IMPLEMENTATION OF REAL-TIME SOFTWARE RECEIVER FOR GPS OR GLONASS L1 SIGNALS

Senlin Peng

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

Wayne A. Scales, Chair

Jeffrey H. Reed

Scott Bailey

Jan 22, 2010

Blacksburg, Virginia

Keywords: GPS, GLONASS, Software Receiver

IMPLEMENTATION OF REAL-TIME SOFTWARE RECEIVER FOR GPS OR GLONASS L1 SIGNALS

Senlin Peng

(ABSTRACT)

A 12 channel real-time GPS L1 C/A-code software receiver has been implemented on a Desktop with 1.84GHz Intel CPU. The software receiver has the capability to acquire new satellites coming in, keep tracking of satellites in view and give a user solution accuracy of 30 meters. This study also explores a real-time correlator for the GLONASS L1 signals. This software receiver is going to be used for scientific research and education. This work is a part of the ongoing effort to develop a low-cost, flexible, and capable GNSS receiver for use as a scientific instrument and for GNSS receiver technology development.

The software receiver developed here makes use of a reconfigurable RF front end called the Universal Software Radio Peripheral (USRP) with a maximum real sampling frequency of 8MHz of complex samples. The USRP uses interchangeable daughter boards to down-convert and digitize RF signals in the range of DC to 2.9GHz, where each daughterboard covers an overlapping subset of this range. This RF front end was chosen for its flexibility and ease of use. The output of the RF front end is 8-bit complex I/Q samples output via a USB cable.

The software receiver processing of the RF front-end outputs is accomplished by using bit-wise parallelism, as described in References [1] and [2]. In order to process the incoming RF data in this manner, the 8-bit complex I/Q samples are quantized to two bits. This is performed in the software receiver prior to signal correlation. In-phase and quadrature accumulations are computed using bit-wise parallel techniques, and these accumulations are used to drive code tracking delay-lock loops (DLLs) and carrier tracking phase-lock loops (PLLs). The computation of accumulations and the implementation of DLLs and PLLs for the GNSS ranging signals are detailed in the thesis.

The software receiver is developed by C++. It consists of two parts: the software receiver core program and a simple interface. The current software receiver runs under Ubuntu Linux systems, but it is convenient to implement on other Linux systems. The software prerequisites for the software receiver are GNUradio and QT4.0. GNUradio is an open source program which provides the driver for the USRP board. The current version used by the software receiver is GNUradio-3.1.3. The user interface program is developed by using the classes provided by QT4.0. The hardware of the whole system consists of computer with intel 1.84 GHz CPU and 2GHz RAM, GPS and GLONASS antenna, USRP, and analogue signal generator. One problem with the USRP is that its on-board oscillator is not particularly stable in terms of frequency and phase. One solution to this problem is to use a high-quality external oscillator. An Agilent N5181A MXG Analog Signal Generator configured to output a 64MHz signal has been used as an external input clock to the USRP. This oscillator has a stated frequency error of 1 ppm/yr, has decent short-term frequency stability, and has a reasonably low phase noise at 64MHz. The outputs of the USRP board are 8 bits complex data with 4MHz sampling frequency with an intermediate frequency of zero. The input data are re-quantized and pack into 32-bit of integers. The total CPU usage of the software receiver is about 30 $\sim$ 40% of the 1.84GHz CPU. The software receiver is started with a FFT based acquisition. The acquisition results are then used to initialize the receiver. The background search of satellites is accomplished by a serial search of PRN code replicas. The novelty of the the software receiver developed in this study is as follows: first, a reconfigurable RF front end is used which makes the software receiver extendable. Second, The software is developed with C++ in the general Linux system; This will make the software receiver easy to maintain and update. Third, the current software receiver also

explores the process of GLONASS L1 signals with bit-wise parallel correlation.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Wayne Scales, who have provided support for this research within GPS lab of Virginia Tech along the way. The work in this dissertation would be impossible without the kind, patient and generous help from him. I would also like to acknowledge Dr. Brent Ledvina, who introduced me into the satellite navigation field. The classes taught by Dr. Ledvina in GPS theory and software receiver provided me the chance to study navigation satellite system systematically. His great passion and expertise helped me to shape my professional career. The help and assistance provided by friends and faculty will also be appreciated. Finally, I am greatly thankful to my parents, thanks your everlasting love and support.

Contents

List of Tables

List of Figures

List of Acronyms

CDMA	Code division multiple access
C/N_0	Carrier-to-Noise ratio
DLL	Delay Lock Loop
DOP	Dilution Of Precision
FDMA	Frequency-division multiple access
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FLL	Frequency Lock Loop
GLONASS	Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IF	intermediate frequency
NCO	numerically controlled oscillator
PLL	Phase Lock Loop
PRN	Pseudo Random Noise
PSD	power spectral density
RF	radio frequency
SDR	Software Defined Radio
SNR	signal-to-noise ratio
USRP	Universal Software Radio Peripheral

Chapter 1

Introduction

Global Navigation Satellite System (GNSS) is the general name given to satellite navigation systems that provide geo-spatial positioning with global coverage. There are currently two global systems in operation: the Global Positioning System (GPS) owned by the United States of America, and GLONASS (Global'naya Navigatsivannaya Sputnikovaya Sistema) of the Russian Federation. A third system called GALILEO is under development by the European Community (EC) countries. The first two test satellite of the Galileo system, GIOVE-A and GIOVE-B, were launched on December 28, 2005, and April 27, 2008, respectively. China is also involved in the development of its own system, Compass. China intends to first provide a regional capability for Compass/Beidou, followed by completion of its full constellation which consists of 30 middle earth orbit and five geostationary satellites after 2015 and before 2020.

1.1 Introduction to GPS

The Global Positioning System (GPS) is a space based satellites navigation system deployed by the United States and managed by the U.S. Department of Defense (DoP). The principle objective of the U.S. Department of Defense in developing GPS was to offer the U.S. military accurate estimates of position, velocity, and time [3]. The GPS system evolved into a dual use system as the results of tremendous application potential in transportation, land surveying, commerce, scientific uses, tracking and surveillance. Two kinds of service are provided:

1 Standard Positioning Service (SPS) for civil use,

2 Precise Positioning Service for military use.

The entire GPS system consists of three segments: the control segment, the space segment and the user segment as shown in Figure 1.1. The space segment consists of the satellites, precision clocks, and the signals they transmit to the ground, including the ranging singles and GPS navigation message [4].The control segment consists of five GPS earth stations. The main function of the control segment is to monitor the performance of the GPS satellites. It also determines the satellite ephemerides by monitoring the ranging signals. The ephemerides are parameters from which the satellite position can be accurately calculated. The ephemerides are regularly updated by the control segment. The user segment consists of receivers and their applications. The user segment covers activities related to the development of military and civil GPS user equipment.

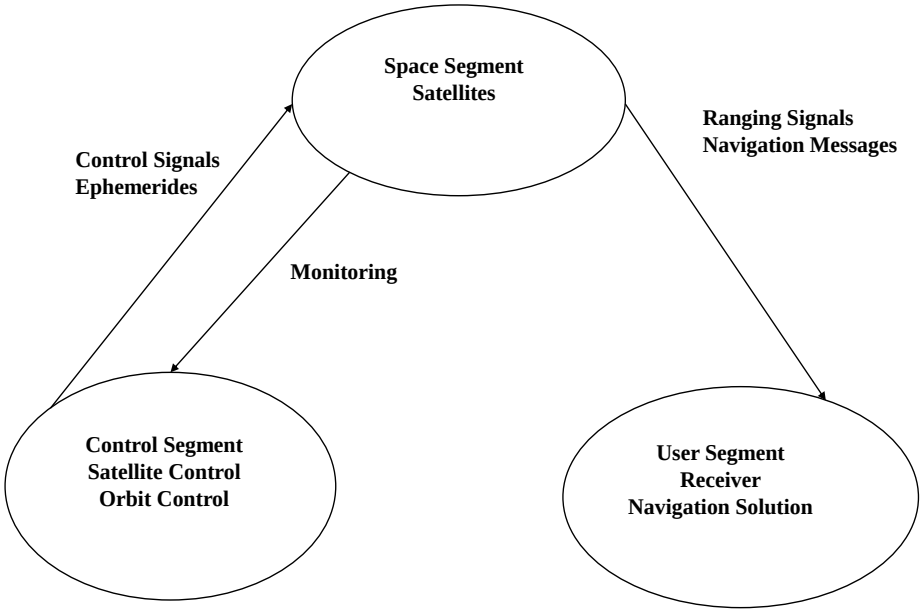

Figure 1.1: Segments of GPS

1.1.1 Space Segment

The space segment (SS) comprises the orbiting GPS satellites or Space Vehicles (SV) in nearly circular orbits with a radius of 26,560 km and a period of about twelve hours. The baseline constellation compromises 24 operational satellites which are distributed in six orbit plans with an inclination of 55 degree relative to the equational plan. The six planes and are separated by 60^o right ascension of the ascending node [3]. The orbits are arranged so that at least six satellites are visible from almost any point on the Earth. The satellites transmit ranging signals and navigation data allowing the user to compute their pseudoranges and to estimate their positions. The carrier signals are synchronized to the onboarded high accuracy atomic clock.

As of September 2009, the GPS constellation consists of 31 actively broadcasting satellites. The additional satellites improve the precision of GPS receiver calculations by providing redundant measurements. With the increased number of satellites, the constellation is arranged unevenly. Such an arrangement improves reliability and availability of the system. The first of the satellites was brought to its orbit in 1978. Since then, the satellites became more sophisticated; meanwhile, five different types of these satellites been developed (Block I, Block II, Block IIA, Block IIR and Block IIF). The next generation (Block IIF) plans to provide a third frequency L5 for civil use, allowing position determinations with even higher precision. The Block IIF satellites may be equipped with hydrogen maser clocks instead of atomic clocks, which provide higher stability.

1.1.2 Control Segment

There are five monitor stations located around the world. The master control station is located at the Schriever Air force base in Colorado Springs, CO. The other four additional monitoring stations are located in Hawaii, Ascension Islands, Diego Garcia and Kawajalein. During 2005, six more monitor stations of the National Geospatial Intelligence Agency (NGA) were added to the grid. Currently, every satellite can be monitored by at least two monitor stations. This allows calculating more precise satellite orbits and ephemerides. For the user segment, a higher position precision can be expected. In the near future, five more NGA stations will be added so that every satellite can be monitored by at least three monitor stations. This improves integrity monitoring of the satellites and

thus improve the precision of the GPS system. Figure 1.2 shows the location of the monitor stations.

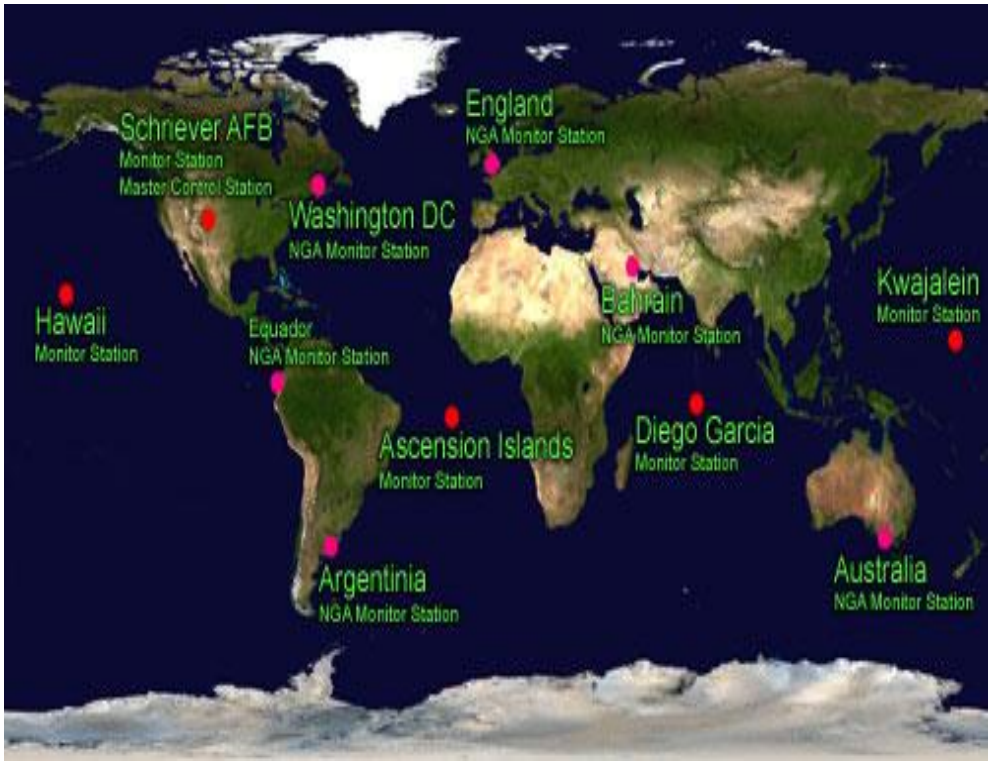

Figure 1.2: Position of the Monitor Stations and the Master Control Stations

1.1.3 User Segment

The user segment consists of the GPS receivers. The receivers compute the user position and velocity, as well as synchronized GPS time by utilizing the signals transmitted from the satellites, together with precise measurement of the signal transmission delays. The users can be classified into two groups: military and civilian. The military users can use both the military signals and the civilian signals but the civilian users are limited to civilian signals. The content and structure of the GPS signals will be detailed in the later part of the paper.

1.2 Introduction to GLONASS

The Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS), which translates into global navigation satellite system in English [5], is the satellite navigation system operated by the Russian Space Agency. A full constellation of the GLONASS satellites consists of 24 satellites deployed in three orbital plans. However, there are plans to upgrade the current system to 32 satellites in the full constellation. As of September 2009, there are 19 operational satellites in orbit, with 17 of these satellites broadcasting valid navigation messages. The Russian Space Agency is replenishing and modernizing GLONASS constellation. They plan to restore the global coverage of GLONASS by 2009 and to increase the number of satellites to 30 by 2011 to match the performance of GPS.

GPS and GLONASS share similarities in their signal structures, which translate into similarities in how a receiver could be designed to use either set of signals for navigation. Both systems broadcast carrier signals in the L-band frequency range. The frequency separation between the center of the GPS L1 band and lowest frequency content of the GLONASS L1 band is about 23MHz. The overall bandwidth of the GLONASS civilian L1 signals is about 18MHz. These systems both employ pseudo-random number codes for code modulation of the carriers. This provides the spreading of the signals and allows for precise measurement of pseudoranges. Additionally, both systems broadcast satellite-dependent navigation messages, which contain information about the satellite positions, clocks, health, and time.

Despite the similarities between GLONASS and GPS systems, there are several significant differences that are listed in Table 1.1. First, the GLONASS satellites are deployed

in three orbital planes while GPS satellites are deployed in six orbital planes. Second, GLONASS uses frequency division multiple access (FDMA) instead of code division multiple access (CDMA) used by GPS [4]. The third difference is in the time reference systems. GLONASS time is referenced to and synchronized with UTC Soviet Union (SU) time, the Russian National time scale, whereas, GPS time is referenced to international UTC with an integer number of leap seconds offset between GPS time and UTC. UTC and UTC (SU) are separated by a constant bias of three hours and a fractional bias that changes with time. The fourth difference is that GLONASS and GPS employ different geocentric coordinate systems. GPS uses WGS 84, while GLONASS uses PZ-90. Fortunately, the differences between these coordinate systems are small and do not vary with time. The GLONASS navigation message content is different from the GPS navigation message, which results in different algorithms being used in the GLONASS software receiver. The main differences between the GPS system and the GLONASS system are listed in table 1.1.

1.3 Introduction to GNSS Software Receiver

A software receiver performs the entirety of the receiver's digital signal processing using software running on a microprocessor. This differs from a hardware receiver, which typically performs correlation on a specialized processor or ASIC (application-specific integrated circuit) [6, 7]. The software receiver architecture can provide GNSS user with operational flexibility that will become more and more important as the modernization of GPS and construction of new satellite navigation systems. The current GPS system is to expand its capabilities to modulate new civilian codes on the L2 frequency and a new L5

Table 1.1: GLONASS and GPS system comparison

	GLONASS	GPS
Number of Orbital plans	3	6
Signal	FDMA	CDMA
Ephemeris representation	Position, Velocity, and Acceleration	Orbital Parameters
Time reference	UTC (SU)	GPS time
C/A Code rate	0.511M chips/s	1.023M chips/s
Coordinate system	PZ-90	WGS 84

frequency. A receiver with a hardware correlator will require hardware modifications in order to use these new signals. However, the software receiver can utilize new signals just by modification of the software.

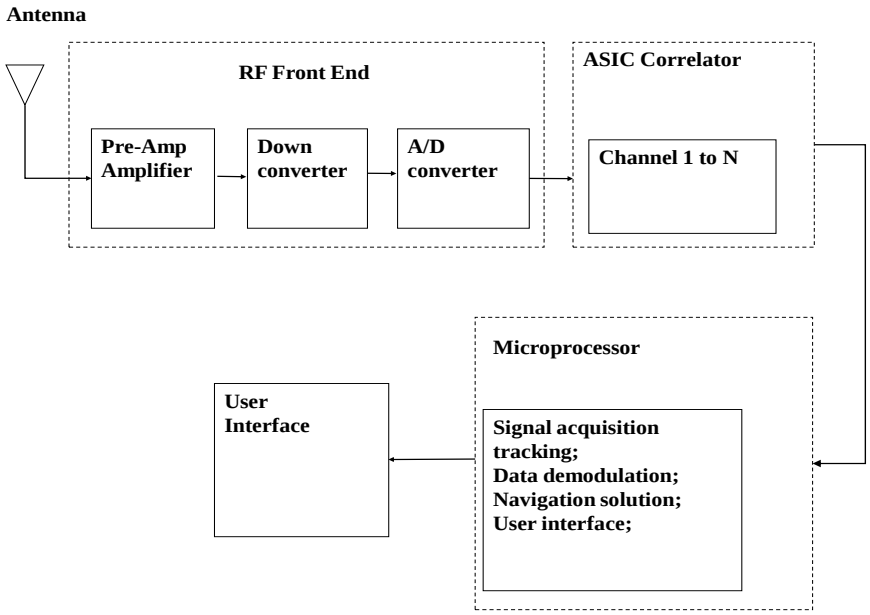

Figure 1.3: Structure of a Typical GNSS Receiver

A typical GNSS receiver can be broken down into separate functional components (see Figure 1.3). The first component is the antenna, possibly followed by some pre-amplifiers. After the antenna comes the RF front end that down converts the GHz GNSS signals to an intermediate frequency of MHz range. Then the signals are digitized and feed into the correlators. The basic function of the correlator is to mix the RF data with the local generated sinusoids and PRN codes. Each correlator is assigned to a different satellite and works parallel to other channels. A modern receiver may have 12 or more channels. The last components of the receiver are software programs that demodulate the navigation message, and compute the navigation solution. In the software receiver, the functions of the correlator chip are moved to software running on a general purpose processor. Software receiver provide more flexibilities than the hardware receiver [1].

