstract—**The modern satellite signals have improved their bit transmission robustness with adding new protection methods like block-interleaving and Forward Error Correction. While these methods provide clear improvement in challenging environment, the implementation of the decoders on receiver side adds more complexity. In this paper we introduce efficient methods for block de-interleaving and for Viterbi decoders suitable for Galileo, GPS L5, and WAAS/EGNOS signals. The presented algorithms' performance is evaluated and together with the resource consumption the results are presented. The aim of the paper is to help GNSS receiver developers to find the choose whether to use software or hardware algorithm in their receiver.**

Keywords – Satellite navigation systems, Receivers, Viterbi algorithm, Benchmark testing;

I. INTRODUCTION

At the moment we have few Global Navigation Satellite System (GNSS) systems which have adapted advanced bit encoding methods. These bit encoding methods are to improve the correct decoding of the navigation data bits on a GNSS receiver. The benefit of the advanced bit encoding is most notable in case of challenging environment, i.e. possible interference combined with low signal power. This paper concentrates on the efficient implementation of receiver side block de-interleaver and Viterbi decoder. These navigation data processing blocks are requirement for receivers that want to utilize the navigation message provided by Galileo, GPS L5 or WAAS/EGNOS signals.

While the bit encoding techniques improve the bit decoding performance, it comes with a cost. The receiver needs dedicated hardware or software solution to provide the service. In GNSS systems particular, the decoders are dealing with relatively low symbol rates, e.g. 250 symbols per second for Galileo I/NAV message on E1-B band. On the other hand, the decoding moments appear heavily on bursts. These bursts occur with the frame rate of the GNSS system. That means that all the satellites that are being tracked from a particular GNSS will receive full frames within few milliseconds of each other. This is due to the fact that while the frames are synchronized between the satellites, the receiver will receive them at slightly different times due to different propagation delays.

In this paper we introduce how to efficiently implement the bit decoding capabilities mentioned. While there are many commercially available Intellectual Property (IP) cores for Viterbi decoders, for example [1], from the open educational point of view it is beneficial to implement fully configurable custom design of Viterbi decoder. There exists already a wide selection of papers about Galileo/GPS L5 ready receivers, but to best of authors' knowledge, these papers do not mention implementation details, or performance, of the Viterbi decoder. A list of few such Galileo/GPS L5 receivers can be found in [2] – [5]. One interesting paper considering Galileo/Beidou capable Viterbi implementation is described in [6]. In our paper we will provide benchmarking results to help GNSS receiver developers decide whether they want to implement the Viterbi functionalities in hardware or software. The choice depends on the available resources of the target receiver platform. In real time receivers, it is the complexity of the algorithms, which sets a limit on what can be achieved with software solution.

This paper is divided into the following sections. First, Section II gives a brief overview of the TUTGNSS Reference Receiver architecture and specifications. Then, Section III provides detailed block by block description of the implemented bit decoding techniques. Section IV provides benchmarking results obtained with the TUTGNSS Reference Receiver. And finally, Section V concludes the presented work and discusses future plans.

II. DEVELOPMENT ENVIRONMENT

The implemented algorithms have been realized on TUTGNSS Reference Receiver [7], which has been developed at Tampere University of Technology (TUT) for educational purposes. TUTGNSS is a hardware/software platform for GNSS receiver algorithm testing and it contains no "black boxes", allowing users learn / further develop any part of the receiver.

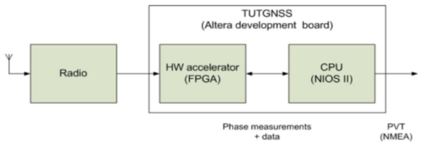


Figure 1. Architecture of TUTGNSS receiver.

The TUTGNSS Reference Receiver uses hardware-accelerated software receiver architecture, where hardware logic performs correlations for acquisition and tracking, and software controls all events that take place in the hardware. Figure 1 shows the high level architecture of TUTGNSS Reference Receiver. Current Field Programmable Gate Array (FPGA) used for development of TUTGNSS is an Altera Stratix II DSP development board (EP2S180) [8], but in principle the design should be platform independent. The hardware parts are implemented in VHDL hardware description language. The software processes are executed on NIOS II softcore processor implemented within the FPGA. Software processes are handled and driven by control process which implements interrupt service routines according to priorities. The source codes for software have been written in C-language.

III. IMPLEMENTED ALGORITHMS

A. Viterbi Decoder

The GNSS signals can be exposed to bit errors while propagating through the channel. These errors tend to appear in bursts. A single long erroneous bit sequence has more severe effect on convolutional encoded bit streams than multiple separate bit errors. The convolutional encoding scheme used in GNSS applications is theoretically able to correct sequential bit errors up to :

$$t = \lfloor (d_{free} - 1)/2 \rfloor, \qquad (1)$$

where d_{free} is maximum free distance of the code. For Galileo code, maximum free distance is 10 [8]. Thus the receiver is able to correct error bursts of up to 4 bits.

A custom Viterbi decoder was designed having particularly the newly available Galileo message structure in mind. First choice was to be made between soft-decision and hard-decision Viterbi decoder. Since hard-decision Viterbi decoder is less complicated to implement, and it better interfaces with TUTGNSS, it was selected as the operating mode. Another design decision is the traceback depth of the decoder [9]. A traceback depth of 5 x K is sufficient for the performance, where K is the constraint length of the convolutional encoder. For Galileo case K = 7, making traceback depth of 35 sufficient. Considering that the length of the I/NAV message to be decoded is 240 symbols, we decided to implement a decoder with traceback depth of 40 [10]. This selection makes the message length divisible with the traceback depth, and no redundant work is done. This means that Viterbi decoder takes 80 symbols in, and output will be 40 decoded bits. Thus, decoding of full 240 symbol message takes three rounds.

For the Viterbi decoder custom software and hardware solutions were developed to evaluate their feasibility on a mass market GNSS receiver, as shown in Figure 2. The software solution was written in a C-language function form where encoded bits are given as input, and output will be the decoded bits. Any satellite with full 240 symbols received will call the function in order to have the bits decoded.

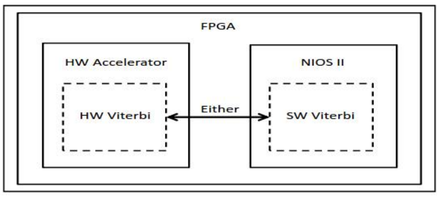


Figure 2. Implemented Viterbi Units.

The hardware Viterbi Unit (VU) is written in VHDL language, having the similar input / output structure as the software version. Initial plan was to assign one VU per channel, so that each channel could use the decoder without congestion or queuing. While the idea is valid, it would also be too wasteful on the resources, making such implementation expensive. Instead, the final architecture of TUTGNSS contains only one VU with queuing mechanism to serve all the satellites needing their data to be decoded. The architecture of TUTGNSS can tolerate the variable delay of the queued Viterbi decoding, since phase measurements are taken already on the sample level, and not on the bit level. Figure 3 provides the overview of the TUTGNSS on how the phase measurements are taken. Details of the units can be found in reference [7]. The delays caused by the Viterbi decoding have minor effect on Time-To-Alert (TTA) response in the receiver. If some anomalies are detected in the satellite, the satellite data can indicate these issues, and the time to decode the message constitutes to the total TTA time. On mass market receivers the additional delay caused by the queued Viterbi is tolerable, and Receiver Autonomous Integrity Monitoring (RAIM) / Fault Detection and Exclusion (FDE) algorithms are expected to correct possible issues.

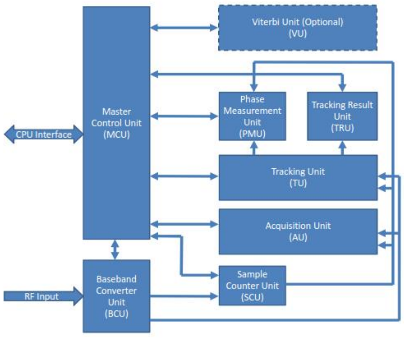

Viterbi Unit (Optional) (VU)
Master Control Unit (MCU)
CPU Interface
Phase Measurement Unit (PMU)
Tracking Result Unit (TRU)
Tracking Unit (TU)
Acquisition Unit (AU)
Sample Counter Unit (SCU)
Baseband Converter Unit (BCU)
RF Input

Figure 3. Hardware Block Diagram of TUTGNSS.

B. On-the-Fly Block De-Interleaving

The idea of the block interleaving is to scramble the initial signal before transmitting it. Later when performing the de-interleaving, on the receiver side, these error bursts will in turn become scrambled. As mentioned, the scrambled bit errors can be corrected by the Viterbi decoder. In short, block interleaving is to protect the bit payload against error burst.

The common method of implementing bit interleaving suggests collecting the whole interleaved frame on the receiver side. After reception, the de-interleaving process is performed. While this approach is entirely valid and very straightforward to implement, it has one slight downside. The downside is once again related to the computational load that appears in bursts. After waiting for the whole frame to arrive, the processor suddenly needs to perform the de-interleaving and consecutive Viterbi decoding in a burst.

To alleviate the burst nature of de-interleaving we have implemented an on-the-fly method for de-interleaving while receiving the frame. This method will balance the processor computational burden better over time. After the receiver has synchronized to the frame structure, we know beforehand where the incoming bits will be placed in the de-interleaving process. Implementation of this on-the-fly method will be described based on the following implementation detail - the algorithm assumes that the received data bits are stored as bits on the processor. As an advantage, the bit implementation saves memory. The disadvantage is that accessing individual bits require shifting and masking operations. This storage format is illustrated in Figure 4. The alternative would be to store received bits into bytes, or into larger variables. While such approach is also viable, the presented formulas will concentrate on the bit implementation. Also, as discussed in the Viterbi section, 80 symbols will the decoded at once. To save those 80 symbols, three 32 bit variables are used in the following algorithm. The algorithm for on-the-fly de-interleaving is presented in Table 1.

IV. Results

In this section the essential performance versus hardware logic consumption analysis is provided. Understanding the tradeoff of these metrics is helps GNSS receiver developers to make the choice between hardware / software implementation on their platform.

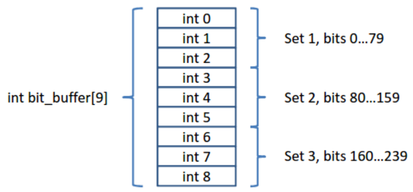


Figure 4. TUTGNSS Frame Storage Format.

Table 1. Pseudocode of On-the-Fly De-Interleaving.

```
// i = counts incoming bits 0…239 after
//     framestart
// databit = incoming bit

pos=8*(i%30)+(i/30);  // bit position 0...239
set=pos/80;           // which set of 3
bit=pos-set*80;       // which bit in the set
sub=bit/32;           // which of the 3 ints
if(bit>63)            // shift amount
   shift=15-bit%16;
else
   shift=31-bit %32;

result=3*set+sub;
databit=databit<<shift;
bit_buffer[result] |= databit;
```

A. Resource Utilization

First we present the hardware logic utilization of TUTGNSS receiver. In Fig. 5 the current resource consumption for Altera Stratix II FPGA is presented. In numbers, VU takes 5767 LC Combinationals, and 5504 LC Registers. From the Fig. 5 it can be noticed that the 16 available tracking channels is the most logic consuming unit, taking approximately 50% of the total TUTGNSS size. Second place goes for the acquisition unit, and close to that comes the implemented Viterbi block. In other words, a single Viterbi entity is quite close to acquisition unit in terms of logic utilization. Directly replicating the Viterbi unit for all the 16 channels is not a desired approach, since it would be a wasting too much logic, making the design too expensive. The single Viterbi approach consumes the least resources, but is prone to short queuing delays for multiple channels.

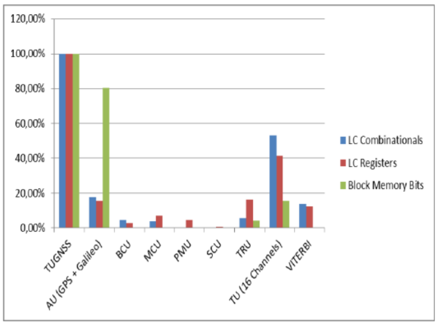


Figure 5. TUTGNSS hardware synthesis results

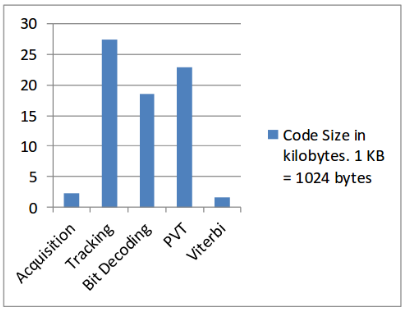


Figure 6. TUTGNSS compiled software code size

The software resource utilization is briefly analyzed in terms of compiled code size. As seen in Fig. 6, the software Viterbi is only a minor part of the whole TUGNSS software size. For simple comparison Fig. 6 lists the code size of the main software tasks of TUTGNSS. Normally software Viterbi would be associated with the Bit Decoding task, but in the Fig. 6 Viterbi code size has been excluded from Bit Decoding size. From the code size point of view, the software Viterbi is good selection since it does not take excessive amount of resources to implement. The total software size of TUTGNSS, including all system libraries, is 416 KB (data + initialized data).

B. Hardware / Software Performance

The main reason to implement algorithms on hardware is speed. The Viterbi unit is a good example of an algorithm that has excellent performance on hardware. The downside of the hardware implementation is that it takes resources which make the design more expensive, as presented in previous section. The software solution is expected to be slower, but on the other hand it does not add to the hardware consumption. The deciding factor between hardware and software will be whether the processor can process all the required instructions in a reasonable time. Many operations of a GNSS receiver need to meet real-time requirement [11].

Figure 7 presents the simulation waveform of the Viterbi unit performed under Questa Sim 10.0c. The individual internal signals seen on the figure are not covered on this paper. Instead the interesting parameter is the end of the simulation time, presented by the yellow vertical line in the figure. The stop time indicates 64700 ns, and the simulation was run under 100 ns clock cycle. The speed of the hardware Viterbi unit is thus 647 clock cycles for processing whole Galileo 240 symbol frame. TUTGNSS clock frequency is 87 MHz, which results a delay of

647 cycles / (87 Mcycles / s) ≈ 7.5 µs.

An interesting testing result is the performance of the software Viterbi running on the actual Nios II core. The benchmarking of the algorithm was made using Altera Performance Counter unit, a block provided by the Quartus II

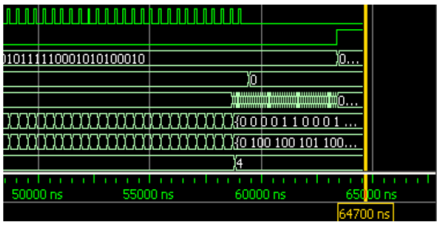


Figure 7. Simulation Waveform of Viterbi Decoder - Decoding Time

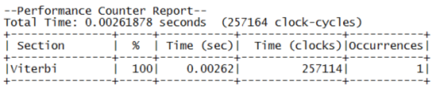

```
--Performance Counter Report--
Total Time: 0.00261878 seconds  (257164 clock-cycles)
+---------------+-----+------------+----------------+-----------+
| Section       |  %  | Time (sec)|  Time (clocks)|Occurrences|
+---------------+-----+------------+----------------+-----------+
|Viterbi        |  100|    0.00262|          257114|          1|
+---------------+-----+------------+----------------+-----------+
```

Figure 8. Performance Report of Software Viterbi Decoder

System-On-a-Programmable-Chip (SOPC) builder. The performance counter allows measuring the performance of code section in terms of clock cycles, and consecutively time spent in the code segment. Figure 8 shows the performance report for one call of the Viterbi function (decoding of 80 symbols). In order to get the total time of decoding full frame, the time must be multiplied with 3. This yields a decoding delay of approximately 7.5 ms on Nios II core.

Remaining issue with the software benchmarking is the actual architecture of TUGNSS receiver, where the single core runs all the software tasks based on interrupts and priorities. This will cause interrupts to the software Viterbi decoding due to the higher priority tasks requesting the processor time. Regardless, the 7.5 ms gives the approximate scale about how long it takes to decode one full frame with the software approach. Naturally, with a single core platform, such as TUTGNSS currently is, multiple satellites need to queue to get their messages decoded with the software Viterbi decoder. In this sense the single hardware VU implementation acts much like the software Viterbi.

For verification the software implementation was also tested with Matlab, to see that the algorithm is working as intended. The Matlab inbuilt Viterbi decoder was compared to the TUGNSS implementation and no difference in behavior was observed. Both hardware and software approaches were tested with real Galileo satellite signal.

Obtained Performance on Altera Stratix II:

- Hardware Viterbi decoding delay 7.5 µs
- Software Viterbi decoding delay 7.5ms

V. CONCLUSIONS AND FUTURE WORK

In this paper we described simple on-the-fly method for fast block de-interleaving, which helps to lessen the computational burst of normal block de-interleaving. Also, a hardware solution of single Viterbi Unit was described, together with equivalent software approach. Both methods were benchmarked in terms of resources consumed (compiled software code size / used hardware resources) and obtainable performance (time to decode a message in software / hardware Viterbi). From the results it can be seen that hardware Viterbi is faster than the software alternative by a factor of 1000. If the higher delay of the software Viterbi can be tolerated, and there is enough processor capacity to run all the software tasks to meet real time requirements, then software Viterbi is viable solution for mass market GNSS receivers.

The future plans of TUTGNSS are to further improve the Hardware Viterbi solution by optimizing the resource usage. One method is to utilize onboard RAM memory to handle some of the memory requirements of the Viterbi. Additionally, TUTGNSS can be configured as multiprocessor platform by including one or more NIOS II cores. Additional NIOS cores will require some extra hardware resources, but on the other hand it provides much more computational power for all the software parts.