The notion of a software GPS receiver has been around for several years. During the early years, the software receivers were implemented in post process [7]. The huge computation burden of the software receiver makes it difficult in full implementation on a simple processor. Due to the improvement in both software receiver algorithms and the performance of processors, real-time software receiver has been implemented on personal computers and DSP processors [8, 9, 10, 11]. With the decrease in the cost of DSP chips, software receiver products start to enter the market. The author predicts that in a few years the software receiver would become more important than the hardware receiver. The current work can be viewed as an extension of the work reported in Ref. 1. The software receiver developed here is using a general RF front end which can process a wide range of GNSS signals. The second improvement is that the software is developed with C++ in the basic Linux system. This will make the software receiver easy to maintain. The current software receiver also explores the process of GLONASS L1 signals.

Chapter 2

GNSS RF Front End

2.1 Software Receiver Architecture

The block diagram of the GNSS L1 software receiver is shown in Figure 2.1. The main difference between the hardware receiver and the software receiver is that the correlator is implemented by software instead of ASIC as shown in Figure 1.3. The analog signal output from the antenna is down-converted and digitized by the RF front end and then output to the software receiver. The main processing component of the GNSS software receiver is the PC's general-purpose microprocessor, which performs signal correlation, acquisition and tracking, decoding of the navigation message, measurement of pseudoranges, computation of the satellite positions, and computation of the navigation solution. The computer system consists of an Intel 1.8 GHz processor, running the UBUNTU Linux system. The rooftop antenna used with this software receiver is NovAtel's GPS-702-GG antenna, which provides access to the GPS L1 and L2 signals, as well as GLONASS's L1 and L2 signals [15]. It is mounted on the top of Whittemore Hall in Blacksburg, VA. The LNA gain of the

antenna is 29 dB. The pre-amplifier next to the antenna has 30dB gain.

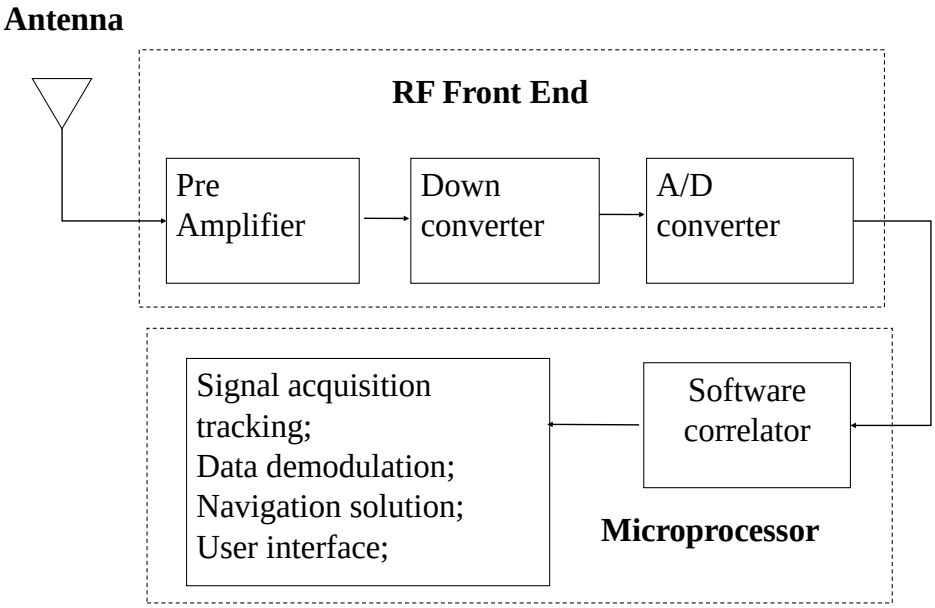

Figure 2.1: Struture of a Software GNSS Receiver

2.2 RF Front End Design

The RF front end used in this real-time software receiver is Universal Software Radio Peripheral (USRP), which is a hardware radio platform that provides RF signal reception and transmission over a broad range of radio frequencies. It provides the ability to rapidly design and implement software radio systems [16]. The USRP has been chosen primarily for its flexibility and ease of use. The USRP consists of a motherboard with A/D and D/A converters, mixers, filters, and an FPGA for low-level signal processing. Interchangeable

add-on daughtercards allow the USRP to process signals from DC to 2.9 GHz [16]. The broad flexibility of USRP allows it to process most of the civilian signals of satellite navigation systems, including GPS L1 and L2C, GLONASS L1, and Galileo E1. Note that due to bandwidth limitations, the URSP can only be used as an RF front end for one set of these signals at a time. Figure 2.2 is a picture of the USRP board.

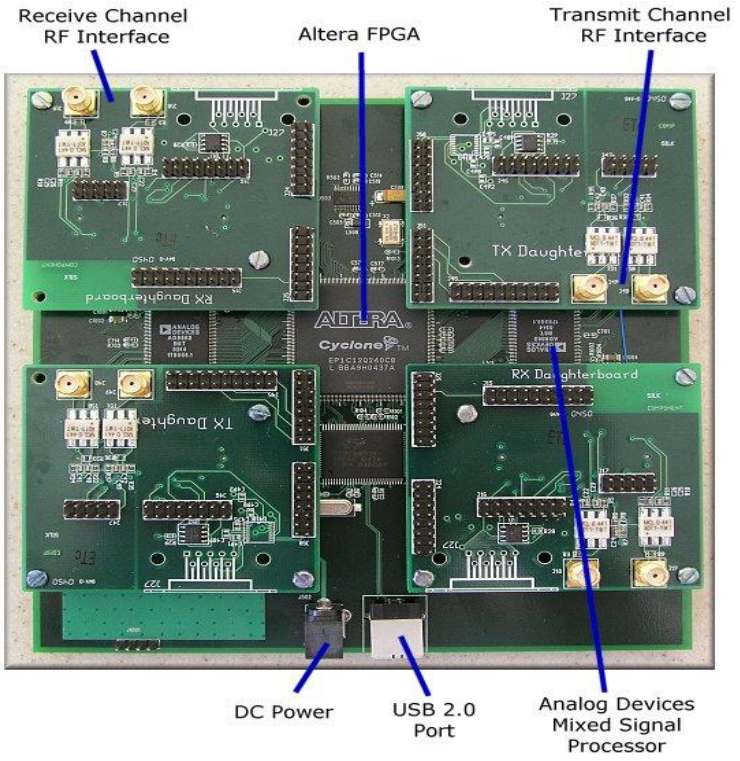

Figure 2.2: USRP mother board with two RX and two TX daughter boards

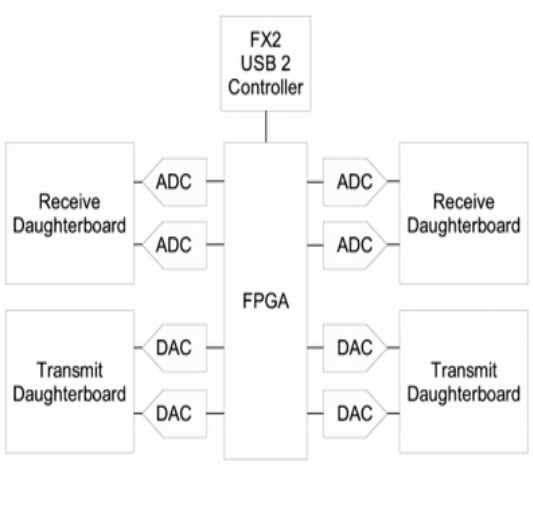

Figure 2.3: Schematic of a USRP board

The schematic of a USRP board is shown in Fig. 2.3. It has four high-speed analog to digital converters (ADCs) with sample frequency of 64MHz and 12 bits per sample. There are also four high-speed digital to analog converters (DACs) with sample frequency of 128MHz and 14 bits per sample. These four input and four output channels are connected to an Altera Cyclone EP1C12 FPGA. The USRP board connects to the computer via a high speed USB2 interface. The maximum sample frequency of the USRP is 8MHz complex sampling.

The driver of the USRP is provided by GNU Radio which is an open source project. The USRP is configured right after the start of the software receiver. As there is only one daughter board mounted on the USRP, the software receiver can only be a GPS L1 receiver or a GLONASS L1 receiver. The center frequency of the incoming signal, the sample frequency, down converted frequency and output data format are initialized at the beginning

of the program. The sample frequency of the incoming signal plays an important role in the design of the software receiver. This sample frequency must first meet the Nyquist sample theorem $F_s \geq 2 * B$, where B is the bandwidth of the signal to be sampled. A higher sample frequency will provide a more stable tracking loop and more accurate code arrival time. However, more computation will be involved in the tracking loop, which will decrease the real-time performance of the software receiver. Therefor, the choice of the sample frequency of the software receiver is based on the balance of these factors.

The software receiver described here works at a sampling rate of 4MHz for GPS L1 signals and 8MHz for GLONASS L1 signals. This sampling frequency covers the majority of the GLONASS bandwidth, but is not ideal for a fully-functional GLONASS L1 receiver since the full bandwidth of the GLONASS L1 signal has a bandwidth of 12.2MHz. However, since the GLONASS satellites are currently only broadcasting on channels -2 to +7, which cover a frequency band of 6MHz, this RF front end is adequate for the time being. One problem with the USRP is that its on-board oscillator is not particularly stable in terms of frequency and phase. One solution to this problem is to use a high-quality external oscillator. An Agilent N5181A MXG Analog Signal Generator configured to output a 64MHz signal has been used as an external input clock to the USRP [17]. This oscillator has a stated frequency error of $\leq \pm 1$ ppm/yr, has decent short-term frequency stability, and has a reasonably low phase noise at 64 MHz [17]. Figure 2.4 shows the USRP, along with the external oscillator, and the USB cable, which connects the USRP to the PC. Figure 2.5 shows the frequency domain plot of the collected GPS L1 data from the USRP board. The input signal is from the GSS6560 simulator. As the bandwidth of the GPS L1 signal is 2MHz, the sample frequency is set to 4MHz to meet the Nyquist sample theory. From the plot, we can clear see that the signal bandwidth is around 2Mhz.

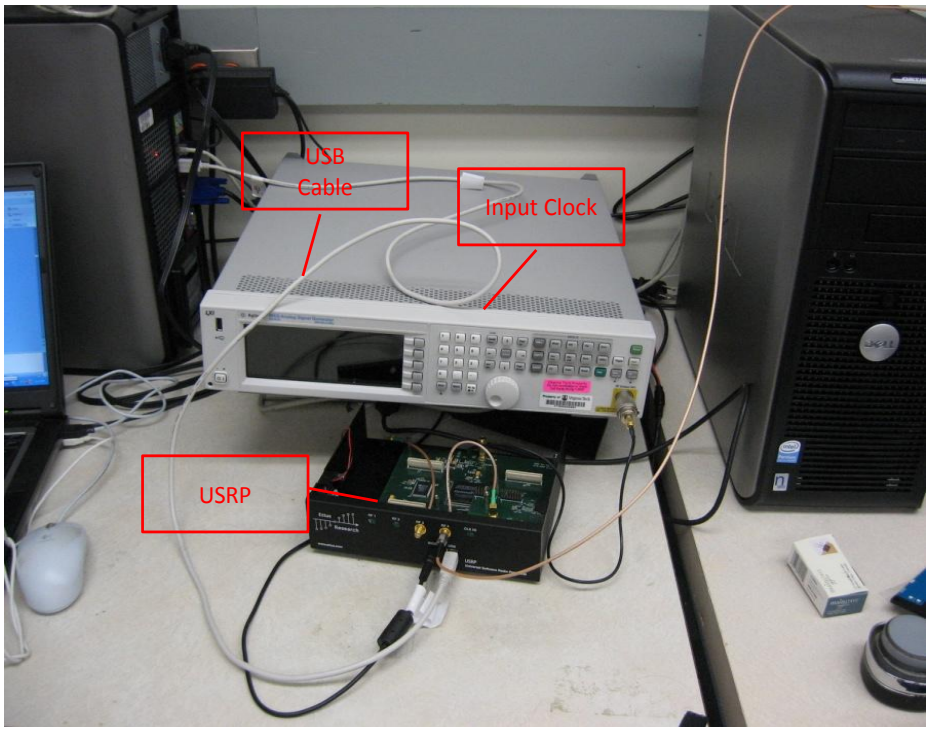

Figure 2.4: Photograph of the USRP and the External 64 MHz Clock

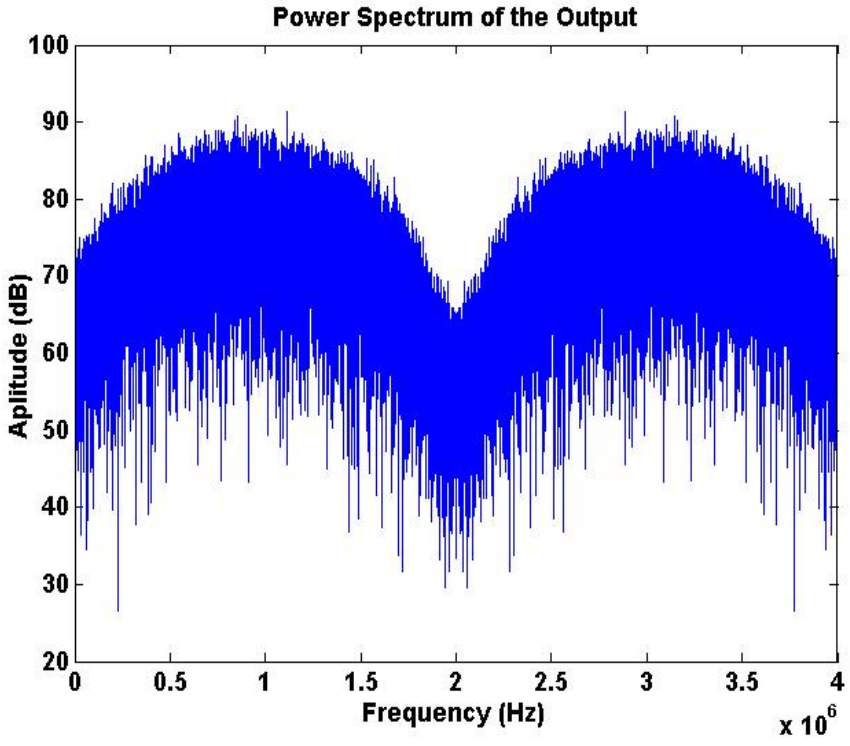

Figure 2.5: Frequency Domain (Magnitude) Plot of Collected GPS L1 data

Chapter 3

Acquisition of GPS and GLONASS L1 Signals

3.1 GPS L1 Signal Structure

In order to design a software receiver, it is essential to understand the structure of the signals. The GPS L1 signals are transmitted at a center frequency of 1575.42MHz. The signals are composed of carrier wave, navigation data and pseudo-random noise ranging code. Two levels of services are available: the military specific precise positioning service and the standard positioning service for civil applications. The broadcasted L1 signal structure is:

$$S_{L1} = \sqrt{2P_{C1}}D(t)X(t)\cos(2\pi f_{l1} + \theta_{l1}) + \sqrt{2P_{Y1}}D(t)Y(t)\sin(2\pi f_{l1} + \theta_{l1}), \tag{3.1}$$

In this equation, $\sqrt{2P_{C1}}$ is the amplitude of the civil signal which is the object of the software receiver discussed here, $D(t)$ is the navigation data, $X(t)$ or $Y(t)$ is the spread spectrum which is also called ranging code, and $\sin(2\pi f_{l1} + \theta_{l1})$ or $\cos(2\pi f_{l1} + \theta_{l1})$ is the

carrier. The coarse/acquisition (C/A) code is being used as the ranging code for the civil signal. Equation (3.1) indicates that all satellites are broadcasting signals at the same center frequency. Code-division-multiple-access technique (CDMA) is used to identify different SVs even though they may transmit at the same frequencies. The C/A code has a chip rate of 1.023Mbps and a period of 1ms, or 1023 chips in length. The C/A codes are a subset of the Golden code family. They are also referred to as pseudo-random noise sequence [Akos], or PRN sequence. The Gold codes are selected as spreading sequences for GPS signals because of their correlation properties. The two important correlation properties of the C/A codes are as follows [3]:

1 Nearly no cross correlation: All C/A codes are almost uncorrelated with each other. For two codes C_i and C_k from two different satellites, the correlation can be written as:

$$C_{ik}(m) = \sum_{l=0}^{1022} C_i(l) \times C_k(l+m) \approx 0, \tag{3.2}$$

Where $m \in (0, 1022)$.

2 Nearly no auto correlations except zero lag: All C/A codes are nearly uncorrelated to themselves, except for zero lag.

$$C_{kk}(m) = \sum_{l=0}^{1022} C_k(l) \times C_k(l+m) \approx 0, \tag{3.3}$$

Where $m \in (0, 1022)$.

The auto correlation properties can be used to find the exact start position of the C/A in the incoming signals. The cross correlation property can be used to separate different signals from different satellites. The generation of the GPS L1 C/A code is sketched in Figure . Two shift registers G_1 and G_2 are contained in the C/A code generator. Each

generator contains 10 cells generating code sequences of 1023 chips. The two sequences of 1023 chips are used to generate a new 1023 chips long C/A code by modulo-2 addition. The two shift registers are reset with all ones for every 1023 chip period. The code generation polynomial for register G_1 is [36]:

$$f(x) = 1 + x^3 + x^{10}, \tag{3.4}$$

This means that the modulo-2 addition of state 3 and state 10 are fed back to the input. The polynomial for register G_2 is:

$$f(x) = 1 + x^2 + x^3 + x^6 + x^8 + x^9 + x^{10}, \tag{3.5}$$

The outputs of the two registers are combined to generate C/A code for all satellites. The G_1 register provides its output, but the output of G_2 register is the modulo-2 addition of two of its states. The different combination of the two states of G_2 register will generate different C/A codes. A complete combinations table is provided in GPS ICD [36].