REFERENCES

[1] Altera Viterbi Compiler, URL: http://www.altera.com/products/ip/dsp/error_detection_correction/m-alt-viterbi-compiler.html

[2] C. Mongrédien, M. E, Cannon, and G. Lachapelle, "Performance Evaluation of a GPS L5 Software Receiver Using a Hardware Simulator," in Proc. of the ENC 07, Geneva, Switzerland, 29May-June 1, *2007*

[3] P. Berglez, "Development of a GPS, Galileo and SBAS receiver," in Proc. of the ELMAR-2008, 10-12 September 2008, Zadar, Croatia

[4] N. Gerein, and M. Olynik, "Prototype Galileo Receiver Development," in Proc. of the ENC GNSS 2004, 16-19 May 2004, Rotterdam, The Netherlands

[5] T. Lück, J. Winkel, M. Bodenbach, E. Göhler, N. Falk, A. Consoli, F. Piazza, D. Gerna, R. Granger, P. Readman, S. Simpson, H-J. Euler, "Artus – A second generation Galileo/GPS receiver," in Proc of ION GNSS 2007, September 25-28, 2007, Fort Worth, TX.

[6] T. Ma, X. Cui, M. Lu, and Z. Feng, "The Optimization of Algorithm and Implementation for Dual-Constellation Navigation Receiver," in Proc. of the ION GNSS 2010, September 21-24, 2010, Portland, OR.

[7] T. Paakki, J. Raasakka, F. Della Rosa, H. Hurskainen, and J. Nurmi, "TUTGNSS – University Based Hardware/Software GNSS Receiver for research Purposes," in Proc. of the UPINLBS 2010 conference, Oct. 14-15, 2010, Kirkkonummi, Finland.

[8] Altera Stratix II, URL: http://www.altera.com/literature/manual/mnl_stx2_pro_dsp_dev_kit_ep2s180.pdf

[9] J. G. Proakis, Digital Communications, 3rd ed., McCraw-Hill, 1995, ISBN 0-07-051726-6.

[10] Galileo Open Service, Signal in Space Interface Control Document, OS SIS ICD, Revision 1, Sep. 2010

[11] T. Paakki, F. Della Rosa, S Thombre, and J. Nurmi, "Profiling of GNSS Receiver Navigation Software on Embedded Processor," in Proc. of the ICL-GNSS 2012, Starnberg, Germany, June 25-27, 2012

PUBLICATION 4

T. Paakki, F. Della Rosa, H. Hurskainen, and J. Nurmi, “Navigation Algorithm Test Environment for GNSS Receivers”, in Proceedings of the *European Navigation Conference (ENC GNSS 2010)*, October 2010, in Braunschweig, Germany.

Reprinted with kind permission.

Navigation Algorithm Test Environment for GNSS Receivers

Tommi Paakki, Francescantonio Della Rosa, Heikki Hurskainen and Jari Nurmi
Department of Computer Systems
Tampere University of Technology
Tampere, Finland
Email: {firstname.lastname@tut.fi}

Abstract— **New global navigation satellite systems are rapidly developing. Receivers that can operate on multiple satellite systems offer better availability of satellites, and better navigation performance. With the availability of the new systems the complexity of the receiver design is increased. In this paper we present an educational architecture for development and evaluation of new navigation algorithms. We present a navigation algorithm test environment which is capable of performing multiple navigation solutions in parallel. This architecture is implemented both on a FPGA, and C++ software platforms. In order to further improve the educational aspects of the presented receiver architecture, we have also implemented a separate debug domain which allows easy access for logging data in the hardware parts of the receiver. In this paper we evaluate few select algorithms with the navigation algorithm test environment and present the results. We believe that the presented architecture provides significant educational platform for evaluating and developing new navigation algorithms for global navigation satellite systems.**

Keywords-GNSS, Receivers, Education

I. Introduction

New Global Navigation Satellite Systems (GNSS) are rapidly developing, like the European Galileo system, Russian GLONASS, and the modernization of the American Global Positioning System (GPS). This development has created a need for a general, easily adjustable receiver platform on which novel algorithms may be tested and evaluated. In this paper we present the TUTGNSS navigation algorithm test environment, which is a university based platform for research purposes. It is an extension to a fully functional GNSS receiver developed at Tampere University of Technology (TUT) [1].

Current trend with the GNSS receivers has moved more and more towards the software receiver approach [2][3][4]. The software receiver architectures offer very convenient platform for quick prototyping of new algorithms for the multi-constellation, multi-frequency GNSS field. The TUTGNSS reference receiver uses hardware-accelerated software receiver architecture, where hardware performs the computationally most exhausting tasks: correlations for acquisition and tracking. Software controls all events that take place in the hardware. Typical GNSS Receivers tasks have been illustrated in Fig. 1.

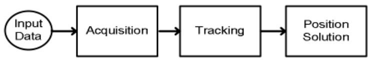


Figure 1. Typical GNSS receiver tasks

In this paper we concentrate on the implementation of a new TUTGNSS Navigation Algorithm Test Environment, which allows the efficient development and evaluation of new navigation algorithms for GNSS receivers. The chosen parallel architecture saves the time of replaying the recorded GNSS signals for each individual navigation algorithm separately. The results can also be seen on-the-fly, while the recorded data is being processed. Additionally we can configure TUTGNSS Reference Receiver to process the data in real-time using real GNSS data. With real data, the amount of parallel navigation solutions running is limited due to the real-time running requirement. Figure 2 shows the TUTGNSS Navigation Algorithm Tests Environment tasks.

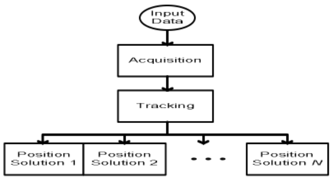


Figure 2. TUTGNSS Navigation Algorithm Test Environment with multiple parallel position solutions

This paper is divided into the following sections.

- Section II gives the introduction to the TUTGNSS reference receiver architecture.
- Section III is dedicated for the detailed description of the new TUTGNSS navigation algorithm test environment.
- Section IV provides examples and results from the test cases we have been developed for the system.
- Section V will be for the conclusions and future work plans.

II. TUTGNSS Reference Receiver

The TUTGNSS Reference Receiver has been implemented at Tampere University of Technology (TUT) [1]. A description of the TUTGNSS Reference Receiver parts is shown in Fig. 3, and a detailed description of each part is given in the following sub chapters.

In addition, TUTGNSS Reference Receiver has been mapped into a full software receiver called C++ TUTGNSS. The full software version of the receiver allows quick prototyping and evaluation platform for the receiver architecture. The C++ TUTGNSS runs on a typical desktop PC and we use thread processing to achieve parallelism in the software. The input data can be either the real data, or recorded data set read from a file [5].

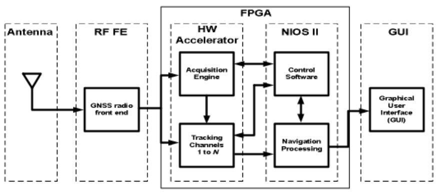


Figure 3. TUTGNSS Reference Receiver Architecture

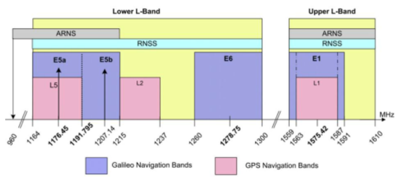


Figure 4. Galileo and GPS Frequency Plan [7].

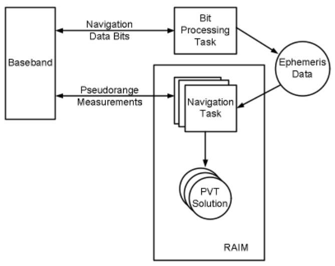


Figure 5. TUTGNSS Navigation Solution Tasks

A. Radio Frequency Front-End

For the Radio Frequency (RF) Front-End (FE), we are using 3rd party hardware. The default RF FE is Atmel ATR0603 GPS L1 RF FE [6]. Other RF FEs can be interfaced to the design with minor modifications. The current RF setup on TUTGNSS only supports single-frequency measurements from E1/L1 bands, Fig. 4. Future goal is to achieve dual-frequency capability on E1/L1 and E5a/L5 bands [8].

B. FPGA

Current implementation platform FPGA (Field Programmable Gate Array) is an Altera Stratix II DSP development board (EP2S180) [9], with Nios II softcore processor [10]. The baseband architecture is implemented in VHDL–language and it supports simultaneous tracking of 16 satellites. At the moment the baseband hardware provides support for open signals at GPS and Galileo E1/L1 and E5a/L5 bands.

C. NIOS II

The NIOS II softcore processor is used for controlling the events happening in the hardware. This includes tasks like choosing the satellites which to acquire and closing of the tracking loops. The PVT solution is also one of the tasks solved on the processor. Navigation solution is divided into 2 distinguishable tasks running independently, as illustrated in Fig. 5. All the written software is in C-language.

- The Bit Processing Task collects navigation data bit flow from each of the tracked satellites, it synchronizes to the frame structure of the particular GNSS system adopted and it checks for possible bit errors. Valid ephemeris data is stored into memory for the use of Navigation task.

- The Navigation Task runs a wider variety of algorithms in order to obtain a reliable navigation solution. The main functionalities require: the pseudorange measurement processing, the satellite location calculation, solving the user location, and checking the reliability of the results. Ultimately the Navigation Task is responsible for producing the Position, Velocity, Time (PVT) solution.

D. *GUI + Debugging Domain*

An important component of every educational tool is the Graphical User Interface (GUI) and the visual impact it has on the final users. In order to get many of the interesting GNSS parameters out from the FPGA, we have selected the standard NMEA format as the output from the Navigation solution. These NMEA messages are the relayed via the serial port (RS-232) to a host computer where the final visualization is made. TUTGNSS currently uses Matlab for this purpose. Additionally the ground track of the GNSS solution can be mapped into format which can be viewed on Google Maps [11].

For debugging purposes a separate debugging domain is implemented on the FPGA board, on the top of a running web server system provided by Altera, while remote users can remotely access for monitoring the TUTGNSS baseband and navigation results in real-time. Hence the main feature of the debugging domain GUI is the remote analysis of log files generated in real-time by the TUTGNSS architecture.

Specifically, the debugging domain implements an HTTP server making use of the sockets interface of the NicheStack TCP/IP Stack-Nios II Edition b, serving with web content from the Nios II board. The server is running on the board and it can process basic requests and serve specified files formats from the Altera read-only file system. It also allows basic remote configuration over Ethernet to handle pre-defined software configuration of the FPGA [12]. Moreover it is able to control different basic baseband outputs directly from the designed web page. Figure 6 shows the system connections and the host set-up. Basically, it requires an Ethernet cable connected to the development boards RJ-45 jack and a JTAG connection with the FPGA board. Through the Ethernet connection the board acquires an IP address, becoming visible for the clients of the network. Results outputs will be visible to all the clients connected to the IP address of the web server.

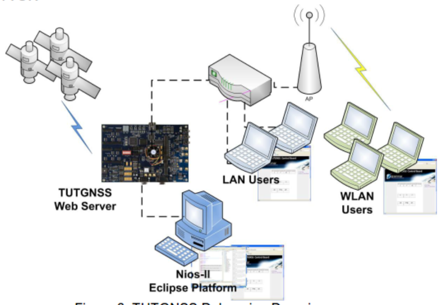


Figure 6. TUTGNSS Debugging Domain

III. TUTGNSS NAVIGATION ALGORITHM TEST ENVIRONMENT

The traditional approach with GNSS receivers is to have one particular navigation algorithm that produces the final PVT solution, like what we have with TUTGNSS Reference Receiver. There are though many well-known methods presented in the literature, like the Least Squares (LS), Kalman Filter, and closed form solutions [13], [14]. With all the variety these different methods offer, and the various ways of implementing them on software, the selection of the "best" navigation algorithm becomes an interesting problem.

It is not straightforward to define the "best" navigation algorithm. One point of view could be that the best navigation algorithm is the one that provides the *best accuracy* to the PVT solution. More practical point of view would be that best algorithm is the one that provides *sufficient accuracy* with least computational effort. In particular we are interested of the computational complexity, memory requirements, and the accuracy of the evaluated algorithm. In order to make this evaluation and comparison into convenient and educative task, we have developed the TUTGNSS navigation algorithm test environment.

One good navigation algorithm is enough for GNSS receivers. From the educational point it is very important to realize the differences of the algorithms. One example of parallelism in the navigation solutions is presented in [15], where traditional navigation algorithm is run in parallel with differential solutions. Another common way is to enable a single custom navigation algorithm to be selected, like in [16]. TUTGNSS navigation algorithm test environment offers a platform where the number of parallel navigation algorithms is only limited by the resources on FPGA board and the real time running capability. As with the TUTGNSS Reference Receiver, we have also implemented the Navigation Algorithm Test Environment as a pure C++ software design. The list of current navigation algorithm modules is seen on Table 1.

Table 1. List of Navigation Modules for TUTGNSS Navigation Algorithm Test Environment

Method	Description	Reference
Fixed-Point Arithmetic	Use custom fixed point library	
Floating Point Arithmetic	Use Nios II supported floating point variables	
RAIM	Receiver Autonomous Integrity Monitoring (RAIM)	[17]
Carrier-smoothed pseudoranges	Carrier smoothing when carrier lock achieved	[18]
Code based pseudoranges	Standard measurements	[18]
Satellite Orbit Approximation	Simple Linear approximation model	[19]
Least Squares	Standard solution	[13]
Weighted Least Squares	C/N0 based weights	[17]
Kalman Filter, P-model	Stationary model	[14]
Kalman Filter, PV-model	Dynamic model	[14]
Sequential Kalman Filter	Kalman Filter without matrix arithmetic library	[14]

On the FPGA platform the available resources for performing all the demanding navigation algorithm computations are more limited. On the other hand it is much easier to predict how much time the processor needs to execute certain tasks, because there is no need to worry about background processes interrupting, like in the standard desktop PC environment. At the moment we have implemented a design where we utilize a single NIOS II softcore processor for performing the simultaneous PVT solutions. Altera Performance Counter Unit has been included to the design to evaluate the time needed for each navigation solution to perform. With the known execution times of the algorithms, the amount of simultaneous PVT solutions can be decided. The more simultaneous PVT solutions, the longer internal PVT solution rate must be selected.

The idea for the TUTGNSS Navigation Algorithm Test Environment originally started from the software platform. Today's desktop PCs have powerful processors capable of executing computationally demanding tasks. Often there is a situation where a lengthy GNSS data set was recorded, and we want to analyze it. The software platform is such an efficient platform that we could analyze the whole data set "faster than real-time". From this point of view it is obvious that there is a benefit in using recorded data sets for testing and evaluating purposes of navigation algorithms. In average, 1 minute of recorded GNSS data can be processed in 25 seconds on the software platform. The execution time is not a constant because of the background processes randomly loading the processor.

Even though the software platform is able to analyze a recorded GNSS data file in a fraction of a time, it is still a lot of time if one needs to replay the scenario multiple times. This led to the idea of implementing software that could perform all the analysis of the PVT solutions in a parallel way.

IV. TEST RESULTS

In order to demonstrate the performance of the TUTGNSS Navigation Algorithm Test Environment we have selected two different test scenarios.

The first scenario will be performed on a static open sky environment, where the TUTGNSS receiver is connected to a roof antenna on the TUT's information building. Since the coordinates of the roof antenna are well known (Latitude 61°27.01, Longitude 23°51.32, Height 182m), we can use those coordinates as reference values to compare our results.

The second scenario will be a dynamic scenario, where the GNSS signal was recorded with a patch antenna attached to the roof of a car. A short measurement route was driven in Hervanta suburban of Tampere, Finland. A traffic circle was included to the route. The problem with the dynamic GNSS test is that we don't have an exact reference solution available. This problem was solved by utilizing Spirent STR4500 GPS/SBAS Simulator System [20]. The procedure for dynamic GNSS scenario was as follows:

- Record the GNSS data with a patch antenna attached to the roof of a car.
- Give the recorded GNSS data as input for Spirent simulator.
- Let Spirent generate the constellation based on the given GNSS data.
- Use Spirent as the GNSS signal source with the newly generated scenario.

With the aforementioned procedure we now know the exact route, based on which Spirent generated the scenario. The results for the dynamic scenario can now be compared to exact reference coordinates. The drawback of the Spirent method is that the generated scenario does not match the original situation. Most obvious difference is the lack of signal blocking and multipaths due to the buildings and environment.

In this test implementation we use our custom fixed-point library; simple RAIM scheme is used; no satellite orbit approximation. The following test results have been achieving by loading the following 3 simultaneous navigation algorithms to the TUGNSS Navigation Algorithm Test Environment:

- Code based pseudoranges + Least Squares
- Carrier-smoothed pseudoranges + Least Squares
- Carrier-smoothed pseudoranges + PV-Kalman Filter

Altera Performance Counter Report for the Navigation Tasks running on Nios II is shown in Table 2.

Table 2. Altera Performance Counter report For Navigation Solutions

Total time: 0.827888 seconds (82788790 clock-cycles)				
Section	%	Time (sec)	Time (clocks)	Occurrences
Satellite_loc	46.5	0.38478	38478290	1
Least_squares	15.0	0.12389	12388550	1
Carrier_LS es	14.7	0.12139	12138670	1
Kalman_filter	23.9	0.19783	19783280	1

A. Static Roof Antenna Test

The test results for static test are illustrated in Fig. 7 The average errors for LS, carrier smoothed LS and Kalman filter methods are 6.7m, 5.0m and 4.7m respectively.

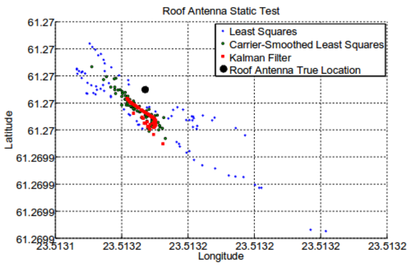


Figure 7. Results for Static Roof Antenna Test

B. Dynamic Test

The test results for dynamic test are illustrated in Fig. 8 and 9. Unfortunately the used GUI does not support yet the color separation of the paths plotted to Google Maps. In our test the carrier smoothed LS proved to be the most accurate (Fig. 9) in a long run and it also was one with the least computational burden (Table 2.)