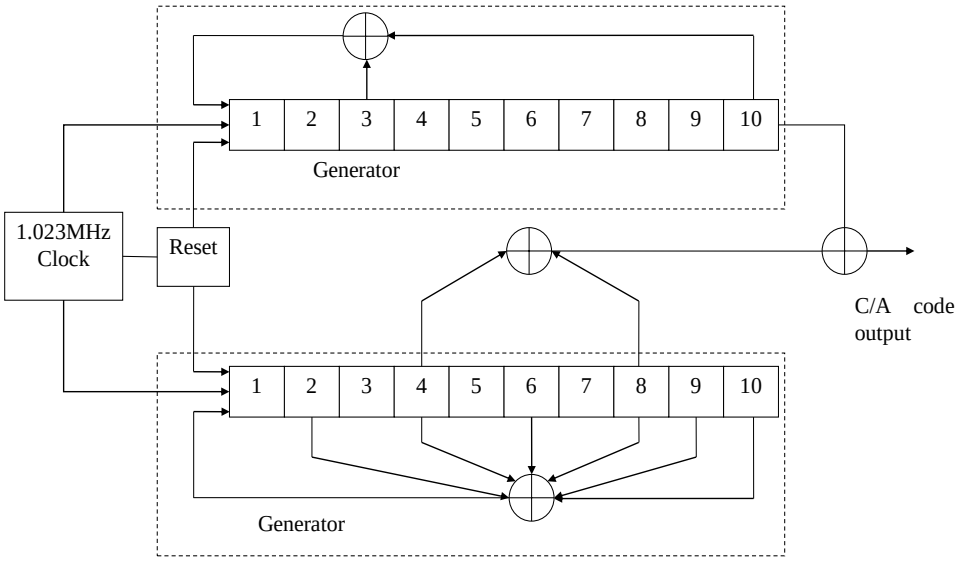

Figure 3.1: Structure of Shift Register Used for Ranging Code Generation

3.2 GLONASS L1 Signal Structure

GLONASS uses frequency division multiple access (FDMA) modulation to provide a receiver access to each satellite signal. This means each satellite transmits its carrier signal on its own sub-band. The nominal values of transmitted L1 carrier frequencies are defined by the following equations [5]:

$$f_{m1} = f + m \times \triangle f_1, f_{01} = 1602MHz;\ \triangle\ f_1 = 562.5KHz, (forL1sub - band) \tag{3.6}$$

where the channel number m ranges from -7 to 13 (0 inclusive), f_{m1} is the transmitted carrier frequency, and $\triangle f_1$ is the frequency offset between carriers. Currently, the GLONASS satellites use frequency channels with $m = [-2, 7]$. The L1 sub-carrier is modulated by a modulo-2 addition of a pseudorandom (PR) ranging code, the navigation data bits, and the meander code. The PR ranging code has a chipping rate of 511k chips per second and a period of 1 millisecond. The data bit rate of the navigation data bits is 50 bps, and the meander code has a bit rate of 100 bps. This meander code is phase-locked to the navigation data bits such that the resultant received navigation message has a bit rate of 100 bps. The PR ranging code, which is the same for each satellite, is generated using the polynomial [5]:

$$G(x) = 1 + x^5 + x^9 \tag{3.7}$$

The complex output signal of the receiver's RF front end is the combination of the signals from all satellites in view and can be written as:

$$y(t_n) \quad = \quad \sum_j A_j D_{jk} M_{jk} C \times [0.001 \times (\frac{t_n - \tau_{jk}}{\tau_{jk+1} - \tau_{jk}})]$$

$$\times[\cos(\omega_{IFj} + \theta_j(t_n)) + \sin(\omega_{IFj} + \theta_j(t_n))] + n_j \tag{3.8}$$

where t_n represents the sample time, j refers to the channel number of a GLONASS satellite in view, A_j is the amplitude, $M_{jk}$is the meander code, D_{jk} is the navigation data bit, $C[t]$ is the periodic PR ranging code, τ_{jk} and τ_{jk+1} are the start of the received k^{th} and $(k+1)^{th}$ PR code periods, ω_{IFj} is the intermediate frequency corresponding to the L1 carrier frequency of satellite j, $\theta_j(t_n)$ is the carrier phase, $\sin(\omega_{IFj}+\theta_j(t_n))$ is due to complex sampling, and n_j is the noise. The task of the receiver is to compute the accurate estimates of $(\tau_j, \tau_{j+1}, \omega_{Doppj}, \theta_j)$ by the correlation between the input signal and a local replica of it, where ω_{Doppj} is the carrier Doppler shift frequency.

3.3 Signal Acquisition

After the signals pass through the RF front end, the signals are down converted, amplified and digitized. The signals are now well suited for processing. The signal from one satellite has the form of:

$$S(t) = \sqrt{C}D(t-\tau)X(t-\tau)\cos(2\pi(f_{IF} + f_D)t + \delta\theta) + n(t), \tag{3.9}$$

where $\sqrt{C}$ is the amplitude of the incoming signal, $D(t-\tau)$ is the navigation data, $X(t-\tau)$is the PRN code, $\delta\theta$is the initial phase, f_{IF} is the intermediate frequency, f_D is the Doppler shift frequency, and $n(t)$is the noise. Doppler shift frequency is caused by the relative motion between the receiver and the satellite. The range of the Doppler shift is $\pm 10KHz$ for civilian applications. In order to demodulate the navigation messages from the satellite signal, a local PRN code and carrier must be generated. The purpose of the acquisition is to find out the visible satellites and the coarse values of the Doppler shift

frequency and the code phase of the signals. Two acquisition methods are used in this software receiver: the FFT based acquisition is used for initialization and the serial search acquisition is used in the background acquisition.

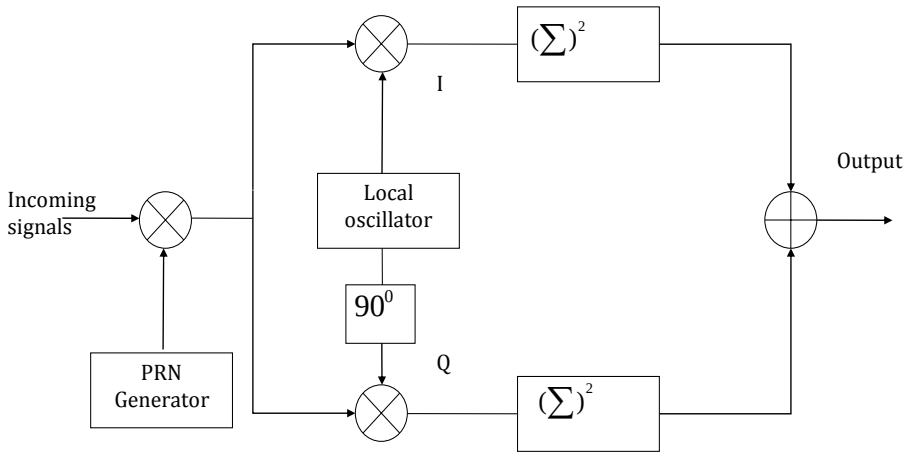

Figure 3.2: Block Diagram of Code Acquisition

The algorithm is based on multiplication of locally generated PRN code and locally generated carrier signals. As the initial phase of the incoming signal is unknown, the in-phase and quaduature carrier are generated to mix with the input. The output is the sum of the in-phase and quaduature mixing. The serial search algorithm performs two different sweeps [6]: a frequency sweep over all possible carrier Doppler shift frequencies and a code phase sweep over all different code phases. Obviously, this serial search involves a large number of computations. To make the software receiver running in real-time, the bit-

wise parallel correlation is used which will be explained in greater detail in the later chapter.

3.4 FFT Based Acquisition

Another more computation-efficient method for signal acquisition is to use the Fast Fourier Transform (FFT) to simultaneously search for all possible code offsets at a particular frequency [28]. This method is especially important when one wishes to increase acquisition sensitivity by extending the integration interval. The FFT based convolution is proved in reference [28]. In order to implement of the FFT based acquisition algorithm, The following steps can be taken:

1 Take the DFT of the PRN code samples c_k to get C_k

2 Select a Doppler frequency

3 Perform complex mixing of the incoming signal with local generated sinusoid. This operation will shift the incoming signal to baseband signal

4 Compute the DFT of the complex mixing results to get G_k

5 Multiply C_k with the complex conjugate of G_k; i.e., calculate $Z_k = C_k * G_k^*$

6 Take the inverse DFT of Z_k to get the correlation sequence z_k

7 Find the max value of $|z_k|$, if the max value exceeds the threshold, then the satellite is visible and the index is the code phase start position

Chapter 4

Signal Tracking

4.1 Software Receiver ARCHITECTURE

After the acquisition process is complete, the rough estimation of the Doppler shift frequency and the code phase are computed. The main purpose of the tracking loop is to refine these values, keep tracking, and demodulate the navigation data [6, 18, 21]. Figure 4.1 shows the scheme used to demodulate the input signal to obtain the navigation messages. First, the input signal is multiplied with a carrier replica. As the initial phase of the incoming signal is unknown, in-phase and quaduature replicas are generated. This multiplication is used to wipe off the carrier from the signal. Second, the signal is multiplied with a code replica, and the output of this multiplication gives the navigation message. The theory derivation is given in the following part.

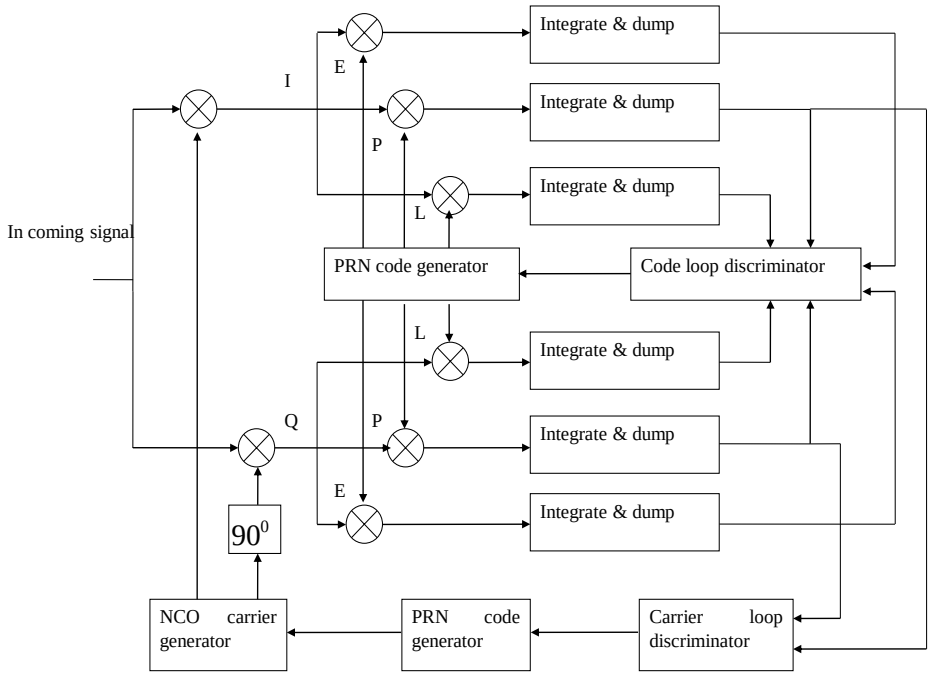

Figure 4.1: Diagram of the Tracking Loop

The signal from one satellite has the form of:

$$S(t) = \sqrt{C}D(t-\tau)X(t-\tau)\cos(2\pi(f_{IF}+f_D)t) + n(t), \tag{4.1}$$

To obtain the navigation data from the signal, the carrier and the PRN code need to be removed. The carrier removal is done by multiplying the input signal with a replica of the carrier. The local replica has the same frequency and phase with the incoing signal. The product of the multiplication is:

$$\begin{aligned} S(t)\times\cos(2\pi(f_{IF}+f_D)t) = {} & \sqrt{C}D(t-\tau)X(t-\tau)\cos(2\pi(f_{IF}+f_D)t)\cos(2\pi(f_{IF}+f_D)t) \\ & +n(t)\cos(2\pi(f_{IF}+f_D)t) \\ = {} & \frac{1}{2}D(t-\tau)X(t-\tau) + \frac{1}{2}D(t-\tau)X(t-\tau)\cos(2\times 2\pi(f_{IF}+f_D)t) \\ & +n(t)\cos(2\pi(f_{IF}+f_D)t) \end{aligned} \tag{4.2}$$

The latter part of the signal can be removed by applying a low pass filter. The resulting signal is:

$$\frac{1}{2}D(t-\tau)X(t-\tau) \tag{4.3}$$

The PRN code can be removed by multiplication of the signal with a local code replica. The phase of the replica should be exactly the same with the signal. The product of the multiplication is:

$$\sum_{n=0}^{N-1} X(t-\tau)X(t-\tau)D(t-\tau) = ND(t-\tau), \tag{4.4}$$

where the $ND(t-\tau)$ is the navigation message multiplied with the number of samples. The derivation above shows that a local carrier replica with accurate frequency and phase and a code replica with the exact phase are essential parts of the tracking loop. The following parts describe the tracking loop in the software receiver in detail. The tracking loop always consists of delay lock loop (DLL), phase lock loop (PLL) and frequency lock loop (FLL). The delay lock loop (DLL) refines the initial estimated code phase by the acquisition loop. The FLL refines the initial estimate of the Doppler frequency and tracks into the future. The PLL tracks the carrier phase and the carrier frequency.

4.2 Phase Lock Loop

The PLL is a basic building block for many subsystems used in the implementation of modern communication systems. PLLs are widely used in frequency synthesis, for frequency multipliers and dividers, for carrier and symbol synchronization, and in the implementation of coherent receivers [1, 2].Below is the block diagram of the phase lock loop [24, 34]:

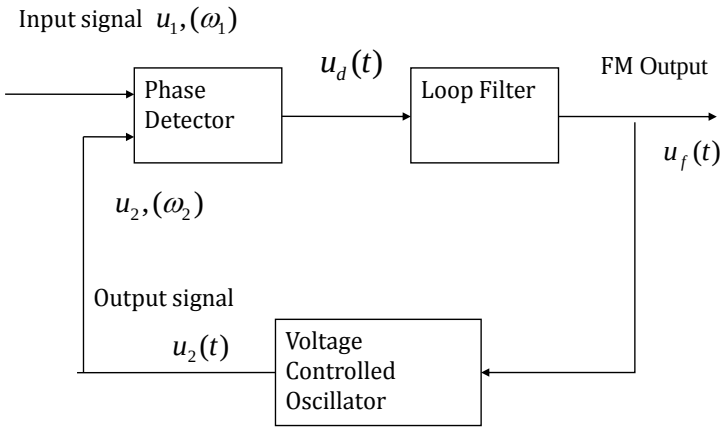

Figure 4.2: Blockdigram of the PLL

The signals of interest within the PLL circuit are defined as follows:

1 The reference or input signal $u_1(t)$

2 The angular frequency ω_1 of the reference signal

3 The output signal $u_2(t)$ of the VCO

4 The angular frequency ω_2 of the output signal

5 $u_d(t)$ the output signal of the phase detector

6 The output signal of the loop filter: $u_f(t)$

7 The phase error defined as the phase difference between input and output signals: θ_e

The VCO oscillates at an angular frequency of ω_2, which is determined by the output signal $u_f(t)$ of the loop filter. The angular frequency of VCO is given by:

$$\omega_2(t) = \omega_0 + K_0 u_f(t) \tag{4.5}$$

Where K_0 is the VCO gain.The output signal of the Phase Detector $u_d(t)$ which is approximately proportional to the phase error θ_e.

$$u_d(t) = K_d \times \theta_e \tag{4.6}$$

Where K_d is the gain of the Phase Detector. Let us analyze how the PLL blocks work together. First we assume the angular frequency of the input signal is equal to the center frequency of the VCO ω_0. The VCO then operates at its center frequency ω_0. In this case, the phase error θ_e is zero. If the phase error is zero, then the output signal u_d of the PD must also be zero. Consequently, the output signal of the loop filter u_f will also be zero. This is the condition that permits the VCO to operate at its center frequency.

Assume now that the frequency of the input signal is changed suddenly by the amount of $\Delta\omega$. The phase of the input signal starts leading the phase of the output signal. A phase error is built up and increases with time. The PD develops an output signal $u_d(t)$ which also increases with time. The output of loop filter, $u_f(t)$, which will also increase with $u_d(t)$. This will cause the VCO to increase the output frequency. The phase error becomes smaller now. After some settling time the VCO will oscillate at a frequency that is exactly the frequency of the input signal. The transfer function which relates the phase θ_1 of the input signal and the phase of the output θ_2 signal is given by:

$$H(s) = \frac{\theta_2(S)}{\theta_1(S)} \tag{4.7}$$

The input signal of a PLL is usually a sinusoid wave:

$$u_1(t) = U_{10} \sin(\omega_1 t + \theta_1) \tag{4.8}$$

The output signal is usually a square wave and can therefore be written as a Walsh function:

$$u_2(t) = U_{20} W(\omega_2 t + \theta_2) \tag{4.9}$$

To simplify the analysis, the Walsh function is replaced by the Fourier series:

$$u_2(t) = U_{20}[\frac{4}{\pi} \cos(\omega_2 t + \theta_2) + \frac{4}{3\pi} \cos(3\omega_2 t + \theta_2) \ldots] \tag{4.10}$$

The output signal of the phase detector when ω_1 equals to ω_2 is:

$$u_d(t) = u_1(t) \times u_2(t) = U_{10} U_{20}[\frac{2}{\pi} \sin(\theta_e) + \ldots] \tag{4.11}$$

When the value of θ_e is small, the output can be linearized as:

$$u_d(t) \approx K_d \sin(\theta_e) \approx K_d \theta_e \tag{4.12}$$

The angular frequency of VCO is given by:

$$\omega_2(t) = \omega_0 + K_0 u_f(t) \tag{4.13}$$

The phase θ_2 is given by the integral over the frequency variation:

$$\theta_2(t) = K_0 \int u_f(t) dt \tag{4.14}$$

The Laplace transform is given by:

$$\theta_2(S) = \frac{K_0}{S} U_f(S) \tag{4.15}$$

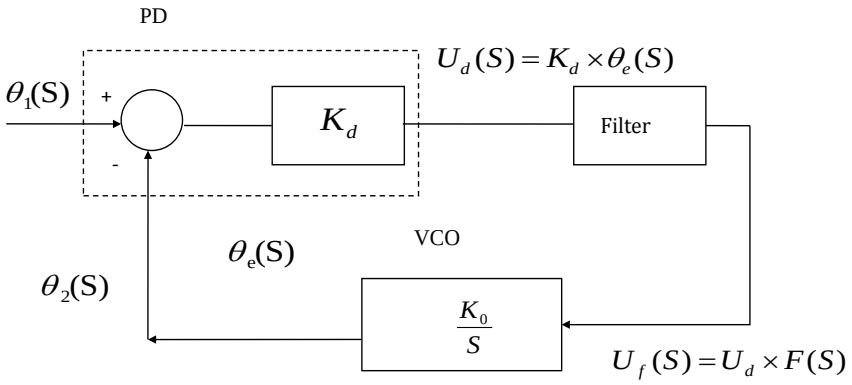

Figure 4.3: frequency domain model of the PLL

Figure 4.3 shows the model of the phase lock loop in frequency domain and how the preceding values related. The input of the loop can be frequency, phase and chips (Delay lock loop). The main purpose of the discriminator is to compute the error between the input and the output, and then we can compute the output through the transfer function. Let the loop gain be:

$$K = K_d * K_0 \tag{4.16}$$

The constant loop gain can be put into the filter function to simplify the phase lock. So the transfer function of the system can be simplified as :

$$H(S) = F(S)/S/(1 + F(S)/S) \tag{4.17}$$

For a second order loop

$$F(S) = K(S + A)/S \tag{4.18}$$

$$H(S) = (K(S + A))/(S^2 + KS + KA) = (K(S + A))/(S^2 + 2\xi\omega S + \omega^2) \tag{4.19}$$

The relationship of these coefficients:

$$\omega^2 = KA, K = 2\xi\omega \tag{4.20}$$

For a second order loop the noise bandwidth is computed as [23]:

$$B_L = \frac{\omega}{8\xi}(4\xi^2 + 1) \tag{4.21}$$

Implementation of the filter in discrete time [26]:

$$S = (Z - 1)/T \tag{4.22}$$

or

$$S = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{4.23}$$

$$F(S) = \frac{K(S + A)}{S} \tag{4.24}$$

$$F(Z) = \frac{(K(Z - 1) + ATK)}{(Z - 1)} \tag{4.25}$$

The output frequency from the carrier filter is:

$$f = F(Z) \times \theta_e \tag{4.26}$$

In software receiver, the VCO is replaced by a local sinusoid generator program. The new center frequency of the sinusoid can be computed by the equation above if the phase error θ_e is known. The Costas loop is adopted in the GPS receiver to estimate the phase error.

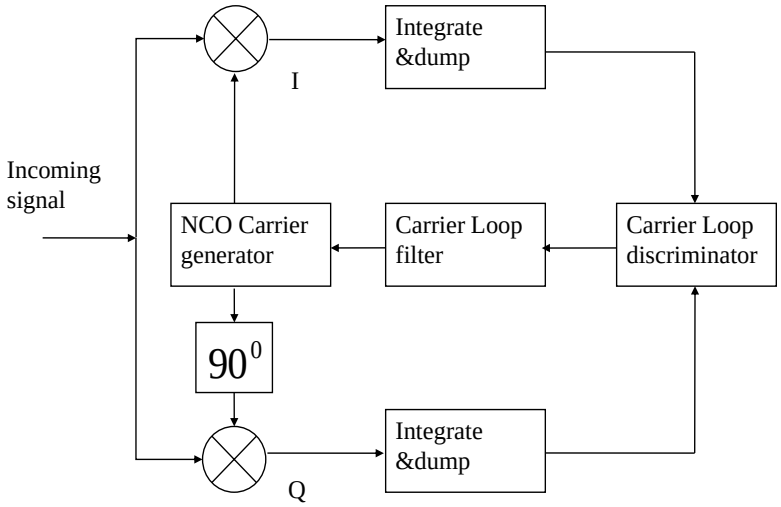

Figure 4.4: Block Diagram of the Costas loop

Figure 4.4 shows the block diagram of the Costas loop where the input signal is $\cos(\omega_{IF}t)$ [27]. The carrier generator generates in-phase $\cos(\omega_{IF}t + \theta_e)$ and quaduature sinusoids $\sin(\omega_{IF}t + \theta_e)$ to mix with the input signal where ω_{IF} is the IF frequency and θ_e is the phase error between in input and the local sinusoid. The in-phase arm mix result is:

$$\cos(\omega_{IF}t) * \cos(\omega_{IF} + \theta_e) = 1/2\cos(\theta_e) + 1/2\cos(2\omega_{IF}t + \theta_e) \tag{4.27}$$

The quaduature arm mix result is:

$$\cos(\omega_{IF}t) * \sin(\omega_{IF} + \theta_e) = 1/2\sin(\theta_e) + 1/2\sin(2\omega_{IF}t + \theta_e) \tag{4.28}$$

Then the mix results are integrated over time interval T. The double IF frequency term is removed by the integration and the following part remains:

$$\frac{1}{2}\cos(\theta_e) \times T \tag{4.29}$$

$$\frac{1}{2}\sin(\theta_e) \times T \tag{4.30}$$

The phase error can be estimated by an *arctan* discriminator:

$$arctan(\frac{\frac{1}{2}\sin(\theta_e) \times T}{\frac{1}{2}\cos(\theta_e) \times T}) = \theta_e \tag{4.31}$$

The noise bandwidth used in the software receiver is 10Hz. The choosing of the loop noise bandwidth is referenced to [3, 18]. The *arctan* discriminator is selected as it is sensitive to frequency error but insensitive to the unknown carrier phase and data bits.

4.3 Frequency Lock Loop

The automatic frequency control (AFC) loop which is also called the frequency lock loop (FLL) is quite similar to the PLL [31]. In contrast to PLL, the FLL has a better performance in the presence of high Doppler rates, spectrum multipath and ionospheric anomalies. As the result, a FLL is used to lock the Doppler shift frequency prior to the PLL in the software receiver. After incoming signal been tracked by the FLL, the carrier loop switches to the more accurate PLL to get the phase information of the signal. A typical discrete time FLL diagram is shown below in Figure 4.5. The integration filters are used to limit the input noise and remove double frequency parts after the mixing of the incoming signal and the local sinusoids.

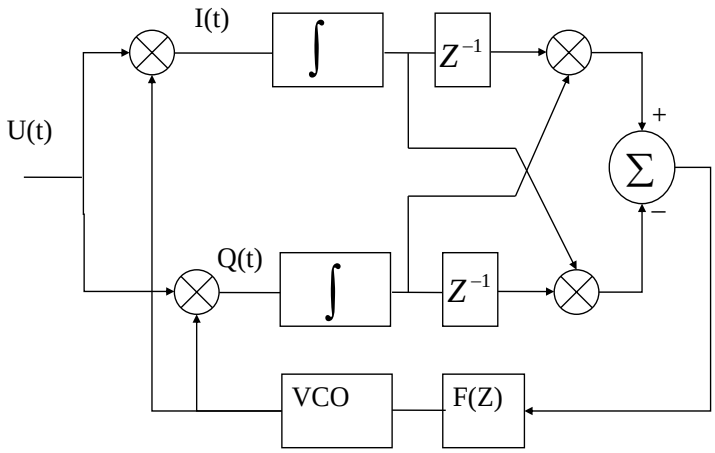

Figure 4.5: Block Diagram of FLL

Figure 4.5 shows the block diagram of the FLL and the effect of noise is not taken into account for simplicity. The initial output frequency of the VCO is an estimation of the incoming signal $f + \Delta f$. We want to keep the VCO output close to the input signal frequency f. The basic operation principle of the FLL is quite similar to the PLL. An in-phase and a quadrature arm are generated to mix with the incoming signal. The mix results for in-phase and quadrature are:

$$I(t) = A\times\cos(2\pi ft)\times\cos(2\pi(f+\Delta f)t) = \frac{1}{2}A\times\cos(2\pi\Delta ft)+\frac{1}{2}A\times\cos(2\pi(2f+\Delta f)t) \quad (4.32)$$

$$Q(t) = A\times\cos(2\pi ft)\times\sin(2\pi(f+\Delta f)t) = \frac{1}{2}A\times\sin(2\pi\Delta ft)+\frac{1}{2}A\times\sin(2\pi(2f+\Delta f)t) \quad (4.33)$$

The high frequency parts are removed by the integrator. Let the integration interval be T.

$$I_{k-1} = \int_0^T A\times\cos(2\pi\Delta ft)dt = \frac{A}{(2\pi\Delta f)}\times\sin(2\pi fT) \quad (4.34)$$

$$Q_{k-1} = \int_0^T A \times \sin(2\pi\Delta f t)dt = \frac{A}{(2\pi\Delta f)} \times (1 - \cos(2\pi\Delta f T)) \tag{4.35}$$

$$I_k = \int_0^T A \times \cos(2\pi(\Delta f t + \Delta f T))dt = \frac{A}{(2\pi\Delta f)} \times [\sin(2\pi\Delta f 2T) - sin(2\pi\Delta f T)] \tag{4.36}$$

$$Q_k = \int_0^T A \times \sin(2\pi(\Delta f t + \Delta f T))dt = \frac{A}{(2\pi\Delta f)} \times [\cos(2\pi\Delta f T) - cos(2\pi\Delta f 2T)] \tag{4.37}$$

The discriminator output is represented by

$$D = \frac{(Q_k \times I_{k-1} - I_k \times Q_{k-1})}{(amplitude(I, Q))} \tag{4.38}$$

By some computation we can simplify the discriminator output as:

$$D = \sin(2\pi\Delta f T) \tag{4.39}$$

So after the linearization of the cross product, the result is the difference between the input frequency and the VCO output frequency times the sample interval T.

In the frequency lock loop, the Jaffe-Rechtin filter is chosen as the low pass filter. Jaffe-Rechtin filter is widely used in software GPS receivers. Unfortunately, I cannot find any material going through in detail about how to implement it. So I try to write down step by step how to implement the Jaffe-Rechtin filter in the frequency lock loop. Here a second order Jaffe-Rechtin filter is used as an example. The transfer function of the filter is given by [30]:

$$F(s) = \frac{(B^2 + \sqrt{2}BS)}{AKS} \tag{4.40}$$

Where B is the noise bandwidth, A,K are amplify coefficients. In S domain the Laplace transform of integration is [33]:

$$L(\int) = \frac{1}{S} \tag{4.41}$$

Then we can simplify the automatic frequency control loop to be as shown in Figure 4.6 which is similar to the block diagram of the PLL:

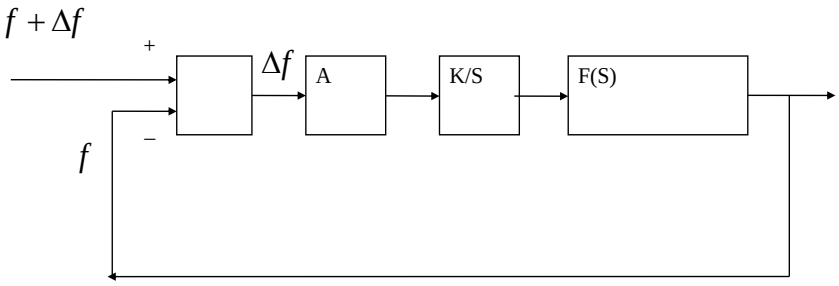

Figure 4.6: Frequency Domain Model of FLL

The input to the frequency loop is $f + \Delta f$. The output from the loop is f. The purpose of the loop is to compute Δf to update the output of the tracking loop. The transfer function of this system is computed as:

$$H(S) = \frac{(AKF(S))}{(S + AKF(S))} = \frac{(B^2 + \sqrt{2}BS)}{(S^2 + \sqrt{2}BS + B^2)} \tag{4.42}$$

The block diagram implementation of the system is shown in Figure 4.7:

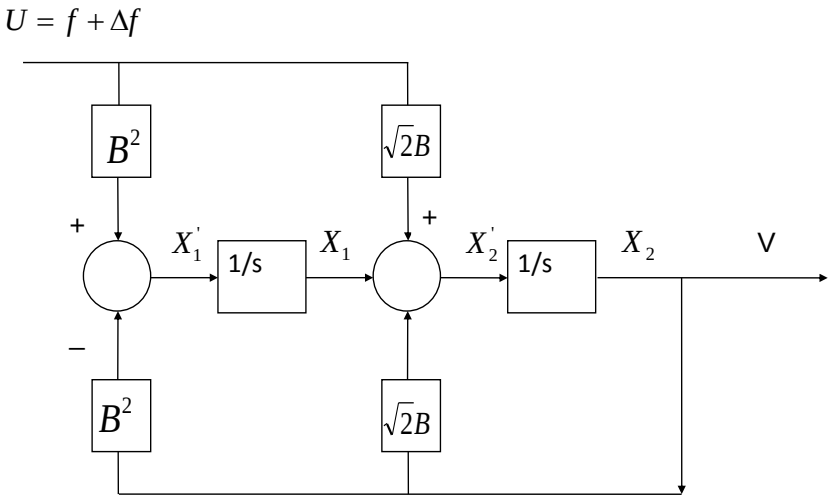

Figure 4.7: block diagram implementation of the FLL

The output of the system is V, the initial guess value of V is f. The initial input value is :

$$U = f + \Delta f \tag{4.43}$$

The state space equations for this system are:

$$X_1^{'} = B^2 \times (U - V) \tag{4.44}$$

$$X_2^{'} = X_1 + \sqrt{2}B \times (U - V) \tag{4.45}$$

$$V = [0\ 1] \times \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = X_2 \tag{4.46}$$

The Laplace transform of these equations above are:

$$S \times X_1 = B^2 \times (U(s) - V(S)) \tag{4.47}$$

$$S \times X_2 = X_1 + \sqrt{2}B \times (U(S) - V(S)) \tag{4.48}$$

Implementation these equations in discrete time by backward integration is:

$$S = \frac{(1 - Z^{-1})}{T} \tag{4.49}$$

So replace the S in (7) and (8) we have [22]:

$$X_1(n+1) = X_1(n) + T \times B^2 \times (U - V) = X_1(n) + T \times B^2 \times \Delta f \tag{4.50}$$

$$X_2(n+1) = X_2(n) + T \times X_1 + T \times \sqrt{2}B \times (U - V) = X_2(n) + T \times X_1 + T \times \sqrt{2}B \times \Delta f \tag{4.51}$$

The next states of the registers can be computed by the equations above if the value of Δf is known. From equation (1) we have:

$$\theta = \frac{(Q_k I_{k-1} - I_k Q_{k-1})}{(amplitude(I, Q))} = \sin(2\pi \Delta f T) \tag{4.52}$$

From the equation above we can get the frequency offset between the input and output. This offset is used as the input to equation to compute the next state values.

4.4 Delay lock loop

The purpose of the code tracking loop is to refine and keep tracking the code phase of the specific ranging code in the signal. The code tracking loop used in this software receiver is the delay lock loop [25]. This loop consists of correlator, accumulator, DLL discriminator and loop filter as is shown in Figure 4.8. The design of DLL is based on the autocorrelation and cross correlation property of the pseudo random code. The DLL correlates the received signal with a slightly early replica and a slightly late replica. As we know from the autocorrelation property, only when the two codes are perfectly aligned will a peak value

be computed. If the two codes are shifted by more than one chip, there is almost no correlation between them. As the shift increases from zero to one chip, the correlation results decrease almost linearly.

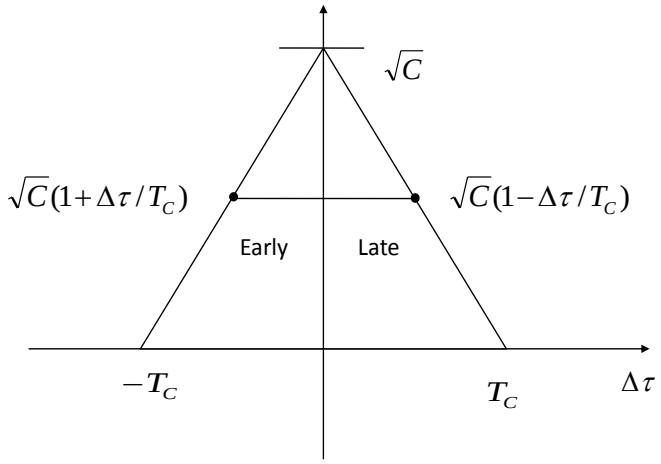

Figure 4.8: Early and Late Correlation Peak

The signal components of the early and late samples are given by:

$$S_E = \sqrt{C}R(\Delta\tau - dT_C/2) \tag{4.53}$$

$$S_L = \sqrt{C}R(\Delta\tau + dT_C/2) \tag{4.54}$$

where $R(\tau)$ is the autocorrelation function, T_C is the length of on chip of the pseudo random code, d is a constant value of the correlator spacing, $\Delta\tau$ is the offset between the prompt code and the object code in the signal, $\sqrt{C}$ is the maximum correlation value. The auto-

correlation function is given by [3]:

$$R(\Delta\tau) = \begin{cases} \frac{\tau}{T_C} + 1, & T_C < \tau < 0, \quad (4.55a) \\ \frac{-\tau}{T_C} + 1, , & 0 < \tau < T_C. \quad (4.55b) \\ 0, otherwise & (4.55c) \end{cases}$$

The time offset be computed by:

$$\Delta\tau = T_C(S_E - S_L)/2\sqrt{C} \quad (4.56)$$

The DLL discriminator provides the linear relationship of the time offset to the correlation results. Several typical types of DLL discriminators are described in the table 4.1.

The normalized early minus late envelope delay lock loop is very widely used. It removes the signal amplitude sensitivity, but it is very computationally expensive. The normalized early minus late envelope discriminator is used is this software receiver to track both GPS and GLONASS L1 signals. The difference between the power of early and late is calculated and sent to the discriminator. This difference indicates which one (early or late ranging code) contains more energy. The NCO must advance or delay the locally generated code according to this difference. When the power of the early and late correlators is the same, this means the prompt code lies perfectly with the incoming code and this is the objective of code tracking loop. The result of the discriminator is filtered and sent to the code NCO to update the chip rate of the pseudo random code.

4.5 Review of Bit-Wise Parallel Correlation

The correlator performs the mixing of input RF data with local code and carrier replicas, and accumulation of the base-band signals. The details of these operations are covered in

Table 4.1: GLONASS and GPS system comparison

Type	Discriminator	Characteristics
Coherent	$I_E - I_L$	Low computational load. Does not require the Q branch but require a good carrier tracking loop.
Non Coherent	$(I_E - I_L) \times I_P + (Q_E - Q_L) \times Q_P$	Dot-product power. This is the only DLL discriminator that uses all three correlators and this results is the lowest baseband computational load. For 1/2 chip early-late spacing, it produces true tracking error within 0.5 chip of input error.
	$(I_E^2 - I_Q^2) + (Q_E^2 - Q_Q^2)$	Early-minus-late power. Moderate computational load. For 1/2 chip early-late spacing, it produces true tracking error within 0.5 chip of input error.
	$\frac{((I_E^2 - I_Q^2) + (Q_E^2 - Q_Q^2))}{((I_E^2 + I_Q^2) + (Q_E^2 + Q_Q^2))}$	Normalized early minus late envelope. Highest computational load. For 1/2 chip early-late spacing, it produces true tracking error within 0.5 chip of input error. Becomes unstable at 1.5 chip input error.

the acquisition and tracking part. The main difficulty to implementing the real time software receiver is the speed of the general processor. In order to understand the computation complexity of the software receiver, we first examine how the hardware receiver works. A modem hardware receiver always has 12 or even more channels. Figure 4.9 shows the structure of a hardware channel from Zarlink GP 2021 [29]. Each channel is programmed individually and contains blocks to generate data used for tracking satellites signals.