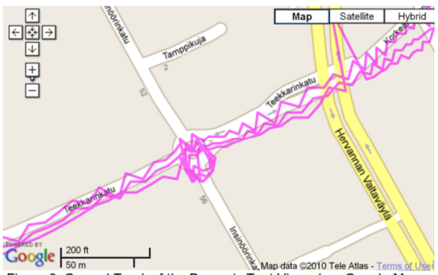


Figure 8. Ground Track of the Dynamic Test Viewed on Google Maps.

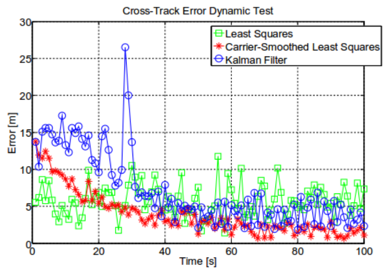


Figure 9. Cross-track Error for the Dynamic Test

V. Conclusions and Future Work

In this paper we have presented the novel architecture of the TUTGNSS Navigation Algorithm Test Environment, which allows evaluation of multiple parallel PVT solutions on-the-fly. We have described the architecture on both FPGA platform, and full C++ software designs. The paper presents examples which give the proof of concept. Few select Navigation algorithms have been evaluated with the presented architecture and of those the carrier smoothed LS method proved to have both the best performance and least computational load. In addition we have described the architecture of a debug domain for our GNSS receiver. The motivation and benefits of the debugging domain have been presented.

The future work will consist of finishing dual frequency capability to the TUTGNSS Reference Receiver. Dual frequency capability major benefit would be getting pseudorange measurements without ionosphere delay error component. Another topic for future work is to implement the TUTGNSS Navigation Algorithm Test Environment as a true multicore application on the FPGA board. The additional NIOS II softcores processors will take resources from the board, but on the other hand dedicated navigation processors have more time to perform the given computational tasks. We believe that TUTGNSS Navigation Algorithm Test Environment will serve as a significant platform for testing of future receiver algorithms without dependency on third party black boxes or limited configurability.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under the GRAMMAR project, grant agreement n◦ 227890.

References

[1] H. Hurskainen, T. Paakki, Z. Liu, J. Raasakka, and J. Nurmi, "GNSS receiver reference design," in Proc. of the ASMS 2008 conference, Aug. 26-28, 2008, Bologna, Italy.

[2] M. G. Petovello, C. O'Driscoll, G. Lachapelle, D. Borio, and H. Murtaza, "Architecture and benefits of an advanced GNSS software receiver," Journal of Global Positioning Systems (2008), Vol. 7, No. 2: 156-168.

[3] S. Söderholm, T. Jokitalo, K. Kaisti, H. Kuusniemi, and H. Naukkarinen, "Smart positioning with Fastrax's software GPS receiver solution," in Proc. of ION GNSS 2008, Savannah, Georgia, Sep. 16-19, 2008.

[4] K. Borre, D. Akos, N. Bertelsen, P. Rinder, and S. Jensen, "A software defined GPS and Galileo receiver: A single-frequency approach," Boston, Birkhäuser, 2007, ISBN 978-0-8176-4390-4.

[5] J. Raasakka, H. Hurskainen, T. Paakki, and J. Nurmi. "Modeling Multi-Core Software GNSS Receiver with Real Time SW Receiver" in Proc. of ION GNSS 2009. Savannah, Georgia.

[6] Atmel, "ATR0603 – GPS Front End IC," Datasheet. Rev 3.5, Nov. 2006.

[7] "Galileo open service, signal in space interface control document (OS SIS ICD)," Feb. 2008, Draft 1.

[8] H. Hurskainen, E.-S. Lohan, J. Nurmi, S. Sand, C. Mensing, and M. Detratti. "Optimal Dual Frequency Combination for Galileo Mass Market Receiver Baseband," in Proc. of the IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS). Tampere, Finland, October 7–-9, 2009.

[9] Altera Stratix II URL: http://www.altera.com/literature/manual /mnl_stx2_pro_dsp_dev_kit_ep2s180.pdf

[10] Altera Nios II Processor URL: http://www.altera.com/products/ip /processors /nios2 /ni2-index.html

[11] Google Maps, URL: http://maps.google.com/

[12] Altera Web Server URL: http://www.altera.com/support/examples /nios2/exm-micro_tutorial.html

[13] E. D. Kaplan, and C. J. Hegarty, "Understanding GPS – Principles and applications," 2nd ed., Norwood, Artech House, 2006, ISBN-10:1-58053-894-0.

[14] R.G. Brown, and P.Y.C Hwang, "Introduction to random signals and applied kalman filtering," 3rd ed., John Wiley & Sons, 1997, ISBN 0-471-12839-2.

[15] T. Lück, J. Winkel, M. Bodenbach, E. Göhler, N. Falk, A. Consoli, F. Piazza, D. Gerna, R. Granger, P. Readman, S. Simpson, H-J. Euler, "Artus – A second generation Galileo/GPS receiver," in Proc of ION GNSS 2007, September 25-28, 2007, Fort Worth, TX.

[16] F. Dominici, P. Mulassano, D. Margaria, and K. Charqane, "SAT-SURF and SAT-SURFER: Novel hardware and software platform for research and education on satellite navigation," in Proc. of ENC-GNSS 2009, Naples, Italy, May 3-6, 2009.

[17] E. Zemskov, and J. Nurmi, "Performance enhancements for embedded software implementation of GNSS navigation algorithmsm," in Proc. of the Industrial Embedded Systems, 2006. IES '06, Oct. 18-20, 2006.

[18] P. Misra, and P. Enge, "Global Positioning System: Signals, measurements, and performance," Lincoln, Ganga-Jamuna Press, 2004, ISBN 0-9708544-0-9.

[19] P. Korvenoja, and R. Piché, "Efficient satellite orbit approximation," in Proc. of ION GPS 2000, Salt Lake City, USA, Sept. 19-22, 2000.

[20] Spirent STR4500 Documentation, URL: http://www.spirent.com/

PUBLICATION 5

T. Paakki, and J. Nurmi, “Faster Than Real-Time GNSS Receiver Testing”, in Proceedings of the *International Conference on Localization and GNSS (ICL-GNSS 2014)*, June 2014, In Helsinki, Finland.

Faster Than Real-Time GNSS Receiver Testing

Tommi Paakki, and Jari Nurmi
Department of Electronics and Communications Engineering
Tampere University of Technology
Tampere, Finland
{firstname.lastname@tut.fi}

Abstract— **The new GNSS systems available present possibilities and challenges in the design of modern GNSS receivers. As a part of this challenge, receiver testing is always very important aspect. Careful testing procedures tend to be slow and tedious. In this paper we present a method for speeding up the test process without limiting the testing capabilities. The proposed method is only applicable to testing performed on recorded GNSS data. In this method the recorded data is streamed as fast as possible, without being bound by normal GNSS data rates. Utilizing this kind of architecture on a GNSS receiver can cut the time spent in the testing process by 70% or more, depending on the computational performance of the GNSS receiver platform. Speeding up the test phase of a GNSS receiver is beneficial for both academic and commercial entities, allowing time to be spent on other tasks.**

Keywords - Satellite navigation systems, Receivers

I. INTRODUCTION

Nowadays many new GNSS signals are available for the user equipment. With the introduction of these new signals, the GNSS receivers need to be capable of processing these signals in order to gain full benefit from them. In addition to new GNSS signals, also other signal processing techniques advance, and it takes further time to develop these new algorithms. In addition, advanced GNSS receivers are expected to operate more and more challenging environments. A lot of time is spent on testing and verifying the new architectures and algorithms. For example in [1], a standard collection of testing procedures for a GNSS receiver is described. Having to test a GNSS receiver this thoroughly will consume a lot of time, and it would be beneficial to find ways to reduce the time needed for the research / verifying cycle for new products. In this paper we concentrate on the GNSS testing, and how to reduce the time spent on testing.

Traditionally GNSS testing has been divided into following 3 distinct methods.

- Live sky testing
- Record and Replay methods
- Simulators

A. Live Sky Testing

While live sky testing is the ultimate method for finding rare scenarios which would be very difficult to simulate, for example complex multipath situations, its usefulness is severely limited by the lack of repeatability.

B. Record and Replay Methods

Record and Replay is very attractive option for GNSS receiver testing. With this method it is possible to find the complex scenarios from the live sky, record them and Replay them at-will. With the replay option it is possible to adjust receiver hardware and see how it affects the performance on the complex scenario. This method has become increasingly popular, and equipment now exists for easy data recording capability [2, 3].

C. Simulators

These days the GNSS signal simulators have also developed a lot and they can provide all the available, and future, GNSS signals. For example in Galileo case the simulators can already simulate the whole Galileo constellation, which does not yet exist in live sky. GNSS simulators provide excellent replay ability, and the scenarios can be extensively modified towards what the GNSS receiver tester wants to test. Using simulators also reduces the test time, since no time is needed for going on the "field" from the simulator premises.

In this paper we introduce a novel method for reducing the time spent on testing GNSS receivers. The proposed method does not require any additional expensive equipment. This method is named Faster Than Real-Time GNSS Receiver Testing, and it is a method where the architecture of a GNSS receiver is operating purely on the samples arriving from the Radio Frequency Front End (RF FE). In other words, time is determined by the incoming samples, not by the internal oscillator clock as usual. These samples can be fed to the receiver faster than real time GNSS data rates would normally allow.