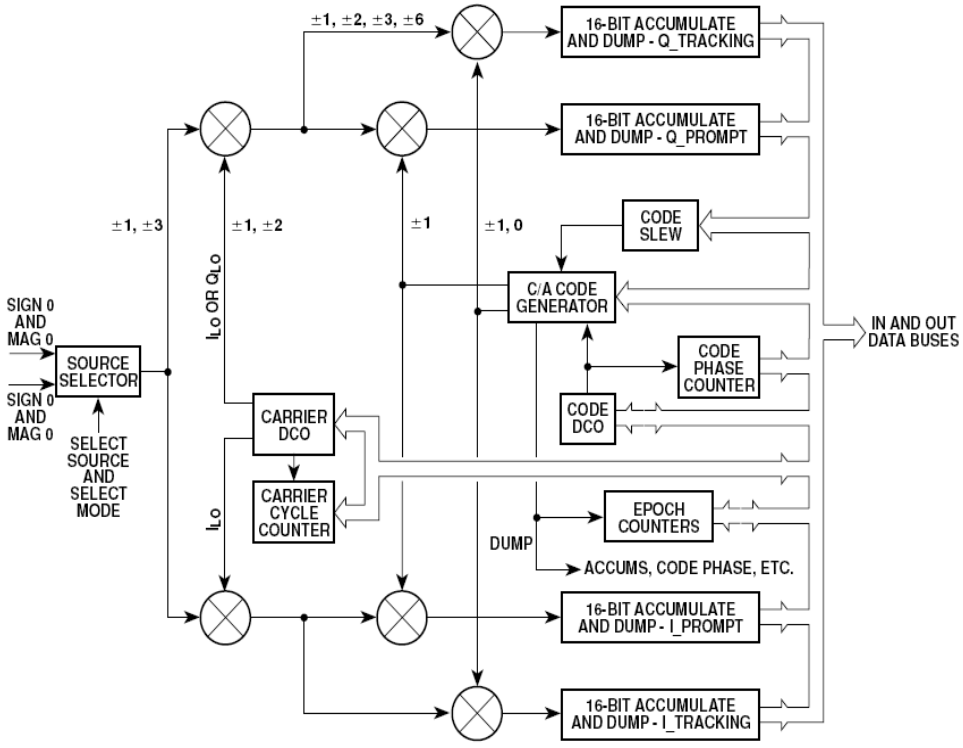

Figure 4.9: Tracking module block diagram

However, for a software receiver, the general purpose processor needs to perform all the tasks of the 12 channels. So the software receiver must be computationally efficient enough in order to operate in real time. An example is provided to show the rough estimation of total computations involved in a 12-channel software receiver, assuming the sample

frequency of the incoming signal is 5MHz complex samples. For in-phase and quadrature signal, the computations are summarized below:

1 Generation of sinusoid, 5M operations.

2 Generation of early, late and prompt codes, 3*5M operations.

3 Mixing between the incoming signal and the sinusoid, 5M operations.

4 Mixing with early, late and prompt code, 3*5M operations.

5 Summation of the mixing results, 3*5M operations.

These operations are performed by both in-phase and quadrature signals of 12 channels. So the total computation for one second is:

$$12 \times 2 \times (11) \times 5M = 1320M \tag{4.57}$$

This huge computation burden limits the capability of a software receiver. In order to speed up the correlation process, the software correlator is implemented using bit-wise parallel correlation [1]. This correlation method achieves computational performance that is 2 to 4 times better than brute-force fixed-point correlation. The basic principle of the bit-wise parallel algorithm is to operate on bit-packed samples in a parallel fashion by using EXCLUSIVE OR operations instead of fixed-point multiplication. This algorithm requires a low number of quantization levels for the RF signal, carrier replicas, and code replicas. The RF data are represented by two bits as shown in table 4.2. The sinusoids are sampled with 4-level and 8-phases as shown in table 4.3 [29], which can also be represented by two bit samples as in table 4.4. The PRN code samples are represented by one bit samples as in

Table 4.2: Sign and magnitude combinations of the input GPS signal.

Sign	Mag	Value
0	0	-1
0	1	-3
1	0	+1
1	1	+3

Table 4.3: 4-level, 8-phase Sample of sinusoids

Destination Arm	Sequence
I_{LO}	-1+1+2+2+1-1-2-2
Q_{LO}	+2+2+1-1-2-2-1+1

Table 4.4: Sign and magnitude combinations of the sinusoids

Sign	Mag	Value
0	0	-1
0	1	-2
1	0	+1
1	1	+2

table 4.5.

Table 4.5: Sample of Input Signal

Sign	Mag
1	+1
0	-1

Table 4.6: Sign, high-magnitude, low-magnitude, and zero- mask combinations of the fully mixed early-minus-late integrand.

Sign	High Mag	Low Mag	Value
0	0	0	-1
0	0	1	-2
0	1	0	-3
0	1	1	-6
1	0	0	+1
1	0	1	+2
1	1	0	+3
1	1	1	+6

A simple EXCLUSIVE OR multiplication of sign bits and a redefinition of data bits accomplishes base-band mixing. Multiplication of the RF front-end output representation of Table 1 by the sine wave representation of Table 2 and the PRN code in table 1 yields a result that can take on the values -6,-3,-2,-1,+1,+2,+3, and +6. These can be represented by 3 bits according to the scheme in Table 3. The high magnitude bit of Table 3 is sim-

ply the magnitude bit of the RF front-end output from Table 1, and the low magnitude bit of Table 3 is the magnitude bit of the base-band mixing sine wave from Table 2. Thus, these two magnitude bits are available without the need for computation. So the fix-point multiplication is replaced by the EXCLUSIVE OR multiplication. If all the sign bits are packed into integers, one EXCLUSIVE OR operation can process 32 samples on a 32 bits processor. As a result, the speed performance of the bit-wise parallel algorithm is improved significantly compared to the traditional fix-point computation.

Figure 3 shows the block diagram of the bit-wise parallel correlator and the code and carrier tracking loops. Code and carrier replicas are required for the code mixing and base-band mixing stages of correlation. The possible code phases of the PRN code and the Doppler shift frequencies are continuous values, so it is undesirable to store all possibilities in the limited computer memory, nor is it practical to generate these signals in real time as this process can be computationally expensive. Instead of storing all the possible values, a finite set of code replicas with initial code phases of fractions of one chip are generated, and carrier replicas covering a coarse grid of carrier Doppler shift frequencies about each intermediate frequency with initial phase offsets of zero are pre-computed and stored into tables in memory. An additional post-correlation fix-up is required because of the use of carrier replicas that occupy coarse Doppler shift frequencies have initial phase offsets with the incoming signals. The fix-up is to rotate the accumulation outputs. The detail of this algorithm is detailed in the paper [1]. The use of the finite set of code and carrier replicas causes a small, but negligible, loss in power.

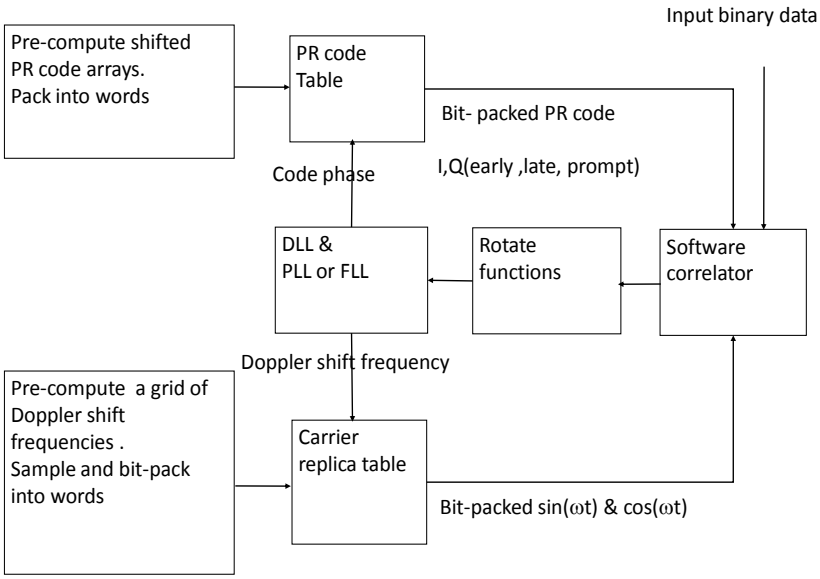

Figure 4.10: Memory usage of the bit-wise parallel The sample of the sinusoid

Chapter 5

Navigation Data Processing

5.1 GLONASS Message Structure

The navigation message provides the user with requisite data for determining the satellite positions, satellite clock offsets, satellite health, satellite time, etc. The algorithms to process the GPS messages are describe in reference [6, 19], so this paper is going to emphasize the processing of the GLONASS navigation message. The structure of the GLONASS navigation message consists of super-frames, sub-frames, strings, and bits. The super-frame has a duration and nominal repetition rate of 2.5 minutes. Each super-frame consists of five subframes, where each subframe has a duration of 30 seconds. Each subframe consists of 15 strings each with a duration of two seconds. The first 1.7 seconds of the GLONASS message string is 85 bits of navigation data encoded by a relative code (also called a differential code), and the last 0.3 second is the 30-bit time mark, which is used to locate the start of the next string, in a fashion similar to locating the preamble in a GPS navigation message. The first four strings of the super-frame contain the immediate data. Immediate

data is GLONASS's term for the satellite position, velocity, acceleration, and timing corrections needed to compute the satellite positions at any time.

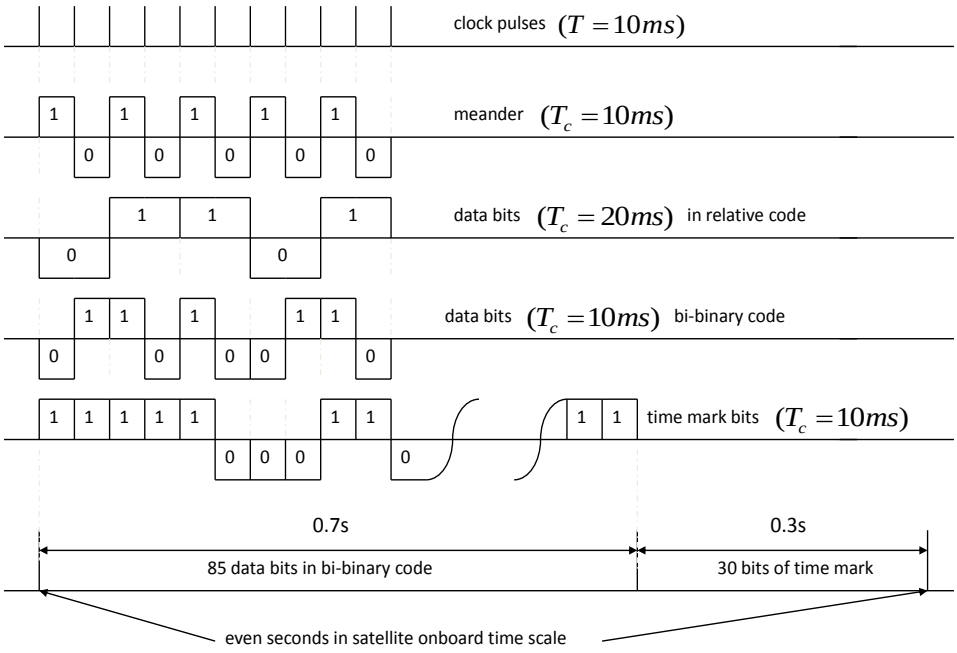

Figure 5.1: Data Sequence Generation

The broadcast navigation message consists of data bits with apparent bit lengths of ten ms. The navigation message in constructed via modulo-2 addition of the true data bit time history with a bit length of 20 ms and the meander code with a bit length of ten ms. The true data bits and the meander code are phase locked so that the resulting navigation message bits have a bit length of ten ms. The steps to decode the GLONASS navigation message are in Figure 5.1. These steps are taken directly out of the GLONASS ICD. For additional detail, please refer to the ICD [36].

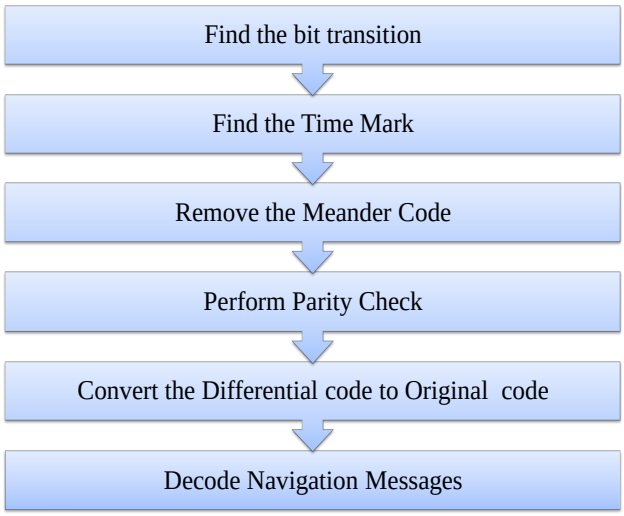

Figure 5.2: Multi-step approach to decoding the navigation message

5.2 Computation of GLONASS Satellite Positions

Computation of the GLONASS satellite positions is a requisite component of the receiver's navigation algorithm. Computation of the satellite positions is undertaken using a procedure that relies on numerical integration of first-order ordinary differential equations that define the motion of the satellites. There are two necessary components for calculating the satellite positions. The first is the initial conditions and the second is the numerical technique used to solve the differential equations.

The initial conditions are part of the broadcast GLONASS navigation message. The initial conditions at time t_b are the satellite position at time t_b, the satellite velocity at time t_b, other parameters needed to compute the satellite acceleration at time t_b, the satellite lock

offset and a relativistic correction. The satellite acceleration parameters contain the perturbations due to the Earth's gravitational force and the lunar and solar force on the satellite. These initial conditions and parameters, as defined by the GLONASS ICD, are given in Table 5.1 along with their units. These initial conditions are defined in the PZ-90 geodetic coordinate system, which is the primary coordinate reference frame used by GLONASS. The initial conditions and parameters are nominally updated every 30 minutes for each satellite by the GLONASS control segment.

Table 5.1: Several important parameters in GLONASS message

Description	Symbol	Units
Reference time	t_b	minutes
Relativistic correction	γ_n	dimensionless
Satellite clock offset	$\tau_n(t_b)$	seconds
Satellite position	$x_n(t_b), y_n(t_b), z_n(t_b)$	km
Satellite velocity	$\dot{x}_n(t_b), \dot{y}_n(t_b), \dot{z}_n(t_b)$	km/s
Lunar and solar perturbations	$\ddot{x}_n(t_b), \ddot{y}_n(t_b)$,$\ddot{z}_n(t_b)$	km/s^2

Once the initial conditions and parameters are obtained via decoding the broadcast navigation message, they are used in conjunction with a set of six first-order ordinary differential equations to solve the satellite positions for any time. The same set of equations is used to define the motion for each GLONASS satellite. Thus, only the broadcast initial conditions are satellite dependent. The set of differential equations used to compute satel-

lite positions are as follows[12, 13]:

$$\frac{dx}{dt} = V_x \tag{5.1}$$

$$\frac{dy}{dt} = V_y \tag{5.2}$$

$$\frac{dz}{dt} = V_z \tag{5.3}$$

$$\frac{(dV_x)}{dt} = -\frac{\mu}{r^3}x - \frac{3}{2}J_0^2\frac{(\mu a_e^2)}{r^5}x(1 - \frac{(5z^2)}{r^2}) + \omega^2 x + 2\omega V_y + \ddot{x} \tag{5.4}$$

$$\frac{(dV_y)}{dt} = -\frac{\mu}{r^3}y - \frac{3}{2}J_0^2\frac{(\mu a_e^2)}{r^5}y(1 - \frac{(5z^2)}{r^2}) + \omega^2 y + 2\omega V_x + \ddot{y} \tag{5.5}$$

$$\frac{(dV_z)}{dt} = -\frac{\mu}{r^3}z - \frac{3}{2}J_0^2\frac{(\mu a_e^2)}{r^5}z(3 - \frac{(5z^2)}{r^2}) + \ddot{z} \tag{5.6}$$

$$r = \sqrt{x^2 + y^2 + z^2} \tag{5.7}$$

where μ is gravitational constant, a_e is the semi-major axis of Earth, J_0^2 is the second zonal harmonic of the geopotential, (x, y, z) is the initial satellite position at time t_b,ω is the Earth rotation rate, (V_x, V_y, V_z) is the initial satellite velocity at time t_b, and $(\ddot{x}_n, \ddot{y}_n, \ddot{z}_n)$ is the initial satellite acceleration due to the Earth's gravitational acceleration and lunar and solar perturbations at t_b. It should be noted that the GLONASS ICD Version 5.0 [36] contains two errors in the set of six differential equations. The equations provided here are corrected equations and are consistent with those found in Ref [13].

Numerical integration is used to solve the six differential equations in order to compute a satellite position for any desired time. The GLONASS ICD recommends using 4th-order Runge-Kutta numerical integration [32]. This integration method has been implemented in the receiver described in this paper.

During development of the GLONASS software receiver described here there was a need to validate the computed satellite positions. This step would provide confidence that the GLONASS navigation message had been decoded correctly and that the numerical integration technique had been coded correctly. Additionally, comparison of the computed GLONASS satellite positions with a truth source would provide validation for the understanding of the time reference frame used by GLONASS.

Two methods have been used to gain confidence in the satellite position calculations. The first relies on using two successive sets of initial conditions decoded from the navigation message for one satellite. Two successive sets are continuous in time and are nominally separated by 30 minutes. Numerical integration forward in time for the earlier set and numerical integration backward in time for the later set should provide satellite position time histories that are continuous in time or nearly continuous, within the bounds on the accuracy of the initial conditions and satellite motion model defined in equations 8-14. Figure 8 shows the result of this test for SV GC719. At the midpoint time, which is precisely 15 minutes after t_b for the first set of initial conditions and 15 minutes prior to t_b in the second set of initial conditions. One can see that the satellite positions are nominally the same and the time histories are apparently continuous. Figure 10 shows a zoomed-in view of Figure 9, and although it is difficult to ascertain from this figure, the difference in the satellite positions at the midpoint time is 0.852 meters, which indicates that the satellite positions time histories are being computed consistently using two continuous sets of navigation messages. The two noted errors in the differential equations reported in the GLONASS ICD were uncovered during this test.

The second method used to gain confidence in the satellite position calculations is to compare the computed satellite positions with positions from the GLONASS official website, which provides satellite positions for each satellite every 15 minutes. The reason these satellite positions can be used for comparison with computed satellite positions is that they are defined at a time that is different than t_b. The reported satellite positions, which are denoted truth positions in Figures 5.3 and 5.4, are defined in GPS reference time. GPS reference time and GLONASS reference time are related by the equation [36]:

$$T_{GPS} - T_{GLO} = \Delta T + \tau_{GPS} \tag{5.8}$$

where ΔT is a bias which is 3 hours plus the current number of leap seconds used in GPS to define the time difference between GPS time and UTC, and τ_{GPS} is the fractional offset which is found in the GLONASS navigation message.

Truth satellite positions for every functional on-orbit GLONASS satellite are available starting at GPS time equal to zero, at the beginning of the GPS week, at a cadence of 15 minutes.

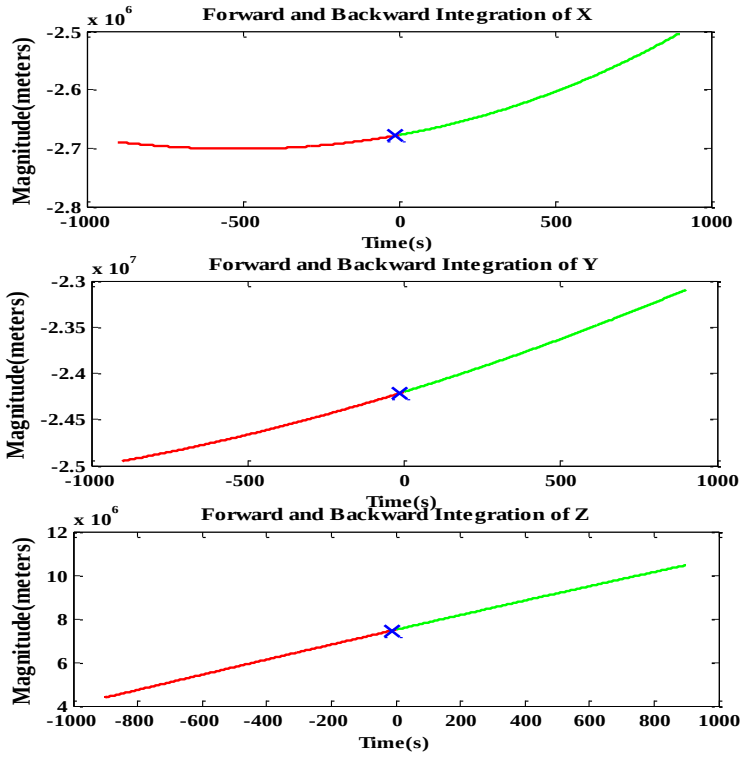

Figure 5.3: Satellite position time histories computed using backward and forward integration of the differential equations using two successive sets of initial conditions separated in time by 30 minutes for one SV

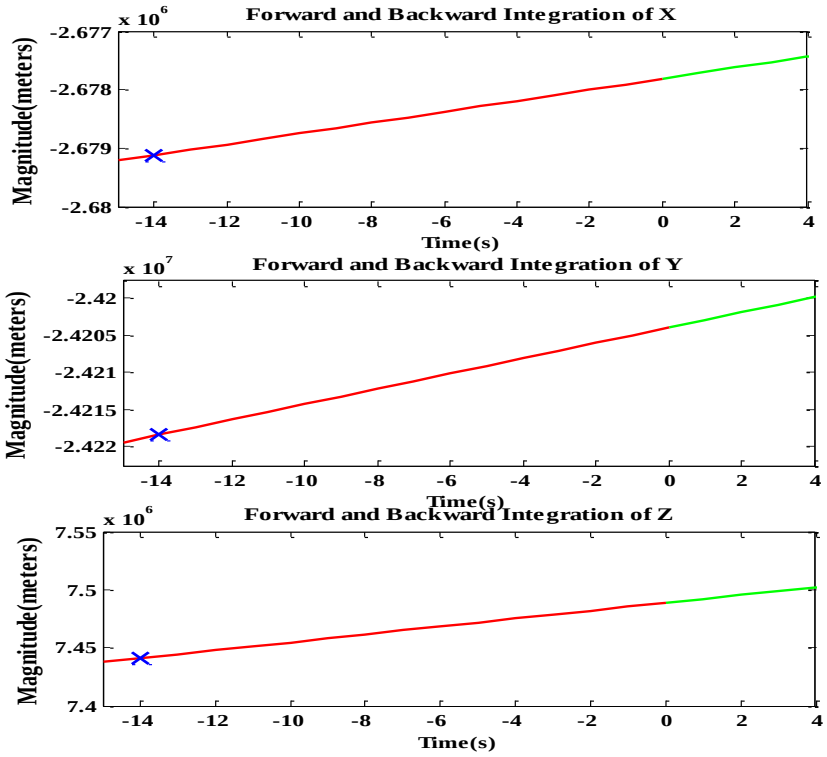

Figure 5.4: Zoomed-in version of Figure 9 showing the satellite position time histories

The residual error between the computed satellite position for SV GC719 and the truth position at time 20:30 GPS time on August 21, 2008 is 2.94 meters. Thus, the computed satellite positions are proving to be reasonable and the error is within the stated error budget for the GLONASS satellite position accuracy. This comparison suggests that both the GLONASS satellite parameters in the navigation message are being decoded properly and that the numerical integration technique is performing well.

5.3 pseudorange correction

As mentioned before, GLONASS is referenced to the UTC (SU), the Russian National Etalon time scale. The equation to correct the GLONASS satellite time is given by this equation [36]:

$$t_{UTC(SU)} + \Delta = t + \tau_c + \tau_n(t_b) - \gamma_n(t_b)(t - t_b) \tag{5.9}$$

where t is the GLONASS satellite time, τ_c and $\tau_n(\tau_b)$ are the correction factors to $t_{UTC(SU)}$, γ_n is the relativistic correction, and Δ is the constant 3-hour time offset between Russian National Etalon time and UTC. The code arrival time is computed using the code tracking loop, and is represented in receiver time. The code transmit time is provided in the GLONASS navigation message in GLONASS satellite time.

To compute the pseudorange measurement from the receiver to the satellite, the transmit time needs to be corrected by equation 16. After the conversion of the message arrival time and the transmit time into the same reference time system, the pseudorange of the satellite can be computed by:

$$PR = c \times (t_{arrival} - t_{transmit}(UTC(SU))) \tag{5.10}$$

where c is the speed of light, $t_{arrival}$ is the code arrival time in receiver time, and $t_{transmit}$ the code transmit time in UTC(SU) time.

Chapter 6

Design And Implementation of The Real Time Software Receiver

6.1 Implementation Of The Real Time Software Receiver

The center part of the software receiver is the computer system with an Intel 1.8GHz CPU and 2G memory. The operating system running on the computer is the Ubuntu Linux, which is widely used and well maintained. GNUradio and QT 4.0 are necessary to be installed. GNUradio is an open source software development toolkit that provides the signal processing runtime, which also provides the driver for the USRP board. The interface of the software receiver is developed by using libraries of QT 4.0. The real time software receiver is developed in C++. Object oriented method is adopted in the design of the software receiver to make the program easy to maintain and update. There are three major classes defined in the software receiver: CChannel, CGPS and CGLONASS. CChannel class contains the member functions for the bit-wise parallel correlation, frequency lock loops, phase

lock loops and delay lock loops with member variables which specify status of the current channel (e.g., RPN number, code phase, carrier phase, Doppler shift frequency, code index, etc.). The CGPS class provides the functions to process GPS messages. The CGLONASS class is responsible to process GLONASS messages. The implementation of the real time software receiver is shown in the Figure 6.1. Figure 6.2 shows more details about the correlator.

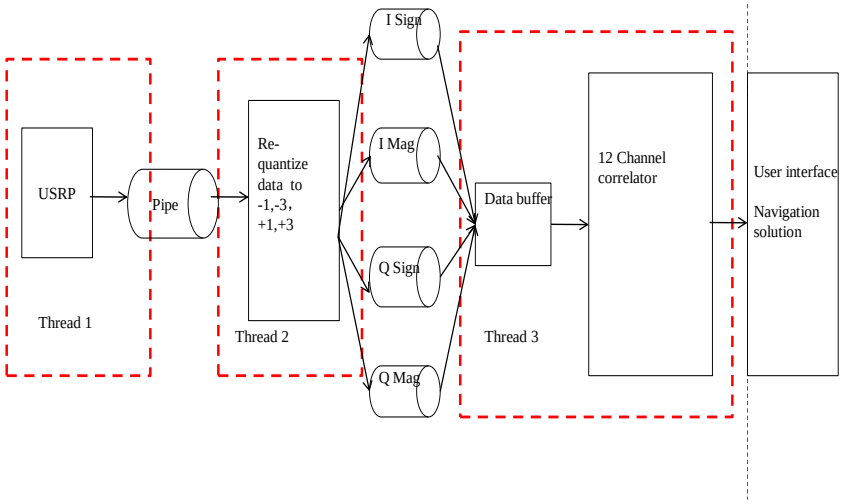

Figure 6.1: Implementation of Real-time Software Receiver

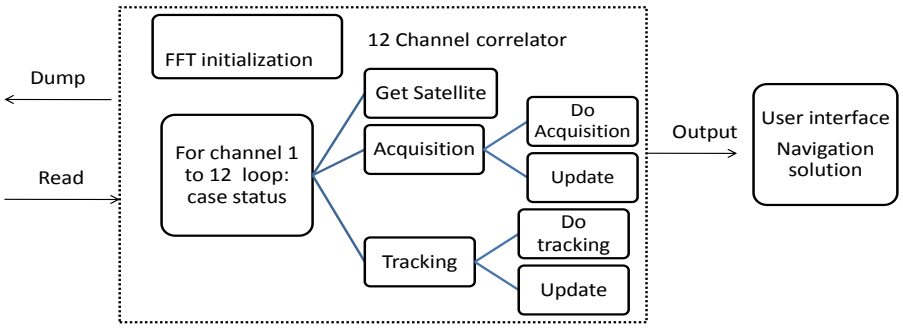

Figure 6.2: Detail of the Correlator

The software receiver program is a multiple thread program which contains three threads. Thread one is responsible for continuously reading data from the RF front end (USRP). The RF data are saved into a pipe, which is a data manage technique provided by Linux. Thread two is responsible for re-quantization of the incoming RF data which is 16-bits in length. As the input to the bit-wise correlator needs to be sign and bit integers, the RF data are re-quantized into two bits. The magnitude bit is set to be one for the sample with absolute value larger than the standard deviation. Otherwise,the magnitude is set to be zero. The sign bits and magnitude bits of in-phase and quadrature signals are sent to four pipes separately. After these preparations, The incoming data are fed to the real time correlator. Thread three is the main thread of the program. It loops through all 12 channels to compute the correlation of the incoming data between the local PRN codes and local carriers. The local PRN codes and carriers are read from the pre-generated PRN code tables and carrier tables. As the correlation data index for different channel are different from each other, a

circle data buffer of 500ms length of data is created. All channels share this data buffer. The indices of the RF data in the data buffer, for different channels, are recorded and updated after each correlation.

A more detailed picture about the correlator is given below: in Figure 6.3. The real time software receiver is not only tracks the current satellites in view but also keeps searching new satellites coming in. A background acquisition algorithm has been implemented in the software receiver to acquire new satellites coming in. The background acquisition is realized by serial search through all possible code shifts and Doppler frequency offsets. There are three possible states for each channel: Getsatellite, Acquisition, and Tracking. Getsatellite means the current channel is finding a satellite which is not in use by other channels in all possible satellites array. Acquisition means the current channel is searching for the existence of a satellite. If the correlation result between the incoming data and the local replica is larger than the threshold, this channel will start the tracking loop. The computation of the threshold is reference to [29]. Tracking states means this channel is tracking a satellite. The states of all channels are initialized by a FFT acquisition. In the tracking loop, a frequency lock loop is used before the more accurate phase lock loop.

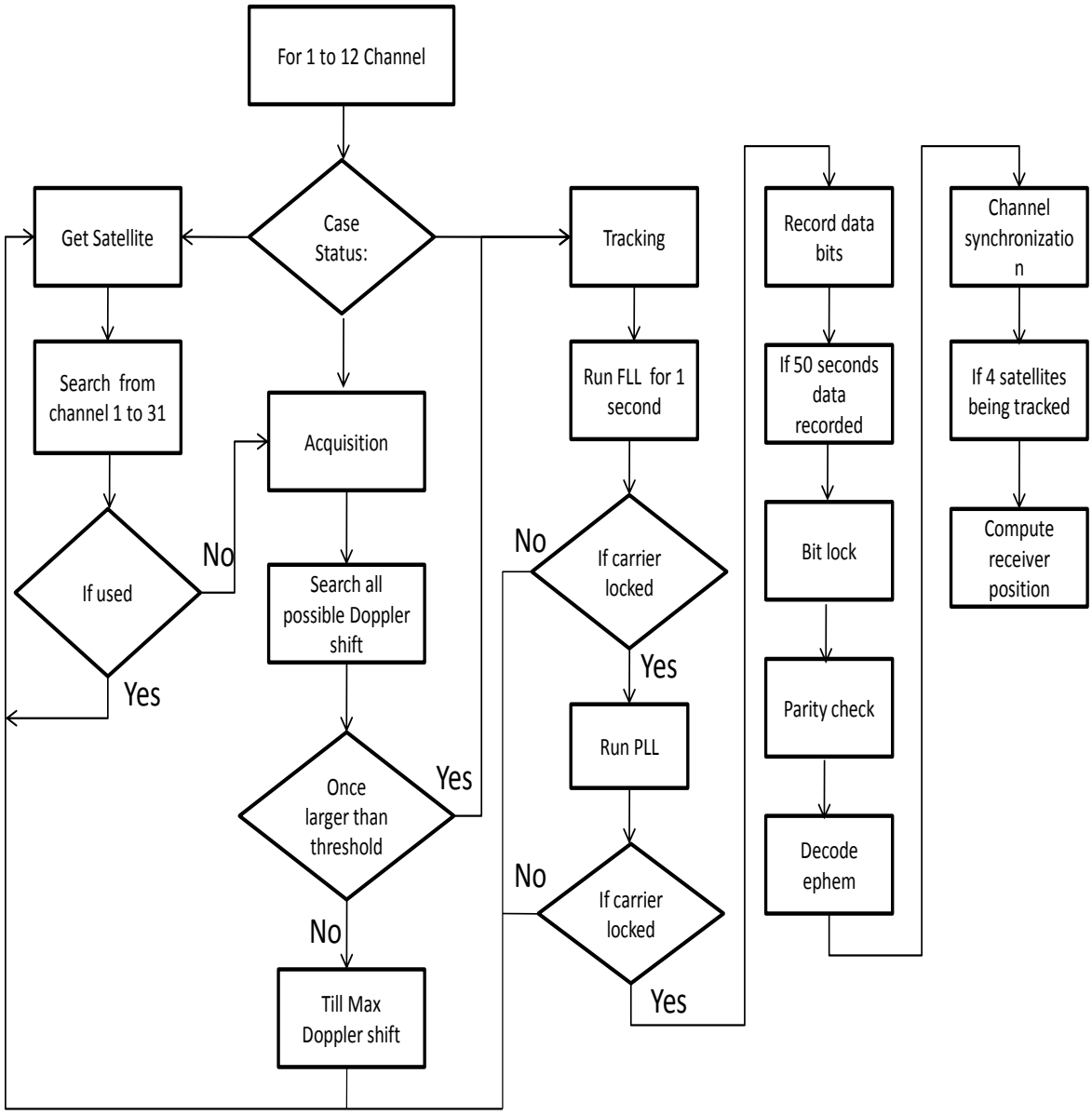

Figure 6.3: Diagram of the Software Receiver

Figure 6.3 shows in detail how the real time software receiver works. The total CPU usage of the software receiver is about 30 $\sim$ 40%of the Intel 1.8GHz CPU. The Linux system and the RF data re-quantization thread use about 10 percent of the total CPU. The 12 real time correlators use about 30% percent of the total CPU. The memory requirement

is also important in software receiver design. The carrier replica table is generated with 100Hz frequency interval and maximum offset of 5000Hz. The total memory usage for the code and carrier replicas is about 350K.

The acquisition, tracking results and interface of the software receiver are gaven below. The algorithms of the software receiver are tested by Matlab first, then implemented by C. Table 6.1 lists the setting of the RF data. As the bandwidth of the GLONASS signal is about 12MHz, the USRP is not a perfect RF front end to process GLONASS signals. Taking account that the current GLONASS system is not making use of its full bandwidth, the sample frequency used in the software receiver is enough to process the available GLONASS signals. The acquisition results of the software receiver for GPS signals is verified by two methods: one method is to compare with the results of the Cascade hardware receiver installed in the GPS lab; another method is to analyze the acquisition results. For GLONASS signals, the official website is helpful to get the available satellites. In order to get better acquisition results, two milliseconds of the data is used the in FFT based acquisition. The setting of the RF data is concluded in table 6.1. The interface of the hardware receiver in the GPS lab is gave in Figure 6.4. From the hardware receiver, the channel information and Doppler frequency are provided. These information can be used to compare with the acquisition results. Figure 6.5 shows the acquisition results of GLANASS channel 2. There is a clear peak in the outputs which means the local generated code lines up with the incoming signal. In order to further verify the acquisition results, the peak is plotted in the code phase as shown in Figure 6.6. As the auto-correlation property of the PRN code, the peak should look like a triangle and the length of the button is about 2 chips. The plot verifies that the GLONASS signal is acquired. Figure 6.7 and 6.8 shows the acquisition results of the GPS L1 signals.

Table 6.2 lists the setting of the FLL and PLL used in the software receiver. Figure 6.9 and Figure 6.10 plot the tracking results of the GLONASS and GPS signals. The integration interval used in the FLL and PLL and both 1ms. FLL is used to lock the incoming signals first, then the PLL is used to refine the frequency and phase. After the signals been tracking, the software receiver will start to decode the navigation messages. As BPSK signals are used in both GPS and GLONASS signals. We can see the energy of the output is mainly contained in the In-phase beam. The Doppler shift frequency changed slowly and linearly after been tracked. The phase transition can be clearly viewed in the In-phase output.