This paper is organized as follows: First, section II gives description of the hardware / software architecture of GNSS receiver suitable for the presented idea. Section III introduces such hardware / software platforms where the proposed architecture has been implemented. Section IV provides obtained results with the presented method. Finally, Section V collects the conclusion and future work regarding the topic.

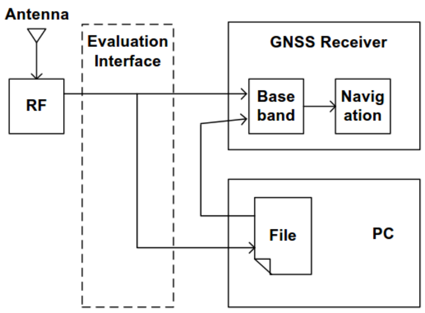


Figure 1. Evaluation Interface and Possible Operating Modes

II. Proposed Architecture

A. Hardware Architecture

A suitable architecture for the proposed method closely follows the traditional design of a GNSS receiver [4]. The traditional blocks are split from RF FE interface, where a specific evaluation interface is defined for the possibility of recording the raw I/Q samples. Figure 1 shows the location of the evaluation interface with respect to the rest of the receiver. With the evaluation interface it is possible to record the incoming I/Q samples to a standard PC via suitable interface, for example USB and Ethernet connection.

In this setup, the RF FE impairments are captured in the recorded data set [5, 6], and when playing back the data, the receiver resulting outputs are same as with the normal operating mode. In other words, the system is transparent to whether it is operated normally, or in Record and Replay mode. With the introduction of the evaluation interface, four operating modes are available:

- Standard Operating Mode. In the Standard mode, the receiver operates just as an ordinary receiver. The evaluation interface is not active on this mode, and should not interfere in any way with the rest of the receiver.
- Data Record Mode. In data record mode the receiver only processes the samples arriving from the RF FE. The samples are buffered and sent to a standard PC for storing and later processing. A suitable interface for handling the high data transmission rate between the receiver and PC could be for example USB or Ethernet standards.
- Data Playback Mode. The receiver now operates on the recorded data samples which are being sent to the receiver from the PC. The receiver is set up to operate solely on the arriving sample rate, which indeed can exceed the real-time data rate limitations.
- Standard Mode with Data Recording Enabled. This mode can be set up if desired. The receiver would then operate normally on the RF FE inputs, while the received I/Q samples are simultaneously being recorded to the PC. Again, care should be taken, so that the data recording process is not interfering in any way with the rest of the receiver.

Finally, the receiver itself must be configured to operate on the arriving sample rate from either RF FE, or the data file.

B. Software Architecure

In order to achieve even greater speedup benefit from the Record and Replay mode, a software implementation of the receiver is introduced. The software receiver, as the name suggests, is software version of the hardware receiver, written for a standard PC. It should match the original hardware receiver as closely as possible, in order to get comparable outputs from both receivers.

Having the software receiver match the hardware version has two distinct design challenges. First one is to cope with the inherent parallel operation of a hardware GNSS receiver. One possible solution is to create software with multiple threads to emulate this parallelism. The other challenge then is to verify carefully all the hardware / software source codes to have one to one match with each other. Some hardware receivers may utilize Central Processing Units (CPU) with limited floating point precision or no floating point support at all, which the software receiver has to adapt to. A collection of various CPUs utilized in GNSS receivers, and their support of floating points, are presented in [7].

The main reason to create a software receiver replica is the freedom to use the recorded data directly from the PCs hard drive, without the need of sending it further via a slower interface. Other clear benefit of software receiver is the computational power available with the PC CPUs, in comparison to the regular, lower performance processors used in the GNSS receivers. On the other hand, having two parallel receivers, the hardware and software, increases the challenges in version control to keep both designs updated.

III. Implementation Platform

A. TUTGNSS

The TUTGNSS Reference Receiver is a fully operational GPS / Galileo receiver, developed at Tampere University of Technology for educational purposes [8]. It provides a platform for testing and developing new algorithms for GNSS field. TUTGNSS contains no “black boxes”, allowing developers to have full control over its further development. The TUTGNSS Reference Receiver uses hardware accelerated software receiver architecture, where hardware logic performs correlations for acquisition and tracking, and software controls all events that take place in the hardware. Figure 2 shows the high level architecture of TUTGNSS Reference Receiver in the Faster Than Real-Time GNSS Receiver Testing configuration.

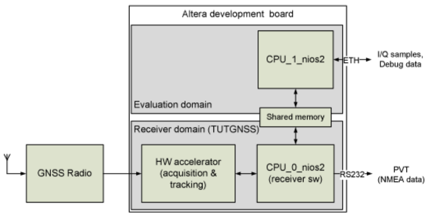


Figure 2. Evaluation Interface of TUTGNSS.

Current Field Programmable Gate Array (FPGA) for TUTGNSS is an Altera Stratix II DSP development board (EP2S180) [9]. The hardware parts of the design are written in VHDL hardware description language. The software processes are executed on NIOS II softcore processor implemented within the FPGA. Software processes are handled and driven by control process which implements interrupt service routines according to priorities. The source codes for software have been implemented in C-language. Several possible Radio Front Ends have been configured to work with TUTGNSS, including commercial and custom ones.

As shown in Figure 2, a 10/100 Mbps Ethernet port is used for the transferring the Radio Front End data between the FPGA. The evaluation domain, including the Ethernet connectivity, is kept separated from the standard TUTGNSS by making it as an additional module running under another dedicated NIOS II softcore. The modularity of TUTGNSS allows easy configuration of operating mode. A high level description of TUTGNSS modules are shown in Figure 3. Most importantly, the Baseband Converter Unit (BCU), is always updated to meet the parameters of used Radio Front End. The BCU has different modes implemented, allowing the I/Q samples to arrive from different sources.

B. *Software TUTGNSS*

The software TUTGNSS was developed for enabling quicker algorithm verification cycles. The software modules can be modified within minutes, whereas every hardware modification including the necessary compilations, would take nearly an hour. From this perspective it is beneficial to have a software platform.

The software version must be as similar as possible with the hardware version. To emulate the concurrent operations of the hardware, thread programming was selected for the software. The different threads of the software use events for maintaining synchronization. The main threads of the software TUTGNSS are listed in Table 1.

Most importantly, the Input Preprosessing thread is the one responsible for adapting to different sources of the I/Q data.

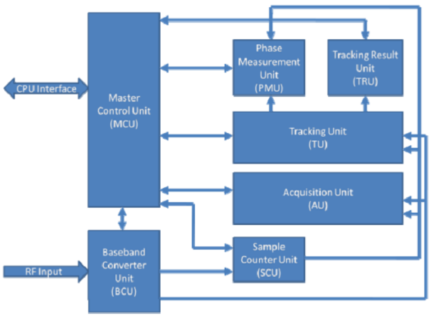

CPU Interface
Master Control Unit (MCU)
Phase Measurement Unit (PMU)
Tracking Result Unit (TRU)
Tracking Unit (TU)
Acquisition Unit (AU)
Sample Counter Unit (SCU)
Baseband Converter Unit (BCU)
RF Input

Figure 3. Hardware Modules of TUTGNSS.

Table 1. Software Threads.

SW TUTGNSS Threads
Input Preprocessing
Acquisiton
Tracking (Multiple threads)
Nav Data Decoding
PVT

IV. Results

The presented Faster Than Real-Time GNSS Receiver Testing method has been verified with both simulator and real GNSS data. The results are presented with the following real data setup:

The used commercially available RF FE was SiGe's SE4120L [10]. SE4120L comes with an evaluation board SE4120L-EK3, which has available USB 2.0 interface for easy connection to a PC. Table 2 presents the configuration parameters for the Radio Front End.

In order to get good average of the performance, a long record of data was selected for further analysis. This data set consists of 15 min of recorded data. With the reported sampling frequency from Table 2, this results in a 3.5 GB File size. In the recorded scenario a steady amount of 6-8 satellites were visible. The scenario also includes a location with strong multipath reflection, making it interesting recording for further analysis.

Table 2. Result Scenario Radio Front End Parameters

Parameter	Value
Sampling Frequency	4.092 MHz
Intermediate Frequency	0 Hz
Data Width	3 bits
Complexity	Both I/Q Samples

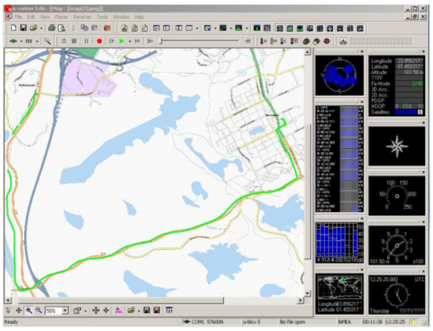


Figure 4. 15 min Real Scenario run in 4 min 20 sec.

The scenario was run under Windows 7, 64-bit machine. Intel core 2 duo CPU, E6850, 3GHz. Available RAM 4 GB. Scenario results were plotted on U-Blox U-Center software via a RS-232 connector [11].

One realization of the test rounds is plotted in Figure 4. The average time for 10 repetitions clocked 4 min 20 sec. For 15 min of real time, resulting in a reduction of 70% time spent into running the scenario.

V. CONCLUSIONS AND FUTURE WORK

In this paper a new architecture suitable for Record and Replay GNSS testing was presented. Both hardware and software architectures were introduced, together with suggestions for suitable evaluation interface. The proposed method was shown to be able to cut time spent on running a typical GNSS scenario by 70%.

Since the obtained results are very encouraging, the future work includes refining the architecture of TUTGNSS to better integrate with the Faster Than Real-Time GNSS Testing method. The future work also includes further improving of the similarity between the hardware and software TUTGNSS receivers. The proposed method also needs to be evaluated with various available Radio Front Ends, and with different sampling frequencies, since it has significant impact on the obtained speedup performance.

REFERENCES

[1] A. Mitelman, P-L. Normark, M. Reidevall, and S. Strickland, "Apples to Apples: A Standardized Testing Methodology For High Sensitivity GNSS Receivers," in Proc. of the ION GNSS 2007 conference, Sept. 25-28, 2007, Fort Worth, Texas.

[2] S. Hickling, and T.Haddrell, "Recording and Replay of GNSS RF Signals for Multiple Constellations and Frequency Bands," in Proc. of the ION GNSS 2013 conference, Sept- 16-20, 2013, Nashville, Tennessee.

[3] S. Backén, D. Akos, and S. Wilson, "RF Replay System for Narrowband GNSS IF Signals," IEEE Transactions on Aerospace and Electronic Systems, Vol. 47, NO. 1, Jan. 2011.

[4] M. S. Braasch and A. J. Van Dierendonck, "GPS receiver architectures and measurements," Proceedings of the IEEE, vol. 87, no. 1, pp. 48-64, 1999

[5] I. Ilie, R. Hini, J-S. Cardinal, P. Blood, and P. France, "Record and Playback System for GNSS: Real Performances for Real Applications,“ Averna White Paper, Available, http://www.insidegnss.com.

[6] M-A. Fortin, I. Ilie, D. Fortin, K. Mollaiyan, and R. Landry, "Wideband Dual-Channel RF Record and Playback for Multi-Constellation Analysis," in Proc. of the ION GNSS 2010 conference, Sept. 21-24, 2010, Portland, Oregon.

[7] S. De Florio, E. Gill, S. D'Amino, and A. Grillenberger, "Performance Comparison of Microprocessors for Space-Based Navigation Applications," in Proc of 7th IAA Symposium on Small Satellites for Earth Observation, May 2009.

[8] T. Paakki, J. Raasakka, F. Della Rosa, H. Hurskainen, and J. Nurmi, "TUTGNSS – University Based Hardware/Software GNSS Receiver for research Purposes," in Proc. of the UPINLBS 2010 conference, Oct. 14-15, 2010, Kirkkonummi, Finland.

[9] Altera Stratix II, URL: http://www.altera.com/

[10] SiGe, "SE4120L – GNSS Receiver IC," Datasheet. Rev 3.5, May 2009

[11] "u-center GNSS evaluation software," URL: http://www.u-blox.com

PUBLICATION 6

T. Paakki, F. Della Rosa, and J. Nurmi, “Stand-Alone GNSS Time Synchronization Architecture”, in Proceedings of the *European Workshop on GNSS Signals and Signal Processing (NAVITEC 2014),* December 2014, In Noordwijk, The Netherlands.

Stand-Alone GNSS Time Synchronization Architecture

Tommi Paakki, Francescantonio Della Rosa, and Jari Nurmi
Department of Electronics and Communications Engineering
Tampere University of Technology
Tampere, Finland
{firstname.lastname@tut.fi}

Abstract— **It is well known property of the GNSS systems that they are able to deliver very precise Position, Velocity and Time information to the user. In this paper we introduce a GNSS receiver architecture, which is able to synchronize its operations to the GNSS signals it processes, be it live satellite signals or simulator signals, without the need of external synchronization equipment. This capability is useful for testing the receiver with GNSS simulators, and can also be used for synchronizing a set of GNSS receivers. Since the architecture does not need any external synchronization signals, it is convenient to move the GNSS receiver between testing premises. Most commercially available GNSS simulators can output some format of scenario information, and this scenario information is by default bound to be ticked on exact full second intervals according to the simulator clock. Having a receiver capable of doing the synchronization to the same time allows the receiver output its own measurements at the same time instants, making the comparisons extremely useful.**

Keywords - Satellite navigation systems, Receivers

I. INTRODUCTION

GNSS is an excellent system for offering positioning and timing services to the users. The time transfer property has been researched from the 1980s [1]. From the earlier methods to more resent research [2, 3] there has been a general trend to utilize external devices for accurate synchronization. In this paper we present a stand-alone synchronization method for GNSS receivers.

GNSS receiver testing is very important part of the product verification cycle. A lot of the verification and testing work is done with the help of GNSS signal simulators. These advanced signal simulators need to be calibrated on a yearly basis to maintain the desired frequency and power stability. In some situation, a simulator Local Oscillator (LO) may have drifted significantly since last maintenance calibration, making the receiver testing complicated. In particular, the simulators tend to pack navigation observables into debug files, which are stamped on exact full seconds according to their internal clocks. This paper introduces a method for the receiver to synchronize with the time provided by the simulator - and to make the receiver measurements match with the time instants when the simulator outputs its debug information. This makes the simulator debug files and the receiver measurements directly comparable. Naturally this method is applicable to real live sky satellites also. Using the proposed method is simple alternative for interpolating/predicting the simulator outputs, so that they would match the receiver measurement time instants.

Additional application of the presented idea is to solve the receiver-level time synchronization problems for cooperative positioning between GNSS receivers [4]. In the suggested cooperative positioning schemes the GNSS receivers can share measurements and information between each other. Some measurements, like pseudorange measurements, are very sensitive to timing, making it necessary to have a reliable method of ensuring that the receivers are well synchronized.

Following the standard GNSS receiver structure, the receiver utilizes its LO signal to maintain the frequency of signal processing operations of the Radio Frequency Front End (RF FE), and on the rest of the receiver structure [5]. This LO often has a flaws of its own, since most commercially available oscillators on consumer grade equipment are not extremely stable. In this paper we present a method of utilizing standard GNSS receiver operations to synchronize with the incoming signal time. This leads to a receiver architecture allowing synchronization of measurements to match those exact full seconds of the simulator being used. The novelty of the method comes from the receiver self-enabling operation, without the need of any external equipment, like cables and other regularly used signaling sources for standard 1 Pulse Per Second (1pps).

This paper is organized as follows: Chapter IÍ introduces the platform where the presented method has been verified. In chapter III we present details of the proposed hardware architecture. Chapter IV presents the algorithms developed. Chapter V shows the obtained results. Finally, Chapter VI collects the conclusion from this work.

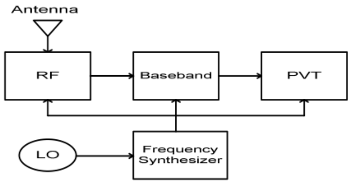


Figure 1. Standard GNSS Receiver Structure

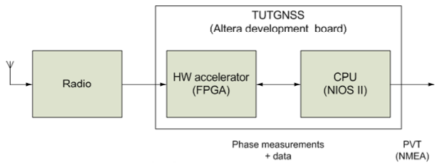


Figure 2 Architecture of TUTGNSS Receiver

II. Implementation Platform

The implemented algorithms have been realized on TUTGNSS Reference Receiver [6], which has been developed at Tampere University of Technology (TUT) for educational purposes. TUTGNSS is a hardware/software platform for GNSS receiver algorithm testing and it contains no "black boxes", allowing users learn / further develop any part of the receiver.

The TUTGNSS Reference Receiver uses hardware-accelerated software receiver architecture, where hardware logic performs correlations for acquisition and tracking, and software controls all events that take place in the hardware. Figure 2 shows the high level architecture of TUTGNSS Reference Receiver. Current Field Programmable Gate Array (FPGA) used for development of TUTGNSS is an Altera Stratix II DSP development board (EP2S180) [7], but in principle the design should be platform independent. The hardware parts are implemented in VHDL hardware description language. The software processes are executed on NIOS II softcore processor implemented within the FPGA. Software processes are handled and driven by control process which implements interrupt service routines according to priorities. The source codes for software have been written in C-language.

III. Proposed Architecture

In the proposed GNSS architecture time is measured in units of samples from the RF FE. This local sample time reference is implemented in hardware as a special module Sample Counter Unit, which will be described in detail in this chapter. In other words, the timekeeping is not directly based on the LO frequency, but rather on the amount of samples arriving from the RF FE with specified sample rate.

There is also another timekeeping term, receiver time t_u, which is implemented as a software counter within the design. This is the time term, which will be synchronized to the GNSS time.

The implemented hardware modules of the design are shown in Figure 3. Next we describe the most important modules concerning the presented method.