Figure 6.11 shows the position solution error of the software receiver in east, north and up separately. The position of the antenna is known as follow:

1 WGS 84: [37.23107 -80.424233 630],

2 ECEF: [845901.844456496 -5014168.241999357 3838223.446543617].

The software receiver can provide position accuracy of 15 meters in most cases. Figure 6.12 shows the velocity errors. The amplitude of the speed errors is in centimeters. At current situation, it quite difficult to acquire 4 GLONASS satellites in the same time. So a two dimensional solution is computed and the solution residual error time history is given in Figure 6.13. A sample screen-shot from the real time software receiver is provided in Figure 6.14. This figure shows the interface of the software receiver with seven GPS satellites being tracked and provides a position offset about 17 meters in altitude. Further work still need to be done to improve the performance of the current software receiver. The integration time in the software correlator is 1ms which is too short for weak signals. As a result, only satellites with high elevation can be tracked by the current software receiver. The total number of satellites in tracking is always two or three less than the hardware

receiver. Several major modifications of the bit-wise parallel corellator need to be done to increase the integration interval. This is a part of the future work in the real-time software receiver.

Table 6.1: RF data setting in the software receiver

Description	GPS	GLONASS
Sample frequency	$4MHz$	$8MHz$
Intermediate frequency	$1MHz$	$1MHz$
Sample length	2 bits	2 bits
Acquisition interval	$2ms$	$2ms$

Table 6.2: FLL and PLL setting in the software receiver

Description	FLL	PLL
Integration interval	$1ms$	$1ms$
Noise bandwitdh	$10Hz$	$15Hz$
Filter Order	second	second
discriminator	actan	$[I_{k-1}, Q_{k-1}] \times [I_k, Q_k]$

```
 Lat    37.23114  Spd    0.3  SVs 10      CTrack FLL    Date 10/01/10
 Lon   -80.42431  Hdg   39.3  Nav 3D      GDOP   1.9    UTC  01:15:20
 Alt    617.0424  ROC    0.3  NS CHAN     DO  -216.1    OscErr   0.14

 CH   SV ELV AZI   DOPP     NCO UERE SF    PRerr PRRerr LOCK C/N0   iS4
  1   22  83  41    180     -33    2  5      1.5   -0.0 CCBF 46.1 +0.02
  2   26  62  15  -1016   -1229    2  5      5.1   -0.1 CCBF 44.5 +0.02
  3   27  16  42  -3149   -3360    4  5      4.6   -0.3 CCBF 34.5 -1.00
  4   30  25 127   2820    2608    2  5      5.8   -0.6 CCBF 37.8 +0.04
  5   31  22 212   3437    3220    2  5     -1.9    0.2 CCBF 37.0 +0.05
  6   12  25  90   1219    1004    2  5     -2.0    0.3 CCBF 38.4 +0.06
  7   18  49 117  -1826   -2039    2  5      1.8    0.1 CCBF 43.0 +0.04
  8   14  60 308    691     478    2  5      4.0   -0.1 CCBF 42.6 +0.02
  9    9  23  42  -3469   -3685    2  5      1.0    0.5 CCBF 35.9 +0.06
 10   21  12 175  -3736   -3956    2  3      6.5    0.6 CCBF 33.5 +0.12
W11   33   0   0      0    3783    0  0      0.0    0.0      30.0 +0.00
N12  ---  --  --     --  -10000   --  -       --     --       0.0   --

>                                                  Cascade 1.62 (070304)
```