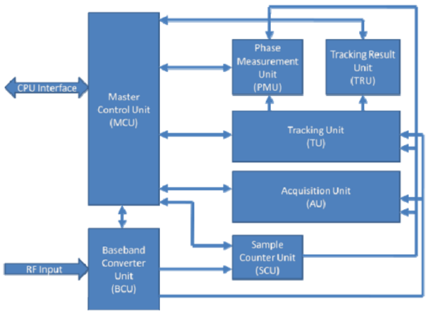

CPU Interface
Master Control Unit (MCU)
Phase Measurement Unit (PMU)
Tracking Result Unit (TRU)
Tracking Unit (TU)
Acquisition Unit (AU)
Sample Counter Unit (SCU)
Baseband Converter Unit (BCU)
RF Input

Figure 3. Hardware Modules of TUTGNSS.

A. Baseband Converter Unit (BCU)

Baseband converter unit changes any type of RF FE input complexity to common three bit I&Q signed presentation. Also its job is to remove the Intermediate Frequency (IF) so that acquisition and tracking units need only to account the Doppler frequency of the incoming satellite signal. The BCU also contains optional decimation for input, separately for both acquisition and tracking. BCU is always updated to meet the parameters of used radio.

B. Master Control Unit (MCU)

Master control unit handles all information passing between CPU and the GNSS HW accelerator. From the SW programming point of view it has only one control register which is reserved for the moment. Control register may be used in the later versions to control transfers between CPU and the GNSS HW accelerator. Idea for having a MCU is that this way we may change transfer protocol without having to change any other unit in the system.

C. Sample Counter Unit (SCU)

Sample counter unit measures time by counting the samples coming from the RF FE. This is the hardware unit which is our local time reference that we need to use to calculate user Position, Velocity, and Time (PVT) information. SCU is also used to synchronize all units together, mainly transformation between acquisition and tracking stages. SCU takes its input from the BCU and updates its counter whenever new sample arrives. When software knows the incoming sample rate this information can be used as very high precision clock, which accuracy is depended on the RF FE oscillator. In the proposed architecture, the software regularly communicates with the SCU to adjust the sample rate so that oscillator offsets are compensated.

D. Phase Measurement Unit (PMU)

Phase measurement unit measures the phases of the tracking channel NCO's at software specified intervals. This phase measuring is related to local time information received from the SCU. In the context of this paper, the local time information from SCU is being synchronized so that the PMU will produce its measurements on the exact full second boundaries. These phases are measured from every tracking channel simultaneously so software can derive pseudorange information for each satellite with the aid of demodulated data from the satellites. PMU stores information about code and carrier phases so advanced methods on pseudorange calculation are also possible.

IV. Developed Algorithms

The core idea is to adjust the PMU interval so that it makes its measurements on the exact same full seconds as the simulator outputs its data to debug files. In the following, the essential steps for the algorithm are presented.

A. PMU Initialization

At the start of the receiver operations, the PMU is configured with the desired measurement update rate. In TUTGNSS the update rate is given as an interval of samples arriving from the RF FE. The update rate is freely configurable, and default update rate is 1 Hz. When doing the first initialization of the PMU, the receiver has not yet synchronized to the GNSS time, and thus PMU measurements are generated with 1 Hz rate, which is not bound to the full second boundaries

B. Receiver Clock Initialization

Aside from the SCU local time reference, a software receiver clock is implemented as a counter, which will be synchronized to the GNSS time. When receiver is turned on, there is no knowledge of GNSS time, and thus synchronization to this time is needed.

Once enough satellites are being tracked, a good quality satellite is selected for being a reference satellite from which the coarse GNSS time is then obtained. The reference satellite code measurements gives the time when the signal was sent from the satellite, so the receiver is initialized with that time + 70 ms. 70 ms is an approximate time the signal takes to travel from satellite to the receiver on Earth [8]. This process should yield accuracy of several milliseconds from the true GNSS time.

C. Obtaining Accurate GNSS Time

With the initial approximate GNSS time it is possible to calculate the first PVT fix. The first fix usually has quite big time bias, since we have utilized the coarse GNSS time at this point. At this point the bias term can be compensated to the software receiver clock, since it is a simple addition operation to the software clock counter. After applying the initial bias component to the receiver time, the receiver has essentially synchronized to the GNSS time.

D. Synchronization to the 1 Second Boundary

At this step, our accurate GNSS time is used to match the next PMU measurement interval to a full second boundary, on which the simulator will output its measurements. This is achieved by a simple mathematical operation.

$$fix = 1 - \mathrm{fmod}(t_u, 1) \tag{1}$$

$$next\_update = \mathrm{round}(Fs * fix) \tag{2}$$

, where
fmod() is the floating point modulus operation,
round() is the rounding operation,
Fs = sample rate of the RF FE.

This small manipulation sets the next PMU interval on the next exact full second of the GNSS time. The continuing measurements are then set to happen at the original rate of 1 Hz.

E. Compensating Local Oscillator offsets

The last remaining step allows even better fine tuning to the exact full second boundaries. From the output of the regular 8-state Position, Velocity (PV) Kalman filter, we can obtain the clock drift error $\dot{b}$, which can be used for fine tuning the update intervals [9]. By multiplying the clock drift error with the sampling frequency, we get an approximation how many samples per second our LO is offset from the simulator. This number needs to be rounded toward nearest integer, since TUTGNSS hardware can set the PMU intervals for integer samples only. This part can be issued always when the LO drifts 1 sample or more. The compensation shown as a function:

$$next\_update = \mathrm{round}(Fs * \dot{b}) \tag{3}$$

V. Results

The proposed algorithm has been tested and verified using TUTGNSS with custom RF FE based on MAX2769 chip [10]. The sample rate has been set to 26 MHz. The simulator used is a Spirent STR4500 GPS/SBAS simulator [11]. Spirent simulator was configured to leave ionosphere and troposphere effects unmodeled to allow easy comparison of pseudorange measurements between TUGNSS and Spirent. In Figure 4, the pseudorange differences between TUTGNSS and Spirent have been plotted after successful synchronization to the Spirent GNSS time. A stable, successful synchronization is achieved within 5 first PVT solutions, i.e. within 5 seconds from first PVT solution. Any remaining difference in Figure 4 is due to remaining time mismatch and TUTGNSS hardware related imperfections. The time difference starts to drift over time, so resynchronization is needed every now and then. Resynchronization is achieved by repeating algorithm step E.

Given the sample rate of 26 MHz, one sample corresponds to length of:

$$c / Fs = \frac{299792448\,\mathrm{m/s}}{26\mathrm{e}6\ \mathrm{Hz}} \approx 11.5m$$

The sampling frequency sets a practical limitation for the performance of the algorithm.

VI. Conclusions

The presented algorithm is shown to be a suitable method for easy synchronization with GNSS signal simulators. Its main motivation is the lack of need for any external cables or synchronization signals. In addition, no interpolation/prediction is needed for the simulator debug files to make them match the receiver measurements. The algorithm is recommended when operating with GNSS simulators which have not been calibrated recently or the last calibration date is unknown. The presented algorithm can be used with real GNSS satellite signals as well.

The receiver hardware needs to be flexible enough to allow the internal adjustment of the measurement instants during the receiver being in operation. A synchronized receiver is able to compare the simulator outputs to its own measurements at an accuracy mostly limited by the RF FE sample rate.

References

[1] D. W. Allan, and M. A. Weiss, “Accurate Time and Frequency Transfer during Common-View of a GPS Satellite,” in Proc. of the 34th Annual Symposium on Frequency Control, 1980, pp 334-346.

[2] P. Defraigne, P. Banerjee, and W. Lewandowski, “Time Transfer through GPS,” Indian Journal of Radio & Space Physics, Vol. 36, August 2007, pp. 303-312

[3] U. Gruenert, S. Thoelert, H. Denks, and J. Furthner, “Using of Spirent GPS/Galileo HW Simulator for Timing Receiver Calibration,” in Proc. of the Position, Location and Navigation Symposium, 2008 IEEE/ION, May 5-8, 2008.

[4] F. Dovis, C-F. Chiasserini, L. Musumeci, and C. Borgiattino, “Context-aware Peer-to-Peer and Cooperative Positioning,” In Proc. of the ICL-GNSS 2014 conference, June 24-26, 2014, Helsinki, Finland.

[5] M. S. Braasch and A. J. Van Dierendonck, “GPS receiver architectures and measurements,” Proceedings of the IEEE, vol. 87, no. 1, pp. 48-64, 1999.

[6] T. Paakki, J. Raasakka, F. Della Rosa, H. Hurskainen, and J. Nurmi, ”TUTGNSS – University Based Hardware/Software GNSS Receiver for research Purposes,” in Proc. of the UPINLBS 2010 conference, Oct. 14-15, 2010, Kirkkonummi, Finland.

[7] Altera Stratix II, URL: http://www.altera.com/

[8] J. Syrjärinne, “Time Recovery through Fusion of Inaccurate Network Timing Assistance with GPS Measurements,” in Proc. 3rd Int. Conference on Information Fusion, Paris, France, July 10-13, 2000.

[9] R.G. Brown, and P.Y.C Hwang, “Introduction to Random Signals and Applied Kalman Filtering,” 3rd ed., John Wiley & Sons, 1997, ISBN 0-471-12839-2.

[10] Maxim 2769, URL: http://www.maxim-ic.com/

[11] Spirent STR4500, URL: http://www.spirent.com/

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-3987-9
ISSN 1459-2045