Figure 6.4: Interface of The Cascade hard receiver in GPS lab

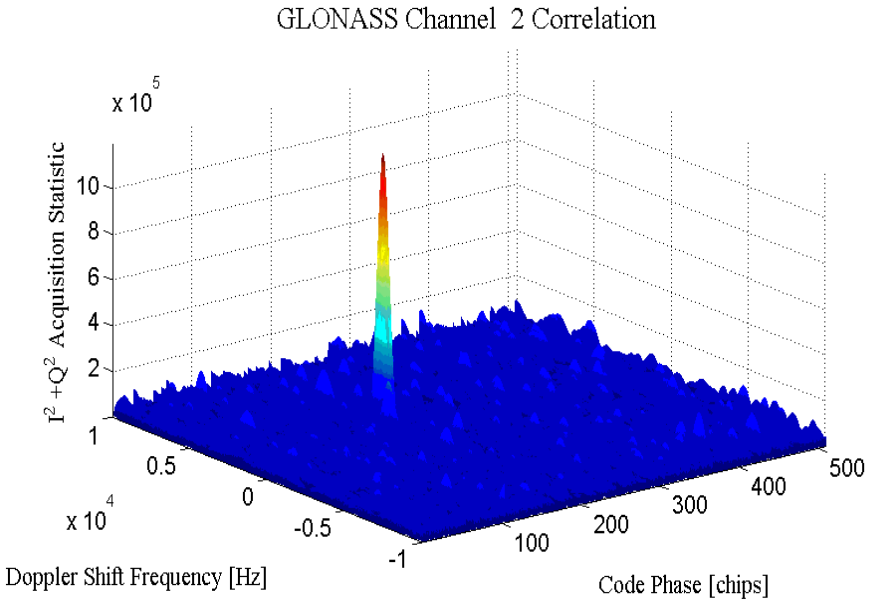

Figure 6.5: GLONASS signal acquisition results

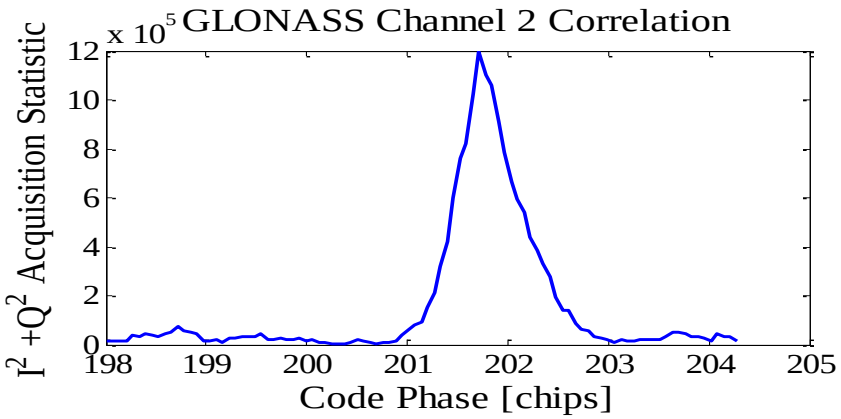

Figure 6.6: View of acquisition results in code phase

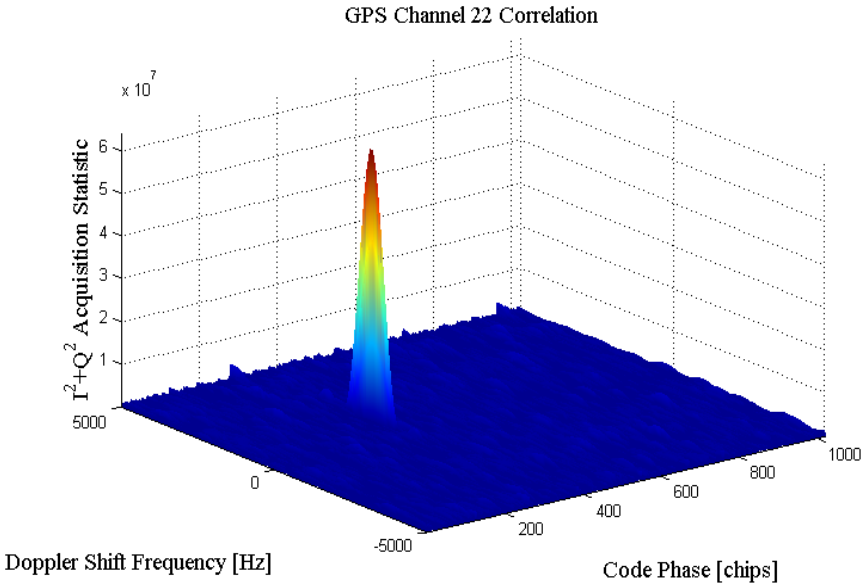

Figure 6.7: GPS signal acquisition results

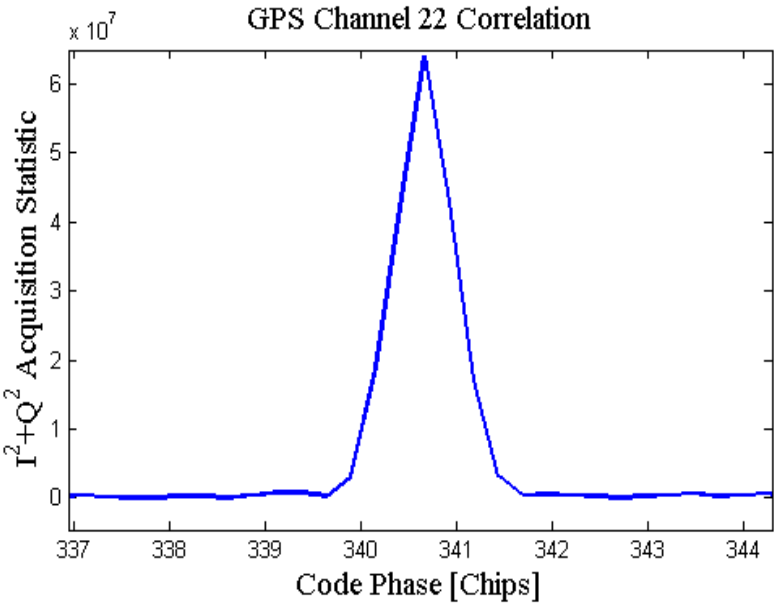

Figure 6.8: View of acquisition results in code phase

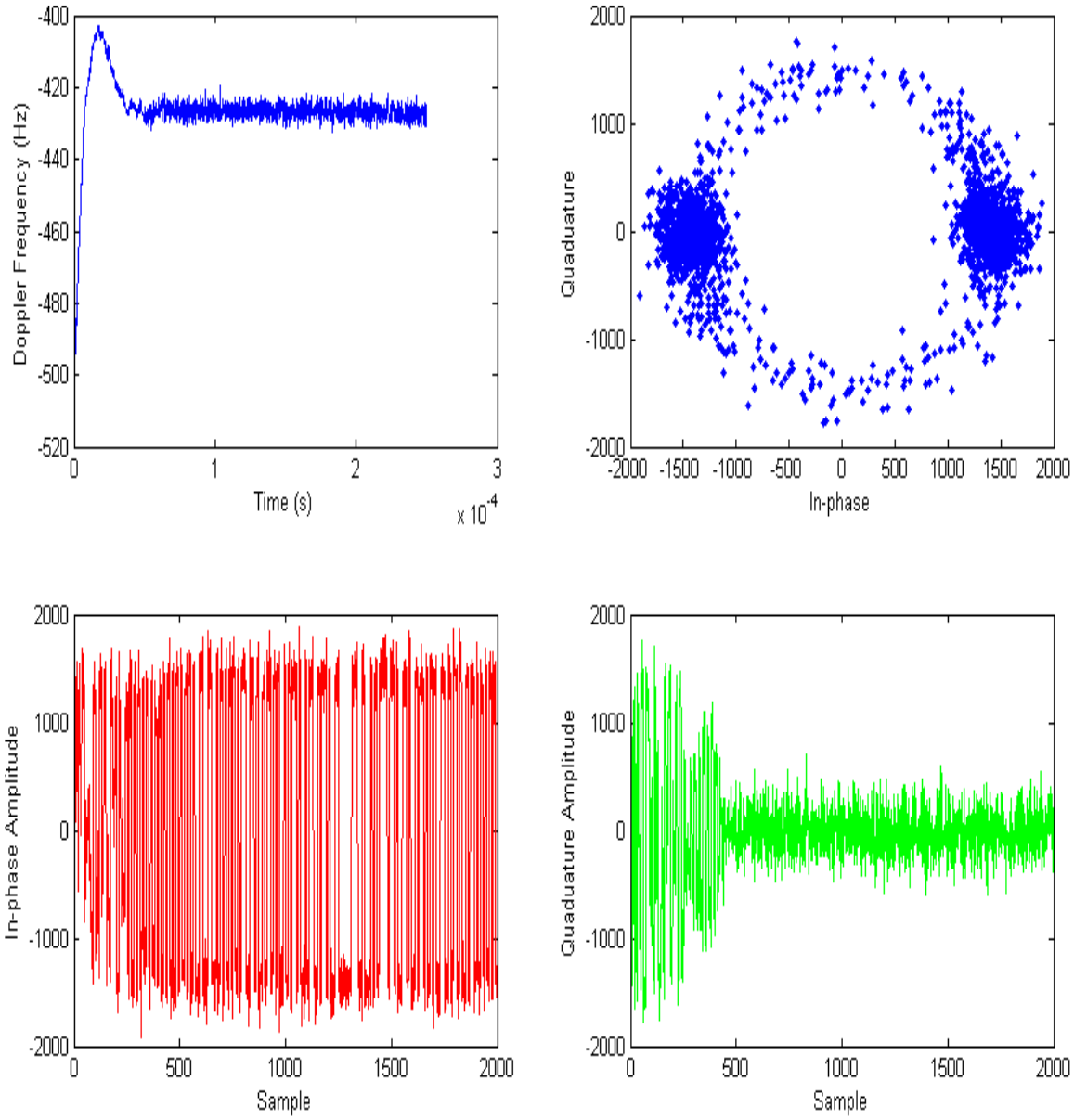

Figure 6.9: GLONASS Channel 2 Tracking Results

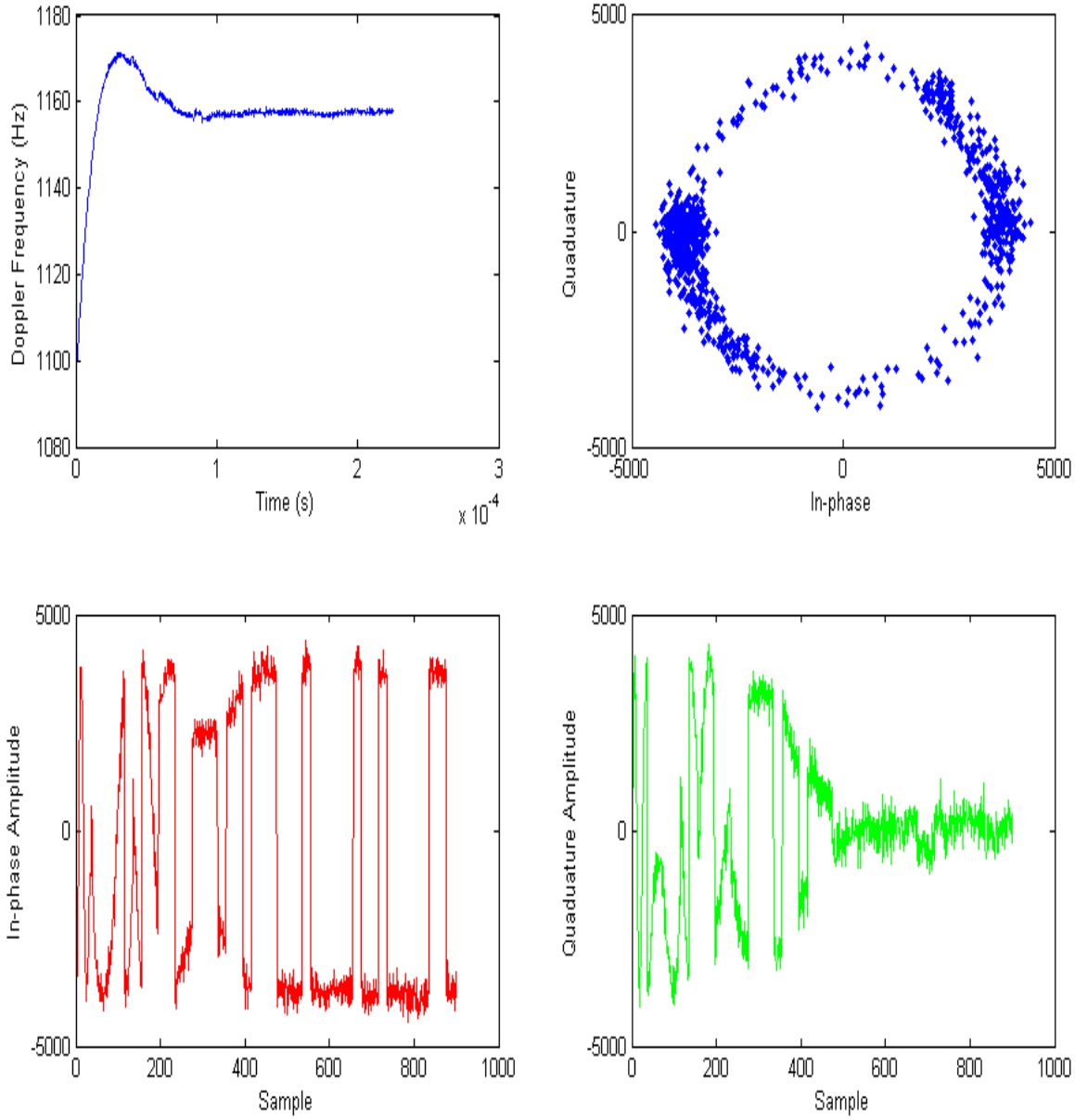

Figure 6.10: GPS Channel 22 Tracking Results

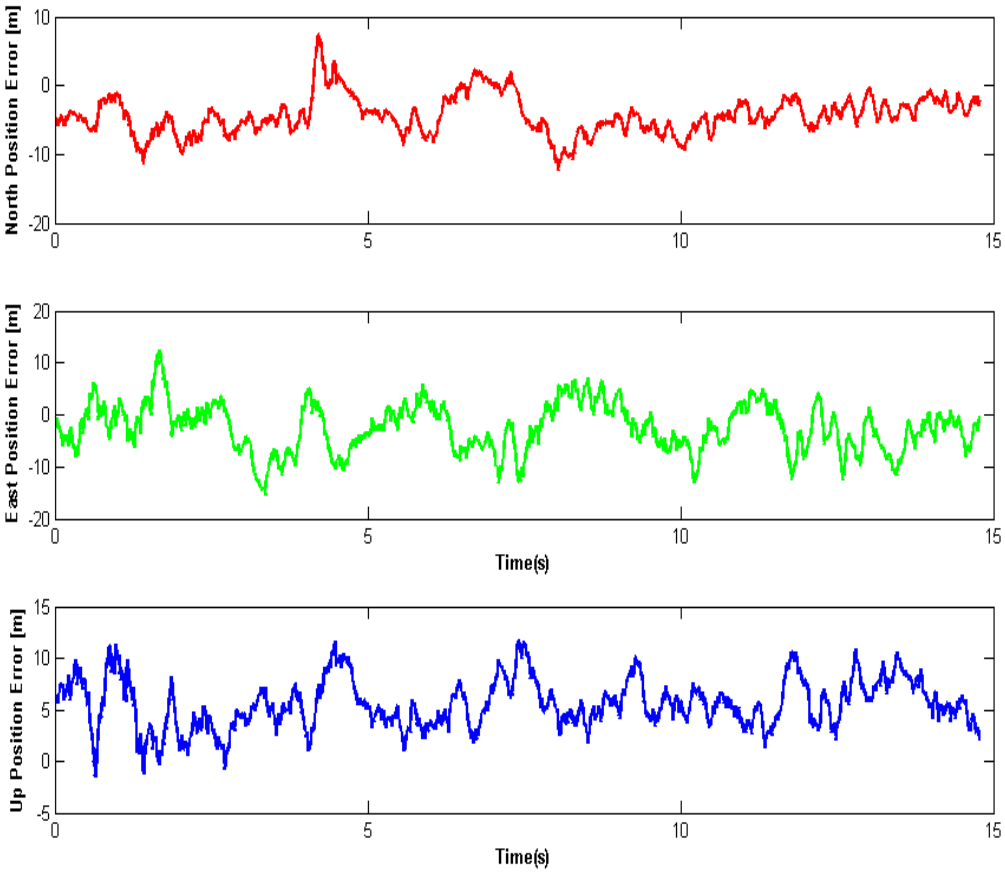

Figure 6.11: GPS Position Solution Error

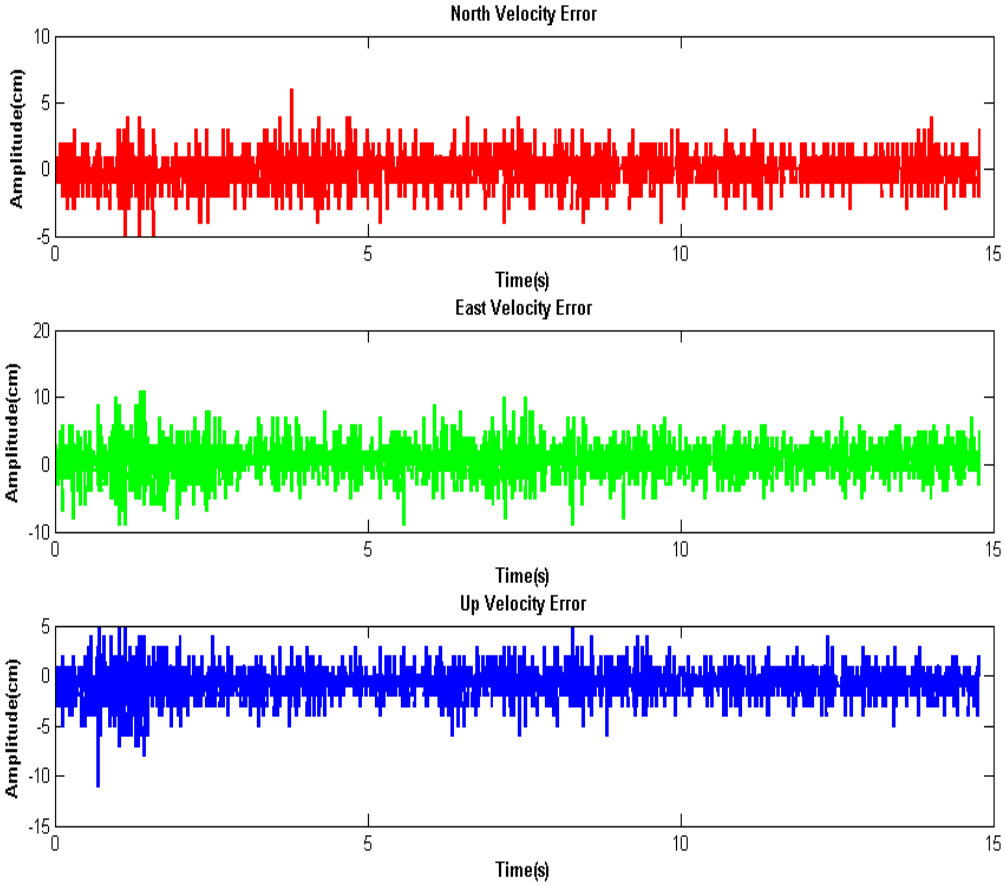

Figure 6.12: GPS Velocity Solution Error

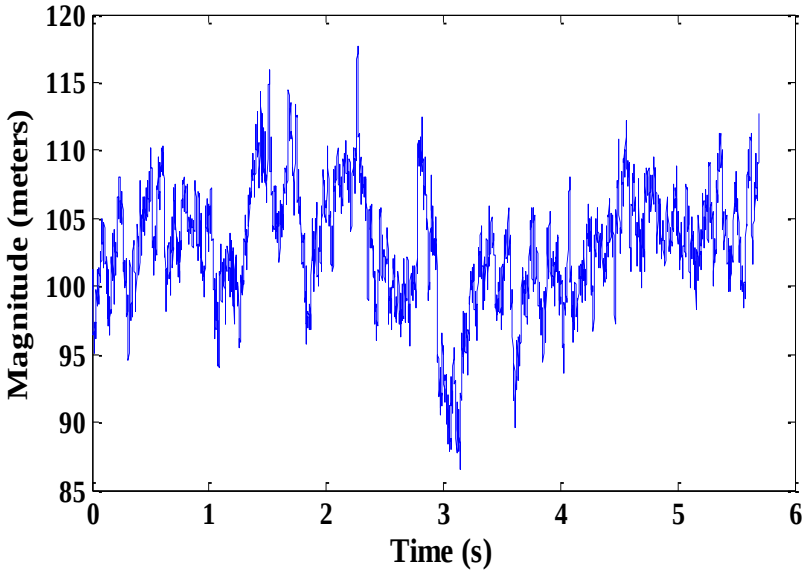

Figure 6.13: 2-dimensional residual navigation errors output from GLONASS software receiver

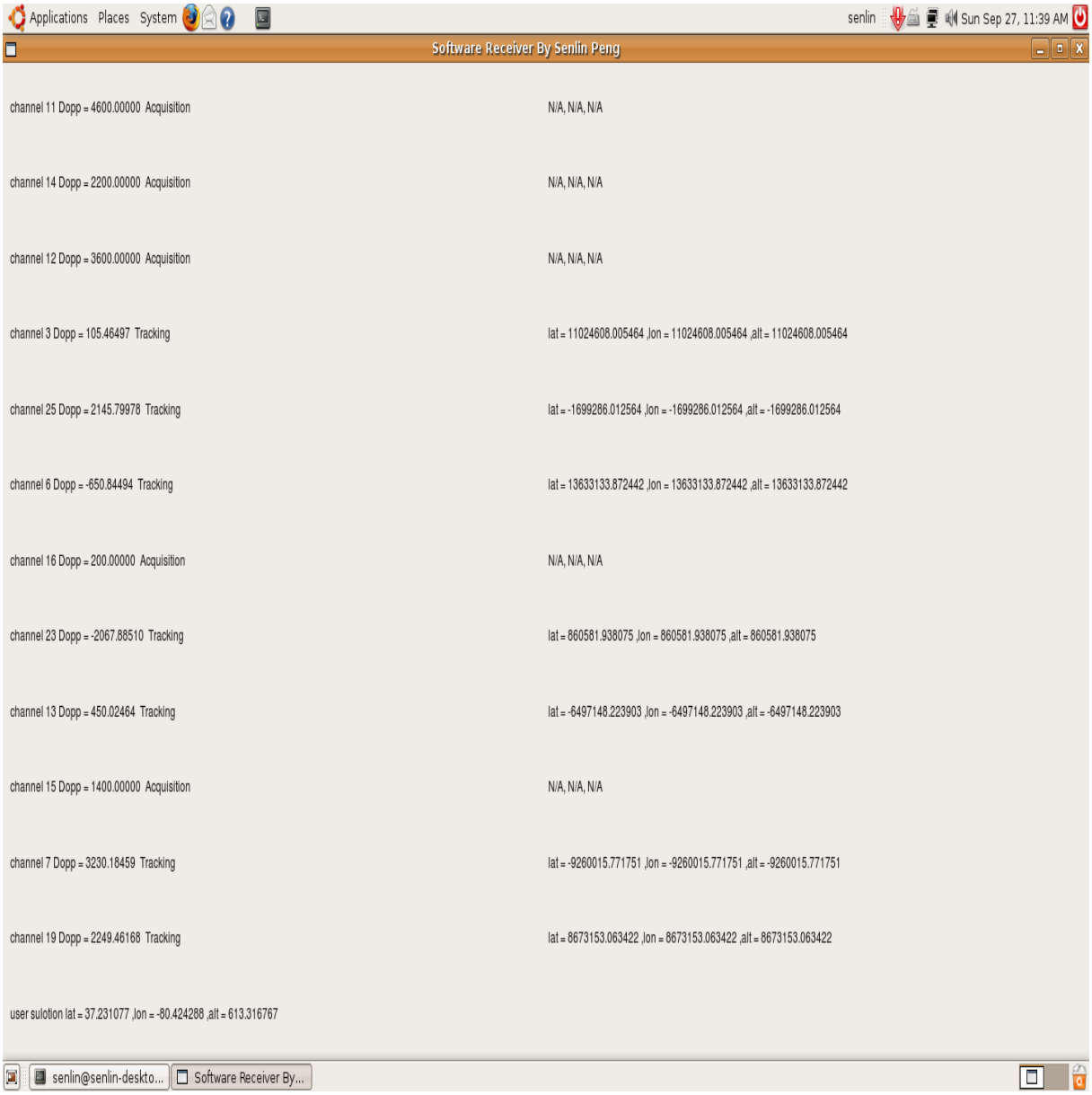

Figure 6.14: Interface of software receiver

Chapter 7

Summary And Future Works

A 12 channel real time GNSS software receiver which utilize USRP as the RF front end has been developed on a desktop with Intel 1.8GHz CPU. The software receiver has the capability to process both GPS and GLONASS L1 signals, and provide a user solution accuracy of 30 meters. The receiver can track up to 20 channels at 2-bit sampling rate of 4MHz complex samples for GPS L1 signals. Bit-wise parallel correlation is used in the design of the software correlator. Object oriented approach is used to make the software receiver flexible, readable, and maintainable.

Challenges: the sample frequency of the USRP board is not enough to process the whole bandwidth of GLONASS L1 signals. From the tracking results of the software receiver, it is hard to find a time when four GLONASS satellites being tracked at the same time. The accumulation interval of the software correlator is 1ms with 2-bit sinusoid samples. The accumulation interval needs to be increased in order to track satellites with low elevation. The increase of the correlation length will cause the modification of the current

bit-wise parallel correlation algorithm. The reason is that the post-rotation in the bit-wise parallel correlation only works for small angle offset.

Prospect: The background search of satellites is implemented by serial search through all possible code phases and frequency offsets which slow down the performance of the real time receiver. A more efficient FFT based acquisition method is going to be designed for future applications. Anther improvement is to increase the accumulation interval to improve the tracking performance of the software receiver. The USRP board is a multiple input multiple output (MIMO) system, so a dual frequency receiver is possibly be implemented.

Bibliography

[1] Ledvina, B.M., Psiaki, M.L., Powell, S.P., and Kintner, P.M., "Bit-Wise Parallel Algorithms for Efficient Software Correlation Applied to a GPS Software Receiver," IEEE Transactions on Wireless Communications, Vol. 3, No. 5, Sept. 2004, pp. 1469-1473.

[2] Ledvina, B.M., Psiaki, M.L., Humphreys, T.E., Powell, S.P., and Kintner, P.M., "Real-Time Software Receiver for the GPS and Galileo L1 Signals," Proc. ION GNSS 2006, Sept. 26-29, 2006, Fort Worth, TX.

[3] Pratap Misra and Per Eng "Global Positioning System: Signals, Measurements, and Performance," Ganga-Jamuna Press, Second Edition, 2006.

[4] Kaplan, E., Understanding GPS: Principles and Applications,Norwood, MA, Artech House, 1990.

[5] Anonymous, "GLONASS Interface Control Document Version 5.0," Moscow, 2002. http://www.glonass-ianc.rsa.ru/.

[6] Kai, Borre., Dennis, M. Akos.,and Nicolaj Bertelsen., A Software- Defined GPS and Galileo Receiver: A Single- Frequency Approach ISBN: 0817643907 Publisher: Birkhauser 2005.

[7] Akos, D., A Software Radio Approach to Global Navigation Satellite System Receiver Design, Ph. D. Dissertation, Ohio University, 1997.

[8] Ledvina, B.M., M. Psiaki, T.E. Humphreys, S.P. Powel, and P.N Kintner (2005) "A Real- Time Software Receiver Tracking of GPS L2 Civilian Signals Using a Hardware Simulator, in the Proceeding of Institute of Navigation GPS 2005, 13-16 September, Long beach CA, pp.1598-1610.

[9] Humphreys, T.D., M. L. Psiaki, P.M. Kintner, Jr., and B.M. Ledvina (2006) "GNSS Receiver Implementation on a DSP: Status, Challenges, and Prospects, in the Proceedings of Institute of Navigation GNSS, 26-29 September, Fort Worth TX, pp.2370- 2382.

[10] Akos, D.M., P.-L. Normark, P. Enge, A. Hansson, and A. Rosenlind., "Real-Time GPS Software Radio Receiver," Proc. of the Institute of Navigation National Technical Meeting, Long Beach, CA, January 22-24, 2001, pp. 809-816.

[11] Senlin, Peng, Brent, Ledvina, A Real-Time Software Receiver for the GLONASS L1 Signal, Proc. ION GNSS 2008, Sept. 22-25, 2008, Savannah, GA.

[12] Mike, Stewart., and Maria, Tsakiri., "GLONASS Broadcast Orbit Computation," GPS Solutions, Vol. 2, No. 2, October, 1998, pp. 16-27.

[13] Lauri, Wirola., and Jari, Syrjarinne., "GLONASS Orbits in GPS/Galileo-style Ephemeredes for Assisted GNSS," Proc. ION NTM 2008, Jan. 28-30, 2008, San Diego, California.

[14] Yuri A. Bazlov., Viktor F. Galazin., Boris L. Kaplan., Valery G. Maksimov., and Vladmir P. Robot., "Propagating PZ 90 to WGS 84 Transformation Parameters," GPS Solutions, Vol. 3, No. 1, July, 1999, pp. 13-16.

[15] Anonymous, Novatel, http://novatel.com/Documents/Papers/GPS701-702GG.pdf/.

[16] Anonymous, "USRP Motherboard Data sheet," Ettus Research LLC£http://www.ettus.com/Download.html.

[17] Anonymous, "Agilent N5181A MXG Analog Signal Generator Data Sheet," Agilent Technologies, http://cp.Literature.agilent.com/litweb/pdf/5989-5311EN.pdf.

[18] Van Dierendonck, A.J., "GPS Receivers," in Global Positioning System: Theory and Applications, Vol.I, Parkinson, B.W. and Spilker, J.J. Jr., eds., American Institute of Aeronautics and Astronautics, Washington, 1996, pp. 329-407.

[19] Bernhard, Hofman-Wellenhof., James, Collins., and Herbert Lichtenegger, "Global Positioning System (GPS): Theory and Practice," Springer-Verlag New York, LLC, 5th Edition, 2001.

[20] Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T., Estimation with Applications to Tracking and Navigation, J. Wiley and Sons, (New York, 2001), pp. 200-232.

[21] Bao, J. and Y. Tsui (2000b) Fundamentals of Global Positioning System Receivers: a Software Approach, John Wiley & Sons, Inc., pp. 133.

[22] Bong-Young, Chung., Charles, Chien., Henry, Samueli., and Rajeev, Jain., "Performance Analysis of an All-Digital BPSK Direct- Sequence Spread-Spectrum IF Re-

ceiver Architecture," IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 11, NO. 7, SEPTEMBER 1993,pp. 1096-1107.

[23] Lindsey, W. C. and Chie, C. M., "A Survey of Digital Phase- Locked Loops, Proc. IEEE, Vol. 69, No. 4, 1981, pp. 410C431.

[24] Best, Roland E., Phase Locked Loops: Design,Simulation, and Applications, 3rd Ed., McGraw Hill, ISBN: 0070060517,1997.

[25] Spilker, J.J.Jr., "Dlay-Lock Tacking of Binary Signals," IEEE Transactions on Space Electronics and Telemetry, Vol. SET-9, March 1963, pp. 1-8.

[26] Oppenheim, Alan V. and Ronald W. Schafer, Discrete-Time Signal Processing, Prentice Hall,ISBN: 0137549202,2rd Ed., 1999.

[27] Proakis, J. G., Digital Communications, McGraw-Hill, 3rd ed., 2001.

[28] Psiaki, M.L.,"Block Acquisition of Weak GPS Signals in a Software Receiver," Proc. ION GPS 2001, Salt Lake City, Utah, Sept. 11-14, 2001, pp. 2838-2850.

[29] Zarlink, (2001). GP2021 Data Sheet, http://www.zarlink.com

[30] R. Jaffe and E. Rechtin, "Design and Performance of Phase-Locked Circuits Capable of Near-Optimum Performance Over a Wide Range of Input Signals (and Noise) Levels", Information Theory, IRE Transactions on Volume 1, Issue 1, March 1955, pp.66 - 76.

[31] Francis, D. Natali., "AFC Tracking Algorithm," IEEE Transactions on Communications, Vol. Com-32, No. 8, August 1984, pp. 935-947.

[32] Walter Gautschi., Numerical Analysis,Cambridge, ISBN:0817638954, 1997.

[33] Chi-Tsong, Chen., Linear System Theory and Design, Oxford University Press, ISBN: 0195115953, 2000.

[34] Floyd, M. Gardner., PHASELOCK TECHNIQUES, Wiley-Interscience, ISBN: 0471042943, 2005.

[35] Michael, L. Workman., Gene F. Franklin., J. David Powell., Digital Control of Dynamic Systems,3rd Ed, Prentice Hall, ISBN:0201820544, 1997.

[36] GPS Joint Program Office, 1997. ICD-GPS-200: GPS Interface Control Document. ARINC Research. Available on line from the United States Coast Guard Navigation Center.